Machine Learning Meets Analog Circuit Design: Intelligent Automation of IC Design

by

Morteza Fayazi

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical and Computer Engineering)
in The University of Michigan
2024

Doctoral Committee:

Professor Ehsan Afshari, Co-chair
Associate Professor Ronald G. Dreslinski, Co-Chair
Professor Michael Flynn
Professor Chad Jenkins
Professor David D. Wentzloff

Morteza Fayazi

fayazi@umich.edu

ORCID iD: 0000-0002-5598-281X

© Morteza Fayazi 2024

To who sees but is not seen

ACKNOWLEDGEMENTS

Throughout my time pursuing my graduate studies at the University of Michigan, I have had the privilege of interacting with many talented and skilled individuals across various disciplines. Foremost, I am deeply grateful to my Ph.D. advisors, Prof. Ronald Dreslinski and Prof. Ehsan Afshari, for all their support and guidance. They were not just mentors, but also brothers and friends to me. They both were very patient and helpful during our discussions through my Ph.D. journey to the end. They gave me the freedom to explore my ideas and various research topics. Beyond offering insightful feedback, they imparted invaluable lessons that extended into my personal life. I feel fortunate to be part of their research groups.

I would like to thank my committee members, Prof. Micheal Flynn, Prof. Chad Jenkins, and Prof. David Wentzloff, for their help and useful discussions. Special thanks to my brother Mohammadhasan Fayazi for all his help, time, and ideas. He is a real role model for me in the academic world. Also, I would like to thank Prof. David Blaauw, Prof. Dennis Sylvester, Prof. Hun-Seok Kim, Prof. Trevor Mudge, and Prof. Benton Calhoun whom I had the honor to collaborate with. Special thanks to Dr. Mohammad Noshad who gave me the opportunity to work at Shyld AI as an intern.

It was a great honor to collaborate with both current and former members of our research groups, CADRE and UNIC, during my time at the University of Michigan: Zach Colter, Dr. Tutu Ajayi, Dr. Sung Kim, Amirata Tabatabavakili, and Dr. Morteza Tavakoli Taba. Special thanks to Dr. Yaswanth Kumar Cherivirala, Dr. Sumanth Kamineni, Dr. Kyumin Kwon, Dr. Mehdi Saligane, and Dr. Kuan-

Yu Chen for their collaborations and useful discussions. Additionally, I would like to extend my gratitude to my good friends and roommates, Dr. Vahid Jamali, Farshad Rahimi, Dr. S. M. Reza Soroushmehr, Dr. Javad Bagherzadeh, Prof. S. M. Hossein Naghavi, Ahmad Almohri, Dr. Ali Sadeghi, Prof. Austin Rovinski, Ali Ghazizadeh, Mohammad Khodabaksh, Dr. Morteza Sheikhsofla, Dr. Morteza Noshad, Dr. Vahnood Pourahmad, S. Mehdi Banihashemi, Moeen Radgolchin, Hamad Alotaibi, Aditya Varma, Dr. Milad Mousavifar, Dr. Omid Bahrami, and Dr. Navid Barani, for all their support. I have thoroughly enjoyed spending time with them. Moreover, I would like to thank all the staff at the University of Michigan for their kindness and assistance, especially Kristen Thornton and Joel VanLaven, for their patience in addressing my inquiries.

Above all, I would also like to express my heartfelt gratitude to my family for their unwavering support throughout my entire life. The unconditional love of my parents and their boundless sacrifices have made all the hardships of life seem easy for me. Despite the physical distance between us, they remain firmly entrenched in my heart. Their prayers are the most potent remedy in overcoming life's challenges. During my time in the US, I have had the best moments spending time with my wife. Eating together, watching TV together, and taking our short walks have always been the highlights of my day. Her kindness and support during difficult times have dispelled any feelings of loneliness. I am also deeply thankful to my parents-in-law for their blessings and kindness; they hold the same place in my heart as my own parents.

Finally, I would like to thank the only true source of love and guidance. He has blessed me abundantly by surrounding me with the finest people, providing me with what is truly the best for me, and illuminating the path for me to know and love Him.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

LIST OF FIGURES viii

LIST OF TABLES xiv

ABSTRACT xvi

CHAPTER

I. Introduction 1

1.1 Analog and Mixed-Signal Circuit optimization 3
1.1.1 Overview and Main Concepts 3
1.1.2 Bayesian-Based Approaches 4
1.1.3 Multiple Starting Point Approach 8
1.1.4 Neural Network Based Approaches 9
1.2 Analog and Mixed-Signal Circuit Performance Modeling . . . 11
1.2.1 Overview and Main Concepts 11
1.2.2 Bayesian Model Fusion Based Approaches 11
1.2.3 Semi Supervised Learning Approaches 13
1.3 Dissertation Thesis 14

II. FASCINET: A Fully Automated Single-Board Computer Generator Using Neural Networks 16

2.1 Introduction 17
2.2 Related Work 20
2.3 Platform Architecture Overview 22
2.4 Datasheet Scrubber 24
2.4.1 Category Recognizer 24
2.4.2 Table Extractor 28

2.4.3 Pin Functionality Extractor 30
2.5 COTS Database 31
2.6 Peripheral Circuit Generator 32
2.7 Evaluation 38
2.7.1 Datasheet Scrubber 38
2.7.2 COTS DB 40
2.7.3 Peripheral Circuit Generator 40
2.8 Conclusion 48

III. Tablext: A Combined Neural Network And Heuristic Based Table Extractor 49

3.1 Related Work 52
3.2 Table Location Identification 54
3.3 Table Extraction 58
3.3.1 Real Line Identification 60
3.3.2 Inferred Line Identification 61
3.3.3 Final Structural Line Identification 64
3.3.4 Neural Network Correction 65
3.3.5 OCR 69
3.4 Evaluation 69
3.4.1 Table identification results 70
3.4.2 Table extraction result comparison 71
3.5 Conclusion 74

IV. AnGeL: Fully-Automated Analog Circuit Generator Using a Neural Network Assisted Semi-supervised Learning Approach 75

4.1 Introduction 76
4.2 Background and Related Work 80
4.2.1 Problem Formulation 80
4.2.2 Reducing Database Size 80
4.2.3 Circuit Design Framework 81
4.3 Proposed Database Generation 82
4.3.1 Overview 82
4.3.2 Topology Combiner Model 84
4.4 Proposed AnGeL Framework 87
4.4.1 Overview 87
4.4.2 Overall Topology Decider 90
4.4.3 Sub-circuit Topology-Selector 94
4.4.4 Sub-circuit Sizing 94
4.4.5 Constraints Transformation 96
4.5 Evaluation 99
4.5.1 OPAMPs 100
4.5.2 Filters 109

4.6 Conclusion 113

V. FuNToM: Functional Modeling of RF Circuits Using a Neural Network Assisted Two-Port Analysis Method 114

5.1 Introduction 115
5.2 Background & Related Work 120
5.2.1 Analog and RF Circuits Functional Modeling 120
5.2.2 Two-port Analysis & S-parameters 122
5.3 Proposed Approach 123
5.3.1 Neural Network Models 123
5.3.2 Circuit Partitioning into E-networks 127
5.4 Evaluation 128
5.4.1 Phase Shifter 129
5.4.2 Two-stage LNA 132
5.5 Conclusion 136

VI. Conclusion & Future Work 137

BIBLIOGRAPHY 143

LIST OF FIGURES

Figure

1.1 The CL-BMF procedure [91]. After fitting the less complex model, f_1, using a small number of labeled samples collected by simulations, a set of pseudo samples are generated with almost no cost. The more complex model, f, is trained then with all the labeled and the pseudo samples. 12

1.2 The semi-supervised co-learning framework [100]. $\mathbf{\Omega}$ and $\mathbf{\Theta}$ denote training and unlabeled dataset respectively, while $\mathbf{y}_{U,x}$ and $\mathbf{y}_{U,z}$ represent pseudo labels for unlabeled samples $\mathbf{\Theta_x}$ and $\mathbf{\Theta_z}$ respectively. After building f and f_1 models from the training set, highly-confident pseudo samples are chosen from the unlabeled dataset. These highly-confident pseudo samples are used to update the training set of the other model at the next iteration. 15

2.1 High-level platform architecture. 22

2.2 The CNN model of the category recognizer. The input is the first 256 words of datasheets and output is one of the 51 categories. 24

2.3 The CNN model of the table position identification. The input is the datasheet page and output is the position of the table within it. . . 28

2.4 Finding invisible cell separator lines. (a) An example of table without cell separator lines. (b) The tool draws straight lines that do not intersect the data all over the table. (c) The final result after removing shortened lines and merging the adjacent ones. For being more clear the horizontal lines have not been shown. 29

2.5 Peripheral circuit generator flow using the HDNN model. 32

2.6 HDNN model: A custom made binary classifier in Keras [141] with 2 Fully Connected (FC) hidden layers. The hidden layers have 32 nodes each with a nonlinear RELU activation function, while the output layer has a nonlinear sigmoid activation function. The inputs with more details are described in Table 2.1. 34

2.7 Steps of determining the required peripheral components by the HDNN model: (a) The main components are inputted. (b) The peripheral components that are directly connected to the main components, *i.e.* first-major, are determined. (c) The peripheral components around the first-major components, *i.e.* second-major, are determined. The next step which is the complete SBC after determining all the required peripheral components has been shown in Fig. 2.13. An example of parallel capacitors that their number is determined by the DNN model is shown with a red box. 37
2.8 The confusion matrix of the category recognizer when the first 256 words are given as inputs. Red rectangles show that, generally, common categories (*e.g.* Comparator) with a larger training set have a higher accuracy in comparison to the niche ones (*e.g.* SoC). 39
2.9 Average prediction accuracy comparison of the category recognizer with different hyperparameters used for training. As it was mentioned in Section 2.4.1, there are 5 branches with different neuron numbers that at the end are flattened and concatenated. The base neurons numbers in the convolutional layer of these branches are {1, 2, 2, 2, 4} that the coefficient in the X-axis is multiplied to them. For instance, neuron number coefficient of 32 leads to a network which its branches have {32, 64, 64, 64, 128} neurons in their convolutional layer. The figure shows different activation functions applied to the convolutional layer. The coefficient of 256, *i.e.* {256, 512, 512, 512, 1024} neurons, with the RELU activation function is the simplest architecture that gives a high prediction accuracy of 95.3%. The first 256 words are given as the input. 41
2.10 The distribution of improvement rate of (a) price and (b) power of found components by the search engine over the human-designed ones. Improvement here means less for both power and price. 43
2.11 The distribution of identical connections over the compared designs. 44
2.12 Identical connections comparison with different hyperparameters used for training. The activation function for all hidden layers are RELU and only the activation function of the output is changed. As you can see, in general, sigmoid at the output layer gives better results. Two hidden layers with 32 neurons at each layer is the simplest architecture that gives a high identical connections of 87% as it is also shown in Fig. 2.6. 45
2.13 An example of a generated SBC schematic by FASCINET. Different colors show a comparison of the similarities between this FASCINET generated SBC and the human-designed one. 46
2.14 The manufactured PCB of the SBC in Fig. 2.13. A simple blinking program is uploaded and run successfully on the SBC. 47
3.1 (Table A) Each horizontal boxes represents a row where a table is identified. The bottom group is highlighted with a thick box (red) for clarification. 58

3.2 (Table A) Vertical boxes represent every column where a table is identified. 59
3.3 (Table A) A cutout of the original table using the coordinates identified. 59
3.4 (Table B) A table sample identified by the proposed table identification method. 60
3.5 (Table B) Inferred lines shown with thick bright (yellow) lines and real lines shown with thin dark (blue) lines. 61
3.6 (Table C) Initial sparse table before inferred vertical lines. A red box has been placed around the, difficult to handle, referenced sparse column for clarity. 62
3.7 (Table C) Sparse table with initialized inferred vertical lines. 62
3.8 (Table C) Sparse table with incremented inferred vertical lines. . . . 64
3.9 (Table B) The merged lines (thin green) created by the final line identification. Vertically split cells highlighted with thick boxes (red). 66
3.10 (Table B) The table after the CNN merged problematic cells. Rows kept are separated for accurate OCR. 66
3.11 (Table B) Tablext output of the table in Fig. 3.4, in CSV format. Thick boxes (red) show the horizontally merged cells. Cells with multiple rows are kept separated for readability. 67
3.12 (Table D) A text-based PDF page in our testbench with potentially problematic columns highlighted with boxes (red). 72
3.13 (Table D) Tablext's final output in CSV format. Boxes (red) are showing the correct handling of problematic cells. 73
3.14 (Table D) Tabula's final output in CSV format. Boxes (red) are showing the incorrect handling of problematic cells 74
4.1 (a) An example of three simpler topologies, T_1, T_2, T_3, that can be combined to determine the circuit behavior of T_{out}. To enhance visibility, we use identical colored boxes/circles to show bodies/loads that are the same in different topologies. (b) The output topology of Fig. 4.1(a) is used to make a more complicated topology. 83
4.2 The topology-combiner model. (a) The general structures of three simpler topologies, T_1, T_2, and T_3, which are combined to determine the circuit behavior of T_{out} along with the example of Fig. 4.1(a). We use the same boxes/circles with the same color for showing bodies/loads that are the same to make them more visible in different topologies. (b) Our proposed co-learning-based NN models: Training of accessory-1 and accessory-2 models using T_1, and T_2 dataset, respectively. (c) Our proposed co-learning-based NN models: Training and deployment of the main NN using T_3 dataset. 85
4.3 High-level platform architecture of AnGeL.
S_o: Desired specifications for the overall circuit given by users.
S_1, S_2: The specifications for each sub-circuit that are determined by the overall topology decider module.
S_F: Specifications of sub-circuits that are determined by the bigger sub-circuits that they are part of. 87

4.4 Overall topology decider flow. (a) Two examples of high-level topologies. (b) Net-specification-calculator model. (c) Overall-specification-breaker model in the training phase. (d) Overall-specification-breaker model in the deployment phase.
S_o: The vector of overall circuit specifications.
$[S_1, \ldots, S_N]$: The associated specifications sets of N sub-circuits.
X_R: A vector of design parameters that are not included in any sub-circuits.
$[NSC_1, \ldots, NSC_M]$: The associated net specification calculator models of M high-level topologies.
[H-T_1, ..., H-T_M]: M high-level topologies. 90
4.5 Constraints transformation algorithm flow chart based on Equation (4.8). The initial target is set as $S_o = [C_1, \ldots, C_{M-1}, C_M]$ where $C_M = \frac{S_{oM}^{high} + S_{oM}^{low}}{2}$. 97
4.6 The supported high-level topologies for OPAMPs. All stages for one- and two-stage OPAMPs can have either single-ended or differential inputs and outputs. There is a DC-biasing sub-circuit before the second stage gain block that is not shown here. (a) One-stage OPAMP. (b) Two-stage OPAMP. (c) There-stage OPAMP. 97
4.7 The supported sub-circuits for OPAMPs. (a) Gain block sub-circuit body structures; The range of supported design parameters is written next to each in the form of [low, high]. (b) Gain block sub-circuit loads; The range of supported design parameters is written next to each in the form of [low, high]. (c) An example of a body structure with a load. (d) An example of the differential mode of a single-ended topology (circuit (c)). (e) The topology of the current source type sub-circuit. (f) The topology of DC-biasing type sub-circuit. 99
4.8 The confusion matrix of OPAMPs topology selection. The NN model with [128, 128, 128, 128] nodes is used for this purpose which gives the highest accuracy among the evaluated methods. 102
4.9 Desired vs determined values by running the entire AnGeL platform for different specifications. More than 1,200 instances including one-, two-, and there-stage OPAMPs are tested. (a) BW. (b) Gain. (c) Power. (d) Noise. 104
4.10 The generated OPAMP circuit by AnGeL with its sizing to meet Table 4.7 specifications. Size of transistors are shown as $\frac{W(\mu)}{L(\mu)}$. . . . 106
4.11 Power at each iteration step while constraints in Equation (4.9) are satisfied. 107
4.12 The supported sub-circuits for filters. (a) Low-pass filters. (b) High-pass filters. (c) Cascading LP and HP filters results in band-pass or band-stop filters based on the relative position of the corner frequency of HP/LP filters. (d) An example of a band-pass/band-stop filter which is created by cascading a Sallen-Key LP with a TIA HP. . . . 108

4.13 The generated band-pass filter by AnGeL with its sizing to meet Table 4.12 specifications. Size of transistors are shown as $\frac{W(\mu)}{L(\mu)}$. . . . 111
4.14 The frequency responses of Fig. 4.13 filters to meet Table 4.12 specifications. (a) LP filter. (b) HP filter. (c) Overall band-pass filter. . . 112
5.1 An overview, concept, and the impact on the training set of (a) the conventional analog and RF circuits functional modeling method vs (b) the proposed modular S-parameters-based approach. The conventional method needs a whole separate dataset and model for each circuit topology while the proposed approach works properly for many circuit topologies with a single general format E-networks' S-parameters dataset.
$\begin{bmatrix} S_{11} & S_{12}S_{21} & S_{22} \end{bmatrix}$: S-parameter matrix.

DP: Design parameters. 117
5.2 (a) Excited and incident waves in a two-port E-network which are used for calculating S-parameters. (b) An example of a simple L-C network with the calculated S-parameters. s = j2πf where f is the frequency. Z_0 is the reference impedance (the impedance of excitation ports) which equals to 50 Ω in most cases. 119
5.3 (a) General circuit division into E-networks. (b) Main NN in the proposed approach with CCI-NN structure. (c) Main NN in the proposed approach with FC-NN structure. (d) Sub-NNs in the proposed approach. (e) The original CCI-NN model structure [177] with the circuit design parameters as inputs. (f) The conventional FC-NN model in the analog and RF functional modeling with the circuit design parameters as inputs.
s_i: S-parameters of the i^{th} E-network; $\mathbf{x_i}$: Design parameters vector of the i^{th} E-network; $\mathbf{x_R}$: A vector of the circuit's design parameters that are not included in any E-networks. 120
5.4 Final FuNToM model by concatenating the sub-NNs and the main NN.
s_i: S-parameters of the i^{th} E-network.
$\mathbf{x_i}$: Design parameters vector of the i^{th} E-network.
$\mathbf{x_R}$: A vector of the circuit's design parameters that are not included in any E-networks. 122
5.5 Different ways for partitioning 8 topologies. (a) First scenario: each is divided into two E-networks. (b) Second scenario: there is only one E-network. 125
5.6 Phase shifters tested on FuNToM. (a) General structure. (b) Network_1 and network_2 topologies. 126
5.7 Main model results for phase shifters. (a) R2-score comparison with different hyperparameters used for training. Three hidden layers with 32 neurons at each layer is the simplest model that gives a high R2-score of 0.99. (b) Loss vs. epoch diagram while three hidden layers with 32 neurons at each layer model is used for training. 128

5.8 An example of a phase shifter tested on FuNToM. (a) Topology. (b) The simulated (ground truth) and the predicted insertion phase by FuNToM over frequency. Left: when both switches are open; right: when both switches are closed. 130
5.9 Using FuNToM as the simulator in circuit sizing by NSGA-II algorithm to meet the desired specification in Table 5.5. (a) The given phase shifter topology. (b) Insertion phase results over frequency for different switch conditions. 130
5.10 Two-stage LNAs tested on FuNToM. (a) General structure; 1^{st} stage is a common-source LNA while the 2^{nd} stage is a cascode LNA. (b) Input and output matching networks for each stage. 132
5.11 An example of a two-stage LNA tested on FuNToM. (a) Topology; size of transistors are shown as $\frac{W(\mu)}{L(\mu)}$. (b) Schematic modeling: The simulated (ground truth) and the predicted transducer gain (G_T) by FuNToM over frequency. Left: Common-source (1^{st}) stage; middle: Cascode (2^{nd}) stage; right: the whole LNA (two-stage). 134
5.12 The layout of Fig. 5.11(a). 135
5.13 Post-layout modeling of Fig. 5.12: The simulated (ground truth) and the predicted transducer gain (G_T) by FuNToM over frequency. Left: Common-source (1^{st}) stage; middle: Cascode (2^{nd}) stage; right: the whole LNA (two-stage). 135
6.1 An example of an operational amplifier. 141

LIST OF TABLES

Table

2.1 Summarized HDNN's inputs, their sources and the number of components that can be determined at each peripheral section. 32
2.2 Average accuracy of the category recognizer and the training time for different number of input words. The training times are normalized to the training time of 512. 40
2.3 Comparison of the search optimization methods on the COTS DB. . 40
2.4 Main components search results within the COTS DB. Main components that are used in 400 human-designed SBCs are set as the target of these searches. 42
2.5 DRC results for the proposed peripheral circuit generator; the test set includes 400 SBCs. 42
2.6 Peripheral circuit generator accuracy in comparison with the human-designed SBCs; the test set includes 400 SBCs. 44
3.1 Comparison of related works. 54
3.2 Different table identification comparison. 71
3.3 ICDAR data extraction results Comparison. Both DeepDeSRT and TableNet are trained on a subset of ICDAR dataset while Tablext does not use any ICDAR data to train with which proves Tablext's ability to extract data from tables in general formats. 71
3.4 Diverse Data Extraction Results Comparison. 72
4.1 Comparison cost of AnGeL and the conventional approaches regarding the curse of dimensionality. n_1 and n_2: number of design parameters in two different sub-circuits. k: number of samples for each parameter. 89
4.2 Statistics of tested specifications for different OPAMP sub-circuits. . 98
4.3 Number of simulations and generated data using the topology-combiner model for single-stage OPAMPs. 100
4.4 MAPE of the determined specifications by the topology combiner. More than 650 instances from 17 different single-stage OPAMP topologies are tested for each specification. 100
4.5 Required number of labeled data per topology comparison between the state-of-the-art works to achieve an average MAPE of 0.045 in estimating the functionality of OPAMPs. 101

4.6 Average runtime and MAPE comparison of different sub-circuit sizing models. 103
4.7 An example of the OPAMP specifications given to AnGeL and the simulated specifications of the associated generated circuit by AnGeL. 105
4.8 Statistics of tested specifications for filters. 107
4.9 Required number of labeled data per topology comparison between the state-of-the-art works to achieve an average MAPE of 0.06 in estimating the functionality of filters. 109
4.10 Model accuracy comparison for the filter topology-selector. There are 8 topologies for HP & LP and 32 topologies for band-pass & band-stop filters in total. 109
4.11 MAPE of determined specifications by running the entire AnGeL platform. More than 1,200 filter instances are tested. 110
4.12 An example of the filter specifications given to AnGeL and the simulated specifications of the associated generated circuit by AnGeL. . 110
5.1 Required number of training data comparison between two scenarios of Fig. 5.5 for the main NN, sub-NNs, and total. 124
5.2 Statistics of performance of interests for phase shifters. 126
5.3 Average R2-score of FuNToM models tested on different specifications of phase shifters shown in Fig. 5.6. 128
5.4 Required number of training data, per frequency, comparison between the state-of-the-art works to achieve an average R2-score of 0.95. All models are tested on 1600 different phase shifters with the general structure of Fig. 5.6 in 36 topologies. 129
5.5 An example of using FuNToM vs. SPICE as the circuit simulator for sizing Fig. 5.9(a) circuit. Desired vs. generated specifications at 2 GHz frequency for different switch conditions are compared. 131
5.6 Statistics of performance of interests for LNAs. 133
5.7 Average R2-score of FuNToM models tested on different specifications of LNAs shown in Fig. 5.10. 133
5.8 Required number of training data per frequency as well as the average time for collecting the training set in post-layout modeling per sample comparison between the state-of-the-art works to achieve an average R2-score of 0.95. All approaches are tested on 1600 different two-stage LNAs with the general structure of Fig. 5.10 that are selected from 2401 available topologies. 136

ABSTRACT

Custom Analog and Mixed-Signal (AMS) circuits have broad applications in wireless communication, biosensors, etc. [1, 2]. Traditionally, the design parameters of AMS circuits *e.g.* transistor size and biasing are calculated manually by designers [3]. However, the complexity of physical models and severe process variations with downscaling of technology node for large AMS circuits leads to inefficiency and hardship of robust manual design especially in corner extraction [4, 5]. Moreover, generating post-layout simulation is very time-consuming. Therefore because of the growing demand of high performance, low power and time-to-market Integrated Circuits (ICs), there is a crucial need of autonomous AMS circuit design. This would catch up with the digital circuit designs, which have been automated for a long time [6, 7].

Automating AMS circuit design procedure has always been challenging as it tightly ties to human expertise and intuition to make a relationship between various parameters and performances. Given the main circuit topology, in order to satisfy the desired specifications, the circuit parameters should be chosen optimally [8]. So, this circuit optimization problem, *i.e.* determining the circuit parameter values to meet the required specifications, can be solved using mathematical optimization methods. As a result, a Computer-Aided Design (CAD) tool would be able to automate this circuit sizing optimization procedure leveraging techniques such as gradient-based, convex optimization, and evolutionary algorithms [9, 10].

Recently, Machine Learning (ML) has shown promising results in different areas from image categorization to speech recognition. ML algorithms are suitable for yield estimation and making high-order models which can be used instead of complex physical

models with a less computational cost for AMS circuit design [11–13]. Furthermore, ML can be leveraged to replace lengthy and costly measurements by a set of simpler measurements to ease the AMS production test [14]. The main ML challenge to have an accurate model is providing robust and enough simulation data for training sets due to essential high-dimensional variation space to model the process variation and costly simulation in a growing AMS system size. Also, providing Intellectual Properties (IPs) from different IC companies [15–17] is another barrier of dataset collection. Automatically optimized AMS circuit sizing is another application of ML. In such approaches, Electronic Design Automation (EDA) tools try combinations of design variables to find a sizing that meets the desired specifications instead of the traditional circuit sizing approach *i.e.* going from the target specification to the corresponding device sizes [18]. After automation of each AMS block sizing, automation of whole System on Chip (SoC) is doable with integrating these pieces [19, 20].

This thesis proposes a series of works that uncover unique methods for automating the design of AMS circuits. **First**, we propose a fully automated Single-Board Computer (SBC) generator tool, FASCINET. FASCINET uses a Neural Network (NN) model to design customized peripheral circuits for SBCs. The tool creates a large Commercial Off-the-Shelf Database (COTS DB) of existing components, efficiently searches through them, and selects optimal components for both main and peripheral components based on the user's requirements. Creating such a broad COTS DB requires processing abundant datasheets. In order to automate this process, we describe a novel NN-based approach for automatically categorizing datasheets and propose an extraction technique for parsing relevant functional information. Our evaluations show that FASCINET is able to design SBCs that are identical to the manually-designed ones except for minor differences. **Second**, we go over our table extraction tool, Tablext, which can be used on datasheets or any other documents.

Third, we present AnGeL, a fully-automated analog circuit generator framework.

AnGeL performs all the schematic circuit design steps from deciding the circuit topology to determining the circuit parameters *i.e.* sizing. Furthermore, we present a method to reduce the size of AnGeL's database. For this purpose, we use NNs to determine the behavior of complicated circuit topologies by combining the more simple ones. By generating such unlabeled data, the time for providing the training set is significantly reduced compared to the conventional approaches. Our results show that for multiple circuit topologies, in comparison to the state-of-the-art works while maintaining the same accuracy, the required labeled data is reduced by **4.7x - 1090x**. Also, the runtime of AnGeL is **2.9x - 75x** faster. **Fourth**, we propose FuNToM, a functional modeling method for Radio Frequency (RF) circuits. FuNToM leverages the two-port analysis method along with NNs for modeling multiple topologies using a single main dataset and multiple small datasets. This significantly reduces the required number of training data. Our results show that for multiple RF circuits, in comparison to the state-of-the-art works while maintaining the same accuracy, the required training data is reduced by **2.8x - 10.9x**. In addition, FuNToM needs **176.8x - 188.6x** less time for collecting the training set in post-layout modeling.

Finally, we conclude our work and give meaningful insights about the current challenges and open issues.

CHAPTER I

Introduction

Custom Analog and Mixed-Signal (AMS) circuits have broad applications in wireless communication, biosensors, etc. [1, 2, 21, 22]. Traditionally, the design parameters of AMS circuits *e.g.* transistor size and biasing are calculated manually by designers [3]. However, the complexity of physical models and severe process variations with downscaling of technology node for large AMS circuits leads to inefficiency and hardship of robust manual design especially in corner extraction [4, 5]. Moreover, generating post-layout simulation is very time-consuming. Therefore because of the growing demand of high performance, low power and time-to-market Integrated Circuits (ICs), there is a crucial need of autonomous AMS circuit design. This would catch up with the digital circuit designs, which have been automated for a long time [6, 7].

Automating AMS circuit design procedure has always been challenging as it tightly ties to human expertise and intuition to make a relationship between various parameters and performances. Given the main circuit topology, in order to satisfy the desired specifications, the circuit parameters should be chosen optimally [8, 23]. So, this circuit optimization problem, *i.e.* determining the circuit parameter values to meet the required specifications, can be solved using mathematical optimization methods. As a result, a Computer-Aided Design (CAD) tool would be able to automate this circuit sizing optimization procedure leveraging techniques such as gradient-based,

convex optimization, and evolutionary algorithms [9, 10]

The main approaches for AMS circuit parameter search automation are deterministic techniques (*e.g.* Linear Programming (LP), convex Nonlinear Programming (NLP), and nonconvex Mixed-Integer Nonlinear Programming (MINLP) [24]), particle swarm intelligence, simulated annealing, evolutionary algorithm, and Bayesian Optimization (BO) [25–28]. Although each of these methods has its own advantages, they have some drawbacks as well. For instance, PSO suffers from a low convergence rate and BO is subjected to having a very long runtime. Furthermore, annealing can readily fall into the local minimum. Also, evolutionary algorithms are stochastic and lack reproducibility. Artificial Neural Networks (ANNs) are another promising method for AMS circuit design automation which can address the aforementioned difficulties [16].

Additionally, artificial intelligence algorithms along with other methods are suitable for yield estimation and making high-order models which can be used instead of complex physical models with a less computational cost for AMS circuit design [11–13]. Also, Deep Reinforcement Learning (DRL) can solve many human decision making problems in general [29]. Furthermore, AI can be leveraged to replace lengthy and costly measurements by a set of simpler measurements to ease the AMS production test [14]. The main AI challenge to have an accurate model is providing robust and enough simulation data for training sets due to essential high-dimensional variation space to model the process variation and costly simulation in a growing AMS system size. Also, providing Intellectual Properties (IPs) from different IC companies [15–17] is another barrier of dataset collection. Automatically optimized AMS circuit sizing is another application of AI. In such approaches, Electronic Design Automation (EDA) tools try combinations of design variables to find a sizing that meets the desired specifications instead of the traditional circuit sizing approach *i.e.* going from the target specification to the corresponding device sizes [18]. After automation of each AMS block sizing, automation of whole System on Chip (SoC) is doable with

integrating these pieces [19, 20].

Now, we are going to survey some recent studies of various AI techniques for AMS circuit design.

1.1 Analog and Mixed-Signal Circuit optimization

1.1.1 Overview and Main Concepts

AMS circuit designers first, decide the circuit topology [30–32] then optimize the corresponding design parameters *e.g.* component sizing, and finally generate the layout [33]. A significant amount of effort has been put into optimizing components sizing because of the large effect that it has on a circuit's performance and power usage.

1.1.1.1 Problem formulation

The AMS design circuit sizing optimization problem can be formulated as follows.

$$\begin{aligned} &\text{minimize } f_1(x), \ldots, f_m(x) \\ &\text{subject to: } c_i(x) < 0, \forall i \in \{1, \ldots, N_c\}, \end{aligned} \tag{1.1}$$

where $x \in R^d$ denotes d design variables *e.g.* width and length of MOS transistors, and $f_l(x)$ $\forall l \in \{1, \ldots, m\}$ are the Figure of Merit (FOM) of the AMS circuits. Each FOM can be deterministic or noisy depending on the design specification. N_c represents the total number of constraints and $c_i(x)$ corresponds to the i-th constraint *e.g.* $x_j \in [p_j^-, p_j^+]$. When $m \neq 1$, usually there is no best design as objectives can be conflicting and it is unlikely to optimize all of them simultaneously. The goal then would be concluding the best trade-off between a set of solutions.

1.1.1.2 Classical approaches

The classical AMS circuit optimization approaches can be classified into the model-based (*e.g.* geometric programming, SVM, ANN, Gaussian Process (GP), etc. [34–45]) and simulation-based methods (*e.g.* Simulated Annealing (SA), PSO, EA, and gradient-based local search with Multiple Starting Points (MSP) which has a better convergence rate than the others [26, 46–56]). Analytical manually derived or regression models with simulated data are leveraged to build global models of the FOM in the model-based approaches, while the optimization is driven directly by the circuit simulations for simulation-based methods.

The model reusability and low computational cost, especially in the case of using Electro-Magnetic (EM) components, are the main advantages of model-based approaches. However, the accuracy of these models are not usually high as the number of design parameters are usually large and object and constraint functions are highly nonlinear [28].

Recently, to combat these drawbacks, hybrid methods that combine both models have been proposed for analog circuits [57,58] as well as mm-wave and Radio Frequency (RF) circuits [59–62]. These hybrid methods run simulations during the optimization procedure to update online models gradually, instead of using pre-built offline models [63–65]. Initially, the model is constructed by the data gathered from random sampling and it guides the selection of the next point towards more optimized performance.

1.1.2 Bayesian-Based Approaches

The Bayesian optimization method after initial sampling, constructs the probabilistic surrogate model of the objective function, which is refined incrementally based on the new data that optimizes the acquisition function *e.g.* Expected Improvement (EI), Lower Confidence Bound (LCB), Probability of Improvement (PI) [66–68]. The model uncertainty is evaluated to balance the exploration, *i.e.* the next point tends to explore

the unknown regions with high uncertainty in the surrogate model, and exploitation, *i.e.* the next point tends to be the optimum with high probability of prediction by the surrogate model, during the optimization [69]. The *de facto* surrogate model used in BO is the Gaussian process model [70] which reduces the required number of circuit simulations and has closed-forms for both its model prediction and model uncertainty. In a GP, letting $f : X \to R$ be a black-box indicating the performance function, for any finite samples $\{x_1, \ldots, x_n\} \in \chi^n$, the vector $[f(x_1), \ldots, f(x_n)]^T$ follows joint multivariate Gaussian distribution *i.e.* $\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \sim N(\mu, K)$, where μ is an $n \times 1$ mean vector and K is an $n \times n$ covariance matrix. The GP can be fully characterized by its mean function, $m(x)$, and its covariance function $k(x_1, x_2)$ *i.e.* $\mu_i = m(x_i), K_{ij} = k(x_i, x_j)$ for all $i, j \in \{1, \ldots, n\}$. K_{ij} denotes the $i-$th row $j-$th column covariance matrix element.

Let training set $D = \{X, y\}$, where $X = \{x_1, \ldots, x_N\}$, $y = \{f(x_1), \ldots, f(x_N)\}$, $m = [m(x_1), \ldots, m(x_N)]^T$, given a new point, x^*,

$$\begin{cases} k(x^*, X) = [k(x^*, x_1), \ldots, k(x^*, x_N)], \\ k(X, x^*) = k(x^*, X)^T. \end{cases}$$

Moreover, the function value, $y^* = f(x^*)$, and y follow the joint Gaussian distribution. The mean $\mu_{y^*|y}$ and the variance $\sigma^2_{y^*|y}$ can be viewed as the prediction, and the confidence of the prediction respectively.

$$\begin{cases} \mu_{y^*|y} = m(x^*) + k(x^*, X)K_N^{-1}(y - m), \\ \sigma^2_{y^*|y} = k(x^*, x^*) - k(x^*, X)K_N^{-1}k(X, x^*). \end{cases} \tag{1.2}$$

A Gaussian noise $\epsilon \sim N(0, \sigma_n^2)$ should be added to $f(x)$ in order to avoid overfitting of the simulation model where σ_n^2 is the variance of the Gaussian noise. If we take

noise into consideration, Equation (1.2) can be rewritten as:

$$\begin{cases} \mu_{y^*|y} = m(x^*) + k(x^*, X)(K_N + \sigma_n^2 I)^{-1}(y - m), \\ \sigma^2_{y^*|y} = k(x^*, x^*) - k(x^*, X)(K_N + \sigma_n^2 I)^{-1} k(X, x^*). \end{cases}$$

Acquisition functions optimally balance the exploration and exploitation. The improvement of y can be formulated as

$$I(y, \tau) = \begin{cases} \tau - y & y < \tau \\ 0 & \text{otherwise,} \end{cases} \tag{1.3}$$

where τ is the minimal value of found $f(x)$. In the GP model, the expectation of improvement can be written as:

$$\begin{aligned} EI(x) &= \mathbb{E}[I(y, \tau)] \\ &= \textstyle\int_{-\infty}^{+\infty} p(y|D, \theta) dy \\ &= (\tau - \mu(x))\Phi(\tfrac{\tau - \mu(x)}{\sigma(x)}) + \sigma(x)\phi(\tfrac{\tau - \mu(x)}{\sigma(x)}), \end{aligned}$$

where D denotes the given training set, θ is the vector of hyper-parameters, $\Phi(.)$ is the Cumulative Distribution Function (CDF) of the standard normal distribution, and $\phi(.)$ is the Probability Distribution Function (PDF) of the standard normal distribution.

Lyu *et al.* [28] propose Weighted EI (WEI) based [71, 72] BO approach for automated multi-objective AMS circuit sizing which can deal with constraints as well. It leverages a GP as its online surrogate model in contrast with the previous AMS circuit optimization methods which consider GP as either offline models or as assistance for EA. This means the surrogate model is refined incrementally when new data are observed. The reason that EI is used as the acquisition function is that it can deal

with constraints as well as the fact that it has a closed-form.

$$\omega EI(x) = \mathbb{E}[I_c(x)] = EI(x) \prod_{i=1}^{N_c} PF_i(x),$$

where $PF_i(x)$ shows the probability that each constraint, $c_i(x)$, be satisfied *e.g.* $c_i(x) < 0$. Moreover, they propose an algorithm for handling multi objective optimization problems.

First, using weighted min-max formulation [73], Lyu *et al.* [28] translate Equation (1.1) to Equation (1.4).

$$\begin{aligned} &\text{minimize} \max_{j \in \{1,\dots,m\}} \omega_j (f_j(x) - f_j^*) \\ &\text{subject to: } c_i(x) < 0, \forall i \in \{1, \dots, N_c\}, \end{aligned} \tag{1.4}$$

where f_j^* denotes the minimum of the j-th objective function. For this purpose, Equation (1.4) is transformed into multi single-objective optimization problems and then, the Bayesian approach can be applied for solving each. However, max-min objective function that is described in Equation (1.4) is not appropriate for the GP model and causes requiring more training set. For solving this issue, they transform Equation (1.4) to

$$\begin{aligned} &\text{minimize } \lambda \\ &\text{subject to: } \omega_j (f_j(x) - f_j^*) < \lambda, \forall j \in \{1, \dots, m\}, \\ &c_i(x) < 0, \forall i \in \{1, \dots, N_c\}, \end{aligned}$$

by leveraging a new variable λ.

Although the standard BO has shown to be promising for the automated AMS circuit design, because of its sequentiality, it chooses only one point at each iteration by optimizing the acquisition function. Lyu *et al.* [3] leverage the idea of selecting a

batch of points at each iteration [74–76] and propose a parallelized BO algorithm via Multi-objective Acquisition Ensemble (MACE) to accelerate the optimization. MACE chooses the best trade-off between acquisition functions by selecting multiple of them and capturing the Pareto front of these functions after the GP model is updated at each iteration.

Yield determines the robustness of a design under process and condition variation [77]. SRAMs need extremely high yield, *e.g.* around 10^{-7} failure rate, which makes the conventional yield estimation techniques such as Monte Carlo-based and corner-based approaches [78,79] computationally expensive. Yield is inherently a stochastic function which makes it impossible to be accessed directly, estimated by yield analysis tools, or optimized by many efficient optimization algorithms such as gradient-based techniques. All conventional yield optimization approaches reduce the computational costs instead of decreasing the number of yield estimations. Wang *et al.* [80] apply BO optimization using the GP model with an EI acquisition function to reduce the number of yield estimations invoking during the optimization procedure.

1.1.3 Multiple Starting Point Approach

MSP optimization generally begins by randomly selecting large amounts of starting points that cover the design space. This first step is called the global phase. Afterward, in the local phase, an efficient local search such as Sequential Quadratic Programming (SQP) [81] is applied to each starting point found in the global phase.

MSP can successfully approximate the global optimum by choosing the best local optimum. Yang *et al.* [82] improve the inherent large MSP computation time by proposing smart-MSP. A heuristic-biased starting point selection is implemented in the global phase to find the starting points that are likely to be close to optimums instead of using randomly selected points. Furthermore, an intermediate phase is added where sparse regression is applied to predict the circuit performances around the starting

points. Moreover, in the local phase, model optimums are used as the starting points for SQP and coupled with Probabilistic-TABU (P-TABU) approach [83,84] to improve the efficiency of local searches.

1.1.4 Neural Network Based Approaches

In the conventional circuit sizing approaches, designers or EDA tools try to find the circuit parameters in an iterative loop while using a simulator in each iteration to evaluate the design. In order to avoid time-consuming simulations, several studies [85–87] have used ANN models to replace and complement the SPICE simulator with an approximated model. They implement the simulator in later stages just to maintain the accuracy. Even though using an ANN instead of a simulator saves time considerably, the training set of such an ANN model should cover the entire design space which consumes a lot of resources. On the other hand, Lourenço *et al.* [18] train an ANN model to directly size and optimize the circuit for given specifications instead of invoking the simulator many times in an iterative loop to achieve the optimized sizing. Although they use circuit sizing solutions from previous optimizations as the training set, they are able to optimize circuit sizes for specifications outside the ones in the training dataset. To have a better NN model, Lourenço *et al.* [18] take into account the following considerations for selecting the ANN hyperparameters *e.g.* number of layers, number of nodes in each layer, etc. First, in order to have a rich encoding, the number of nodes in each layer is increased in the first layers then is decreased toward the output layer. Second, to find a less complicated model, it is better to design it for having a low error model during the training and compensate the overfitting using L2 regularization [88]. Third, for achieving a model which gives the best cross-validation score, it is necessary to explore the hyperparameter space. Two general approaches are grid search which considers all combinations of the specified hyperparameters and random search which takes samples from a hyperparameter space with a specified

distribution [89].

Although it successfully predicts the analog IC sizing for the target performance, its cost for providing training data is still too high, which makes it impractical. Pan *et al.* [90] propose a performance exploration method to tackle the major evolutionary-based approaches shortcoming, low accuracy, while significantly speeding up the runtime. For this purpose, they apply Bayesian regression to better model the device variables and use an SVM to increase the model performance space. In other words, they replace the performance evaluation process with SVM predictions. However, they leverage supervised learning which requires a large dataset. This dataset is difficult to obtain because most analog IPs are not available to the public.

Wang *et al.* [16] leverage RL which learns to efficiently optimize the transistor parameters automatically without any prior knowledge about circuit design rules. In each iteration, after observations, including monitoring DC operating points, AC magnitude, and phase responses, they change the circuit parameters based on the simulator results. Then, a reward would be received to optimize the desired FOM. They compare their approach for optimizing an amplifier and meet the specifications constraints such as the specified bandwidth, gain, power, area, etc. with other methods such as a human expert, [3], etc. The results show that the other compared methods either are not able to meet the constraints at the same runtime or they have a less efficient design in comparison to [16]. Hakhamaneshi *et al.* [29] find the optimal size of analog circuits by predicting the feasibility of a design using a DNN classifier. They propose a framework that in each iteration utilizes an evolutionary algorithm to create new candidate designs and leverage a DNN classifier to recognize "bad" new offspring by comparing with a reference design that is chosen from the previous "good" population.

1.2 Analog and Mixed-Signal Circuit Performance Modeling

1.2.1 Overview and Main Concepts

The main goal of performance modeling is finding f as a function of device-level variations, $\mathbf{x}$, and device-level operation point to approximate the Performance of Interest (PoI). The PoI is represented with the variable y. An example of device-level variations is $\Delta_{V_{TH}}$. DC bias voltage and voltage gain of an operational amplifier are examples of device-level operations and a performance model respectively. In other words, performance modeling tries to find f which establishes a mapping from $\mathbf{x}$ to y such that:

$$y \approx f(\mathbf{x}) = \alpha.\mathbf{g}(\mathbf{x}) \tag{1.5}$$

Equation (1.5) shows an inner product of $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_M]$, the model coefficients vector, and $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_M(\mathbf{x})]^T$, the basis functions vector. Linear and quadratic polynomials are examples of these basis functions. Finding the model coefficients is the only way for solving Equation (1.5). One way for doing so is the traditional least-squares fitting approach which collects N sampling points of $\mathbf{x}$ and y where $\mathbf{y} = [y^1, y^2, \ldots, y^k]^T$. Then, it solves $\min_{\alpha} \|\mathbf{y} - \mathbf{G} \times \alpha^T\|_2^2$. $\mathbf{G}$ is a $N \times M$ matrix where M denotes the total number of basis functions and $\| \centerdot \|_2^2$ stands for the L_2-norm.

1.2.2 Bayesian Model Fusion Based Approaches

There are several studies which have applied Bayesian Model Fusion (BMF) [15, 91–97] to accurately estimate the parametric and the statistical distribution of circuit performance for both pre-silicon verification and post-silicon validation. Wang *et al.* [94] leverage the idea of BMF for performance modeling. The fusion in BMF means passing information as the prior knowledge from a simple early-stage model, *e.g.* schematic-level, to a late-stage performance model, *e.g.* post-layout, to reduce the

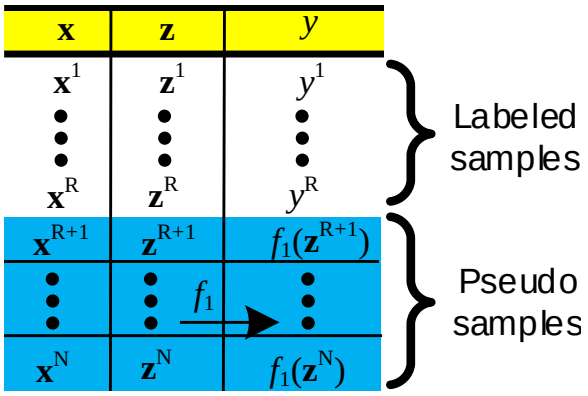


Figure 1.1: The CL-BMF procedure [91]. After fitting the less complex model, f_1, using a small number of labeled samples collected by simulations, a set of pseudo samples are generated with almost no cost. The more complex model, f, is trained then with all the labeled and the pseudo samples.

expensive late-stage modeling cost. Hence, BMF after approximating the early-stage performance model using simulated data, generates sample points for the late-stage model for the same performance using this early-stage model. This prior knowledge is then combined with very few late-stage sampling points to solve the late-stage model coefficients via Bayesian inference [98]. As a result, only a small number of sampling points are required to fit a high-dimensional late-stage model.

In addition to AMS performance modeling, BMF has applications on other AMS design aspects. In order to have a reliable circuit, yield should be estimated and optimized accurately by designers. The conventional yield estimation approaches *e.g.* Monte Carlo based and corner-based approaches require collecting many simulation samples to accurately estimate the performance which makes them extremely computationally expensive. Li *et al.* [92] propose BMF for AMS circuit parametric yield estimation. Fang *et al.* [96] use BMF to estimate Bit Error Rate (BER) in high-speed I/O links. While all the aforementioned studies consider modeling a single performance, Huang *et al.* [95] leverage BMF for approximating multiple correlated performances assuming the probability distribution of these metrics is jointly Gaussian.

Wang *et al.* [91] use the concept of co-learning [99] and propose Co-Learning BMF

(CL-BMF) to even reduce the training set in the early-stage model. They pass the knowledge from a less complex model which predicts the performance of a circuit with a small trained dataset to a more complex model that predicts the performance of the same circuit. Using this prior knowledge, the complex model skips the initial steps of training which saves time. In other words, the more complex model reuses the information generated accurately from the less complex one for its training set instead of relying on the expensive training data only. Fig. 1.1 depicts CL-BMF steps where $\mathbf{z}$ is a vector of performance metrics that 1) is low-dimensional, inexpensive to simulate, 2) does not share any common element with $\mathbf{x}$, and 3) $y \approx f_1(\mathbf{z})$ such that f_1 is a less complex model and needs smaller training set than f in Equation (1.5).

1.2.3 Semi Supervised Learning Approaches

All the aforementioned approaches are classified as supervised learning as they use only labeled training sets. To reduce the amount of work required for gathering the labeled data, several studies have shifted gears toward semi-supervised learning [13, 15, 97, 100]. This is used because less labeled data is required to build accurate models. Alawieh *et al.* [15, 97] propose Bayesian co-learning hierarchical performance modeling. It is called as such because they partition the entire circuit into multiple blocks, *e.g.* Low Noise Amplifier (LNA), mixer, etc. for a transceiver circuit and use the concept of co-learning. In this semi-supervised learning approach, the block-level performance can be modeled with low computational cost. After partitioning the circuit, they map block-level performance metrics, *e.g.* gain of an LNA, to the PoI at the circuit level using a low-dimensional model. They train both block-level and mapping models on a small number of labeled samples and these models generate pseudo samples of un-labeled data at almost zero cost. Finally, the high-dimensional circuit level model is fitted by combining the aforementioned low-dimensional block-level and mapping models. This leverages both labeled and unlabeled data at a very

low cost. Although this approach remarkably reduces the modeling cost, its application is limited to circuits where a block-level partitioning is possible and intuitive.

Alawieh *et al.* [100] propose a semi-supervised co-learning-based framework without any assumption about the AMS circuit structure that leverages two views for each device in the circuit. The device level variations *e.g.* $\Delta_{V_{TH}}$, and independent random Process Variables (PV). They use a small number of labeled samples to build an initial model for each view. Then, at each iteration, highly-confident pseudo samples which are generated by the other model are combined with the small number of available labeled samples to update the model. In other words, they build $y \approx f_1(\mathbf{z}) = \beta.\mathbf{c}(\mathbf{z})$ model along with $y \approx f(\mathbf{x}) = \alpha.\mathbf{g}(\mathbf{x})$ where y is the PoI, $\mathbf{x}$ and $\mathbf{z}$ are vectors containing the device level variations and PV respectively. Then, to update, *i.e.* make a better estimation of $y \approx f(\mathbf{x}) = \alpha.\mathbf{g}(\mathbf{x})$, they use the combination of pseudo samples generated by $y \approx f_1(\mathbf{z}) = \beta.\mathbf{c}(\mathbf{z})$ and a small number of labeled samples. This procedure is shown in Fig. 1.2.

1.3 Dissertation Thesis

This thesis proposes a series of works that uncover unique methods for automating the design of AMS circuits. First, we propose a fully automated Single-Board Computer (SBC) generator tool, FASCINET [101]. FASCINET creates a large Commercial Off-the-Shelf Database (COTS DB) of existing components, efficiently searches through them, and selects optimal components for both main and peripheral components based on the user's requirements. Moreover, in order to automate populating the COTS DB, we describe a novel NN-based approach for categorizing datasheets and parsing relevant functional information. Second, we go over our table extraction tool, Tablext, which can be used on datasheets or any other documents [102]. Third, we present AnGeL [103], a fully-automated analog circuit generator framework. AnGeL performs all the schematic circuit design steps from deciding the circuit topology to

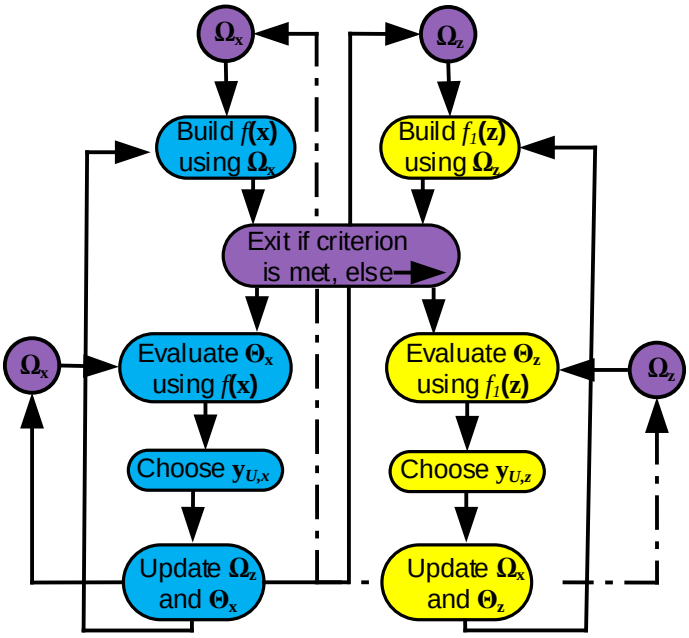


Figure 1.2: The semi-supervised co-learning framework [100]. $\mathbf{\Omega}$ and $\mathbf{\Theta}$ denote training and unlabeled dataset respectively, while $\mathbf{y}_{U,x}$ and $\mathbf{y}_{U,z}$ represent pseudo labels for unlabeled samples $\mathbf{\Theta_x}$ and $\mathbf{\Theta_z}$ respectively. After building f and f_1 models from the training set, highly-confident pseudo samples are chosen from the unlabeled dataset. These highly-confident pseudo samples are used to update the training set of the other model at the next iteration.

determining the circuit parameters *i.e.* sizing. Fourth, we propose FuNToM [104], a functional modeling method for Radio Frequency (RF) circuits that significantly reduces the required number of training data. Finally, we conclude our work and give meaningful insights about the current challenges and open issues.

In addition to the aforementioned works, I have also collaborated on multiple projects for data retrieval [102, 105], System-on-Chip (SoC) design automation, and manually designing RF and VLSI circuits. In FASoC [17, 19, 20, 106], we presented the world's first autonomous mixed-signal SoC design framework, driven entirely by user constraints. The VLSI projects [107–109] were mainly focused on designing energy-efficient and high-performance processors. In the RF circuit design project that I participated in, we implemented a fully-integrated wideband Terahertz (THz) autodyne Frequency-Modulated Continuous-Wave (FMCW) radar for dental imaging [110].

CHAPTER II

FASCINET: A Fully Automated Single-Board Computer Generator Using Neural Networks

Designing Single-Board Computers (SBCs) is becoming more challenging given the growing number of discrete components that are made available and the rate at which this number grows. Keeping track of all available components options, revisions, and functionalities is challenging for SBC designers who are striving for faster design cycles. Moreover, the procedure of deciding peripheral components, their values, and connections of an SBC is not only difficult because of various parameters that need to be considered, but also is time-consuming as there exist numerous components on a typical SBC nowadays. In this project, an SBC generator tool, FASCINET, is presented that uses a Neural Network (NN) model to design customized peripheral circuits for SBCs. The tool creates a large Commercial Off-the-Shelf Database (COTS DB) of existing components, efficiently searches through them, and selects optimal components for both main and peripheral components based on the user's requirements.

Creating such a broad COTS DB requires processing abundant datasheets. A manual approach is time-consuming, even if only a fraction of all available datasheets is considered. In order to automate this process, we describe a novel NN-based approach for automatically categorizing datasheets and propose an extraction technique for parsing relevant functional information from tables within. Our evaluation using a

test set that contains over 770,000 components shows that the category of datasheets is identified correctly over **95%** of the time. Additionally, the table extractor has a precision above **96%**. Our proposed fully autonomous SBC design approach reduces the time for generating the schematic of an SBC to **as little as two minutes**. For validating the accuracy of our model, the netlists of 400 SBCs designed by FASCINET are compared to the human-designed versions. This evaluation shows that FASCINET is able to design SBCs that are identical to the manually-designed ones except for minor differences.

2.1 Introduction

Single Board Computers (SBCs) are widely used in electronic industries, medical instruments, etc. They are low cost and have broad applications from control and measurement to online communication [111, 112]. Typically, SBCs include two parts: one or more main discrete components such as microprocessors, microcontrollers, etc. that target the main goal of the SBC, and second, discrete peripheral components *e.g.* capacitors, voltage regulators, etc. around the main ones for different functionalities– power, reset, and interface ports to name but a few [113]. It is worth mentioning that discrete components are single electronic components in a simple package which can be soldered to a Printed Circuit Board (PCB) [114].

Finding the desired main components, considering multiple parameters and specifications, is a tedious task for SBC designers as the number of available components is growing at a fast pace. This becomes even more challenging when taking the peripheral components into account. A designer needs to determine various peripheral components, their values and connections. This process is time-consuming, challenging, and will make automated SBC generation a necessity rather than a luxury given the numerous components on an SBC.

In this work, we present a tool, FASCINET, for automatic SBC generation that ad-

dresses all the above-mentioned challenges. FASCINET designs customized peripheral circuits and generates the SBC schematic in the industry standard format by giving the main components of the SBC as the input. This design procedure includes all steps of automatically and intelligently determining all the required peripheral components, how they connect to each other and their values *e.g.* resistance of resistors, capacity of capacitors, model of voltage regulators, etc. FASCINET leverages our novel Hierarchical Deep Neural Network (HDNN) approach to determine the required peripheral components starting with components that could be directly connected to the main components. Then, it hierarchically determines components further away. Next, a separate Deep Neural Network (DNN) model determines the value of each component specified by the HDNN. The generated schematic is passed to a downstream PCB layout generator tool which produces the final PCB of the SBC.

Users have two options for inputting the information of the main components. First, directly inputting the datasheets of the main components. Second, giving the high-level intents and specifications of the main components that they are looking for, and asking FASCINET to find it. For the first case, we propose a novel, open-source[1], fully automated datasheet scrubber to extract the key specifications in the associated datasheet. The datasheet scrubber implements a Convolutional Neural Network (CNN) based approach [115] for category recognition and table extraction on a single platform.

For the second case, FASCINET efficiently searches through its large Commercial Off-the-Shelf Database (COTS DB) in order to find and select an optimal set of components that meet the user's input requirements. With specifying the target optimization method *e.g.* power and price, the tool is also able to optimize the SBC in the desired way while satisfying the other user intents such as the specifications of the desired main components. For instance, specifications of a microcontroller include

[1]Source code for the datasheet scrubber can be downloaded from `https://github.com/idea-fasoc/datasheet-scrubber`

memory capacity, processing power, type of I/O ports, frequency range, throughput, etc. [116]. A user may need two microcontrollers with different specifications as the main components while minimizing their total price. FASCINET searches through the COTS DB and finds the microcontrollers with the closest specifications to each of the desired ones and selects two of them with the minimum total price. This gives a great advantage to FASCINET over a manual design by considering a variety of possibilities, constraints and components which is never possible in a manual design.

FASCINET COTS DB contains meta-data for more than 770,000 records of existing components in more than 50 different categories *e.g.* microcontroller, voltage regulator, Analog-to-Digital Converter (ADC), etc. Populating such a broad database is tedious and is subjected to errors when done manually. Typically, functionality descriptions of the discrete electronic components are reported in datasheets [17]. Annually, thousands of new components are designed and added to the multitude of existing ones. The datasheet scrubber automatically processes these documents and summarizes their performance specifications which enables creating such a large database. Moreover, for each component, it recognizes the main functionality of each pin, *e.g.* power, ground, reset, etc. which is an essential step for automatically connecting components.

An exclusive database with more than 2,500 open-source SBCs [117,118] is gathered for training the HDNN model and testing the SBC peripheral circuit generation. To validate the performance, the netlist of more than 400 SBCs designed by FASCINET are compared with the human-designed ones. Our evaluations demonstrate that the SBCs generated by FASCINET are identical to the human-designed ones except for minor differences. Even these minor differences are mainly due to different design approaches, rather than functionality deficiencies.

The contributions of this project can be summarized as follows:

- Datasheet scrubber: A novel CNN-based approach to categorize datasheets and extract information within their tables.

- Peripheral circuit generator: A novel HDNN-based model that intelligently designs customized peripheral circuits by giving the main components of an SBC.

- We propose FASCINET that leverages the datasheet scrubber, the COTS DB, and the peripheral circuit generator to design the schematic of an SBC in only **two minutes**.

2.2 Related Work

Several papers and patents have previously described methods for helping users to find the desired components. Birmingham *et al.* [119] propose a method that can help the SBC designer choose various components within a database. Curran *et al.* [120] expand on this and estimate the power, performance, and price of a custom board. The user is prompted for device constraints and the tool recommends components from a library. Likewise, Birmingham and Siewiorek [121] propose an approach which suggests components to designers, then it logically connects those components using templates that describe the purpose of pins. Tools such as these make finding viable components quicker for a designer, however these implementations require a lot of human effort to both continually update their databases and make the components metadata readable by the tools.

Recent works implement more intelligence in suggesting components from their libraries. Anderson *et al.* [122] propose a tool that provides a GUI platform for novice designers. The tool enables users to describe the circuit in a behavioral way. However, it is only compatible with Arduino microcontrollers. When a user searches in library of AutoFritz [123] and Circuito [124] to find a component, the tools also suggest some other components to complete the functionality of the desired component. In AutoFritz, after selecting a component by users, the tool suggests some other components within

its database that are usually connected to that component based on its datasheet. It also suggests the possible connections between these components. However, users are supposed to determine at each step whether they want to use any of the suggested components and if so which one. Therefore, there is still a significant amount of human-in-the-loop interaction required. Also, AutoFritz does not get any high-level specifications of the user's goal and limitations as inputs. For instance, users cannot determine the input voltage that they have access to and ask the tool to find the components accordingly or describe the functionality of the components and ask the tool to find the best ones. Moreover, AutoFritz does not support components that do not exist in its database at all. On the other hand, Circuito has fixed pre-defined peripheral circuits for some components that always come with them. However, similar to AutoFritz, it does not get any high-level inputs from users which makes its designs inflexible. Moreover, it does not support components out of its database. The other drawback of Circuito is that there are many limitations in its designs. For instance, it cannot support more than one microcontroller at each design. Our proposed tool gets high-level user specifications as inputs and automatically designs SBCs with a no-human-in-the-loop approach. FASCINET also using ML learns how to use different peripheral components. So, it supports designs that contain components out of its database.

Several works attempt to ease the process of connecting components. Some of them discuss HW/SW co-design approaches to automate generating systems from high-level user intents [125–127]. Bachrach *et al.* [127], takes PCB hardware description and application software from the user and generates PCB layout in addition to uploading the software code on it. Lin *et al.* [128] propose a system that supports designers in different levels: High-level Description Language (HDL), electronic modeling (to check the connections, power, etc.) and netlist generation. All the aforementioned works rely on a complete manual hardware description from the user and there is

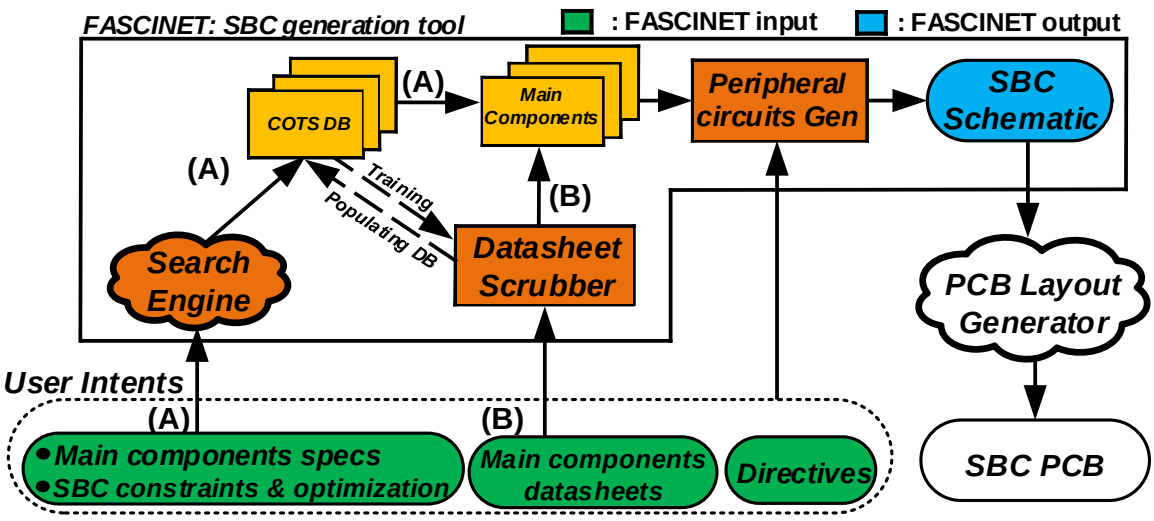


Figure 2.1: High-level platform architecture.

no intelligence by their tool in determining any components or connections. On the other hand, EDASolver [129] after finding components in their libraries, make some connections between them using pin matching. However, this pin mapping is limited to the components in its library and it cannot go beyond that.

The above-mentioned works either help in choosing components or ease connecting them in limited applications while still users have to determine most of the components and connections. Recently Machine Learning (ML) is applied for circuit design as well [130]. By using ML both to keep an up-to-date database of components and to design SBCs, this project aims to address the shortcomings of the previous works.

2.3 Platform Architecture Overview

A high-level representation of the FASCINET platform is shown in Fig. 2.1. Main components are the parts that target the main and overall goal of the SBC. The boxes in green are three inputs that users specify using a config file. Users can choose one of the two ways for inputting the information of the main components: the specifications of them (*path* A in Fig. 2.1) or directly their datasheets (*path* B in Fig. 2.1). Beside

one of these two, users input the directives that contain three main points:

- The voltage range of the input power supply that the SBC is connected to.
- If the Battery/ USB port is needed.
- The description of the interconnections of the main components to each other.

As it will be discussed in Section 2.6, the power supply voltage range and the requirement for any Battery/ USB port are two of the HDNN model inputs to determine the required peripheral circuits. There are in general two types of connectivities for a main component. It is either connected to another main component or a peripheral component. In the first case scenario, the description of interconnections of the main components which is provided by the user is used for automatically connecting them. As an example, a user may specify the clock pin of a main component is connected to the clock pin of another main component. As it will be explained in Section 2.4.3, the datasheet scrubber is able to recognize pins functionality and find the exact pin, *e.g.* clock in this example. The procedure of automatic connection of the main components to peripheral ones as well as peripheral components to each other will be described in Section 2.6.

If users decide to input the specifications of the main components (*path* A in Fig. 2.1), they also would be able to determine some constraints. The total SBC main components power consumption at nominal voltage and their total price budget are the constraints that the SBC can be optimized for. The tool outputs the closest components that it could find within its COTS DB regarding these overall constraints. If the tool finds more than one component that meet the required specifications given by the user, it picks the one with the lower price or power consumption based on the desired optimization approach. The output of the COTS DB is a complete summary of the specifications of the found main components and pins functional information which were parsed previously from the associated datasheet by the scrubber tool.

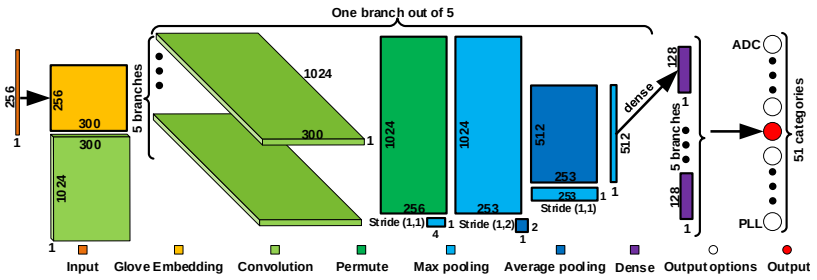


Figure 2.2: The CNN model of the category recognizer. The input is the first 256 words of datasheets and output is one of the 51 categories.

In the case of directly inputting the information of the main components by users (*path* B in Fig. 2.1), the datasheet scrubber parses the datasheets using COTS DB training data and passes the summarized overall and pin functional meta-information to the peripheral circuit generator.

The summarized information of the main components includes their categories, specifications, and other information required for proper functionality. For instance, this summarized information may be: an 8-bit ADC with an input voltage range between 2-5V. Using this meta-information in addition to the directives, the peripheral circuit generator creates the SBC schematic in the industry standard format as an output which contains all the main and peripheral components, and their connections. This schematic can be passed to a downstream PCB layout generator tool.

2.4 Datasheet Scrubber

2.4.1 Category Recognizer

The first step to automatically scrub datasheets is to recognize the category of components, *e.g.* ADC, Phase Locked Loop (PLL), etc. Although the category of some datasheets found in some websites may have been labeled, there are many datasheets without labeling that need manual category recognition and separation which wastes much time and energy and is subjected to errors. Moreover, these labels may not be

precise enough. For instance, a category such as PLL might need to be broken into two subcategories *e.g.* analog PLL and Delay Locked Loop (DLL). Using the datasheet scrubber, this procedure can be done automatically with no-human-in-the-loop which saves a lot of time and energy.

Category recognition is a crucial task since the scrubber should look for different specifications based on the category of the components. In order to effectively and efficiently identify the the correct category, a customized CNN-based model is used which identifies between 51 common categories. It is worth mentioning that some existing approaches such as searching the categories name within the datasheet and selecting the most frequent one might be simpler, but are subjected to a lot of errors. For instance, the datasheet of a microcontroller may talk more about the memory and contains more "memory" keywords rather than "microcontroller". Another not accurate enough approach is searching the categories name within the title. The problem is that mentioning the category in the title is not a mandate and there are many datasheets without having the category mentioned in their titles. Moreover, automating the procedure of realizing the title for the tool would be complicated enough despite it being trivial for humans. Also, the title may contain more than one category. Furthermore, Support Vector Machines (SVMs) have long been used for the text classification and they have achieved reasonable success depending on the database. The general procedure, when using an SVM, is to identify the quantity of each word in a document, then train the SVM using those quantities as inputs [131]. This approach would not produce accurate results for our database due to the similarity between different categories. For instance, when differentiating between an ADC and a Digital to Analog Converter (DAC), the order of the words is very important. This issue can be somewhat resolved by searching for phrases instead of words, but choosing the proper phrases to use is a non-trivial and labor-intensive process. As a result, our customized CNN-based approach is designed to be as simple as possible while having

an acceptable accuracy.

The customized CNN-based model, instead of examining the entire document, analyzes the first 256 words. There are several factors that determine the ideal amount of words to examine per document. Looking at more words allows more information to be evaluated by the network. However, this leads to a wider network, more memory, and a higher computational power. Analyzing hundreds of datasheets shows that a vast majority of them front-load their important information with a short description. This description most often varies between 200 to 250 words in length. With this information, the size of the network has been optimized to take in all of this concentrated data in a vast majority of cases.

The first 256 words are converted to vectors of 300 numbers with the GloVe word embeddings [132]. This step allows the network to understand how words relate to one another and crucially which words are synonyms. A conventional CNN approach would then stack many convolutional layers that are followed by several dense layers. The stacked convolutional layers allow relationships to be discovered between distant words. However, this method has several downsides. Making a robust model with this approach requires a massive amount of weights, increasing the training time and making the model less portable. Another issue is the vanishing gradient problem that arises from stacking many layers [133].

Our customized network has an innovative design that identifies relations between distinct words without the need for a significant number of stacked layers. In this model, there are several branches that compute unique relations between the words. The simplest branches start with filters that are single word in length, so each filter has a size of 1×300. This is in practice a 1D convolution because the size of the filter is equivalent to feature map size in one dimension. The output of this operation is a two-dimensional array where one dimension still represents the location of the words, but the other represents each filter. This array shows how closely the word

represented by a particular filter matches the meaning of the input word. The next step is unconventional where the two axes are permuted before the data is passed into another convolutional layer.

The size of the filter in the next layer, like the previous one, is equivalent to the length of the feature map in one dimension, making its functionality similar to a 1D convolution. However, because the data is permuted, this filter travels across the dimension created by the number of filters in the previous step. This means that it has access to data created by every input word simultaneously. Therefore, it can identify the relations between different words that are an arbitrary distance away. The length of the filter determines the amount of word-meanings that can be compared at each stride. It should be noted that the objective function for this CNN is cross entropy.

This approach does have some downsides compared to a conventional Deep Convolutional Neural Network (DCNN). Because the second convolutional layer looks for words in specific places, it might not be general enough. To rectify this issue, many different branches are used to cover any potential weaknesses. These branches have different filter lengths for the first convolutional layer. This causes phrases to be identified instead of words, allowing the second layer to identify different phrases that are an arbitrary distance away. Additionally, some of these branches have maxpooling layers between the two convolutional layers, helping regain generality.

The CNN model has been depicted in Fig. 2.2. There are in total 5 different branches. Each branch has a specific length of the first filter that are chosen based on the testing results as it will be more discussed in Section 2.7.1.1. In the final step, all of the branches are flattened and then concatenated. The combined network ends with nodes that represent all the different possible categories. To train this network, we use the 770,000 presorted datasheets in the COTS DB.

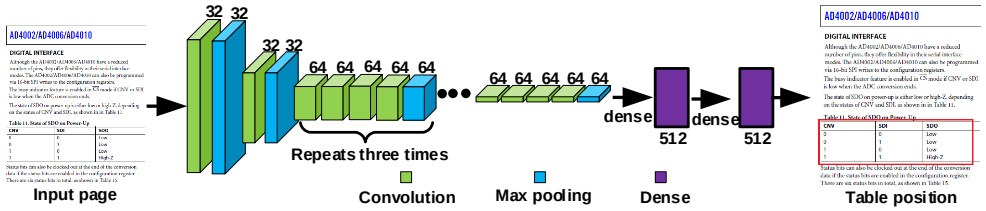


Figure 2.3: The CNN model of the table position identification. The input is the datasheet page and output is the position of the table within it.

2.4.2 Table Extractor

A majority of relevant specifications and pin information of a datasheet are found within its various tables. However, an agreed upon format for the layout of tables in datasheets does not exist. Although there have been some studies on automation of table extraction as it will be discussed in Section 2.2, most of them are either not able to extract all PDF files, or they do not have enough accuracy, or they are computationally expensive. However, our method combines computer vision techniques and CNN models that allows CNNs to just focus on complex problems while computer vision methods take care of the rest. This approach leads to a high accuracy with a low computational cost.

Our customized table extractor starts by converting the datasheets into a series of images. A CNN then goes through these images and finds the locations of the tables [102]. The CNN model for the table position identification is shown in Fig. 2.3. Afterwards, computer vision methods are used to identify the high-level layout of the tables that helps identify and separate cells: First, the locations of high contrast lines within the tables are identified. Additionally, the locations of text within the tables are found by locating high contrast sections that do not belong to the previously found lines.

The identified high-contrast lines are useful for separating cells. However, many tables do not have lines separating every individual cell. An example of this case is shown in Fig. 2.4 (a). In this case, the positions of the data can be used instead.

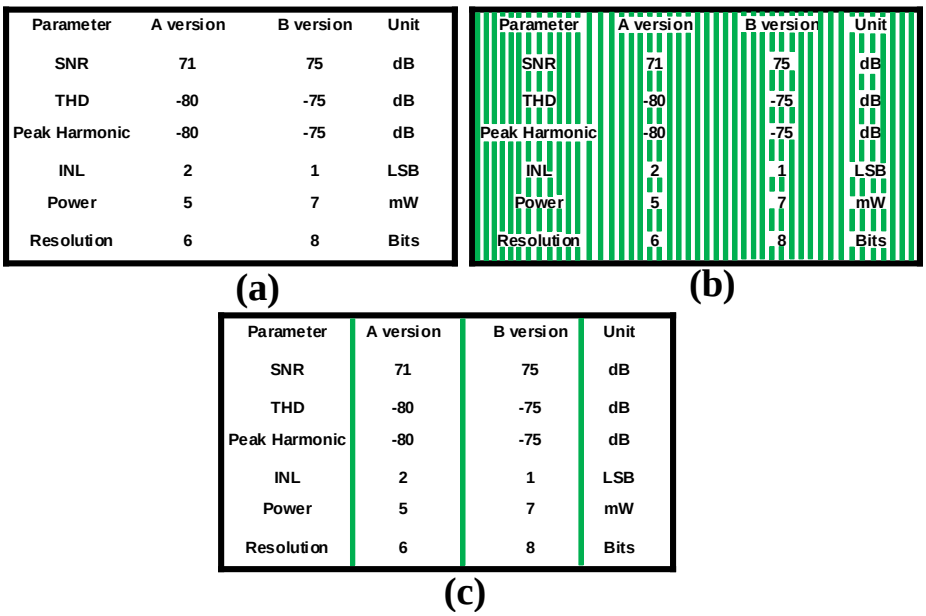


Figure 2.4: Finding invisible cell separator lines. (a) An example of table without cell separator lines. (b) The tool draws straight lines that do not intersect the data all over the table. (c) The final result after removing shortened lines and merging the adjacent ones. For being more clear the horizontal lines have not been shown.

With the locations of the data known, the tool finds these invisible cell separator lines. For this purpose, the tool draws straight lines that do not intersect the data all over the table. Fig. 2.4 (b) shows such vertical lines in green. For being more clear, the horizontal lines are not shown. Assuming l_{vm} and l_{hm} denote the maximum length of vertical and horizontal drawn lines respectively, the tool only keeps vertical and horizontal lines with l_{vm} and l_{hm} lengths respectively, and removes the rest. Finally, the adjacent remained lines are merged as it is depicted in Fig. 2.4 (c).

This process alone is sufficient to properly split a table with consistent rows and columns across the entire table. However, more complex tables which contain cells that do not conform to the general table's layout will have incorrectly split and merged cells. For instance, a single cell may be larger than its adjacent cells and may span two columns. To fix these issues, the cells are sent into a CNN [102] with the purpose of finding which cells should be merged together and which cells should be split. The fixed

cells are then individually sent into the open-source Optical Character Recognition (OCR) software Tesseract [134] to obtain the cell's data. The data and the previously identified locations of the cells are used to recreate the table in a machine readable form.

When the table extractor is used for extracting the specifications of main components in both *path* A and B in Fig. 2.1, there are some common specifications that FASCINET always searches for *e.g.* power, input voltage range, etc. However, the rest are more dependent on users that which specifications are more important for them and the category of the desired component.

2.4.3 Pin Functionality Extractor

After finding the location of the pin description table within a datasheet using the table extractor, the data needs to be interpreted. It is important to know the purpose of each pin for automatically connecting components. The pin functionality extractor identifies if a pin is a ground, power, clock, reset, General-Purpose Input Output (GPIO), etc. For this purpose, a CNN looks at both the pin name and the description.

To learn from the pin name, the CNN takes in each of the individual characters as one-hot vectors. This is done so that patterns within similar pin names can be learned. For instance, "VDD" almost always indicates a pin that should be connected to the power supply. This is true regardless of any additional characters that are in the pin name, so "DVDD" and "VDD0" can both be classified as a power pin.

Additionally, a glove matrix which has 4 layers of 1D convolution followed by a CNN, with cross entropy as the objective function, is used to learn from the description of the pins at a word level. The CNN models for both learning from pins name and descriptions have two hidden layers of 1D convolution. These two networks are concatenated together in order to achieve accurate results.

2.5 COTS Database

In order to find the desired main components based on the user specifications, a database has been configured to allow for immediate access to the collected data. The COTS DB contains more than 770,000 component records from different sources [117, 118, 135]. Because the category of each component has unique specifications, a MongoDB platform is adopted for its flexible data-scheme. MongoDB is a document-oriented NoSQL database that offers high performance, scalable, flexible data-schema [136]. From the performance perspective, MongoDB has been proven to be effective for storing a huge amount of data [137] and it outperforms MySQL in read and write tests [138].

Optimizing the search performance within the COTS DB is crucial to reduce the latency. When a query operation is performed, MongoDB performs a collection scan, meaning it scans every record in the collection to select the ones that match the query statement. Multiple methods such as indexing, projections, and aggregation pipelines [139] are implemented to optimize the COTS DB.

Indexes are special data structures that store a small portion of the collection's dataset in an easy-to-traverse form. This optimizes the execution of queries [140]. The indexing is applied to the price and power of the components. The projections method decreases the query time by limiting the number of either searched or returned fields. Before the projection, the initial number of searched fields was 28 while they have been shrunk to 5 fields based on the most commonly performed queries.

Using the COTS DB, FASCINET is also able to minimize both the total power consumption and price of the main components with regarding to the specifications of the main components. Moreover, FASCINET has the feature of prioritizing the specs which enables users to specify the more important specs for them. In this regard, even if the queried component do not exist within the COTS DB, FASCINET finds the most close one to it.

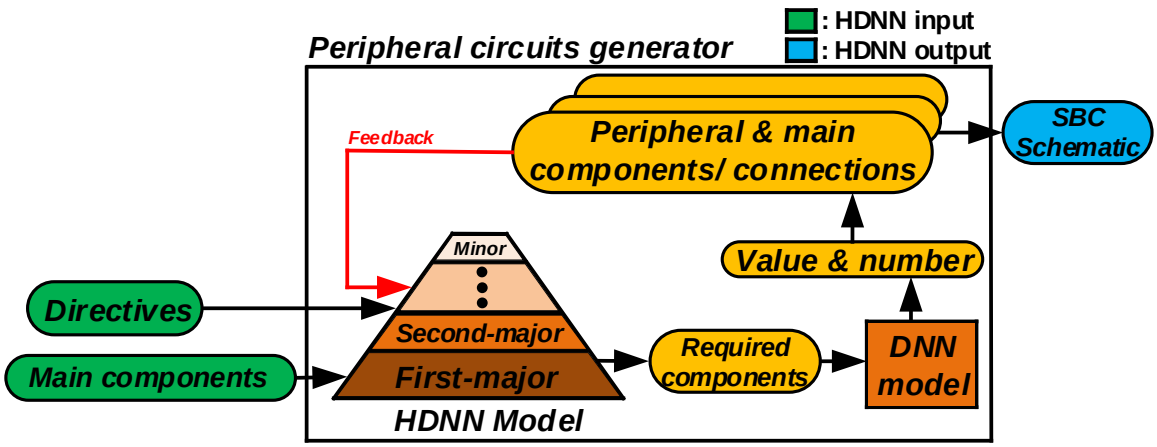


Figure 2.5: Peripheral circuit generator flow using the HDNN model.

Peripheral section	Inputs							Supported peripheral compo-nents #
	Power pins #+	Input voltage range	Battery/ USB port is needed	LED pins #	Components category	Reset pins #	Clock/ oscillator pins #	
Power supply	M, P**	M, P, D	D	-	-	-	-	57
LED	-	-	-	M, P	M, P	-	-	8
Reset	-	-	-	-	M, P	M, P	-	7
External inter-faces	-	-	D	-	M, P	-	-	10
Clock	-	-	-	-	-	-	M, P	3

Table 2.1: Summarized HDNN's inputs, their sources and the number of components that can be determined at each peripheral section.

* #: Number.
** Inputs sources: Main components (M), Directives (D), Peripheral components (P).

2.6 Peripheral Circuit Generator

The main components specify the main intention of SBCs and target their main goals. Using the summarized information of the main components *e.g.* categories, specifications, etc. (that is either extracted by the datasheet scrubber or found within the COTS DB based on the user input), in addition to the directives (that are inputted by users directly), the peripheral circuit generator designs the peripheral circuits around the main components. It then outputs the SBC schematic in the industry standard format. For this purpose, the peripheral circuit generator uses our

novel Hierarchical Deep Neural Network (HDNN) model.

Fig. 2.5 depicts the peripheral circuit generation flow. Given the main components and directives by the user, the HDNN model determines the required peripheral components that could be directly connected to the main components ("first-major"), then it hierarchically and successively determines the other required components that are further away ("second-major", ..., "minor"). Then, a DNN model determines the values *e.g.* resistance of resistors, etc. of these peripheral components that are determined by the HDNN model. As an example, the HDNN model determines if a decoupling capacitor is required to be connected to the main component, and the DNN model after that determines its value *i.e.* the capacity. At the end, unconnected pins would be connected to the correct signal based on the extracted pin functional information by the datasheet scrubber and inputted directives. Finally, the SBC schematic is generated.

The HDNN's inputs come from three different sources. Main components and directives that are fixed at runtime and specified by the user. The third input source is the previously generated peripheral components which is updated during runtime. As the feedback loop is shown in Fig. 2.5, the HDNN model runs multiple iterations to design all peripheral components. It is necessary to update the inputs based on the previously generated components because of the dependencies between them. For example, adding an LED increases the total required power. To account for this, the updated circuit is sent back into the model so that it can determine the peripheral power supply section accurately. Table 2.1 summarises the HDNN's inputs and their sources as well as the number of peripheral components that can be determined by the HDNN for each peripheral section *e.g.* power supply, LED, reset, etc. In total, there are 7 different inputs to the HDNN model and the HDNN is able to determine 85 different peripheral components. It should be noted that "components category" is considered as one input set which contains the category of all peripheral

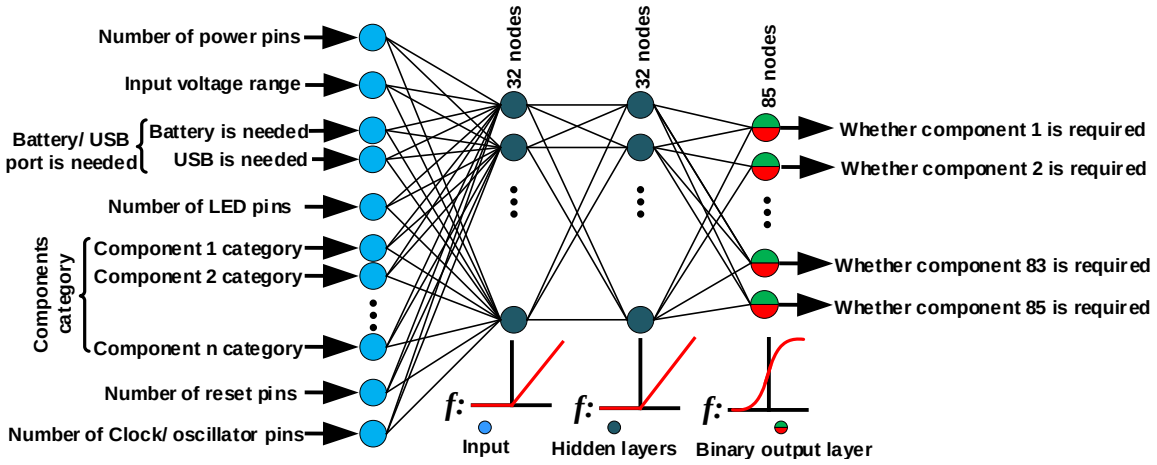


Figure 2.6: HDNN model: A custom made binary classifier in Keras [141] with 2 Fully Connected (FC) hidden layers. The hidden layers have 32 nodes each with a nonlinear RELU activation function, while the output layer has a nonlinear sigmoid activation function. The inputs with more details are described in Table 2.1.

and main components. Also, "battery/ USB port is needed" is divided into two inputs. Each of the peripheral sections contains the components that together have a specific functionality. For instance, power supply section provides constant, stable voltage for whole the SBC and may include some capacitors, voltage regulators, diodes, etc. that are determined by the HDNN model.

As it is shown in Fig. 2.6, the HDNN model uses a custom made binary classifier in Keras [141] with 2 Fully Connected (FC) hidden layers. The hidden layers have 32 nodes each with a nonlinear RELU activation function, while the output layer has a nonlinear sigmoid activation function and cross entropy as the objective function to determine the required peripheral components. As it is shown in Fig. 2.5, it begins by determining the required components at the first level of the hierarchy, first-major, that could be directly connected to the main components in each peripheral section *e.g.* power supply, LED, reset, etc. First-major components may be surrounded by other components, second-major, and so on. The components at the last level of hierarchy, minor, are the ones that already all of the components that they are connected to have been determined by the model and usually one of their pins are connected to

ground *e.g.* decoupling capacitors.

In the HDNN model, the binary classifier at each step determines whether a peripheral component that is observed in the training set is required for the SBC. Those 85 different peripheral components that are determinable by the HDNN model (see Table 2.1), regardless of their values, are different components with different functionalities that are connected to unique nets and have been observed at least in one of the training sample designs [117, 118]. For example, the LED that is connected to a specific pin of a main component and the one that is connected to the power signal are considered as two different components. As another example, voltage regulators are part of the power supply peripheral section that usually are connected directly to the main components. In our training set, there are four different voltage regulator components which means some SBCs have been observed that each has one or more of these voltage regulators. For instance, one voltage regulator may have 3 pins such as "LM117" [142] that is used usually when the input voltage is less than 40V, and the other may have 5 pins such as "AP2112" [143] which is leveraged if the input voltage is between 2-6V. So, the HDNN determines whether the SBC requires any of these voltage regulators.

Automatically connecting the peripheral components to each other and the main ones through the corresponding pins are implied by determining the required peripheral components by the HDNN model. The pin information extracted by the datasheet scrubber is used for this purpose. In other words, during training, the HDNN model not only learns which peripheral component are required to be connected to the main ones, but it also learns through which pins they should be connected to. For example, determining requirement of a voltage regulator around the main component by the HDNN model means that the output pin of the voltage regulator is connected to the power pin of the main component. So, determining which pin of the main component works as power is the last piece of puzzle for automatically connecting them that pin

functionality extractor part of the datasheet scrubber does that. Finally, FASCINET automatically connects all ground pins to ground, all power pins to the appropriate power supply value using their datasheets extracted information, and main components to each other based on the input directives of users. FASCINET also connects all the main components pins to headers. This allows users always have access to inputs and outputs of the main components directly.

The step by step hierarchically design procedure of a sample SBC has been depicted in Fig. 2.7. After passing the information of the main components, the peripheral components that are directly connected to the main components, *i.e.* first-major are determined. A similar design procedure that was described for determining the voltage regulators are happening for this and the next steps. As an example, in Fig. 2.7 (c), for determining which power signals should be connected to the voltage regulator ("AP2112"), there are a lot of options that among them "V_USB" and "V_BAT" are selected.

After determining the required peripheral components, a multi-label DNN classifier determines the values of these peripheral components *e.g.* resistance of resistors, capacity of capacitors, etc. Similar to the HDNN procedure, for each peripheral component, the value is determined among all values that have been observed for that component within the training set. For instance, for a decoupling cap, the DNN model determines which one of the values of 47μF, 10μF, 1μF, 100nF, etc. that are observed in the training should be chosen. Using a multi-label classifier is better than a normal regression for this purpose as the values that may be obtained by the regression model may not even exist *e.g.* there is no 5.34μF capacitor. Occasionally, multiple components of the same category are connected in parallel [144], *e.g.* capacitors that are connected between supply voltage and ground as it is depicted in Fig. 2.7 (c). In these cases, the DNN will also determine the number of each component. It should be noted that sometimes the HDNN model determines the value of components as

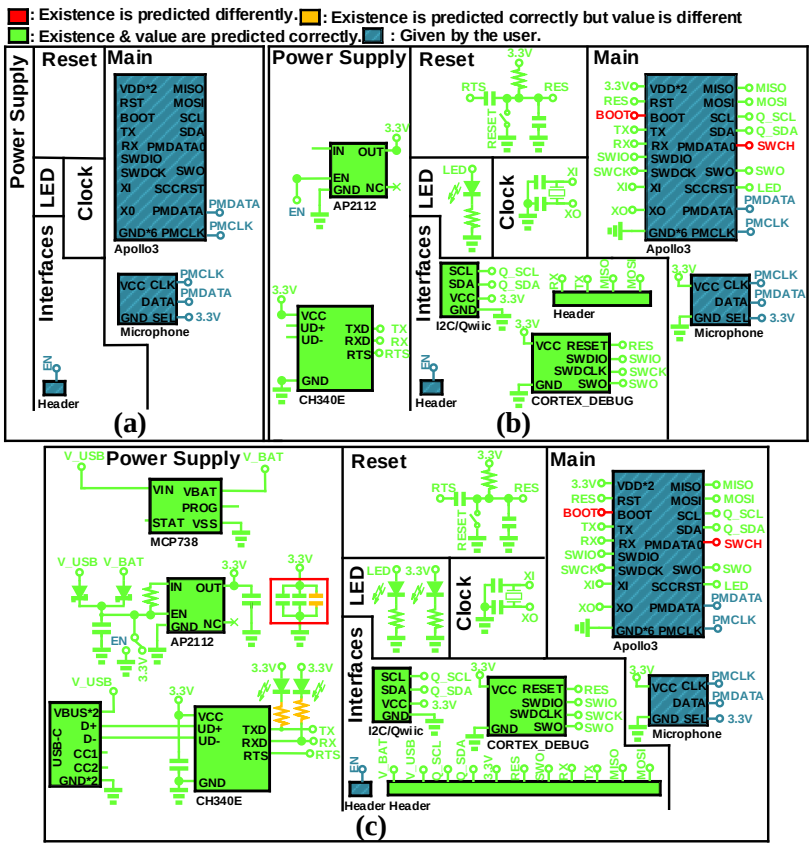


Figure 2.7: Steps of determining the required peripheral components by the HDNN model: (a) The main components are inputted. (b) The peripheral components that are directly connected to the main components, *i.e.* first-major, are determined. (c) The peripheral components around the first-major components, *i.e.* second-major, are determined. The next step which is the complete SBC after determining all the required peripheral components has been shown in Fig. 2.13. An example of parallel capacitors that their number is determined by the DNN model is shown with a red box.

well in the cases when their values affect the rest of the design. Voltage regulator is an example of this since specific components are connected to each of them based on their models.

The training set for both the HDNN and DNN models contains more than 2,500 open-source SBCs [117, 118]. A hierarchical approach is also used for training the HDNN model [15]. A unique training set is provided for determining peripheral components around each component. There is one training set for determining

first-major components. Each of these first-major components has its own specific training set for determining second-major components around them and so on. For instance, two different training sets are provided for determining components around "AP2112" [143] and "LM117" [142] voltage regulators that are trained separately on each. This idea of hierarchical training leads to an optimized result for each as well as reducing the complexity of the network [15].

There are other SBCs which contain peripheral components that are not included in any designs in the training set (*e.g.* transformers). However, FASCINET would be able to support such components easily if a proper number of SBC designs that include this new component is added to its training set. Also, it is probable that more directives from the user are needed while the rest of the HDNN procedure is the same. The HDNN model is optimized to support current 85 components with the minimum number of training data and directives while maintaining a high accuracy. While the whole HDNN procedure would be the same, adding new components is with cost of having more training data and directives from the user. This leads to have a more complicated model which not only needs more data to be trained, but also needs more runtime.

2.7 Evaluation

2.7.1 Datasheet Scrubber

2.7.1.1 Category Recognizer

To identify the accuracy of the model, a stratified k-fold with 5 splits is used with the labeled categories $[117, 118, 135]$ as the ground-truth. The first 256 words are given as inputs. Using this method, the model has an average accuracy of 95.3% on the test set. Fig. 2.8 shows a confusion matrix of the predicted categories. In order to compare the results, the Bidirectional Encoder Representations from Transformers

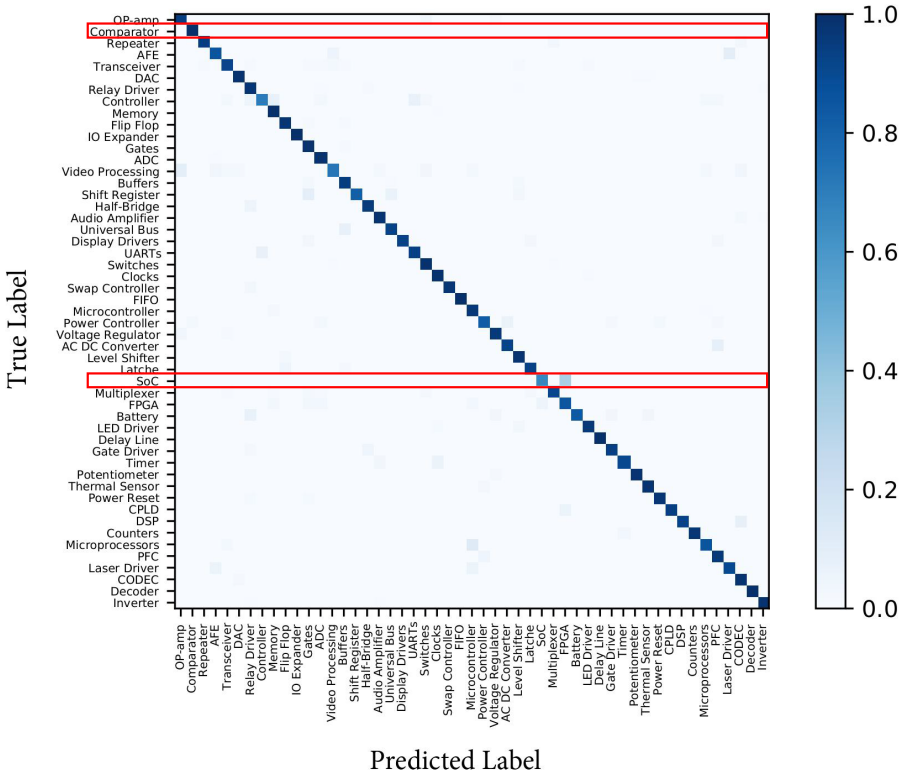


Figure 2.8: The confusion matrix of the category recognizer when the first 256 words are given as inputs. Red rectangles show that, generally, common categories (*e.g.* Comparator) with a larger training set have a higher accuracy in comparison to the niche ones (*e.g.* SoC).

(BERT) text classification [145] is also analyzed on the same training and test sets. For this purpose, "Bert For Sequence Classification" model is used and tested with two test cases. For one test case scenario, the first 128 words are given as inputs with 256 tokens while the other case uses the first 256 words with 360 tokens. They give 92.7% and 92.9% average prediction accuracy respectively, which are less than the proposed CNN method.

Table 2.2 summarizes the model prediction average accuracy and the training time for different number of input words. As you can see, even though the accuracy of 512 words is 0.4% better than the 256 one's, its training time is 1.88x more than the 256 one's. This shows almost no improvement is gained while the cost of computation time has gotten roughly double if the first 512 words are given as inputs instead of the first

Input words number	128	256	360	480	512
Average prediction accuracy	91.73%	95.3%	95.2%	94%	95.7%
Normalized training time	0.28	0.53	0.72	0.94	1

Table 2.2: Average accuracy of the category recognizer and the training time for different number of input words. The training times are normalized to the training time of 512.

Method	Time
Without indexing	0.037s
With indexing	0.006s
With indexing and projection	0.005s
With indexing and aggregation pipelines	0.002s
Without indexing and aggregation pipelines	0.001s

Table 2.3: Comparison of the search optimization methods on the COTS DB.

256 words. Except for the 512 model, the accuracy of the 256 model is better than others. Fig. 2.9 shows the average prediction accuracy with different hyperparameters.

2.7.1.2 Pin Functional Extractor

To test the pin extraction network, the pins from 2,000 datasheets are collected and split 80, 20 into training and testing datasets. The network achieves 96.5% accuracy on the testing dataset, proving its effectiveness.

2.7.2 COTS DB

Table 2.3 compares the search time for different methods. A search is performed on 28 fields of a collection of more than 770,000 datasheets. Implementing indexing optimizes the query search time by approximately 6x, while the aggregation pipeline method achieves a 37x speedup in comparison to a normal search.

2.7.3 Peripheral Circuit Generator

To test the SBC generator tool, 400 SBCs are used that are independent from the training set. For this purpose, the main components of 400 human-designed schematics

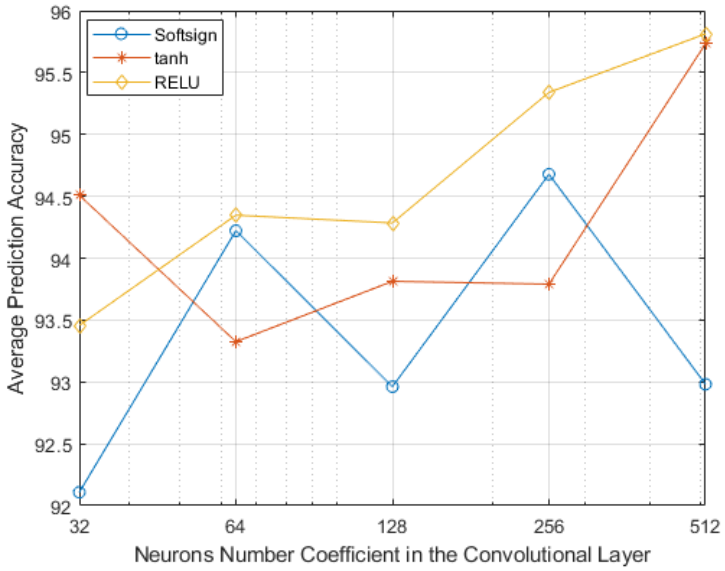


Figure 2.9: Average prediction accuracy comparison of the category recognizer with different hyperparameters used for training. As it was mentioned in Section 2.4.1, there are 5 branches with different neuron numbers that at the end are flattened and concatenated. The base neurons numbers in the convolutional layer of these branches are {1, 2, 2, 2, 4} that the coefficient in the X-axis is multiplied to them. For instance, neuron number coefficient of 32 leads to a network which its branches have {32, 64, 64, 64, 128} neurons in their convolutional layer. The figure shows different activation functions applied to the convolutional layer. The coefficient of 256, *i.e.* {256, 512, 512, 512, 1024} neurons, with the RELU activation function is the simplest architecture that gives a high prediction accuracy of 95.3%. The first 256 words are given as the input.

are given to the FASCINET as input. This allows to fairly compare the generated SBCs by FASCINET with the human-designed ones as ground truth. All experiments are performed on a server with an NVIDIA TITAN V GPU. For giving the main components information, both directly inputting their datasheets, and describing their specifications and searching within the COTS DB have been tested. In the case of searching the specifications of main components, the ability of FASCINET to optimize the total price, power, or meeting the total price/power budget constraints are also tested in addition to satisfying the desired specifications. In such a scenario, FASCINET is limited to components that are within its COTS DB. This means if

	Meet specs	Meet specs & equal power	Meet specs & equal price	Meet specs & better power	Meet specs & better price	Meet specs & meet power/price budget
Total search #*	400	150	150	150	150	100
Successful search #	398	141	135	7	15	100

Table 2.4: Main components search results within the COTS DB. Main components that are used in 400 human-designed SBCs are set as the target of these searches.

* #: Number.

Total components #*	Total main components #	Total pins #	Total connections #	Floating (open) ports #	Short-circuit #	Free DRC error design #	Design number with DRC error
16,600	850	50,000	41,500	750	15	278	122

Table 2.5: DRC results for the proposed peripheral circuit generator; the test set includes 400 SBCs.

* #: Number.

the COTS DB does not contain any component with the exact desired specs, the search would not be successful. However, in this regard, FASCINET finds the most close components within its COTS DB to the requested ones using the feature of prioritizing the specs.

For each minimum total price and minimum total power consumption of main components, 150 searches are performed to find the main components that are used in human-designed SBCs. For the 100 rest, the total power and price of the main components in each of the human-designed SBCs are set as the budget constraints. The found components within the COTS DB are compared manually, both in terms of meeting the specifications and price/power constraints, with the target main components that are used in human designs. The results are summarize in Table 2.4. For instance, if the found main components meet the performance requirement, *i.e.* having the same specifications as the ground-truth main components in the human-designed ones, while having equal and less total power/price, it is counted as successfully "meet performance & equal power/price" and "meet performance & better power/price" respectively. As it is shown in Table 2.4, there are only two unsuccessful searches because, for these two, the COTS DB does not contain any components with

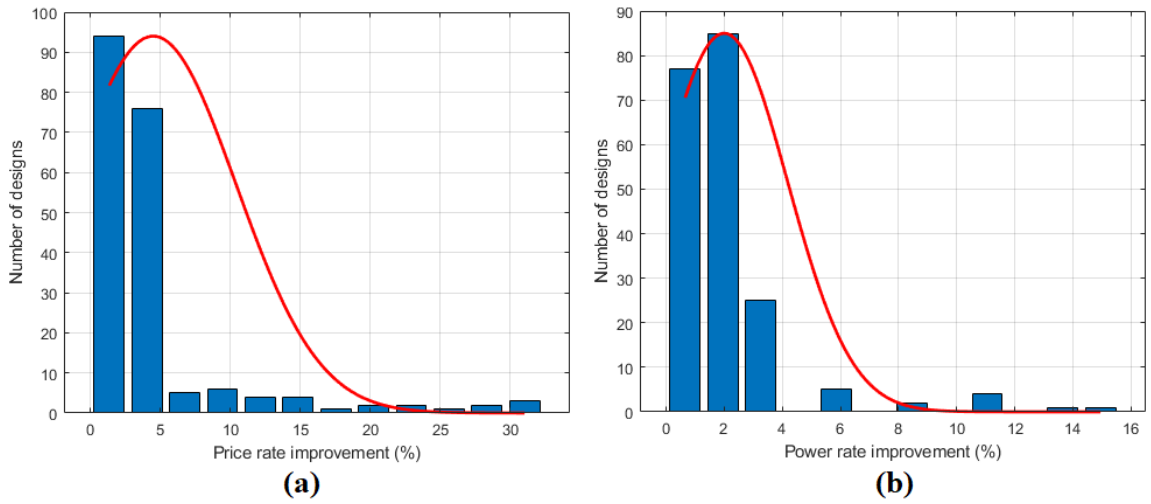


Figure 2.10: The distribution of improvement rate of (a) price and (b) power of found components by the search engine over the human-designed ones. Improvement here means less for both power and price.

the same specifications with the requested ones. However, using prioritizing feature of FASCINET, the most close components within the database is outputted. In order to fairly compare the schematic generation of FASCINET with the human-designed ones, the exact main components that are used in the human-designed schematic are given to FASCINET regardless of the COTS DB output. The distribution of improvement rate of price and power of found components by the search engine over the human-designed ones are depicted in Fig. 2.10.

In addition to testing the satisfaction of main components constrains, all of the generated schematics are evaluated to check whether they are valid. For this purpose, a Design Rule Check (DRC) is run to check any floating (open) port or any short-circuit connections. As the results are summarized in Table 2.5, we can see that less than 1.5% of ports of the components are float. Also, the percent of short-circuit connections are negligible. Even this small error is due to misunderstanding the functionality of the pins, not the SBC peripheral generator itself. Because, as it is discussed in Section 2.6, the HDNN model selects the components around each component among the ones that are connected to that component in the training set. Therefore, it is probable

Identical connections	Precision	Recall	F1-Score
87%	97.9%	96%	96.9%

Table 2.6: Peripheral circuit generator accuracy in comparison with the human-designed SBCs; the test set includes 400 SBCs.

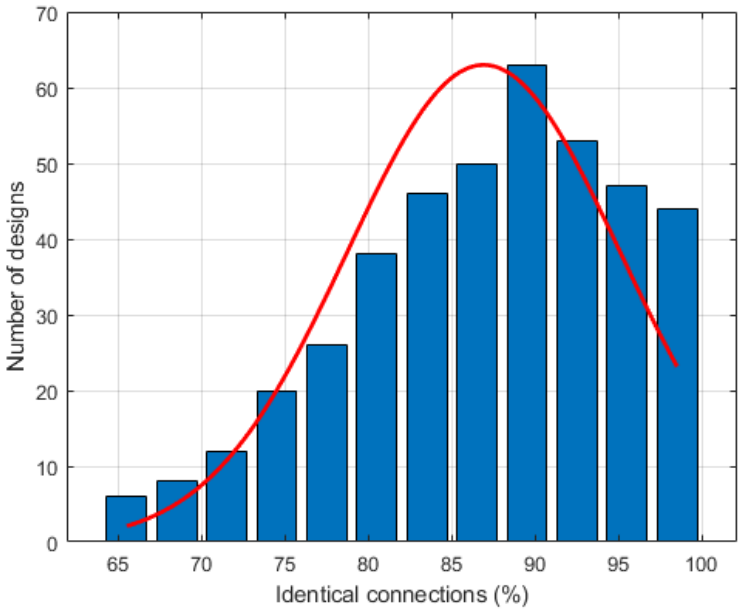


Figure 2.11: The distribution of identical connections over the compared designs.

that the HDNN model does not realize where should it connect the pins to and leaves them float. But, it is impossible to connect them to wrong pins (short-circuit) as there is no sample like that in the training set.

In addition to the DRC check, each individual connection in the schematics designed by FASCINET is compared with those in the human-designed ones, manually as well as by using a netlist comparison tool [146]. Table 2.6 summarizes similarities between the generated peripheral circuits by FASCINET and the human-designed ones. Identical connections are defined as the portion of connections that are completely identical, meaning they are connected to the identical pins of the components which both the components and their values are correctly determined. For calculating the precision, recall and F1-Score, true-positive is defined as the number of connections that are

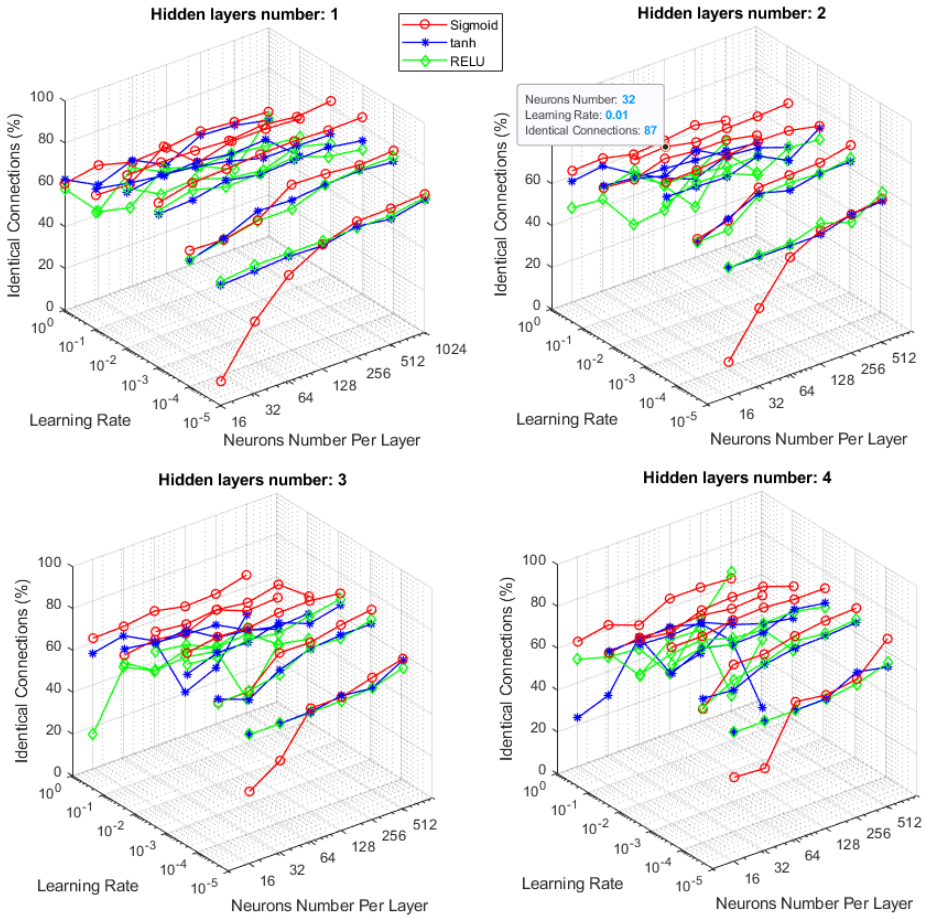


Figure 2.12: Identical connections comparison with different hyperparameters used for training. The activation function for all hidden layers are RELU and only the activation function of the output is changed. As you can see, in general, sigmoid at the output layer gives better results. Two hidden layers with 32 neurons at each layer is the simplest architecture that gives a high identical connections of 87% as it is also shown in Fig. 2.6.

connected to the same component in both tool and human designs without considering the component value. False-positive and false-negative are defined as the number of connections that exist in the tool or human designs, respectively but they do not exist in the other. It is noteworthy that the deviations of FASCINET designs from the human-designed references are mainly due to different design approaches, rather than functionality deficiencies.

Fig. 2.11 demonstrates the distribution of identical connections over the compared

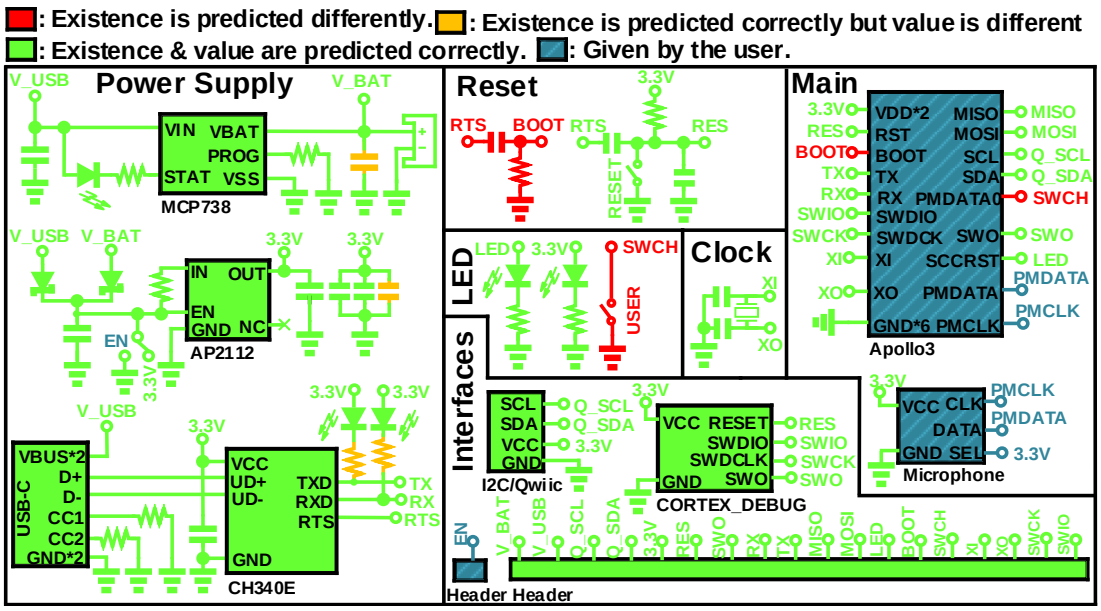


Figure 2.13: An example of a generated SBC schematic by FASCINET. Different colors show a comparison of the similarities between this FASCINET generated SBC and the human-designed one.

designs. The generated SBCs with a lower identical connections usually either contain new main components that have not been observed in the training set, or they have more than 4 main components, or they contain more than 70 components in total which makes them more complicated. It should be noted that in our test set with 400 SBCs, there are 40 designs that contain new main components that have not been observed in the training set. Between such designs, 25 of them contain main components that have the same category, *e.g.* microcontroller, with the main components of some designs in the training set. The average identical connections for these designs is 77.5% while it is 71.1% for the other 15 designs with totally new main components. Fig. 2.12 shows the identical connections while different hyperparameters are applied for training.

The output of FASCINET is an schematic in the industry standard format. Fig. 2.13 depicts an example of such a generated SBC schematic by FASCINET with 40 determined peripheral components and more than 120 connections. The blue hatched

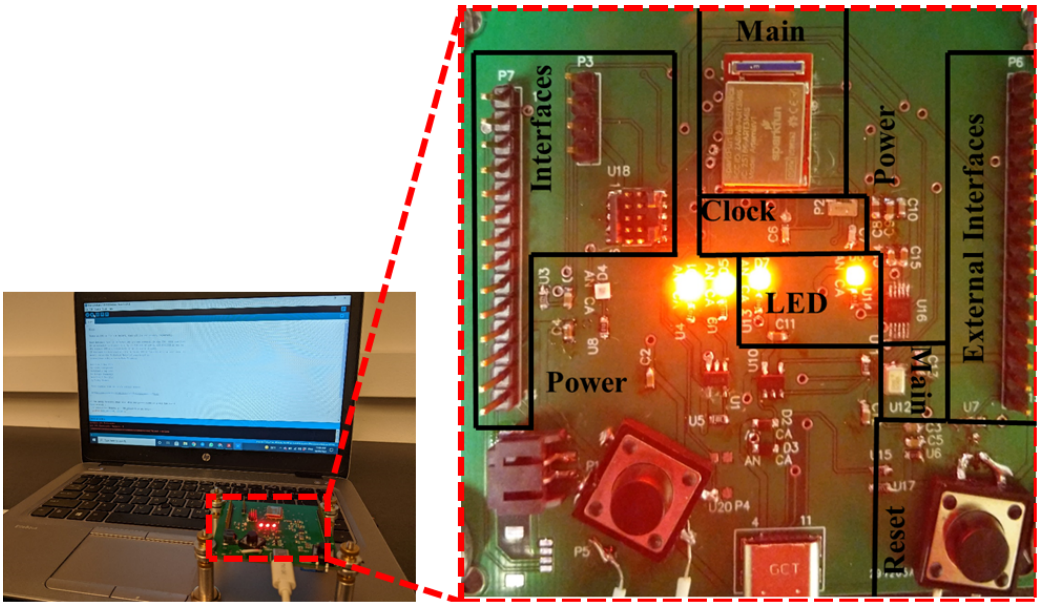


Figure 2.14: The manufactured PCB of the SBC in Fig. 2.13. A simple blinking program is uploaded and run successfully on the SBC.

(main) components and connections are indicated by the user as the input. To obtain their information, both methods of describing specifications and directly inputting their datasheets have been tested. The final generated schematics are completely the same in these two approaches. However, it takes around 2 minutes on average to generate an SBC schematic when the specifications of the main components are inputted by the user, while it is around 5 minutes when a datasheet is inputted directly. It goes without saying that both methods are drastically faster than a human designer. The reason that the second method takes more time is because the datasheet scrubber is used rather than the COTS DB's search engine. In Fig. 2.13, the green color depicts components and connections that both the components and their values are determined correctly by FASCINET, meaning they are the same with the human-designed one. Orange is used to show determined components with differing values. Finally, red is used to show the components and connections that differ from the human design. Red and orange components do not mean that the tool has made an incorrect design. Rather, they just show a difference from how a human designer

would make the circuit.

Fig. 2.14 shows a snapshot of the manufactured PCB of Fig. 2.13. A simple blinking program is uploaded and run successfully on the SBC. The SBC layout is generated by passing the Fig. 2.13 schematic file to an automatic PCB layout generator software [147,148]. For this purpose, the automatic routing feature by specifying more width for power traces than normal signals using the directives of the layout generator software is used while the placement has been done manually.

2.8 Conclusion

This work presents a novel automatic SBC generator tool, FASCINET, which uses the HDNN-based approach for deciding the required peripheral components, determining their values and connections. As a part of this work, to build an efficient database of reusable components, a fully automated approach to extract the functionality and specifications of components from datasheets is proposed. The automatic category recognizer is trained and tested with more than 770,000 PDF documents within the COTS DB. It achieves an accuracy greater than **95%**. The table extractor achieves a precision greater than **96%** on a randomly selected testing set. The automated SBC generator tool is validated by comparing the netlist of the generated SBCs with more than 400 arbitrary human-designed ones. The required time for an SBC generation is reduced to **2-5 minutes** on average using FASCINET. The PCB layout of a sample SBC designed by FASCINET is also successfully generated.

CHAPTER III

Tablext: A Combined Neural Network And Heuristic Based Table Extractor

A significant portion of the data available today is found within tables. Therefore, it is necessary to use automated table extraction to obtain thorough results when data-mining. Today's popular state-of-the-art methods for table extraction struggle to adequately extract tables with machine-readable text and structural data. To make matters worse, many tables do not have machine-readable data, such as tables saved as images, making most extraction methods completely ineffective. In order to address these issues, a novel, open-source, general format table extractor tool, Tablext, is proposed. This tool uses a combination of computer vision techniques and machine learning methods to efficiently and effectively identify and extract data from tables.

Tablext begins by using a custom Convolutional Neural Network (CNN) to identify and separate all potential tables. The identification process is optimized by combining the custom CNN with the YOLO object detection network. Then, the high-level structure of each table is identified with computer vision methods. This high-level, structural meta-data is used by another CNN to identify exact cell locations. As a final step, Optical Characters Recognition (OCR) is performed on every individual cell to extract their content without needing machine-readable text. This multi-stage algorithm allows for the neural networks to focus on completing complex tasks, while

letting image processing methods efficiently complete the simpler ones. This leads to the proposed approach to be general-purpose enough to handle a large batch of tables regardless of their internal encodings or their layout complexity. Additionally, it becomes accurate enough to outperform competing state-of-the-art table extractors on the ICDAR 2013 table dataset.

Tables are widely used in documents, articles, web-pages, etc. as they can concisely show complex information in a way that is suitable for human readers [149]. Autonomous table extraction enables translating this abundant amount of information to a machine-readable format which has broad applications in data-mining and information-retrieval [150,151]. The complexity of some tables' layouts and the various formats in which they may be found are current problems that prevent the existence of a universal, highly-accurate, automatic table extraction tool. The purpose of this project is to introduce a highly accurate extractor that can extract previously unobtainable information.

Formats such as Hypertext Markup Language (HTML), Portable Document Format (PDF), and Portable Network Graphics (PNG) are common encodings that may contain tables. Because of the many different ways to represent a table, designing separate extraction methods for every encoding is time-consuming and prone to errors. Moreover, an environment may have multiple formats that are not encoded uniformly. For instance, a web-page may contain an image that is not encoded in HTML. Unifying internal encodings of files is the first step to overcome universality issues.

In this work, we present an open-source[1] table extractor tool, Tablext, which addresses all the above-mentioned problems. Tablext solves the universality issue by extracting data from images. There are several Java and Python open-source libraries to convert specific file formats to images such as pdf2image [152] (PDF to PNG), GrabzIt [153] (HTML to PNG), etc.

[1]Source code for the Tablext can be downloaded from `https://github.com/idea-fasoc/datasheet-scrubber/tree/master/src/table_extraction`

Tablext uses separate Convolutional Neural Networks (CNNs) [115,130] to identify and extract information from tables, but it does not solely rely on them. Before data extraction, Tablext uses common conventional computer vision techniques, such as line identification, to extract positional information from tables before another CNN is used to fix small mistakes. This approach can be both faster and more accurate than having a single neural network.

By first identifying important features within the tables, each neural network only has to solve a small sub-task. This means that the size required for each network is drastically reduced. The smaller size of the neural networks reduces the computation requirement as well as the risk of overfitting.

Tablext begins by using an efficient CNN to identify the exact location of the tables within the input images. Using an ensemble of this CNN and an object detection network, YOLO [154], allows for the extraction algorithm to be confined to relevant areas. By isolating the relatively expensive extraction algorithm to areas found by the identification CNN, Tablext is efficient in handling massive databases.

Tablext achieves a high extraction accuracy by both identifying the lines separating various cells, as well as looking at the positions of the text within the tables. Together, these methods can recognize complex table structures. It can generally be assumed that the identified lines correctly split the table into cells, but when no lines exists or they cannot be correctly identified, the positional data is used to separate the cells. After correctly segmenting the table into cells, the data is cleaned by a CNN before the open-source Optical Character Recognition (OCR) software, Tesseract [134], identifies the text in every individual cell.

The main contributions of this project can be summarized as follows.

- A new method to identify cells that utilizes the positions of the data and lines in order to obtain high precision preliminary results at a low computational cost.
- A hybrid pipeline that allows the neural network to focus its attention on complex

problems, resulting in a high accuracy.

- A novel, general format, open-source tool that identifies and extracts all complex layout tables with a high accuracy within various internal encodings files.

3.1 Related Work

Several methods have been studied table identification and extraction. However, some of them describe various approaches to identify tables from different types of documents and do not focus on data extraction [155–161].

Several systems have been proposed for the identification and extraction of tables within PDF files [150, 162–165]. Oro and Ruffolo [150] find positional data for the text and they use an agglomerative hierarchical clustering algorithm [166] to combine words into lines. The main limitation of this approach is that it only supports unlocked, machine-readable PDFs. However, a large amount of tables are found in locked PDFs or images. Perez-Arriaga *et al.* [162] find columns and rows by comparing the locations of the text-boxes and combining text-boxes below a distance threshold. Although the method has a decent recall, its false-positive rate is high. Liu *et al.* [165] propose a method in which, tables are found by grouping text with similar positions and font size. This approach works with HTML format in addition to PDF files. However, it is only optimized for research papers and its extraction performance is poor especially on general documents. Similar to the other mentioned studies, this method cannot work with locked PDF files or images.

Some other studies focus on table extraction that are not related to PDFs [149, 151, 167]. Tengli *et al.* [149] propose a machine-learning-based technique for table extraction from HTML files. Nishida *et al.* [167] focus on HTML extraction too. They use a Recurrent Nerual Network (RNN) model to extract data from HTML tags. These such approaches are not applicable to formats other than HTML and

are therefore, limited in scope. Pinto *et al.* [151] propose a method to identify the horizontal lines within tables that are formatted using white space and fixed fonts to align columns in Web pages, however they do not concentrate on table extraction.

Koci *et al.* and Puha *et al.* [168, 169] propose methods that can handle tables in more diverse formats. Koci *et al.* [168] propose a graph representation of tables so that they can use heuristics approaches in order to extract information. Their proposed methodology is optimized to extract data from various spreadsheets. Puha *et al.* [169] propose a heuristic method for general table extraction. They recognize tables in images by first using OCR to identify data within cells, then using the locations of the data.

Many state-of-the-art general table extraction methods solely use neural networks. Qasim *et al.* [170] use a Graph Neural Network (GNN) to identify cell locations within tables. DeepDeSRT [171] uses separate, specially made neural networks to identify tables and extract their data, while TableNet [172] uses a single network for both identification and extraction. They argue this is an efficient approach since these tasks are interdependent. However, this also means that the network is less optimized for both tasks. Tablext outperformed both of these methods when tested on a common dataset. A comparison between the related works has been summarized in Table 3.1.

By utilizing Machine Learning (ML), modern table extractors can handle more diverse tables than earlier works. However, identifying and extracting data is a difficult undertaking that requires many sub-tasks to be completed. Many of these tasks, such as identifying lines that can make up a table's structure, are good candidates for conventional computer vision methods. Instead of forcing a neural network to learn and solve every one of these tasks, Tablext introduces a new approach that combines both conventional computer vision and ML techniques. This allows the neural network to be smaller in size and focused on its particular task of identifying abnormal cells. This is especially important given the limited amount of labeled training data available

Work	Table identification	Table extraction	File format	Approach
[155, 157, 161]	✓	✗	All	Image processing w/o ML
[156]	✓	✗	PDF	PdfExtra paradigm
[158]	✓	✗	All	DNN
[159]	✓	✗	PDF	CNN
[160]	✓	✗	All	SVM
[149]	✗	✓	HTML	ML
[167]	✗	✓	HTML	RNN
[168]	✗	✓	Spreadsheets	Graph representation
[169]	✗	✓	All	Image processing w/o ML
[170]	✗	✓	All	GNN
[150]	✓	✓	PDF	Agglomerative hierarchical clustering algorithm
[162]	✓	✓	PDF	Comparing the locations of the text-boxes
[165]	✓	✓	PDF & HTML	Grouping text
[171]	✓	✓	All	NN
[172]	✓	✓	All	NN
Tablext	✓	✓	All	Heuristic-based + CNN

Table 3.1: Comparison of related works.

for tables.

3.2 Table Location Identification

While the main focus of this paper is table extraction, a novel algorithm is created and used to identify tables before extraction. It is imperative to find the table locations before extraction because doing so increases both the performance and accuracy of the algorithm. The boost in performance comes from reduced search space for the complex information extraction. The increased accuracy is a result of the table identification

network separating tables from one another and removing irrelevant text and lines that could potentially confuse the table extractor. To further increase the accuracy of the table identification, it is later combined with YOLO.

The identification algorithm needs to find both the minimum and maximum X and Y coordinates for each table that exists within each page. To efficiently calculate these coordinates, two relatively simple CNNs are used in succession. These two networks identify at which Y and X coordinates, respectively, a table exists.

All input images, after being converted from any file format, are first converted to gray-scale and resized. Preliminary tests show that a width of 800 pixels allows for accurate table extraction without a significant performance penalty. The height at this stage is scaled proportionally to avoid distortion. This resized image cannot directly be sent to a conventional CNN because its height is variable. Additionally, the large size of the input data would require a significant amount of memory for each batch, slowing down the training process. In order to reduce the input size and make the input regular, the image is sliced into horizontal strips, each with a height of 64 pixels. This number is chosen to reduce the training time without significantly degrading accuracy.

While cutting the image into strips reduces the input and output size of the network, this process introduces a new problem. The accuracy of identifying a table at the top and the bottom of the slices is lower than identifying a table near the middle. Intuitively this is because the network only has information in one direction from these edge outputs. To combat this issue, each strip only detects the existence of a table within its innermost 32 rows of pixels. To cover the entire table, a moving window approach is used where each input strip partially overlaps its adjacent strips.

The neural network has 4 output nodes for every horizontal strip. Each node determines if there is a table at a particular location within the centermost 32 pixels. In the innermost 32 pixels, node $i, 0 \leq i \leq 3$, of the neural network is trained to

identify if a table exists between any of the $[8 * i, 8 * i + 7]$ pixels. The outputs of all the slices are concatenated and stored in an array that identifies in which rows a table exists. With this approach, the top and bottom of each page cannot be searched for tables, because only the inner pixels are represented in the outputs. To fix this problem, the input image is extended by 16 pixels at the top and bottom with a white border.

Fig. 3.1 shows the concatenated output of all the slices from the neural network. In this figure, the thin horizontal boxes represent every row the network checked for a table. These boxes show the Y locations where the neural network predicts a table exists. While this particular example shows a table that takes up the entire width of the document, this network can also identify tables that are within one of several columns on the page.

The next step is to clean the output by concatenating rows of positively identified tables into groups. If no groups are created, because none of the rows contain a table, Tablext outputs that no table is found in the image. Otherwise, the groups are prepared and sent to the second neural network in order to find the correct X coordinates. This process is simpler than having to find the correct Y coordinates as the amount of area to search is smaller. For this reason, a CNN is used without a sliding window approach.

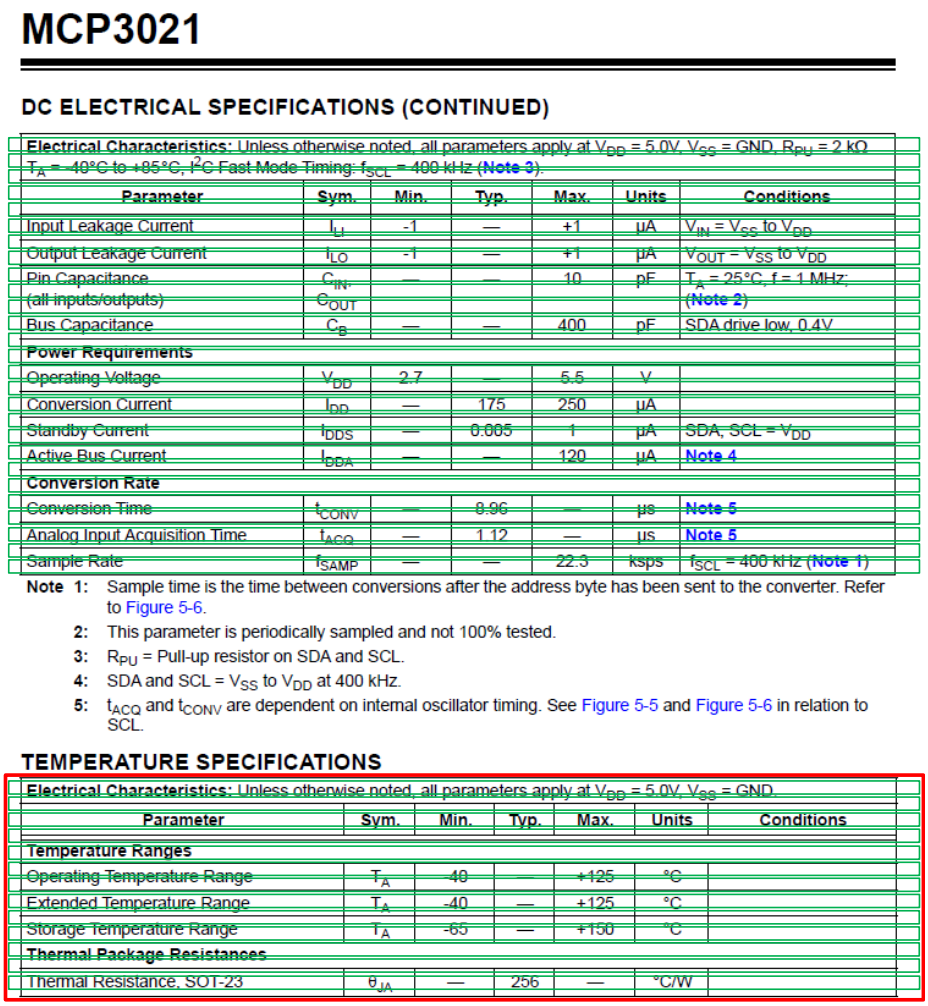
Each group of rows is resized into a 400×400 pixel square to efficiently achieve high accuracy. This input size is obtained by preliminary testing. The image is sent into a CNN that identifies which columns contain a table. Fig. 3.2 shows this technique being applied to the bottom group of rows in Fig. 3.1. The actual result produced by the network looks distorted to a human because the table is scaled into a square. Instead a representative, recreated image is shown. Here the vertical boxes show where the network predicts that a table exists. If the table is only in a single column of a multi-column image, the lines would cover only that section. This network can also

identify two separate adjacent tables. This is achieved by not identifying a table in between the two table groups. Each vertical group is separately sent to this second stage. As an example, Fig. 3.1 would send three separate images into the second network.

The columns are concatenated into groups, in a similar manner to the rows. The scaling that converted the group of rows into the square, shown in Fig. 3.2, is inverted and applied to the X coordinates. Additionally, the scaling used to convert the original image into the resized image, shown in Fig. 3.1, is inverted and applied to the Y coordinates. Doing this gives accurate coordinates of each table's start and end position regarding the original image. Fig. 3.3 shows a cutout of the first table, using the found X and Y coordinates.

The CNN by itself tends to over propose table regions because the existence of lines on the page are very influential to the model. This causes the CNN to predict areas around these lines as tables whether these lines correspond to tables or not. This will cause the extraction portion of the algorithm to take more time as it has to run OCR on these extra regions to figure out they are not tables. To counteract this, a YOLO [154] network is trained on table images to see if this more popular object detection method could outperform the CNN. YOLO typically identifies all the right table regions, but the bounding boxes that it proposes do not cover the entire table. For this reason, an ensemble of the CNN and YOLO is used to create an optimal identification model. Both networks first independently propose table regions. When these models propose regions that overlap, the union of these two regions is taken. This ensures both that the final regions are likely to contain tables and that the regions will likely contain the entirety of the tables.

While this approach can perfectly identify tables in multiple columns, there is an extremely rare case where two adjacent tables in separate columns have vastly different heights. In this case, extra data could be identified as a table. This rare



MCP3021

DC ELECTRICAL SPECIFICATIONS (CONTINUED)

Electrical Characteristics: Unless otherwise noted, all parameters apply at V_{DD} = 5.0V, V_{SS} = GND, R_{PU} = 2 kΩ, T_A = -40°C to +85°C, I^2C Fast Mode Timing: f_{SCL} = 400 kHz (Note 3).

Parameter	Sym.	Min.	Typ.	Max.	Units	Conditions
Input Leakage Current	I_{LI}	-1	—	+1	µA	V_{IN} = V_{SS} to V_{DD}
Output Leakage Current	I_{LO}	-1	—	+1	µA	V_{OUT} = V_{SS} to V_{DD}
Pin Capacitance (all inputs/outputs)	C_{IN}, C_{OUT}	—	—	10	pF	T_A = 25°C, f = 1 MHz; (Note 2)
Bus Capacitance	C_B	—	—	400	pF	SDA drive low, 0.4V
Power Requirements						
Operating Voltage	V_{DD}	2.7	—	5.5	V	
Conversion Current	I_{DD}	—	175	250	µA	
Standby Current	I_{DDS}	—	0.005	1	µA	SDA, SCL = V_{DD}
Active Bus Current	I_{DDA}	—	—	120	µA	Note 4
Conversion Rate						
Conversion Time	t_{CONV}	—	9.96	—	µs	Note 5
Analog Input Acquisition Time	t_{ACQ}	—	1.12	—	µs	Note 5
Sample Rate	f_{SAMP}	—	—	22.3	ksps	f_{SCL} = 400 kHz (Note 1)

Note 1: Sample time is the time between conversions after the address byte has been sent to the converter. Refer to Figure 5-6.

2: This parameter is periodically sampled and not 100% tested.

3: R_{PU} = Pull-up resistor on SDA and SCL.

4: SDA and SCL = V_{SS} to V_{DD} at 400 kHz.

5: t_{ACQ} and t_{CONV} are dependent on internal oscillator timing. See Figure 5-5 and Figure 5-6 in relation to SCL.

TEMPERATURE SPECIFICATIONS

Electrical Characteristics: Unless otherwise noted, all parameters apply at V_{DD} = 5.0V, V_{SS} = GND.

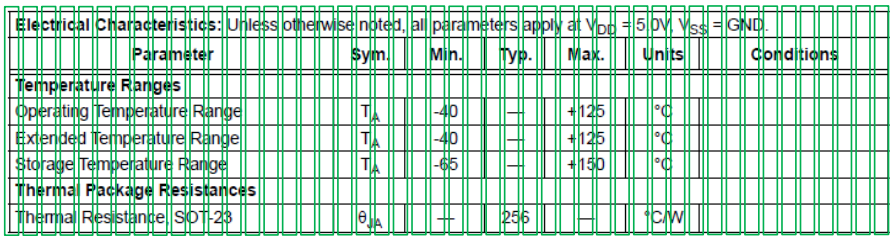
Parameter	Sym.	Min.	Typ.	Max.	Units	Conditions
Temperature Ranges						
Operating Temperature Range	T_A	-40	—	+125	°C	
Extended Temperature Range	T_A	-40	—	+125	°C	
Storage Temperature Range	T_A	-65	—	+150	°C	
Thermal Package Resistances						
Thermal Resistance, SOT-23	θ_{JA}	—	256	—	°C/W	

Figure 3.1: (Table A) Each horizontal boxes represents a row where a table is identified. The bottom group is highlighted with a thick box (red) for clarification.

issue is rectified later because the proposed extraction method can tolerate some extra area identified as a table.

3.3 Table Extraction

After all of the tables are identified, the extraction method starts independently on each table. In order to process high resolution images efficiently, a resized copy of each individual table is created with a width of 800 pixels and a proportional height. This copy is used for a majority of the cell identification steps, however the original



Electrical Characteristics: Unless otherwise noted, all parameters apply at V_{DD} = 5.0V, V_{SS} = GND.						
Parameter	Sym.	Min.	Typ.	Max.	Units	Conditions
Temperature Ranges						
Operating Temperature Range	T_A	-40	—	+125	°C	
Extended Temperature Range	T_A	-40	—	+125	°C	
Storage Temperature Range	T_A	-65	—	+150	°C	
Thermal Package Resistances						
Thermal Resistance, SOT-23	θ_{JA}	—	256	—	°C/W	

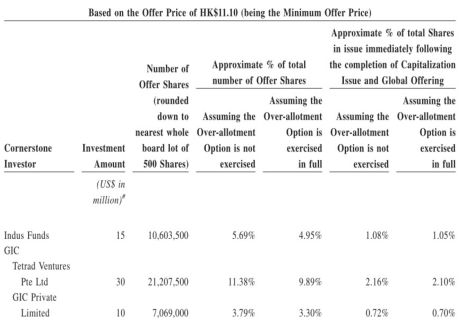
Figure 3.2: (Table A) Vertical boxes represent every column where a table is identified.

Electrical Characteristics: Unless otherwise noted, all parameters apply at V_{DD} = 5.0V, V_{SS} = GND.						
Parameter	**Sym.**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
Temperature Ranges						
Operating Temperature Range	T_A	-40	—	+125	°C	
Extended Temperature Range	T_A	-40	—	+125	°C	
Storage Temperature Range	T_A	-65	—	+150	°C	
Thermal Package Resistances						
Thermal Resistance, SOT-23	θ_{JA}	—	256	—	°C/W	

Figure 3.3: (Table A) A cutout of the original table using the coordinates identified.

image is kept because it is used later during OCR.

Several methods are used to identify and isolate the cells within each table. The table extraction begins by identifying the high-level, general structure of the table. It identifies the vertical and horizontal lines, both visible and implied, that correctly divide a majority of the cells. There are several ways to locate this information within the tables. One approach is to identify the visible lines separating various cells. Another method is to look at the positions of the text within the tables and extrapolate how the cells should be positioned in accordance to each other. Tablext uses both of these approaches in order to obtain the structure of the table. Afterwards, a neural network is used to clean up any cells that do not conform to the general structure. Fig. 3.4 shows a table that already went through table identification. This will be used as an example to explain the steps of the table extraction.



Based on the Offer Price of HK$11.10 (being the Minimum Offer Price)						
			Approximate % of total number of Offer Shares		Approximate % of total Shares in issue immediately following the completion of Capitalization Issue and Global Offering	
Cornerstone Investor	Investment Amount	Number of Offer Shares (rounded down to nearest whole board lot of 500 Shares)	Assuming the Over-allotment Option is not exercised	Assuming the Over-allotment Option is exercised in full	Assuming the Over-allotment Option is not exercised	Assuming the Over-allotment Option is exercised in full
	(US$ in million)#					
Indus Funds	15	10,603,500	5.69%	4.95%	1.08%	1.05%
GIC Tetrad Ventures Pte Ltd	30	21,207,500	11.38%	9.89%	2.16%	2.10%
GIC Private Limited	10	7,069,000	3.79%	3.30%	0.72%	0.70%

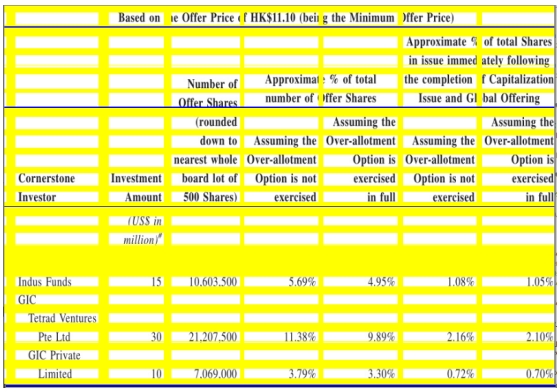
Figure 3.4: (Table B) A table sample identified by the proposed table identification method.

3.3.1 Real Line Identification

Vertical and horizontal lines are commonly used to separate and organize the contents of tables. To identify these lines, the second-derivative gradient field is approximated for the gray-scale image of the table. Assuming that $f(x, y)$ represents the brightness of a pixel at the location (x, y). The values of $\frac{\partial^2}{\partial x^2} f(x, y)$ and $\frac{\partial^2}{\partial y^2} f(x, y)$ are approximated for each pixel by looking at their adjacent pixels and the gradients of them. These gradients are then stored into separate matrices. Then a 3×3 max pooling is independently applied to these matrices, to account for noise in the image and imperfect lines. The max pooling has a stride of 1 in both X and Y directions to retain as much precision as possible when identifying the lines.

With this information, the vertical lines are found by searching for vertical segments where all the pixels in the segment have high $\frac{\partial^2}{\partial x^2} f(x, y)$ values relative to the pixels in rest of the image and similar $\frac{\partial^2}{\partial y^2} f(x, y)$ values to one another. Similarly, horizontal lines are found by searching for horizontal segments where all the pixels in the segment have high $\frac{\partial^2}{\partial y^2} f(x, y)$ values relative to the pixels in rest of the image and similar $\frac{\partial^2}{\partial x^2} f(x, y)$ values to one another.

The segment length has to be a small fraction of the table's height or width for the vertical and horizontal lines respectively. However, once a line is found, that line is extended to cover the entire width or height of the table. Identifying lines in



	Based on	e Offer Price (	f HK$11.10 (bei	g the Minimum	Offer Price)	
					Approximate %	of total Shares
					in issue immed	ately following
		Number of	Approximat	% of total	the completion	f Capitalization
		Offer Shares	number of	Offer Shares	Issue and GI	bal Offering
		(rounded		Assuming the		Assuming the
		down to	Assuming the	Over-allotment	Assuming the	Over-allotment
		nearest whole	Over-allotment	Option is	Over-allotment	Option is
Cornerstone	Investment	board lot of	Option is not	exercised	Option is not	exercised
Investor	Amount	500 Shares)	exercised	in full	exercised	in full
	(US$ in million)					
Indus Funds	15	10,603,500	5.69%	4.95%	1.08%	1.05%
GIC						
Tetrad Ventures						
Pte Ltd	30	21,207,500	11.38%	9.89%	2.16%	2.10%
GIC Private						
Limited	10	7,069,000	3.79%	3.30%	0.72%	0.70%

Figure 3.5: (Table B) Inferred lines shown with thick bright (yellow) lines and real lines shown with thin dark (blue) lines.

this way helps define the structure of the table, but some cells that do not conform to the general format might be split by lines that do not cover the entire width or height of the table. This issue is solved later with a neural network that stitches together improperly cut cells. Fig. 3.5 depicts a debug image, created by the real line identification algorithm, that shows the few real lines in dark blue. The rest of the lines in this figure are created with another process that will be explained in the following section.

3.3.2 Inferred Line Identification

Real line identification alone is sufficient for regular tables that have all of their data properly split into cells with defined borders. However, as Fig. 3.4 shows, many tables only have proper borders for some of the cells or do not have any well defined lines at all. To correctly identify cells that real line identification alone could not, Tablext uses the positions of the data. Finding the locations of the data is trivial. All of the high contrast pixels, not already identified as a real line, belong to the data. With the locations of the data known, Tablext attempts to segment the pieces of data into columns and rows. To achieve this, Tablext creates straight lines that do not intersect the data. These such lines will be referred to “inferred lines” as the table

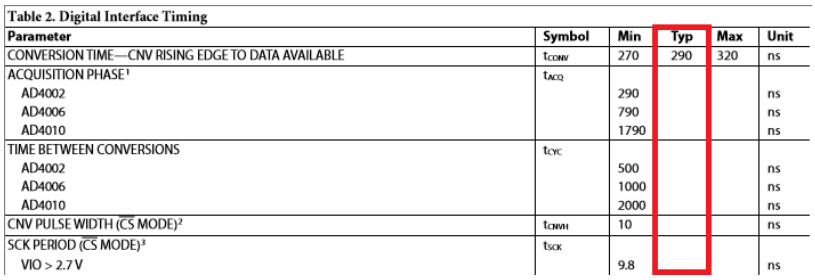


Table 2. Digital Interface Timing					
Parameter	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
CONVERSION TIME—CNV RISING EDGE TO DATA AVAILABLE	t_{CONV}	270	290	320	ns
ACQUISITION PHASE[1]	t_{ACQ}				
AD4002		290			ns
AD4006		790			ns
AD4010		1790			ns
TIME BETWEEN CONVERSIONS	t_{CYC}				
AD4002		500			ns
AD4006		1000			ns
AD4010		2000			ns
CNV PULSE WIDTH ($\overline{CS}$ MODE)[2]	t_{CNVH}	10			ns
SCK PERIOD ($\overline{CS}$ MODE)[3]	t_{SCK}				
VIO > 2.7 V		9.8			ns

Figure 3.6: (Table C) Initial sparse table before inferred vertical lines. A red box has been placed around the, difficult to handle, referenced sparse column for clarity.

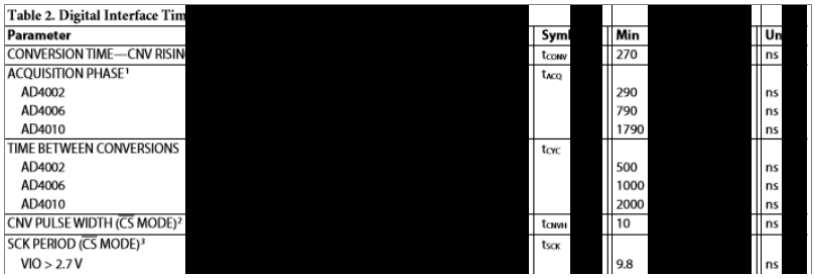


Table 2. Digital Interface Tim			
Parameter	**Syml**	**Min**	**Un**
CONVERSION TIME—CNV RISIN	t_{CONV}	270	ns
ACQUISITION PHASE[1]	t_{ACQ}		
AD4002		290	ns
AD4006		790	ns
AD4010		1790	ns
TIME BETWEEN CONVERSIONS	t_{CYC}		
AD4002		500	ns
AD4006		1000	ns
AD4010		2000	ns
CNV PULSE WIDTH ($\overline{CS}$ MODE)[2]	t_{CNVH}	10	ns
SCK PERIOD ($\overline{CS}$ MODE)[3]	t_{SCK}		
VIO > 2.7 V		9.8	ns

Figure 3.7: (Table C) Sparse table with initialized inferred vertical lines.

requires these non-existent lines to properly convey its information. Fig. 3.5 shows when all inferred and real lines of Fig. 3.4 are drawn.

The task of finding inferred lines is made harder by the existence of cells that do not conform to the general table's structure. In Fig. 3.4, multiple cells span two columns. If inferred lines are defined as lines that do not overlap any pieces of data, the columns underneath the cells that span multiple columns would not be separated. In order to solve this issue, Tablext starts by defining the term "threshold distance".

For vertical lines, the threshold distance is the proportion of the table's height an inferred line must not touch any data in order to be considered valid. If the table is dense, meaning that most cells contain data, a small, static threshold distance would produce accurate results. However if the table is sparse, meaning that many cells do not contain data, this approach would not work. A small, static threshold distance would combine adjacent empty cells, into wide inferred lines. Fig. 3.6 and Fig. 3.7 show the inferred vertical lines initialization. Fig. 3.7 illustrates that a few pieces of

data within these columns or rows could be overwritten by the inferred lines.

To accommodate both sparse and dense tables, the threshold distance is adaptable. Algorithm 1 describes a simplified procedure of acquiring the final inferred lines with a variable threshold distance. In the algorithm, possible inferred lines refers to inferred lines found at the given threshold distance, Δ is a small learned constant, and number of groups refers to the amount of separable groups of inferred lines that exist within the possible inferred lines.

Algorithm 1: Setting the threshold distance

```
1: Variables: num_group1, num_group2, threshold_distance, temp_lines
2: Output: final_lines
3: Start:
    4: threshold distance = 0.6; num_group1 = 0; final_lines = [];
    5: Loop:
        6: temp_lines = possible inferred lines;
        7: num_group2 = number of groups;
        8: if num_group2 ≥ num_group1:
            9: num_group1 = num_group2;
            10: final_lines = temp_lines;
            11: threshold_distance += Δ;
            12: go to Loop;
        13: return final_lines;
```

For a visual representation of this algorithm, images of this process are produced by the code. Fig. 3.8 shows the final output after applying this algorithm to the sparse table in Fig. 3.7. At the start, the vertical inferred lines, represented as black columns, completely cover the column beginning with "Typ". If the threshold distance was a low static number, this column would be merged with one of its neighboring columns. However, by using Algorithm 1, the lines, over several iterations, split around the "Typ" column as shown by Fig. 3.8.

During this process, every inferred line is assigned a "quality score". The quality score is the percentage of the maximum possible line length that a particular inferred line covers without intersecting a piece of data. These quality scores are used later for combining the various lines.

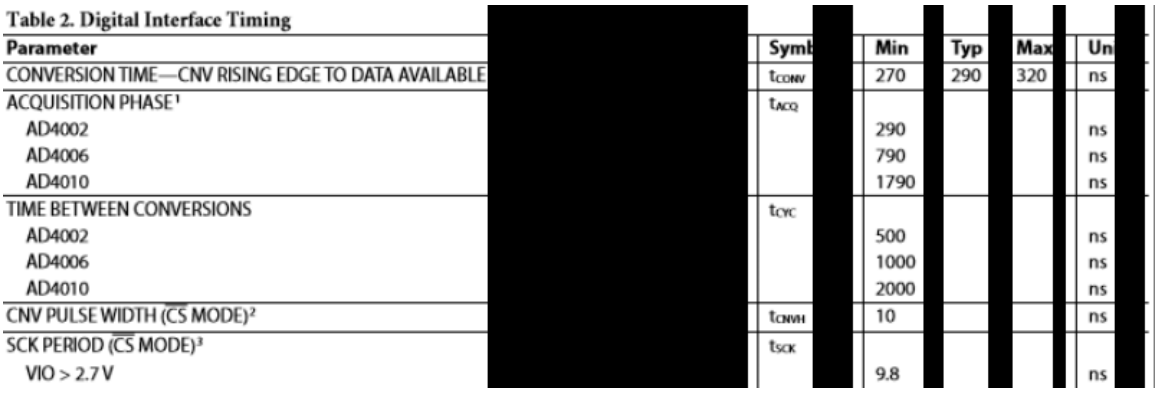


Table 2. Digital Interface Timing

Parameter	Symb	Min	Typ	Max	Un
CONVERSION TIME—CNV RISING EDGE TO DATA AVAILABLE	t_{CONV}	270	290	320	ns
ACQUISITION PHASE[1]	t_{ACQ}				
AD4002		290			ns
AD4006		790			ns
AD4010		1790			ns
TIME BETWEEN CONVERSIONS	t_{CYC}				
AD4002		500			ns
AD4006		1000			ns
AD4010		2000			ns
CNV PULSE WIDTH ($\overline{CS}$ MODE)[2]	t_{CNVH}	10			ns
SCK PERIOD ($\overline{CS}$ MODE)[3]	t_{SCK}				
VIO > 2.7 V		9.8			ns

Figure 3.8: (Table C) Sparse table with incremented inferred vertical lines.

Inferred horizontal lines are created in a similar manner as inferred vertical lines, except the threshold distance is a percentage of the table's width instead of the table's height. To better show this complete process, both vertical and horizontal inferred line identification are used on Fig. 3.4. The resulting inferred lines are used in conjunction with previously calculated real lines to create the debug image shown in Fig. 3.5.

3.3.3 Final Structural Line Identification

Now that all of the real and inferred lines have been located, Tablext begins to group all of this information so that it has a high level understanding of the particular table's layout. Inferred lines are combined with their neighbors, if they are within close proximity, to create different groups of inferred lines. If a real line is within a group of inferred lines, or it is only several pixels away from a group, only the real line is kept. The real lines are considered to be the ground truth, so nearby inferred line groups are redundant. If there are no real lines near an inferred line group, the quality scores of the inferred lines are used to determine which line in the group is the best line to split the cells. A Simple Moving Average (SMA) of the quality scores, looking both forwards and backwards two pixels, is calculated for each group. Lines that do not have a valid inferred line are given a zero quality score. Using an SMA helps reduce the impact of noise in the image, while also fixing corner cases where a

single pixel thick inferred line might sneak between characters in a piece of data.

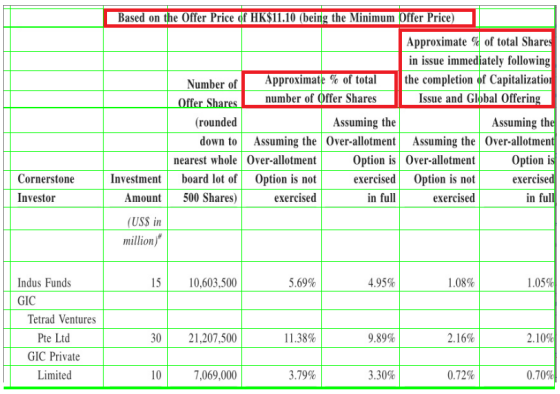
The index that contains the maximum SMA value is taken for each inferred group which is not near a real line. Then, the locations of the real lines and the maximum SMA indexes are concatenated together. These combined lines are the final lines for the high level table identification. This process is completed independently for both the vertical and horizontal lines. Fig. 3.9 is created by applying this process on the lines found in Fig. 3.5.

3.3.4 Neural Network Correction

Since the final lines describe the regular structure of the table, cells that do not conform to that structure might be sliced into two or more pieces. Fig. 3.9 shows several cells that are cut in half by the final lines. In order to fix this issue, a CNN is used to recombine improperly sliced cells. A simpler solution would be to check if any high contrast pixels exist underneath an inferred line. However, this method has several problems. One issue is if characters in a cell are spaced out significantly, a vertical inferred line could exist in between two characters and the cell's contents would be split into two cells.

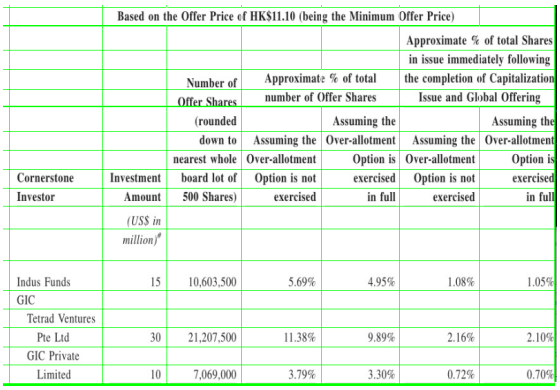
While these concerns can be partially alleviated by looking a certain distance to the left and right for high contrast pixels, this introduces a new risk of merging cells unnecessarily. Additionally, merging with that method can be triggered by noise in the tables. For these reasons, a neural network has been created to handle the possible merging of cells.

The CNN takes in two adjacent cells as inputs and resizes them to be 100×100 pixels each. Testing shows that this size allows the OCR to accurately extract data. The cells are resized independently so that the line that divides them is always in the center of the two cells. If instead, the cells are merged and then resized, the CNN would need to predict which data belongs to which cell or it would require the location



	Based on the Offer Price of HK$11.10 (being the Minimum Offer Price)					
					Approximate % of total Shares in issue immediately following the completion of Capitalization Issue and Global Offering	
		Number of Offer Shares (rounded down to nearest whole board lot of 500 Shares)	Approximate % of total number of Offer Shares			
			Assuming the Over-allotment Option is not exercised	Assuming the Over-allotment Option is exercised in full	Assuming the Over-allotment Option is not exercised	Assuming the Over-allotment Option is exercised in full
Cornerstone Investor	Investment Amount (US$ in million)#					
Indus Funds	15	10,603,500	5.69%	4.95%	1.08%	1.05%
GIC Tetrad Ventures Pte Ltd	30	21,207,500	11.38%	9.89%	2.16%	2.10%
GIC Private Limited	10	7,069,000	3.79%	3.30%	0.72%	0.70%

Figure 3.9: (Table B) The merged lines (thin green) created by the final line identification. Vertically split cells highlighted with thick boxes (red).



	Based on the Offer Price of HK$11.10 (being the Minimum Offer Price)					
					Approximate % of total Shares in issue immediately following the completion of Capitalization Issue and Global Offering	
		Number of Offer Shares (rounded down to nearest whole board lot of 500 Shares)	Approximate % of total number of Offer Shares			
			Assuming the Over-allotment Option is not exercised	Assuming the Over-allotment Option is exercised in full	Assuming the Over-allotment Option is not exercised	Assuming the Over-allotment Option is exercised in full
Cornerstone Investor	Investment Amount (US$ in million)#					
Indus Funds	15	10,603,500	5.69%	4.95%	1.08%	1.05%
GIC Tetrad Ventures Pte Ltd	30	21,207,500	11.38%	9.89%	2.16%	2.10%
GIC Private Limited	10	7,069,000	3.79%	3.30%	0.72%	0.70%

Figure 3.10: (Table B) The table after the CNN merged problematic cells. Rows kept are separated for accurate OCR.

of the dividing line as an additional input. These approaches result in significantly lower accuracy and higher complexity, respectively.

The proposed CNN outputs the chance that these two cells are merged and if one or both of the cells are empty. By using a neural network to compute this information, the effect of imperfections in the image can be greatly reduced. It is important to know which cells are empty to avoid running OCR on those cells. This increases performance, as the OCR is the most time consuming step. The following sections talk about the implementation details of the CNN, the training data, and the post CNN concatenation.

	Based on the Offer Price of HK$11.10 (being the Minimum Offer Price)	< EXTEND	< EXTEND	< EXTEND	< EXTEND	
					Approximate % of total Shares	< EXTEND
					in issue immediately following	< EXTEND
		Number of	Approximate % of total	< EXTEND	the completion of Capitalization	< EXTEND
		Offer Shares	number of Offer Shares	< EXTEND	Issue and Global Offering	< EXTEND
		(rounded		Assuming the		Assuming the
		down to	Assuming the	Over-allotment	Assuming the	Over-allotment
		nearest whole	Over-allotment	Option is	Over-allotment	Option is
Cornerstone	Investment	board lot of	Option is not	exercised	Option is not	exercised
Investor	Amount	500 Shares)	exercised	in full	exercised	in full
	(US$ in					
	million)*					
Indus Funds	15	10,603,500	5.69%.	4.95%	1.08%	1.05%
GIC						
Tetrad Ventures						
Pte Ltd	30	21,207,500	11.38%	9.89%	2.16%	2.10%
GIC Private						
Limited	10	7,069,000	3.79%	3.30%	0.72%	0.70%

Figure 3.11: (Table B) Tablext output of the table in Fig. 3.4, in CSV format. Thick boxes (red) show the horizontally merged cells. Cells with multiple rows are kept separated for readability.

3.3.4.1 CNN Construction

The two cells that have been sent into the CNN are merged and rotated so that the potential improper line separating these two be always a vertical line in the center of the merged image. The merged image, as the CNN sees it, always has a dimension of 200×100.

The CNN has three branches that are concatenated into a dense network. The first branch has three 2D convolution layers with 16 filters with a size of 3×3, each layer is directly followed by a max pooling layer of size 2×2. This branch's main purpose is to give low fidelity information around the possibly improper line that splits the cells, but it also helps identify which cells contain data.

The second branch begins with an asymmetric average pooling layer of size 1×100. The output of this layer has the dimension of 200×1. This layer is followed by five 1D convolution layers with four filters each with a size of 3. The average pooling gives a good noise resistant indication if any data exists within the particular columns. This information helps identify the existence of data within one or both of the cells, but also helps confirm the presence of an improper split of the two cells.

The third and final branch does not take in the full merged image like the first two, instead it takes in the centermost 20×100 pixels. The exclusive purpose of this branch is to identify an improper splitting of the cells. Intuitively, the pixels where

the cells are merged and the surrounding pixels contain the most crucial information regarding whether the cells should be merged. More complex neural network layers are used in this branch due to the importance of the information and the limited amount of inputs. Five convolutions layers, each with 64 filters of size 3×3, are stacked upon one another.

All of these branches are individually passed through a dropout layer with a 20% dropout rate before they are flattened and concatenated. The concatenated data is sent into a dense layer of size 256, which is followed by another dropout layer, with a dropout rate of 50% and another dense layer of equal size. This second dense layer is then connected to the final output which contains 3 nodes. The first two nodes signify if the two cells contain data and the third node is high when two cells should be merged. The first dropout layers help guide the network away from relying purely on a single branch. Additionally, all of the dropout layers help avoid overfitting that would otherwise be present due to the limited amount of training data.

3.3.4.2 CNN Training

To train this CNN, a dataset that includes over 1000 tables from various domains is procured and manually annotated. The ground truth of every table's location, along with its respective cells' locations, is stored within Extensible Markup Language (XML) files. To translate this dataset into useful data for training purposes, Tablext's line identification techniques are used to split up the cells. All possible pairs of adjacent cells are then merged and these merged cells are used as the training data.

To create correct labels for training, the cells' locations are used. If a cell that contains data, according to the XML data, exists on either half of the merged cell, the respective half is considered to have data. When the XML data signifies that a single cell covers some part of both halves of the merged cell, the label for concatenation is high.

3.3.4.3 Post CNN Concatenation

The outputs of the CNN fill in matrices that record which cells should be merged and which cells contain data. Multiple cells in both the vertical and horizontal directions can be combined, if necessary, into a single large merged cell. Fig. 3.10 is a debug image that shows this result. Cells with multiple independent horizontal lines are not combined at this stage, so that the OCR is fed a single line of data for having a higher accuracy. These horizontal lines can be merged later to allow for a more concise table, or left separated to better represent a table's proportions. With the cells properly defined, the data is almost ready for OCR. However, the pixel resolution used by the table extractor, 800 pixels, is generally not sufficient to obtain accurate results from OCR. To address this problem, the lines defining the cell boundaries are scaled and used to cut cells into the original image.

3.3.5 OCR

All the non-empty, individual cells are sent to Tesseract, an open-source OCR tool. The returned text is then stored in a Column Separated Values (CSV) format with the location provided by Tablext. The output is shown in Fig. 3.11. All the data within an extended cell is placed within a single cell in the CSV. To allow for accurate data interpretation, all of the merged cells point back to that original data by using the keyword "EXTEND" and a directional arrow. These cells are emphasized with boxes in Fig. 3.11. For clarity, cells that take up multiple horizontal lines are not merged in the CSV.

3.4 Evaluation

Unlike Tablext, most state-of-the art table extraction studies do not have their code available for download. However, both DeepDeSRT [171] and TableNet [172]

have tested their design on the ICDAR 2013 table competition dataset [173]. To compare Tablext quantitatively to these works, Tablext is also tested against this dataset. Additionally, Tablext's extraction capability is compared against the popular, open-source, PDF extractor Tabula [174] on a manually procured diverse dataset with over 100 tables. Although table identification is not the focus of this project, the proposed method produces high quality results, while only using a small fraction of the total runtime.

3.4.1 Table identification results

The table identifier is tested on a dataset with over 400 tables from various domains. This dataset includes many abstract tables without any lines that are much harder to identify than most tables.

Let AP be the area predicted in an image and AL be the true occupied area. Precision and recall, for an image, can be calculated by precision $= \frac{AP \cap AL}{AP}$ and recall $= \frac{AP \cap AL}{AL}$. The results of the custom CNN, the YOLO model, and a combination of the custom CNN and YOLO model have been summarized in Table 3.2. After averaging the precision and recall across the dataset, with each image weighted equally, final results of 86% and 87% are achieved for precision and recall respectively for the combined model.

Although the YOLO only model achieves higher precision, the higher recall of the combination of the custom CNN and YOLO is preferable since it is better to over propose regions rather than to miss any tables. This is because regions that are not tables can be discarded later in the extraction process while completely missed tables cannot be recovered.

Approach	Recall	Precision	F1-Score
Custom CNN	0.5093	0.3802	0.4354
YOLO	0.8654	0.8989	0.8818
Custom CNN + YOLO	0.8716	0.8663	0.8689

Table 3.2: Different table identification comparison.

Work	Recall	Precision	F1-Score
Tablext	0.9091	0.9221	0.9156
DeepDeSRT	0.8736	0.9593	0.9144
TableNet	0.9001	0.9307	0.9151

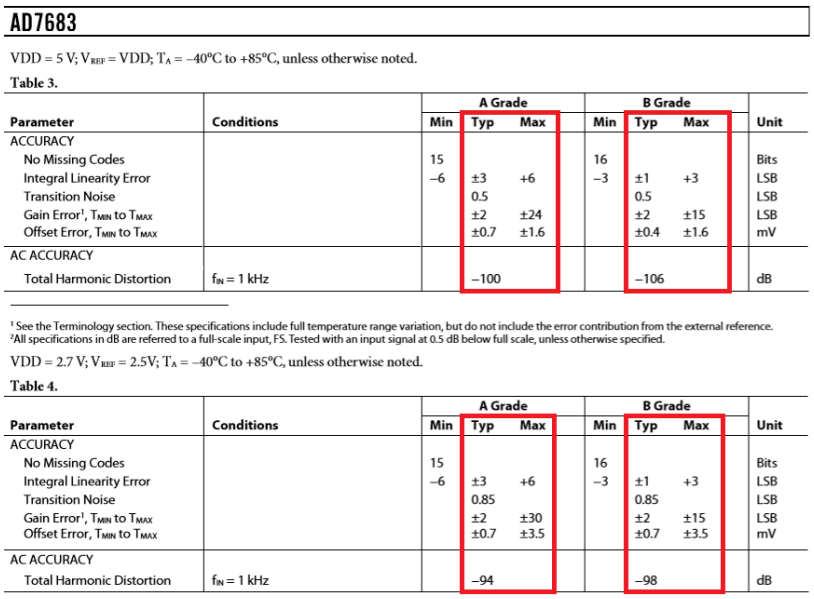
Table 3.3: ICDAR data extraction results Comparison. Both DeepDeSRT and TableNet are trained on a subset of ICDAR dataset while Tablext does not use any ICDAR data to train with which proves Tablext's ability to extract data from tables in general formats.

3.4.2 Table extraction result comparison

3.4.2.1 ICDAR comparision

Both DeepDeSRT and TableNet use 34 randomly selected images from the ICDAR dataset for their testing set. Then, they calculate precision and recall in the specific way mandated by ICDAR. For every cell within a table, "adjacency relations" defined by ICDAR, are found with its nearest horizontal and vertical neighbors. In order to get precision and recall, the adjacency relations are compared with the ground truth. Precision and recall are computed for each document then the average is taken across all the documents.

To compare Tablext to the other works, the same method to calculate precision and recall is used. It should be noted that both DeepDeSRT and TableNet fine tune their networks by training on the remaining ICDAR dataset. In order to prove its ability to extract data from tables in general formats, Tablext does not use any ICDAR data to train with. The results comparing the different methods can be found in Table 3.3. Tablext has both the highest precision and overall F1-Score out of the methods.



AD7683

VDD = 5 V; V_{REF} = VDD; T_A = –40°C to +85°C, unless otherwise noted.

Table 3.

Parameter	Conditions	A Grade Min	A Grade Typ	A Grade Max	B Grade Min	B Grade Typ	B Grade Max	Unit
ACCURACY								
No Missing Codes		15			16			Bits
Integral Linearity Error		–6	±3	+6	–3	±1	+3	LSB
Transition Noise			0.5			0.5		LSB
Gain Error[1], T_{MIN} to T_{MAX}			±2	±24		±2	±15	LSB
Offset Error, T_{MIN} to T_{MAX}			±0.7	±1.6		±0.4	±1.6	mV
AC ACCURACY								
Total Harmonic Distortion	f_{IN} = 1 kHz		–100			–106		dB

[1] See the Terminology section. These specifications include full temperature range variation, but do not include the error contribution from the external reference.
[2] All specifications in dB are referred to a full-scale input, FS. Tested with an input signal at 0.5 dB below full scale, unless otherwise specified.

VDD = 2.7 V; V_{REF} = 2.5V; T_A = –40°C to +85°C, unless otherwise noted.

Table 4.

Parameter	Conditions	A Grade Min	A Grade Typ	A Grade Max	B Grade Min	B Grade Typ	B Grade Max	Unit
ACCURACY								
No Missing Codes		15			16			Bits
Integral Linearity Error		–6	±3	+6	–3	±1	+3	LSB
Transition Noise			0.85			0.85		LSB
Gain Error[1], T_{MIN} to T_{MAX}			±2	±30		±2	±15	LSB
Offset Error, T_{MIN} to T_{MAX}			±0.7	±3.5		±0.7	±3.5	mV
AC ACCURACY								
Total Harmonic Distortion	f_{IN} = 1 kHz		–94			–98		dB

Figure 3.12: (Table D) A text-based PDF page in our testbench with potentially problematic columns highlighted with boxes (red).

Work	Recall	Precision	F1-Score
Tablext	0.9192	0.9437	0.9313
Tabula	0.7110	0.734	0.7223

Table 3.4: Diverse Data Extraction Results Comparison.

3.4.2.2 Diverse Dataset comparison

Without the source code for DeepDeSRT or TableNet, Tablext cannot compare to them qualitatively or quantitatively on a large diverse dataset. Therefore, a comparison is made with the popular PDF extractor Tabula. Tabula can only extract non-scanned, text-based PDFs. So a testbench, that exclusively contains text-based PDFs, is created by randomly selecting over 100 tables within PDFs from various domains. Tabula is given the PDF pages with the text and meta-information, while Tablext is given a scanned image of each PDF page. Fig. 3.12 is an example of one such page within our testbench.

Fig. 3.12 is extracted by both Tablext and Tabula. With this input, Fig. 3.13 and

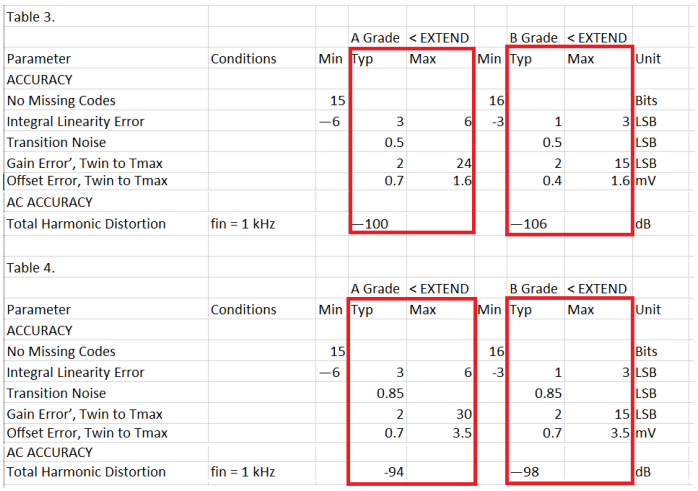


Table 3.

			A Grade	< EXTEND		B Grade	< EXTEND	
Parameter	Conditions	Min	Typ	Max	Min	Typ	Max	Unit
ACCURACY								
No Missing Codes		15			16			Bits
Integral Linearity Error		−6	3	6	-3	1	3	LSB
Transition Noise			0.5			0.5		LSB
Gain Error', Twin to Tmax			2	24		2	15	LSB
Offset Error, Twin to Tmax			0.7	1.6		0.4	1.6	mV
AC ACCURACY								
Total Harmonic Distortion	fin = 1 kHz		−100			−106		dB

Table 4.

			A Grade	< EXTEND		B Grade	< EXTEND	
Parameter	Conditions	Min	Typ	Max	Min	Typ	Max	Unit
ACCURACY								
No Missing Codes		15			16			Bits
Integral Linearity Error		−6	3	6	-3	1	3	LSB
Transition Noise			0.85			0.85		LSB
Gain Error', Twin to Tmax			2	30		2	15	LSB
Offset Error, Twin to Tmax			0.7	3.5		0.7	3.5	mV
AC ACCURACY								
Total Harmonic Distortion	fin = 1 kHz		-94			−98		dB

Figure 3.13: (Table D) Tablext's final output in CSV format. Boxes (red) are showing the correct handling of problematic cells.

Fig. 3.14 respectively show Tablext's and Tabula's final CSV output. Both have a high accuracy when reproducing the cells' contents, though Tabula is simply reading text provided to it, while Tablext is using OCR. One issue with the Tablext output is that Tesseract appears to not recognize the plus-minus character. Tabula also did not perfectly print the data within the cells, despite reading straight from the PDF. The minus character could not be recognized by Tabula. This problem appears because this PDF is using an abnormal character instead of the standard ASCII symbol.

Several severe structural errors appear in Tabula's output and these errors have been highlighted in Fig. 3.14. Looking back at the original image in Fig. 3.12, it becomes clear what caused these errors. The cells "A Grade" and "B Grade" overlap both the columns beginning with "Typ" and "Max". Errors like this are common for conventional text-based extractors [17]. Tablext meanwhile, with its dynamic threshold ratio, easily separates these two columns then merges the split "A Grade" and "B Grade" cells back into a single cell.

The highlighted cells in Fig. 3.14 emphasize the importance of accurate cell location identification. Even with clever post-processing, it would be impossible to tell which column the merged cells with a single piece of data belong to. For instance, looking

			A Grade		B Grade	
Parameter	Conditions	Min	Typ Max	Min	Typ Max	Unit
ACCURACY						
No Missing Codes		15		16		Bits
Integral Linearity Error		?6	±3 +6	?3	±1 +3	LSB
Transition Noise			0.5		0.5	LSB
Gain Error1, TMIN to TMAX			±2 ±24		±2 ±15	LSB
Offset Error1, TMIN to TMAX			±0.7 ±1.6		±0.4 ±1.6	mV
AC ACCURACY						
Total Harmonic Distortion	fIN = 1 kHz		?100		?106	dB
			A Grade		B Grade	
Parameter	Conditions	Min	Typ Max	Min	Typ Max	Unit
ACCURACY						
No Missing Codes		15		16		Bits
Integral Linearity Error		?6	±3 +6	?3	±1 +3	LSB
Transition Noise			0.85		0.85	LSB
Gain Error1, TMIN to TMAX			±2 ±30		±2 ±15	LSB
Offset Error1, TMIN to TMAX			±0.7 ±3.5		±0.7 ±3.5	mV
AC ACCURACY						
Total Harmonic Distortion	fIN = 1 kHz		?94		?98	dB

Figure 3.14: (Table D) Tabula's final output in CSV format. Boxes (red) are showing the incorrect handling of problematic cells

at the row beginning with "Transition noise", there is no way to know if the data belongs to the "Typ" or "Max" column.

The results comparing the two methods can be found in Table 3.4. Tablext's ability to handle complex tables greatly surpasses Tabula's, despite all of the extra meta-information that Tabula has access to.

3.5 Conclusion

This project introduces a novel, general approach for table extraction. By utilizing both deep learning and computer vision techniques, the neural network can focus its attention on complex problems, while allowing more conventional methods to handle the simpler tasks. This focus allows Tablext to beat competing state-of-the-art neural-network-based table extraction methods as well as a popular open-source tool that requires table meta-data.

CHAPTER IV

AnGeL: Fully-Automated Analog Circuit Generator Using a Neural Network Assisted Semi-supervised Learning Approach

Machine Learning (ML) has shown promising results in predicting the behavior of analog circuits. However, in order to completely cover the design space for today's complicated circuits, supervised ML requires a large number of labeled samples which is time-consuming to provide. Furthermore, a separate dataset must be collected for each circuit topology making all other previously gathered datasets useless. In this project, we first present a database including labeled and unlabeled data. We use neural networks to determine the behavior of complicated topologies by combining the more simple ones. By generating such unlabeled data, the time for providing the training set is significantly reduced compared to the conventional approaches. Using this database, we propose a fully-automated analog circuit generator framework, AnGeL. AnGeL performs all the schematic circuit design steps from deciding the circuit topology to determining the circuit parameters *i.e.* sizing. Our results show that for multiple circuit topologies, in comparison to the state-of-the-art works while maintaining the same accuracy, the required labeled data is reduced by **4.7x - 1090x**. Also, the runtime of AnGeL is **2.9x - 75x** faster.

4.1 Introduction

The existence of various design parameters and specifications in present-day complex circuits, in addition to severe process variations, have made the manual circuit design procedure challenging, time-consuming, and inefficient [175, 176]. All these challenges make the automated analog circuit generation a necessity. Schematic circuit design includes two main steps: deciding the topology and determining the value of the circuit elements (*i.e.* sizing) to meet the desired specifications.

Model-based approaches are one of the main techniques in the automated sizing of circuits *e.g.* non-convex polynomial optimization, geometric programming, and Neural Network (NN) [18, 41]. In such approaches, a global model is built based on the collected training set, which makes the model reusable for other data. However, to maintain high accuracy while covering the whole design space, a large labeled training set is required [177]. SPICE simulation is used for gathering such a large set which is time-consuming. To make matters worse, a separate dataset is required for each circuit topology, even if a single element is added or removed.

Even though the model-based approaches are reusable, their accuracies are not usually high due to the large number of design parameters and nonlinearity of object and constraint functions [28]. On the other hand, the other approaches in automated sizing, simulation-based algorithms [8, 26, 28, 178], optimize circuits directly by the gathered simulated data and usually have higher accuracy. However, they are more time-consuming in comparison with model-based methods and are non-reusable. Here, non-reusable means that even if the target specifications slightly changes, the whole process needs to reoccur. The third approach in circuit sizing is a hybrid of model- and simulation-based methods. In such approaches, after building the initial model, the model is gradually updated by running simulations during the optimization procedure instead of using an offline model.

In order to reduce the size of the labeled dataset, we propose a co-learning-based

NN approach. The term co-learning here means passing the knowledge from usually less complex models to more complicated ones to reduce the training cost [91]. In our case, the previously gathered datasets of simpler circuits are leveraged to shrink the required labeled training sets for more complicated circuit topologies. In other words, we use NNs to determine the circuit behavior of complicated topologies by combining the simpler ones.

Using the presented database, we propose a fully-automated analog circuit generator framework, AnGeL. The goals of AnGeL are threefold: (a) achieve a reusable, accurate, and fast model to meet the given specifications of the overall circuit, (b) reduce the number of required labeled training samples, and (c) perform all schematic circuit design steps such as deciding the overall circuit topology, selecting the topology of sub-circuits, and sizing them. Since we are using both labeled and unlabeled data in the database, our approach is classified as semi-supervised learning.

In order to achieve a reusable, accurate, and fast model, AnGeL uses a hybrid (model-simulation-based) approach for sizing. Between model-based approaches, NNs have shown promising results in all circuit design levels such as single-board computer, sizing, layout, post-layout simulation, and system-on-chip design [17, 101, 179–181]. Also, once the model is trained, it is able to execute a large input set in a very short time. In our approach, we use an NN to estimate the functionality of circuits such that it determines the performance of interest (circuit's specifications) when the circuit parameters are given. We use such an NN as the global optimization engine to determine the circuit parameters to meet the desired specifications. To improve the accuracy, we implement Particle Swarm Optimization (PSO), which is a simulation-based algorithm while the trained NN is invoked instead of the SPICE simulator to provide the necessary data. Indeed, PSO is a population-based approach that has a better precision compared to other methods *e.g.* genetic algorithms [8]. In this regard, our proposed approach is accurate because of implementing PSO in the

local optimization as well as reusable and fast due to using an NN as the functional estimator of the circuit.

In addition to using an efficient database, in order to further reduce the size of the training data, AnGeL divides the overall circuit into multiple sub-circuits and analyzes each individually while meeting the given specifications of the overall circuit. This approach allows dealing with circuits with less design parameters. Therefore, a smaller training set is required to analyze them while maintaining equivalent accuracy in comparison with analyzing the overall circuit [15]. Moreover, the global and local optimum points are found faster since design spaces of sub-circuits are smaller. Furthermore, sub-circuits are analyzed in parallel which reduces the runtime. Another advantage of this circuit division is supporting many topologies while AnGeL is trained on a few of them. As an example, when AnGeL trains on 12 single-stage operational amplifier (OPAMP) topologies, it supports $12\times12=144$ two stages topologies. To this end, since the given specifications are for the overall circuit, AnGeL breaks down them to appropriate sub-circuits specifications while considering the effect of other sub-circuits *e.g.* loading effect. For further division, even a sub-circuit may be broken into smaller chunks. For instance, a two-stage OPAMP may consist of two differential stages where each stage may consist of a gain block, current source, and DC-biasing chunks. So, AnGeL hierarchically determines the specifications of each sub-circuit regarding the specifications of the parent circuit. This hierarchical approach enables AnGeL to design circuits with many design parameters ($\sim$40) in a short time and with high accuracy.

AnGeL also selects the most suitable high-level circuit topology (number of stages, type of sub-circuits *e.g.* gain block, DC-biasing) between available candidates as well as determining the target specifications of each sub-circuit. In a similar way, the topology of each sub-circuit is selected among the available candidates. The topology of sub-circuits as well as their sizing are determined based on the sub-circuit

specifications.

To validate our proposed method, all the design steps of one-, two-, and three-stage OPAMPs as well as filters have been demonstrated. The performance of the synthesized circuits by AnGeL is validated by SPICE simulations. Our results show that for multiple circuit topologies, in comparison to the state-of-the-art works while maintaining the same accuracy, the required labeled data is reduced by **4.7x - 1090x**. Also, the runtime of AnGeL is **2.9x - 75x** faster. Moreover, testing on more than 1,450 different circuits illustrates that the performance of circuits that are determined using combining simpler topologies has an average percentage error of less than **0.043**.

The main contributions of this project can be summarized as follows:

- Creating a database using a co-learning-based NN model that significantly reduces the size of the required labeled training set compared to the state-of-the-art works.
- An efficient and intelligent fully-automated analog circuit generator framework that uses the created database and designs circuits from deciding the topology to sizing.
- Achieving a fast and accurate method using a hybrid of model- and simulation-based approaches.
- Designing circuits with many design parameters ($\sim$40) using a hierarchical approach.
- Supporting many circuit topologies while few topologies are used in the training by leveraging the circuit division into sub-circuits.

4.2 Background and Related Work

4.2.1 Problem Formulation

Estimating the functionality of circuits and optimizing them are two important areas in automating the design of analog circuits [130]. In estimating the functionality of circuits, the main goal is to find f as a function of circuit parameters, $\mathbf{x}$, to approximate the performance of interest, y. DC bias voltages and size of transistors (W, L) are examples of circuit parameters, and the voltage gain of an OPAMP is an example of the performance of interest: $y \approx f(\mathbf{x})$.

The goal of analog circuit optimization is to determine the design parameters such that

$$\begin{aligned} &\text{minimize } f_1(\mathbf{x}), \ldots, f_M(\mathbf{x}) \\ &\text{subject to: } c_1(\mathbf{x}) < 0, \ldots, c_N(\mathbf{x}) < 0, \end{aligned} \tag{4.1}$$

where $f_1, \ldots, f_M$ are the figure of merit of the circuit [182]. $c_1, \ldots, c_N$ are constraints such as $x_j \in [p_j^-, p_j^+]$ or bandwidth (BW) $>$ 1GHz.

4.2.2 Reducing Database Size

In estimating the functionality of circuits, decreasing the number of required labeled data is crucial. There are several works that have used the concept of co-learning for this purpose [15, 91, 97]. The main idea of them is to find two representations for modeling a circuit's specification where one of them is simpler and less accurate than the other. By passing the information from the simpler model to the more complicated one, they achieve an accurate model with fewer labeled data. Hassanpourghadi *et al.* [177] and Alawieh *et al.* [15] have divided the overall circuit into multiple sub-circuits to conclude simpler models, and as a result, the size of the training set is decreased. Even though the aforementioned approaches have helped reduce the number of labeled data,

a completely new dataset is required, even with a small modification in the topology. Wang *et al.* [183] have mentioned the idea of transferring knowledge between topologies using graph neural networks. However, this transfer is applied from a two-stage to a three-stage OPAMP, not among single sub-circuits.

4.2.3 Circuit Design Framework

There are several automated circuit design frameworks that both create a topology and size the circuits [32, 184–186]. Since they are mainly simulation-based approaches, they are slow and have relatively high computation costs due to the need of running a large number of simulations. Moreover, they are non-reusable.

Several studies have focused on the topology construction [187–189] as their main idea. They either use an external tool for sizing [187] or perform an initial sizing [188]. Topology construction is categorized into two types in general: (a) topology selection from a predefined library and (b) topology generation from basic building blocks [187, 189]. Generating topologies from basic building blocks is more creative as it may lead to creating circuit topologies that have not been seen before. Furthermore, traditional topology selectors support fewer topologies [188]. FEATS [187] represents circuits with a graph and uses abstract building blocks to generate circuit topologies. Even though it supports more than 10,000 topologies, it takes hours for it to synthesize. FUBOCO [188] performs hierarchical design using functional blocks. Although it is faster than the previous approaches [187], it takes it still several hours to design as it consequently designs stages and sizes many topologies before concluding. On the other hand, in our approach, using the idea of division and hierarchical parallel design, we are able to support more than 10,000 topologies while spending a few minutes to conclude the design. Indeed, our work is positioned between the traditional topology selector approaches and basic-building-blocks-based approaches since AnGeL still selects the sub-circuit topologies but works with them as basic blocks in the top-level

topology.

Many studies have assumed the circuit topology is given as an input [130], and they just have focused on sizing. NNs have a fast runtime and have a high degree of freedom due to their multiple hidden layers [177]. Several studies use fully-connected-NNs for sizing of analog circuits [190–192]. Their main disadvantages are requiring a large labeled training set which is time-consuming to provide, as well as inaccuracy. Liu *et al.* [193] use transfer learning to decrease the size of the post-layout training set by leveraging the schematic gathered data. Recently, reinforcement learning, graph neural network, and constraint programming [16,194,195] also have been used in analog circuit sizing. The most common method to automate circuit sizing is simulation-based approaches [196]. Bayesian optimization is one of the most popular techniques in this regard [28, 197]. However, because of the inherent expensive computation cost of simulation-based approaches, Zhang *et al.* [6] try to extract the "good" features and define the Gaussian Process based on them using NNs. PSO is another popular technique in circuit sizing [8, 198–200] however, the runtime is long. To alleviate these drawbacks, a hybrid model-simulation-based approach is implemented by replacing the slow SPICE simulation with a fast circuit functional model, which usually is an NN [58, 86]. Supporting only one topology, having a narrow design space, and requiring a large training set are their main shortcomings.

4.3 Proposed Database Generation

4.3.1 Overview

Collecting datasets for simpler topologies is less expensive since they have smaller design spaces. On the other hand, in order to completely cover the design space for complex circuits with many design parameters (*e.g.* size of transistors), an abundant number of samples are needed. As an example, T_{out} in Fig. 4.1(b) has 21 design

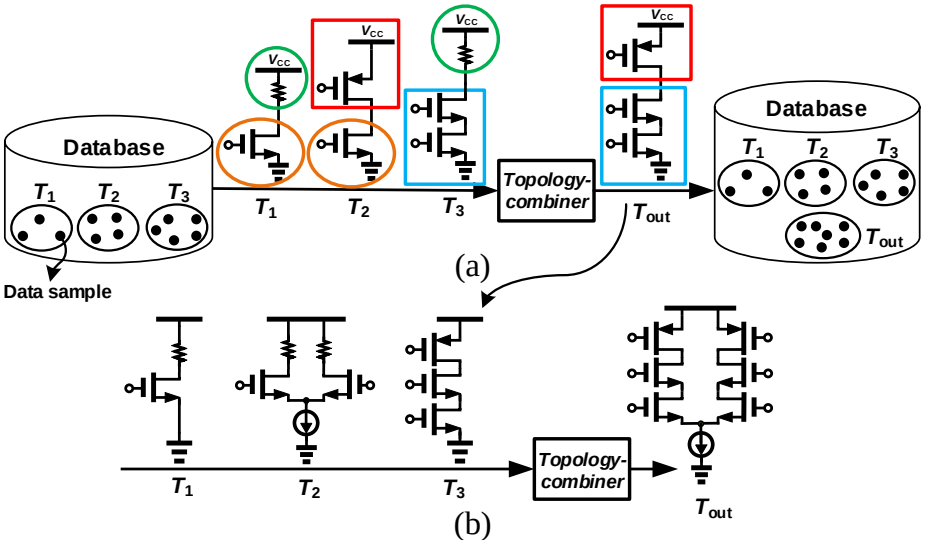


Figure 4.1: (a) An example of three simpler topologies, T_1, T_2, T_3, that can be combined to determine the circuit behavior of T_{out}. To enhance visibility, we use identical colored boxes/circles to show bodies/loads that are the same in different topologies. (b) The output topology of Fig. 4.1(a) is used to make a more complicated topology.

parameters *i.e.* width, length, v_{bias} for 6 transistors and a current source. Therefore, gathering only labeled data by running simulations would be computationally expensive.

Fig. 4.1(a) demonstrates the proposed co-learning-based database generation flow. The goal of the topology-combiner is to determine the circuit behavior (*i.e.* the effect of design parameters on output specifications) of complicated topologies using the simpler ones in order to reduce the size of the required labeled data. The datasets for more basic, simple topologies, *e.g.* T_1, T_2, T_3, are gathered using simulations. Such datasets contain the value of the specification associated with different circuit design parameters for each topology. Then, the circuit behavior of the more complicated topology (T_{out}) is determined using combinations of the simpler ones. For example, in Fig. 4.1(a), T_{out} is built by replacing the resistive load of T_3 with the PMOS load of T_2. Moreover, T_1 has the same body and resistive load as T_2 and T_3, respectively. So, intuitively, the behavior of T_{out} can be modeled by leveraging the previously gathered

T_1, T_2, T_3 datasets information. Then, a large pseudo sample set that covers the design space of T_{out} is generated with almost zero cost using such a model (unlabeled data). So, our database is a combination of labeled and unlabeled data. Now that the dataset of T_{out} is generated, it can be used to build more complicated topologies. As an example, the output topology in Fig. 4.1(a) is used to make a more complicated topology in Fig. 4.1(b).

4.3.2 Topology Combiner Model

Fig. 4.2(a) demonstrates the general structure of three simpler topologies (T_1, T_2, and T_3) that can be combined to determine the circuit behavior of T_{out}. Each topology is composed of two parts: B (body) and L (load). T_1 and T_2 have the same body (B_1) and T_1, T_3 have the same load (L_1). Moreover, T_{out} has the same body (B_2) and load (L_2) as T_3 and T_2, respectively. We use the notation of $\mathbf{x}_B$ and $\mathbf{x}_L$ for showing the design parameters of body and load, respectively, and $f(\mathbf{x}_B, \mathbf{x}_L)$ for denoting a circuit specification when the body is B, and load is L. This f can be any of circuit specifications *e.g.* gain, BW, etc. The training process needs to be done individually for each circuit specification.

Our goal is to determine the mapping from $\mathbf{x}_{B_2}, \mathbf{x}_{L_2}$, *i.e.* the design parameters of T_{out} topology, to $f(\mathbf{x}_{B_2}, \mathbf{x}_{L_2})$ without having any direct mapping samples. That is, to determine the mapping $\xi_{T_{out}}$:

$$\xi_{T_{out}} : (\mathbf{x}_{B_2}, \mathbf{x}_{L_2}) \mapsto f(\mathbf{x}_{B_2}, \mathbf{x}_{L_2}) \tag{4.2}$$

Fig. 4.2(b) and (c) show the NN implementation for our proposed topology-combiner and how the dataset of each of T_1, T_2, and T_3 is used in the training of such NNs. The first step is to train the accessory-1 model using the training set of T_1. As it is depicted in Fig. 4.2(b) on the left side, inputs are the design parameters of T_1, and the output is the desired specification in T_1, *i.e.* $f(\mathbf{x}_{B_1}, \mathbf{x}_{L_1})$. Similarly, the accessory-2

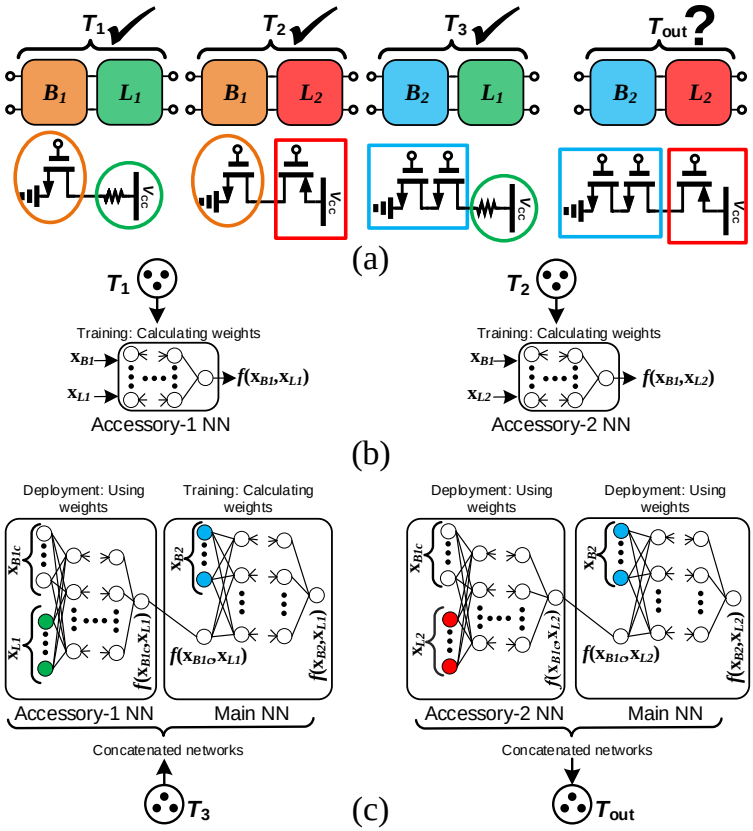


Figure 4.2: The topology-combiner model. (a) The general structures of three simpler topologies, T_1, T_2, and T_3, which are combined to determine the circuit behavior of T_{out} along with the example of Fig. 4.1(a). We use the same boxes/circles with the same color for showing bodies/loads that are the same to make them more visible in different topologies. (b) Our proposed co-learning-based NN models: Training of accessory-1 and accessory-2 models using T_1, and T_2 dataset, respectively. (c) Our proposed co-learning-based NN models: Training and deployment of the main NN using T_3 dataset.

model is trained by leveraging the T_2 dataset while inputs and outputs are the design parameters and the desired specification of T_2, respectively. The accessory-2 NN is shown in Fig. 4.2(b) on the right side.

Next, the accessory-1 and the main NNs are concatenated as shown in Fig. 4.2(c) on the left side. Note that the accessory-1 NN is used in the deployment phase now. By giving a fixed, constant $\mathbf{x}_{B_1}$ which we denote with $\mathbf{x}_{B_{1c}}$, the inputs variable of this concatenated network would be $\mathbf{x}_{B_2}$ and $\mathbf{x}_{L_1}$ which are the design parameters of T_3.

As it is shown, the output is the desired specification in T_3, *i.e.* $f(\mathbf{x}_{B_2}, \mathbf{x}_{L_1})$. Since the accessory-1 NN is already trained, by training this concatenated network, the weights of the main NN are calculated. In other words, we will have:

$$\xi_{Main} : (\mathbf{x}_{B_2}, f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_1})) \mapsto f(\mathbf{x}_{B_2}, \mathbf{x}_{L_1}). \tag{4.3}$$

The inputs of the main NN are $\mathbf{x}_{B_2}$, and $f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_1})$ while the output is the desired specification of the topology with B_2 body and L_1 load. Therefore, during the deployment phase of the main NN, by replacing L_1 with L_2 (giving $f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_2})$ instead of $f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_1})$), we will have the desired specification of the topology with B_2 body and L_2 load, which is our goal, T_{out}. This means we will have:

$$\xi'_{Main} : (\mathbf{x}_{B_2}, f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_2})) \mapsto f(\mathbf{x}_{B_2}, \mathbf{x}_{L_2}), \tag{4.4}$$

which has been derived by replacing $\mathbf{x}_{L_1}$ with $\mathbf{x}_{L_2}$ in (4.3). This is perfectly aligned with our goal in (4.2) as if we replace $\mathbf{x}_{L_2}$ with the corresponding $f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_2})$ in (4.2) we will conclude the same equation as (4.4). So, the last step is to map $\mathbf{x}_{L_2}$ to the corresponding $f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_2})$ which is achieved by using the trained accessory-2 NN. So, during the deployment phase of accessory-2 NN, by giving the $\mathbf{x}_{B_{1c}}$ as the body, we would have the mapping of $\mathbf{x}_{L_2}$ to $f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_2})$. This procedure is illustrated in Fig. 4.2(c) on the right side. Hence, to determine the behavior of T_{out} from its design parameters, we first find $f(\mathbf{x}_{B_{1c}}, \mathbf{x}_{L_2})$ leveraging the accessory-2 NN. Next, we feed the result to the main NN along with the design parameters of B_2.

To give more intuition about our models, it should be noted that the key is to have $f(\mathbf{x}_{B_1}, \mathbf{x}_{L_1})$ in the main NN instead of directly depending on the design parameters of the load. Indeed, $f(\mathbf{x}_{B_1}, \mathbf{x}_{L_1})$ abstracts the "effect" of a load in a topology with the B_1 body as single input for the main NN. This is the reason that we are able to replace L_1 with L_2 in the deployment phase of the main NN. Moreover, the other

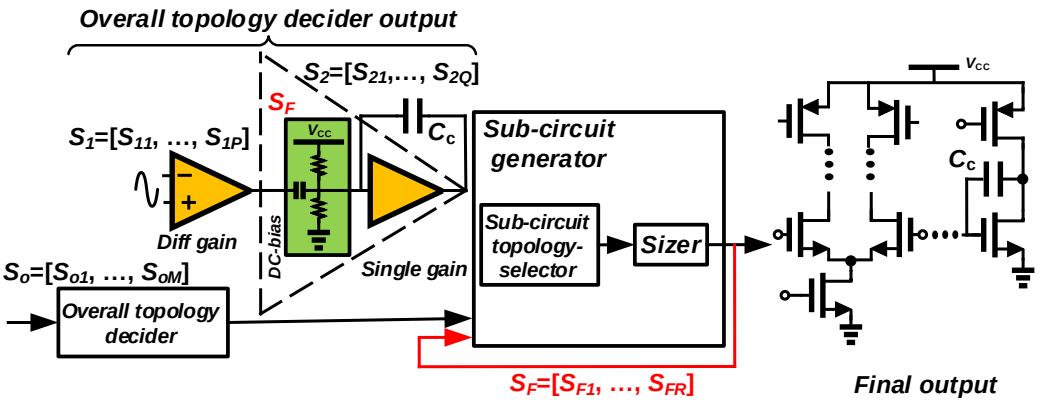


Figure 4.3: High-level platform architecture of AnGeL.
S_o: Desired specifications for the overall circuit given by users.
S_1, S_2: The specifications for each sub-circuit that are determined by the overall topology decider module.
S_F: Specifications of sub-circuits that are determined by the bigger sub-circuits that they are part of.

inputs of the main NN are the design parameters of B_2 which is the body of T_{out} too so, it works exactly as we want.

4.4 Proposed AnGeL Framework

Using the presented database, we create a fully-automated analog circuit generator framework, AnGeL.

4.4.1 Overview

As it was mentioned earlier, AnGeL divides the overall circuit into multiple sub-circuits and analyzes each individually. This results in dealing with smaller circuits and hence, requiring a smaller training set size while keeping the same accuracy in comparison with analyzing the overall circuit. The other advantages of dividing the circuit are finding global and local optimum points faster since design spaces of sub-circuits are smaller, faster runtime due to analyzing sub-circuits in parallel, and supporting many topologies while only a few of them are used in the training set.

Fig. 4.3 shows a high-level platform architecture of the proposed circuit design flow of AnGeL. The desired specifications, *i.e.* $S_o = [S_{o1}, \ldots, S_{oM}]$, are given to the overall topology decider module. Here, we assume the given specifications are all in equality format *e.g.* $S_{o1} = C_1, S_{o2} = C_2, \ldots$. The goal of AnGeL is to design a circuit at the transistor level so that its output specifications are as close as possible to the given ones. Transforming constraints in inequality format *e.g.* $S_{o1} > C_1, S_{o1} < C_2, \ldots$ along with objective functions *e.g.* minimize S_{oM} are explained in Section 4.4.5.

The overall topology decider module determines the high-level topology of the circuit accordingly. Indeed, this module determines the number of stages, type of each sub-circuit (*e.g.* gain block, DC-biasing), how the sub-circuits are connected to each other (*e.g.* having feedback, being in parallel or series), and having single-ended or differential input/output ports for sub-circuits. Moreover, the overall topology decider module determines the specifications for each sub-circuit (*i.e.* $S_1, S_2, \ldots$). The determined specifications for each sub-circuit is passed to the sub-circuit generator module to build a circuit at the transistor level.

The feedback loop around the sub-circuit generator module shows that the specifications of some sub-circuits (*i.e.* S_F) are determined by the bigger sub-circuits that they are part of. For instance, the specifications of the DC-bias sub-circuit cannot be determined before determining the input DC voltage of the following gain stage. In this regard, AnGeL hierarchically determines the specifications of each sub-circuit regarding the specifications of the parent circuit. This hierarchical approach and circuit division enables AnGeL to design circuits with many design parameters ($\sim$40) in a short time and with high accuracy. For example, a band-pass active filter consists of low-pass and high-pass filters that each may include a two-stage OPAMP. All these sub-circuits are designed in a hierarchical approach and in parallel with each other.

The first step in the proposed sub-circuit generation flow is to determine the topology. The topology is selected among the available topologies in the database.

$[n_1, n_2, k]$	[5, 5, 3]	[5, 5, 5]	[5, 7, 4]	[7, 6, 4]	[7, 7, 5]
Conventional cost	1,000	100,000	20,736	28,561	537,824
AnGeL cost	250	6,250	3,026	3,697	33,614

Table 4.1: Comparison cost of AnGeL and the conventional approaches regarding the curse of dimensionality. n_1 and n_2: number of design parameters in two different sub-circuits. k: number of samples for each parameter.

Based on the selected topology, AnGeL determines the value of design parameters (sizing the sub-circuit). There is a separate model for sizing each topology. To apply AnGeL to a new technology node, the training process needs to be redone. In order to decrease the number of such new training samples, the idea of transfer learning [201] can be used which is out of the scope of this project.

Using the topology combiner and the overall topology decider module, we do not majorly face the curse of dimensionality. The reason is that we are able to determine the behavior of complicated topologies that require more training sets (by leveraging simpler topologies with almost zero cost) and efficiently break down multi-stage circuits into multiple single stages. However, the nonideality of the topology combiner may slightly increase inaccuracy in the whole design process. It should be noted that, as mentioned in Section 4.5, the topology combiner has a very high accuracy, but this slight nonideality is in the trade-off with the curse of dimensionality. For a better cost comparison between AnGeL and the conventional approaches, we denote the number of design parameters in two different sub-circuits with n_1 and n_2, respectively. Assuming we take k samples for each parameter, the cost of AnGeL for analyzing a two-stage circuit with those two sub-circuits is ${n_1}^k + {n_2}^k$, considering only the effect of breaking down multi-stage circuits into multiple single stages. However, this cost for the conventional approach is $(n_1 + n_2)^k$. As summarized in Table 4.1, the cost of AnGeL is significantly lower than the conventional approaches for different typical n_1, n_2, and k values.

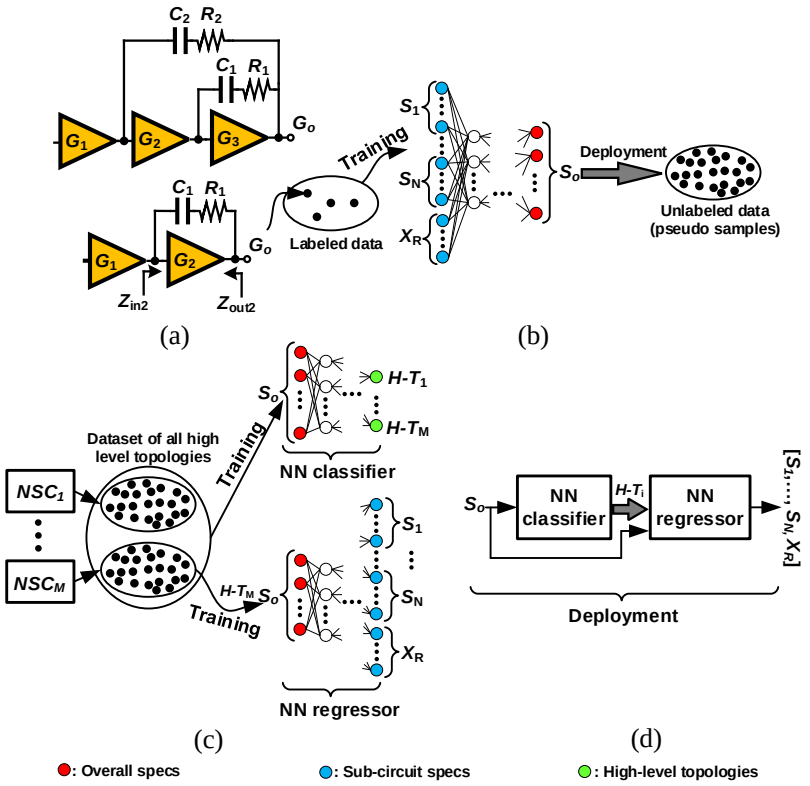


Figure 4.4: Overall topology decider flow. (a) Two examples of high-level topologies. (b) Net-specification-calculator model. (c) Overall-specification-breaker model in the training phase. (d) Overall-specification-breaker model in the deployment phase.
S_o: The vector of overall circuit specifications.
$[S_1, \ldots, S_N]$: The associated specifications sets of N sub-circuits.
X_R: A vector of design parameters that are not included in any sub-circuits.
$[NSC_1, \ldots, NSC_M]$: The associated net specification calculator models of M high-level topologies.
$[H\text{-}T_1, \ldots, H\text{-}T_M]$: M high-level topologies.

4.4.2 Overall Topology Decider

The main goal of the overall topology decider module is to determine the high-level topology of the circuit and break down the overall circuit specifications into usable specifications for sub-circuits. Determining the sub-circuit specifications via overall specifications is challenging. The assigned sub-circuits' specifications need to meet the overall specification when they are assembled together as the associated high-level topology. This requires a proper training set that teaches the module how to assign

sub-circuits' specifications such that the net impact of them in the associated high-level topology meets the overall specification. Furthermore, there are multiple ways for breaking down each overall specification, but many of them are not feasible for sub-circuits considering other specifications. For instance, assigning a large gain and BW to a sub-circuit may not be feasible simultaneously. Therefore, a large, proper training set is needed for learning all these relationships.

To overcome the aforementioned challenges we implement two main models: 1) net-specification-calculator model for calculating the net impact of different sub-circuits' specifications at the module-level on the overall circuit; 2) overall-specification-breaker model which uses a large dataset generated by the net-specification-calculator model for properly breaking down the overall specifications into sub-circuits specifications as well as determining the high-level topology. The overall-specification-breaker model consists of two NN models: A) a classifier for selecting the most suitable high-level circuit topology; B) a regressor for determining the target specifications of each sub-circuit.

Fig. 4.4(a) illustrates two examples of high-level circuit topologies. According to Fig. 4.4, the overall specifications (S_o) is a function of sub-circuits' specifications ($S_1, \ldots, S_N$), and other circuit's design parameters that are not included in any sub-circuits (X_R). For example, the overall DC gain (G_o) is a function of sub-circuits' DC gain (G_1, G_2, G_3) and input/output impedance of each sub-circuit (Z_{in}, Z_{out}). It should be mentioned that each of S_i $\forall i \in \{1, \ldots, N\}$ is a vector of specifications. So, to calculate the overall gain, we have the input/output resistance of each sub-circuit in addition to their DC gains, as an example. R_1, R_2, C_1, C_2 are examples of design parameters that are not included in any sub-circuits (*i.e.* X_R) in Fig. 4.4(a) that are used for calculating phase margin in amplifiers for instance. By taking the input/output impedance of sub-circuits as well as X_R into account, the loading effects are properly considered in our model.

Fig. 4.4(b) depicts the NN implementation of the net-specification-calculator model. During training, a labeled dataset is used for learning the net impact of different sub-circuits specifications at the module-level. A separate regression NN model is used for learning such net impacts on each high-level topology. Once the net-specification-calculator models are trained, in the deployment phase, a large set of different unlabeled data $(S_1, \ldots, S_N, X_R)$ are fed to them to generate pseudo samples $(\{S_1, \ldots, S_N, X_R\},\ S_o)$. Indeed, the output is the overall circuit specifications, S_o, associated with each unlabeled input in different high-level topologies. Generating such a large set is with almost zero cost since the net specification calculator is already trained. This large unlabeled dataset is used for training another model, overall specification breaker. This unlabeled dataset addresses both challenges of being large and being able to teach how to assign sub-circuits' specifications that the net impact of them meet the overall specifications. The reason is that for all of the large dataset samples $(\{S_1, \ldots, S_N, X_R\},\ S_o)$, the overall specification equals the net impact of sub-circuits' specifications by definition.

The overall specification breaker learns how to properly break down the overall specifications into sub-circuits' specifications as well as how to determine the high-level topology. Fig. 4.4(c) demonstrates the overall-specification breaker-model in training phase. During training, using a dataset of all high-level topologies populated by the net-specification-calculator model, an NN classifier [141] learns how to select the most suitable high-level circuit topology between available candidates regarding the overall specifications. Moreover, a separate NN regressor is implemented to break down the overall specifications into sub-circuits' specifications for each high-level topology using the large provided training set. This large training set works as a lookup table for the regressor model. As shown in Fig. 4.4(d), during deployment, first, the classifier selects the high-level topology. Then, based on such a high-level topology, the regressor model breaks down the overall specifications into sub-circuits' specifications.

It should be mentioned that with increasing the number sub-circuits or elements that are not included in sub-circuits (X_R), both the net-specification-calculator and overall-specification-breaker models become more complicated. However, our approach in such cases still outperforms the conventional methods which do not have the overall topology decider module as the conventional models become much more complicated than ours.

To summarize, these are the overall topology decider steps and models:

- The net-specification-calculator model calculates the net impact of different sub-circuits' specifications at the module-level on the overall circuit.
- In the deployment phase, for each overall circuit topology, the net-specification-calculator model generates a large set of overall specs by getting sub-circuit specs as inputs.
- Using the generated datasets by the net-specification-calculator models as the training set, the overall-specification-breaker model breaks down the overall specifications into sub-circuits specifications and determines the high-level topology as well.
- The overall-specification-breaker model consists of two NN models:
    1. A classifier for selecting the most suitable high-level circuit topology between available candidates.
    2. A regressor for determining the target specifications of each sub-circuit.
- With taking input and output impedances of sub-circuits as well as X_R into account, the loading effects are properly considered in our model.

4.4.3 Sub-circuit Topology-Selector

In order to decide the transistor level topology for each sub-circuit, we input the target specifications of the sub-circuit to a classifier model, and it selects the most suitable topology. There are multiple candidates in the database where the classifier decides which one is the most appropriate for the given specifications. To this end, three different classification models *i.e.* random forest, Support Vector Machine (SVM), and NN classifier [141, 202, 203] are analyzed and the one which gives the highest accuracy is selected as the sub-circuit topology-selector model. Moreover, the SVM is analyzed using four different kernels *i.e.* linear, polynomial, sigmoid, and Radial Basis Function (RBF). During the training, the circuit parameters and the associated topology are given as input and output, respectively.

More complicated topologies in the database have more instances since their design spaces are larger, which causes imbalanced data. To solve this, a subset of the database is considered for training that has the same number of samples for all topologies. This method is called down-sampling [204].

4.4.4 Sub-circuit Sizing

When the topology of sub-circuits is decided, a proper sizing is needed to determine the design parameters value *e.g.* size of transistors and value of voltages. The goal of sizing is to determine the design parameters such that:

$$\begin{aligned} &\text{minimize } \sum_i \omega_i |\frac{S_{si} - S^*_{si}}{S_{si}}|, \\ &\text{subject to: } |\frac{S_{si} - S^*_{si}}{S_{si}}| < \varepsilon_i, \end{aligned} \tag{4.5}$$

where S_{si} are the desired specifications and S^*_{si} are the determined specifications by the sub-circuit sizing module. ω_i are the weights that are used to prioritize specifications that are more important for users. $|\frac{S_{si}-S^*_{si}}{S_{si}}| < \varepsilon_i$ in Equation (4.5), ensures each output

specification is as close as possible to the associated desired specification.

In order to minimize Equation (4.5), we implement global and local optimization engines. For this purpose, first, we train a separate regression NN for each topology to estimate the functionality of sub-circuits when the circuit parameters are given. These NNs work as a fast circuit simulator instead of invoking time-consuming SPICE. We have used the idea of multiple starting points [82] in our optimization, which means we apply the local optimization on the multiple designs that have resulted in the minimum of Equation (4.5) in the global phase. The final result is the best one among these local optimums.

At the global optimization phase, we find the closest designs to the desired specifications by implementing a grid search that covers the design space. To this end, we apply the NN functionality estimator to each set of parameters in the design space. Then, the estimated result specifications are compared with the desired ones. Note that the design spaces of sub-circuits are relatively small since they have only a few design parameters (4-8). Moreover, once the NN model is trained, it is able to execute a large input set in a very short time. So, the global optimization phase takes only a few seconds.

Particle Swarm Optimization (PSO) is implemented as the local optimization engine. In PSO, if we denote the position of the i^{th} particle at iteration k as X_i^k, we will have:

$$X_i^{k+1} = X_i^k + V_i^{k+1}, \tag{4.6}$$

where V_i^k is the velocity of particle i^{th} at iteration k. The velocity is updated as:

$$V_i^{k+1} = wV_i^k + b_1 r_1(pbest_i - X_i^k) + b_2 r_2(gbest - X_i^k), \tag{4.7}$$

where r_1, r_2 are random numbers and w, b_1, b_2 are constant hyperparameters. Furthermore, $pbest_i$ denotes the position that gives the best-explored value for the i^{th} particle

while *gbest* is the best-explored value by all the particles [199]. In order to accelerate the local optimization process, we use the trained NN to estimate the functionality of sub-circuits instead of invoking SPICE.

As explained in more detail in Section 4.5, we have tested multiple algorithms as the global and local optimization engines. Using any of the simulation-based algorithms as the global engine makes the process slow as it requires many iterations ($\sim$ 300-1500), and such iterations happen sequentially. Moreover, for the local engine, one of the advantages of PSO over other optimization algorithms (*e.g.* Bayesian Optimization (BO), Simulated Annealing (SA), etc.) is that PSO processes multiple particles at each iteration which causes much faster design space exploration. Especially since we are using the NN regression model as the circuit simulator, this process is very fast. We use only 5-10 iterations in the local phase using PSO.

4.4.5 Constraints Transformation

As it was mentioned earlier, AnGeL is designed to get the desired overall specifications, $S_o = [S_{o1}, \ldots, S_{oM}]$, as an input while all constraints are defined in equality format *e.g.* $S_{o1} = C_1, S_{o2} = C_2, \ldots$. The goal of AnGeL is to implement a circuit that its output specifications are as close as possible to the given ones. However, it is common to have constraints in inequality format along with an objective function, as shown in Equation (4.8).

$$\text{minimize } S_{oM}$$
$$\text{subject to: } \begin{cases} S_{o1} > C_1 \\ S_{o2} < C_2 \\ \vdots \\ S_{oM-1} > C_{M-1} \end{cases} \tag{4.8}$$

We use the algorithm depicted in Fig. 4.5 to solve Equation (4.8) using AnGeL.

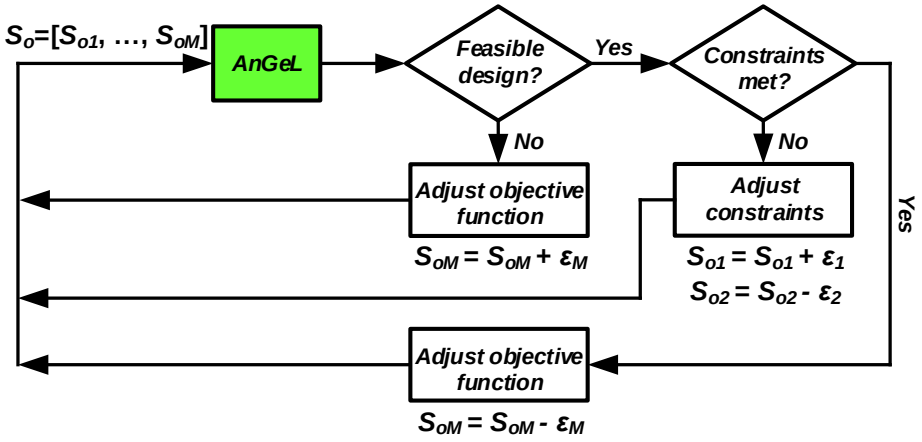


Figure 4.5: Constraints transformation algorithm flow chart based on Equation (4.8). The initial target is set as $S_o = [C_1, \ldots, C_{M-1}, C_M]$ where $C_M = \frac{S_{oM}^{high} + S_{oM}^{low}}{2}$.

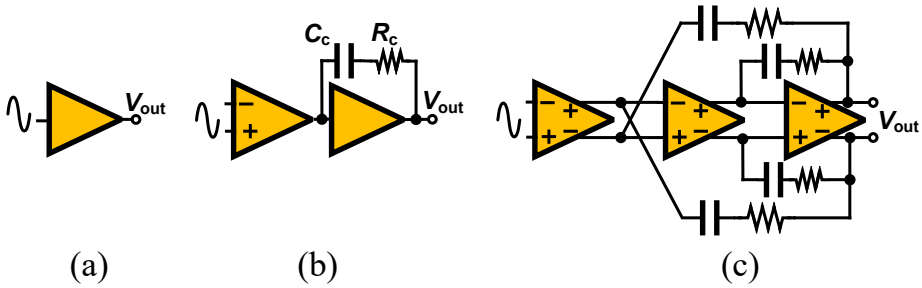


Figure 4.6: The supported high-level topologies for OPAMPs. All stages for one- and two-stage OPAMPs can have either single-ended or differential inputs and outputs. There is a DC-biasing sub-circuit before the second stage gain block that is not shown here. (a) One-stage OPAMP. (b) Two-stage OPAMP. (c) There-stage OPAMP.

At each iteration, $S_o = [S_{o1}, \ldots, S_{oM}]$ is given to AnGeL as the overall specifications where the values of $S_{o1}, \ldots, S_{oM}$ are determined by the algorithm. We use the idea of binary search [205] for optimizing the objective function, S_{oM}. Initially, $S_o = [C_1, \ldots, C_{M-1}, C_M]$ is given where C_M is the average of supporting values for S_{oM}, *i.e.* $C_M = \frac{S_{oM}^{high} + S_{oM}^{low}}{2}$. If AnGeL is able to design a circuit with the given specifications, it means the design is feasible and we check if the constraints are met as the next step. Otherwise, this means the chosen value for the objective function is too small. Therefore, a bigger value *i.e.* $S_{oM} = S_{oM} + \epsilon_M$, will be targeted for the next iteration where $\epsilon_M > 0$.

Sub-circuit	Specification	Min	Max	Average	SD
Gain block	BW [GHz]	0.001	15	0.17	0.68
	Gain [dB]	-0.89	53.2	33.48	9.46
	Power [mW]	0.02	28.8	2.55	2.29
	Noise [mV]	0.009	1.31	0.09	0.06
	Swing [V]	0	0.95	0.62	0.21
	PM [°]*	-121	170	99.8	35.4
	GM [dB]**	-20	20	3.2	4.5
DC-biasing	V_{out} [V]	0.25	0.8	0.55	0.12
	C_{in} [μF]	5	99.6	40.1	10.5
	R_{out} [kΩ]	35	400	250	50
Current source	Current [mA]	0.01	20.4	1.6	1.5

Table 4.2: Statistics of tested specifications for different OPAMP sub-circuits.

* Phase margin; PM is defined only for two-stage OPAMPs.
** Gain margin; GM is defined only for two-stage OPAMPs.

To check if the constraints are met, the output of AnGeL is simulated either by SPICE or by AnGeL's functional estimator models. If the constraints are met, the target value of the objective function will be reduced *i.e.* $S_{oM} = S_{oM} - \epsilon_M$. Otherwise, the constraints that are not met would be adjusted. For this purpose, target specifications are increased (if $S_{oi} > C_i$) or decreased (if $S_{oi} < C_i$) to give more margin for constraint satisfaction in the next iteration. For example, in Equation (4.8), if S_{o1} and S_{o2} are not met, $S_{o1} = S_{o1} + \epsilon_1$ and $S_{o2} = S_{o2} - \epsilon_2$ in the next iteration. The maximum number of iterations for adjusting the constraints and the objective function, as well as the values of $\epsilon_i > 0 \ \forall i \in \{1, \ldots, M\}$ are given by users.

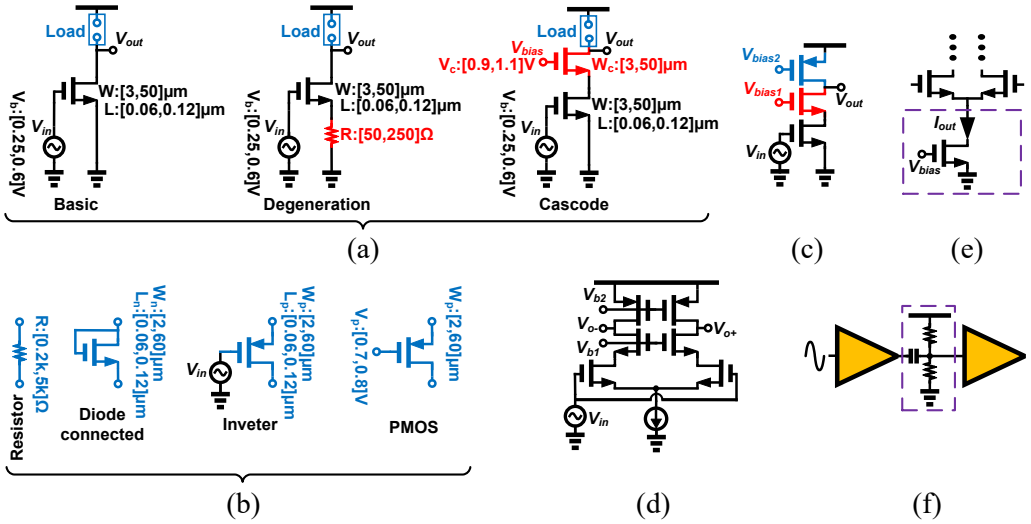


Figure 4.7: The supported sub-circuits for OPAMPs. (a) Gain block sub-circuit body structures; The range of supported design parameters is written next to each in the form of [low, high]. (b) Gain block sub-circuit loads; The range of supported design parameters is written next to each in the form of [low, high]. (c) An example of a body structure with a load. (d) An example of the differential mode of a single-ended topology (circuit (c)). (e) The topology of the current source type sub-circuit. (f) The topology of DC-biasing type sub-circuit.

4.5 Evaluation

In this section, the implementation of one-, two-, and three-stage OPAMPs as well as filters using AnGeL is evaluated as examples. The performance of the synthesized circuits by AnGeL is validated by SPICE simulations. The design parameters include all transistor sizes (W, L, v_{bias}) as well as capacitor and resistor values. All the SPICE simulations are in the 55nm technology node and based on the pre-layout parasitic analysis. Going from pre-layout parasitic to post-layout can be done using transfer learning [193], which is out of the scope of this work. It should be noted that in the post-layout, the value of our work would be even shown more as more time would be saved using our method in comparison to the conventional approaches. Also, all the NN models are built using the TensorFlow platform with the Adam optimizer. The learning rate is set to 0.001 and RELU is used as the activation function for all

Sub-circuit	Simulations #*	Unlabeled generated data #
Single-ended gain block	7,350	16,400
Differential gain block	450	23,300
Current source	450	0
DC-biasing	250	0

Table 4.3: Number of simulations and generated data using the topology-combiner model for single-stage OPAMPs.

* #: Number.

Specification	BW	Gain	Power	Noise	Swing
MAPE	0.070	0.029	0.049	0.051	0.038

Table 4.4: MAPE of the determined specifications by the topology combiner. More than 650 instances from 17 different single-stage OPAMP topologies are tested for each specification.

hidden layers. In order to avoid overfitting, the idea of early stopping [141] with the patience parameter of 150 is implemented. For this purpose, 10% of the data are used for validation during the training phase. In order to validate the results properly, a random separate test set with the size of 10% of the training set is used. Moreover, Scikit-learn is used for the training and testing of all random forest and SVM models. We use [206] for implementing PSO. All the training and testing of our models are run on a server with an NVIDIA GA102 GPU.

4.5.1 OPAMPs

4.5.1.1 Database Generation & Overview

Fig. 4.6 demonstrates the supported high-level topologies. The supported sub-circuits (*i.e.* gain block, current source, and DC-biasing) are shown in Fig. 4.7. For the gain block sub-circuits, all combinations of 3 body structures with 4 loads that are shown in Fig. 4.7(a), and (b) are supported in the single-ended and differential

Work	CCI-NN [177]			CCI-NN + TC*			AnGeL		
Number of stages	1	2	3	1	2	3	1	2	3
Data per topology	2,000	4,200	6,200	355	750	1,200	355	20	6

Table 4.5: Required number of labeled data per topology comparison between the state-of-the-art works to achieve an average MAPE of 0.045 in estimating the functionality of OPAMPs.

* TC: Topology Combiner.

modes. So, in total, $3 \times 4 \times 2 = 24$ different topologies are supported for the gain block sub-circuits. The range of supported design parameters is written next to each in the form of [low, high] in Fig. 4.7(a), and (b). The specifications for such DC-biasing and current source sub-circuits are determined by the associated gain stage through the feedback loop that is explained in Fig. 4.3. The supporting specifications statistics for each circuit type are summarized in Table 4.2.

From twelve supporting single-ended gain block sub-circuits in Fig. 4.7(a)-(b), the topology-combiner model is used for generating the dataset of six of them *i.e.* the degeneration and Cascode body structures with all loads except the resistor. Moreover, the topology-combiner model generates the datasets of all twelve differential gain stages except the one with the basic body and resistive load. The main and accessory NNs in our topology combiner model have 3 hidden layers with 64 nodes. As indicated in Table 4.3, for single-ended and differential gain blocks, the generated data are 2.23x and 51.74x more than the labeled data (simulations), respectively. Assuming each SPICE run takes 4s, that results in a time savings of more than 44h. The reason that the required number of labeled data for complicated differential sub-circuits is less than single-ended, is that they are generated by combining single-ended topologies. This shows the beauty of our topology combiner model that when topologies get more complicated, we need less labeled data to process them.

Testing on more than 650 instances from 17 different topologies for 5 specifications shows the topology combiner has an average Mean Absolute Percentage Error (MAPE)

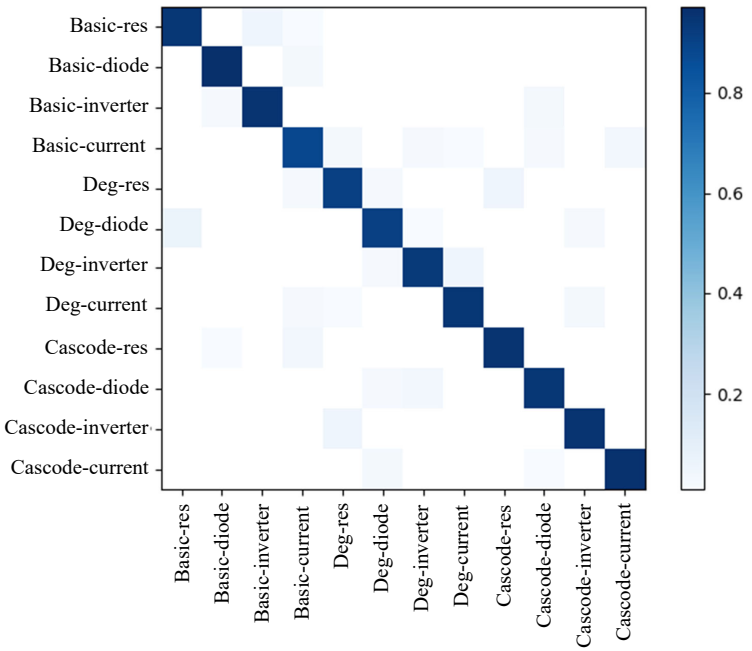


Figure 4.8: The confusion matrix of OPAMPs topology selection. The NN model with [128, 128, 128, 128] nodes is used for this purpose which gives the highest accuracy among the evaluated methods.

of 0.047. MAPE is calculated as $\sum_{i=1}^{n} \frac{1}{n} |\frac{y_{ti} - y_{pi}}{y_{ti}}|$, where y_{ti} and y_{pi} are the i^{th} true and predicted instance, respectively and n is the total number of instances. Table 4.4 lists MAPE of each specification.

The training data for all of the following evaluations are gathered from both labeled and unlabeled generated data. In order to validate the results properly, a random separate test set is built by SPICE runs.

4.5.1.2 Overall Topology Decider

In total, 14,000 labeled samples are used for the training of the net specification calculator models in two high-level topologies *i.e.* Fig. 4.6(b) and (c). 250,000 pseudo samples of two-, and three-stage OPAMP specifications are generated and fed to the specification breaker model.

The net specification calculator and specification breaker models enable us to

Optimization engine		Parameters	Estimator	Runtime[s]	MAPE
Global	Local				
NN	PSO	Iterations:10	NN	34.15	0.059
PSO	PSO	Iterations:50, Particles: 3,000	NN	100.4	0.09
			SPICE	7,800,000	0.018
SA	SA	Iterations: 5,000	NN	2,562	0.17
			SPICE	237,465	0.04
BO	BO	Iterations:500	NN	510	0.179
			SPICE	26,912	0.08

Table 4.6: Average runtime and MAPE comparison of different sub-circuit sizing models.

support $24 \times 24 = 576$ two-stage OPAMP topologies as we support 24 topologies for each stage. Moreover, we support $12 \times 12 \times 12 = 1,728$ three-stage OPAMP topologies (we only support differential gain for three-stage OPAMPs). This means with only having datasets of 7 topologies from labeled data, we support 2,328 different one-, two, and three-stage OPAMP topologies. The number of design parameters in such OPAMPs varies between 4-31.

The required number of labeled training data of AnGeL is compared with the state-of-the-art work [177]. Hassanpourghadi *et al.* [177] divide the circuit into sub-circuits and implement a Circuit-Connectivity-Inspired ANN (CCI-NN) model. However, they use neither the idea of the topology combiner nor the net specification calculator model. In order to measure the usefulness of the net specification calculator model in multistage circuits, we also analyze when CCI-NN is integrated with the topology combiner. For this purpose, we test all works on 3,500 instances while all have the same average MAPE of 0.045 in estimating the functionality of circuits. AnGeL that integrates both the topology combiner and net specification calculator model requires 22,500 labeled data in total which means it needs around 5.5 labeled data per topology for covering 1728 three-stage OPAMP topologies. Using the net specification calculator model, we estimate the functionality of multistage OPAMPs with different

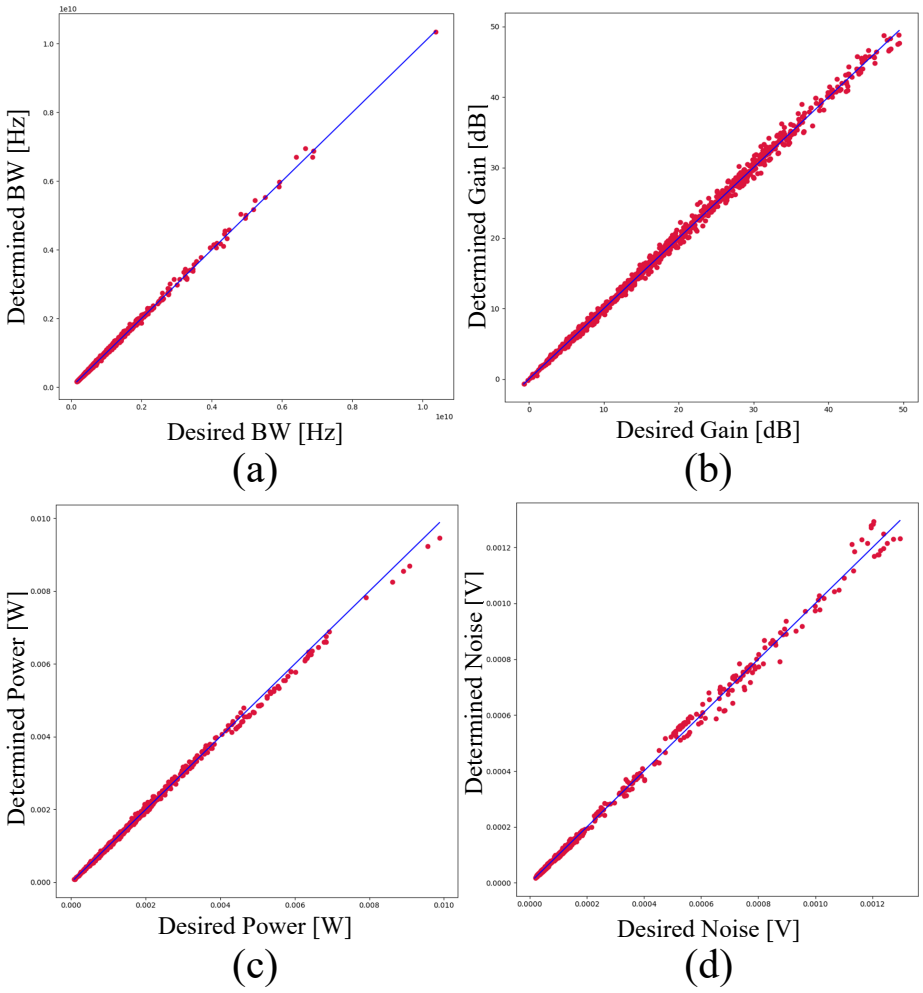


Figure 4.9: Desired vs determined values by running the entire AnGeL platform for different specifications. More than 1,200 instances including one-, two-, and there-stage OPAMPs are tested. (a) BW. (b) Gain. (c) Power. (d) Noise.

topologies without requiring a separate dataset for each topology. As summarized in Table 4.5, for three-stage OPAMPs, CCI-NN [177] and CCI-NN integrated with the topology combiner need 1,090x and 200x more labeled data than AnGeL, respectively.

4.5.1.3 Sub-circuit Topology-Selector

Random forest, SVM (linear, polynomial, sigmoid, and RBF kernels), as well as NN classifier models, are analyzed for the topology-selector. For the gain block sub-circuits, there are 40,000 and 4,000 samples for the training and test sets, respectively.

Spec	BW	Gain	GM	PM	Noise	Swing	Power
1^{st} stage desired	1.35 GHz	16.5 dB	-*	-*	0.15 mV	0.4V	1.5 mW
1^{st} stage AnGeL	1.4 GHz	17.1 dB	-	-	0.14 mV	0.43V	1.6 mW
2^{nd} stage desired	0.9 GHz	17.5 dB	-	-	0.15 mV	0.50V	2.4 mW
2^{nd} stage AnGeL	0.94 GHz	17.3 dB	-	-	0.15 mV	0.53V	2.2 mW
3^{rd} stage desired	1.8 GHz	15 dB	-	-	0.15 mV	0.75V	2.4 mW
3^{rd} stage AnGeL	1.7 GHz	15 dB	-	-	0.15 mV	0.73V	2.3 mW
Overall desired	15 MHz	49 dB	5 dB	60°	0.23 mV	0.75V	6.3 mW
Overall AnGeL	15.5 MHz	49.4 dB	5.3 dB	62.1°	0.22 mV	0.73V	6.1 mW

Table 4.7: An example of the OPAMP specifications given to AnGeL and the simulated specifications of the associated generated circuit by AnGeL.

* GM & PM is defined only for the overall circuit not for the sub-circuits.

Our evaluation shows the NN with four layers and 128 nodes at each layer gives an accuracy of 92.9% which is the highest among the compared methods. This NN is used as the topology-selector model. Fig. 4.8 shows the confusion matrix of the predicted topologies using this model. It should be noted that even if a topology is not classified correctly, it does not mean that AnGeL cannot meet the specifications with that topology. This is because there may be more than one topology that can achieve the same specifications.

4.5.1.4 Sub-circuit Sizing

As it was mentioned earlier, we use NN and PSO as the global and local optimization engines, respectively. Table 4.6 summarizes the runtime and MAPE as well as the parameters of different methods when tested on more than 30 samples. Our approach results in the minimum runtime while it has a reasonable MAPE of 0.059. In fact, the runtime of our approach is 2.93x - 228,400x faster than other methods while it has better or comparable MAPE. The NN for the sub-circuit sizing model has 3 hidden layers with 64 nodes. For a better comparison, training time is taken into account too, which is 34,000s assuming each SPICE run takes 4s on average for generating

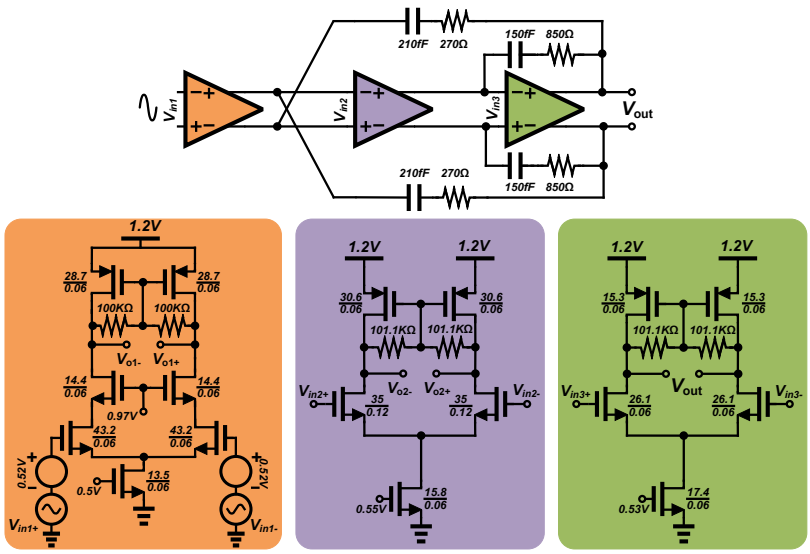


Figure 4.10: The generated OPAMP circuit by AnGeL with its sizing to meet Table 4.7 specifications. Size of transistors are shown as $\frac{W(\mu)}{L(\mu)}$.

each of the 8,500 labeled data. When the estimator is the NN, it means the same training set as ours is used for other approaches (*i.e.* when both global and local engines are PSO, SA, and BO). So, our approach still outperforms them. However, considering both training and deployment time on a single run when SPICE is used as the estimator, BO as both global and local engines is faster than our method. In such conditions based on Table 4.6, even though our MAPE is better, BO (which has the lowest SPICE runtime) takes 26,912s while our method takes 34,034s. In fact, the NN estimator shows its actual advantage when the tool is used multiple times for generating circuits in the inference mode. For instance, if we run the tool 10 times, our approach takes 34,341.5s while BO takes 269,120s which means we save 65.21 hours during these runs using our method.

4.5.1.5 AnGeL Entire Platform

The entire AnGeL platform, from selecting the topology to sizing, is tested on more than 1,200 samples including one-, two-, and there-stage OPAMPs. The samples

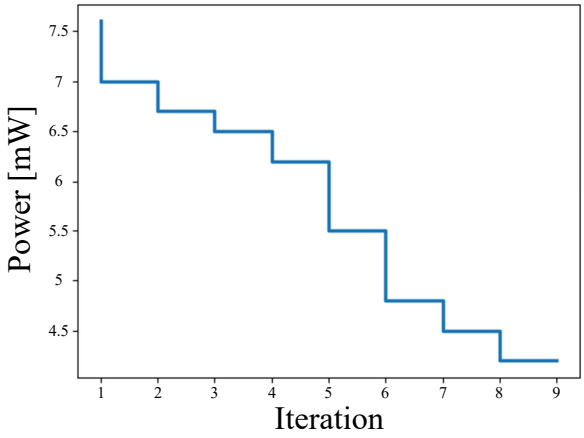


Figure 4.11: Power at each iteration step while constraints in Equation (4.9) are satisfied.

Specification	Min	Max	Average	SD
Pass-band BW (ω_{3dB}) [MHz]	10^{-4}	46.5	0.3	0.4
Stop-band frequency [MHz]*	1.5×10^{-6}	2.5×10^{3}	0.12	0.19
Gain [dB]	-13.4	18.6	2.3	5.2
Power [mW]	0.02	24.5	1.9	1.8
Noise [mV]	10^{-10}	8	2.2	4.9
Overshoot [dB]	6×10^{-13}	12.1	0.32	1.26
Group delay [μs]	1.5×10^{-4}	20.7	0.28	1.3

Table 4.8: Statistics of tested specifications for filters.

* The frequency that the gain has 40dB attenuation compared to the ω_{3dB} pass-band frequency.

cover Table 4.2 specifications. The circuits that are generated by AnGeL for the given specifications are simulated to evaluate the performance. Fig. 4.9 shows the desired vs determined values for different specifications. An average MAPE of 0.027 is achieved in total.

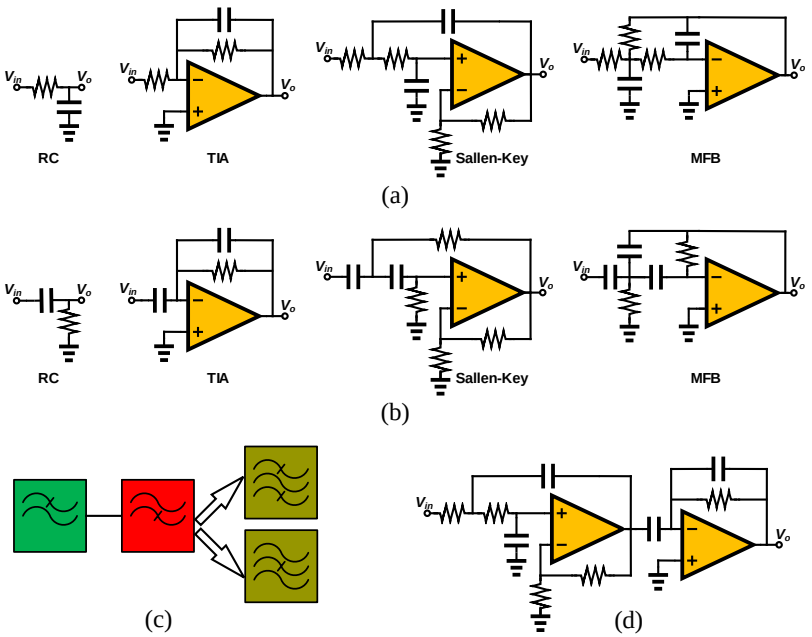


Figure 4.12: The supported sub-circuits for filters. (a) Low-pass filters. (b) High-pass filters. (c) Cascading LP and HP filters results in band-pass or band-stop filters based on the relative position of the corner frequency of HP/LP filters. (d) An example of a band-pass/band-stop filter which is created by cascading a Sallen-Key LP with a TIA HP.

An example of these 1,200 desired specifications is as follows:

$$\text{minimize power}$$
$$\text{subject to:} \begin{cases} \text{BW} > 15\ \text{MHz}, \text{Gain} > 48\ \text{dB}, \\ \text{GM} > 0\ \text{dB}, \text{Noise} < 0.25\ \text{mV}, \\ \text{PM} > 60^{\circ}, \text{Swing} > 70\ \text{mV}. \end{cases} \tag{4.9}$$

Fig. 4.10 demonstrates the generated circuit by AnGeL in one of the iterations for minimizing the power while the constraints are satisfied. There are 27 design parameters in this design. Table 4.7 summarizes the desired and determined specifications of each stage as well as the overall circuit performance. Each stage desired specifications are generated by the overall topology decider module. The overall circuit desired specifications are determined by the constraint transformer module (Fig. 4.5). Fig. 4.11

Work	CCI-NN [177]	Hierarchical CCI-NN	AnGeL
Non-ideal OPAMP models	1,012	1,012	212
Transistor-level OPAMPs models	4,800	0.0253	0.0032

Table 4.9: Required number of labeled data per topology comparison between the state-of-the-art works to achieve an average MAPE of 0.06 in estimating the functionality of filters.

Model	Hyperparameters	Accuracy	
		HP & LP	Band-pass & band-stop
Random Forest	n_estimators = 10	99.68%	99.41%
	n_estimators = 100	99.71%	99.70%
	n_estimators = 400	99.69%	99.63%
SVM	Linear, $C = 1$	92.16%	86.54%
	Polynomial, $C = 1$, $\gamma = 1$, degree = 3	95.44%	93.12%
	Sigmoid, $C = 1$, $\gamma = 1$	40.17%	27.95%
	RBF, $C = 1$, $\gamma = 1$	95.45%	90.98%
DNN	NN layers: [64, 64, 64]	99.49%	94.40%
	NN layers: [128, 128, 128]	99.42%	95.22%
	NN layers: [128, 128, 128, 128]	99.74%	94.83%

Table 4.10: Model accuracy comparison for the filter topology-selector. There are 8 topologies for HP & LP and 32 topologies for band-pass & band-stop filters in total.

shows how the power gets smaller at each iteration. 4.2mW is the minimum achieved power while all constraints are met.

4.5.2 Filters

4.5.2.1 Database Generation & Overview

In general, there are two types of filters: passive and active. Passive filters use only passive elements *e.g.* resistors, inductors, and capacitors, while active filters use active components such as OPAMPs in addition to passive elements. Fig. 4.12 shows

Spec	Pass-band BW	Stop-band frequency	Gain	Overshoot	Group delay
MAPE	0.044	0.065	0.056	0.068	0.067

Table 4.11: MAPE of determined specifications by running the entire AnGeL platform. More than 1,200 filter instances are tested.

Specification	ω^{*}_{3dB} LP [MHz]	ω_{3dB} HP [KHz]	ω^{+}_{s} LP [MHz]	ω_s HP [Hz]	Gain [dB]	Power [mW]	Noise [mV]	Overshoot LP [μdB]	Overshoot HP [mdB]	Group delay [μs]
Band-pass desired	15	2	200	20	13	15	6	35	1	3
LP desired	15	-	170	-	7	8	3.5	35	-	$6 \times 10{-}3$
LP AnGeL output	15.9	-	172.7	-	7.7	7.56	3.3	32	-	$6.4 \times 10{-}3$
HP desired	-	2	-	20	6	7	3	-	1	3
HP AnGeL output	-	1.9	-	18.9	6.03	6.8	2.9	-	1.09	3.2
Band-pass AnGeL	15.9	1.9	205.1	18.9	13.73	14.36	5.9	32	1.09	3.2

Table 4.12: An example of the filter specifications given to AnGeL and the simulated specifications of the associated generated circuit by AnGeL.

* The 3dB bandwidth of pass-band frequencies.

+ The frequency that the gain has 40dB attenuation compared to the ω_{3dB} pass-band frequency.

the supported filter sub-circuits. There are 4 topologies for each of Low-Pass (LP) and High-Pass (HP) filters. Moreover, band-pass and band-stop filters are achievable by cascading LP and HP filters as illustrated in Fig. 4.12(c). So, there are $4 \times 4 = 16$ different topologies for each of band-pass and band-stop filters which leads to support 40 filters in total. The specifications for the OPAMP in active filters are determined by the associated filter stage through the feedback loop that is explained in Fig. 4.3. Taking one- and two-stage OPAMP topologies into account results in supporting more than 6,400,000 topologies. The number of design parameters varies between 2-42. The supporting specifications statistics for filters are listed in Table 4.8.

In total, 8,500 simulations are performed as the labeled dataset for all filters assuming a non-ideal OPAMP model is used in active filters. By a non-ideal OPAMP model we mean an OPAMP with limited gain and bandwidth whose characteristic is

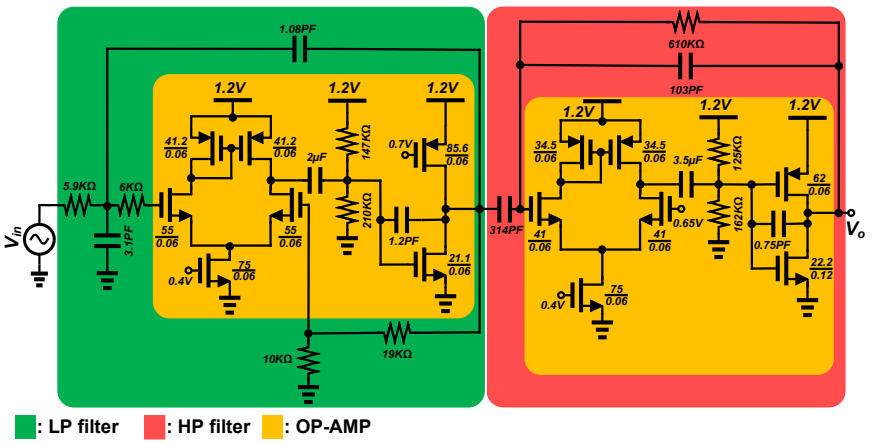


Figure 4.13: The generated band-pass filter by AnGeL with its sizing to meet Table 4.12 specifications. Size of transistors are shown as $\frac{W(\mu)}{L(\mu)}$.

modeled however, it is not made by circuit elements *e.g.* transistors. Moreover, 32,000 unlabeled data are generated using the topology-combiner model. The labeled data is used for generating the database of LP and HP filters. The datasets of band-pass and band-stop filters are generated by the topology-combiner model (except when RC is used in both stages of the band-pass or band-stop filters). Testing on more than 800 instances from 32 different topologies for 5 specifications shows the topology combiner has an average MAPE of 0.04.

Similar to OPAMPs, the required number of labeled training data of AnGeL is compared with the state-of-the-art works while all approaches achieve an average MAPE of 0.06 in estimating the functionality of filters testing on more than 1,200 samples. We also, analyze when the idea of the hierarchical design is used in CCI-NN meaning the specifications of OPAMPs in active filters are determined based on the specifications of the filter. This hierarchical approach allows for smaller training set as it breaks down the circuit. As it is summarized in Table 4.9, when actual transistor-level OPAMPs are used, hierarchical CCI-NN requires significantly less data than the normal CCI-NN [177]. Moreover, AnGeL requires 7.9x less labeled data in comparison with the hierarchical CCI-NN.

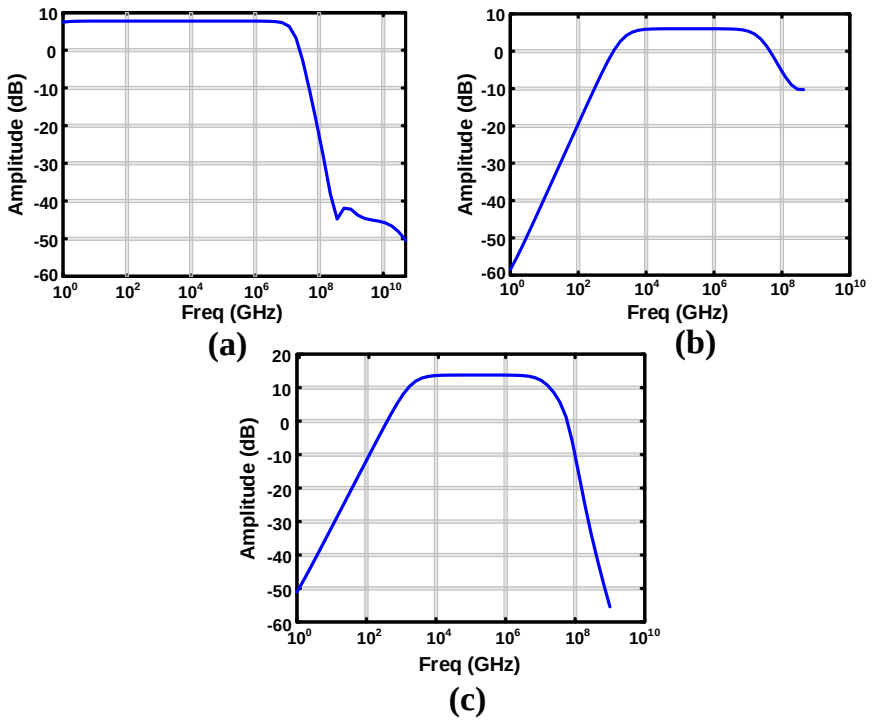


Figure 4.14: The frequency responses of Fig. 4.13 filters to meet Table 4.12 specifications. (a) LP filter. (b) HP filter. (c) Overall band-pass filter.

4.5.2.2 Sub-circuit Topology-Selector

There are 35,000 and 3,500 samples for the training and test sets, respectively. Some specifications of band-pass/band-stop filters are different from LP/HP filters so, they are analyzed separately. The accuracy of different models for topology classification is summarized in Table 4.10.

4.5.2.3 AnGeL Entire Platform

Similar to OPAMPs, the entire AnGeL platform is tested on 1,200 samples which cover Table 4.8. Actual transistor-level OPAMPs that are created by AnGeL in Section 4.5.1 are used in all active filter samples. It should be noted that more than 35,000 are used for the training of the sub-circuit sizing module. The NN model of the sub-circuit sizing module has 3 hidden layers with 64 nodes. Table 4.11 summarizes the average MAPE for different specifications when the entire AnGeL is tested. An

average MAPE of 0.06 is achieved in total.

Fig. 4.13 demonstrates an example of a band-pass filter designed by AnGeL, which is among such 1,200 testing samples and it is generated to meet specifications that are shown in Table 4.12. AnGeL has selected a Sallen-key topology for the LP filter and a TIA topology for the HP filter. There are 31 design parameters in this design. The LP and HP stage desired rows in Table 4.12, are generated by the overall topology decider module. The average MAPE is 0.054. Fig. 4.14 also illustrates the frequency responses of each LP and HP filter as well as the overall band-pass filter of Fig. 4.13.

4.6 Conclusion

In this work, we first use neural networks to determine the behavior of complicated circuit topologies by combining the more simple ones and presenting a database. The number of labeled data in this dataset is **4.7x - 1090x** less than the state-of-the-art works while maintaining the same accuracy. Using this database, we propose a fully-automated analog circuit generator framework, AnGeL. AnGeL performs all the necessary schematic circuit design steps, from deciding the circuit topology to sizing using NN models. To validate our work, multiple designs are generated by AnGeL which includes one-, two-, and three-stage OPAMPs as well as different filters. The results show that the runtime of AnGeL is **2.9x - 75x** faster than the state-of-the-art works while maintaining the same accuracy.

CHAPTER V

FuNToM: Functional Modeling of RF Circuits Using a Neural Network Assisted Two-Port Analysis Method

Automatic synthesis of analog and Radio Frequency (RF) circuits is a trending approach that requires an efficient circuit modeling method. This is due to the expensive cost of running a large number of simulations at each synthesis cycle. Artificial intelligence methods are promising approaches for circuit modeling due to their speed and relative accuracy. However, existing approaches require a large amount of training data, which is still collected using simulation runs. In addition, such approaches collect a whole separate dataset for each circuit topology even if a single element is added or removed. These matters are only exacerbated by the need for post-layout modeling simulations, which take even longer. To alleviate these drawbacks, in this project, we present FuNToM, a functional modeling method for RF circuits. FuNToM leverages the two-port analysis method for modeling multiple topologies using a single main dataset and multiple small datasets. It also leverages neural networks which have shown promising results in predicting the behavior of circuits. Our results show that for multiple RF circuits, in comparison to the state-of-the-art works, while maintaining the same accuracy, the required training data is

reduced by **2.8x - 10.9x**. In addition, FuNToM needs **176.8x - 188.6x** less time for collecting the training set in post-layout modeling.

5.1 Introduction

The growing demand for analog and Radio Frequency (RF) circuits, due to their broad applications, has led to a crucial need for automated circuit synthesis [130]. There are two classical automated synthesis approaches for analog and RF circuits, *i.e.* simulation- and model-based, which both need a circuit modeling tool. In the simulation-based approaches, the circuit modeling tool is invoked multiple times to evaluate the circuit and generate new parameter candidates to meet the desired specifications [177]. In addition, the modeling tool is used to generate the training set of the model-based approaches. Since both methods invoke the modeling tool frequently (especially the simulation-based), using the common modeling tool, *i.e.* SPICE simulation, is drastically time-consuming. Furthermore, technology scaling has caused much longer run times in both schematic and post-layout simulations [193].

Artificial Intelligence (AI) methods are an alternative to SPICE because of their speed and relative accuracy [207]. However, current approaches require a large amount of training data, which are still collected using simulation runs. So, they face the same time-consuming problem during training. Bayesian Model Fusion (BMF) is one of the common analog circuits functional modeling methods [92, 93, 208]. Neural Networks (NNs) also have shown promising results in all circuit design levels such as single-board computer, sizing, layout, System-on-Chip (SoC) design, and performance modeling [20, 101, 103, 192, 209].

Despite all these advances in functional modeling of circuits, current works [15, 64, 92, 93, 177, 192, 208, 210, 211] train directly from circuit design parameters (capacitance value of capacitors, size of transistors, etc.) to Performance of Interests (PoI). Therefore, such conventional modeling approaches require a unique training set for each topology,

because design parameters change if the topology changes. This means their whole training process needs to be redone even if a single element is added or removed from the circuit. Moreover, they assign an individual feature to each design parameter, which makes their models high-dimensional for modern complex analog and RF circuits. The matter is only exacerbated by the post-layout modeling, where collecting data takes longer.

To combat all the aforementioned challenges in circuit functional modeling, we propose an NN-assisted two-port analysis-based method, FuNToM. The goals of FuNToM are threefold: (a) decreasing the required number of new training sets after modifying a circuit topology; (b) further reducing the number of training sets for each topology; (c) achieving a fast, accurate RF functional modeling method.

In order to decrease the required number of new training sets after modifying a circuit topology, we propose a method to properly model multiple topologies while using a single main dataset and multiple small datasets. For this purpose, we divide the circuit into multiple Electrical-networks (E-networks) and analyze them modularly via NNs. An E-network is a collection of electrical components, *e.g.* capacitors, resistors, etc., that are interconnected [212]. To modularly analyze E-networks, we leverage the Scattering parameters (S-parameters) concept. The S-parameter is a well-known powerful concept in RF circuits that is defined using two-port analysis and describes the circuit behavior of E-networks. In other words, in two-port analysis, the S-parameter is a 2×2 matrix that summarizes the circuit behavior of the associated E-network, such as input/output return losses, and insertion loss/gain [213]. Therefore, the S-parameter can be replaced with the associated E-network in the circuit analysis.

The main advantage of modularly modeling the circuit is its generality. In our case, by passing the S-parameters to the NN, the NN learns the relationships between the general format of E-networks' S-parameters and the PoI. Since the S-parameter works as a wrapper around the E-networks, the NN accurately determines the functionality

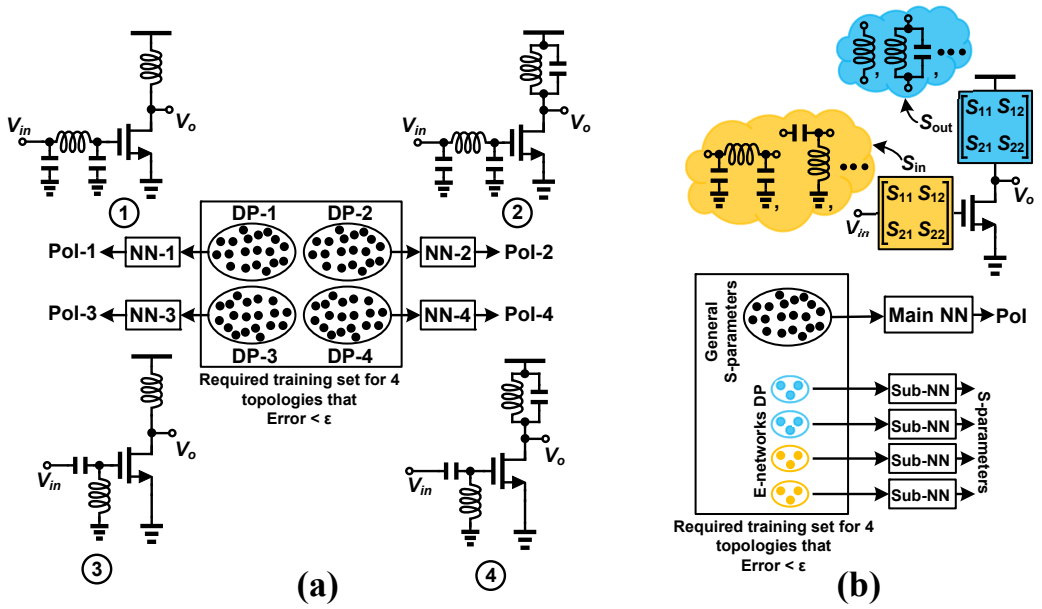


Figure 5.1: An overview, concept, and the impact on the training set of (a) the conventional analog and RF circuits functional modeling method vs (b) the proposed modular S-parameters-based approach. The conventional method needs a whole separate dataset and model for each circuit topology while the proposed approach works properly for many circuit topologies with a single general format E-networks' S-parameters dataset.
$\begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix}$: S-parameter matrix.

DP: Design parameters.

of the circuit with any E-networks' topology. In other words, training via S-parameters is a one-time process which works properly for many circuit topologies because of its modularity.

There are two types of NN models in our approach: 1) One main circuit NN for determining the PoI using S-parameters of E-networks; 2) multiple sub-NNs that each determines the S-parameters of the associated E-network using the circuit design parameters within such E-network. Creating new datasets for sub-NNs is the only required modification in the dataset of the proposed approach when the circuit topology changes as the training set of the main NN works for all topologies. However, since the circuit is divided into multiple E-networks and each E-network has a separate dataset,

the size of such datasets is small because each deals with only a few design parameters. This dividing of the circuit is another advantage of the proposed method that is especially prevalent when going from schematic to the post-layout modeling. Instead of collecting a new large dataset by running time-consuming post-layout simulations for the whole circuit with many design parameters, it is only needed to gather small datasets for the E-networks, each with only a few design parameters. Fig. 5.1(a) demonstrates the conventional modeling method using NNs, while Fig. 5.1(b) shows the big picture of the proposed approach. As it is shown, our approach requires much less training data than the conventional method to have the same accuracy in modeling of four different topologies. Note that the circuit elements (design parameters) that are not part of E-networks, *e.g.* the transistor in Fig. 5.1(b), are also considered in our proposed model as explained with more details in Section 5.3.1. FuNToM supports all RF circuits, including active and passive components, as multiple of them are evaluated in Section 5.4.

We use the Circuit-Connectivity-Inspired-NN (CCI-NN) structure [177] in our main model to further reduce the number of training sets for each topology. As stated by Hassanpourghadi *et al.* [177], CCI-NN structure requires less training data in comparison with the Fully-Connected NNs (FC-NNs). Moreover, by condensing the design parameters using S-parameters, we may have a smaller design space than the conventional approach. If the number of design parameters in each E-network is more than the indexes in the S-parameter matrix, the NN model of the proposed approach will have fewer features than the conventional method and hence, less training sets will be required to have the same accuracy. This mainly has an impact on large circuits.

To validate our proposed method, FuNToM is tested by modeling the functionality of multiple phase shifters at the schematic level as well as multiple two-stage Low-Noise-Amplifiers (LNAs) at both schematic and post-layout levels. Our results show that using FuNToM, the required training set is reduced by a factor of **2.8x - 10.9x**

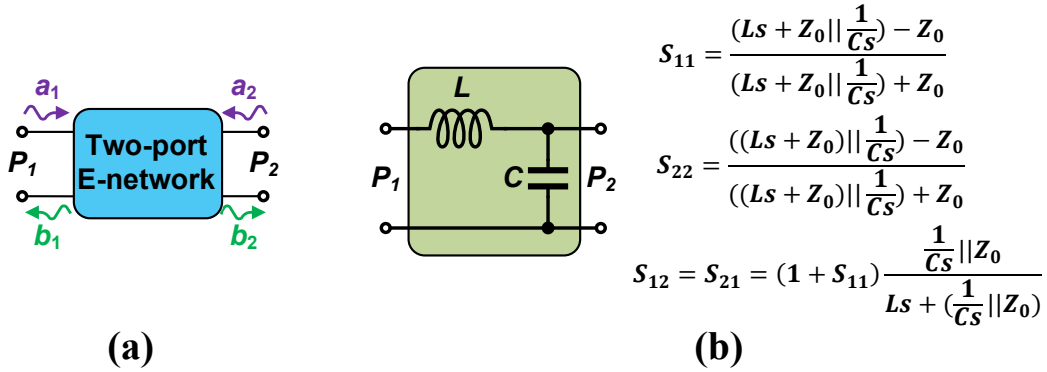


Figure 5.2: (a) Excited and incident waves in a two-port E-network which are used for calculating S-parameters. (b) An example of a simple L-C network with the calculated S-parameters. s = j2πf where f is the frequency. Z_0 is the reference impedance (the impedance of excitation ports) which equals to 50 Ω in most cases.

while maintaining the same accuracy in comparison to the state-of-the-art works. Also, the time for collecting the training set in the post-layout modeling using FuNToM is **176.8x - 188.6x** faster. The results illustrate that FuNToM achieves an average R2-score of **0.95** when tested on 3,200 samples.

The main contributions of this project can be summarized as follows:

- Proposing FuNToM, an efficient RF functional modeling method which achieves an average R2-score of **0.95** when it is tested on 3,200 samples with different topologies and specifications.
- Reducing the number of required training set points compared to the state-of-the-art works by **2.8x - 10.9x** by using two-port analysis and dividing the circuit into multiple E-networks.
- Decreasing the time for collecting the training set in the post-layout modeling by **176.8x - 188.6x** compared to state-of-the-art works.

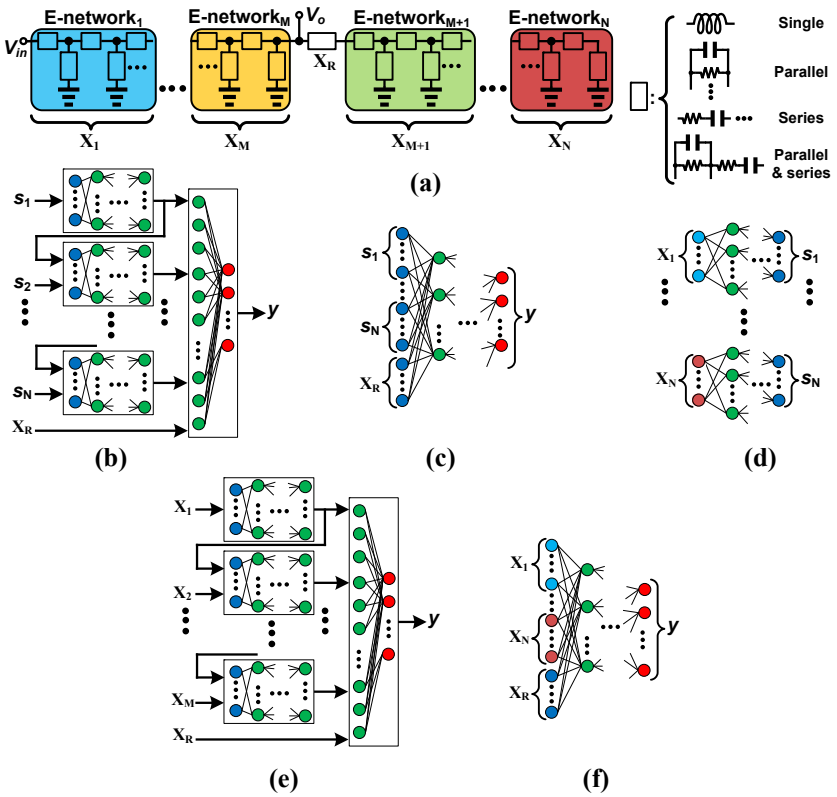


Figure 5.3: (a) General circuit division into E-networks. (b) Main NN in the proposed approach with CCI-NN structure. (c) Main NN in the proposed approach with FC-NN structure. (d) Sub-NNs in the proposed approach. (e) The original CCI-NN model structure [177] with the circuit design parameters as inputs. (f) The conventional FC-NN model in the analog and RF functional modeling with the circuit design parameters as inputs.
s_i: S-parameters of the i^{th} E-network; $\mathbf{x_i}$: Design parameters vector of the i^{th} E-network; $\mathbf{x_R}$: A vector of the circuit's design parameters that are not included in any E-networks.

5.2 Background & Related Work

5.2.1 Analog and RF Circuits Functional Modeling

The goal in analog and RF circuits functional modeling is to approximate a PoI vector (y) of a given circuit by a function ($\hat{f}$) of the design parameters vector ($\mathbf{x}$), while the modeling error (e) is minimized. For this purpose, the SPICE is invoked as

the ground truth function (f). In other words,

$$y = f(\mathbf{x}, t) \tag{5.1}$$

$$\begin{cases} y = \hat{f}(\mathbf{x}, t) + e, \\ \text{subject to: minimize } e. \end{cases}$$

Note that t is used to account for time/frequency variant PoIs. Power gain is an example of a PoI for an LNA, while resistance values of resistors are examples of the design parameters vector. The advantage of the functional modeling methods over SPICE is that once they get trained, they can determine the functionality of circuits, even when the design parameters change, in near zero time. This is because the circuit behavior is stored as weights for the functional modeling methods.

For minimizing the modeling error, different models, such as posynomials regression, genetic programming, BMF, and NN, have been studied [64, 92, 192, 210]. The fusion in BMF means passing information (prior knowledge) from the schematic model to the post-layout model to reduce the expensive post-layout performance modeling cost [130]. This passed information is then combined with a few post-layout training data points to solve the model coefficients via Bayesian inference [98].

The other goal of the functional modeling methods is to minimize the number of times that SPICE is invoked since it is time-consuming. Several studies have attempted to reduce the number of training samples while maintaining the same error, such as sparse regression, Co-Learning BMF (CL-BMF), hierarchical CL-BMF, and Circuit-Connectivity-Inspired NN (CCI-NN) [15, 177, 211]. In hierarchical Co-Learning BMF, the circuit is partitioned into multiple stages and CL-BMF is applied to each stage [15]. In this regard, they consider lower dimensional models and as a result the training set size is decreased. Among all aforementioned works, NNs have shown the lowest modeling error however, they need a large training set [177]. Similar to [15],

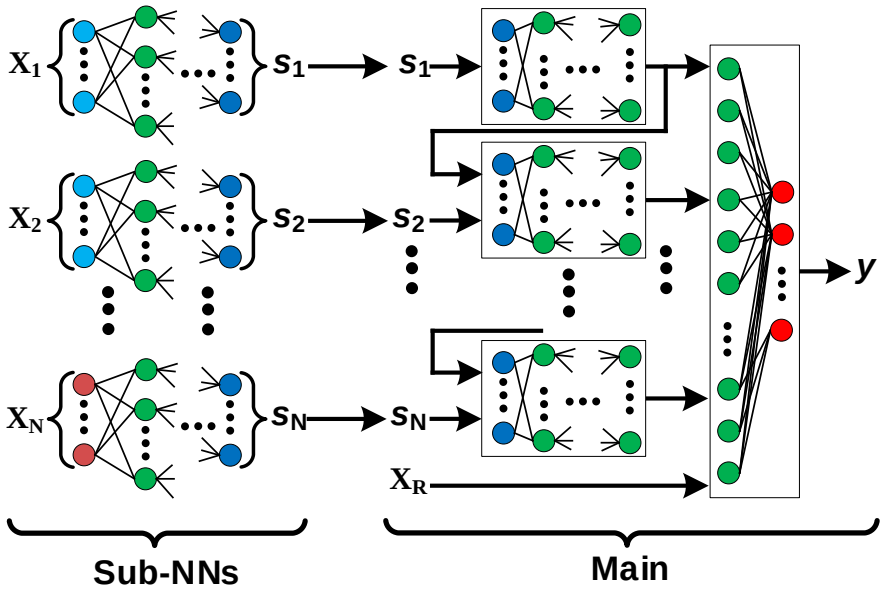


Figure 5.4: Final FuNToM model by concatenating the sub-NNs and the main NN.
s_i: S-parameters of the i^{th} E-network.
$\mathbf{x_i}$: Design parameters vector of the i^{th} E-network.
$\mathbf{x_R}$: A vector of the circuit's design parameters that are not included in any E-networks.

Hassanpourghadi *et al.* [177] break down a circuit into multiple stages, but they apply NN instead of CL-BMF for modeling. Zhao *et al.* [214] first, model the transistor using NNs. By leveraging such a model, they perform DC and AC modeling for the circuit. Despite of the accuracy and efficiency, there is no word about the post-layout modeling.

5.2.2 Two-port Analysis & S-parameters

The two-port analysis is used to determine the response of a two-port E-network against the applied signals to its terminals. The circuit behavior of E-networks is determined using two-port analysis [213]. Using two-port analysis, an S-parameter matrix is derived which is used for characterizing circuits. Fig. 5.2 shows the procedure of determining S-parameters for a two-port E-network: each port is excited by incident waves, a_1 and a_2, and the reflected voltage waves, b_1 and b_2, are measured. The

reflected voltage waves are a function of the incident waves:

$$\begin{aligned} b_1 &= S_{11}a_1 + S_{12}a_2, \\ b_2 &= S_{21}a_1 + S_{22}a_2, \end{aligned} \tag{5.2}$$

where $S = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix}$ is called the S-parameter matrix.

S-parameters are calculated by setting a_1 and a_2 in Equation (5.2) to zero one by one and getting the ratios of the reflected waves over the incident waves as follows:

$$\begin{aligned} S_{11} = \frac{b_1}{a_1}|_{a_2=0}, S_{12} = \frac{b_1}{a_2}|_{a_1=0}, \\ S_{21} = \frac{b_2}{a_1}|_{a_2=0}, S_{22} = \frac{b_2}{a_2}|_{a_1=0}. \end{aligned} \tag{5.3}$$

It is noteworthy that setting $a_i = 0$ means that the i^{th} port is terminated with a Z_0 impedance so that the reflection from that port (a_i) becomes zero. Z_0 equals 50 Ω in most cases. Fig. 5.2(b) demonstrates an example of S-parameters extraction of a simple L-C E-network. Based on the reciprocity theorem, $S_{12} = S_{21}$ for almost all passive E-networks such as the network mentioned in Fig. 5.2(b). Moreover, for the symmetric E-networks (see Fig. 5.6(b)), $S_{11} = S_{22}$.

5.3 Proposed Approach

5.3.1 Neural Network Models

Breaking down the top-level circuit into multiple stages is used for simplifying f in Equation (5.1) and decreasing the number of training sets [177]. In our approach, we have gone further and we break down the circuit into multiple E-networks. There are two types of models in our approach: 1) One main model for determining the PoI using S-parameters of E-networks and design parameters that are not included in

Scenario	Main NN	Sub-NNs	Total
First scenario (Fig. 5.5(a))	1,800	400	2,200
Second scenario (Fig. 5.5(b))	450	3,200	3,650

Table 5.1: Required number of training data comparison between two scenarios of Fig. 5.5 for the main NN, sub-NNs, and total.

any E-networks; 2) multiple sub-models that each determines the S-parameters of the associated E-network using the circuit design parameters within such E-network. In other words:

$$y = \hat{g}([s_1, \ldots, s_N, \mathbf{x_R}], t) + e_1, \tag{5.4a}$$

$$s_i = \hat{h_i}(\mathbf{x_i}, t) + e_{2,i} \quad i = 1, \ldots, N, \tag{5.4b}$$

where s_i and $\mathbf{x_i}$ are the S-parameters and design parameters vector of the i^{th} E-network ($s_i = [S_{11,i}, \ldots, S_{22,i}]$), respectively assuming there are N E-networks. Indeed, $\mathbf{x_i}$ represents the value of circuit elements *e.g.* capacitors, inductors, etc., inside the i^{th} E-network. $\mathbf{x_R}$ is a vector of the circuit's design parameters that are not included in any E-networks *e.g.* the transistor in Fig. 5.1(b).

Fig. 5.3(a) shows a general circuit division into multiple E-networks while Fig. 5.3(b) and (c) illustrate two structures for our proposed main NN model. Both of them depict the NN representation of Equation (5.4a). Fig. 5.3(b) and Fig. 5.3(c) have CCI-NN and FC-NN structures, respectively. As it is shown, the CCI-NN structure [177] breaks a giant NN into smaller chunks while considering the sequential paths and concatenating all at a final 1-layer NN. It should be mentioned that the CCI-NN is trained with the same dataset as the FC-NN *i.e.* $[(s_1, \ldots, s_N, \mathbf{x_R}); y]$. We analyze both FC-NN and CCI-NN structures for our main NN and select the one which gives the best results. As stated by [177], CCI-NN requires less training data. Fig. 5.3(d) depicts our sub-NN structures which are representations of Equation (5.4b).

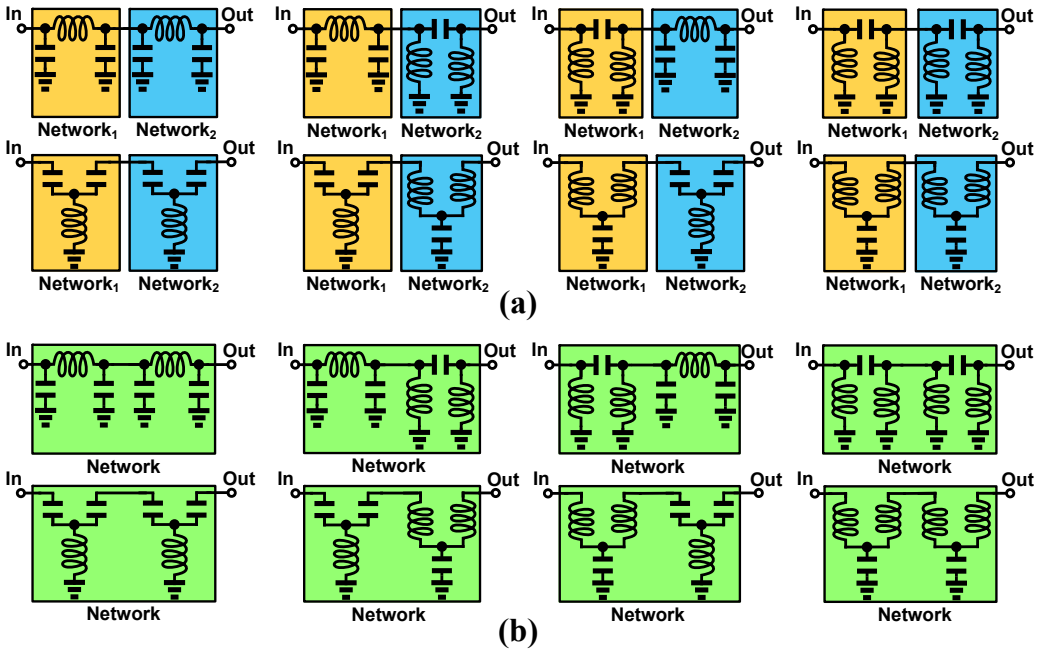


Figure 5.5: Different ways for partitioning 8 topologies. (a) First scenario: each is divided into two E-networks. (b) Second scenario: there is only one E-network.

Our approach can be summarized as follows. The E-networks are identified and we assign a sub-NN to each. Then, the design parameters of each E-network are fed as inputs to the corresponding sub-NN for determining their S-parameters. In parallel, general S-parameters of E-networks are fed as an input to the main NN to determine the PoI. By combining these models (Equations ((5.4a)) and (5.4b), *i.e.* $\hat{g}([\hat{h_1}(\mathbf{x_1}, t), \ldots, \hat{h_N}(\mathbf{x_N}, t), \mathbf{x_R}], t)$), the PoI is determined based on the circuit design parameters. It should be noted that $\hat{g}$ in Equation (5.4a) is a function of $s_1, \ldots, s_N$ in general, not specifically for the s_i in Equation (5.4b). In other words, Equation (5.4a) finds the relationship between the PoI and general S-parameters, and in parallel Equation (5.4b) maps the design parameters to S-parameters in the desired circuit.

Equation (5.4a) is independent of the design parameters of E-networks. Therefore, it does not change if E-networks change, proving the generality and re-usability of our model for different E-networks. Furthermore, since each of the sub-NNs (Equation (5.4b)) takes a subset of design parameters as inputs, they are simpler

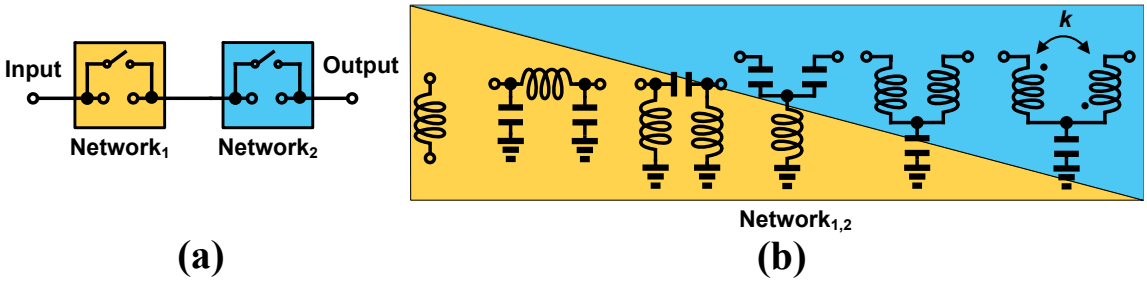


Figure 5.6: Phase shifters tested on FuNToM. (a) General structure. (b) Network$_1$ and network$_2$ topologies.

Specification	Min	Max	Average	SD
Input return loss [dB]	-117	0	-4.87	-10.12
Insertion loss [dB]	-133	0	-3.17	-10.26
Insertion phase [degree°]	-180	180	-23.02	84.5
Output return loss [dB]	-117	0	-4.87	-10.12

Table 5.2: Statistics of performance of interests for phase shifters.

than the conventional NN model (Fig. 5.3(e)). This significantly eases the collection of datasets for sub-NNs if E-networks change or during the post-layout modeling. Moreover, even in analyzing one specific circuit (without considering different E-networks), $\hat{g}$ may be simpler than $\hat{f}$ as S-parameters condense design parameters. This attribute is more pronounced in large RF circuits with many design parameters in each E-network. In addition, as mentioned in Section 5.2.2, $S_{12} = S_{21}$ for almost all passive E-networks and $S_{11} = S_{22}$ for the symmetric ones. These cause having even simpler $\hat{g}$. So, in such cases, the required training set is further reduced compared to the conventional method.

To summarize, in our proposed approach by concatenating the sub-NNs and the main NN, we determine the PoI by getting the circuit design parameters as inputs as shown in Fig. 5.4. As illustrated, sub-NNs determine the S-parameters of the associated E-network by getting the circuit design parameters within such E-network as inputs while the main NN determines the PoI by getting S-parameters of E-networks

as inputs. Two structures (CCI-NN in Fig. 5.3(b) and FC-NN in Fig. 5.3(c)) are analyzed for the main NN. As explained and our evaluations show, the CCI-NN structure gives the best results. The original CCI-NN structure proposed by [177] is shown in Fig. 5.3(e) where the circuit design parameters are given as inputs and the PoI is the output. The conventional FC-NN model in the analog and RF functional modeling is also illustrated in Fig. 5.3(f).

5.3.2 Circuit Partitioning into E-networks

There are multiple ways for partitioning a circuit into E-networks that each has some pros and cons. With decreasing the number of E-networks, the number of circuit elements at each increases as the total number of circuit elements is constant. So, the number of sub-NNs decreases while each gets more complicated. Usually, the required number of training data for a big complicated NN is more than the training data of multiple simple NNs. Moreover, with decreasing the number of E-networks, there would be fewer chunks in the CCI-NN structure of the main NN and since the input number of each chunk ($s_i = [S_{11,i}, \ldots, S_{22,i}]$) is almost fixed, the width of the main NN intuitively reduces. However, in order not to lose the accuracy, it may get deeper.

Similar to all machine learning techniques, to find the best results, different scenarios need to be tested. As an example to show different ways for partitioning a circuit, we consider two scenarios. To this end, we consider 8 different topologies. In the first scenario, each of these topologies is divided into two E-networks as shown in Fig. 5.5(a). In the second scenario, the whole circuit is considered as one E-network as illustrated in Fig. 5.5(b). Table 5.1 compares the required number of training data between these two scenarios for the main NN, sub-NNs, and total to have the same accuracy. As it is summarized, the first scenario (Fig. 5.5(a)) requires more data for the main NN while it needs less training data for sub-NNs and in total. Note that using such analysis and trying different scenarios, currently the circuit partitioning is

Model	Sub-NNs	main NN			
		Input return loss	Insertion loss	Insertion phase	Output return loss
R2-score	0.967	0.995	0.989	0.941	0.997

Table 5.3: Average R2-score of FuNToM models tested on different specifications of phase shifters shown in Fig. 5.6.

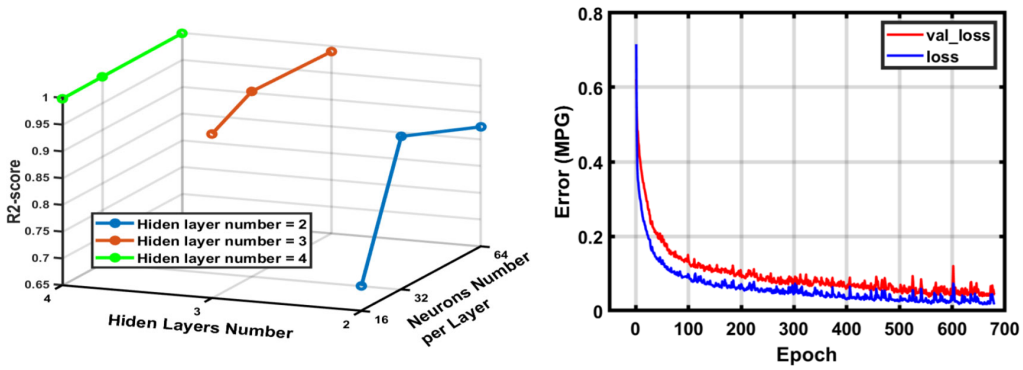


Figure 5.7: Main model results for phase shifters. (a) R2-score comparison with different hyperparameters used for training. Three hidden layers with 32 neurons at each layer is the simplest model that gives a high R2-score of 0.99. (b) Loss vs. epoch diagram while three hidden layers with 32 neurons at each layer model is used for training.

done manually, but it can be replaced with an automated approach going forward.

5.4 Evaluation

In this section, we model the functionality of two different RF circuit types using FuNToM and compare the results with the state-of-the-art modeling methods. We model multiple phase shifters at the schematic level, and multiple two-stage LNAs at both schematic and post-layout levels. The design parameters include all transistor widths as well as capacitor and inductor values. The inductors and capacitors are used from PDK models of 55nm BiCMOS library, *i.e.* quality factor and self-resonance of the inductors and capacitors are taken into account. All the SPICE simulations are

Work	[192]	[177]	FuNToM FC-NN (Fig. 5.3(c))	FuNToM CCI-NN (Fig. 5.3(b))
Training data number	57,600	19,500	17,000	7,000

Table 5.4: Required number of training data, per frequency, comparison between the state-of-the-art works to achieve an average R2-score of 0.95. All models are tested on 1600 different phase shifters with the general structure of Fig. 5.6 in 36 topologies.

run on a server with an NVIDIA TITAN V GPU and the frequency range for the simulations is 1 Hz-15 GHz. Also, all the NN models are built using the TensorFlow platform with the Adam optimizer and a learning rate of 0.001 and mean absolute error as the loss function. Moreover, all hidden layers have the RELU activation function. In order to avoid overfitting, the idea of early stopping [141] with the patience parameter of 125 is implemented. For this purpose, 10% of the data are used for validation during the training phase. In order to validate the results properly, a random separate test set with the size of 10% of the training set is used. As a default, all the results of FuNToM are based on the CCI-NN structure for the main NN unless that is mentioned.

5.4.1 Phase Shifter

Fig. 5.6(a) depicts the general structure of the phase shifters tested on FuNToM. All combinations of E-networks shown in Fig. 5.6(b) are used for both networks$_{1,2}$. Therefore, in total, FuNToM is trained and tested on 6×6=36 different phase shifter topologies. There are on average 8 design parameters at each topology. The tested specification statistics are listed in Table 5.2. In total, 6,600 and 400 simulations are performed for the training set of the main NN and sub-NNs, respectively, per frequency. Since most of the specifications change with frequency, different datasets are needed to properly model the specifications over the frequency.

The number of hidden layers and neurons per layer for the main model varies

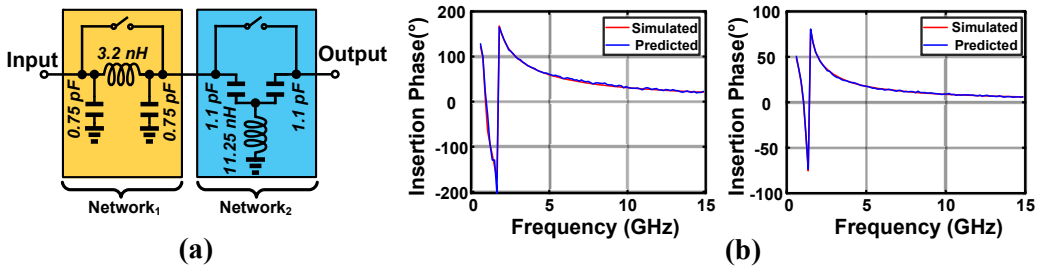


Figure 5.8: An example of a phase shifter tested on FuNToM. (a) Topology. (b) The simulated (ground truth) and the predicted insertion phase by FuNToM over frequency. Left: when both switches are open; right: when both switches are closed.

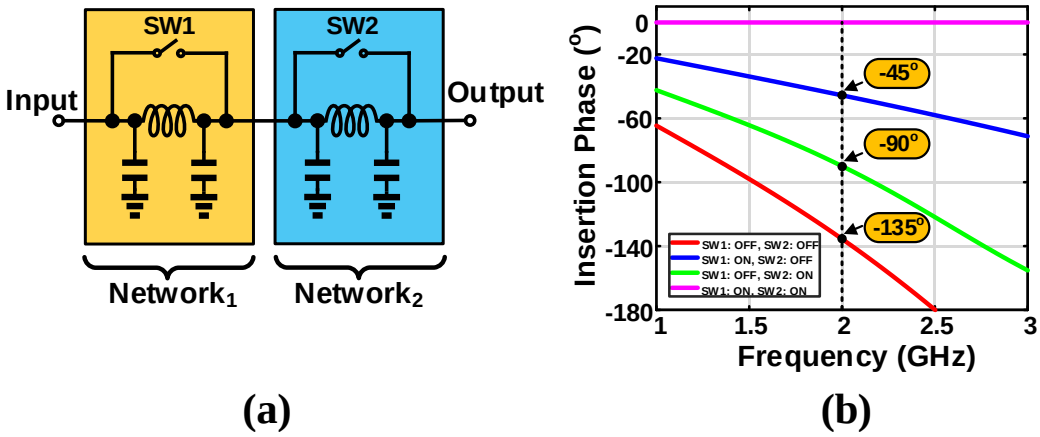


Figure 5.9: Using FuNToM as the simulator in circuit sizing by NSGA-II algorithm to meet the desired specification in Table 5.5. (a) The given phase shifter topology. (b) Insertion phase results over frequency for different switch conditions.

between 2-4 and 16-64, respectively. Fig. 5.7(a) demonstrates a comparison between the R2-score achieved by different hyperparameters used for training the main model. Fig. 5.7(b) also shows the loss vs. epoch of training and validation data of the main model. As it is shown, the training stops before overfitting happens. All sub-NNs have two hidden layers with 32 nodes at each layer.

Table 5.3 summarizes the average R2-score of sub-NNs and main NNs tested on different specifications of phase shifters. The average R2-score of all models is around 0.97. Moreover, Fig. 5.8(a) depicts an example of a phase shifter tested on FuNToM. Fig. 5.8(b) shows the predicted insertion phase as an example modeling specification

SW1	SW2	Specification	Simulator result	
			FuNToM	SPICE
ON	OFF	Insertion phases = -45°	-45.61°	-45°
OFF	ON	Insertion phases = -90°	-89.95°	-89.96°
ON	OFF	Input return loss < -35 dB	-43.5 dB	-70.6 dB
OFF	ON	Input return loss < -35 dB	-37.8 dB	-61.3 dB
Runtime			180s	3,600s

Table 5.5: An example of using FuNToM vs. SPICE as the circuit simulator for sizing Fig. 5.9(a) circuit. Desired vs. generated specifications at 2 GHz frequency for different switch conditions are compared.

by FuNToM over frequency compared to the simulated values.

The required number of training data per frequency is compared with the state-of-the-art works [177, 192] while all achieve the same average R2-score of 0.95. We also consider both CCI-NN (Fig. 5.3(b)) and FC-NN (Fig. 5.3(c)) structures for the main NN of FuNToM. 1600 different phase shifters with the general structure of Fig. 5.6 in 36 topologies are given to all approaches for testing. Moreover, a wide range of design parameters is given to cover Table 5.2. As summarized in Table 5.4, [192], and [177] need 8.2x and 2.8x more training data than FuNToM, respectively. Moreover, FuNToM with FC-NN structure requires 2.4x more training data than FuNToM with CCI-NN structure.

In order to further demonstrate the functionality of FuNToM, we apply it as the circuit simulator to a circuit sizing method. For this purpose, we employ the well-known multi-objective genetic algorithm NSGA-II [215] to optimize the phase-shifter shown in Fig. 5.9(a). An example of desired specifications at 2 GHz frequency is listed in Table 5.5. In the NSGA-II algorithm, the population size and number of generations are set to 30 and 30, respectively. The results when FuNToM and SPICE are used as the simulator in the sizing are summarized in Table 5.5. It should be

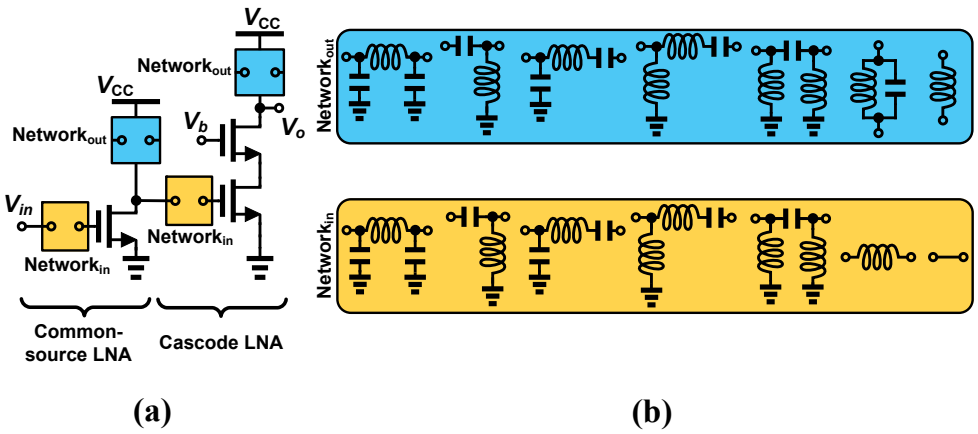


Figure 5.10: Two-stage LNAs tested on FuNToM. (a) General structure; 1^{st} stage is a common-source LNA while the 2^{nd} stage is a cascode LNA. (b) Input and output matching networks for each stage.

mentioned that the FuNToM results are obtained when the generated circuit using FuNToM is verified by SPICE. FuNToM is 20x faster than SPICE, while both meet the desired specifications. The insertion phase over frequency of the sized circuit using FuNToM is shown in Fig. 5.9(b) for different switch conditions. The $network_1$ and $network_1$ are designed to have an insertion phase of -45° and -90°, respectively, at 2 GHz based on Table 5.5. As expected, the phase shifter insertion phase is 0° when both switches are ON (meaning input and output are short-circuited). On the hand, when both switches are OFF, the insertion phase is -45°+(-90°) = -135°.

5.4.2 Two-stage LNA

Fig. 5.10(a) demonstrates the general structure of the two-stage LNAs tested on FuNToM. All combinations of E-networks shown in Fig. 5.10(b) are used for input and output matching. So, in total, FuNToM is trained and tested on 7×7=49 different LNA topologies at each stage which results in 49×49=2401 total topologies. There are, on average, 14 design parameters at each topology. The tested specification statistics for the two-stage LNAs are summarized in Table 5.6. In total, 14,000 and 600 simulations are performed for the training set of the main NN and sub-NNs,

Specification	Min	Max	Average	SD
Transducer gain [dB]	-183.7	10.73	-20.04	14.73
Power gain [dB]	-169.4	69.1	-9.1	14.6
Available gain [dB]	-169.4	14.58	-8.17	12.93
Noise Figure [dB]	0.06	3082	17.15	18.48
Input return loss [dB]	-60	0	-4.58	-10.75
Output return loss [dB]	-60	0	-4.58	-11.06
Rollett's stability factor	-2.4×10^{5}	10×10^{19}	4.2×10^{12}	4.5×10^{15}
Stability factor	2.1×10^{-4}	2.01	0.36	0.33
Maximum power gain frequency [GHz]	0.3	1.89	1.4	0.78
DC power [mW]	2.4	2.7	2.54	0.15

Table 5.6: Statistics of performance of interests for LNAs.

Model	Sub-NNs	Common-source LNA main NN	Cascode LNA main NN
R2-score	0.987	0.975	0.982

Table 5.7: Average R2-score of FuNToM models tested on different specifications of LNAs shown in Fig. 5.10.

respectively, per frequency.

The number of hidden layers and neurons per layer for the main model varies between 2-4 and 32-128, respectively. Our results show that three hidden layers with 32 neurons at each layer is the simplest model that gives a high R2-score of 0.97. Table 5.12 summarizes the average R2-score of sub-NNs and main NNs while tested on two-stage LNAs. Fig. 5.11(a) shows an example of a two-stage LNA tested on FuNToM. Fig. 5.11(b) illustrates the predicted transducer gain (G_T) as a sample modeling specification by FuNToM in the schematic modeling over frequency compared to the simulated values for each stage as well as the whole circuit.

Fig. 5.12 demonstrates the layout of Fig. 5.11(a) which is used for post-layout modeling. As mentioned earlier, for the post-layout modeling we only need to gather training sets for sub-NNs, while the training set of the main NN is the same as

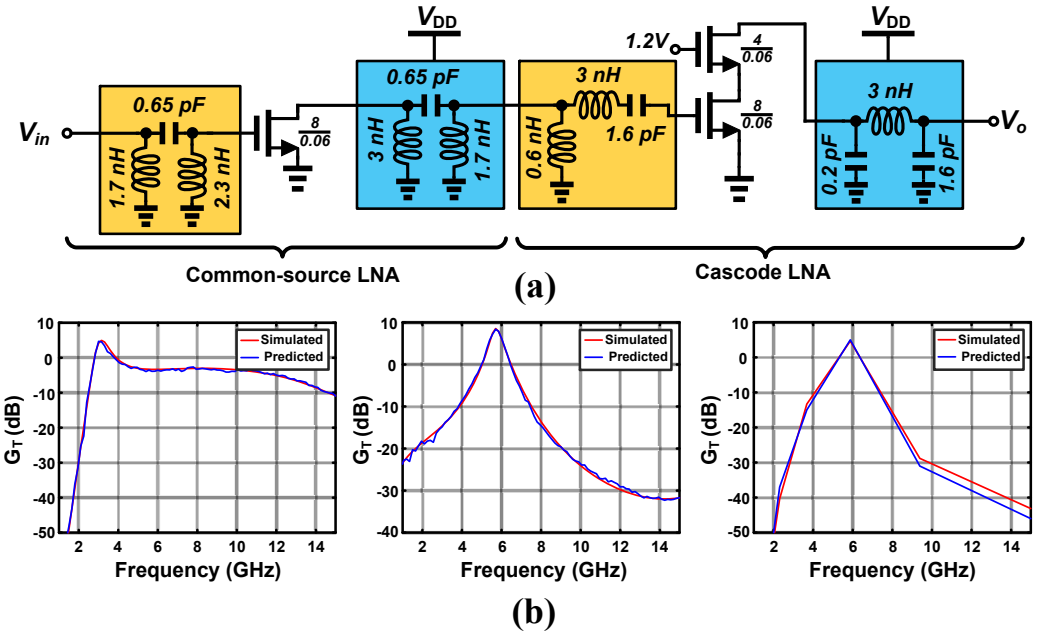


Figure 5.11: An example of a two-stage LNA tested on FuNToM. (a) Topology; size of transistors are shown as $\frac{W(\mu)}{L(\mu)}$. (b) Schematic modeling: The simulated (ground truth) and the predicted transducer gain (G_T) by FuNToM over frequency. Left: Common-source (1^{st}) stage; middle: Cascode (2^{nd}) stage; right: the whole LNA (two-stage).

what has been gathered for the schematic analysis. Moreover, the idea of transfer learning [193] can be used to further reduce the number of training samples when going from schematic to the post-layout modeling. Fig. 5.13 shows the predicted transducer gain (G_T) in the post-layout modeling by FuNToM over frequency compared to the simulated values for each stage as well as the whole circuit.

The required number of training data per frequency as well as the post-layout modeling training time of FuNToM are also compared with [177, 192] while all are achieving the same average R2-score of 0.95. 1600 different two-stage LNAs with the general structure of Fig. 5.10 that are selected from 2401 available topologies are given to all approaches for testing. Moreover, a wide range of design parameters is given to cover Table 5.6. As summarized in Table 5.8, [192], and [177] need 493.1x and 10.9x more training data than FuNToM, respectively. Moreover, FuNToM with FC-NN structure requires 3.3x more training data than FuNToM with CCI-NN structure.

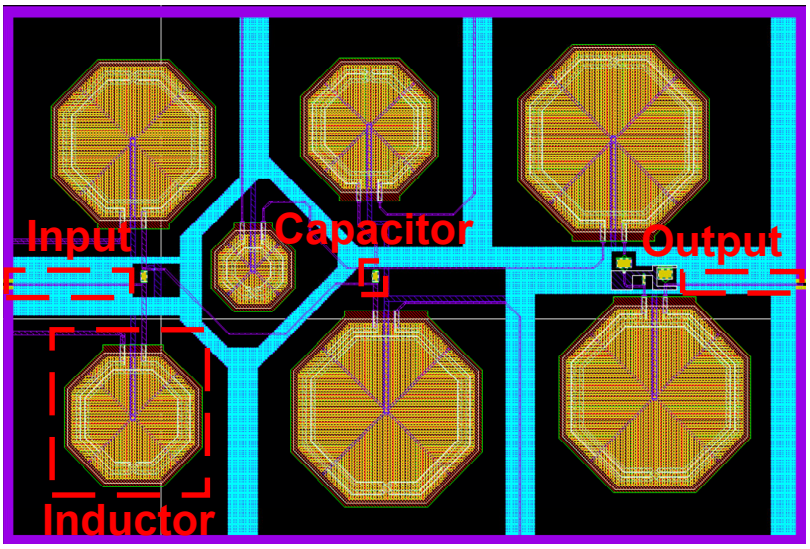


Figure 5.12: The layout of Fig. 5.11(a).

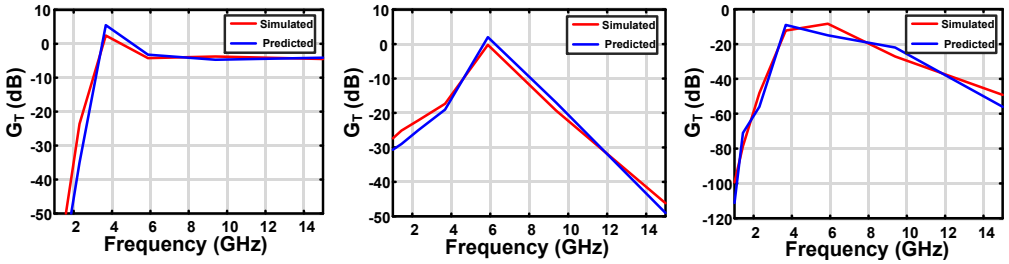


Figure 5.13: Post-layout modeling of Fig. 5.12: The simulated (ground truth) and the predicted transducer gain (G_T) by FuNToM over frequency. Left: Common-source (1^{st}) stage; middle: Cascode (2^{nd}) stage; right: the whole LNA (two-stage).

Furthermore, the average time for collecting the training set in the post-layout modeling of FuNToM is 188.6x and 176.8x less in comparison with [192] and [177], respectively.

To calculate this average time, we measure the average post-layout simulation time for generating each training sample needed in each approach. In [177, 192], the post-layout simulation is run on the whole and half of the circuit, respectively, for generating each sample which is very time-consuming considering 4-8 inductors are included. However, in FuNToM, 14,000 of the required data are for the main NN which their post-layout simulations are very fast as they include only transistors and modular S-parameters. Moreover, the 600 training data for sub-NNs are gathered

Work	[192]	[177]	FuNToM FC-NN (Fig. 5.3(c))	FuNToM CCI-NN (Fig. 5.3(b))
Training data number	7,200,000	160,000	48,100	14,600
Average time for collecting the training set in post-layout modeling per sample	5,790s	5,430s	30.7s	30.7s

Table 5.8: Required number of training data per frequency as well as the average time for collecting the training set in post-layout modeling per sample comparison between the state-of-the-art works to achieve an average R2-score of 0.95. All approaches are tested on 1600 different two-stage LNAs with the general structure of Fig. 5.10 that are selected from 2401 available topologies.

by simulating the E-networks that include only one or two inductors which are relatively fast. So, as the results show, the average time for running each simulation for FuNToM is much less than the other approaches. We save around 240,000 hours ($160{,}000 \times 5{,}430\text{s} - 14{,}600 \times 30.7\text{s} = 240{,}000\text{h}$) by using FuNToM instead of [192] in such a post-layout training.

5.5 Conclusion

This work presents a novel RF functional modeling method, FuNToM. FuNToM divides circuits into multiple E-networks and analyzes them modularly via NNs. Leveraging such a modular analysis using the concept of S-parameters has made the NN models used in FuNToM general enough that work for multiple circuit topologies. To validate our work, FuNToM is tested on more than 1,600 phase shifters and 1,600 two-stage LNAs. The results indicate that FuNToM reduces the size of the required training data by **2.8x - 10.9x** in comparison with the state-of-the-art works while maintaining the same accuracy. Moreover, FuNToM needs **176.8x - 188.6x** less time for collecting the training set in post-layout modeling.

CHAPTER VI

Conclusion & Future Work

Broadly speaking, this dissertation focuses on automating the design of analog and mixed-signal circuits as well as single-board computers, essential for diverse applications like wireless communication and automotive systems. In other words, when user gives specifications of the desired circuit, our tools design a circuit that meets those requirements. Our novel approach integrates circuit design intuition with machine learning, achieving significant reductions in data requirements and runtime.

This dissertation demonstrated three main projects: 1) FASCINET, a fully automated single-board computer generator; 2) AnGeL, a fully-automated analog circuit generator; and 3) FuNToM, a functional modeling tool for RF circuits. The main contribution of each work can be summarized as follows:

- FASCINET
  - Proposing datasheet scrubber, a novel CNN-based approach to categorize datasheets and extract information within their tables.
  - Implementing peripheral circuit generator, a novel HDNN-based model that intelligently designs customized peripheral circuits by giving the main components of an SBC.
  - Reducing the schematic design of an SBC two only **two minutes**.

- AnGeL
  - Creating a database using a co-learning-based NN model that significantly reduces the size of the required labeled training set compared to the state-of-the-art works.
  - Proposing an efficient and intelligent fully-automated analog circuit generator framework that uses the created database and designs circuits from deciding the topology to sizing.
  - Achieving a fast and accurate method using a hybrid of model- and simulation-based approaches.
  - Designing circuits with many design parameters (∼40) using a hierarchical approach.
  - Supporting many circuit topologies while few topologies are used in the training by leveraging the circuit division into sub-circuits.
- FuNToM
  - Proposing FuNToM, an efficient RF functional modeling method which achieves an average R2-score of **0.95** when it is tested on 3,200 samples with different topologies and specifications.
  - Reducing the number of required training set points compared to the state-of-the-art works by **2.8x - 10.9x** by using two-port analysis and dividing the circuit into multiple E-networks.
  - Decreasing the time for collecting the training set in the post-layout modeling by **176.8x - 188.6x** compared to state-of-the-art works.

In terms of the future work, as we mentioned earlier, one of the main challenges of AMS circuit automation is its dependency on circuit knowledge expertise. This

fact leads to shifting gears from viewing an AMS circuit design problem as a physics-based question to a math problem. Viewing an AMS circuit optimization task as a mathematical optimization problem, especially an AI problem, causes significant breakthroughs in the circuit design automation. However, a lot of circuit insight, that would help make a more efficient design, has been thrown away. Leveraging this circuit knowledge helps in both circuit optimization and performance modeling by either having a more optimized algorithm or by decreasing the amount of required training data.

While using a math model, instead of a highly complex physical model, brings many advantages, some trade-offs are made. The physical model still contains some information that the pure math model has trouble identifying. Therefore having a hybrid approach could potentially lead to better results. For instance, having a very simple circuit design insight which can be used as the design rule of thumbs can guide a better search to find the most optimum design. Moreover, it can lead to narrowing down the required training set and providing a ground truth to evaluate the AI model.

As an example, consider the operational amplifier shown in Fig. 6.1. Some of simple circuit design principles for this circuit can be written as follows.

- In order to eliminate the common-mode gain, which contradicts the purpose of having a differential circuit we should have $M_1 = M_2$, $M_3 = M_4$, $M_5 = M_6$, and $M_7 = M_8$.

- The gain is independent of M_7 and M_8 widths. Moreover, in a well-designed circuit, M_{10} and M_{11} have a minimal contribution to the amplifier gain as they work as a unity-gain buffer.

- As far as the input-referred noise is concerned, M_1, M_2, M_7, and M_8 are the main contributors. The remaining devices have minimal to no effect.

- Unity-gain bandwidth, which denotes the maximum frequency that the circuit

can amplify a voltage, to a first-order, equals $\frac{g_{m1}}{C}$. This means that it is independent from the other design parameters.

- Slew rate is a function of V_{b1}, M_9, and C. The rest of the circuit does not play a role.

These very simple principles can significantly reduce the design space in the circuit design automation. For example, if the tool knows the slew rate is just a function of three parameters, *i.e.* V_{b1}, M_9, and C, its design space would shrink considerably. Similarly, determining parameters that are independent of gain, input-referred noise, and unity-gain bandwidth avoids the necessity to optimize them, making the optimization model more efficient. Moreover, as it is described in Section 1.1 and 1.2, one technique that accelerates the searching procedure of finding the optimum sizing or performance model with less training set is the ability to guide the prediction procedure. For instance, Hakhamaneshi *et al.* [29] detects "good" and "bad" candidates to avoid processing all of them in order to guide the learning procedure. A similar guidance mechanism can be applied using these simple circuit design principles.

Training can be guided by simple circuit design principles such as methods to avoid common-mode gain or which attributes are proportional or inversely proportional to specifications. It is worth mentioning that applying these circuit design principles is not in contradiction with the goal of artificial intelligence and circuit design automation. These are just very simple rules for a better guidance which significantly increases the design procedure efficiency while the rest of the design would be still using pure artificial intelligence and circuit design automation techniques.

Moreover, even it is possible that the tool learns these circuit principles which opens a new window for the future works. As it is explained in Section 1.2, the idea of hierarchical learning, *e.g.* [15, 97] can be very useful as it breaks down a large design space into multiple smaller ones. In this case, at the first level of the hierarchy, the tool can learn the effects of each parameter or a group of parameters on every specification,

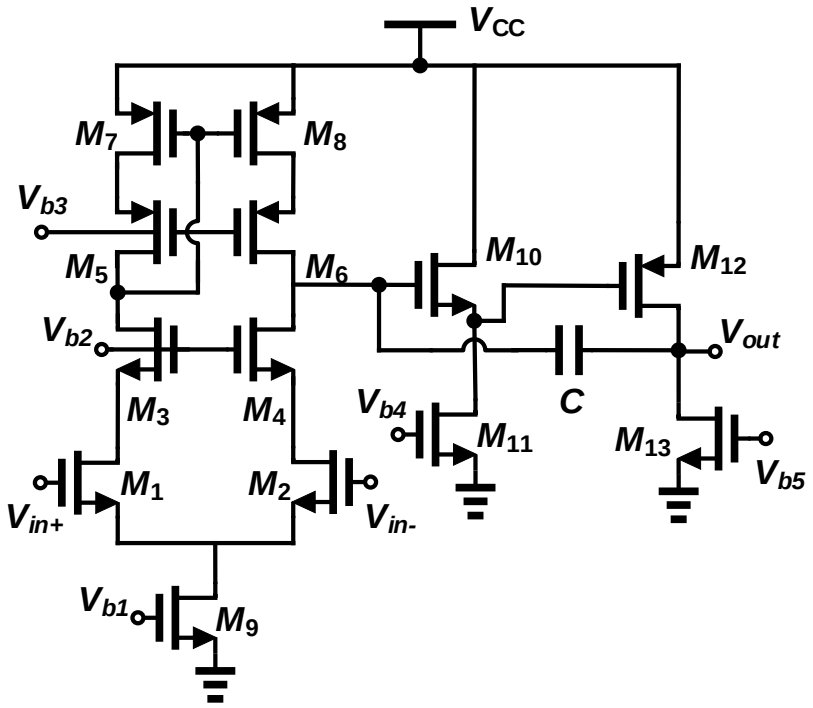


Figure 6.1: An example of an operational amplifier.

then in the second level of the hierarchy, the learned principles of the first level can be leveraged to design the circuit.

The other perspective for the future works would be using AI to improve the efficiency or relaxing the complexity of a specific AMS circuit design. Most studies so far have focused on either AMS circuit optimization or performance modeling in general. However, narrowing down the scope for a specific circuit type *e.g.* Phase Lock Loop (PLL), ADC, etc. and then analyzing them can be more efficient. Moreover, each of these circuits could benefit from AI in a unique way. For example, in [12], applying AI to an ADC leads to a more efficient and less complex design. Another application of AI for AMS circuit design would be using a simple AI model to detect and remove noise instead of a complex conventional circuit. In order to automate this procedure, one of the main challenges is first identifying and classifying different circuit types from their netlist. This task would be an extrapolation of what Liou *et al.* [216] propose for classifying sub-circuits into digital and analog. This can be considered

as one example of AMS circuits self-healing. Using AI for automating 1) diagnosis failure and degraded specifications, and 2) solving them can address many circuit design challenges [217].

BIBLIOGRAPHY

[1] B. Razavi and R. Behzad, *RF Microelectronics*, vol. 2, Prentice Hall New York, 2012.

[2] A. Turner, I. Karube, and G. S. Wilson, *Biosensors: Fundamentals and Applications*, Oxford university press, 1987.

[3] W. Lyu and *et al.*, "Batch Bayesian Optimization via Multi-objective Acquisition Ensemble for Automated Analog Circuit Design," in *International Conference on Machine Learning*, pp. 3306–3314, 2018.

[4] X. Li, J. Le, and L. T. Pileggi, *Statistical Performance Modeling and Optimization*, vol. 3, Now Publishers Inc, 2007.

[5] H. Zhang, T.-H. Chen, M.-Y. Ting, and X. Li, "Efficient Design-Specific Worst-Case Corner Extraction for Integrated Circuits," in *2009 46th ACMIEEE Design Automation Conference*, pp. 386–389, IEEE, 2009.

[6] S. Zhang and *et al.*, "Bayesian Optimization Approach for Analog Circuit Synthesis Using Neural Network," in *2019 DATE*, pp. 1463–1468, IEEE, 2019.

[7] A. Amarnath, J. Bagherzadeh, J. Tan, and R. G. Dreslinski, "3DTUBE: A Design Framework for High-Variation Carbon Nanotube-based Transistor Technology," in *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 1–6, IEEE, 2019.

[8] P. P. Prajapati and M. V. Shah, "Two Stage CMOS Operational Amplifier Design Using Particle Swarm Optimization Algorithm," in *2015 IEEE UPCON*, pp. 1–5, IEEE, 2015.

[9] A. R. Conn and *et al.*, "Gradient-Based Optimization of Custom Circuits Using a Static-Timing Formulation," in *Proceedings of ACM/IEEE DAC*, pp. 452–459, 1999.

[10] T. Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford university press, 1996.

[11] Y. Lin, M. B. Alawieh, W. Ye, and D. Z. Pan, "Machine Learning for Yield Learning and Optimization," in *2018 IEEE ITC*, pp. 1–10, IEEE, 2018.

[12] J. Zhang, L. Huang, Z. Wang, and N. Verma, "A Seizure-detection IC Employing Machine Learning to Overcome Data-conversion and Analog-processing Non-idealities," in *CICC*, pp. 1–4, IEEE, 2015.

[13] M. B. Alawieh, S. A. Williamson, and D. Z. Pan, "Rethinking Sparsity in Performance Modeling for Analog and Mixed Circuits using Spike and Slab Models," in *2019 ACMIEEE DAC*, pp. 1–6, IEEE, 2019.

[14] M. J. Barragan and *et al.*, "On the use of causal feature selection in the context of machine-learning indirect test," in *2019 DATE*, pp. 276–279, IEEE, 2019.

[15] M. B. Alawieh, F. Wang, and X. Li, "Efficient Hierarchical Performance Modeling for Analog and Mixed-Signal Circuits via Bayesian Co-Learning," *IEEE TCAD*, vol. 37, no. 12, pp. 2986–2998, 2018.

[16] H. Wang, J. Yang, H.-S. Lee, and S. Han, "Learning to Design Circuits," *arXiv preprint arXiv1812.02734*, 2018.

[17] R. Dreslinski, D. Wentzloff, M. Fayazi, K. Kwon, D. Blaauw, D. Sylvester, B. Calhoun, M. Coltella, and D. Urquhart, "Fully-Autonomous SoC Synthesis using Customizable Cell-Based Synthesizable Analog Circuits," tech. rep., University of Michigan Ann Arbor United States, 2019.

[18] N. Lourencco and *et al.*, "On the Exploration of Promising Analog IC Designs via Artificial Neural Networks," in *2018 15th International Conference on SMACD*, pp. 133–136, IEEE, 2018.

[19] T. Ajayi, Y. Cherivirala, K. Kwon, S. Kamineni, M. Saligane, M. Fayazi, S. Gupta, C.-H. Chen, D. Sylvester, D. Blaauw, *et al.*, "Fully Autonomous Mixed Signal SoC Design & Layout Generation Platform," 2020.

[20] T. Ajayi, S. Kamineni, Y. K. Cherivirala, M. Fayazi, K. Kwon, M. Saligane, S. Gupta, C.-H. Chen, D. Sylvester, D. Blaauw, *et al.*, "An Open-source Framework for Autonomous SoC Design with Analog Block Generation," in *2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC)*, pp. 141–146, IEEE, 2020.

[21] S. H. Naghavi, M. T. Taba, A. Tabatabavakili, A. Mostajeran, M. Aseeri, A. Cathelin, and E. Afshari, "24.4 Sub-THz Ruler: Spectral Bistability in a 235GHz Self-Injection-Locked Oscillator for Agile and Unambiguous Ranging," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67, pp. 418–420, IEEE, 2024.

[22] A. Tabatabavakili, J. Gruber, and E. Afshari, "A Review on Sub-THz FMCW Radars," in *2023 Asia-Pacific Microwave Conference (APMC)*, pp. 787–789, IEEE, 2023.

[23] Y. K. Cherivirala, M. Saligane, and D. D. Wentzloff, "An Open Source Compatible Framework to Fully Autonomous Digital LDO Generation," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 2023.

[24] M.-H. Lin, J.-F. Tsai, and C.-S. Yu, "A Review of Deterministic Optimization Methods in Engineering and Management," *Mathematical Problems in Engineering*, vol. 2012, 2012.

[25] P. K. Rout, D. P. Acharya, and G. Panda, "A Multiobjective Optimization Based Fast and Robust Design Methodology for Low Power and Low Phase Noise Current Starved VCO," *IEEE Transactions on Semiconductor Manufacturing*, vol. 27, no. 1, pp. 43–50, 2013.

[26] R. Phelps and *et al.*, "Anaconda: Simulation-Based Synthesis of Analog Circuits Via Stochastic Pattern Search," *IEEE TCAD*, vol. 19, no. 6, pp. 703–717, 2000.

[27] B. Liu, F. V. Fern'andez, G. Gielen, R. Castro-L'opez, and E. Roca, "A Memetic Approach to the Automatic Design of High-Performance Analog Integrated Circuits," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 14, no. 3, pp. 1–24, 2009.

[28] W. Lyu and *et al.*, "An Efficient Bayesian Optimization Approach for Automated Optimization of Analog Circuits," *IEEE TCAS-I Regular Papers*, vol. 65, no. 6, pp. 1954–1967, 2017.

[29] K. Hakhamaneshi, N. Werblun, P. Abbeel, and V. Stojanovi'c, "Late Breaking Results: Analog Circuit Generator based on Deep Neural Network enhanced Combinatorial Optimization," in *Proceedings of the DAC*, pp. 1–2, 2019.

[30] T. Sripramong and C. Toumazou, "The Invention of CMOS Amplifiers Using Genetic Programming and Current-Flow Analysis," *IEEE TCAD*, vol. 21, no. 11, pp. 1237–1252, 2002.

[31] O. Mitea, M. Meissner, and L. Hedrich, "Topology Synthesis of Analog Circuits with Yield Optimization and Evaluation using Pareto Fronts," in *2011 IEEEIFIP 19th on VLSI-SoC*, pp. 78–81, IEEE, 2011.

[32] F. Jiao and A. Doboli, "A Low-Voltage, Low-Power Amplifier Created by Reasoning-Based, Systematic Topology Synthesis," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2648–2651, IEEE, 2015.

[33] M. P.-H. Lin, Y.-W. Chang, and C.-M. Hung, "Recent Research Development and New Challenges in Analog Layout Synthesis," in *2016 ASP-DAC*, pp. 617–622, IEEE, 2016.

[34] S. P. Boyd and S. J. Kim, "Geometric Programming for Circuit Optimization," in *Proceedings of the 2005 international symposium on Physical design*, pp. 44–46, 2005.

[35] O. Okobiah, S. P. Mohanty, and E. Kougianos, "Exploring Kriging for Fast and Accurate Design Optimization of Nanoscale Analog Circuits," in *2014 IEEE Trans Very Large Scale Integr VLSI Syst*, pp. 244–247, IEEE, 2014.

[36] O. Okobiah, S. Mohanty, and E. Kougianos, "Fast Design Optimization Through Simple Kriging Metamodeling: A Sense Amplifier Case Study," *IEEE Trans Very Large Scale Integr VLSI Syst*, vol. 22, no. 4, pp. 932–937, 2013.

[37] T. McConaghy and G. Gielen, "Analysis of Simulation-Driven Numerical Performance Modeling Techniques for Application to Analog Circuit Optimization," in *2005 IEEE ISCAS*, pp. 1298–1301, IEEE, 2005.

[38] R. A. Rutenbar, G. G. Gielen, and J. Roychowdhury, "Hierarchical Modeling, Optimization, and Synthesis for System-Level Analog and RF Designs," *Proceedings of the IEEE*, vol. 95, no. 3, pp. 640–669, 2007.

[39] M. del Mar Hershenson, "Design of pipeline analog-to-digital converters via geometric programming," in *Proceedings of the 2002 IEEEACM DAC*, pp. 317–324, 2002.

[40] J. Yuan, N. Farhat, and J. Van der Spiegel, "GBOPCAD: ASynthesis Tool for High-Performance Gain-Boosted Opamp Design," *IEEE TCAS-I*, vol. 52, no. 8, pp. 1535–1544, 2005.

[41] S.-H. Lui, H.-K. Kwan, and N. Wong, "Analog circuit design by nonconvex polynomial optimization: Two design examples," *International Journal of Circuit Theory and Applications*, vol. 38, no. 1, pp. 25–43, 2010.

[42] A. K. Singh, K. Ragab, M. Lok, C. Caramanis, and M. Orshansky, "Predictable Equation-Based Analog Optimization Based on Explicit Capture of Modeling Error Statistics," *IEEE TCAD*, vol. 31, no. 10, pp. 1485–1498, 2012.

[43] T. McConaghy, P. Palmers, G. Gielen, and M. Steyaert, "Simultaneous Multi-Topology Multi-Objective Sizing Across Thousands of Analog Circuit Topologies," in *2007 44th ACMIEEE DAC*, pp. 944–947, IEEE, 2007.

[44] S. Hammouda and *et al.*, "Chameleon ART: A Non-Optimization Based Analog Design Migration Framework," in *Proceedings of DAC*, pp. 885–888, 2006.

[45] M. Dessouky, A. Kaiser, M.-M. Louerai, and A. Greiner, "Analog Design for Reuse - Case Study: Very Low-voltagespl $\Delta\Sigma$ Modulator," in *Proceedings DATE*, pp. 353–360, IEEE, 2001.

[46] R. Vural and T. Yildirim, "Analog circuit sizing via swarm intelligence," *AEU-International journal of electronics and communications*, vol. 66, no. 9, pp. 732–740, 2012.

[47] B. Liu and *et al.*, "Analog circuit optimization system based on hybrid evolutionary algorithms," *Integration*, vol. 42, no. 2, pp. 137–148, 2009.

[48] G. Huang, L. Qian, S. Saibua, D. Zhou, and X. Zeng, "An Efficient Optimization Based Method to Evaluate the DRV of SRAM Cells," *IEEE TCAS-I Regular Papers*, vol. 60, no. 6, pp. 1511–1520, 2013.

[49] A. Nieuwoudt and Y. Massoud, "Multi-level Approach for tntegrated Spiral Inductor Optimization," in *Proceedings of the 42nd annual Design Automation Conference*, pp. 648–651, 2005.

[50] B. Peng, F. Yang, C. Yan, X. Zeng, and D. Zhou, "Efficient Multiple Starting Point Optimization for Automated Analog Circuit Optimization via Recycling Simulation Data," in *2016 DATE*, pp. 1417–1422, IEEE, 2016.

[51] M. Li, D. Zhou, S.-G. Wang, and X. Zeng, "FMSSQP: An Efficient Global Optimization Tool for the Robust Design of Rail-to-Rail Op-Amp," in *IEEE 10th International Conference on ASIC*, pp. 1–4, IEEE, 2013.

[52] G. Huang, D. Zhou, X. Zeng, and S. Wang, "A Practical Method for Auto-Design and Optimization of DC-DC Buck Converter," in *2013 IEEE 10th International Conference on ASIC*, pp. 1–4, IEEE, 2013.

[53] L. Qian, Z. Bi, D. Zhou, and X. Zeng, "Automated Technology Migration Methodology for Mixed-Signal Circuit Based on Multistart Optimization Framework," *IEEE Trans Very Large Scale Integr VLSI Syst*, vol. 23, no. 11, pp. 2595–2605, 2014.

[54] M. Krasnicki, R. Phelps, R. A. Rutenbar, and L. R. Carley, "MAELSTROM: Efficient Simulation-Based Synthesis for Custom Analog Cells," in *Proceedings DAC*, pp. 945–950, IEEE, 1999.

[55] M. Naumowicz, M. Melosik, and P. Katarzy'nski, "Technology Migration of Analogue CMOS Circuits Using Hooke-Jeeves Algorithm and Genetic Algorithms in Multi-Core CPU Systems," in *Proceedings of the 20th International Conference Mixed Design of Integrated Circuits and Systems-MIXDES 2013*, pp. 267–272, IEEE, 2013.

[56] D. Liu, S. Sidiropoulos, and M. Horowitz, "A Framework for Designing Reusable Analog Circuits," in *ICCAD*, pp. 375–380, IEEE, 2003.

[57] G. İslamoğlu, T. O. Çakici, E. Afacan, and G. Dündar, "Artificial Neural Network Assisted Analog IC Sizing Tool," in *2019 16th SMACD*, pp. 9–12, IEEE, 2019.

[58] Y. Li, Y. Wang, Y. Li, R. Zhou, and Z. Lin, "An Artificial Neural Network Assisted Optimization System for Analog Design Space Exploration," *IEEE TCAD*, vol. 39, no. 10, pp. 2640–2653, 2019.

[59] B. Liu, D. Zhao, P. Reynaert, and G. G. Gielen, "Synthesis of Integrated Passive Components for High-Frequency RF ICs Based on Evolutionary Computation

and Machine Learning Techniques," *IEEE TCAD*, vol. 30, no. 10, pp. 1458–1468, 2011.

[60] B. Liu, Y. He, P. Reynaert, and G. Gielen, "Global Optimization of Integrated Transformers for High Frequency Microwave Circuits Using a Gaussian Process Based Surrogate Model," in *2011 DATE*, pp. 1–6, IEEE, 2011.

[61] B. Liu, N. Deferm, D. Zhao, P. Reynaert, and G. G. Gielen, "An Efficient High-Frequency Linear RF Amplifier Synthesis Method Based on Evolutionary Computation and Machine Learning Techniques," *IEEE TCAD*, vol. 31, no. 7, pp. 981–993, 2012.

[62] B. Liu, D. Zhao, P. Reynaert, and G. G. Gielen, "GASPAD: A General and Efficient mm-Wave Integrated Circuit Synthesis Method Based on Surrogate Model Assisted Evolutionary Algorithm," *IEEE TCAD*, vol. 33, no. 2, pp. 169–182, 2014.

[63] D. M. Colleran and *et al.*, "Optimization of Phase-Locked Loop Circuits via Geometric Programming," in *Proceedings of the IEEE 2003 CICC.*, pp. 377–380, IEEE, 2003.

[64] W. Daems, G. Gielen, and W. Sansen, "Simulation-Based Generation of Posynomial Performance Models for the Sizing of Analog Integrated Circuits," *IEEE TCAD*, vol. 22, no. 5, pp. 517–534, 2003.

[65] Y. Wang, M. Orshansky, and C. Caramanis, "Enabling Efficient Analog Synthesis by Coupling Sparse Regression and Polynomial Optimization," in *Proceedings of DAC*, pp. 1–6, 2014.

[66] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[67] B. Liu and A. Nikolaeva, "Efficient Global Optimization of MEMS Based on Surrogate Model Assisted Evolutionary Algorithm," in *2016 DATE*, pp. 555–558, IEEE, 2016.

[68] S. J. Park, B. Bae, J. Kim, and M. Swaminathan, "Application of Machine Learning for Optimization of 3-D Integrated Circuits and Systems," *IEEE Trans Very Large Scale Integr VLSI Syst*, vol. 25, no. 6, pp. 1856–1865, 2017.

[69] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[70] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*, vol. 2. MIT press Cambridge, MA, 2006.

[71] M. Schonlau, W. J. Welch, and D. R. Jones, "Global versus Local Search in Constrained Optimization of Computer Models," *Lecture Notes-Monograph Series*, pp. 11–25, 1998.

[72] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham, "Bayesian Optimization with Inequality Constraints," in *ICML*, vol. 2014, pp. 937–945, 2014.

[73] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[74] C. Chevalier and D. Ginsbourger, "Fast Computation of the Multi-Points Expected Improvement with Applications in Batch Selection," in *International Conference on Learning and Intelligent Optimization*, pp. 59–69, Springer, 2013.

[75] J. Wu and P. Frazier, "The Parallel Knowledge Gradient Method for Batch Bayesian Optimization," in *Advances in Neural Information Processing Systems*, pp. 3126–3134, 2016.

[76] A. Shah and Z. Ghahramani, "Parallel Predictive Entropy Search for Batch Global Optimization of Expensive Objective Functions," in *Advances in Neural Information Processing Systems*, pp. 3330–3338, 2015.

[77] B. Liu, F. V. Fernández, and G. G. Gielen, "Efficient and Accurate Statistical Analog Yield Optimization and Variation-Aware Circuit Sizing Based on Computational Intelligence Techniques," *IEEE TCAD*, vol. 30, no. 6, pp. 793–805, 2011.

[78] A. Singhee, S. Singhal, and R. A. Rutenbar, "Practical, Fast Monte Carlo Statistical Static Timing Analysis: Why and How," in *2008 IEEE/ACM DAC*, pp. 190–195, IEEE, 2008.

[79] M. Sengupta, S. Saxena, L. Daldoss, G. Kramer, S. Minehane, and J. Cheng, "Application Specific Worst Case Corners using Response Surfaces and Statistical Models," *IEEE TCAD*, vol. 24, no. 9, pp. 1372–1380, 2005.

[80] M. Wang and *et al.*, "Efficient Yield Optimization for Analog and SRAM Circuits via Gaussian Process Regression and Adaptive Yield Estimation," *IEEE TCAD*, vol. 37, no. 10, pp. 1929–1942, 2018.

[81] J. Nocedal and S. Wright, *Numerical Optimization*, Springer Science & Business Media, 2006.

[82] Y. Yang and *et al.*, "Smart-MSP: A Self-Adaptive Multiple Starting Point Optimization Approach for Analog Circuit Synthesis," *IEEE TCAD*, vol. 37, no. 3, pp. 531–544, 2017.

[83] F. Glover, "Tabu Search—part I," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[84] F. Glover, "Tabu Search—part II," *ORSA Journal on computing*, vol. 2, no. 1, pp. 4–32, 1990.

[85] G. Alpaydin, S. Balkir, and G. Dundar, "An Evolutionary Approach to Automatic Synthesis of High-Performance Analog Integrated Circuits," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 240–252, 2003.

[86] G. Wolfe and R. Vemuri, "Extraction and Use of Neural Network Models in Automated Synthesis of Operational Amplifiers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 2, pp. 198–212, 2003.

[87] H. Liu, A. Singhee, R. A. Rutenbar, and L. R. Carley, "Remembrance of Circuits Past: Macromodeling by Data Mining in Large Analog Design Spaces," in *Proceedings of DAC*, pp. 437–442, 2002.

[88] C. Cortes, M. Mohri, and A. Rostamizadeh, "L2 Regularization for Learning Kernels," *arXiv preprint arXiv:1205.2653*, 2012.

[89] P. Liashchynskyi and P. Liashchynskyi, "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS," *arXiv preprint arXiv:1912.06059*, 2019.

[90] P.-C. Pan, C.-C. Huang, and H.-M. Chen, "Late Breaking Results An Efficient Learning-based Approach for Performance Exploration on Analog and RF Circuit Synthesis," in *2019 ACMIEEE DAC*, pp. 1–2, IEEE, 2019.

[91] F. Wang and *et al.*, "Co-learning Bayesian Model Fusion Efficient Performance Modeling of Analog and Mixed-Signal Circuits Using Side Information," in *ICCAD*, pp. 575–582, IEEE, 2015.

[92] X. Li, W. Zhang, F. Wang, S. Sun, and C. Gu, "Efficient Parametric Yield Estimation of Analog/MixedSignal Circuits via Bayesian Model Fusion," in *ICCAD*, pp. 627–634, IEEE, 2012.

[93] C. Fang, F. Yang, X. Zeng, and X. Li, "BMF-BD: Bayesian Model Fusion on Bernoulli Distribution for Efficient Yield Estimation of Integrated Circuits," in *2014 51st ACMEDACIEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2014.

[94] F. Wang and *et al.*, "Bayesian Model Fusion: Large-Scale Performance Modeling of Analog and Mixed-Signal Circuits by Reusing Early-Stage Data," *IEEE TCAD*, vol. 35, no. 8, pp. 1255–1268, 2015.

[95] Q. Huang, C. Fang, F. Yang, X. Zeng, and X. Li, "Efficient Multivariate Moment Estimation via Bayesian Model Fusion for Analog and Mixed-Signal Circuits," in *Proceedings of DAC*, pp. 1–6, 2015.

[96] C. Fang and *et al.*, "Efficient Bit Error Rate Estimation for High-Speed Link by Bayesian Model Fusion," in *DATE*, pp. 1024–1029, IEEE, 2015.

[97] M. Alawieh, F. Wang, and X. Li, "Efficient Hierarchical Performance Modeling for Integrated Circuits via Bayesian Co-Learning," in *Proceedings of the DAC*, pp. 1–6, 2017.

[98] C. M. Bishop, *Pattern Recognition and Machine Learning*, springer, 2006.

[99] S. Yu and *et al.*, "Bayesian Co-Training," in *Advances in neural information processing systems*, pp. 1665–1672, 2008.

[100] M. B. Alawieh, X. Tang, and D. Z. Pan, "S^2-PM: Semi-Supervised Learning for Efficient Performance Modeling of Analog and Mixed Signal Circuits," in *Proceedings of the 24th ASP-DAC*, pp. 268–273, 2019.

[101] M. Fayazi, Z. Colter, Z. Benameur-El Youbi, J. Bagherzadeh, T. Ajayi, and R. Dreslinski, "FASCINET: A Fully Automated Single-Board Computer Generator Using Neural Networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 12, pp. 5435–5448, 2022.

[102] Z. Colter, M. Fayazi, Z. Benameur-El Youbi, S. Kamp, S. Yu, and R. Dreslinski, "Tablext: A combined neural network and heuristic based table extractor," *Array*, vol. 15, p. 100220, 2022.

[103] M. Fayazi, M. T. Taba, E. Afshari, and R. Dreslinski, "AnGeL: Fully-Automated Analog Circuit Generator Using a Neural Network Assisted Semi-Supervised Learning Approach," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2023.

[104] M. Fayazi, M. T. Taba, A. Tabatabavakili, E. Afshari, and R. Dreslinski, "FuNToM: Functional Modeling of RF Circuits Using a Neural Network Assisted Two-Port Analysis Method," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, IEEE, 2023.

[105] S. Kamp, M. Fayazi, Z. Benameur-El, S. Yu, and R. Dreslinski, "Open Information Extraction: A Review of Baseline Techniques, Approaches, and Applications," *arXiv preprint arXiv:2310.11644*, 2023.

[106] T. Ajayi, S. Kamineni, M. Fayazi, Y. K. Cherivirala, K. Kwon, S. Gupta, W. Duan, J. Lee, C.-H. Chen, M. Saligane, *et al.*, "Fully-Autonomous SoC Synthesis Using Customizable Cell-Based Analog and Mixed-Signal Circuits Generation," in *VLSI-SoC: Design Trends: 28th IFIP WG 10.5/IEEE International Conference on Very Large Scale Integration, VLSI-SoC 2020, Salt Lake City, UT, USA, October 6–9, 2020, Revised and Extended Selected Papers 28*, pp. 65–85, Springer, 2021.

[107] S. Kim, M. Fayazi, A. Daftardar, K.-Y. Chen, J. Tan, S. Pal, T. Ajayi, Y. Xiong, T. Mudge, C. Chakrabarti, *et al.*, "Versa: A 36-Core Systolic Multiprocessor With Dynamically Reconfigurable Interconnect and Memory," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 4, pp. 986–998, 2022.

[108] S. Kim, M. Fayazi, A. Daftardar, K.-Y. Chen, J. Tan, S. Pal, T. Ajayi, Y. Xiong, T. Mudge, C. Chakrabarti, *et al.*, "Versa: A Dataflow-Centric Multiprocessor with 36 Systolic ARM Cortex-M4F Cores and a Reconfigurable Crossbar-Memory Hierarchy in 28nm," in *2021 Symposium on VLSI Circuits*, pp. 1–2, IEEE, 2021.

[109] K.-Y. Chen, C.-S. Yang, Y.-H. Sun, C.-W. Tseng, M. Fayazi, X. He, S. Feng, Y. Yue, T. Mudge, R. Dreslinski, *et al.*, "A 507 GMACs/J 256-Core Domain Adaptive Systolic-Array-Processor for Wireless Communication and Linear-Algebra Kernels in 12nm FINFET," in *2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, pp. 202–203, IEEE, 2022.

[110] M. T. Taba, S. H. Naghavi, M. Fayazi, A. Sadeghi, A. Cathelin, and E. Afshari, "A Compact CMOS 363 GHz Autodyne FMCW Radar with 57 GHz Bandwidth for Dental Imaging," in *2023 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–2, IEEE, 2023.

[111] N. Alee, M. Rahman, and R. Ahmad, "Performance Comparison of Single Board Computer: A Case Study of Kernel on ARM Architecture," in *2011 6th International Conference on Computer Science & Education (ICCSE)*, pp. 521–524, IEEE, 2011.

[112] J. Doerr, "Low-Cost Microcomputing: The Personal Computer and Single-Board Computer Revolutions," *Proceedings of the IEEE*, vol. 66, no. 2, pp. 117–130, 1978.

[113] S. J. Kopec Jr, Y.-f. Chan, and R. F. Hartmann, "Programmable interface for computer system peripheral circuit card," May 5 1992, US Patent 5,111,423.

[114] N. Patin, *Power Electronics Applied to Industrial Systems and Transports, Volume 1: Synthetic Methodology to Converters and Components Technology*, Elsevier, 2015.

[115] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object Recognition with Gradient-Based Learning," in *Shape, contour and grouping in computer vision*, pp. 319–345, Springer, 1999.

[116] A. Ahmad, M. F. Roslan, and A. Amira, "Throughput, latency and cost comparisons of microcontroller-based implementations of wireless sensor network (WSN) in high jump sports," in *AIP Conference Proceedings*, vol. 1883, p. 020010, AIP Publishing LLC, 2017.

[117] Adafruit Industries, "Adafruit." `https://www.adafruit.com/`, Last accessed 2020-05-08.

[118] Sparkfun Electronics, "Sparkfun." `https://www.sparkfun.com/`, Last accessed 2020-05-08.

[119] W. P. Birmingham, A. Brennan, A. P. Gupta, and D. P. Siewiorek, "MICON: A Single-Board Computer Synthesis Tool," *IEEE Circuits and Devices Magazine*, vol. 4, no. 1, pp. 37–46, 1988.

[120] M. A. Curran, R. J. Diehl, G. A. Peck, and K. L. Rahaman, "Single board computer quotation and design system and method," May 24 2005, US Patent 6,898,580.

[121] W. P. Birmingham and D. P. Siewiorek, "MICON: A Knowledge Based Single Board Computer Designer," in *21st Design Automation Conference Proceedings*, pp. 565–571, IEEE, 1984.

[122] F. Anderson, T. Grossman, and G. Fitzmaurice, "Trigger-Action-Circuits: Leveraging Generative Design to Enable Novices to Design and Build Circuitry," in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pp. 331–342, 2017.

[123] J.-Y. Lo, D.-Y. Huang, T.-S. Kuo, C.-K. Sun, J. Gong, T. Seyed, X.-D. Yang, and B.-Y. Chen, "AutoFritz: Autocomplete for Prototyping Virtual Breadboard Circuits," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2019.

[124] Circuito, "Circuito." `https://www.circuito.io/`, Last accessed 2021-10-15.

[125] A. M. Mehta, J. DelPreto, B. Shaya, and D. Rus, "Cogeneration of Mechanical, Electrical, and Software Designs for Printable Robots from Structural Specifications," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2892–2897, IEEE, 2014.

[126] F. Slomka, M. Dorfel, R. Munzenberger, and R. Hofmann, "Hardware/Software Codesign and Rapid Prototyping of Embedded Systems," *IEEE Design & Test of Computers*, vol. 17, no. 2, pp. 28–38, 2000.

[127] J. Bachrach, D. Biancolin, A. Buchan, D. W. Haldane, and R. Lin, "Jitpcb," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2230–2236, IEEE, 2016.

[128] R. Lin, R. Ramesh, C. Chi, N. Jain, P. Dutta, and B. Hartmann, "Supporting Circuit Design with a Block-Based, Generator Language," in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–8, 2020.

[129] EDASolver, "EDASolver - Automatic component selection and pin matching." `http://edasolver.com/`, Last accessed 2021-10-15.

[130] M. Fayazi, Z. Colter, E. Afshari, and R. Dreslinski, "Applications of Artificial Intelligence on the Modeling and Optimization for Analog and Mixed-Signal Circuits: A Review," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2021.

[131] J. G. Shanahan and N. Roma, "Improving SVM Text Classification Performance through Threshold Adjustment," in *European Conference on Machine Learning*, pp. 361–372, Springer, 2003.

[132] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[133] S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[134] R. Smith *et al.*, "Tesseract OCR: Open source scientific tools for Python." `https://github.com/UB-Mannheim/tesseract`, Last accessed 2020-05-15.

[135] Digi-key Electronics, "Digi-key." `https://www.digikey.com`, Last accessed 2020-05-15.

[136] S. Chickerur, A. Goudar, and A. Kinnerkar, "Comparison of Relational Database with Document-Oriented Database (MongoDB) for Big Data Applications," in *2015 8th International Conference on Advanced Software Engineering & Its Applications (ASEA)*, pp. 41–47, IEEE, 2015.

[137] B. Dipina Damodaran, S. Salim, and S. M. Vargese, "Mongodb Vs Mysql: A Comparative Study of Performance in Super Market Management System," *International Journal of Computational Science and Information Technology (IJCSITY)*, vol. 4, no. 2, pp. 31–38, 2016.

[138] G. Alfian, M. Syafrudin, and J. Rhee, "Real-Time Monitoring System Using Smartphone-Based Sensors and NoSQL Database for Perishable Supply Chain," *Sustainability*, vol. 9, no. 11, p. 2073, 2017.

[139] M. B. Arethiya and M. Garg, "Comparative analysis of simple and aggregate query in MongoDB," *International Journal of Advance Research, Ideas And Innovations In Technology*, vol. 4, no. 3, 2018.

[140] Y. Lou, F. Ye, *et al.*, "Research on Data Query Optimization Based on SparkSQL and MongoDB," in *2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pp. 144–147, IEEE, 2018.

[141] A. Gulli and S. Pal, *Deep Learning with Keras*, Packt Publishing Ltd, 2017.

[142] Texas Instruments, "LM117." https://www.ti.com/lit/ds/symlink/lm117.pdf, Last accessed 2020-05-10.

[143] Diodes Incorporated, "AP2112." https://www.diodes.com/assets/Datasheets/AP2112.pdf, Last accessed 2020-05-10.

[144] R. J. Smith and R. C. Dorf, *Circuits, Devices and Systems: A First Course in Electrical Engineering*, John Wiley & Sons, 1991.

[145] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to Fine-Tune BERT for Text Classification?," in *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pp. 194–206, Springer, 2019.

[146] Cadence, "Cadence Design Systems." https://www.cadence.com/en_US/home.html, Last accessed 2024-06-12.

[147] Altium, "Altium." https://www.altium.com/, Last accessed 2024-06-12.

[148] Eagle, "Eagle." https://www.autodesk.com/products/eagle/overview, Last accessed 2024-06-12.

[149] A. Tengli, Y. Yang, and N. L. Ma, "Learning Table Extraction from Examples," in *Proceedings of the 20th international conference on Computational Linguistics*, p. 987, Association for Computational Linguistics, 2004.

[150] E. Oro and M. Ruffolo, "PDF-TREX: An Approach for Recognizing and Extracting Tables from PDF Documents," in *2009 10th International Conference on Document Analysis and Recognition*, pp. 906–910, IEEE, 2009.

[151] D. Pinto, A. McCallum, X. Wei, and W. B. Croft, "Table Extraction Using Conditional Random Fields," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 235–242, 2003.

[152] E. Belval, "pdf2image: Open source scientific tools for Python." https://github.com/Belval/pdf2image. last accessed 2024-06-12.

[153] "Grabzit: Open source scientific tools for Python." https://github.com/GrabzIt/grabzit. last accessed 2024-06-12.

[154] J. Redmon and A. Farhadi, "Yolov3: An Incremental Improvement," *arXiv*, 2018.

[155] F. Cesarini, S. Marinai, L. Sarti, and G. Soda, "Trainable table location in document images," in *Object recognition supported by user interaction for service robots*, vol. 3, pp. 236–240, IEEE, 2002.

[156] M. Fan and D. S. Kim, "Table Region Detection on Large-scale PDF Files without Labeled Data," *CoRR, abs/1506.08891*, 2015.

[157] B. Gatos, D. Danatsas, I. Pratikakis, and S. J. Perantonis, "Automatic Table Detection in Document Images," in *International Conference on Pattern Recognition and Image Analysis*, pp. 609–618, Springer, 2005.

[158] A. Gilani, S. R. Qasim, I. Malik, and F. Shafait, "Table Detection Using Deep Learning," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1, pp. 771–776, IEEE, 2017.

[159] L. Hao, L. Gao, X. Yi, and Z. Tang, "A Table Detection Method for PDF Documents Based on Convolutional Neural Networks," in *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pp. 287–292, IEEE, 2016.

[160] T. Kasar, P. Barlas, S. Adam, C. Chatelain, and T. Paquet, "Learning to Detect Tables in Scanned Document Images Using Line Information," in *2013 12th International Conference on Document Analysis and Recognition*, pp. 1185–1189, IEEE, 2013.

[161] F. Shafait and R. Smith, "Table detection in heterogeneous documents," in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pp. 65–72, 2010.

[162] M. O. Perez-Arriaga, T. Estrada, and S. Abad-Mota, "TAO: System for Table Detection and Extraction from PDF Documents," in *The Twenty-Ninth International Flairs Conference*, 2016.

[163] T. Hassan and R. Baumgartner, "Table Recognition and Understanding from PDF Files," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2, pp. 1143–1147, IEEE, 2007.

[164] B. Yildiz, K. Kaiser, and S. Miksch, "pdf2table: A Method to Extract Table Information from PDF Files," in *IICAI*, pp. 1773–1785, 2005.

[165] Y. Liu, K. Bai, P. Mitra, and C. Giles, "Searching for Tables in Digital Documents," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2, pp. 934–938, IEEE, 2007.

[166] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering: A Review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[167] K. Nishida, K. Sadamitsu, R. Higashinaka, and Y. Matsuo, "Understanding the Semantic Structures of Tables with a Hybrid Deep Neural Network Architecture," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[168] E. Koci, M. Thiele, W. Lehner, and O. Romero, "Table Recognition in Spreadsheets via a Graph Representation," in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pp. 139–144, IEEE, 2018.

[169] A. Puha, O. Rinciog, and V. Posea, "Enhancing Open Data Knowledge by Extracting Tabular Data from Text Images," in *DATA*, IEEE, 2018.

[170] S. R. Qasim, H. Mahmood, and F. Shafait, "Rethinking Table Recognition using Graph Neural Networks," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 142–147, IEEE, 2019.

[171] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed, "DeepDeSRT: Deep Learning for Detection and Structure Recognition of Tables in Document Images," in *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, vol. 1, pp. 1162–1167, IEEE, 2017.

[172] S. S. Paliwal, D. Vishwanath, R. Rahul, M. Sharma, and L. Vig, "TableNet: Deep Learning model for end-to-end Table detection and Tabular data extraction from Scanned Document Images," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 128–133, IEEE, 2019.

[173] M. Göbel, T. Hassan, E. Oro, and G. Orsi, "ICDAR 2013 Table Competition," in *2013 12th International Conference on Document Analysis and Recognition*, pp. 1449–1453, IEEE, 2013.

[174] M. Aristarán *et al.*, "Tabula: Open source scientific tools for Python." `https://github.com/tabulapdf/tabula`, Last accessed 2020-05-15.

[175] R. Mina, C. Jabbour, and G. E. Sakr, "A Review of Machine Learning Techniques in Analog Integrated Circuit Design Automation," *Electronics*, vol. 11, no. 3, p. 435, 2022.

[176] M. Fayazi, A. Fotowat, and Z. Kavehvash, "A simplified approach to two-port analysis in feedback," *arXiv preprint arXiv:1908.10274*, 2019.

[177] M. Hassanpourghadi *et al.*, "Circuit Connectivity Inspired Neural Network for Analog Mixed-Signal Functional Modeling," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 505–510, IEEE, 2021.

[178] E. S. Ochotta, R. A. Rutenbar, and L. R. Carley, "Synthesis of High-Performance Analog Circuits in ASTRX/OBLX," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 3, pp. 273–294, 1996.

[179] Z. Wang, X. Luo, and Z. Gong, "Application of Deep Learning in Analog Circuit Sizing," in *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence*, pp. 571–575, 2018.

[180] A. F. Budak, K. Zhu, H. Chen, S. Poddar, L. Zhao, Y. Jia, and D. Z. Pan, "Joint Optimization of Sizing and Layout for AMS Designs: Challenges and Opportunities," in *Proceedings of the 2023 International Symposium on Physical Design*, pp. 84–92, 2023.

[181] K. Settaluri *et al.*, "AutoCkt: Deep Reinforcement Learning of Analog Circuit Designs," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 490–495, IEEE, 2020.

[182] A. F. Budak, D. Z. Pan, H. Chen, K. Zhu, M. Liu, M. B. Alawieh, S. Zhang, W. Shi, and X. Tang, "CAD for Analog/Mixed-Signal Integrated Circuits," *Advances in Semiconductor Technologies: Selected Topics Beyond Conventional CMOS*, pp. 43–60, 2022.

[183] H. Wang *et al.*, "GCN-RL Circuit Designer: Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2020.

[184] J. R. Koza, F. H. Bennett, D. Andre, M. A. Keane, and F. Dunlap, "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming," *IEEE Transactions on evolutionary computation*, vol. 1, no. 2, pp. 109–128, 1997.

[185] E. Kaya, E. Afacan, and G. Dundar, "An Analog/RF Circuit Synthesis and Design Assistant Tool for Analog IP: DATA-IP," in *2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pp. 1–9, IEEE, 2018.

[186] R. S. Zebulum, M. A. Pacheco, and M. M. B. Vellasco, *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms*, CRC press, 2018.

[187] M. Meissner and L. Hedrich, "FEATS: Framework for Explorative Analog Topology Synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 213–226, 2014.

[188] I. Abel and H. Graeb, "FUBOCO: Structure Synthesis of Basic Op-Amps by FUnctional BlOck COmposition," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 27, no. 6, pp. 1–27, 2022.

[189] J. Lu, L. Lei, F. Yang, L. Shang, and X. Zeng, "Topology Optimization of Operational Amplifier in Continuous Space via Graph Embedding," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 142–147, IEEE, 2022.

[190] N. Kahraman and T. Yildirim, "Technology Independent Circuit Sizing for Fundamental Analog Circuits Using Artificial Neural Networks," in *2008 Ph. D. Research in Microelectronics and Electronics*, pp. 1–4, IEEE, 2008.

[191] R. M. Hasani *et al.*, "Compositional Neural-Network Modeling of Complex Analog Circuits," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2235–2242, IEEE, 2017.

[192] M. Fukuda, T. Ishii, and N. Takai, "OP-AMP sizing by inference of element values using machine learning," in *2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pp. 622–627, IEEE, 2017.

[193] J. Liu *et al.*, "From Specification to Silicon: Towards Analog/Mixed-Signal Design Automation using Surrogate NN Models with Transfer Learning," in *ICCAD*, pp. 1–9, IEEE, 2021.

[194] Z. Dong, W. Cao, M. Zhang, D. Tao, Y. Chen, and X. Zhang, "CktGNN: Circuit Graph Neural Network for Electronic Design Automation," *arXiv preprint arXiv:2308.16406*, 2023.

[195] I. Abel, M. Neuner, and H. Graeb, "Constraint-Programmed Initial Sizing of Analog Operational Amplifiers," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*, pp. 413–421, IEEE, 2019.

[196] E. Afacan, N. Lourenço, R. Martins, and G. Dündar, "Machine learning techniques in analog/RF integrated circuit design, synthesis, layout, and test," *Integration*, vol. 77, pp. 113–130, 2021.

[197] J. Huang, C. Wang, Y. Yan, C. Tao, F. Yang, C. Yan, W. Hu, D. Zhou, and X. Zeng, "An Analog Circuit Building Block Generator via Nested Multi-Fidelity Modeling," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2023.

[198] R. A. Thakker, M. S. Baghini, and M. B. Patil, "Low-Power Low-Voltage Analog Circuit Design Using Hierarchical Particle Swarm Optimization," in *2009 22nd international conference on VLSI design*, pp. 427–432, IEEE, 2009.

[199] D. Joshi, S. Dash, U. Agarwal, R. Bhattacharjee, and G. Trivedi, "Analog Circuit Optimization Based on Hybrid Particle Swarm Optimization," in *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 164–169, IEEE, 2015.

[200] D. Joshi, S. Dash, S. Reddy, R. Manigilla, and G. Trivedi, "Multi-objective Hybrid Particle Swarm Optimization and its Application to Analog and RF Circuit Optimization," *Circuits, Systems, and Signal Processing*, vol. 42, no. 8, pp. 4443–4469, 2023.

[201] Z. Wu and I. Savidis, "Transfer Learning for Reuse of Analog Circuit Sizing Models Across Technology Nodes," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1033–1037, IEEE, 2022.

[202] G. Biau and E. Scornet, "A Random Forest Guided Tour," *Test*, vol. 25, no. 2, pp. 197–227, 2016.

[203] M. Somvanshi, P. Chavan, S. Tambade, and S. Shinde, "A Review of Machine Learning Techniques using Decision Tree and Support Vector Machine," in *2016 international conference on computing communication control and automation (ICCUBEA)*, pp. 1–7, IEEE, 2016.

[204] F. Provost, "Machine Learning from Imbalanced Data Sets 101," in *Proceedings of the AAAI'2000 workshop on imbalanced data sets*, vol. 68, pp. 1–3, AAAI Press, 2000.

[205] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[206] L. J. V. Miranda, "PySwarms, a research-toolkit for Particle Swarm Optimization in Python," *Journal of Open Source Software*, vol. 3, 2018.

[207] J. Liu, , *et al.*, "Transfer Learning with Bayesian Optimization-Aided Sampling for Efficient AMS Circuit Modeling," in *ICCAD*, pp. 1–9, 2020.

[208] F. Wang *et al.*, "Bayesian Model Fusion: Large-Scale Performance Modeling of Analog and Mixed-Signal Circuits by Reusing Early-Stage Data," in *DAC*, pp. 1–6, 2013.

[209] A. Gusmão *et al.*, "Semi-supervised artificial neural networks towards analog IC placement recommender," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 2020.

[210] T. McConaghy, T. Eeckelaert, and G. Gielen, "CAFFEINE: Template-Free Symbolic Model Generation of Analog Circuits via Canonical Form Functions and Genetic Programming," in *DATE*, IEEE, 2005.

[211] X. Li, W. Zhang, and F. Wang, "Large-Scale Statistical Performance Modeling of Analog and Mixed-Signal Circuits," in *Proceedings of the IEEE 2012 CICC*, pp. 1–8, IEEE, 2012.

[212] B. D. Anderson and S. Vongpanitlerd, *Network Analysis and Synthesis: A Modern Systems Theory Approach*, Courier Corporation, 2013.

[213] D. M. Pozar, *Microwave Engineering*, John wiley & sons, 2011.

[214] Z. Zhao and L. Zhang, "Efficient Performance Modeling for Automated CMOS Analog Circuit Synthesis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 11, pp. 1824–1837, 2021.

[215] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[216] G.-H. Liou, S.-H. Wang, Y.-Y. Su, and M. P.-H. Lin, "Classifying Analog and Digital Circuits with Machine Learning Techniques Toward Mixed-Signal Design Automation," in *2018 15th International Conference on SMACD*, pp. 173–176, IEEE, 2018.

[217] S. M. Bowers, K. Sengupta, K. Dasgupta, and A. Hajimiri, "A Fully-Integrated Self-Healing Power Amplifier," in *RFIC*, pp. 221–224, IEEE, 2012.