IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 20 July 2024, accepted 5 August 2024, date of publication 15 August 2024, date of current version 28 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3444702

RESEARCH ARTICLE

Deep Learning-Based Framework for Power Converter Circuit Identification and Analysis

BHARAT BOHARA, (Member, IEEE), AND HARISH SARMA KRISHNAMOORTHY, (Senior Member, IEEE)
Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004, USA
Corresponding authors: Bharat Bohara (bbohara@uh.edu) and Harish Sarma Krishnamoorthy (hskrishn@uh.edu)

This work was supported in part by the National Science Foundation (NSF) under Grant 2239966.

ABSTRACT This paper introduces a deep learning-based framework for identifying hand-drawn schematics of power converter circuits and performing automated simulations. The framework employs cutting-edge computer vision-based object detection models, such as YOLOv8, to achieve a high mean average precision (mAP) of 96.7% to accurately identify components. Wire tracing and connectivity are achieved through a combined architecture built upon classical image processing techniques and deep learning approaches. Detailed information extracted from a hand-drawn circuit schematic is used to automatically create its netlist for automated simulation through the spice engine. The proposed framework is successfully tested on various nonisolated (buck, boost) and isolated (flyback, full-bridge) converters under both continuous conduction mode (CCM) and discontinuous conduction mode (DCM) operations. In the comprehensive assessment of the entire framework, its efficacy is tested on 140 newly drawn circuit diagrams. The overall accuracy in the generation of netlists reaches a high value of 95.71%, utilizing the robust component detection capabilities of YOLOv8. Moreover, the framework enables the generation of both graphical representations and adjacency matrices for circuit diagrams. This output serves as a valuable dataset generator, contributing to the rapidly advancing domains of machine learning, including graph neural networks and geometric learning, particularly in the application space of power and energy systems. This framework can be further employed as an educational tool, and the ideas introduced can be developed to generate fully automated and efficient power converter designs for real-world applications.

INDEX TERMS Automated circuit simulation, computer vision, deep learning, hand-drawn circuit diagram, NetList, spice, power converter, automated graph generation.

I. INTRODUCTION

Hand-drawing sketches and informal handwritten texts have been fundamental methods of expressing ideas and facilitating human-computer interactions throughout history. From ancient rock carvings to modern blueprints, hand drawing has evolved as an integral part of human intelligence [1]. Before the widespread use of computers and digital platforms, hand drawing and handwriting were the primary means of conveying ideas, from simple concepts to complex thoughts. Randall Davis [2] emphasizes that sketching provides individuals with the freedom to explore their thoughts, and cognitive sciences support the notion that designers are more adept at generating diverse design alternatives and ideas through sketches compared to computer-aided design tools.

Handwritten notes, mathematical equations, numerical computations, and schematic diagrams represent the innate and initial manifestation of ideas and brainstorming, primarily within the STEM domain [3]. Researchers and engineers commonly use hand-drawn schematics to brainstorm engineering diagrams [4], [5]. However, transferring these hand-drawn sketches to digital form is time consuming and prone to human errors. Automated segmentation and recognition of handwritten schematics for computer-aided design and manufacturing systems are gaining broad interest in the research community [6]. This automated mechanism

The associate editor coordinating the review of this manuscript and approving it for publication was Fengjiang Wu.

has the potential to decrease errors and significantly streamline the circuit design and simulation process.

This paper presents a comprehensive computer vision-based framework designed to tackle the difficulties associated with digitizing hand-drawn circuit diagrams. The framework automatically identifies components, traces circuit connections, and conducts simulations based on the recognized topology. Using the power of the SPICE simulation engine and the freedom to sketch on paper, this technique streamlines the design process from hand sketching to computer simulation. The proposed digitization pipeline uses state-of-the-art computer vision models, image processing techniques, and optical character recognition methods to map the interconnections of electronic components, nodes, and sources. Implementing this tool in industries can help automate the conversion of hand sketches into ready-to-simulate designs, saving significant time. It can also accelerate the development and drafting processes. Moreover, it can be scaled to a computer software or mobile application that digitizes hand-drawn schematics, expediting the circuit design and analysis processes and enhancing the brainstorming phase. Furthermore, this framework is beneficial as a smart teaching and learning tool for novice engineers in power electronics to accelerate their learning process.

Electronic Design Automation (EDA) software provides robustness and user-friendliness for electronics design as well as automation. Such an automation has not been fully realizable yet in power converter development; but, further efforts are being made to advance such technologies. There is also merit in seamlessly converting hand-drawn schematics into digital formats and directly performing simulations from a PC or mobile device, especially for educational purposes. The tool proposed in this paper takes a step toward addressing this gap by leveraging advances in machine learning and computer vision, which make it feasible to accurately recognize and interpret hand-drawn schematics. The initial converter development phase of system design and brainstorming can then be made more flexible and intuitive. The potential impacts of this approach on the EDA tools include emphasizing the users to focus on fundamental understanding of the circuit diagrams and fostering their creativity, rather than bounding them to use any specific software interfaces, especially in the early stages of design phase. Instead of competing with existing EDA software, this computer vision-driven tool can serve as a complementary add-on feature to provide an alternative method to assist users to input their design input, thus bridging the gap between traditional circuit design methods and modern fully digitized workflows. Additionally, this tool can serve as a potential educational tool that offers immediate feedback to users' designs through quick simulations.

The generated netlist and auto-simulation feature of the proposed method can be integrated with the existing EDA and simulation software. Additionally, engineers and designers can quickly sketch their ideas on a piece of paper and this tool can automate the entire process of converting those raw information to digital formats for further refinement and development process, thus accelerating the initial prototyping phase. In addition to rapid prototyping, it can help students and educators by easily digitizing hand-sketched diagrams from textbooks or classroom notebooks for quick simulation and understanding the working principles of the raw circuit diagrams. This approach will not only reduce the product development time but also reduce the learning curve for novice learners, thus facilitating them to prioritize more on understanding fundamental principles. Moreover, it makes power electronics circuit design accessible to hobbyists and those in regions with limited or no access to such expensive software and tools.

The remaining sections of this article are structured as follows: Section II provides a review of related research on component recognition, auto-netlist generation, and simulation of hand-drawn electrical schematics. Section III outlines the methodology of the proposed framework. Section IV discusses the results obtained from the research. Finally, Section V summarizes the overall results of the work and highlights potential avenues for future research.

II. LITERATURE REVIEW AND RELATED WORK

Even before the advent of machine learning, recognizing hand-drawn circuits was a captivating research area. The techniques used for recognition included the analysis of ink density, pen strokes [7], pixel gradients, basic geometric information such as lines and arcs, as well as template matching and probabilistic graph matching [8] to identify symbols. Hand sketching has always been a natural and effective means of communication and problem solving for designers and engineers [7], [9]. The rise in popularity of computing devices such as smartphones, tablets, and touchscreen computers has made handwriting the most accessible and commonly used interactive tool in academia and industries. Although there is a substantial body of literature on component and text recognition, more research is still needed to address the challenge of identifying connections between components and simulating circuits based on observed relationships between components, nodes, and textual information. This section presents relevant prior art along with their strengths and weaknesses.

There is limited literature on a comprehensive framework that combines hand-drawn circuit topology recognition and automated simulation for power electronic converters. Existing research primarily focuses on recognizing electronic components in hand-drawn circuit diagrams. One study [10] used a composite design integrating local binary pattern (LBP) and statistical pixel density distribution to extract features from components, which were then classified using SVM. Another article [11] proposed a sparse auto-encoder method based on convolutional neural networks (CNN) for feature extraction and softmax for component classification, achieving 95% precision. A hybrid approach was presented in [12], utilizing a texture feature descriptor based on

histogram of oriented gradients (HOG) and shape-based features to recognize analog and digital components, achieving 93.83% accuracy using sequential minimal optimization (SMO). In [13], a two-stage system based on CNNs achieved 97.33% accuracy to recognize analog and digital components. Other methods, such as YOLOv5 [6] and Sketic [14], focused respectively on handwritten and hand-drawn logic circuits, demonstrating improved performance in component detection and simulation. The work in [15] proposed a combined recognition system for identifying components, mesh, branches, and nodes in circuit diagrams, while [16] presented an approach for detecting electrical component information and interconnections through instance segmentation. Similarly, various other techniques have been proposed for circuit recognition and sketch understanding, such as Hidden Markov models (HMMs) [17], [18], [19] to detect sketches as temporal signals produced by continuous pen movements. Other approaches involve architectures like MobileNet SSD [20], CNN classifiers [21], Faster R-CNN [22], and fully-connected neural networks [23].

Recent studies have expanded beyond component recognition to automating netlist generation in hand-drawn electrical circuit diagrams. One paper [24] employed optical character recognition (OCR) to detect characters, symbols, and numeric values in circuits and generate a ready-to-use netlist for circuit simulation. In another study [25], an artificial neural network (ANN)-based netlist generator achieved a component classification accuracy of 98%. Likewise, a paper [26] introduced a deep neural network algorithm aimed at detecting hand-drawn circuit diagrams within digitized electronic circuits and generating corresponding netlists. In a separate work by [27], a combination of image processing and machine learning techniques was applied to derive netlists from hand-drawn circuit diagrams. This involved utilizing line length ratios to identify components, as well as employing HOG feature extraction with SVM classification specifically for resistors, diodes, and inductors. Moreover, both [25] and [27] integrated optical character recognition (OCR) techniques to extract textual and numerical information from the components in order to assign their values within the netlist.

While a substantial amount of literature is available on symbol and text recognition, there are comparably fewer studies that directly tackle the issue of detecting connectivity and automating the simulation of identified circuit diagrams. The paper [28] attempts to automate the complete circuit design process, achieving a 98% accuracy in component detection and classification using a CNN model and sliding window techniques. However, it focuses mainly on basic electrical circuits. Likewise, a circuit simulator called "Voltique Designer" [29] was created for Android devices, merging the convenience of hand sketching on smartphones with the capabilities of a SPICE-based simulation engine, providing a seamless learning experience for students. There are significantly fewer papers focused on power converter circuits. One of them [30] presented a Bayesian Regularization-based artificial neural network (BR-ANN) and random forest (RF) approach with bootstrap aggregation to simulate the steady-state responses of power converters. Another paper [31] proposed an end-to-end framework to identify and simulate hand-drawn power converter circuits, but only non-isolated converters are considered. YOLOR is used to detect components, achieving an mAP of 91.6%.

This paper proposes an end-to-end computer vision-based framework that generates the netlist of a hand-drawn schematic for an isolated or non-isolated power converter (in continuous conduction mode, CCM or discontinuous conduction mode, DCM) and performs an automatic circuit simulation. To establish the framework, this research employs a wide range of state-of-the-art object detection models, including YOLOR, YOLOv7, and YOLOv8, to accurately detect electronic components, achieving a significantly higher accuracy up to 96.7%. Most often a straightforward Euclidean distance approach is utilized to establish connectivity between the symbols, but this proves inadequate for intricate and densely structured network architectures. To overcome this drawback, the proposed framework adopts a graph traversal technique to identify the shortest connected paths between the components. The main contributions of this paper are summarized as follows.

1) The proposed method encompasses a comprehensive computer vision-based framework from identifying circuit topologies for conducting automated simulations targeting power converters that include complex arrangements of electrical and magnetic elements like power transistors and transformers.
2) The challenging task of detecting nodes and connectivity within the converter circuit diagrams is achieved via a collection of algorithms, encompassing traditional image processing techniques and deep learning models.
3) The automated graph generation step incorporated into the framework enables the instant generation of graphical representations and adjacency matrices for hand-drawn circuit schematics.
4) The ablation study conducted on various converter topologies encompassing both non-isolated and isolated converters operating in both CCM and DCM modes, utilizing the state-of-the-art object detection models demonstrates a high net accuracy in netlist generation.

III. PROPOSED METHODOLOGY

The proposed framework comprises eight major stages for converting the manually drawn circuit diagram into a digital format and subsequently automating the simulation process. Each step is briefly described in this section, and their intermediate outputs, considering a buck converter, are demonstrated in Fig. 1. The entire operational procedure of the proposed method is succinctly outlined in Algorithm 1, highlighting common and custom image processing functions.

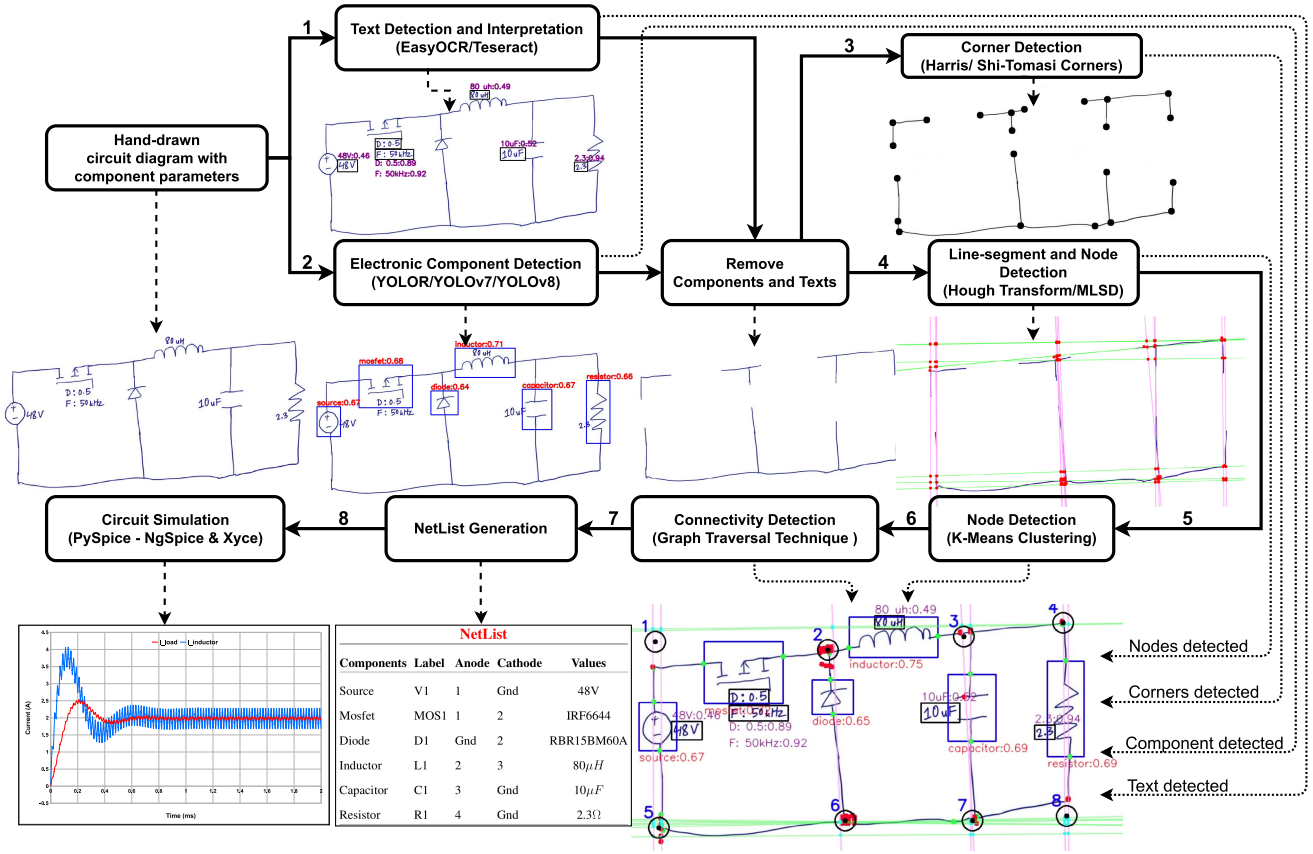

FIGURE 1. The overall architecture of the proposed framework combining computer vision-based segmentation and identification of hand-drawn power converter circuit diagram followed by the auto-generation of circuit netlist and the automated circuit simulation is presented. Each stags are numbered i.e., 1: text detection, 2: component detection, 3: corner detection, 4: line-segment detection, 5: nodes detection and clustering, 6: connectivity detection, 7: netlist generation, and 8: circuit simulation.

Algorithm 1 : Proposed End-to-End Pipeline

▷*Different model options provided for user selection*
corner-detectors ← {Harris operator, Shi-Tomasi operator}
component-detectors ← {YOLOR, YOLOv7, YOLOv7-tiny, YOLOv8m, YOLOv8n}
line-segment-detectors ← {Hough Transform, MLSD}
OCR ← {EasyOCR, Tesseract}
▷*Sequential Order of Functions used for Circuit NetList Generation*
grayImage ← ***rgb2gray***(rgbImage)
texts ← ***OCR***(grayImage)
components ← ***component-detectors***(rgbImage)
corners ← ***corner-detectors***(grayImage)
nodes1, vLines, hLines ← ***line-segment-detectors***(grayImage)
nodes2 ← ***get-point-of-intersections***(vLines, hLines)
nodes3 ← ***terminals-component-matching***(grayImage, corners)
kNodes ← ***kMeans-clustering***(nodes1, nodes2, nodes3)
netList ← ***node-connectivity***(components, kNodes, textBoxes, corners)
▷*Run Circuit Simulation*
converter ← ***identify-converter***(components)
analysis-results ← ***simulation***(netList)

A. DATASET PREPARATION

An open-sourced dataset - Handwritten Circuit Diagram Images, CGHD (version 6) [32] which consists of 2208 raw images with nearly 185,641 bounding box annotations, and 58 object classes is used in this paper to train the models. To narrow down the scope of the problem, only 14 major classes that frequently appear in most of the power converter circuits, including helper symbols such as nodes, junctions, and crossovers, are considered in this paper. CGHD is the only publicly available dataset for hand-drawn electrical circuit diagrams. In addition, 176 manually annotated power converter circuits and 432 null images (that do not contain any electrical circuit diagrams) were added to fine-tune the existing dataset for the problem at hand. The dataset includes schematics drawn on different surfaces using various writing instruments and markers of different colors. The samples also contain distortions such as buckling, kinking, bending, spots, trans-illumination, and paper cracks. The images are captured from different view angles and at various lighting conditions. The data set combines the IEEE/ANSI and IEC/DIN symbols for most generic electronic components. In addition, to prevent memory overloading and enhance model training, the dataset is modified by excluding irrelevant components (e.g., logic circuits, texts) and grouping functionally similar components (e.g., transistors, IGBTs, MOSFETs, and manual switches). Prior to feeding the images into the neural networks,

several preprocessing techniques are applied, including auto-orientation and resizing each image to dimensions of 640 × 640. Simultaneously, diverse data augmentation methods were employed to significantly increase the overall size of the training dataset. These augmentation techniques encompass orientation flipping, rotation, zooming, shearing, hue adjustment, brightness modification, image blurring, and the introduction of occlusions and noises. The purpose of employing some of these augmentations is to bolster the model's robustness against camera artifacts and varying environmental conditions. Consequently, the training data has been expanded by more than tenfold, explicitly increasing the sample count from 2,300 to 23,075. The complete code repository [33] and a link to the updated dataset [34] can be accessed online.

B. ELECTRONIC COMPONENT DETECTION

A range of cutting-edge object detection models, primarily from the YOLO family, such as YOLOR, YOLOv7, YOLOv8, and their corresponding lightweight versions, are evaluated to detect various components in the images. These models are trained on a server with NVIDIA Tesla V100-PCIE-16GB, 16151MiB hardware specifications. The fully trained models are subsequently tested on a personal computer equipped with NVIDIA GeForce GTX 1650-4GB.

C. HANDWRITTEN TEXT RECOGNITION

Texts are the vital information displayed on a circuit diagram through which users can transcribe components' values and simulation parameters such as switching frequency, duty cycle, simulation run-time, etc. Hence, identifying, interpreting, and coupling the texts in the diagrams to the respective components are the crucial steps in netList autogeneration and performing circuit simulation. For scene text detection and interpretation, various open-source optical character recognition (OCR) tools are available, i.e., Tesseract, EasyOCR, MMOCR, PaddleOCR, etc. In the proposed framework, Tesseract [35] and EasyOCR [36] are implemented to read the textual information in the circuit diagram. Thereafter, the association between the text and the component is obtained using the Euclidean distance between their respective bounding boxes' midpoints.

1) EASYOCR

It supports 80+ languages and popular scripts, including Latin, Chinese, Arabic, Devanagari, Cyrillic, etc. It uses Character Region Awareness for Text Detection (CRAFT) for accurate scene text detection and supports multilingual text recognition. To interpret the detected texts, a CRNN-based model is used that is built upon three main components: 1) RestNet and VGG for feature extraction, LSTM for text sequence labeling, and Connectionist Temporal Classification (CTC) for text sequence decoding.

2) TESSERACT

It was originally developed by Hewlett-Packard (HP) between 1984 and 1994 and is now available as an open-source OCR engine. Tesseract supports more than 100 languages, and it supports Unicode (UTF-8). The recent version, i.e., Tesseract-4, uses an LSTM-based OCR engine focused on online text recognition.

D. LINE SEGMENT DETECTION

Detecting line segments and other geometric shapes in an image has broad application in various fields, such as robotics, image processing, and computer graphics. There are several approaches to detect line segments - ranging from traditional methods such as classical Hough Transform (HT) [37] to more recently developed deep learning-based techniques [38]. Line segment detection (LSD) application in natural scenes has received increased attention, primarily due to its notable use in the detection of lanes and structural features within autonomous driving systems and other computer vision applications. Due to its computational intensity, unstable performance, and inaccuracies, such as misdetection of certain nodes in the circuit diagram, HT is excluded from utilization in this framework. In this paper, a lightweight deep learning-based model for line segment detection, named Mobile-LSD (MLSD) [39], is employed for real-time implementation. MLSD is tailored for resource-constrained platforms, such as mobile and edge computing devices, and adeptly identifies lines and curves within circuit diagrams.

1) MOBILE LINE SEGMENT DETECTION (M-LSD)

A lightweight line segment detection model is used for real-time implementation, optimized for resource-constrained platforms, i.e., mobile and edge-computing devices. A paper [39] has built an efficient and lightweight deep neural network architecture with minimized backbone network layers that remove typical multi-module processes for predicting lines. The proposed architecture uses a novel training scheme - a combination of SoL (Segments of Line segment) augmentation, matching, and geometric loss, which captures additional geometric clues needed for accurate line predictions. This architecture reduces the model size by 2.5%, whereas the inference speed on the GPU is improved by 130.5% compared to the previous best real-time LSD, i.e., TP-LSD-Lite. Consequently, MLSD can achieve higher model inference time, i.e., 56.8 FPS and 48.6 FPS on Android and iPhone phones, respectively.

E. CORNER DETECTION

An additional corner detection stage is added in the proposed framework to detect nodes, junctions, and terminals in the circuit diagram. Corner detection is explicitly used to identify the points of interest in the image invariant to rotation, translation, and scaling. Adding corner detection provides several benefits: finding the missing circuital nodes,

locating anode and cathode terminals for each component, and detecting the components' placement or orientation in the circuit diagram. The two algorithms used primarily for corner detection in computer vision are the Harris operator [40] and the Shi-Tomasi operator [41]. In this paper, both techniques are explored; however, the Shi-Tomasi operator is observed to perform relatively better in the context of hand-drawn circuit diagrams.

1) HARRIS CORNER DETECTION

In this technique, corners are detected as regions in an image with significant gradient changes in the intensity of the pixel in all directions. As an improvement over Moravec's corner detector, Chris Harris and Mike Stephens introduced this method in 1988.

2) SHI-TOMASI CORNER DETECTION

Shi-Tomasi detection is similar to the Harris corner detector but with slightly different corner selection criteria. Harris operator picks the corner based on the score computed by two eigenvalues. In contrast, the Shi-Tomasi operator uses eigenvalues only to check whether a pixel is a corner. However, corner selection is based on a score computed with a different approach.

F. CIRCUITAL NODE AND CONNECTIVITY DETECTION

The detected line segments are classified as horizontal or vertical based on the slopes, as shown in (1), and (2) and the circuit nodes are identified through their points of intersection [31]. It is observed that the chosen LSD models detect superfluous horizontal and vertical lines, thus leading to an excessive number of intersection points. To address this issue, K-means clustering is employed to group closely located points. However, a drawback of this technique is that the number of clusters, or nodes in the circuit, must be predefined, and doing so in the context of a dense circuit network is tedious and time-consuming. To overcome this challenge, Kirchhoff's law [15]: $M = B - N + 1$ (where M is no. of meshes, B is no. of branches, and N is no. of nodes in a circuit) that defines a relationship between node, branch, and mesh in an arbitrary circuit design is utilized to obtain the complete topology state for the given circuit.

$$slope(m) = \begin{cases} \infty, & \text{if } x_1 = x_2 \\ \left| \dfrac{y_2 - y_1}{x_2 - x_1} \right|, & otherwise, \end{cases} \tag{1}$$

$$Line = \begin{cases} horizontal, & \text{if } m \leq 0^\circ + \epsilon \\ vertical, & \text{if } m \geq 90^\circ - \epsilon \ \text{ or } \ m = \infty \end{cases} \tag{2}$$

where ϵ represents a threshold value set by the user to define the level of alignment of horizontal and vertical lines and limited to the range $\epsilon \in \{0, 5\}$. A graph traversal (or graph search) technique is adopted to assign nodes to a particular component, given that their distance lies within a boundary of a predefined threshold. Several node detection methods are stacked sequentially to populate nodes in a list, programmed so that no nodes are missed. Eventually, an entire pool of nodes is condensed down to an appropriate number using K-means clustering. The working mechanism of this technique is concisely depicted in Algorithm 2.

Algorithm 2 : Node and Connectivity Detection

```
function node-connectivity(components,kNodes,textBoxes, corners)
    for all cmp ∈ components do
        for all node ∈ kNodes do
            distNodes ← Euclidean distance between cmp and node
        end for
        sortedNodes ← sort indexes and nodes in distNodes
        if targe-nodes=N then
            cmp-nodes ← kNodes[sortedNodes[0:N]][values]
        end if
        for all textbox ∈ text-boxes do
            distText ← Euclidean distance between cmp and textbox
        end for
    end for
    argminText ← index corresponding to the minimum distance
    component-value ← texts[argminText][label]
end function
```

G. NETLIST GENERATION

Netlist is a textual representation or description of electrical components and their values and interconnections in a given circuit. It is compiled with the general circuit information, including component types, values, node terminals, and model definitions of compact devices such as diodes, transistors, MOSFETs, etc. A structural configuration of a netlist typically consists of three major components: 1) labels and values, 2) connectivity information, and 3) hierarchical relationships between the components. Netlist provides a clear picture of the overall layout and functionality of the circuit design, which is ultimately fed into various circuit simulation and PCB design software such as LTSpice, PSpice, Altium, KiCad, etc. A converter topology can be acknowledged using the netlist information as demonstrated in Algorithm 3.

H. GRAPH GENERATION

Graphs serve in the structural representation of complex data to understand the interaction between different objects in the real world, such as social networks, biology networks, citation networks, traffic networks, molecular structures, etc. [42]. Mathematically, a graph is defined as a tuple $G = (V, E)$, where V represents a set of objects, called vertices or nodes, and E is a set of connections or linkages, called edges, which are designed to understand complex systems and relationships between their entities. In the realm of electrical circuitry, there is currently no publicly available graphical data suitable for training architectures like geometric deep learning and graph neural networks (GNN), which are among the fastest-growing classes in machine learning [43]. The framework is designed to transform schematic diagrams of electrical circuits into comprehensive graphical data.

Algorithm 3 : Converter Identification

```
function identify-converter(components)
    converter ← "Unknown Converter"                    ▷ define a variable
    [a]tfcenter_x ← abscissa of transformer bounding-box center
    switches ← {"switch","transistor","mosfet"}        ▷ define a set
    for all cmp ∈ components do
        if cmp≠"transformer" & (cmp lies between nodes 2 & 6) then
            if cmp="diode" then
                converter ← "Buck converter"
            else if cmp ∈ switches then
                converter←"Boost converter"
            else if cmp="inductor" then
                converter←"Buck-Boost converter"
            end if
            break
        else
            if [b]cmp_x <tfcenter_x & cmp ∈ switches then
                swtich-count← no. of cmp at left-side of transformer
            else if cmp_x > tfcenter_x & cmp="diode" then
                diode-count← no. of cmp at right-side of transformer
            end if
        end if
    end for
    if swtich-count=1 & diode-count=1 then
        converter ← "Flyback Converter"
    else if switch-count=2 & diode-count=4 then
        converter ← "Half-bridge Converter"
    else if switch-count=4 & diode-count=4 then
        converter ← "Full-bridge Converter"
    end if
end function
```

[a] tfcenter_x is x-coordinate of transformer's bounding-box center
[b] cmp_x is x-coordinate of the component's bounding-box center

I. CIRCUIT SIMULATION AND ANALYSIS

SPICE (Simulation Program with Integrated Circuit Emphasis) is a general-purpose program that simulates electrical circuits. It is robust, powerful, and has become the de facto standard circuit simulation tool in industries. Spice allows different types of analysis: 1) DC analysis for time-invariant sources, 2) Transient analysis for time-variant sources, and 3) AC analysis for small-signal analysis of the circuits. The spice engine parses the textual netList script and, henceforth, calculates a circuit's nodal voltages and branch currents. In the proposed framework, an open source Python circuit simulator, i.e., PySpice [44], is used to auto-simulate a hand-drawn circuit diagram. It runs Ngspice and Xyce circuit simulator engines on the back-end while providing users with a Pythonic front-end utility for convenient human-computer interaction. In contrast to the existing Spice-based simulators, an automated simulation framework offers a broader range of flexibility and interactive features for circuit design, such as an intelligent approach using textual and visual input, digital twin implementation [45], and hardware-in-the-loop (HiL) simulation.

IV. RESULTS AND DISCUSSION

To assess the performance of the proposed framework, different power converter topologies designed for a 250 W DC power system are taken into account. These topologies encompass non-isolated converters (such as buck and boost converters) and isolated converters (such as flyback and full-bridge DC-DC converters).

Comprehensive experiments are conducted using the latest object detection models, including YOLOR-P6, YOLOv7, YOLOv8, and their lightweight variants, aiming to enhance component detection accuracy and accelerate model inference speed. As detailed in Table 1, YOLOv8m achieves the highest detection accuracy of 89.90% mAP50 with test inferences of 13.37 FPS, while YOLOv8n achieves a slightly lower accuracy of 78% but at the highest FPS of 18.55. Model training to 100 epochs (number of iterations) for the YOLOv8 series took, on average, 80% less time than YOLOR and YOLOv7 series. Moreover, the fully trained model size for YOLOv8n is the least compared to any other architecture, while having the lowest number of network layers and trainable parameters. The lighter model size of YOLOv8 allows for conducive implementation even in lower-power edge computing devices such as FPGAs and smartphones. YOLOv8m and YOLOv8n are further trained at 250 epochs, thus improving the detection accuracy to 96.7% and 89.15%, respectively.

The proposed framework undergoes testing on a total of 140 freshly hand-sketched circuit diagrams representing three major non-isolated converters - buck, boost, and buck-boost. The net netlist generation accuracy, as depicted in (3), serves as a comprehensive measure of the combined accuracy across various stages within the framework. This encompasses the integration of diverse text interpretation, component detection, corner detection, and line segment detection models, culminating in the final netlist generation performance of the entire framework. As illustrated in Table 2, YOLOR demonstrates its efficiency by accurately generating netlists for 71 test images, achieving a high net accuracy of 50.71%. In contrast, YOLOv7 proves to be less effective, yielding only 56 correct netlists with a net accuracy of 40%. Remarkably, YOLOv8 stands out by successfully producing correct netlists for 134 test images, thereby contributing to the total netlist generation accuracy.

$$\eta_{net} = \eta_{\text{text}} \cdot \eta_{\text{component}} \cdot \eta_{\text{corner}} \cdot \eta_{\text{lsd}} \cdot \eta_{\text{netlist}} \tag{3}$$

where, η_{net} represents the overall netlist generation efficiency for the entire proposed framework, encompassing all intermediate stages. η_{text} denotes the combined accuracy of text detection and interpretation, while $\eta_{\text{component}}$ signifies the average accuracy in component detection. Furthermore, η_{corner} represents the accuracy of corner detection, η_{lsd} pertains to line-segment detection accuracy, and η_{netlist} encapsulates the combined accuracy of node detection, clustering, and connectivity detection stages.of 95.71%.

The computational performance of each of the stages in the proposed framework has been thoroughly examined and is presented in both Table 3 and Fig. 4. In Table 3, the computational time for each stage in the proposed framework is presented, considering different typologies. It is observed that each stage takes nearly the same amount of time to compute, regardless of the converter types. In particular, the

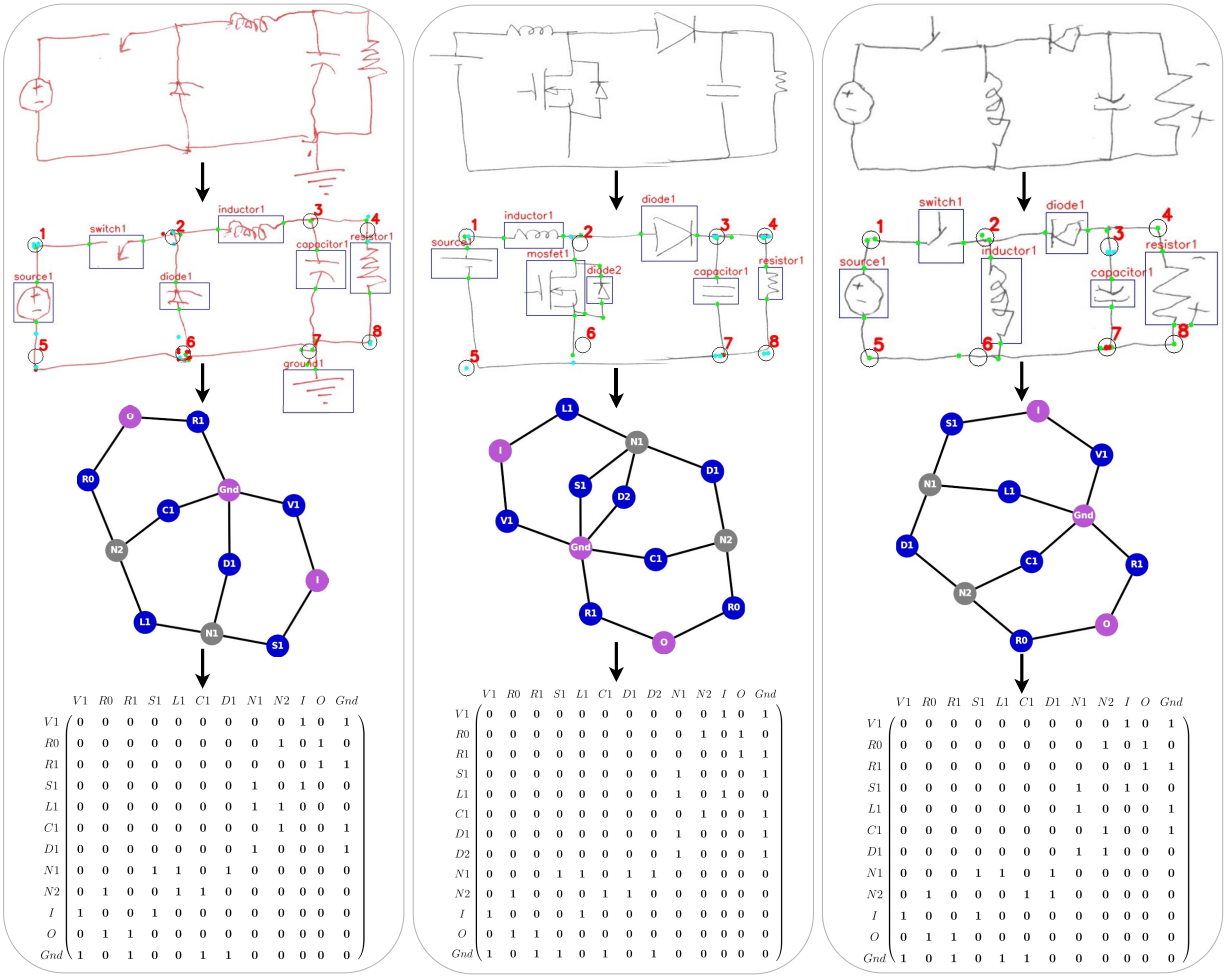

FIGURE 2. Automated graphical representation along with their respective adjacency matrices obtained for a hand-drawn schematic diagram for a (a) buck converter, (b) boost converter, and (c) buck-boost converter.

TABLE 1. State-of-the-art models trained for 100 epochs (inter-model comparison study) to detect electronic components.

Models	#Layers	#Parameters (Millions)	Flops (Giga)	Img. size	mAP50 %	mAP50-95 %	FPS	Training time (hrs.)	Model Size (MB)
YOLOR-P6	665	36.90	81.84	640	79.50	38.80	0.74	45.35	295.90
YOLOv7	415	37.28	105.40	640	84.30	45.40	0.96	42.12	75.00
YOLOv7-tiny	263	6.05	13.30	640	75.00	35.90	2.91	42.78	12.40
YOLOv8m	218	25.85	78.70	640	**89.90**	57.20	**13.37**	**8.52**	52.00
YOLOv8-nano	168	**3.01**	**8.10**	640	78.00	40.80	**18.55**	**8.72**	**6.20**

TABLE 2. Comparative analysis of combined net accuracy in netlist generation using various component detection models.

Models	Total Test Samples	Correct Samples	Accuracy
YOLOR	140	71	50.71%
YOLOv7	140	56	40.00%
YOLOv8	140	134	**95.71%**

circuit simulation stage emerges as the most computationally demanding, followed by text recognition. In contrast, corner detection and netlist generation stages impose a relatively minor computational burden.

An additional ablation study examines both the corner detection and line segment detection stages, evaluating the effectiveness of different combinations. Table 4 highlights that pairing Shi-Tomasi corner detection with MLSD line segment detection results in the highest netlist generation accuracy of 95.71%. In contrast, the Harris-HT pairing demonstrates the lowest accuracy.

When different component detection models are examined, almost all stages exhibit similar computational speeds, with the exception of the component detection stage. As depicted in Fig. 4, YOLOv7 requires an average of

TABLE 3. Time taken by each process (in seconds) with YOLOv8-based component detection model.

Converters Type	Text Recognition	Component Recognition	Corner Detection	Line-segment Detection	NetList Generation	Simulation	Entire Process Time (s)
Buck	1.4487	0.3069	0.0055	0.5342	0.0251	6.5384	8.86
Boost	1.4309	0.3266	0.0045	0.4854	0.0256	6.4913	8.76
Flyback	1.4737	0.3104	0.0064	0.4917	0.0380	6.5946	8.91
Fullbridge	1.4486	0.3146	0.0064	0.4835	0.0534	6.6485	8.96
Average	**1.4505**	**0.3146**	**0.0057**	**0.4987**	**0.0355**	**6.5682**	**8.87**

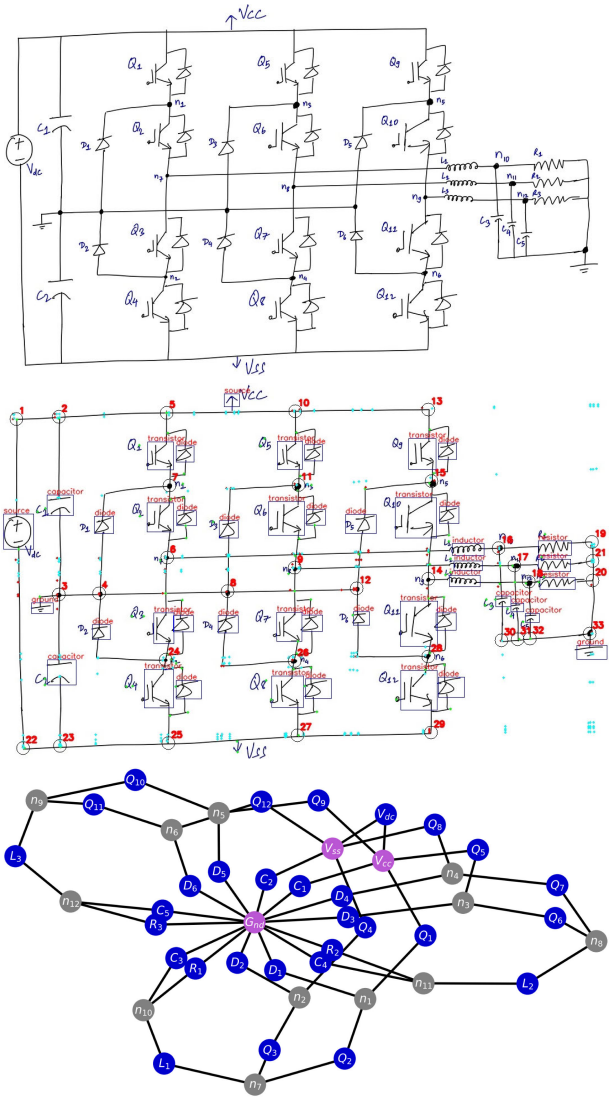

FIGURE 3. Hand-drawn schematic diagram of three level neutral point clamped (NPC) inverter, showing components and nodes identified, and the netlist graph constructed using proposed framework.

TABLE 4. An overall netlist generation accuracy under various combination of corner detection and line segment detection techniques.

Corner Detection \ Line Segment Detection	Hough Transform	MLSD
Harris	47.86%	60.00%
Shi-Tomasi	60.71%	**95.71%**

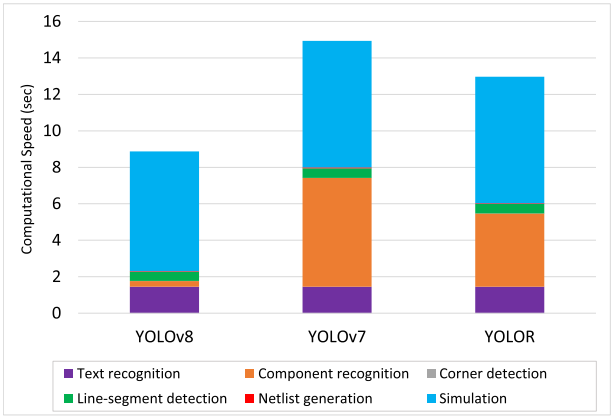

FIGURE 4. Computational Speed for the individual stages of the framework with different detection models.

5.9615 seconds to detect components, followed by YOLOR with 4.0145 seconds, while YOLOv8 is faster with an average of 0.3146 seconds. In a comparative analysis, the use of YOLOv8 as component detection model results in an average overall computational process time (ranging from component detection to automated simulation) of 8.87 seconds. This is followed by YOLOR with an average of 12.97 seconds, and YOLOv7 with 14.93 seconds.

A notable contribution of this study lies in the ability of the proposed framework to generate the network or graphical representation of hand-drawn schematic diagrams. Fig. 2 shows the graphs generated for the hand-drawn circuit diagrams of (a) the buck converter, (b) the boost converter (with the transistor featuring an anti-parallel body diode), and (c) the buck-boost converter. The innovative approach presented in this work involves generating graphs for various electrical circuit diagrams through hand-drawn sketches, offering the potential to create datasets for machine learning applications. The framework's efficacy in auto netlist generation and graph representation was also successfully evaluated on a hand-drawn schematic diagram of a complex three-level Neural Point Clamped (NPC) inverter [46], depicted in Fig. 3. Beyond the field of electrical circuits, this methodology can be potentially extended to other domains such as organic chemistry, social science, and medical science, enabling the

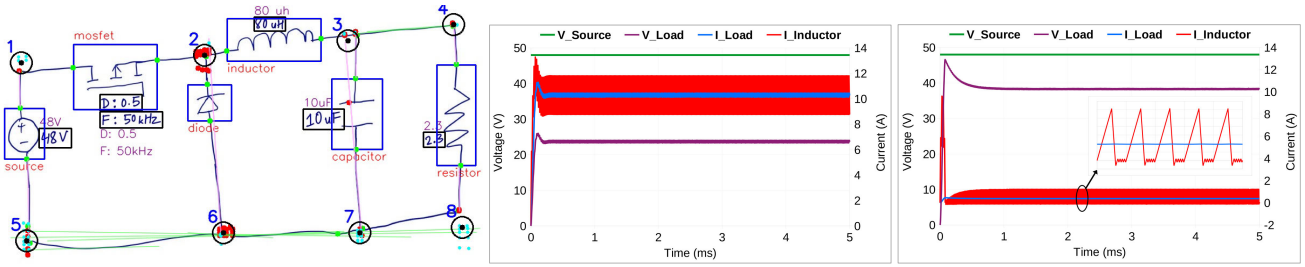

FIGURE 5. A hand-drawn schematic of a non-isolated buck converter with components, textual information, and nodes identified (left), and automated simulation results obtained under CCM mode operation (middle), and DCM mode operation (right).

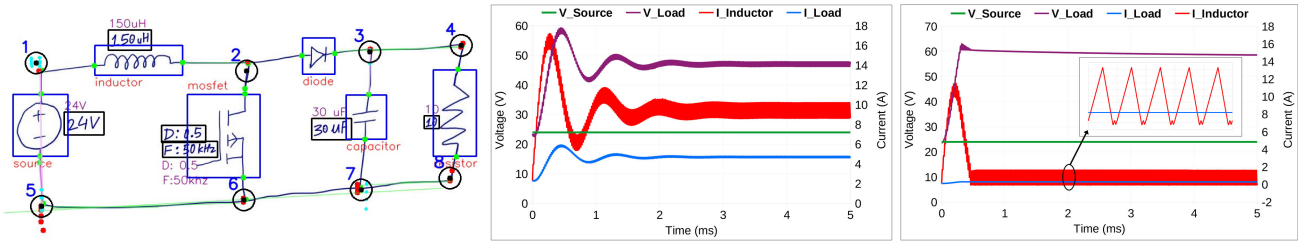

FIGURE 6. A hand-drawn schematic of a non-isolated boost converter with components, textual information, and nodes identified (left), and automated simulation results obtained under CCM mode operation (middle), and DCM mode operation (right).

creation of graphical datasets through roughly drawn hand sketches. The comprehensive set of netlists and graphical data derived from the complete proposed framework pipeline for 140 test images is available online [47]. Some samples of the hand-drawn circuit diagrams along with their netlist and graphical information are presented in Appendix B.

An automated simulation is performed for all these converters in both continuous-conduction-mode (CCM) and discontinuous-conduction-mode (DCM) operation. To validate the results obtained from an automated simulation, the topologies with identical configurations and parameters are simulated in LTSpice, and the resulting voltage and current waveforms are cross-examined as depicted in figures 5-9. Open-loop simulations for both automated and LTSpice-based framework are performed with 50 kHz switching frequency while fixing the duty cycle to 50% and maintaining consistent PWM rise/fall signal for all the considered power converters. The converter sizing and the component parameters are borrowed from [30]. The inductances and capacitances of the converters are optimized to maintain the ripple of the peak current of the inductor within the allowed range of 30% and the ripple of the peak load voltage within the range of 2%. Detailed information regarding these adjustments are shown in Table 5.

Figures 5, 6, 7, and 8 illustrate the hand-drawn schematics of non-isolated converters i.e., buck, boost, and isolated converters i.e., flyback, and full-bridge converter respectively, accompanied by the predicted bounding boxes for each of the components and their identified corresponding parametric information. The complete netlists obtained for each of the test converters are presented in Appendix A, i.e., Tables 6,

TABLE 5. Converters design parameters.

Converters	Type	Vin (V)	Duty (%)	Freq. (kHz)	Load (Ω)	Inductance (μH)	Capacitance (μF)
Buck	Non-Isolated	48	50	50	2.3	80	10
Boost	Non-Isolated	24	50	50	10	150	30
Flyback	Isolated	48	50	50	2.3	100	100
Full-bridge	Isolated	48	50	50	2.3	20	100

7, 8, and 9. The voltages' and currents' waveforms obtained through PySpice-based auto-simulation for the CCM and DCM modes of operation are displayed alongside the hand-drawn topologies. In order to compare the performance of the auto-simulation framework, an example scenario of an isolated DC-DC full-bridge converter (whose schematic is depicted in Fig. 9), which is manually built on LTSpice by replicating the component and simulation parameters from the handwritten schematic. Comparison of the simulation results obtained from both the automated and LTSpice methods reveals a nearly perfect match in voltage and current waveforms, including the corresponding ripple effects. This alignment is evident when examining the zoomed sections as highlighted within the respective figures. This near 100% match arises because both platforms utilize the SPICE engine in the background to simulate a provided circuit, whether it is created manually through schematic design in LTSpice or generated automatically through hand-drawn schematics with the proposed framework. In retrospect, it is important to highlight that despite distinct circuit design approaches, the shared reliance on the SPICE engine ensures consistent simulation results. However, minor discrepancies in simulation outcomes, especially in voltage and current

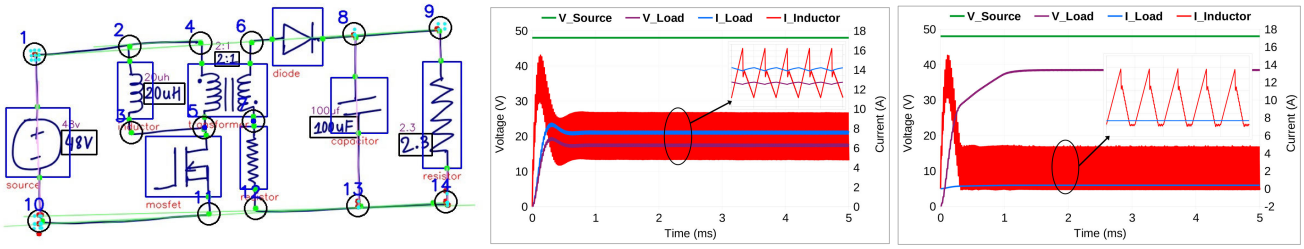

FIGURE 7. A hand-drawn schematic of a isolated flyback converter with components, textual information, and nodes identified (left), and automated simulation results obtained under CCM mode operation (middle), and DCM mode operation (right).

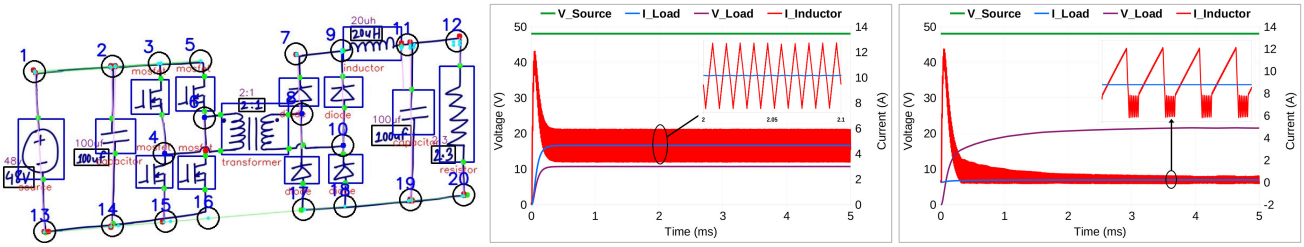

FIGURE 8. A hand-drawn schematic of an isolated DC-DC full-bridge converter with components, textual information, and nodes identified (left), and automated simulation results obtained under CCM mode operation (middle), and DCM mode operation (right).

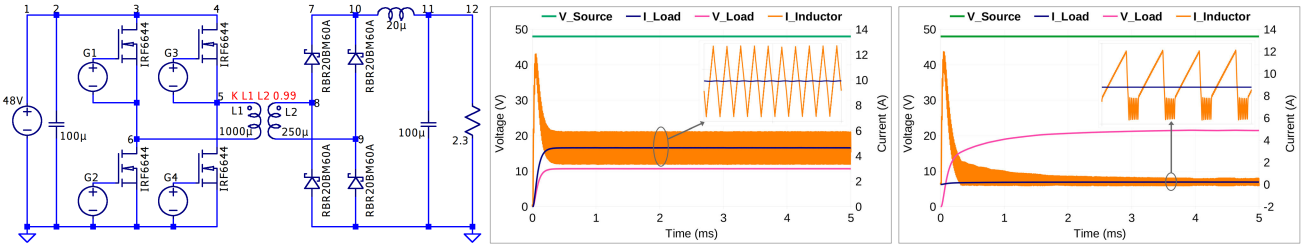

FIGURE 9. A schematic diagram of an isolated DC-DC full-bridge converter built on LTSpice with exactly similar parameters as in Fig. 8 (left), and simulation results obtained under CCM mode opertion (middle), and DCM mode operation (right).

ripple waveforms, can be ascribed to variances in simulation parameters, including sampling run-time, and the influence of internal parasitic elements within the circuits.

The proposed framework, the first of its kind in the field of electrical power conversion, combines various image processing and machine learning techniques, enhancing them through hyperparameter tuning. It applies refined methods in corner detection, MLSD, and YOLOv8 to better detect electrical components, node identification, and connectivity analysis. This approach marks a pioneering effort to streamline electrical circuit design and analysis.

V. CONCLUSION AND FUTURE WORK

This paper introduces a comprehensive computer vision-based framework that utilizes deep learning techniques to identify hand-drawn power electronic circuits. The framework enables the automatic generation of circuit netlists and facilitates instant simulation. The approach combines object detection models, OCR, and various image processing and machine learning techniques to identify components, interpret textual information, and trace connections. It employs classical techniques, such as Harris, Shi-Tomasi corner detection, Hough Transform, etc., to machine learning approaches such as MLSD to detect node and wire tracing throughout the circuit. Subsequently, K-means clustering groups neighboring nodes, while graph traversal method associates the identified text with respective bounding boxes. The framework is tested on different converters and shows excellent agreement with LTSpice results. The method achieves a high accuracy of 96.7% using YOLOv8m and 89.15% using YOLOv8n for component detection. Furthermore, its performance is assessed using 140 recently drawn hand-sketched schematics, achieving an overall netlist generation accuracy of approximately 95.71% with the YOLOv8-based component detection model. With all these benefits, the proposed framework can be used as an educational tool or as a quick-simulation application to learn or evaluate electrical circuits on computers or mobile devices. Further enhancement of the features of this framework can include automating the control circuit design, scaling for PCB board digitization and simulation, and integrating into HuggingFace Spaces using Gradio for online demonstrations.

TABLE 6. NetList obtained for buck converter.

Components	Anode terminal	Cathode terminal	Value
Source (V1)	1	5	48 V
Mosfet (S1)	1	2	d=0.5, 50 kHz
Inductor (L1)	2	3	80 μH
Diode (D1)	6	2	
Capacitor (C1)	3	7	10 μF
Resistor (R1)	4	8	2.3 Ω

* Terminals 5, 6, 7, and 8 are ground terminal

TABLE 7. NetList obtained for boost converter.

Components	Anode terminal	Cathode terminal	Value
Source (V1)	1	5	24 V
Inductor (L1)	1	2	150 μH
Mosfet (S1)	2	6	d=0.5, 50 kHz
Diode (D1)	2	3	
Capacitor (C1)	3	7	30 μF
Resistor (R1)	4	8	10 Ω

* Terminals 5, 6, 7, and 8 are ground terminal

TABLE 8. NetList obtained for flyback converter.

Components	Anode terminal	Cathode terminal	Value
Source (V1)	1	10	24 V
Inductor (L1)	2	3	20 μH
Mosfet (S1)	5	11	d=0.5, 50 kHz
Transformer (T1)	4, 6	5, 7	
Diode (D1)	6	8	
Capacitor (C1)	8	13	100 μF
Resistor (R1)	9	14	2.3 Ω

* Terminals 10, 11, 12, 13, and 14 are ground terminal

TABLE 9. NetList obtained for full-bridge converter.

Components	Anode terminal	Cathode terminal	Value
Source (V1)	1	13	48 V
Capacitor (C1)	2	14	100 μF
Mosfet (S1)	3	4	
Mosfet (S2)	4	15	
Mosfet (S3)	5	6	
Mosfet (S4)	6	16	
Transformer (T1)	6, 8	4, 10	
Diode (D1)	8	7	
Diode (D2)	17	8	
Diode (D3)	10	9	
Diode (D4)	18	10	
Inductor (L1)	9	11	20 μH
Capacitor (C2)	11	19	100 μF
Resistor (R1)	12	20	2.3 Ω

* Terminals 13, 14, 15, 16, 17, 18, 19, and 20 are ground terminal

A critical challenge in hand-drawn circuit recognition is the risk of missing components or nodes, which can result in the failure of the entire automated simulation process. Although the proposed tool for converting hand-drawn schematics to netlists and simulations offers significant advantages, it comes with a few limitations that need to be addressed through future work. The accuracy of component recognition and schematic interpretation heavily depends on the quality and consistency of the hand-drawn sketches, potentially leading to errors in complex or poorly drawn schematics. Integration with existing EDA tools might encounter compatibility issues, and professionals accustomed to traditional EDA software may be resistant to adopting new add-on features and workflows, especially if this tool does not seamlessly integrate with their established processes.

TABLE 10. NetList for hand-drawn circuit in Fig. 10.

Components	Anode terminal	Cathode terminal
Source1 (V1)	1	5
Mosfet1 (S1)	1	2
Diode1 (D1)	6	2
Inductor1 (L1)	2	3
Capacitor1 (C1)	3	7
Resistor1 (R1)	4	8

* For graph generation the terminals are renamed as 1: I, 2: N1, 3: N2, 4: O, 5,6,7,8: Gnd

APPENDIX A
NETLIST OF THE CONVERTERS CONSIDERED FOR AUTO-SIMULATION

See Tables 6–9.

APPENDIX B
NETLIST AND GRAPH INFORMATION OF SOME ADDITIONAL HAND-DRAWN CIRCUIT DIAGRAMS

See Figs. 10–12 and Tables 10–12.

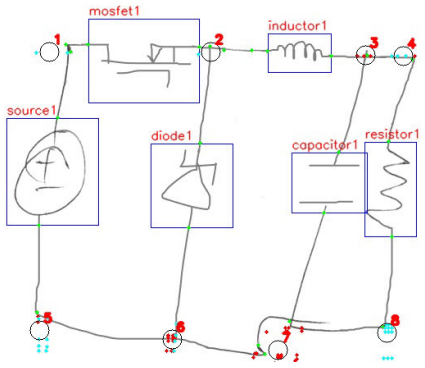

FIGURE 10. A hand-drawn schematic diagram of a non-isolated buck converter.

Graph Information of circuit drawn in Fig. 10

Nodes: V1, I, Gnd, D1, N1, N2, C1, L1, R1, O, S1
Edges: (D1, N1), (D1, Gnd), (N1, S1), (N1, L1), (Gnd, V1), (Gnd, R1), (Gnd, C1), (V1, I), (I, S1), (R1, O), (O, N2), (L1, N2), (N2, C1)
Number of nodes: 11
Number of edges: 13
Average degree: 2.3333

Graph Information of circuit drawn in Fig. 11

Nodes: V1, I, Gnd, D1, N1, N2, C1, L1, R1, O, S1
Edges: (V1, I), (V1, Gnd), (I, L1), (Gnd, C1), (Gnd, R1), (Gnd, S1), (D1, N1), (D1, N2), (N1, L1), (N1, S1), (N2, C1),

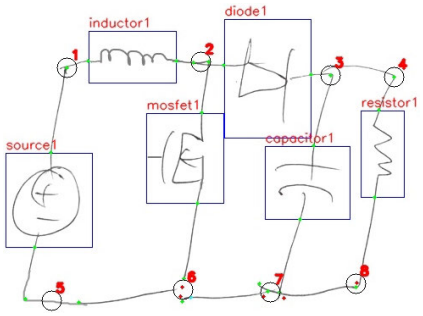

FIGURE 11. A hand-drawn schematic diagram of a non-isolated boost converter.

TABLE 11. NetList for hand-drawn circuit in Fig. 11.

Components	Anode terminal	Cathode terminal
Source1 (V1)	1	5
Mosfet1 (S1)	2	6
Diode1 (D1)	2	3
Inductor1 (L1)	1	2
Capacitor1 (C1)	3	7
Resistor1 (R1)	4	8

* For graph generation the terminals are renamed as 1: I, 2: N1, 3: N2, 4: O, 5,6,7,8: Gnd

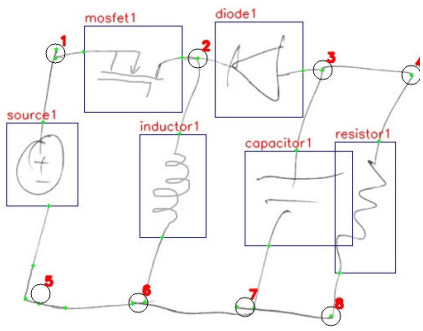

FIGURE 12. A hand-drawn schematic diagram of a non-isolated buck-boost converter.

TABLE 12. NetList for hand-drawn circuit in Fig. 12.

Components	Anode terminal	Cathode terminal
Source1 (V1)	1	5
Mosfet1 (S1)	1	2
Diode1 (D1)	3	2
Inductor1 (L1)	2	6
Capacitor1 (C1)	3	7
Resistor1 (R1)	4	8

* For graph generation the terminals are renamed as 1: I, 2: N1, 3: N2, 4: O, 5,6,7,8: Gnd

(N2, R0), (R1, O), (O, R0)
Number of nodes: 11
Number of edges: 13
Average degree: 2.3333

Graph Information of circuit drawn in Fig. 12
Nodes: V1, I, Gnd, D1, N1, N2, C1, L1, R1, O, S1
Edges: (V1, I), (V1, Gnd), (N1, L1), (Gnd, C1), (Gnd, R1), (I, S1), (D1, N1), (D1, N2), (Gnd, L1), (N1, S1), (N2, C1), (R1, O), (O, N2)
Number of nodes: 11
Number of edges: 13
Average degree: 2.3333

ACKNOWLEDGMENT

The authors express their gratitude to Roboflow Inc. for their assistance in managing the dataset, including providing the essential tools for annotating and enhancing the dataset. They would like to acknowledge the assistance provided by AI systems, including ChatGPT, in improving grammar, correcting typographical errors, and editing the sentences in this article, which has enhanced the clarity and coherence of their manuscript.

REFERENCES

[1] G. Jain, S. Chopra, S. Chopra, and A. S. Parihar, "TransSketchNet: Attention-based sketch recognition using transformers," in *Proc. ECAI*, 2020, pp. 2907–2908.
[2] R. Davis, "Magic paper: Sketch-understanding research," *Computer*, vol. 40, no. 9, pp. 34–41, Sep. 2007.
[3] K. M. Daly, "Hand-drawn graph problems in online education," M.S. thesis, Massachusetts Inst. Technol., Cambridge, MA, USA, 2015.
[4] S. Mani, M. A. Haddad, D. Constantini, W. Douhard, Q. Li, and L. Poirier, "Automatic digitization of engineering diagrams using deep learning and graph search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 673–679.
[5] S. Bickel, B. Schleich, and S. Wartzack, "Detection and classification of symbols in principle sketches using deep learning," *Proc. Des. Soc.*, vol. 1, pp. 1183–1192, Aug. 2021.
[6] S. Amraee, M. Chinipardaz, M. Charoosaei, and M. A. Mirzaei, "Hand-written logic circuits analysis using the Yolo network and a new boundary tracking algorithm," *IEEE Access*, vol. 10, pp. 76095–76104, 2022.
[7] L. Gennari, L. B. Kara, T. F. Stahovich, and K. Shimada, "Combining geometry and domain knowledge to interpret hand-drawn diagrams," *Comput. Graph.*, vol. 29, no. 4, pp. 547–562, Aug. 2005.
[8] F. C. A. Groen, A. C. Sanderson, and J. F. Schlag, "Symbol recognition in electrical diagrams using probabilistic graph matching," *Pattern Recognit. Lett.*, vol. 3, no. 5, pp. 343–350, Sep. 1985.
[9] J. Pu and D. Gur, "Automated freehand sketch segmentation using radial basis functions," *Comput.-Aided Des.*, vol. 41, no. 12, pp. 857–864, Dec. 2009.
[10] R. L. Naika, R. Dinesh, and S. Prabhanjan, "Handwritten electric circuit diagram recognition: An approach based on finite state machine," *Int. J. Mach. Learn. Comput.*, vol. 9, no. 3, pp. 374–380, Jun. 2019.
[11] H. Wang, T. Pan, and M. K. Ahsan, "Hand-drawn electronic component recognition using deep learning algorithm," *Int. J. Comput. Appl. Technol.*, vol. 62, no. 1, p. 13, 2020.
[12] S. Roy, A. Bhattacharya, N. Sarkar, S. Malakar, and R. Sarkar, "Offline hand-drawn circuit component recognition using texture and shape-based features," *Multimedia Tools Appl.*, vol. 79, nos. 41–42, pp. 31353–31373, Nov. 2020.
[13] M. Dey, S. M. Mia, N. Sarkar, A. Bhattacharya, S. Roy, S. Malakar, and R. Sarkar, "A two-stage CNN-based hand-drawn electrical and electronic circuit component recognition system," *Neural Comput. Appl.*, vol. 33, no. 20, pp. 13367–13390, Oct. 2021.
[14] M. Abdel-Majeed, T. Almousa, M. Alsalman, and A. Yosf, "Sketic: A machine learning-based digital circuit recognition platform," *TURKISH J. Electr. Eng. Comput. Sci.*, vol. 28, no. 4, pp. 2030–2045, Jul. 2020.
[15] Q. Li, D. Liang, Y. Xu, N. Xiao, and Y. Li, "Improved algorithm for circuit diagram image recognition," in *Proc. 2nd Int. Conf. Comput. Sci. Softw. Eng.* New York, NY, USA: Association for Computing Machinery, May 2019, pp. 96–101.
[16] J. Bayer, A. K. Roy, and A. Dengel, "Instance segmentation based graph extraction for handwritten circuit diagram images," 2023, *arXiv:2301.03155*.

[17] Y. Zhang, C. Viard-Gaudin, and L. Wu, "An online hand-drawn electric circuit diagram recognition system using hidden Markov models," in *Proc. Int. Symp. Inf. Sci. Eng.*, Dec. 2008, pp. 143–148.

[18] P. Sala, "A recognition system for symbols of electronic components in hand-written circuit diagrams," *Proc. IEEE*, pp. 257–286, Jun. 2004.

[19] T. M. Sezgin and R. Davis, "Sketch interpretation using multiscale models of temporal patterns," *IEEE Comput. Graph. Appl.*, vol. 27, no. 1, pp. 28–37, Jan. 2007.

[20] I. Singh, V. Singhal, D. Singhal, H. Devarajan, R. Verma, and S. Sofana Reka, "A system for conversion of hand-drawn electrical circuit to digital circuit: A deep learning approach," in *Proc. 12th Int. Conf. Comput. Commun. Netw. Technol. (ICCCNT)*, Jul. 2021, pp. 1–4.

[21] M. Günay, M. Köseoglu, and Ö. Yildirim, "Classification of hand-drawn basic circuit components using convolutional neural networks," in *Proc. Int. Congr. Hum.-Comput. Interact., Optim. Robot. Appl. (HORA)*, Jun. 2020, pp. 1–5.

[22] M. Günay and M. Köseoğlu, "Detection of circuit components on hand-drawn circuit images by using faster R-CNN method," *Int. Adv. Researches Eng. J.*, vol. 5, no. 3, pp. 372–378, Dec. 2021.

[23] M. Rabbani, R. Khoshkangini, H. S. Nagendraswamy, and M. Conti, "Hand drawn optical circuit recognition," *Procedia Comput. Sci.*, vol. 84, pp. 41–48, Jan. 2016.

[24] S. Sridar and K. Subramanian, "Circuit recognition using netlist," in *Proc. IEEE 2nd Int. Conf. Image Inf. Process. (ICIIP)*, Dec. 2013, pp. 242–246.

[25] A. E. Sertdemir, M. Besenk, T. Dalyan, Y. D. Gokdel, and E. Afacan, "From image to simulation: An ANN-based automatic circuit netlist generator (Img2Sim)," in *Proc. 18th Int. Conf. Synth., Modeling, Anal. Simulation Methods Appl. Circuit Design (SMACD)*, Jun. 2022, pp. 1–4.

[26] R. R. Rachala and M. R. Panicker, "Hand-drawn electrical circuit recognition using object detection and node recognition," *Social Netw. Comput. Sci.*, vol. 3, no. 3, pp. 1–11, May 2022.

[27] A. Mohan, A. Mohan, B. Indushree, M. Malavikaa, and C. P. Narendra, "Generation of netlist from a hand drawn circuit through image processing and machine learning," in *Proc. 2nd Int. Conf. Artif. Intell. Signal Process. (AISP)*, Feb. 2022, pp. 1–4.

[28] M. M. Arnadottir and O. Yngvason, "From component detection to simulation of analog circuits using CNNs," CS230: Deep Learn., Stanford Univ., Stanford, CA, USA, Tech. Rep. 47, Winter 2018.

[29] D. D. Xu, M. Hy, S. Kalra, D. Yan, N. Giacaman, and O. Sinnen, "Electrical circuit creation on Android," in *Proc. ITx, New Zealand's Conf. IT*, M. Lopez and M. Verhaart, Eds., Auckland, New Zealand: CITRENZ, 2014, pp. 210–216.

[30] H. S. Krishnamoorthy and T. N. Aayer, "Machine learning based modeling of power electronic converters," in *Proc. IEEE Energy Convers. Congr. Expo. (ECCE)*, Sep. 2019, pp. 666–672.

[31] B. Bohara and H. S. Krishnamoorthy, "Computer vision based framework for power converter identification and analysis," in *Proc. IEEE Int. Conf. Power Electron., Drives Energy Syst. (PEDES)*, Dec. 2022, pp. 1–6.

[32] F. Thoma, J. Bayer, Y. Li, and A. Dengel, "A public ground-truth dataset for handwritten circuit diagram images," in *Proc. Int. Conf. Document Anal. Recognit.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 12916, 2021, pp. 20–27.

[33] B. Bohara and H. S. Krishnamoorthy (Aug. 2023). *PedApp: AI Tool for Automated Circuit Simulation*. [Online]. Available: https://github.com/Varat7v2/PEDApp-AI-Tool-for-Automated-Circuit-Simulation

[34] B. Bohara and H. S. Krishnamoorthy, Jun. 8, 2023, "Hand-drawn power converter circuit diagrams," IEEE Dataport, doi: 10.21227/e2tg-zj02.

[35] R. Smith, "An overview of the tesseract OCR engine," in *Proc. 9th Int. Conf. Document Anal. Recognit. (ICDAR)*, Sep. 2007, pp. 629–633.

[36] JaidedAI. (Mar. 2020). *EasyOCR*. Github Repository. [Online]. Available: https://github.com/JaidedAI/EasyOCR

[37] P. V. C. Hough, "A method and means for recognition complex patterns," U.S. Patent 3 069 654 A, 1962.

[38] H. Zhang, Y. Luo, F. Qin, Y. He, and X. Liu, "ELSD: Efficient line segment detector and descriptor," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 2949–2958.

[39] G. Gu, B. Ko, S. Go, S.-H. Lee, J. Lee, and M. Shin, "Towards light-weight and real-time line segment detection," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2022, vol. 36, no. 1, pp. 726–734.

[40] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Alvey Vis. Conf.*, 1988, pp. 23.1–23.6.

[41] J. Shi and Tomasi, "Good features to track," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 1994, pp. 593–600.

[42] X. Guo and L. Zhao, "A systematic survey on deep generative models for graph generation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5370–5390, May 2023.

[43] P. Reiser, M. Neubert, A. Eberhard, L. Torresi, C. Zhou, C. Shao, H. Metni, C. van Hoesel, H. Schopmans, T. Sommer, and P. Friederich, "Graph neural networks for materials science and chemistry," *Commun. Mater.*, vol. 3, no. 1, pp. 1–18, Nov. 2022. [Online]. Available: https://www.nature.com/articles/s43246-022-00315-6

[44] F. Salvaire. (2015). *PySpice: Simulate Electronic Circuit Using Python and the Ngspice/Xyce Simulators*. [Online]. Available: https://github.com/PySpice-org/PySpice

[45] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2405–2415, Apr. 2019.

[46] R. Maheshwari, S. Munk-Nielsen, and S. Busquets-Monge, "Design of neutral-point voltage controller of a three-level NPC inverter with small DC-link capacitors," *IEEE Trans. Ind. Electron.*, vol. 60, no. 5, pp. 1861–1871, May 2013.

[47] B. Bohara and H. S. Krishnamoorthy, Jan. 29, 2024, "Hand-drawn electrical circuit diagram graph generation," IEEE Dataport, doi: 10.21227/dw8x-qp19.

BHARAT BOHARA (Member, IEEE) received the Bachelor of Electrical and Electronics Engineering degree from Kathmandu University (KU), Nepal, in 2015, and the Master of Technology (M.Tech.) degree in energy and control systems engineering from Indian Institute of Technology (IIT) Bombay, in 2019. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Houston, Houston, TX, USA. His research interests include artificial intelligence (AI) applications in power electronics, machine learning, reinforcement learning, generative modeling, and efficient implementation of neural networks on embedded systems and FPGA.

HARISH SARMA KRISHNAMOORTHY (Senior Member, IEEE) received the B.Tech. degree in electrical and electronics engineering from the National Institute of Technology (NIT), Tiruchirappalli, India, in 2008, and the Ph.D. degree in electrical and computer engineering (ECE) from Texas A&M University, College Station, TX, USA, in 2015.

Since August 2017, he has been a faculty with the ECE Department, University of Houston (UH), where he currently holds the rank of an Associate Professor. He has more than 110 conference/journal papers in refereed publications, two granted U.S. patents, and three U.S. patent applications. He has also contributed to a book chapter for the IET. His research interests include high-density power conversion using SiC and GaN devices, and statistical methods for better quality and reliability of power converters, advanced power electronics, and control for applications, such as renewable energy, electric vehicles, oil and gas, and 4G/5G envelope tracking. He was named an 'OTC Emerging Leader' by the Offshore Technology Conference, in 2022, and an Early Career Research Fellow (ECRF) by the Gulf Research Program of the U.S. National Academies. He received the Teaching Excellence Award, in 2021, and the UH College of Engineering Research Excellence Award, in 2022. In 2023, he received the NSF CAREER Award and the IEEE PELS Young Professional Exceptional Service Award. He is also on the organizing committees of IEEE APEC-2023, ECCE-2023, and INTELEC-2024. He is an Associate Editor of IEEE TRANSACTIONS ON POWER ELECTRONICS (TPEL) and the Standards Liaison of the IEEE PELS TC7.