Computer Vision based Framework for Power Converter Identification and Analysis

Bharat Bohara, *Student Member, IEEE*
Electrical and Computer Engineering
University of Houston
Houston, United States
bbohara@uh.edu

Harish S. Krishnamoorthy, *Senior Member, IEEE*
Electrical and Computer Engineering
University of Houston
Houston, United States
hskrishn@uh.edu

2022 IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES) | 978-1-6654-5566-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/PEDES56012.2022.10080752

***Abstract*—This paper proposes a computer vision-based framework to identify the topology of a hand-drawn image or schematic of a power converter circuit and perform automated simulations. For component detection, a deep learning-based model, i.e., YOLOR, the state-of-the-art object detection model, is used with a model accuracy mAP0.5 of 91.6%. In order to trace the wire connections in the circuit diagram, a classical Hough transform algorithm is used. The nodes of the circuit diagram are identified with K-Means clustering of the point-of-intersections between the horizontal and vertical lines. With the help of the position of the components detected and the nodes, a netlist of the circuit diagram is generated that can be fed into any spice-based circuit simulator. An automated simulation of the schematic of the power converter is done with the help of PySpice - an open-source python module, to simulate the electronic circuit that runs a spice-based simulation engine, i.e., ngspice and xyce on the backend. The proposed methods have been verified using the main non-isolated DC-DC converters (buck, boost, and buck-boost). It is envisioned that this framework can also act as an educational tool. Moreover, the proposed concepts can be extended to create fully automated and optimal power converter designs for practical applications.**

***Index Terms*—Power Electronic Converters, Hand-drawn circuit, Electronic component detection, Automated Simulation, Netlist, PySpice**

I. Introduction

Digitizing hand-drawn electronic circuit diagrams or schematics requires accurate detection models for components and precisely tracing the circuit connection/wires as well as nodes. Some challenges in identifying the components are frequent variations in the individual drawing sketch, pencil/ink qualities, and unsteady wriggle in the sketch; for example, the same component can be minute to significantly different every time drawn. This paper proposes an end-to-end computer vision-based framework to identify a hand-drawn converter circuit diagram or schematic and then simulate the identified topology. The proposed concepts will help engineers and students in power electronics in the form of a learning toolkit that can be implemented on mobile platforms. This will further expand the understanding of how power converters would behave under various conditions and with different combinations of component arrangements.

978-1-6654-5566-4/22/$31.00 ©2022 IEEE

Some researchers/learners are more creative with paper and pencil rather than repeatedly designing the schematics on the simulation software, which is a time-consuming and tedious process. On the other hand, drawing schematics on paper is mostly the first draft for any circuit and saves much time. Once the user is satisfied with the design, they can proceed with the simulation software for validation and further work. In this paper, the component values are randomly generated within the specified range. Nevertheless, in the future, deep learning-based optical character recognition (OCR) models can be integrated to read the values from the schematics automatically. This research can be further extended to simulate more extensive and complex circuit diagrams so that researchers can take advantage of reduced research time frames. One of the challenges of hand-drawn circuit recognition is that missing any of the components or nodes in the circuit will result in the failure of the entire netlist generation and automated simulation process.

II. Literature Review

There is hardly any literature on an end-to-end framework designed for an integrated hand-drawn circuit topology recognition system with an automated simulator. No work was found on the automated simulation of the hand-drawn schematics of the power electronic circuits or converters. However, most literature is limited to component recognition in hand-drawn schematic diagrams of electronic circuits. Paper [1] used a combined architecture with local binary pattern (LBP) and statistical pixel density distribution to extract the components' features, SVM to classify them, and finite state machine (FSM) to identify the sequence of the identified components: finally, this entire framework was used to detect the overall circuit topology. Paper [2] proposed a CNN-based sparse auto-encoder method to extract features of the hand-drawn electronic components and softmax to classify them with an overall model accuracy of 95%. Paper [3] proposed a combined HOG-based texture feature descriptor and shape-based features that included centroid distance, tangent angle, and chain code histogram to identify the hand-drawn analog and digital components and a sequential minimal optimization (SMO) to classify them with an overall efficiency of 93.83%. Paper [4] presented a two-stage CNN-based hand-drawn ana-

log and digital component recognition system with the overall model efficiency of 97.33%: in the first stage, similar-looking components were grouped together, and in the later stage, the components were classified.

Some of the recent works on hand-drawn electric circuit diagrams push further to the netlist generation in an automated fashion. Paper [5] developed an ANN-based netlist generator that classifies the components with an accuracy of 98%. Paper [6] proposed a deep neural network-based algorithm to detect hand-drawn electrical circuit diagrams for digitized electronic circuits and generate the netlist. Paper [7] used image processing and machine learning techniques to produce a netlist of hand-drawn circuit diagrams. This paper employed an idea of line length ratios to identify the voltage source, capacitor and ground, then used a histogram of oriented gradients (HOG) for feature extraction and support vector machine (SVM) to classify the components (resistor, diode, and inductor). All of the papers [5], [6], [7] employ Hough Transform to detect the line segments and the intersection of the detected lines to find the circuital nodes. Additionally, papers [5], [7] used optical character recognition (OCR) to extract the textual, and numerical information of the identified components to assign their values in the netlist configuration.

There is significantly less work focused on the automated simulation of the identified circuit diagrams. Paper [8] proposed a method to simulate hand-drawn circuit diagrams through smartphones by using augmented reality and deep capsule networks targeted to engineering education. The authors claimed that the capsule networks required less training data to achieve comparable performance results as the CNN architectures and that it could identify the components of any scale and rotations with an accuracy of 96%. The model was used to simulate more straightforward circuit arrangements, but not the power converters - which consist of discontinuous switching behaviors, which are typically difficult to analyze. However, paper [9] proposed a Bayesian Regularization (BR)-based ANN and random forest (RF) with a bootstrap aggregation to model the steady-state responses of the power converters.

In this paper, the state-of-art object detection model, i.e., YOLOR (You Only Learn One Representation) [10] is used to detect the electronic components with a mAP0.5 of 91.6%, Hough Transform [11] to trace the wire connections, and K-mean clustering to identify the nodes in the hand-drawn circuit diagram. The proposed end-to-end computer vision-based framework is taken forward to generate the netlist of the hand-drawn schematic of a power converter and automatically simulate the circuit.

III. Proposed Computer Vision Based Method

A scanned image of a hand-drawn schematic of a power converter is first resized to an appropriate dimension and then fetched into a CNN-based component detection model. The state-of-art object detection model, i.e., YOLOR, is used in this study to detect the electronic components in the circuit diagram. Once the components are identified, wire connections need to be traced to find the circuital terminal nodes. One of the methods is to trace the wires in the circuit by finding all the possible lines in an image and locating the point of intersection between the lines. Initially, a huge number of lines will be predicted due to the distorted wire connection and noises on the sketch. So, it is a better idea to keep those lines that are perfectly aligned, either horizontal or vertical, while neglecting the rest of the wires. A Hough Transform algorithm is used to detect all the possible lines in the circuit. But, before applying the Hough transform, all the components are removed from the image to prevent the lines from getting predicted because of the components' geometrical shape. The predicted lines are segregated into horizontal and vertical lines by conditioning over their slopes. To elaborate it, a line is categorized as horizontal if its slope is within the range $m \in \{0^\circ - 5^\circ\}$, and vertical if its slope is within the range $m \in \{85^\circ - 90^\circ\}$ as shown in Eqn. 2.

A terminal node is located by finding the point-of-intersection of each horizontal and vertical line. However, since the number of intersection points exceeds the number of nodes in the circuit, some approaches, such as euclidean distance and K-means clustering, can be useful. Both methods were applied, and K-means clustering performed better in all the power converter circuits. Since the chosen converters (buck, boost, and buck-boost) are usually represented with a schematic consisting of eight nodes, the number of K-means clusters is taken to be eight. The centroid point of each of the predicted clusters is considered as the terminal nodes, which are numbered separately. The component positions are located in the schematic diagram by computing the euclidean distance between the center of each component's bounding box and all the identified nodes. Finally, using the component labels and their respective position coordinates, a netlist for the specified circuit is generated that can be loaded to any simulation platform to simulate the circuit. To elucidate the proposed method in more detail, the workflow algorithm adopted in this study is divided into six main subsections, as illustrated in fig. 1.

A. Dataset Preparation

Since there was no publicly available dataset on electronic components detection (ECD) during the time of doing this work, a custom dataset was made by manually annotating 176 hand-drawn electrical circuit diagrams using RoboFlow's annotation tool [12]. Various data augmentation techniques are applied, such as horizontal and vertical flip, rotation, zoom, shear, hue, brightness, saturation, exposure, blur, and adding random noise on the images, and hence, the size of the dataset is increased by 3x, i.e., from 176 to 528. The image quality is improved by applying preprocessing techniques such as auto-orient, scaling to higher resolution, i.e., 416x416, and auto-adjusting the image contrast through the adaptive equalization method. Altogether 12 different components are labeled, such as a resistor, capacitor, diode, MOSFET, etc. Besides labeling the components in the power converter circuits, different symbolic representations of the individual components are

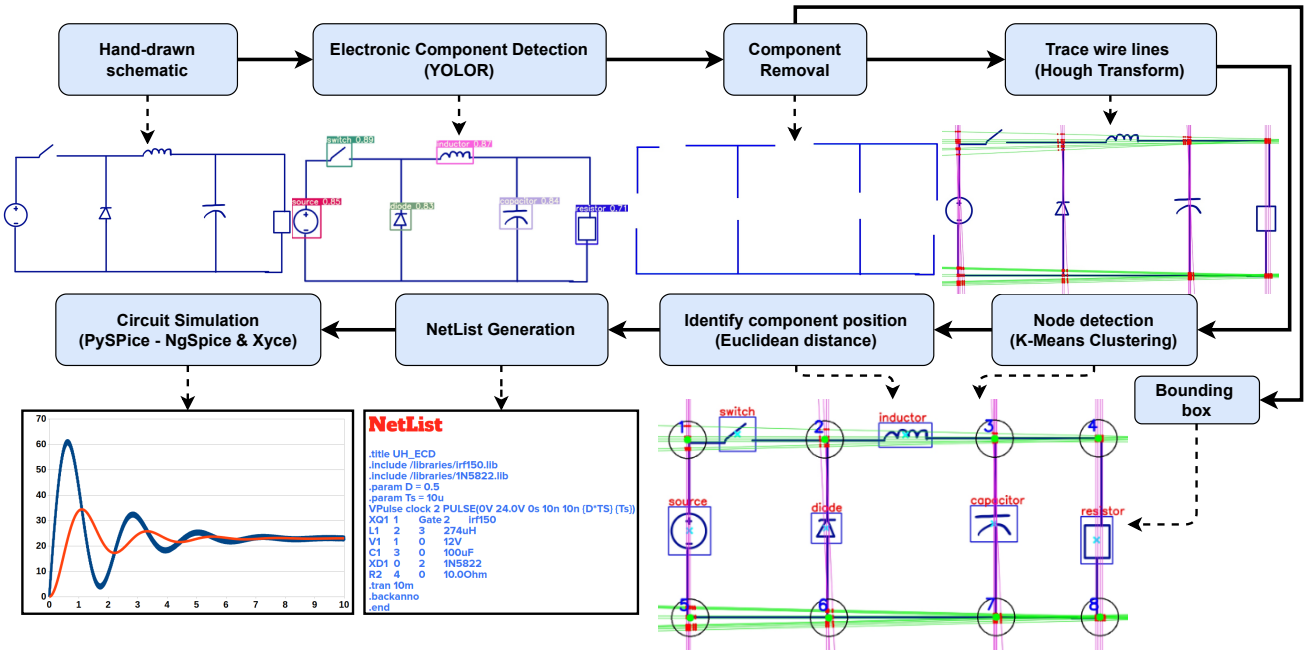

Fig. 1: Proposed framework for the automated simulation of the hand-drawn schematic of the power converters that are identified with the computer vision techniques

also annotated separately to balance the component class distribution. However, some components like resistors and capacitors dominate, whereas MOSFET and switches are underrepresented. Maintaining the balance between the classes to avoid the domination or inferiority of any of the components will be done in future work to make the model unbiased for all the component types.

B. Electronic Component Detection

Components are detected using a deep learning architecture, i.e., YOLOR, the state-of-the-art object detection model in terms of both the model accuracy and inference time. Though identifying the individual components and nodes and their position in an image is time-consuming and complicated, it has several advantages. For instance, with component and node placement information, we can create a netlist that can be directly used in any third-party software for circuit simulation and PCB design. Similarly, with the component detection methods, there are no limitations on the type of circuit topology to which this automated simulation technique can be implemented. Because of this flexibility, different representations of each component are also included in the dataset, along with the annotated power converter circuit diagrams.

C. Line Segment Detection

Detecting lines or geometric shapes in an image has a broad significance in image processing and computer graphics. Line segment detection in natural scenes is attracting more attention these days due to its application in detecting lanes for automated driving and other computer graphics and vision applications. In this study, Hough Transform (HT) is used to trace wire connections in the circuit. HT is a feature extraction technique that is usually used for detecting simple geometric shapes, such as lines, circles, and other geometric shapes in an image. The main idea behind HT is that all the geometric shapes in an image are parameterized with an angle and a radius (polar coordinates). A line is detected by voting the maximal local response points in the transformed parametric space through edge detectors (Canny or Sobel), and hence, it is also known as a line-to-point transform. One of the main advantages of HT is occlusion insensitive, but it is computationally expensive, and the performance is unstable. A line can be represented by using a pair (ρ, θ) in a polar coordinate system, as shown in Eqn. 1, where r is the shortest distance from the origin to the line.

Edge Detection: In order to apply Hough Transform, an edge detection must first be run on the desired image. This image processing technique detects the outlines or boundaries of objects or shapes in an image. In image processing, edges are identified with the sudden change in the pixel intensity in the image. In this study, a canny edge detection algorithm is used, where the sharpness of the edges can be adjusted via the hysteresis thresholding technique.

$$r_\theta = x\cos\theta + y\sin\theta, \quad \forall \ r > 0, 0 < \theta < 2\pi \tag{1}$$

A line in a polar coordinate system is detected with the points estimated by the edge detector. The minimum number of points required to estimate a line is adjusted with a threshold value: the higher the value, the more points will be required, and as a result, fewer lines will be detected; identically, more lines will be detected with the lower threshold value. Hence, the number of lines to be detected can be controlled by adjusting the threshold value. In the future, this threshold parameter

can be made to be adaptive to nullify the possible variation and noise in the hand-drawn circuit diagram. Eventually, the line segments obtained from the Hough Transform are separated into horizontal and vertical lines based on the slopes, as shown in Eqn. 2.

$$\text{slope}(m) = \begin{cases} \infty, & \text{if } x_1 = x_2 \\ \left|\frac{y_2 - y_1}{x_2 - x_1}\right|, & \text{otherwise} \end{cases}, \tag{2}$$

$$\text{Line} = \begin{cases} \text{horizontal}, & \text{if } m \leq 0^\circ + \epsilon \\ \text{vertical}, & \text{if } m \geq 90^\circ - \epsilon \text{ or } m = \infty \end{cases} \tag{3}$$

where ϵ is a user-defined limit that specifies how perfectly aligned the horizontal and vertical lines ought to be considered. This paper chooses this limit to be in the range $\epsilon \in \{0, 5\}$.

D. Node Recognition

Once the vertical and horizontal lines are separated out, the circuital nodes are identified through their intersection points. The point of intersection between any two line segments $L_1 : ax + by + c = 0$ and $L_2 : a'x + b'y = c'$ intersecting at a point (x, y) can be computed by using Eqn. 4.

$$(x, y) = \left(\frac{b_1c_2 - b_2c_1}{a_1b_2 - a_2b_1}, \frac{c_1a_2 - c_2a_1}{a_1b_2 - a_2b_1}\right) \tag{4}$$

where, $a = y_1 - y_2, b = x_2 - x_1, c = x_1y_2 - x_2y_1$ and $a' = y_1' - y_2', b' = x_2' - x_1', c' = x_1'y_2' - x_2'y_1'$, assuming (x_1, y_1) and (x_2, y_2) points lie on line L_1, and (x_1', y_1') and (x_2', y_2') lie on line L_2. However, a huge number of horizontal and vertical lines will be detected and hence, innumerable points of intersection and, consequently, an excessive number of nodes. In order to group up the closer points of intersection by the distance between them, a K-Means clustering method is employed. One limitation of using the K-Means clustering technique is that the number of clusters to form (or nodes in the circuit) need to be defined before running the program. Specifying the number of nodes only once will work fine for identical circuit diagrams such as buck, boost, and buck-boost converters. However, for varying circuit designs, defining the number of nodes repetitively would be a hectic job. In future work, finding the exact number of nodes in an arbitrary hand-drawn circuit design will be introduced using an appropriate image processing technique.

E. Netlist Generation

A netlist is a textual format of the description of the connectivity of the electronic components in an arbitrary electrical circuit. It consists of information such as component names, component values, nodes, and the model definition of a diode, transistor, or MOSFET. A general netlist structure is built upon three major components, i.e., a) component labels and the values, b) component connectivity information (anode and cathode), and c) hierarchical relationships of the electronic components. Explicitly it is designed to define the overall layout and functionality of the circuit design. Netlist is usually used as an input file in various circuit simulators and hardware compiler software such as SPICE, LTSpice, PSpice, etc. The detailed version of the netlist can be generated by adding additional device information such as part number, component values, and their physical dimensions; this can be used as an input file to numerous PCB designing software. An intelligence-based algorithm can be developed on top of the netlist to allow automated placement of the components, optimal wire routing, and ultimately generating the industry-ready manufacturing files (BOM, Gerber files).

F. Circuit Simulation

An automated simulation is done with the help of PySpice [13], an open-source python-based module to simulate an electronic circuit - that is interfaced with Ngspice and Xyce circuit simulators. It combines the power of python programming language with the spice-based electric circuit simulator. PySpice parses the textual netlist script and generates the equivalent pythonic architecture of the circuit or directly instantiates the circuit to simulate programmatically. It runs spice-based Ngspice and Xyce circuit simulator engines on the backend while rendering python-based frontend utility to the users for easy and flexible human-computer interactions. One of the advantages of using PySpice over the existing spice-based simulators is that it provides the users with a broad scope of flexibility and interactive feature of a circuit design. For instance, adding intelligence to the simulation, automated circuit design upon the textual or visual inputs, digital-twin, and hardware in-loop simulation may be some add-on features.

IV. Results and Discussion

The hand-drawn schematics of the non-isolated power converter topologies, i.e., buck, boost, and buck-boost, are designed for a 200 W DC power system to validate the simulation results of this work. The identical topologies with precisely similar component values and the parameters are simulated in LTSpice, and voltage and current waveforms from both the platforms are comparatively studied as shown in figs 2 to 7. For open-loop simulation on both the PySpice and LTSpice platforms, the switching frequency of $50kHz$, a duty of 60%, and PWM signal rise/fall times are kept the same for all the power converter circuits. However, the component value ranges are taken from the paper [9]. The inductance and capacitance values for all the converters are fine-tuned such that the inductor peak-to-peak current ripple is within an allowable range of 30%, and the load peak-to-peak voltage ripple is within 2% range, as shown in table I.

TABLE I: Parameters used in designing the converters

Converters Type	Vin (V)	Duty ($\%$)	Frequency (kHz)	Load (Ω)	Inductance (μH)	Capacitance (μF)
Buck	48	60	50	4	120	10
Boost	24	60	50	18	280	16
Buck-Boost	24	60	50	6.2	172	30

Figures 2, 4, and 6 illustrates the hand-drawn schematic of a buck converter, boost converter, and buck-boost converter respectively, along with their predicted bounding boxes for each of the components. The voltage and current waveform obtained from the PySpice-based automated simulation framework are also depicted alongside the hand-drawn schematics. Similarly,

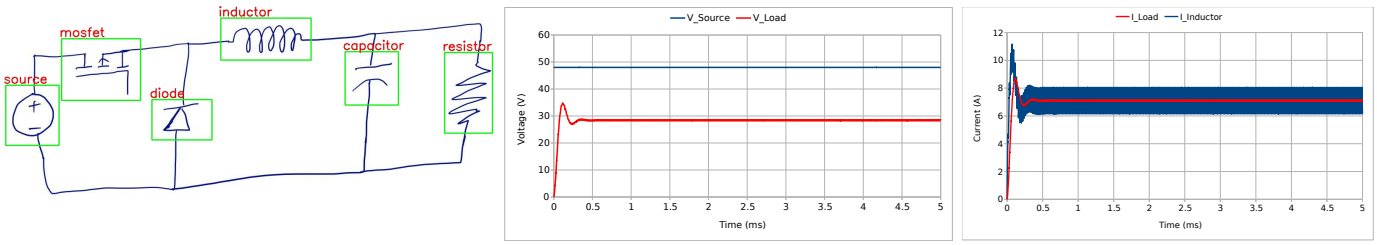

Fig. 2: Hand-drawn schematic of a buck converter with voltage and current waveform obtained from the proposed framework

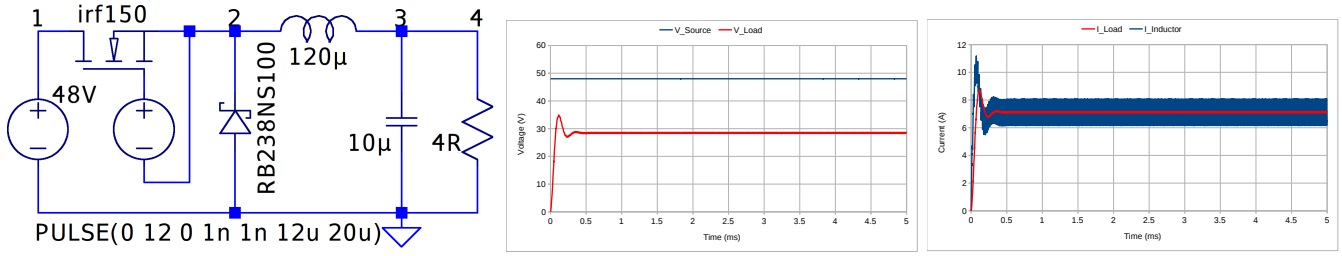

Fig. 3: Schematic of a buck converter with voltage and current waveform obtained from the LTSpice simulation

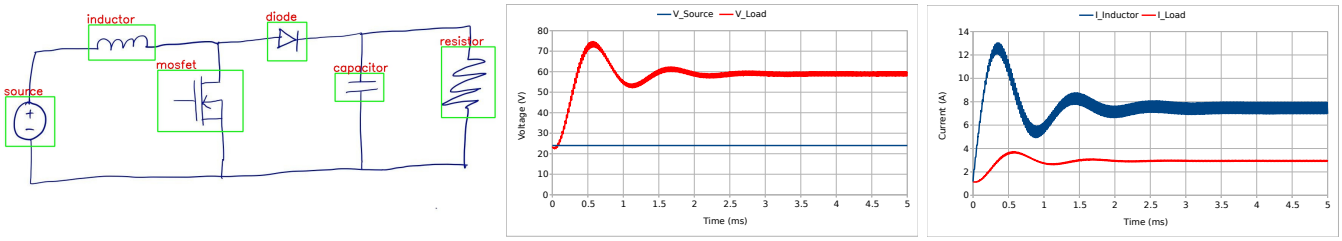

Fig. 4: Hand-drawn schematic of a boost converter with voltage and current waveform obtained from the proposed framework

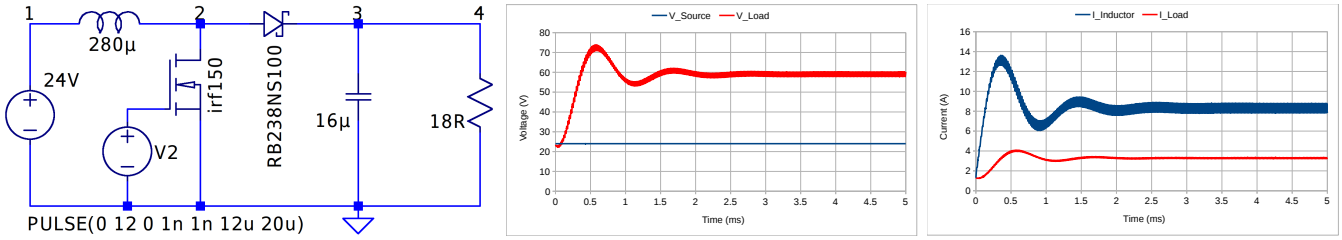

Fig. 5: Schematic of a boost converter with voltage and current waveform obtained from the LTSpice simulation

the figures 3, 5, 7 shows the schematic diagram of a buck converter, boost converter, and buck-bost converter, respectively, that are manually designed in LTSpice with same values of the components, and the simulation parameters. From the simulation results obtained from both the proposed method and the LTSpice, it is observed that the voltage and current waveform match perfectly with each other. The simple reason behind nearly the 100% match between their waveform is that both the platforms, i.e., LTSpice and PySpice (NgSpice), run on the same SPICE simulation engine on the backend. In other words, though the method or way of performing the circuit simulation is different, i.e., a) manually designed schematics on the LTSpice and b) automated netlist generated from the hand-drawn schematics in the proposed framework, both will be simulating the circuits with the same SPICE engine. However, a slight difference in the simulation results of the LTSpice and the proposed PySpice-based framework can be observed in the converters' power losses, as shown in table II.

TABLE II: Converter power losses observed in LTSpice vs. PySpice simulation designed for 200W DC system

Converters	LTSpice Simulation			PySpice Simulation		
	Avg. P_{in} (W)	Avg. P_{out} (W)	P_{loss} $(\%)$	Avg. P_{in} (W)	Avg. P_{out} (W)	P_{loss} $(\%)$
Buck	206.84	202.12	2.28	206.8	202.03	2.31
Boost	201.28	192.15	4.53	201.2	192	4.57
Buck-Boost	204.1	191	6.42	203.87	190.58	6.52

V. Conclusion

This paper proposes a computer vision-based end-to-end deep learning framework to identify hand-drawn power elec-

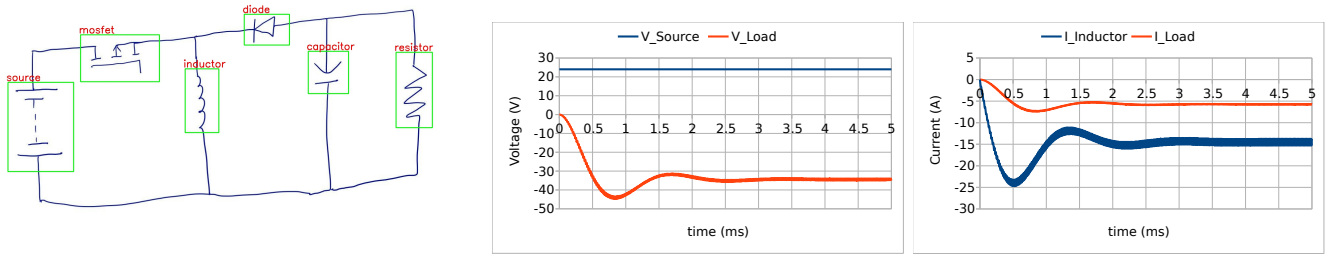

Fig. 6: Hand-drawn schematic of a buck-boost converter with voltage & current waveform obtained from proposed framework

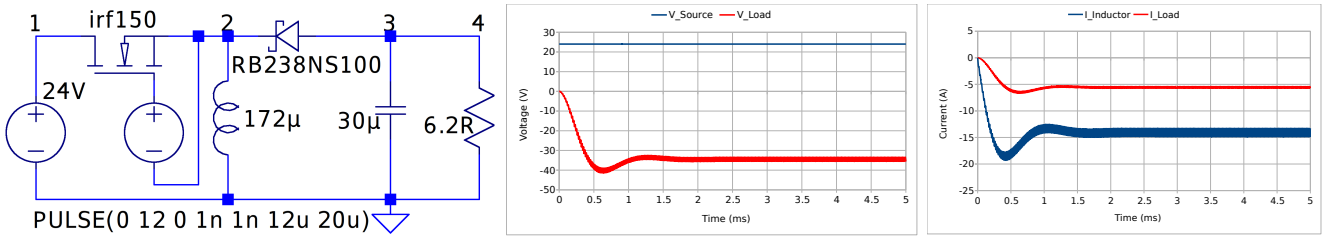

Fig. 7: Schematic of a buck-boost converter with voltage and current waveform obtained from the LTSpice simulation

tronic circuits or converter schematic diagrams for real-time automated simulation. A state-of-the-art object detection model - YOLOR, is used to detect the components with mAP0.5 of 91.6%, and the Hough Transform is used to trace the wires in the circuit diagram. The circuit terminal nodes are identified by applying K-means clustering on the point-of-intersections of the horizontal and vertical lines detected by the Hough transform. A netlist for the circuit is generated with the detected components and the terminal nodes. A python-based power electronic simulator, i.e., PySpice, is used to simulate the circuit. The proposed method is tested on the basic power electronic converters, i.e., buck, boost, and buck-boost. The PySpice-based simulated voltage/current waveform of the converters matches precisely to the LTSpice simulation results keeping the components values and the simulation parameters the same. In future work, the classical Hough transform will be replaced with a deep-learning-based, highly accurate line segment detector so that wire connections with any wriggle and curvatures can be accurately identified. This work can be pushed further to identify the real-time surface-mounted electronic components on the PCB to simulate an actual hardware configuration.

VI. Acknowledgement

The authors would like to thank Roboflow Inc. for supporting with the dataset management by providing the necessary features to annotate and augment the dataset.

References

[1] R. Lakshman Naika, R. Dinesh, and S. Prabhanjan, "Handwritten electric circuit diagram recognition: An approach based on finite state machine," *International Journal of Machine Learning and Computing*, vol. 9, no. 3, pp. 374–380, jun 2019.

[2] H. Wang, T. Pan, and M. K. Ahsan, "Hand-drawn electronic component recognition using deep learning algorithm," *International Journal of Computer Applications in Technology*, vol. 62, no. 1, pp. 13–19, 2020. [Online]. Available: https://dl.acm.org/doi/abs/10.1504/ijcat.2020.103905

[3] 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 and Applications*, vol. 79, no. 41-42, pp. 31 353–31 373, nov 2020. [Online]. Available: https://link.springer.com/article/10.1007/s11042-020-09570-6

[4] 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 Computing and Applications*, vol. 33, no. 20, pp. 13 367–13 390, oct 2021. [Online]. Available: https://link.springer.com/article/10.1007/s00521-021-05964-1

[5] 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)," *Proceedings - 2022 18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods, and Applications to Circuit Design, SMACD 2022*, 2022.

[6] R. Reddy Rachala and M. Raveendranatha Panicker, "Hand-Drawn Electrical Circuit Recognition Using Object Detection and Node Recognition," vol. 3, p. 244, 2022. [Online]. Available: https://doi.org/10.1007/s42979-022-01159-0

[7] 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," *2022 2nd International Conference on Artificial Intelligence and Signal Processing, AISP 2022*, 2022.

[8] M. Alhalabi, M. Ghazal, F. Haneefa, J. Yousaf, and A. El-Baz, "Smartphone Handwritten Circuits Solver Using Augmented Reality and Capsule Deep Networks for Engineering Education," *Education Sciences 2021, Vol. 11, Page 661*, vol. 11, no. 11, p. 661, oct 2021. [Online]. Available: https://www.mdpi.com/2227-7102/11/11/661/htm https://www.mdpi.com/2227-7102/11/11/661

[9] H. S. Krishnamoorthy and T. Narayanan Aayer, "Machine Learning based Modeling of Power Electronic Converters," *2019 IEEE Energy Conversion Congress and Exposition, ECCE 2019*, pp. 666–672, sep 2019.

[10] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "You Only Learn One Representation: Unified Network for Multiple Tasks," may 2021. [Online]. Available: https://arxiv.org/abs/2105.04206v1

[11] P. V. C. Hough, "A method and means for recognition complex patterns; US Patent: US3069654A," *US Patent*, p. 6, 1962.

[12] B. Dwyer and J. Nelson, "Roboflow (version 1.0) [software]." [Online]. Available: https://roboflow.com/

[13] "Simulate Electronic Circuit using Python and the Ngspice / Xyce Simulators — PySpice 1.6 documentation." [Online]. Available: https://pyspice.fabrice-salvaire.fr/releases/v1.6/