Distributed Systems: PNG2SPICE

WiSe 2023/2024

Group PNG2SPICE (TU-Berlin)

Jakob Tschavoll
j.tschavoll@campus.tu-berlin.de
485081

Kristof Konya
konya@campus.tu-berlin.de
485076

Contents

1 Motivation

As analog circuit designers, we have observed that the recreation of reference designs, while essential for quality designs and educational purposes in analog circuitry, often involves repetitive and manual processes comprised of tedious copying of electrical parts and their parameters into electronic design automation (EDA) and simulation software. From a hobbyist perspective, this task, although critical, presents an opportunity for automation to enhance efficiency. Within professional context, it's important to point out that analog electronic design is not only intricate but also incurs significant costs. These costs can impede technological advancements within organizations. Despite the apparent need, the market currently lacks a freely available tool that addresses this specific challenge. This gap in the market underscores the potential value of a solution like PNG2Spice, which automates the conversion of image-based reference designs into usable formats for EDA software, thereby streamlining the design process in both educational and professional settings.

2 Problem description

The analog electronic circuit design process is a sophisticated and methodical approach that encompasses a series of stages, each critical for the successful development of functional and efficient circuits. Analog electronic circuit design remains a cornerstone in the field of electronics, crucial for a myriad of applications. In analog electronic circuit design, reference designs – pre-developed circuit layouts and schematics – hold substantial importance, serving as critical tools that guide and streamline the design process. They offer a starting point for determining the feasibility and scope of the new design, based on proven frameworks, in order not "to reinvent the wheel".

There now exists a gap between freely accessible visual representations of schematics in the form of image files (e.g., PNG format) and their virtual representations within EDA software. While a proven schematic may act in determined ways when combined with other established designs or in isolation, they may behave vastly different when applied to novel designs. To avoid problems during development, simulation software like the widely-recognized industry standard for circuit simulation LTspice, plays a key role. However, wanting to simulate new designs now forces the designer to manually pick and place every part of the desired schematic manually, having to cross-reference an image and the UI of the simulation program. This presents a significant challenge in utilizing these designs for simulation and further development. Developers are faced with very repetitive, from an information-perspective redundant tasks that take up time that could have been spent creating new designs.

Addressing this gap, the PNG2Spice tool emerges as a pivotal innovation. It enables designers to convert image-based reference designs into the LTspice format (.asc), a widely-recognized industry standard for circuit simulation. This conversion capability streamlines the integration of reference designs into the design workflow, significantly enhancing efficiency and accuracy.

3 Related Work

The topic of converting one representation of an electrical schematic into another has been around since two decades. While today classification tasks as needed in the recognition of schematics leads to the usage of neural networks, researchers laid the foundation for schematic recognition even before practical examples of such networks even existed. In 2005 Bailey and Moretti proposed a schematic recognition approach based on image analysis and rule-based templates of primitives (lines, circles, arrows) [1]. They were able to reduce a schematic of adequate resolution down to primitives, which were associated with symbols based on a table of rules for each symbol. Such rules included length of adjacent lines and their relationships to other lines in terms of total length, angle or number of connections. This approach, they claim, makes the system expandable to other kinds of components. The output of their approach is a netlist without component description, as no efforts were made to include optical character recognition (OCR). They additionally acknowledge the difficulty of recognizing different styles of the same component. Using today's standard methods involving neural networks, Dey et al. proposed a two-stage CNN for the classification of hand-drawn components [2]. The first stage was used to classify a component's group to distinguish conceptionally different parts such as logic gates, transistors and passive components from each other. The second stage could then use the output of the first stage to activate a specifically trained CNN to further classify the component. As this approach only deals with component classification, no comments on logical schematic recognition were made. In similar and more applicable fashion, GitHub-user *Contigiani* [3] created a repository under the "unlicense"-license which uses a CNN approach to classify electrical components based on a purely augmented dataset of schematic symbols. A transfer-learning approach using the pre-trained VGG16 model [4] is compared to an untrained model. This experiment proves the effectiveness of transfer-learning based on powerful general-purpose models and the importance of data augmentation in case of data scarcity. Complete schematic recognition approach was proposed by Rachala and Panicker [5], who used a transfer learning model of YOLOv5 [6] for object detection on hand-drawn schematics involving passive components and sources. The detection of symbols reaches a 98% accuracy, while the overall correct classification of components and their location within the circuit reach 80% accuracy. Here, the symbols location and area of interest are extracted first and then classified. In a second step, already detected components are removed from the schematic, leaving only the wires, which are then highlighted by a Hough-transform and used to obtain intersection points. Lastly, the relationship of components and intersections is used to obtain the terminals of the components. While this work shows the most promising results, it does not discuss how the obtained data can be streamlined into an actual development process, which is the area where PNG2Spice aims to expand its features into.

4 Progress of work

Incorporating the previously discussed context of automating the recreation of analog circuit reference designs, the development of an advanced tool such as PNG2Spice involves a sequence of sophisticated image processing and machine learning techniques. The tool chain is logically separated into six stages or three sections, visible in Figure 1. Initially, data is obtained from pasting an image from the operating systems clipboard into the PNG2Spice application. This picture is then analysed by the toolchain.

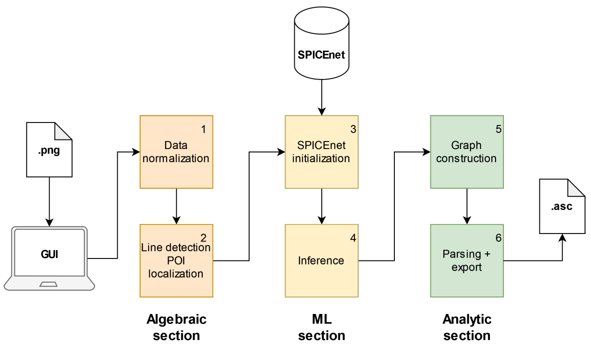

Figure 1: UML diagram for PNG2Spice

- **Stage 1: Data normalization**. The pasted image is padded along the edges to avoid component positioning too close to the borders of the image. This is relevant for the subdivison of the picture later on. Additionally, a binary threshold contrast is applied, which separates the schematic's lines and symbols from the background. The image is now black and white.
- **Stage 2: Line detection and POI localization**. Using the Hough Transform, straight lines can be extracted from the image. Since the lines, the image size and the components form a relationship which indicates how "zoomed-in" the image is, a valuable metric, called a "scaling factor", can be obtained. While this stage is mainly concerned with line detection, an OCR-algorithm is applied to obtain a histogram of letter sizes, demonstrated in Figure 2. Under the assumption that most letters in a schematic share the same size and that a letter forms a fixed size ratio with a component, a scaling factor can be obtained, which is then subsequently used to fine-tune other algorithms later in the process. The ends of detected lines, from here on called "terminals", are then registered as possible POIs, inheriting the terminals from the adjacent lines. Detected lines are demonstrated in Figure 3. The POI locations are then extracted as square images which now correspond to the input data the neural net SPICEnet expects, see stage 3 and 4.

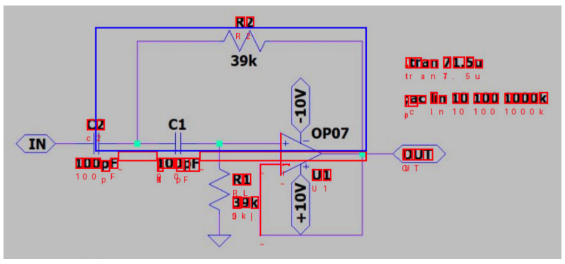

Figure 2: Letter sizes from OCR algorithm. Red: Similar sizes. Blue: Outliers.

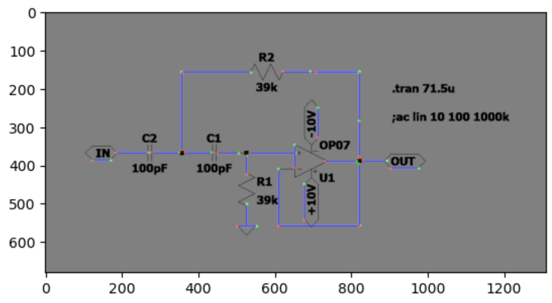

Figure 3: Detected lines with terminals (green, orange)

- **Stage 3: SPICEnet initialization**. After the algebraic section, contemporary machine learning methods are employed. The accompanying convolutional neural network SPICEnet based on a transfer-learning approach using VGG16, is used to classify the contents of the square POI images provided by stage 2. SPICEnet is located in a separate repository and its training is decoupled from PNG2Spice. It was trained using a pure augmentation approach, by which only a handful of source images of the supported components *resistor*, *capacitor*, *inductor*, *diode* and *ground* were used to generate a dataset. For the isolated evaluation of SPICEnet, see Section 6.1.

- **Stage 4: Inference**. With the initialized network, inference is now performed on every extracted POI image. For this, a specific folder structure is used, which is typical for applications using the *Keras* framework for *Tensorflow*. In addition to SPICEnet, another OCR algorithm is now applied to all images to obtain characters, which may yield part descriptions. These are used for debugging and the high level data structures and can further indicate the type of component, but are not considered critical. Results of the inference and OCR are stored as run-time variables and given to processes in stage 5.
- **Stage 5: Graph construction**. A virtual representation of the obtained data and its connections, called a graph, is generated from the POI positions of stage 2, the classifications and the OCR of stage 4. The graph gets assembled by iterating through the data in a symmetrical matrix shape, which causes every POI to be compared to every other POI. Here, terminals are linked and duplicates removed. This linked dataset is then utulized for a rotational analysis of the POIs. This stage relies heavily on manually fine-tuned parameters, which is an issue closer described in Section 6.2. Additionally, coordinates are aligned in such a way that the schematic fits the minimal grid size specified by the LTspice graphical user interface.
- **Stage 6: Parsing and export**. After sufficient processing of the graph, it is handed to a parser for the generation of LTspice syntax. While the format is open and readable, the documentation regarding the relationship between coordinates and component anchor points within the LTspice ecosystem is insufficient and had to be traced manually. After converting every point in the graph to a line of text, a file in *.asc* format can be exported. At this point, the process is finished and the file can be found in a LTspice readable format in the output folder that the user specified in the beginning.

5 Interface documentation

The development of PNG2Spice is aimed at facilitating the workflow of analog circuit designers who seek to expedite their prototyping process. Analog circuit design typically commences with a reference or an existing schematic, which is frequently provided in a format that is visually comprehensive but not directly compatible with electronic design automation (EDA) software. Such references may originate from datasheets or be contributions from individuals within forums who have undertaken exploratory work on electronic engineering concepts. PNG2Spice empowers users to capture screenshots of these schematics using any operating system and selection tool, addressing a common challenge in the analog circuit design domain. Traditionally, designers are required to manually recreate these schematics in their preferred EDA simulation software, a process that is time-consuming and prone to inaccuracies. LTspice emerges as a popular choice among professionals for this task. The integration of PNG2Spice into this workflow offers a significant advancement by enabling the direct insertion of schematic screenshots into the application's graphical user interface (GUI). Subsequently, the tool automatically generates a file compatible with LTspice, thereby streamlining the transition from schematic visualization to simulation readiness. This capability is particularly advantageous in the realm of analog circuit design, where multiple solutions may exist for a single problem, each with its unique set of advantages and disadvantages. These intricacies often necessitate extensive simulation to evaluate the efficacy and performance of different circuit configurations. Thus, PNG2Spice not only simplifies the initial stages of circuit design but also enhances the efficiency of subsequent modifications and evaluations, enabling designers to navigate the complexities of analog circuit optimization with greater ease and precision.

6 Results and evaluation

6.1 SPICEnet

Evaluation of the neural network SPICEnet itself can be done without the context of PNG2Spice. Since the training commenced with the usage of the standard *Tensorflow* and *Keras* APIs, metrics are obtained during and after the training process automatically. These values are listed in Table 1, but should be treated with caution. Since the training data is purely augmented and the base model *VGG16* is already very fine-tuned, overfitting can occur, which leads to very high accuracies, but is nevertheless a weak indicator for the performance in real-world examples. A basic evaluation is performed by using images which were randomly excluded from the training set after augmentation. Another evaluation was done on the original data sources, which were not part of the training process. The results of these evaluations are listed in Table 2.

Number of Images	344,000
Number of augmentation methods	4
Parameters	14,718,792
Train/test ratio	4/1
Epochs	2
Accuracy	99.5%
Loss	1.3%

Table 1: Basic performance metrics for SPICEnet.

Data	**Accuracy**	**Loss**
Augmented ($n = 10$)	100%	0%
Source ($n = 43$)	97.6%	13.6%

Table 2: Additional performance metrics for SPICEnet.

The listed metrics lead us to believe that SPICEnet is almost never the weakest member in the chain. However, classification can still create problems if the detected POIs are not part of the supported classes or the scaling is drastically different from the training set.

6.2 PNG2SPICE

Evaluating the performance of the entire process can only be achieved in a qualitative manner, since evaluation data does not exist. We provide an example and demonstrate advantages and weaknesses of PNG2Spice as a whole. Figure 4 shows a screenshot which is used for demonstrative purposes, while Figure 5 is a screenshot of the detected schematic opened in LTspice. It is visible that all components were detected successfully, and almost all wires were aligned. The wider spacing between the components suggests that the obtained scaling factor was not optimal. It was observed that OCR fails entirely for a few schematics, which causes the scaling factor to fall back to a default value, forming a weakness in the system. Additionally, the polarized capacitor was detected as normal capacitor, which is to be expected, as this special class was not used in training. The orientation of the diodes is incorrect, because the current rotation detection relies on the position of the terminals, which only works for components symmetrical along the rotation axis. The misplacement of the wire leading to the ground-symbol is an error related to the proximity of two POIs. It is here that fine-tuning was necessary, which sadly introduced case-based problems. In future works, these kind of parameters should be calculated by methods of *hyper parameter search* and similar methods.

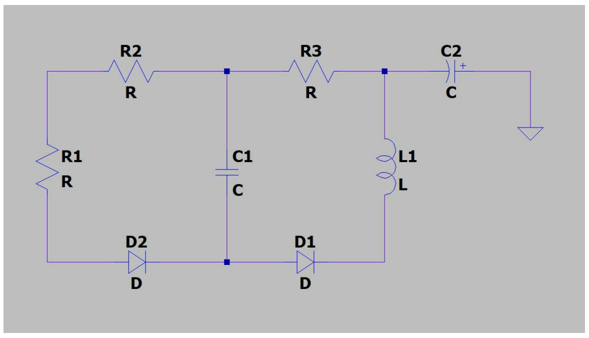

Figure 4: Input screenshot

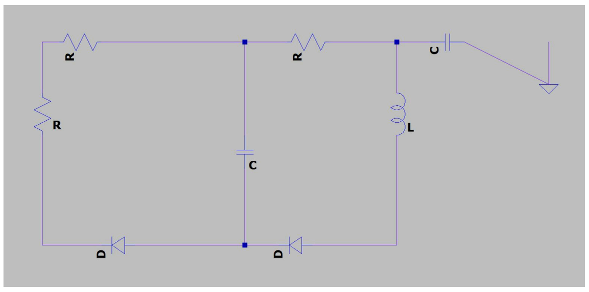

Figure 5: Output screenshot

We observed sadly that the fine-tuning of variables introduces large variances in the quality of the recognition. This suggests that various factors in the calculations are not related to the scaling factor in linear fashion but rather more complex ones.

7 Conclusion

Using analytical and machine learning methods, we were able to show that the issue of reference design conversion in the context of EDA software can be resolved in an applicable manner. By linking computer vision, machine learning and analytical blocks, we were able to reconstruct a virtual and therefore functional representation of electrical schematics from an image. This feat can be an assistive tool for engineers in the electronics sector, as it automates the task of placing parts in the simulation software LTspice according to a digital print of an existing design.

While machine learning is very potent at pattern recognition, we found that electrical schematics involve a multitude of syntactic and semantic representations of information which can't be solved with a single method alone. Since such a needed compilation of methods is hard to explore and to verify, we now understand why a fully-integrated solution to the presented issue is not present on the market. Additional problems, such as the graphical quality of the reference design image, the lack of a global symbol standard or mere styling choices for components, add further complexity to the issue. Future works could involve the usage of widely successful, “multi-talented” LLMs for a process fully utilizing machine learning.

Both SPICEnet and PNG2Spice can be found as separate repositories under the provided links [7], [8].

Bibliography

[1] D. Bailey, A. J. Norman, and G. Moretti, “Electronic Schematic Recognition”, 2005. Accessed: Jan. 27, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Electronic-Schematic-Recognition-Bailey-Norman/5d22b9113739b3b6217aa603f48e5cd3caf6fb64

[2] M. Dey *et al.*, “A Two-Stage CNN-based Hand-Drawn Electrical and Electronic Circuit Component Recognition System”, *Neural Computing and Applications*, vol. 33, no. 20, pp. 13367–13390, Oct. 2021, doi: 10.1007/s00521-021-05964-1.

[3] H. Contigiani, “Hernancontigiani/Electronic-symbol-recognition-system”. Accessed: Jan. 27, 2024. [Online]. Available: https://github.com/hernancontigiani/Electronic-symbol-recognition-system

[4] K. Team, “Keras Documentation: VGG16 and VGG19”. Accessed: Jan. 27, 2024. [Online]. Available: https://keras.io/api/applications/vgg/

[5] R. R. Rachala and M. R. Panicker, “Hand-Drawn Electrical Circuit Recognition Using Object Detection and Node Recognition”, *SN Computer Science*, vol. 3, no. 3, p. 244, May 2022, doi: 10.1007/s42979-022-01159-0.

[6] G. Jocher, “YOLOv5 by Ultralytics”. Accessed: Jan. 27, 2024. [Online]. Available: https://github.com/ultralytics/yolov5

[7] Jakob, “Jake-Is-ESD-protected/SPICEnet”. Accessed: Feb. 19, 2024. [Online]. Available: https://github.com/jake-is-ESD-protected/SPICEnet

[8] “Jake-Is-ESD-protected/Png2spice”. Accessed: Feb. 19, 2024. [Online]. Available: https://github.com/jake-is-ESD-protected/png2spice