Netlistify: Transforming Circuit Schematics into Netlists with Deep Learning

Chun-Yen Huang, Hsuan-I Chen, Hao-Wen Ho
College of Artificial Intelligence
National Yang Ming Chiao Tung University

Pei-Hsin Kang, Mark Po-Hung Lin
Institute of Pioneer Semiconductor Innovation
National Yang Ming Chiao Tung University

Wen-Hao Liu, Haoxing Ren
Nvidia Research

***Abstract*—Analog and mixed-signal (AMS) circuits are commonly represented as schematic images in textbooks, research papers, patents, and other technical documents to facilitate understanding. However, AMS design workflows rely on netlists rather than schematics for tasks such as modeling, simulation, automated circuit sizing, and layout synthesis. Converting schematic images into netlists is a challenging task due to the wide variety of circuit components and the complexity of their connectivity. This paper presents Netlistify, an automated deep learning-based tool that transforms schematic diagrams into netlists through precise component recognition, orientation detection, and connectivity analysis and extraction. Unlike traditional methods that rely on fixed algorithmic rules for connectivity extraction, Netlistify employs deep learning to comprehensively analyze both components and their interconnections, offering a detailed understanding of circuit structures and significantly improving netlist generation accuracy. By addressing the challenges of connectivity complexity with a data-driven approach, Netlistify provides a robust and efficient solution for schematic interpretation. Experimental results demonstrate that Netlistify achieves high accuracy, surpassing state-of-the-art methods and showing great promise for AMS design data collection and seamless integration into modern design automation workflows.**

I. Introduction

The application of machine learning (ML) and large language models (LLMs) in electronic design automation (EDA) for analog and mixed-signal (AMS) design [1]–[12] depends on extensive data collection and processing. However, much of the AMS design data remains confidential and unavailable in the public domain. Instead, AMS design knowledge is primarily documented in technical resources such as textbooks, research papers, and patents, which commonly use schematics to visually represent circuits for clarity and ease of understanding. For EDA workflows, these schematic diagrams must be transformed into netlists, or text-based descriptions of circuit connectivity, that are critical for macro modeling [3], circuit classification [4]–[6], circuit synthesis and simulation [7], [9], parasitic estimation [8], layout synthesis [10]–[12], and other AMS EDA tools. Although the idea of automatic figure extraction and classification from technical documents has been proposed in the literature [13], addressing the challenge of converting visual schematics into machine-readable formats is essential for advancing automation in AMS design. Fig. 1(a) illustrates a schematic image of a voltage follower, sourced from [14]. Automating the transformation of such schematic image into its corresponding netlists, as depicted in Fig. 1(b), is highly desirable for efficient data collection and processing in the application of ML and LLMs for AMS EDA.

A. Related Works

The process of transforming a schematic into a netlist, or *schematic interpretation*, involves two primary tasks: *component detection and recognition*, and *connectivity analysis and extraction*. Most previous works have focused on component recognition, leaving the challenge of connectivity extraction unaddressed. Huoming [15] employed the k-nearest neighbors (KNN) classifier based on pixel distribution to identify components, leveraging spatial relationships within the pixel data. While this approach was straightforward, it relied heavily on carefully selected features, making it highly sensitive to variations in image quality and component representation. Moetesum [16] utilized histogram of oriented gradients (HOG) features combined with a support vector machine (SVM) classifier, offering a higher level of abstraction compared to raw pixel-based methods. However, both methods struggled to generalize effectively across diverse datasets, particularly when the visual characteristics of components varied significantly. These early efforts underscored the challenges of hand-crafted feature extraction and paved the way for modern approaches that leverage deep learning to address these limitations and improve robustness.

More recently, Img2Sim [17], [18] utilized YOLOv5 [19] for component detection, combined with the Hough Transform for connectivity analysis. This hybrid approach effectively integrates advanced deep learning techniques for object detection with classical line detection methods, achieving reasonable accuracy in identifying component connections, particularly in structured and noise-free environments. In contrast, AMSNet [20] takes a different approach by leveraging YOLOv8 [21] for component detection and employing a breadth-first search algorithm on black pixels representing interconnecting wires for connectivity analysis. Furthermore, it uses 2D convolution to detect wire intersections, enhancing its ability to resolve complex circuit connections. In addition to [17], [18], modern LLMs, such as ChatGPT [22], excel at describing the content of various images. However, their performance remains suboptimal and unsuitable for the critical task of converting schematics into precise and accurate netlists within AMS design workflows.

We have observed several shortcomings and challenges in state-of-the-art approaches for transforming circuit schematics into netlists, which can be summarized as follows:

- Most existing approaches are limited to recognizing basic device symbols in analog circuits, such as transistors, diodes, capacitors, resistors, inductors, and current/voltage sources. However, AMS circuit schematics often include a wider range of component symbols, such as operational amplifiers, hierarchical functional blocks, transmission gates, and various digital logic gates, which are typically represented as subcircuits in a SPICE netlist.
- Device symbols in circuit schematics can appear in multiple orientations. Existing approaches may fail to accurately identify these orientations, leading to errors in component detection and connectivity analysis, and hence degrading overall

979-8-3315-3762-3/25/$31.00 ©2025 IEEE

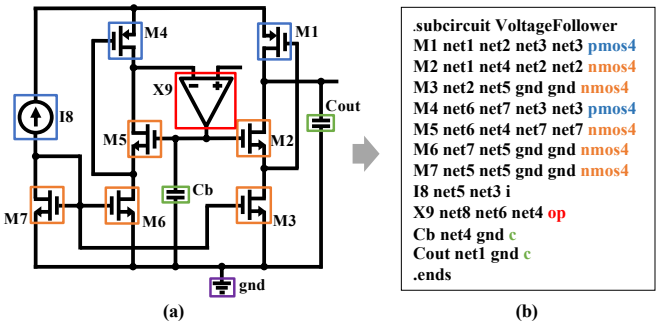

Fig. 1. (a) A schematic image of a voltage follower. (b) The corresponding SPICE netlist of the voltage follower in (a).

performance.

- For connectivity analysis and extraction, real-world schematic images often come from diverse sources, leading to variations in resolution. High-resolution images allow fixed-size 2D convolution to detect intersections effectively, but lower-resolution images with thin wire segments and sparse pixel density often result in inaccuracies. These inconsistencies undermine robustness, making reliable detection across diverse datasets challenging.

B. Our Contributions

To address the aforementioned shortcomings, this paper proposes a deep learning-based methodology to tackle the key challenges in schematic interpretation, including component detection and recognition, orientation determination, and connectivity analysis and extraction. Unlike previous approaches, our method leverages data-driven learning to adapt to schematic variability, minimizing manual intervention while improving the efficiency and reliability of netlist generation. The new contributions of our method are summarized in the following:

- Our method extends beyond basic device symbols in analog circuits to detect and recognize a broader range of schematic symbols in AMS circuits, including operational amplifiers, hierarchical functional blocks, and digital logic gates. This expanded capability allows our approach to accommodate a wider variety of circuit types, significantly enhancing the versatility and applicability of netlist generation.
- Unlike previous works, we decouple orientation determination from component detection by employing separate ML models, YOLOv8 [21] for component detection and ResNet [23] for orientation determination, enabling each model to specialize in its respective task. This separation significantly improves component detection accuracy, as demonstrated by our experimental results.
- We extend a transformer-based model for connectivity analysis and extraction. To handle varying resolutions and different schematic image sizes, we adopt a window-based approach for processing wire segments, which mitigates the risk of gradient explosion and ensures stable training.
- In the most recent work, AMSNet [20], the dataset is primarily derived from textbooks, posing a significant challenge in obtaining sufficient training data. To overcome this, we generate synthetic schematics using a commercial tool, exporting schematic images to build an initial training dataset. Our transformer model, which requires large datasets for connectivity analysis and extraction, is first trained on these synthetic schematics and then fine-tuned on textbook-derived data to enhance performance.
- In addition to developing an extensive dataset for our work, we also release these synthetic schematics as open source after the paper is published. This initiative aims to contribute to the research community and drive further advancements in schematic interpretation.

The rest of this paper is organized as follows: Section II presents the problem formulation. Section III introduces the proposed deep-learning methodology. Section IV demonstrates our experimental results, and finally Section V concludes this paper.

II. Problem Formulation

Given the schematic image of an AMS circuit, which includes various circuit components in different orientations and their corresponding interconnections, as shown in Fig 1(a), the task of **schematic interpretation** is to accurately generate the corresponding netlist, typically in SPICE format, as depicted in Fig 1(b). The types of components and their possible orientations are detailed in Table I and Fig. 3, respectively.

III. The Proposed Deep-Learning-Based Methodology

We present **Netlistify**, a deep learning-based tool for schematic interpretation, designed to transform the schematic image of an AMS circuit into its corresponding netlist, as illustrated in Fig 2. Our methodology comprises four key stages: *Component Detection* (Fig 2(a)), *Orientation Determination* (Fig 2(b)), *Connectivity Analysis* (Fig 2(c)), and *Connectivity Extraction* (Fig. 2(d)).

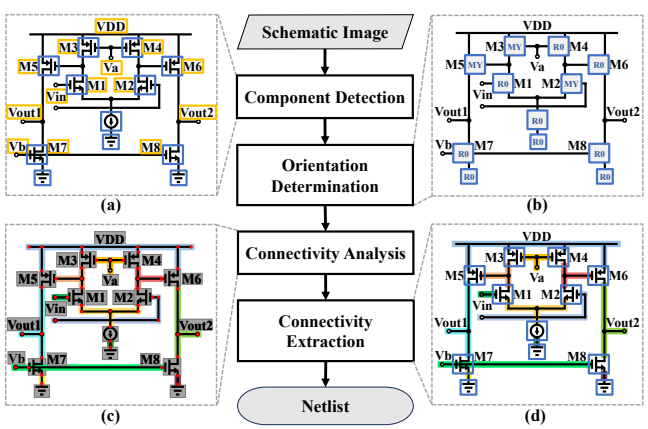

Fig. 2. The proposed deep learning-based flow for schematic interpretation consists of the following stages: (a) component detection, (b) orientation determination, (c) connectivity analysis, and (d) connectivity extraction.

A. Component Detection

Similar to AMSNet [20], Netlistify utilizes YOLOv8 [21] for component detection within a schematic image. YOLO is a state-of-the-art object detection model renowned for its ability to identify and localize objects in images or videos through a single, unified process, providing an excellent balance of speed and accuracy. Unlike traditional multi-stage detectors, YOLO processes the entire image in a single pass, directly predicting bounding boxes and class probabilities using a neural network. This efficiency and precision make it particularly well-suited for schematic interpretation applications.

A key distinction between AMSNet [20] and our approach, Netlistify, lies in how orientation determination is handled. Unlike

Fig. 3. The eight possible orientations of a component symbol.

AMSNet [20], Netlistify does not determine component orientation during the component detection stage but treats each component type with all possible orientations as a single category. In contrast, AMSNet [20] assigns a label to each component bounding box that includes both the component category and orientation, using YOLO to simultaneously detect component types and their orientations. This results in a significantly larger number of categories, as AMSNet [20] must differentiate between various orientations of the same component type. However, this approach faces challenges, as the visual features of components with different orientations are often highly similar, particularly in complex circuit schematics. This similarity makes it difficult for a single model to accurately perform both detection and orientation tasks simultaneously, increasing the risk of misclassification.

B. Orientation Recognition

After detecting each component and identifying its type in the schematic image, we crop the detected components and use ResNet [23] to determine their orientations. Fig. 3 shows all possible orientations of a component symbol. ResNet is particularly well-suited for orientation determination due to its ability to learn robust, hierarchical feature representations. Its residual connections enable efficient training of deep networks without the vanishing gradient problem, allowing it to capture subtle differences critical for distinguishing orientations. Additionally, ResNet's capacity to identify spatial hierarchies and fine-grained visual distinctions ensures strong generalization across similar classes, making it an ideal choice for accurately determining component orientations.

By decoupling orientation determination from component detection, each model optimized for its specific role. YOLO is dedicated to accurately detecting the presence and location of components, while ResNet focuses on determining their precise orientations. This division of tasks simplifies feature extraction for each model, reduces ambiguity caused by visually similar component orientations, and significantly enhances detection accuracy by minimizing the risk of misclassification. Furthermore, Netlistify not only enhances detection accuracy but also introduces adaptability, which is capable of handling various types of components and schematics. The modular design of Netlistify facilitates the seamless integration of new models or techniques, making it easy to expand the range of supported components or further improve accuracy in future developments.

C. Connectivity Analysis

After completing component detection and orientation determination, the next step is connectivity analysis, a crucial process for accurately extracting the relationships between the detected components and identifying the physical interconnections represented as wires. This step ensures that the extracted connections faithfully reflect the actual circuit design depicted in the schematic image. Inspired by [24], we utilize a Transformer-based model [25] for this task, leveraging its ability to detect wire segments and analyze their spatial relationships. The Transformer, a deep learning architecture built on self-attention mechanisms, is particularly well-suited for tasks requiring both local and global context understanding. Unlike traditional neural networks, such as CNNs, which primarily focus on local features, the Transformer excels at modeling long-range dependencies, making it ideal for connectivity mapping in schematic interpretation.

In circuit schematics, accurately identifying whether intersecting wires belong to the same net or represent different connections is a complex task. This challenge is further intensified by the presence of components and text labels, which must be carefully distinguished from wire segments to avoid errors in netlist generation. The Transformer, with its ability to capture both local details and global contexts, enables precise discrimination between intersecting wires and effective differentiation between connections and non-wire elements, such as components and labels. To address these challenges, we propose to divide the schematic image into smaller cells. This segmentation strategy simplifies the schematic complexity, allowing the model to process localized, less intricate regions, thereby stabilizing the training process. Although this approach may result in nets being split across multiple cells, it preserves the distribution of net features and mitigates the risk of gradient explosions. This ensures stable training and enhances the model's ability to learn connectivity features effectively. By focusing on smaller, localized regions, the model is better equipped to handle the variability present in complete schematic images, improving accuracy and reliability.

Figure 4 illustrates the proposed Transformer model for connectivity analysis in schematic images. The inputs to the model are image cells of a schematic, where detected components are masked. Each segmented cell is processed through a ResNet [23] to extract feature maps that capture the essential visual characteristics of the schematic content. These feature maps are then augmented with position encoding, which embeds spatial information into the feature representation. Position encoding allows the Transformer to understand the relative locations of elements within the images, providing crucial context for interpreting spatial relationships. The augmented feature maps are subsequently fed into the Transformer encoder, which generates a rich representation of the input by capturing both local and global relationships within the schematic. The encoder consists of multiple layers, each comprising self-attention mechanisms and feed-forward networks. These components work together to model the contextual dependencies between different parts of the image effectively. After encoding, the processed representations are passed to the Transformer decoder. The decoder begins with a set of initial learnable queries, which represent potential wire segments in the schematic image. It attends to both these queries and the encoded feature maps, interpreting the information to predict the connectivity of wire segments. By combining the context provided by the encoder with the initialized queries, the decoder identifies the positions of wire segments within the schematic. The final stage involves passing the decoder's output through a Feed-Forward Network (FFN), which consists of fully connected layers that refine the predictions. In the proposed model, the FFN predicts the exact start and end coordinates of each wire segment. By transforming the rich feature representations from the decoder into precise numerical values, the FFN provides the final output required for schematic interpretation. This detailed process enables accurate connectivity analysis, ensuring the generated predictions faithfully represent the schematic design.

Dividing the schematic image into smaller cells, while simplifying analysis, introduces additional challenges. Wires and components may be inadvertently split during segmentation, potentially

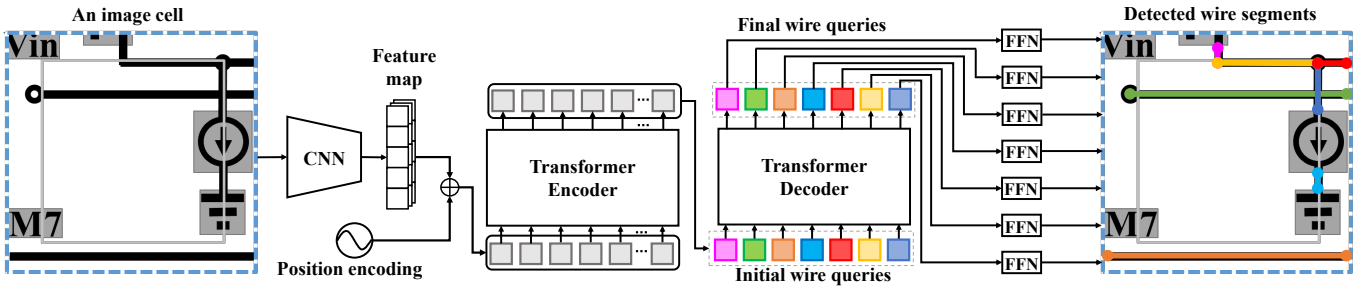

Fig. 4. The proposed Transformer model for connectivity analysis in schematic images. The schematic image is divided into cells, which serve as inputs to the Transformer, enabling the model to detect wire segments within each cell.

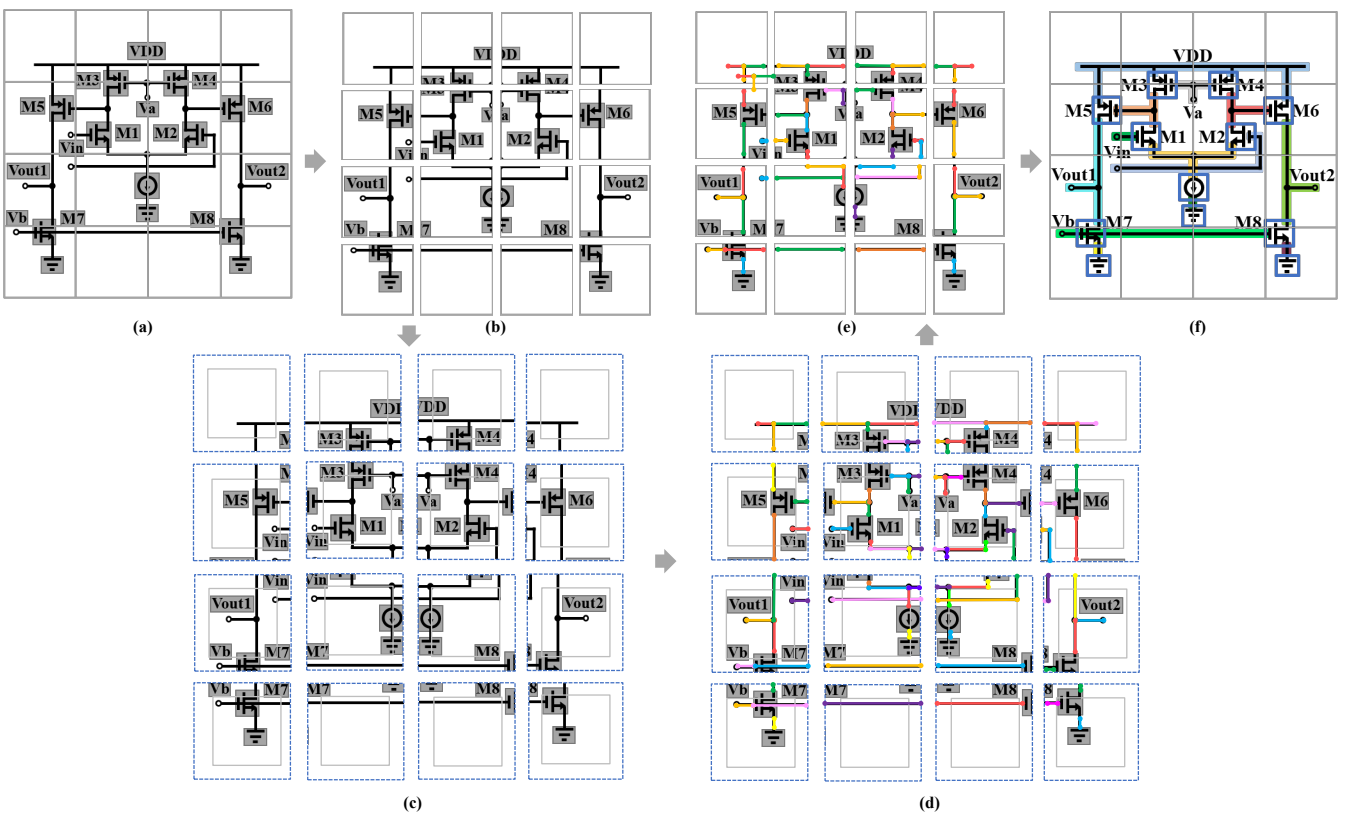

Fig. 5. An example illustrating the process of connectivity analysis and extraction: (a) A schematic image divided into cells with detected components and texts masked. (b) Segmented image cells. (c) Expanded image cells to include features near cell boundaries. (d) Wire segments detected by the Transformer model. (e) Shrinking of image cells to their original size. (f) Connectivity extraction based on detected components and wire segments.

causing errors in model interpretation. For instance, components or text labels might be misclassified as wire segments, degrading the model's performance. To address this, we implement a masking mechanism that applies masks to the predicted locations of components and text, identified during the component detection stage. This ensures that even if components or text are partially cut, their identity and role within the schematic are preserved.

Furthermore, we expand the image cells to include a broader context around their boundaries, creating overlapping regions between adjacent cells. These overlapping regions provide the model with additional information about neighboring areas, enabling it to better interpret connectivity at the cell boundaries. This approach ensures that the model has a more comprehensive understanding of net connectivity, even at the edges of individual cells. By incorporating masking and overlapping regions, the proposed methodology not only improves the accuracy of connectivity analysis but also enhances the reliability of netlist generation. Fig. 5 gives an example illustrating the whole process of connectivity analysis and extraction based on the proposed approach.

During model training, the optimal assignment between the ground truth and predicted wire segments must be determined. To achieve this, the Hungarian algorithm is used to find the best match between predicted and ground truth wire segments by minimizing the overall assignment cost. This process ensures that each predicted wire segment is paired with the most appropriate ground truth wire segment, which is essential for accurate error calculation in subsequent steps. The optimal assignment between the ground truth and predicted wire segments resulting from Hungarian algorithm is denoted as:

$$\text{Hungarian}(\hat{\mathbf{w}}, \mathbf{w}) \rightarrow \text{matched wire segment pairs}, \quad (1)$$

where $\hat{\mathbf{w}} = (\hat{w_1}, \hat{w_2}, \hat{w_3}, \ldots, \hat{w_m})$ and $\mathbf{w} = (w_1, w_2, w_3, \ldots, w_n)$ corresponding to the predicted and ground truth sequence of wire segments, respectively. Each sequence may contain different numbers of wire segments, denoted by m and n, respectively.

We define the localization loss, $\mathcal{L}_{\text{loc}}$, to measure how well the predicted coordinates match the ground truth for each matched wire segment pair. If $\mathbf{p}_i = (x_i^{\text{start}}, y_i^{\text{start}}, x_i^{\text{end}}, y_i^{\text{end}})$ represents the predicted coordinates and $\mathbf{g}_i = (x_i^{\text{gt-start}}, y_i^{\text{gt-start}}, x_i^{\text{gt-end}}, y_i^{\text{gt-end}})$ the ground truth coordinates for $\hat{w_i}$ and w_i, respectively, then:

$$\mathcal{L}_{\text{loc}} = \frac{1}{N} \sum_{i=1}^{N} \text{Smooth}_{L1}(\mathbf{p}i, \mathbf{g}i), \tag{2}$$

where N is the number of matched wire segments, and

$$\text{Smooth}_{L1}(z) = \begin{cases} 0.5z^2 & \text{if } |z| < 1, \\ |z| - 0.5 & \text{otherwise.} \end{cases} \tag{3}$$

D. Connectivity Extraction

Finally, using the detected component types, orientations, and wire segments, along with the physical locations of all components and interconnecting wires, Netlistify extracts the logical connections from the schematic image and reconstruct the corresponding netlist, which is the text-based representation of an AMS circuit that accurately reflects the original schematic design.

IV. Experimental Results

We conducted our experiments on two datasets. The first dataset was synthetically generated by creating random SPICE netlists for various analog and mixed-signal circuits, generating schematic diagrams using Custom Compiler [31], and exporting the corresponding schematic images. The second dataset comprises schematic images of analog circuits captured from textbooks [26]–[30]. Tables I and II provide detailed information about each dataset, including the number of schematic images and the distribution of component types. Both datasets were divided into training, validation, and test subsets. Table III further summarizes the test dataset, classified into three categories: (1) Real Analog Circuits from textbooks, (2) Synthetic Analog Circuits, and (3) Synthetic Mixed-Signal Circuits.

In our experimental evaluation, we employed distinct metrics to comprehensively assess the performance of our proposed method in three key areas: component detection, connectivity analysis, and component orientation determination. The evaluation was divided into three parts: Component Metrics, Connectivity Metrics, and Component Orientation Metrics. We also conducted experimental comparisons between our method, Netlistify, and state-of-the-art approaches, including GPT-4o [22] and AMSNet [20].

For component evaluation, we used metrics derived from the results of the component detection process. Specifically, we calculated accuracy, precision, recall, and F1 score to evaluate the model's performance in identifying components within schematic images. Accuracy measured the overall correctness of component detection, while precision, recall, and F1 score provided a more detailed assessment of the model's ability to detect and classify components accurately. In addition to component detection, we assessed the accuracy of the connections formed between components, referred to as connectivity metrics. For this evaluation, we defined connectivity pairs based on the generated SPICE netlist. A connectivity pair captures the logical relationship between specific component pins, representing how components are interconnected within the circuit. These pairs derived from the pin connections associated with each component are formally defined by the following equation:

$$P_c = \{(p_i, p_j) \mid p_i \in T(C_a), p_j \in T(C_b), \text{net}(p_i) = \text{net}(p_j)\}, \tag{4}$$

where $T(C_a)$ and $T(C_b)$ represent the sets of pins associated with components C_a and C_b of specific types, and $\text{net}(p)$ denotes the net to which pin p is connected. Using this representation, we computed the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) by comparing the predicted connectivity pairs with those from the ground truth netlist. From these values, we derived accuracy, precision, recall, and F1 score to quantitatively evaluate the quality of the model's connectivity predictions. The component orientation metrics include accuracy, precision, recall, and F1 score, which collectively evaluate the model's ability to correctly recognize the orientation of components within schematic images. Similarly, we define TP, TN, FP, and FN specifically for component orientation. If a component's orientation is incorrectly predicted, both FP and FN are incremented by one: the incorrect prediction is counted as a false positive for the predicted orientation and as a false negative for the true orientation. This approach provides a detailed and fair assessment of the model's performance in determining the precise rotational states of components.

Our experimental results are summarized in Tables IV and V. Notably, GPT-4o [22] underperformed in generating netlists from schematic images, failing to surpass both AMSNet [20] and our proposed method in terms of component and connectivity metrics. For component metrics, AMSNet [20] employs YOLO [21] to simultaneously recognize component types and orientations. This approach resulted in reduced accuracy, recall, and F1 scores for components, particularly on the "Real Analog Circuits" and "Synthetic Analog Circuits" datasets, thereby compromising the overall integrity of the generated netlists. AMSNet was not tested on the "Mixed-Signal Circuits" dataset because it does not support the broader range of component types required for AMS design.

Focusing on the connectivity metrics, our approach achieved superior performance compared to AMSNet [20] in all connectivity metrics: accuracy, precision, recall, and F1 score were improved by 2.4%, 0.4%, 20.7%, and 12.4%, respectively. These improvements demonstrate the enhanced generalizability of our model, as further evidenced by our better results on the "Synthetic Analog Circuits" test dataset. On the "Synthetic Mixed-Signal Circuits" test dataset, our method also outperformed GPT-4o [22] significantly, which highlights the inadequacy of large language models in addressing this problem effectively.

The component orientation metrics presented in Table V demonstrate that our method outperforms AMSNet across both datasets. Specifically, our approach achieves higher accuracy, precision, recall, and F1 scores. This improvement is largely attributed to the decoupled design of our detection pipeline, which separates component detection from orientation recognition. By handling these tasks independently, specialized models can optimize each task more effectively. This separation minimizes confusion caused by visually similar components with different orientations, a challenge that AMSNet faces due to its simultaneous handling of detection and orientation classification.

In summary, the experimental results demonstrate that our proposed approach, Netlistify, achieves superior accuracy in all aspects when transforming schematic images into netlists, outperforming state-of-the-art methods. It shows great potential for AMS design

TABLE I
DATASET OF SYNTHETIC AMS SCHEMATIC IMAGES FEATURING VARIOUS TYPES OF AMS COMPONENTS.

Dataset	Image	GND	PMOS	NMOS	PNP	NPN	Resistor	Capacitor	Inductor
Synthetic AMS Circuits	40000	134002	15768	15902	15739	15770	15736	15721	15883

Dataset	Diode	AND	OR	XOR	INV	FUNC	OP	TG
Synthetic AMS Circuits	15621	63339	47740	15995	15718	15684	15854	15956

TABLE II
DATASET OF ANALOG SCHEMATIC IMAGES EXTRACTED FROM FIVE TEXTBOOKS, FEATURING ONLY BASIC ANALOG DEVICE COMPONENTS.

Textbook	Image	GND	PMOS	NMOS	PNP	NPN	Resistor	Capacitor	Voltage source	Current source
[26]	13	26	18	21	2	5	12	6	0	11
[27]	37	57	3	22	3	64	80	12	27	25
[28]	44	67	105	122	0	13	35	7	8	30
[29]	51	60	184	206	2	4	27	39	2	24
[30]	214	392	252	443	9	0	273	77	16	68
Total	359	602	562	814	16	86	427	141	53	158

TABLE III
TEST DATASET

Test Dataset	Image	GND	PMOS	NMOS	PNP	NPN	Resistor	Capacitor	Inductor	Voltage source	Current source
Real Analog Circuits	61	93	71	109	5	19	79	22	0	16	25
Synthetic Analog Circuits	800	1562	1191	1199	0	0	1183	1178	1176	0	0
Synthetic Mixed-Signal Circuits	1500	5002	587	598	605	572	553	609	622	0	0

Test Dataset	Diode	AND	OR	XOR	INV	FUNC	OP	TG
Synthetic Mixed-Signal Circuits	618	2382	1155	599	593	653	635	596

TABLE IV
EXPERIMENTAL RESULTS ON THREE TEST DATASETS: REAL ANALOG CIRCUITS [26]–[30], SYNTHETIC ANALOG CIRCUITS, AND SYNTHETIC MIXED-SIGNAL CIRCUITS. COMPARISON OF COMPONENT DETECTION ACCURACY, CONNECTIVITY ACCURACY, PRECISION, RECALL, AND F1 SCORE ACROSS THREE DIFFERENT METHODS: GPT-4O [22], AMSNET [20], AND OUR PROPOSED APPROACH.

Dataset	Method	Component Metrics (%)				Connectivity Metrics (%)			
		Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Real Analog Circuits [26]–[30]	GPT-4o [22]	90.3	98.7	91.3	94.1	86.9	56.2	34.1	40.0
	AMSNet [20]	99.1	100.0	99.1	99.5	96.2	95.9	70.4	81.2
	Netlistify [This Work]	**99.7**	**100.0**	**99.7**	**99.7**	**98.6**	**96.3**	**91.1**	**93.6**
Synthetic Analog Circuits	GPT-4o [22]	77.4	94.1	90.8	86.3	83.4	51.1	26.4	32.7
	AMSNet [20]	96.2	100.0	96.2	98.1	86.5	72.3	29.5	41.9
	Netlistify [This Work]	**100.0**	**100.0**	**100.0**	**100.0**	**91.2**	**80.1**	**81.0**	**77.2**
Synthetic Mixed-Signal Circuits	GPT-4o [22]	72.2	88.4	77.1	81.8	80.9	15.8	4.6	6.6
	Netlistify [This Work]	**100.0**	**100.0**	**100.0**	**100.0**	**92.9**	**83.3**	**77.1**	**78.4**

TABLE V
COMPARISON OF COMPONENT ORIENTATION METRICS BETWEEN AMSNET [20] AND OUR PROPOSED METHOD.

Dataset	Method	Component Orientation Metrics (%)			
		Accuracy	Precision	Recall	F1 Score
Real Analog Circuits	AMSNet [20]	93.7	97.4	95.1	96.0
	Netlistify	**99.4**	**99.7**	**99.5**	**99.6**
Synthetic Analog Circuits	AMSNet [20]	97.7	98.9	98.4	98.6
	Netlistify	**100.0**	**100.0**	**100.0**	**100.0**

data collection, enabling the application of ML and LLMs for AMS EDA.

V. CONCLUSIONS

In this work, we presented Netlistify, a novel deep learning-based approach for transforming schematic images into netlists, addressing key challenges in AMS circuit data collection from literature. Our methodology features separating component detection from orientation recognition, which significantly enhances accuracy by leveraging specialized models for each task. Additionally, we proposed a Transformer-based framework for connectivity analysis, effectively capturing both local and global relationships within schematics to ensure precise wire segment detection and connectivity extraction. Experimental evaluations on both synthetic and real-world datasets demonstrated that Netlistify outperforms state-of-the-art methods, achieving superior accuracy in component detection, orientation recognition, and connectivity analysis. Furthermore, our method supports a broader range of component types, making it versatile for AMS design workflows. By enabling reliable schematic interpretation and netlist generation, Netlistify shows great promise for advancing AMS design data collection and facilitating seamless integration into modern electronic design automation (EDA) systems.

ACKNOWLEDGEMENTS

The authors would like to express their sincere gratitude to NVIDIA, Inc., and to the National Science and Technology Council (NSTC) of Taiwan, under Grant No. NSTC 111-2221-E-A49-137-MY3, for their generous support of this research. The authors also wish to thank the creators of AMSNet [20] for their assistance with the comparative study.

REFERENCES

[1] S. Maji, A. F. Budak, S. Poddar, and D. Z. Pan, "Toward end-to-end analog design automation with ML and data-driven approaches (invited paper)," in *Proceedings of the Asia-South Pacific Design Automation Conference*, 2024, pp. 657–664.

[2] S. Poddar, A. Budak, L. Zhao, C.-H. Hsu, S. Maji, K. Zhu, Y. Jia, and D. Z. Pan, "A data-driven analog circuit synthesizer with automatic topology selection and sizing," in *Proceedings of the Design, Automation & Test in Europe*, 2024.

[3] Y. Lin, Y. Li, M. Madhusudan, S. S. Sapatnekar, R. Harjani, and J. Hu, "MMM: Machine learning-based macro-modeling for linear analog ICs and ADC/DACs," in *ACM/IEEE Workshop on Machine Learning for CAD*, 2023.

[4] G.-H. Liou, S.-H. Wang, Y.-Y. Su, and M. P.-H. Lin, "Classifying analog and digital circuits with machine learning techniques toward mixed-signal design automation," in *Proceedings of the IEEE International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design*, 2018, pp. 173–176.

[5] K. Kunal, T. Dhar, M. Madhusudan, J. Poojary, A. K. Sharma, W. Xu, S. M. Burns, J. Hu, R. Harjani, and S. S. Sapatnekar, "GNN-based hierarchical annotation for analog circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 9, pp. 2801–2814, 2023.

[6] H.-Y. Hsu and M. P.-H. Lin, "Automatic analog schematic diagram generation based on building block classification and reinforcement learning," in *Proceedings of the IEEE International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design*, 2022, pp. 43–48.

[7] J. Zhang, J. Bao, Z. Huang, X. Zeng, and Y. Lu, "Automated design of complex analog circuits with multiagent based reinforcement learning," in *Proceedings of the ACM/IEEE Design Automation Conference*, 2023.

[8] B. Shook, P. Bhansali, C. Kashyap, C. Amin, and S. Joshi, "MLParest: Machine learning based parasitic estimation for custom circuit design," in *Proceedings of the ACM/IEEE Design Automation Conference*, 2020.

[9] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolic, "AutoCkt: Deep reinforcement learning of analog circuit designs," in *Proceedings of the Design, Automation & Test in Europe*, 2020, pp. 490–495.

[10] P.-H. Wu, M. P.-H. Lin, T.-C. Chen, C.-F. Yeh, X. Li, and T.-Y. Ho, "A novel analog physical synthesis methodology integrating existent design expertise," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 199–212, 2015.

[11] M. P.-H. Lin, H.-Y. Chi, A. Patyal, Z.-Y. Liu, J.-J. Zhao, C.-N. J. Liu, and H.-M. Chen, "Achieving analog layout integrity through learning and migration invited talk," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 2020.

[12] P.-C. Wang, M. P.-H. Lin, C.-N. J. Liu, and H.-M. Chen, "Layout synthesis of analog primitive cells with variational autoencoder," in *Proceedings of the IEEE International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design*, 2023.

[13] K.-C. Chen, C.-C. Lee, M. P.-H. Lin, Y.-J. Wang, and Y.-T. Chen, "Massive figure extraction and classification in electronic component datasheets for accelerating pcb design preparation," in *ACM/IEEE Workshop on Machine Learning for CAD*, 2021.

[14] M. Hu, J. Jin, Y. Guo, X. Liu, and J. Zhou, "A power-efficient SAR ADC with optimized timing-redistribution asynchronous SAR logic in 40-nm CMOS," *Circuits, Systems, and Signal Processing*, vol. 40, p. 3125–3142, 2021.

[15] Z. Huoming and S. Lixing, "Research on K nearest neighbor identification of hand-drawn circuit diagram," in *Journal of Physics: Conference Series*, vol. 1325, no. 1. IOP Publishing, 2019, p. 012233.

[16] M. Moetesum, S. W. Younus, M. A. Warsi, and I. Siddiqi, "Segmentation and recognition of electronic components in hand-drawn circuit diagrams," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 5, no. 16, pp. e12–e12, 2018.

[17] 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 *Proceedings of the IEEE International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design*. IEEE, 2022.

[18] H. B. Gurbuz, A. Balta, T. Dalyan, Y. D. Gokdel, and E. Afacan, "Img2Sim-V2: A CAD tool for user-independent simulation of circuits in image format," in *Proceedings of the IEEE International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design*, 2023.

[19] G. Jocher, "YOLOv5 by Ultralytics," May 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[20] Z. Tao, Y. Shi, Y. Huo, R. Ye, Z. Li, L. Huang, C. Wu, N. Bai, Z. Yu, T.-J. Lin, and L. He, "AMSNet: Netlist dataset for AMS circuits," in *IEEE LLM Aided Design Workshop*, 2024.

[21] G. Jocher, J. Qiu, and A. Chaurasia, "Ultralytics YOLO," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[22] OpenAI, *GPT-4o API Documentation*, 2024, available online at https://platform.openai.com/docs/gpt-4o

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[24] Y. Xu, W. Xu, D. Cheung, and Z. Tu, "Line segment detection using transformers without edges," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4257–4266.

[25] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[26] D. A. Johns and K. Martin, *Analog integrated circuit design*. John Wiley & Sons, 2008.

[27] D. O. Pederson and K. Mayaram, *Analog integrated circuits for communication: principles, simulation and design*. Springer Science & Business Media, 2007.

[28] P. R. Gray, P. J. Hurst, S. H. Lewis, and R. G. Meyer, *Analysis and design of analog integrated circuits*. John Wiley & Sons, 2009.

[29] P. E. Allen and D. R. Holberg, *CMOS analog circuit design*. Elsevier, 2011.

[30] B. Razavi, *Design of Analog CMOS Integrated Circuits*, 1st ed. McGraw-Hill, 2001.

[31] Synopsys Inc, "Custom Compiler," 2023.

APPENDIX

A. Abstract

This artifact contains the complete setup, codebase, and pre-trained models for replicating the experimental results of Netlistify, a modular deep learning framework designed to convert analog and mixed-signal (AMS) circuit schematics into SPICE/HSPICE-compatible netlists. Netlistify integrates YOLOv8 for component detection, ResNet for orientation classification, and a modified DETR Transformer for precise wire detection and connectivity analysis.

The package includes scripts for preprocessing schematic images, training and evaluating model components, as well as synthetic datasets. With the provided preprocessing scripts, synthetic datasets, and pre-trained models, Netlistify enables accurate, end-to-end schematic interpretation and netlist reconstruction out-of-the-box. All necessary assets to run inference and evaluate the method are publicly available.

B. Description

A.2.1 Check-List (meta-information)

- **Program: Netlistify**
- **Compilation: Python 3.11 with PyTorch 2.7.1**
- **Binary: Python scripts and Jupyter notebooks**
- **Model: YOLOv8 + ResNet + Transformer**
- **Dataset: Synthetic schematics generated via Synopsys Custom Compiler**
- **Run-time environment: Ubuntu 24.04, CUDA 12.2**
- **Hardware: NVIDIA GPU 24+ GB VRAM, 64+ GB RAM**
- **Output: Predicted SPICE netlists**
- **Experiments: See Section C**
- **How much disk space required (approximately)?: 30GB**
- **How much time is needed to complete experiments (approximately): 2 days**
- **Publicly available?: Yes (via GitHub)**

A.2.2 How to access

Our source code are available on Github: https://github.com/NYCU-AI-EDA/Netlistify

A.2.3 Hardware dependencies

- A modern Intel CPU (e.g., Core i5 or above)
- An NVIDIA GPU with a minimum of 24 GB VRAM is recommended for efficient training and inference

A.2.4 Software dependencies

- Linux-based operating system (Ubuntu recommended)
- Python 3.11
- PyTorch 2.7.1

A.2.5 Datasets

Synthetic AMS Circuits: 100,000 images auto-generated with over 15 different component classes (e.g., PMOS, NMOS, OPAMP)

C. Experiment workflow

After successfully cloning and installing the required dependencies from our repository, the complete experiment pipeline can be executed with the following command:

```
$ python main.py
```

As the process runs, progress and intermediate results (e.g., best training and validation checkpoints) are saved to the designated output directory. The exact paths to these outputs are clearly displayed in the terminal, ensuring traceability and facilitating easy access for inspection, evaluation, or debugging purposes.

D. Evaluation and expected results

Once you have finished the training, you can then run:

```
$ python inference.py
```

Some demonstration images (`data/test_images`) can be used to validate the pipeline that generates the correct SPICE netlists. All outputs are given in the folder `results`.

E. Experiment customization

Key parameters (e.g., number of components, model checkpoint paths, image resolution) are configured in `main_config.py`.

F. Methodology

Submission, reviewing and badging methodology:

- https://www.acm.org/publications/policies/artifact-review-and-badging-current
- http://cTuning.org/ae/submission-20201122.html
- https://github.com/ml-eda/artifact-evaluation/