scientific reports

Check for updates

OPEN

JUHCCR-v1: a database for hand-drawn electrical and electronics circuit component recognition

Ayush Roy[1], Saptarshi Pani[1], Samir Malakar[2], Erik Cuevas[3]✉, Marco Pérez-Cisneros[3] & Ram Sarkar[4]

Automatic electrical and electronics component/symbol recognition from hand-drawn circuits is a challenging research problem. However, the literature survey reveals that there has been no significant progress in this domain. One possible reason for this might be a lack of publicly available datasets. To this end, in this work, we have developed a dataset, called JUHCCR-v1, which comprises 20 different hand-drawn circuit components that are commonly found in electrical and electronic circuits. Additionally, we have prepared a synthetic dataset having different variations (like orientations, stroke lengths, and distortions) of collected circuit components that may occur while extracting the components from an entire hand-drawn circuit diagram. This augmented dataset with the original ones would help train the deep learning based circuit component recognition algorithms. In order to provide a base result on this dataset, we have designed a weighted ensemble-based hand-drawn circuit component recognition method applied to snapshots of the convolutional block attention module-aided DenseNet-121 architecture. This benchmarking method achieves an accuracy of 91.15% on test set images. All datasets prepared here, along with codes, are made publicly available for the research community at: https://github.com/AyushRoy2001/Circuit-Component-Analysis.

The basic definition of a circuit is a trail that allows the flow of electrical current. It comprises conducting wires, a power supply (AC or DC), resistance, inductance, diodes, and various other electrical components. The aim of the circuit components of a circuit is a controlled, uninterrupted flow of electricity or power transfer. For a generalized understanding and analysis of electrical circuits, various internationally accepted universal symbols are standardized for all the components used in a circuit. The uniqueness of the symbols for each circuit component makes it easy for their identification and detection. In cases of circuits with a large number of components, manual detection of components for analysis is difficult and a tedious job for humans. To overcome this problem, automatic circuit component detection and recognition are the two primary tasks that can be accomplished with digital image processing and machine learning algorithms[1–3].

Commercially available tools like Computer-Aided Design (CAD), and Circuit Maker are used for drawing circuits in online mode[4,5]. However, these techniques consume much more time than drawing circuits by hand. Thus, a method that can recognize hand-drawn circuits in their imaging form and convert them to machine-encoded forms would serve the requirement in a better way. This would not only save the valuable time of non-expert users of the said commercial tools in learning and mastering, but also would keep the recognition part usable for real-world scenarios for hand-drawn circuits. The manual inspection of large circuits is a time-consuming and tedious job. This can be overcome by automatic circuit component analysis using artificial intelligence[1,3,6].

It is a well-known fact that the fundamental unit of any circuit diagram is circuit components[7]. To interpret a digital circuit diagram, one needs to recognize the components present in a circuit diagram i.e., it is required to identify the components with their symbols. However, the difficulty of identification of hand-drawn circuit components in their image form is much greater than their digital counterparts (see Fig. 1). The typical challenges faced in the case of hand-drawn symbols are extremely varying drawing styles, deformed, non-uniform, incomplete, or imperfectly shaped symbols, changing ink intensity, lower quality of paper, and noise while capturing images[8–12].

Therefore, a number of works[1,6,13,13–18] emphasize circuit component recognition as found in the literature. However, the authors of these works performed their experiments on self-made datasets. Even in most cases, the

[1]Department of Electrical Engineering, Jadavpur University, Kolkata 700032, India. [2]Department of Computer Science, UiT The Arctic University of Norway, Tromsø 9019, Norway. [3]Departamento de Electrónica, Universidad de Guadalajara, Guadalajara C.P-44430, México. [4]Department of Computer Science and Engineering, Jadavpur University, Kolkata 700032, India. ✉email: erik.cuevas@cucei.udg.mx

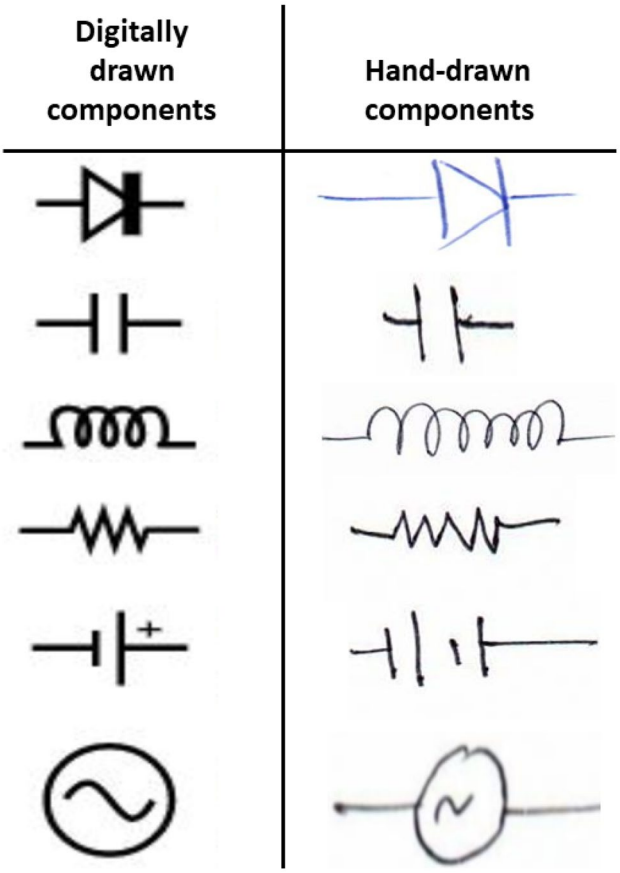


Fig. 1. The difference in complexity level of symbols of hand-drawn and digitally generated circuit components.

authors have worked with only a handful number of images with fewer classes[8,13,15,18]. Here lies the importance of a publicly available hand-drawn circuit component recognition dataset to be used by the research community. This would help researchers to assess any existing and new algorithms. Moreover, the challenges mentioned earlier increase the complexity and difficulty of the problem. All these have highlighted the need for a proper dataset that mimics many real-world challenges that a recognition model must tackle.

This work focuses on designing an image-level dataset consisting of isolated circuit components. The initial dataset is prepared by the current authors by collecting hand-drawn circuit components. The dataset consists of hand-drawn circuit component images having more samples per class and the number of classes compared to the state-of-the-art methods used. The dataset comprises images of 20 circuit components (e.g., Ammeter, Voltmeter, Transformer, Resistance, AC Source, Capacitance, Diodes, Transistors, etc.). This dataset is so prepared that it can include most of the challenges one might face in real-life scenarios. Moreover, this dataset tries to mitigate the problems that might arise due to fewer image samples while training a deep learning framework by including augmented circuit component images. The image augmentation techniques used here help in adding some missing real-life challenges that might occur during the recognition of isolated circuit components in their image forms. Apart from designing a challenging dataset for circuit component recognition, this work also focuses on designing a competent recognition model to provide a benchmark recognition result on the currently prepared datasets. For this, we have used a Convolutional Neural Network (CNN)-aided model empowered with Convolutional Block Attention Module (CBAM) attention mechanism and snapshot ensemble mechanism.

The key **contributions** of this work are as follows:

1. Developed a dataset, dubbed JUHCCR-v1, which comprises 20 commonly appearing components in electrical and electronic circuits.
2. Prepared a synthetic dataset having different variations (like orientations, stroke lengths, distortions, etc.) of circuit components that are quite common while extracted from hand-drawn circuit diagrams. The Synthetic data introduces more complex scenarios that might occur while working on real-life data and it would also help in better training of deep learning models.
3. Designed a snapshot ensemble method applied to a CBAM attention-aided DenseNet-121 architecture for the classification of the hand-drawn circuit components.
4. Benchmarked the results on developed datasets using the proposed method after performing an exhaustive set of experiments.

The remaining part of this article is organized as follows. Section "Related work" describes some previous methods that performed circuit component recognition. The preparation of the dataset that is made publicly available is described in Section "Dataset preparation". The benchmarking method on the present dataset is illustrated in Section "Benchmarking technique" while Section "Results and discussion" describes the results obtained and subsequent discussion. Finally, the article is concluded in Section "Conclusion and future scope".

Related work

Although there are a few works on hand-drawn electrical and electronic circuit analysis, scientists still find it an open research problem due to many challenging factors like the quality of the image, brightness, rotation, non-uniform and deformed shapes, etc. Hence, an analysis of hand-drawn electrical and electronic circuit images on a wide and diverse dataset is highly essential for research purposes. The existing research has considered two different approaches for circuit component recognition, viz., (a) isolated component-based recognition[1,14,19,20] and (b) detection and recognition of the components at one go[21,22]. The methods in the first approach have considered the isolated circuit components[14,19] or extracted the components from an entire circuit diagram through some image processing[23,24] prior to recognize them. In the second approach the circuit components' detection and recognition were performed simultaneously using some object detection[25–27] based deep learning frameworks. It is noteworthy to mention that the present work is designed to support the first approach via an open access dataset. Here, we discuss research attempts of both approaches made by researchers in the past.

Several researchers followed the standard pattern recognition approach to recognize circuit components. As a result, these researchers built their approach as a classification problem where each circuit components were considered as a pattern class and the feature extraction models were designed solely for the isolated circuit components. A number of methods[14,19] considered isolated circuit components to concentrate only on component classification. However, this approach of circuit component recognition approach has been used in the methods for circuit analysis tasks[23,24], and therefore, they used some component extraction techniques prior to the recognition task.

In the work[19], De et al. proposed a technique to recognize components from an electronic circuit diagram using feature point identification followed by a statistically supervised parametric classifier. In another work, Dewangan and Dhole[20] proposed a K-Nearest Neighbors (KNN) based methodology to directly recognize electrical and electronic components using hand-crafted features like geometric area, centroid, eccentricity, convex area, and orientation angle of hand-drawn electrical circuit images. Roy et al.[1] proposed a feature selection-based recognition model where pre-processed electrical and electronic circuit component images were used to extract a feature set consisting of the histogram of oriented gradients (HOG) and several shape-based features. Irrelevant texture-based features were filtered out by the ReliefF algorithm[28,29], and the sequential minimal optimization (SMO) classifier[30] was used for the classification of circuit components. In another work, Dey et al.[14] implemented a two-stage CNN-based model that classifies hand-drawn electrical and electronic circuit components. In the first stage, the circuit components with visual similarity were clustered together into a single unit, and then in the second stage, similarly looking components were further classified into their actual output class.

As stated earlier, a few method exists that use circuit component extraction prior to component recognition for the circuit diagram analysis task. Bailey et al.[23] devised a two-step method for the recognition of electrical circuits. In the first stage, wires were removed from scanned hand-drawn electronic circuits, and then circuit component recognition was performed using a template matching algorithm. In another work, Rabbani et al.[24] used artificial neural networks (ANN) to directly extract electrical circuit components from a hand-drawn circuit using a two-step method. In another work, Dai and Braytont[31] developed a circuit-based convolution operation with dynamic pooling where a deep learning framework was utilized. Feng et al.[32] proposed a system of offline circuit recognition and simulation using digital image processing. The proposed model consists of segmentation, feature extraction, classification, and redrawing and repositioning. Finally, they used this circuit for simulation purposes by substituting values for each component to generate output waveforms and characteristic graphs. Lakshman et al.[33] proposed a hand-drawn electronic circuit diagram recognition model where, firstly, detection is done by constructing the feature vector by combining Local binary pattern (LBP) and statistical features based on pixel density, followed by classification using a support vector machine (SVM) classifier.

Several researchers designed object detection[25–27] based deep learning models for the detection and recognition of circuit components. In these works, the researchers mostly relied on different incremental versions of you only look once (YOLO) models[34], fast region based convolutional neural network (Fast R-CNN)[35], and faster region based convolutional neural network (Faster R-CNN)[36]. For example, Rachala and Panicker[21] proposed an algorithm for the automatic recognition of hand-drawn electronic circuits using YOLO-v5 architecture. Subsequently, they rebuilt the circuit schematic based on object detection and circuit node recognition with high values of precision and accuracy. In another work, Amraee et al.[22] used another technique that analyzes hand-drawn logic circuits with deep neural networks empowered with YOLO object detection and recognition architecture. They have analyzed the connection among the circuit components using a new simple boundary tracking method, followed by the binary function related to the hand-drawn circuit. The authors created a hand-drawn circuit diagram dataset, but the dataset was not made public. In another work, Yang et al.[16] used a YOLO model to segment and recognize power components from substation one-line diagram (SOLD) images, which are like printed images. Bohara et al.[9] used the YOLO-v8 model to detect and recognize the circuit components. In another work, Mathur and Achar[8] compared the performances of YOLO-v5 and Faster R-CNN models for electronic components' orientations in a hand-drawn circuit diagram. AlMughrabi and Hiary[11] used Faster R-CNN for the same purpose. In this connection, Bhutra et al.[17] compared the performance of Faster R-CNN and Fast R-CNN for circuit component detection and recognition and found that Faster R-CNN is better compared to Fast R-CNN.

Apart from these two above mentioned categories, the present work might contribute to other objectives like circuit topology understanding[37], missing connection imputation in circuit[38], and training in the educational system[39,40]. Hu et al.[37] performed parsing of integrated circuit images using a Graph Attention Network (GAT). They followed a bottom-up approach to understand the circuit topology. Circuit recognition processes were also used in the education system[39,40]. Al et al.[40] investigated the importance of augmented reality and machine learning in the study of Electrical engineering, while Loong et al.[39] investigated the usefulness of machine learning in structural analysis in the study of Civil engineering. In the first case, the authors elaborated on the need for electrical circuit recognition, while in the other case, the authors strongly recommended automated digitization of hand-drawn civil engineering drawings. Very recently, Said et al.[38] utilized state-of-the-art graph neural network (GNN) models[41,42] to solve one of the key issues: missing connection imputation in circuit diagram while realizing it. To overcome this problem, the authors came up with a novel two-step solution. First, they formulate missing circuit component identification as a graph classification task in the graph-based representation of a partial circuit, and second, they treat the placement and connectivity of the predicted component as a link completion problem.

Dataset preparation

A suitable dataset is one of the most important prerequisites for evaluating the performance of any method. To the best of our knowledge, the isolated hand-drawn circuit component dataset is missing in the literature to date, although circuit component recognition is a challenging image classification problem and has its needs in the engineering domain. Thus, we have prepared a hand-drawn circuit component dataset and made it public to the research community. This dataset contains hand-drawn samples of 20 analog and/or digital circuit components (see Fig. 2).

Data collection

The hand-drawn components collected here are drawn by different individuals like students, faculty members, and research scholars, who have contributed voluntarily. The circuit component images have been extracted from two different types of sources, namely (i) entire circuit diagram images (see Fig. 3), inspired from[43,44], and (ii) filled-in pre-formatted data-sheets containing isolated circuit components (see Fig. 4), inspired from the works[45,46]. All the documents are scanned using a flat-bedded scanner with a 300 dpi resolution and stored as bitmap (BMP) files. Next, we have employed the circuit component extraction technique proposed by Bhattacharya et al.[47] to extract circuit component images from the circuit diagram images. For the other category of documents, we have used the program of the works[48,49] to extract the circuit component images. In both cases, images are stored in BMP file format.

The circuit component images are drawn using a ball or gel pen with varying ink colors: red, blue, and black. The circuit component pairs like (AND gate and NAND gate), (OR gate and NOR gate), and (PNP Transistor and NPN Transistor), and triads (AC source, Ammeter, and Voltmeter) considered here have an overall similar shape with variations found locally (see Fig. 5a). This figure shows the similarly shaped circuit components horizontally marked with dissimilar portions within red-colored circles. Besides, variations are found in the same component drawn by different individuals (see Fig. 5b), where a circuit component that is drawn differently

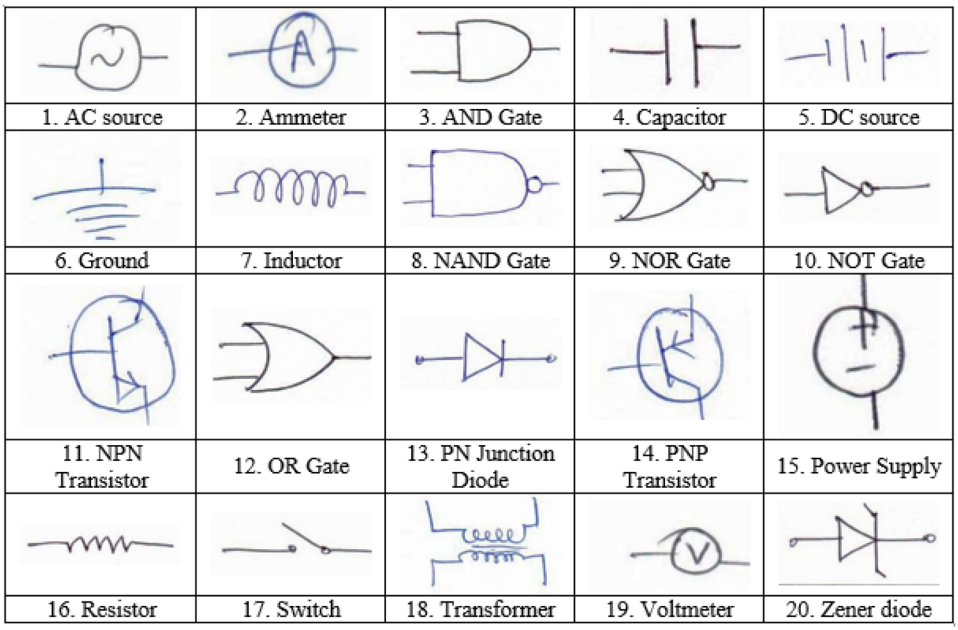
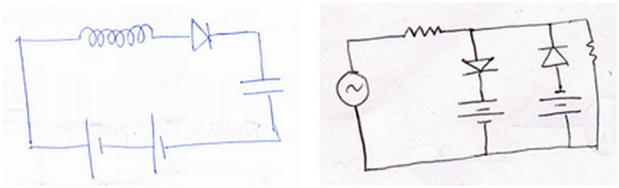


Fig. 2. Sample images of analog and digital circuit components considered here. Numbers preceding the name of circuit components indicate the class number of the corresponding circuit component used in our dataset.



(a) Sample circuit diagram 1 **(b)** Sample circuit diagram 2

Fig. 3. Two sample images representing a complete circuit diagram.

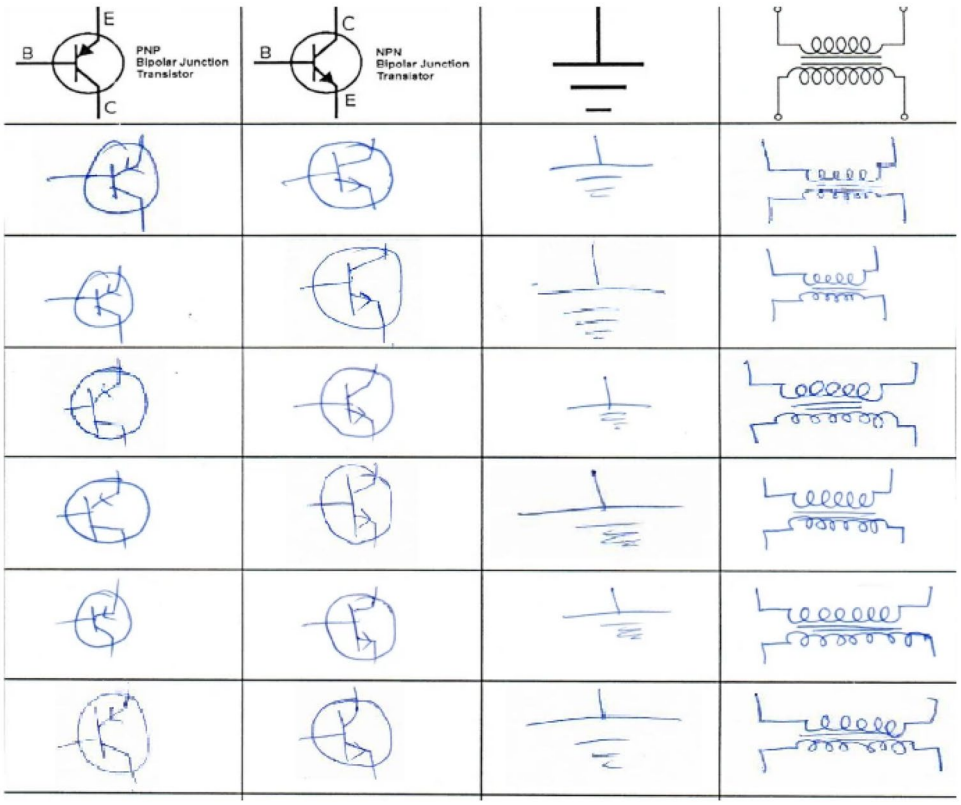
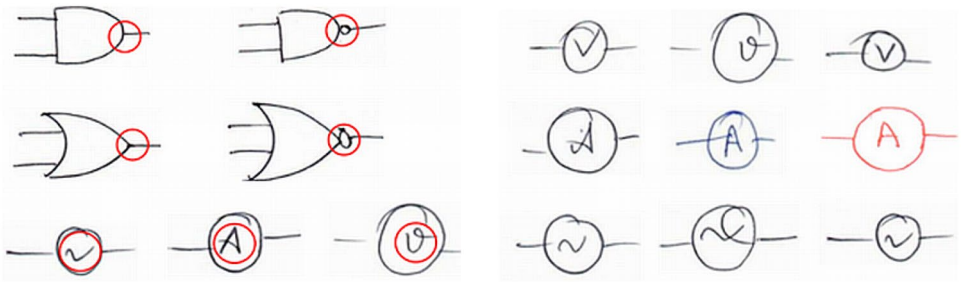


Fig. 4. Filled-in pre-formatted document page used for collecting circuit component samples from individuals.



(a) Inter-class but similarly looked circuit components **(b)** Intra-class but differently drawn circuit components

Fig. 5. Examples of some complex cases found in the present dataset.

is shown horizontally. All these cases make the classification task a challenging one and lead to classification errors[1,9,14]. We name this dataset "JUHCCR-v1.o", where JUHCCR represents "Jadavpur University Hand-drawn Circuit Component Recognition", "v1" represents version 1 of the dataset, and 'o' represents original images. This dataset consists of 150 samples per class, for a total of 3000 samples. The entire dataset is partitioned into two subsets: train and test. The training set consists of 50 sample images per class (i.e., $50 \times 20 = 1000$ in total), while the test set consists of 100 sample images per class (i.e., $100 \times 20 = 2000$ in total). Some samples from this dataset are shown in 1^{st} row of Fig. 6. The images are named as "original_dd", where "dd" denotes the file number $00, 01, \ldots, 99$.

Augmented dataset preparation

To add more variations to samples of the dataset, like orientations, stroke lengths, distortions, etc. that are quite common while circuit components are extracted from real hand-drawn circuit diagrams, we have employed nine different data augmentation processes. These augmentations are applied on each samples of the dataset maintaining the train and test split intact i.e., augmented samples kept in test and train sets. The augmented training dataset consists of 450 sample images per class (i.e., 9000 in total), while the testing dataset consists of 900 sample images per class (i.e., 18000 in total). This dataset consists of 1350 samples per class, for a total of 27000 samples. The augmentations including different variations like orientations, stroke lengths, distortions, etc. of circuit components that are quite common when extracted from hand-drawn circuit diagrams, i.e., augmented dataset is created to incorporate more challenges that may arise while recognizing components extracted from entire diagram. The newly generated augmented circuit component images look very close to the real samples (see Fig. 6), but with increased complexity. Adding augmented image samples also helps in increasing the number of samples per class and thus, this dataset becomes more suitable to train a CNN model. We have named the new dataset as "JUHCCR-v1.a" where the letter 'a' represents augmentation and the rest are as defined earlier. The augmented images are saved as "augmented_aa_bbb", where "aa", and "bbb" represent the augmentation identifier index (see Fig. 6) and file number, respectively. The augmentations used here is described below.

01: **Rotation by an arbitrary angle:** the original images are rotated randomly at an angle θ

Aug. ID	Augmentation Type	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
00	Original					
01	Rotation by an arbitrary angle					
02	Rotation by a negative angle					
03	Rotation by a positive angle					
04	Change in brightness					
05	Change in contrast					
06	Adding Gaussian noise in the background					
07	Adding Gaussian noise in the foreground					
08	Stroke thickening					
09	Adding distortion by suppressing data pixels					

Fig. 6. Illustration of the augmentations applied on the original images.

($\in [-20^\circ, -10^\circ] \cup [10^\circ, 20^\circ]$). Some sample images of this category are shown in 2^{nd} row of Fig. 6.

02: **Rotation by a negative angle:** in this case, the original images are rotated randomly with an angle θ ($\in [-1^\circ, -10^\circ]$) and some sample images are shown in 3^{rd} row of Fig. 6.

03: **Rotation by a positive angle:** here the original images are rotated randomly with an angle θ ($\in [1^\circ, 10^\circ]$). Four Sample images of this category are shown in 4^{th} row of Fig. 6.

04: **Change in brightness:** the brightness of each original image is increased by a factor of 1.5 (see 5^{th} row of Fig. 6).

05: **Change in contrast:** the contrast of each original image is increased by a factor of 1.75 (see 6^{th} row of Fig. 6).

06: **Adding Gaussian noise in the background:** a random amount of Gaussian noise (say, *N*) with mean and standard deviation 0 and 20, respectively, has been added to each original circuit component image (say, *I*). Next, the noise component is then blended with the original image, where a linear blending function (say, *f*()) between 2 functions $f_1()$ and $f_2()$ is formulated using Eq. 1.

$$f(x) = \alpha * f_1(x) + (1 - \alpha) * f_2(x) \tag{1}$$

In Eq. 1, $\alpha \in [0, 1]$ is a constant value. Now, the resultant image (say, *R*) is generated using Eq. 2.

$$R(x, y) = \alpha * I(x, y) + (1 - \alpha) * (N(x, y)) + \gamma \tag{2}$$

In Eq. 2, γ is a constant value. Here we have taken $\alpha = 0.5$ and $\gamma = 30$. Some samples of this category of augmented samples have been shown in 7^{th} row of Fig. 6.

07: **Adding Gaussian noise in the foreground:** a definite amount of noise is injected in the foreground of the original image by using a bilateral filter, which can be formulated using Eq. 3.

$$BF[X]_k = \frac{1}{N_p} \Sigma M_{\sigma_s} ||k - l|| * M_{\sigma_r} |X_k - X_l| * X_l \tag{3}$$

In Eq. 3, $\frac{1}{N_p}$ is normalization factor, Σ is the summation operation over the range of $l \in S$, $M_{\sigma_s}||k - l|| =$ space weight, $M_{\sigma_r}||X_k - X_l|| =$ range weight. In our bilateral filter, we used the value of σ in color space as 75, σ in coordinate space as 75, and the diameter of each pixel neighborhood as 9. Some samples of this category of augmented samples have been shown in 8^{th} row of Fig. 6.

08: **Stroke thickening:** the circuit components' stroke has been increased using this augmentation. For this, in the first step, we employed morphological dilation with a structuring element of dimension 7×7 and called the output image I_{dia}. Next, we have subtracted the image I_{dia} from the original image (say, I_{org}) and stored the resultant image as I_{dif} i.e., $I_{dif} = I_{org} - I_{dia}$. Finally, the augmented image (say, I_{aug}) is generated following the rule mentioned in Eq. 4.

$$I_{aug}(x, y) = \begin{cases} I_{org}(x, y), & \text{if } I_{dif}(x, y) == 0 \\ t(x, y), & \text{otherwise} \end{cases} \tag{4}$$

In Eq. 4, *t*(*x*, *y*) is the mean of the top-t darkest neighbors of the pixels at position (*x*, *y*) in 3×3 neighbors (see 9^{th} row of Fig. 6.).

09: **Adding distortion by suppressing data pixels:** the original image I_o is converted to a binary image I_b . Next, a random integer (say, $i \in [100, 110]$) is generated for each data pixel (i.e., $I_b(x, y) == 0$) of $I_o(x, y)$. Now the augmented image I_a is generated following Eq. 5.

$$I_a(x, y) = \begin{cases} 255, & \text{if } I_o(x, y) < i \text{ and } I_b(x, y) = 0 \\ I_o(x, y), & \text{otherwise} \end{cases} \tag{5}$$

Finally, I_a has been converted to an RGB image (see 10^{th} row in Fig. 6).

Combination of original and augmented datasets

This dataset comprised all samples of the original and augmented components. However, categorical marking (e.g., augmented, original, and augmentation index), as mentioned earlier, is not present in this dataset. We have defined these samples as more competent and applicable for circuit component recognition. This dataset is called "JUHCCR-v1.ao", and it contains 1500 samples per class (500 and 1000 samples per class in training and test sets, respectively). Here, the term "ao" represents the collection of augmented and original samples. The images are called "mixed_ccc", where "ccc" is the shuffled file number $000, 001, \ldots, 999$ for train, and $000, 001, \ldots, 499$ for test samples.

Benchmarking technique

In this work, we have proposed a CNN model for classification of circuit components in order to generate benchmark results on developed datasets. The model is empowered with the CBAM attention technique and snapshot ensemble learning. CBAM is a simple yet effective attention module that enhances the performance of CNNs by assigning more weight to specific channels and spatial locations in the applied feature map. The snapshot ensemble is a technique that generates different trained modules (known as snapshots) while training only a CNN model at different instances (i.e., at different training iterations). We have considered DenseNet-121 as the CNN baseline architecture using the transfer learning protocol. The entire process is shown in Fig. 7.

DenseNet-121 architecture

In[50], the authors have elaborately explained how densely connected CNNs or DenseNets are superior to their traditional and residual counterparts, which have later been used in many applications successfully[51–53]. In traditional CNNs, a convolutional layer gets its input from the immediate predecessor, i.e., for *n* layers, there are *n-1* direct connections. However, in DenseNets, a particular convolutional layer (say, *a*) receives its inputs from each of its preceding layers with respect to *a*, except the first layer which receives the input image (input layer), i.e., in DenseNets, there are $\frac{n*(n+1)}{2}$ connections. In this way, DenseNets are quite effective in dealing with the vanishing gradient problem, which mainly arises as the CNN tends to have deeper layers.

Here, we have used the DenseNet-121 architecture that has the following layers: 1 Convolution layer of kernel dimension 7×7, 58 Convolution layers of kernel dimension 3×3, 61 Convolution layers of 1×1 kernel dimension, 4 AvgPool, and 1 fully connected layer. The n^{th} layer receives the feature maps of all previous layers, x_0,...,x_{n-1}, as inputs, and defined as Eq. 6.

$$x_n = H_n([x_0, x_1, ..., x_{n-1}]) \tag{6}$$

In Eq. 6, $[x_0,x_1,...,x_{n-1}]$ represents the concatenated feature map. Several inputs of H_n are concatenated into a single tensor for ease of implementation. Implementing a concatenation operation is not feasible when the size of feature maps changes. To tackle this, DenseNets are divided into DenseBlocks. Inside each block, the dimensions of the feature maps remain constant, but the number of filters between them is changed. Between the blocks, there are transition layers that reduce the number of channels to half of that of the already existing channels. For each layer, from equation 6, H_n is defined as a composite function that applies three consecutive operations: batch normalization, a rectified linear unit (ReLU), and a convolution. After passing through each dense layer, the size of the feature map increases with each layer, adding *K* features on top of the existing features. This parameter *K* defines the growth rate of the network, which controls the amount of information added in each layer of the network. If each function, H_n produces k feature maps, then

$$K_n = K_0 + (n-1) * k \tag{7}$$

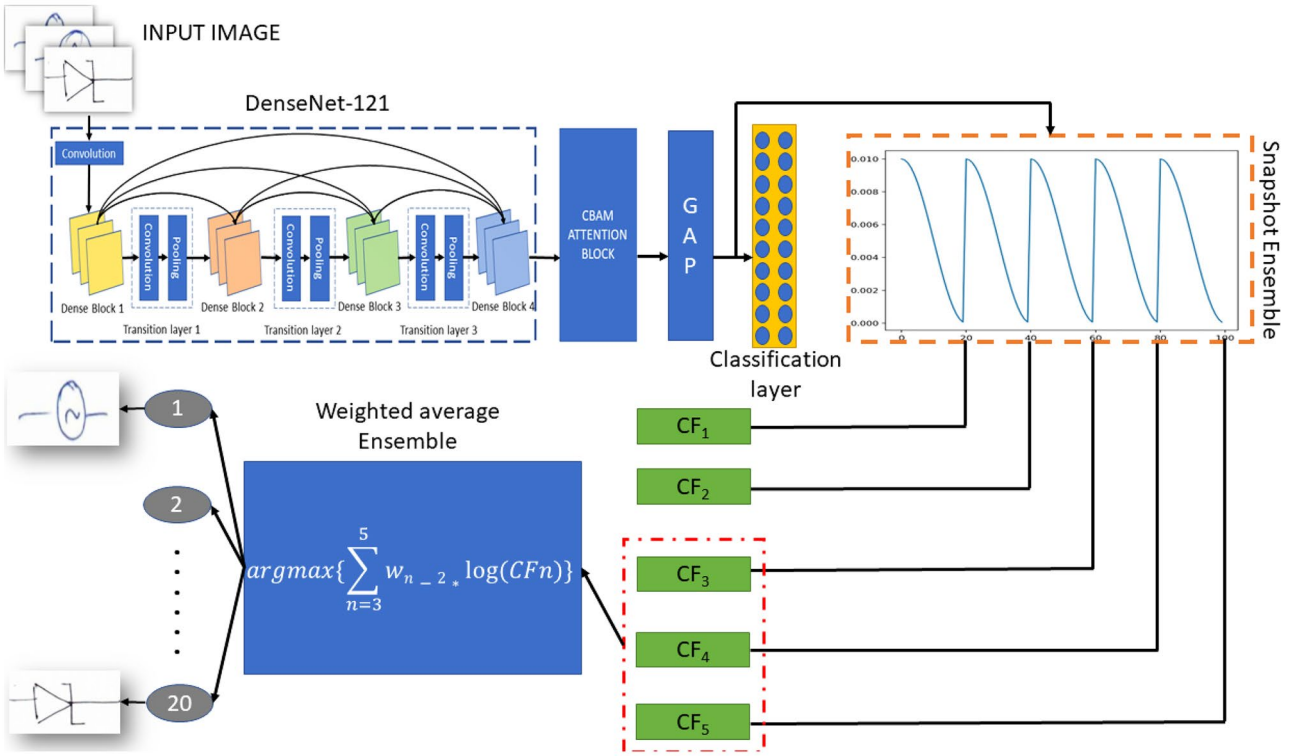


Fig. 7. Block diagram to illustrate the proposed circuit component recognition technique designed to facilitate benchmark results on the present datasets. The feature map from DenseNet-121 is passed on to the CBAM attention mechanism. The feature map received from CBAM is flattened using global average pooling (GAP), which serves as an input for the classification layer. The model is trained using the snapshot ensemble, where five snapshots of the model have been saved. The confidence scores of the top-3 snapshots (say, CF_3, CF_4, and CF_5) undergo a weighted average ensemble technique, where $w_i : i = 1, 2, 3$ is the weight assigned to the confidence score of snapshots to predict the final class label.

In Eq. 7, K_n is the number of input feature maps and K_0 is the number of channels in the input layer. When the number of inputs becomes quite high, a 1×1 convolution layer can be introduced as a bottleneck layer before each 3×3 convolution to reduce the computational burden in DenseNet-121. Hence, DenseNets require fewer parameters than an equivalent traditional CNN, and this allows feature reuse in DenseNets. A pictorial representation of the DenseNet-121 architecture is shown in Fig. 8.

CBAM attention

The CBAM attention mechanism[54] is applied to the last feature map of dimension $C \times H \times W$ generated from any CNN architecture. Here, C, H, and W represent the number of channels, height, and width of the feature map, respectively. The CBAM attention is comprised of a 1D Channel Attention Module (CAM) and a 2D Spatial Attention Module (SAM). The CAM essentially assigns weights to channels of the feature maps, i.e., enhances particular channels that contribute more towards boosting the model's performance. The 1D channel attention network outputs a feature map (say, F_c) of dimension $C \times 1 \times 1$. F_c can be defined using Eq. 8.

$$F_c = \sigma(MLP(GAP(F)) + MLP(GMP(F))) \tag{8}$$

In Eq. 8, '+' denotes the element-wise addition operation, F is the input feature map, and σ represents the sigmoid activation function. Now, $F'_c = F_c \otimes F$ is fed to the SAM ($\otimes$ denotes element-wise matrix multiplication), which is the domain space encapsulation attention mask applied to enhance the feature representation F'_c. It outputs a feature map (say, $F^{''}$) of dimension $C \times H \times W$. $F^{''}$ can be formulated using Eq. 9.

$$F^{''} = f^{7\times7}[DL(GAP(F'_c)); DL(GMP(F'_c))] \tag{9}$$

In Eq. 9, ';' denotes the concatenation of the two features. In Eq. 9, $f^{7\times7}$ is the convolutional layer of kernel size 7×7 with dilation of 4, and DL represents Dense Layers. The dense layers (DL) comprise two dense layers that use the ReLU activation function. The first dense layer takes C dimensional input and outputs $\frac{C}{r}$ (r is the reduction ratio) dimensional vector. This output is fed to the second dense layer, which returns an output feature of dimension C. DL is shared by the global average pooling (GAP) layer and global max pooling (GMP) layer. Thus, F_{CBAM} (see Eq. 10) is the output of the CBAM attention module having dimensions $C \times H \times W$ (see Fig. 9).

$$F_{CBAM} = F^{''} \otimes F'_c \tag{10}$$

Snapshot ensemble

The basic ideology of using the snapshot ensemble is to collect multiple learned models (called snapshots) obtained by training a CNN architecture over several iterations, and these snapshots are combined to make the final decision. By doing so, the computational burden for running different CNN models to form an ensemble is greatly reduced, and the overall performance is improved. In this work, this mechanism is used to create five snapshots while training the base CNN architecture model. We develop the snapshot ensemble in two parts - the first part involves a custom callback to save the model at the bottom of each learning rate schedule, while the second part involves loading the saved models and using them to make an ensemble prediction. Here, we have used the cosine annealing learning rate schedule. We have implemented the cosine annealing schedule as described in the work[55], which is shown in Eq. 11.

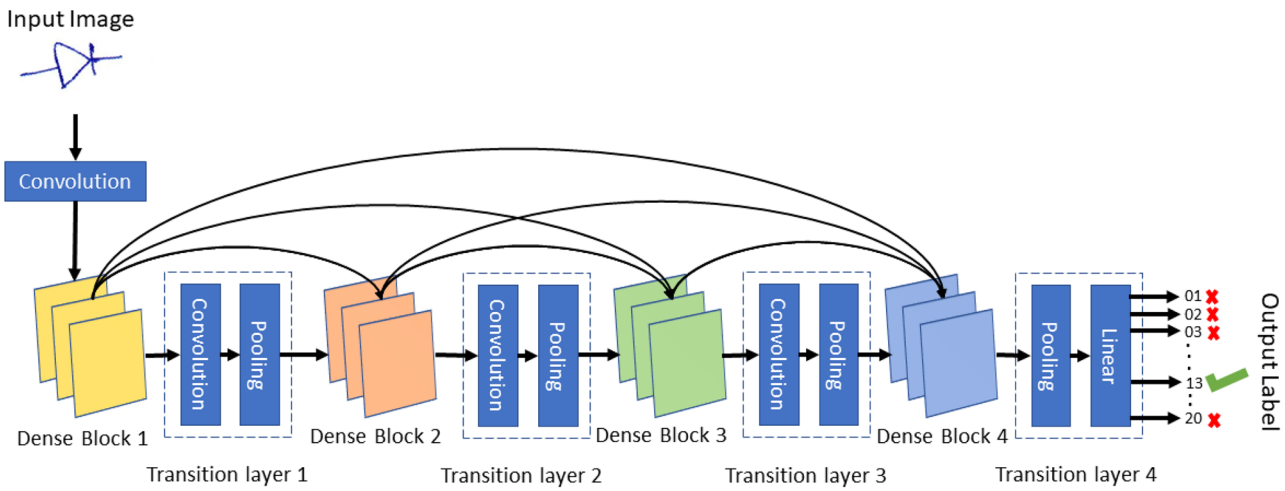


Fig. 8. A block diagram representation of the DenseNet-121 architecture.

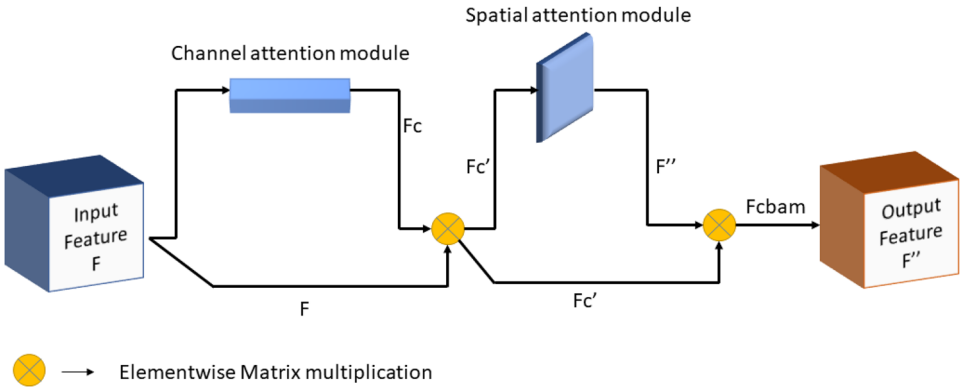


Fig. 9. A pictorial representation of the CBAM attention module.

$$w(t) = \frac{w_0}{2}\ cos(\frac{\pi \bmod (t-1, \lfloor \frac{N}{M} \rfloor)}{\lfloor \frac{N}{M} \rfloor} + 1) \tag{11}$$

In Eq. 11, $\lfloor . \rfloor$ represents the floor function, N is the total number of training epochs, M is the number of cycles (in our case we have taken $N = 100$ epochs for which the cosine annealing curve repeats itself 5 times over the entire training duration, i.e., $M = 5$), the *mod*() is the modulo operation, w_0 is the maximum learning rate, and $w(t)$ is the learning rate at epoch t.

Weighted average ensemble

The confidence scores from the top-3 snapshots, decided by their validation accuracy, are combined. While combining the class-level confidence scores, the highest weightage is assigned to the best snapshot. The predicted class is the index of the modified confidence score matrix corresponding to the highest value. Let the confidence score vector generated by top-3 snapshots be CF_1, CF_2, and CF_3 for the best, second best, and third best snapshots, respectively, and the corresponding weights are w_1, 2_2, and w_3, respectively. The combined confidence score vector CF_{avg} is calculated in Eq. 12.

$$CF_{avg} = \sum_{n=1}^{3} w_n \times log(CF_n) \tag{12}$$

In the current study, the values of the parameters w_1, w_2, and w_3 are set as 0.4, 0.3, and 0.3 respectively heuristically. It is to be noted that in Eq. 12, to normalize the elements present in the confidence score vectors (i.e., CF_1, CF_2, and CF_3), the logarithmic operator is used following the suggestions from the work[56]. The predicted class of n sample (say, c) can be obtained using Eq. 13.

$$c = argmax_{(x \in C)} CF_{avg_x} \tag{13}$$

In Eq. 13, $C = \{1, 2, 3, ..., N\}$,where N is the number of classes. In our case, $N = 20$.

Proposed benchmarking model

In this work, the CNN-aided hand-drawn circuit component classifier uses DenseNet-121 as the base CNN architecture. The output feature map (i.e., F) generated from the DenseNet-121 architecture after Dense Block 4 (see Fig. 8) has the dimension $C \times H \times W$, and it is then fed to the CBAM attention module. Next, the GAP is applied on the output feature map generated from CBAM (i.e., F_{CBAM} of dimension $C \times H \times W$) to the dimension C. Afterward, a dense layer with 20 neurons coupled with the softmax activation function is employed to obtain the predicted confidence scores of each class. We use the Snapshot ensemble technique while training the model to get five snapshots. Next, a weighted averaging ensemble is applied using the confidence scores of the three best snapshots. From the probability score CF_{avg} obtained by applying a weighted averaging ensemble, we get the predicted class using Eq. 13. The overall block diagram of the proposed model is shown in Fig. 7.

Results and discussion

In this section, we discuss various experiments that have been performed to compare results on the JUHCCR-v1 dataset. We have also analyzed the obtained results with the help of figures, graphs, and tables. We have used JUHCCR-v1.oa for training the models and model performance is evaluated on all three test sets present in JUHCCR-v1.o, JUHCCR-v1.a and, JUHCCR-v1.oa. We have used some standard metrics like accuracy (see Eq. 14), precision (see Eq. 15), recall (see Eq. 16), and F1 score (see Eq. 17) for the evaluation of the models.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

$$F1score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{17}$$

In Eqs. 14–16, *TP*, *TN*, *FP*, and *FN* represent True Positive, True Negative, False Positive, and False Negative, respectively. The performance of our proposed model and its implementation process are discussed in the following subsections.

Experimental setup

We have utilized the Open-CV and PIL libraries for augmentations performed on the images. For the implementation of the base CNNs, CBAM attention, and snapshot ensemble, we have utilized the Tensorflow-Keras library of Python. Also, the Sklearn library is utilized for metrics (i.e., accuracy, precision, recall, and F1-score) to evaluate the model for benchmarking. All experiments have been conducted using NVIDIA P100. Here, we have utilized the transfer learning protocol to train the baseline CNN architectures. The pre-trained weights obtained after training on the ImageNet dataset are used here. Adam optimizer is used with a learning rate of 0.01 for training all the base CNN architectures (see following subsection), and all these models are finetuned for 150 epochs. The training set of JUHCCR-v1.o has been split in a ratio of 70:30 (training data - 7000 and validation data - 3000) for deciding on hyperparameters, base CNN model, CBAM attention, top-3 snapshots, and the weights for weighted ensemble technique. All images are resized to dimension 160×160. The training curves of the DenseNet-121 architecture for 150 epochs are shown in Fig. 10.

Selection of base CNN model

Five baseline CNN architectures viz., ResNet-50[57], MobileNet-V2[58], Inception-V3[59], NasNet-Mobile[60], and DenseNet-121[50] are trained and evaluated to select a reasonably good base CNN model. The feature length C ($C = 1024$ for DenseNet-121, 1280 for MobileNet-V2, 1056 for NasNet-Mobile, and 2048 for Inception-V3 and ResNet-50) are obtained from these CNN models while applying GAP on the last feature maps of dimension $5 \times 5 \times C$. These features are then fed to the classification layer, which is a DL with softmax activation and has 20 units (for 20 different circuit components). All layers of the CNN model are frozen. The validation results of the base CNNs are shown in Fig.11. The DenseNet-121 outperforms the other base CNN models. Therefore, we have performed the rest experiments with the DenseNet-121 model as the base CNN model.

Usefulness of CBAM

The CBAM attention mechanism is next applied to the best-performing model, i.e., DenseNet-121. The CBAM is applied on the last block of DenseNet-121 (i.e., Dense block 4 in Fig. 8). It is to be noted here that the last block is made trainable for this purpose and trained for 40 epochs by keeping other hyperparameters, mentioned earlier, frozen. The decision to make this block trainable is made experimentally. It is observed that making the last layer trainable increases the validation accuracy from 89.18 to 89.90%. The enhanced performance of DenseNet-121 after applying CBAM attention is illustrated in Fig. 12. Also, the training and validation curves over varying epochs are shown in Fig. 10.

Choice of snapshots for ensemble

To enhance the model's performance further, we have applied the snapshot ensemble technique to the CBAM-aided DenseNet-121 model. We have captured five snapshots (after every 20 epochs), i.e., trained for 100 epochs by keeping other hyperparameters, mentioned earlier, frozen. The performance of the snapshots on the validation dataset is shown in Fig. 13. Also, the training and validation natures are shown in Fig. 10. Top-3 snapshots i.e., Snapshot 3, Snapshot 4, and Snapshot 5 are used for weighted ensemble described in subsection "Weighted average ensemble".

Performance on test sets

The performance of the test samples is evaluated with the help of the classifier ensemble method (see subsection "Weighted average ensemble"). For this, we have considered the top-3 snapshots (decided based on performance on validation samples), which in our case are Snapshot 3, Snapshot 4, and Snapshot 5 (see Fig. 13). The results of the weighted average voting ensemble are shown in Fig. 14. This figure contains benchmark performances on all three test datasets: JUHCCR-v1.o, JUHCCR-v1.a, and JUHCCR-v1.oa when model trained on train samples of JUHCCR-v1.oa dataset. In the weighted ensemble method, the weights: $w_1 = 0.40$, $w_2 = 0.30$, and $w_3 = 0.30$ (see Eq. 12) are used, which are selected heuristically (see Table 2). The benchmark performances are satisfactory at its current state considering the complexity of the problem and designing a simple benchmarking technique on the dataset for the initial attempt. The complexity increases with issues like varying drawing styles, non-uniform, incomplete, or imperfectly drawn symbols, change in ink intensity, and lower paper quality. The complexity increases with intra-class (see Fig. 5b) variation due to drawing style and inter-class shape similarity (see Fig. 5a) due to slight variation in components' representation. In addition to these, the benchmarking model designed here is simple does not considered these complexity explicitly. All these may attribute to the lower

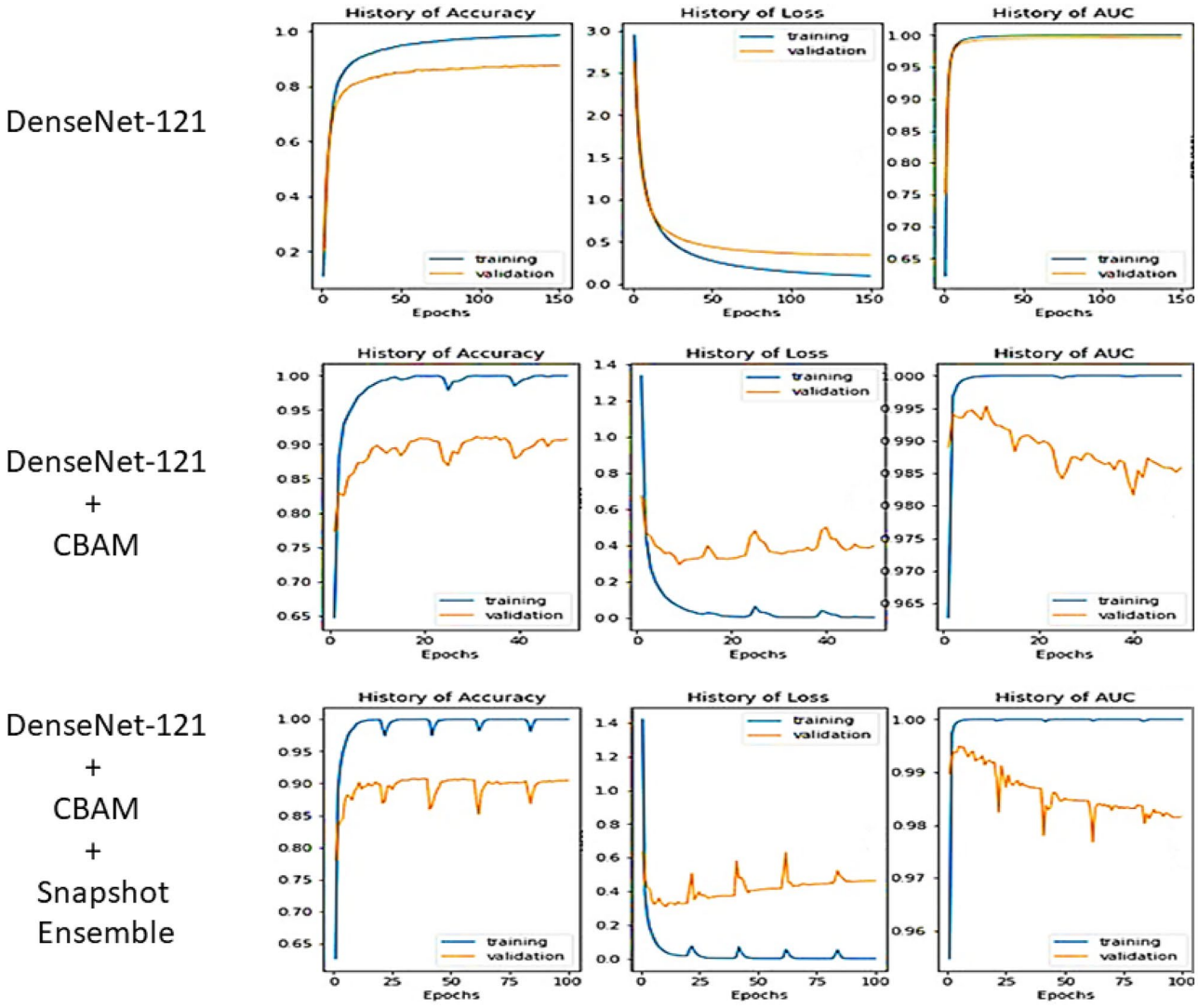


Fig. 10. Training and validation curves for all models on JUHCCR-v1.0a training and validation sets over varying epochs. Here only DenseNet-121 is considered.

performance of the current benchmarking model compared to state-of-the-art other classification models. As a result, in future, designing a more sophisticated circuit component recognizer is essential to handle these said complexities.

Discussion

The proposed benchmarking method for circuit component recognition is empowered with several deep learning and AI-aided tools. We have described the usefulness of each of the steps. The DenseNet-121 model is selected after experimenting with 5 popular CNN models. These results are already shown in Fig. 11. The recognition performance is further improved with the help of the CBAM attention module (see Fig. 12). Not only these, but we have also made use of a weighted ensemble technique that in turn, helps to raise the benchmark performance of the model (see Fig. 14). To explain these improvements, we have shown the confusion matrices at different levels in Fig.15. From these confusion matrices, it is evident that the number of wrong classifications of NOT Gate as PN Diode is 23 for DenseNet-121, which reduces to 19 when we apply CBAM, and it further reduces to 17 when a weighted averaging ensemble is applied. Also, the misclassification of the Capacitor as NOR Gate is 9 for DenseNet-121, 8 for CBAM-aided DenseNet-121, 4 after applying a snapshot ensemble on CBAM-aided DenseNet-121, and finally reduces to 3 in the case of the weighted averaging ensemble.

Apart from explaining results through confusion matrices, we have also shown the component-wise performance in terms of Recall, Precision, and F1-score of the proposed technique on "JUHCCR-v1.0" dataset in Table 1. From the results, it can be found that the benchmarking technique performs best (considering F1-score) for Power Supply (class 15), Resistor (class 16), Transformer (class 18), Inductor (class 7), and NAND Gate (class 8), while performs poorly for NPN Transistor (class 11) followed by PNP Transistor (class 14). The components of the associated class numbers are mentioned in Fig.2. In this

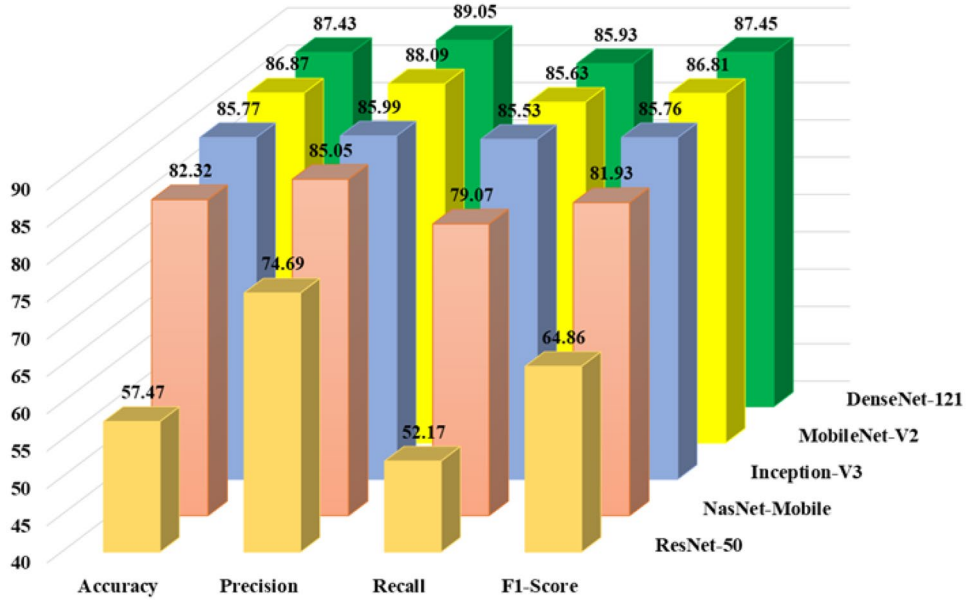


Fig. 11. Performance comparison in terms of various evaluation metrics of base CNN models on the validation samples.

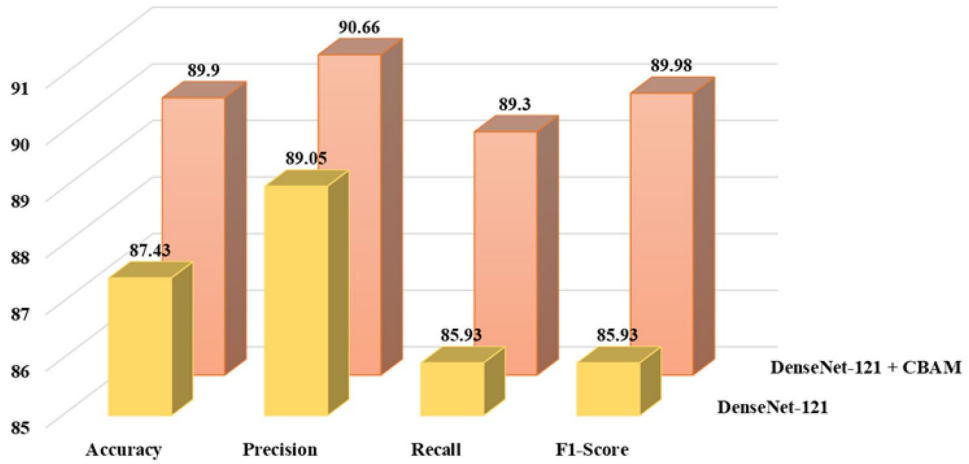


Fig. 12. Performance improvement in terms of various evaluation metrics obtained due to employing CBAM on validation data samples.

table, performances of the base CNN models (used to decide the base model for the proposed method) i.e., MobileNet-V2, NasNet-Mobile, ResNet-50, Inception-V3, and DenseNet-121, are also recorded. The comparative results show that the proposed benchmarking technique performs better or the same (in terms of F1-score) for 11 components compared to others (MobileNet-V2, and DenseNet-121 for 7 and 8 components, respectively). The present model does not perform well for components like NOT Gate (difference from best= 0.04), NPN Transistor (difference from best= 0.02), PN Diaode (difference from best= 0.06), and PNP Transistor (difference from best= 0.02) while outperforms others for components like AC Source (difference from second best= 0.07), OR Gate (difference from second best= 0.05), Resistor (difference from second best= 0.06), and NAND Gate (difference from second best= 0.03). In summary, ResNet-50 performs worst in most cases, and the present benchmarking technique performs best in most cases.

Apart from the mentioned experiments, we have also performed an ablation study to test whether our choice of weights: $w_1 = 0.40$, $w_2 = 0.30$, and $w_3 = 0.30$ (see Eq. 12), selected heuristically, in the weighted average ensemble is justified or not. The study is conducted on the JUHCCR-v1.0 test set, and the results are shown in Table 2. In this table, the values $w_1 = 3342$, $w_2 = 0.3324$, and $w_3 = 0.3335$ are taken considering the weighted

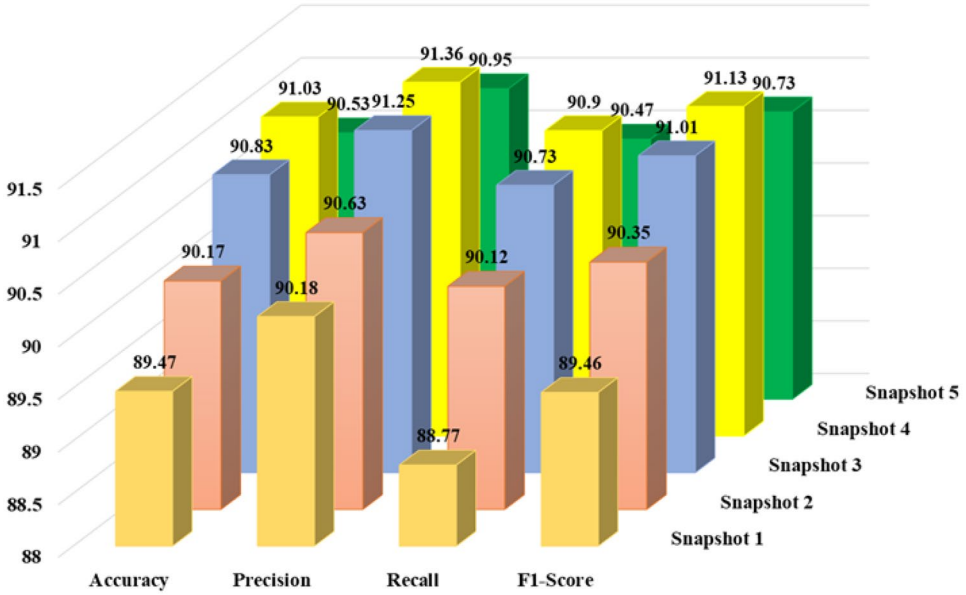


Fig. 13. Performance comparison using various evaluation metrics on the validation data of five snapshots captured here.

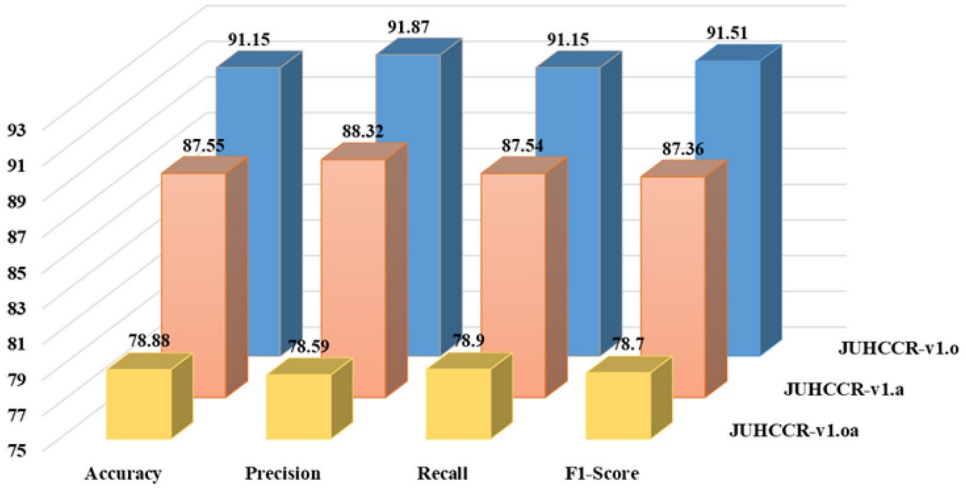


Fig. 14. Benchmark results of the weighted averaging ensemble on JUHCCR-v1.o, JUHCCR-v1.a, and JUHCCR-v1.oa datasets.

ensemble technique followed in the work[61]. From the results recorded in this table, we can safely comment that our choice is well justified with respect to the alternatives tested here.

All experiments have been conducted using a hold-out approach. Due to the random nature of deep learning models, the performance may vary over different runs on the hold-out test set. To check this uncertainty of performances, we have trained the benchmarking model 5 times using the training samples of "JUHCCR-v1.o" dataset and evaluated performance on the original test samples (i.e., test samples of "JUHCCR-v1.o"). Performances are recorded in Tabel 3. The results show that the standard deviation in performances over 5 runs is 0.16, 0.10, 0.15, and 0.11 for accuracy, precision, recall, and F1-score, respectively. These small variations in performances over several runs indicate the stability of the reported benchmark performances.

The model's complexity and size are important measures to understand the deployment feasibility of a model. For this, we have recorded the information related to the number of parameters (trainable and non-trainable) and Giga floating point operations (GFLOPs) for the present benchmarking model in Table 4 along with other base CNN models (used to decide the base model for the proposed benchmarking process), i.e., MobileNet-V2, NasNet-Mobile, ResNet-50, Inception-V3, and DenseNet-121. The proposed

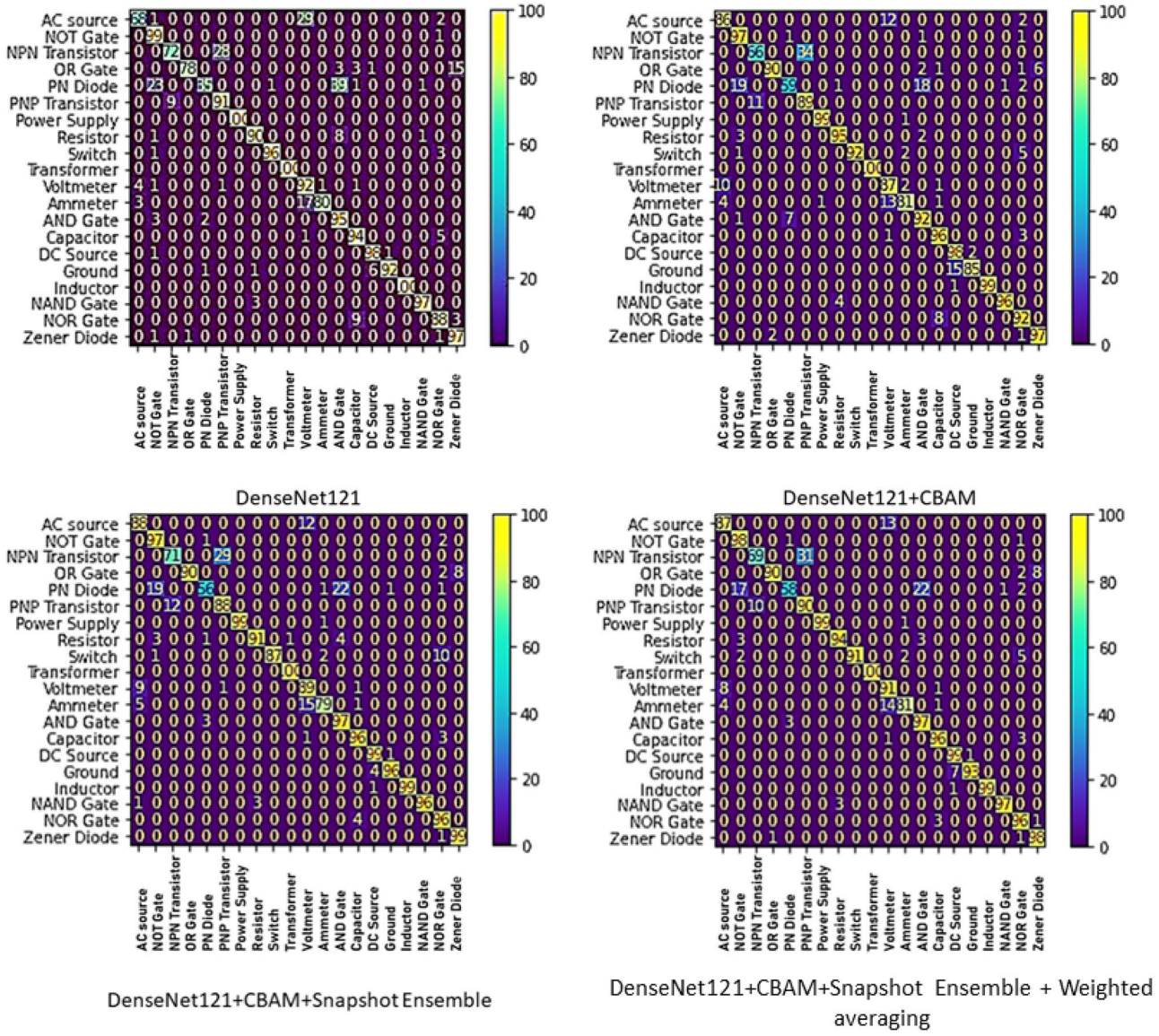


Fig. 15. Confusion matrices of the various models on the original testing dataset (o).

benchmarking model is heavier and uses more floating point operations as compared to other models, but provides better performance (see Table 1). Table 4 also shows that the proposed benchmarking model takes a longer time to execute compared to other base CNN models. The increased complexity (in terms of the number of parameters, GFLOPs, and execution time) is due to the use of CBAM and snapshot learning on top of DenseNet-121.

Experimentation on whole circuit diagrams

We have also estimated the performance of the proposed circuit component recognition system on the whole hand-drawn circuit images. To do this, we first extract the circuit components using the method proposed by Bhattacharya et al.[47]. The whole circuit diagrams (20 in total) that are considered here were made public by Bhattacharya et al.[47]. Some of the segmented outputs are shown in Fig. 16b, e, and h for the original circuit diagrams shown in Fig.16a, d, and g, respectively. In Fig. 16b, e, and h, it can be seen that sometimes coroners got identified as components (marked within the orange colored circle). Therefore, with a simple thresholding technique (i.e., components having less than 15 pixels are not valid components), such over-segmented components can be filtered out. Next, the detected valid circuit components are fed to the benchmarking model trained on the JUHCCR-v1.oa dataset to recognize them. The predicted class information is marked in Fig. 16c, f, and i for Fig. 16b, e, and 16h, respectively.

There are 110 valid circuit components in these circuit diagrams (the number of circuit components varies from 2 to 12). Out of these 110 circuit components, the circuit diagram segmentation technique[47] successfully segmented 103 circuit components (7 components get over/under-segmented). Some of the

Component	MobileNet-V2	NasNet-Mobile	Inception-V3	ResNet-50	DenseNet-121	Proposed
AC Source	0.88/0.69/0.78	0.65/0.80/0.71	0.76/0.94/0.84	0.22/0.06/0.09	0.91/0.68/0.78	0.89/0.81/0.85
NOT Gate	0.90/0.93/0.92	0.57/0.91/0.70	0.91/0.87/0.89	0.45/0.29/0.35	0.76/0.99/0.86	0.79/1.00/0.88
NPN Transistor	0.78/0.66/0.71	0.64/0.58/0.61	0.63/0.80/0.71	0.34/0.70/0.45	0.89/0.72/0.80	0.86/0.70/0.77
OR Gate	0.96/0.77/0.86	0.80/0.68/0.74	0.80/0.78/0.79	0.44/0.43/0.43	0.99/0.78/0.87	0.98/0.87/0.92
PN Diode	0.66/0.85/0.74	0.56/0.40/0.47	0.82/0.59/0.69	0.58/0.15/0.24	0.92/0.35/0.51	0.95/0.83/0.68
PNP Transistor	0.89/0.72/0.75	0.58/0.70/0.63	0.63/0.55/0.59	0.21/0.29/0.25	0.76/0.91/0.83	0.75/0.89/0.81
Power Suply	1.00/0.98/0.99	0.99/0.96/0.97	0.95/0.99/0.97	0.94/0.44/0.60	1.00/1.00/1.00	0.99/1.00/1.00
Resistor	0.98/0.56/0.71	0.93/0.86/0.90	0.91/0.97/0.94	0.81/0.51/0.63	0.96/0.90/0.93	0.99/1.00/1.00
Switch	0.97/0.98/0.98	0.93/0.77/0.84	1.00/0.86/0.92	0.77/0.47/0.58	0.99/0.96/0.97	1.00/0.95/0.97
Transformer	0.83/1.00/0.90	0.99/1.00/1.00	1.00/1.00/1.00	0.83/1.00/0.91	1.00/1.00/1.00	1.00/1.00/1.00
Voltmeter	0.66/0.85/0.74	0.54/0.62/0.58	0.86/0.67/0.75	0.48/0.11/0.18	0.66/0.92/0.77	0.69/0.91/0.78
Ammeter	0.83/0.76/0.79	0.84/0.48/0.61	0.83/0.79/0.81	0.65/0.30/0.41	0.99/0.80/0.88	0.97/0.76/0.85
AND Gate	0.89/0.90/0.90	0.75/0.79/0.77	0.68/0.89/0.77	0.45/0.27/0.34	0.66/0.95/0.78	0.78/0.93/0.85
Capacitor	0.81/0.87/0.84	0.85/0.71/0.77	0.73/0.74/0.73	0.33/0.74/0.46	0.87/0.94/0.90	0.91/0.89/0.90
DC Source	0.99/0.93/0.96	0.95/0.76/0.84	1.00/0.89/0.94	0.78/0.76/0.77	0.93/0.98/0.96	0.91/0.99/0.95
Ground	0.97/0.99/0.98	0.94/0.97/0.96	0.94/0.98/0.96	0.89/0.80/0.84	0.99/0.92/0.95	0.98/0.90/0.94
Inductor	1.00/1.00/1.00	1.00/0.93/0.96	0.94/1.00/0.97	0.69/0.99/0.81	1.00/1.00/1.00	1.00/1.00/1.00
NAND Gate	1.00/0.86/0.92	0.98/0.97/0.97	0.99/0.89/0.94	0.71/0.87/0.78	0.98/0.97/0.97	1.00/1.00/1.00
NOR Gate	0.84/0.78/0.81	0.68/0.81/0.74	0.69/0.74/0.71	0.47/0.25/0.33	0.88/0.88/0.88	0.87/0.92/0.89
Zener Diode	0.77/0.96/0.86	0.71/0.85/0.77	0.80/0.79/0.79	0.26/0.77/0.39	0.84/0.97/0.90	0.91/0.97/0.94

Table 1. Component-wise performance (Precision/Recall/F1-score) of the proposed technique on the "JUHCCR-v1.0" dataset. All scores are provided in [0, 1].

Weight parameter			Performance score (in %)			
w_1	w_2	w_3	Accuracy	Precision	Recall	F1-score
0.40	**0.30**	**0.30**	**91.15**	**91.87**	**91.15**	**91.01**
0.30	0.40	0.30	90.90	90.61	90.90	90.74
0.30	0.30	0.40	90.90	91.60	90.90	90.74
0.50	0.30	0.20	90.70	91.46	90.70	90.54
0.50	0.20	0.30	90.80	90.52	90.80	90.64
0.30	0.50	0.20	90.70	91.46	90.70	90.53
0.20	0.50	0.30	90.80	90.52	90.80	90.64
0.20	0.30	0.50	90.65	91.33	90.65	90.52
0.30	0.20	0.50	90.50	91.19	90.50	90.37
0.60	0.20	0.20	90.70	91.42	90.69	90.53
0.20	0.60	0.20	90.65	91.40	90.65	90.50
0.20	0.20	0.60	90.25	91.12	90.89	90.25
0.3342	0.3324	0.3335	90.85	91.56	90.85	90.69

Table 2. Ablation study concerning hyperparameters of the weighted averaging ensemble on the JUHCCR-v1.0 dataset. The values 0.4, 0.3, and 0.3 are considered for w_1, w_2, and w_3 since they give the best results.

Run	Accuracy	Precision	Recall	F1-score
1	91.15	91.87	91.15	91.51
2	91.28	91.95	91.38	91.66
3	91.02	91.72	90.96	91.34
4	90.89	92.03	91.08	91.55
5	91.31	91.84	91.28	91.56
Overall	91.13 ± 0.16	91.88 ± 0.10	91.17 ± 0.15	91.52 ± 0.11

Table 3. Performances of the proposed model in terms of different evaluation metrics (in %) on the "JUHCCR-v1.0" dataset across five runs of the training process.

Model	# Parameters			GFLOPs	Execution time	
	Trainable	Non-trainable	Total		Per-step (ms)	Total (s)
MobileNet-V2	25,620	2,257,984	2,283,604	0.3	887	30
NasNet-Mobile	21,140	4,269,716	4,290,856	1.1	1000	43
ResNet-50	40,980	23,587,712	23,628,692	4.1	3000	108
Inception-V3	40,980	21,802,784	21,843,764	6.0	2000	53
DenseNet-121	20,500	7,037,504	7,058,004	2.9	151	8
Present	282,743	7,037,504	7,320,247	3.3	4000	139

Table 4. Model architecture specifications with computational metrics and execution time. It is to be noted that "Per-step" (in ms) and "Total time" (in s) indicate the average time taken to process a single batch during inference and the total inference time for the complete test dataset, respectively.

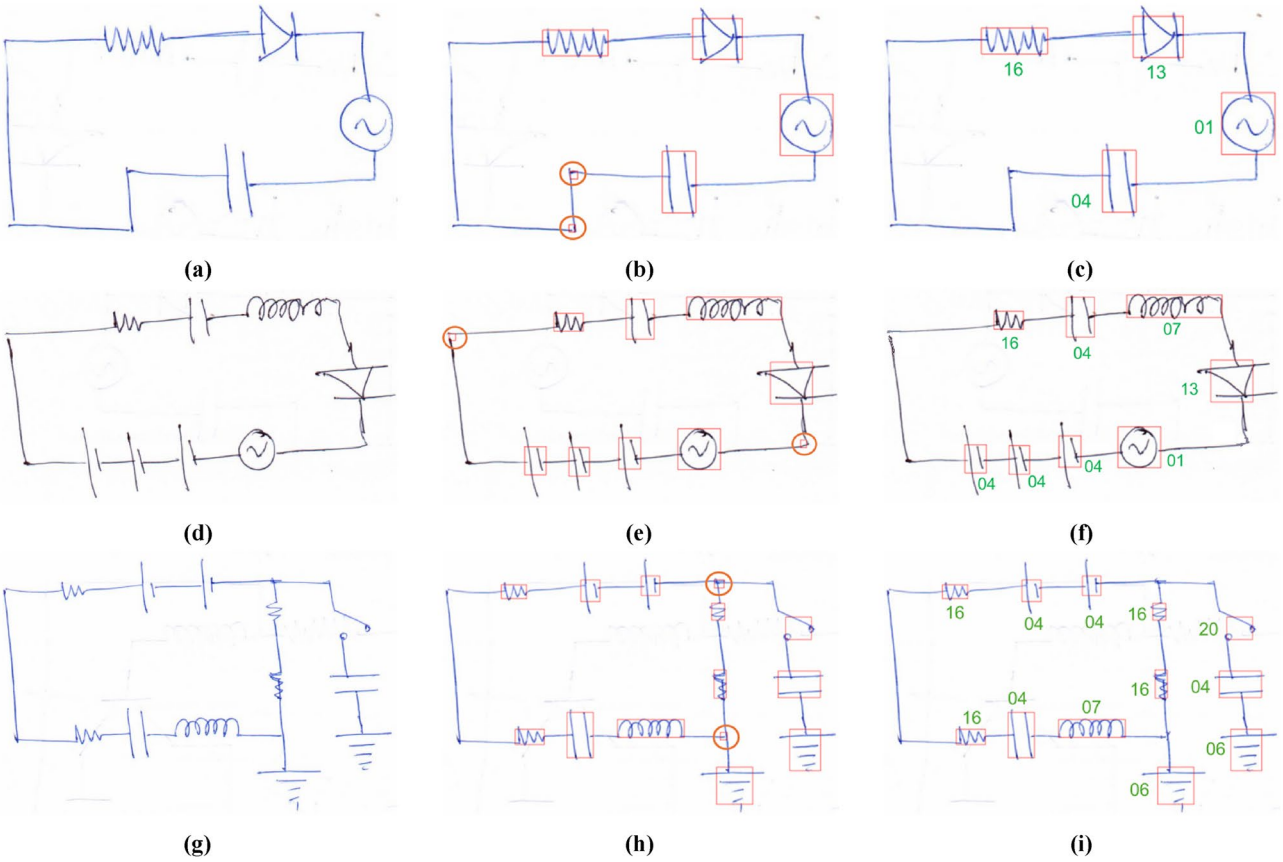
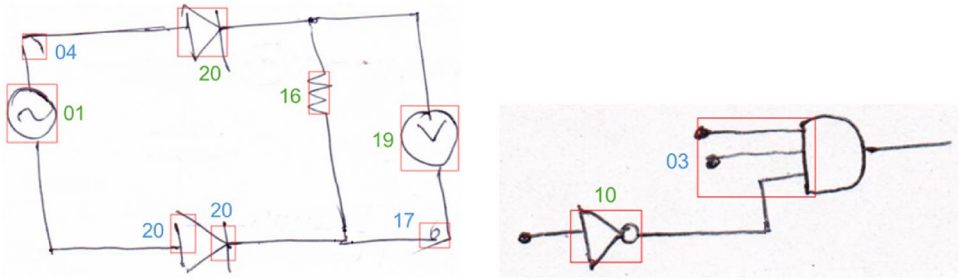


Fig. 16. Results (successful cases) on recognition of circuit components present in a whole circuit diagram. (**a**), (**d**), and (**g**): original circuit diagrams. (**b**), (**e**), and (**h**): detected circuit components using the technique by Bhattacharya et al.[47]. (**c**), (**f**), and (**i**): recognized circuit component information i.e., the predicted class number (see Fig. 2) for all detected components. The green colored text indicates successful recognition.

over/under-segmented cases are shown in Fig. 17. Out of these 103 properly segmented components, the proposed circuit component recognition technique that is designed for benchmarking purposes here, recognizes 102 circuit components correctly, i.e., $\sim 99\%$ correct classifications while all the components belonging to 17 circuit diagrams (i.e., 85% of the circuit diagrams) are successfully segmented and recognized. In summary, a better segmentation technique is required to have better circuit diagram recognition system.

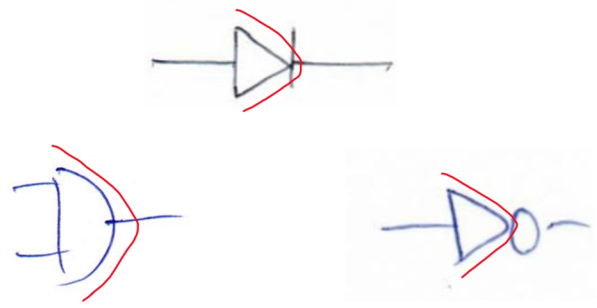
Error analysis

The proposed model accurately recognizes a particular circuit component in its image form. However, inter-class misclassification occurs when the shapes (see Fig. 5) are quite similar. Some cases of misclassification can be explained as follows.



(a) (b)

Fig. 17. Results (unsuccessful cases) on recognition of circuit components present in a whole circuit diagram. The recognized circuit component information i.e., the predicted class number (see Fig. 2) for all the detected components are shown. The green color text indicates successful recognition while blue color indicates the erroneous ones.



(a) Similarity between PN Diode, AND Gate and NOT Gate

(b) Similarity between NPN and PNP transistor symbols **(c)** Similarity between AC Source and Voltmeter symbols

Fig. 18. Source of misclassification due to structural similarities.

- A major portion (3.90%) of the entire misclassification in the testing dataset arises due to the PN junction diode. The structural and shape similarity of the PN junction diode with AND and NOT gates might be the reason for such misclassification (see Fig. 18a).
- Another major misclassification in the dataset tested by the proposed model is between the PNP Transistor and the NPN Transistor. This can be explained by observing that these two components have exactly similar structures (see Fig. 18b). The only difference between these two shapes is the direction of the arrow.
- A very minimal misclassification occurs between the AC source and voltmeter because the portion inside the circle of these two components looks somewhat similar in the hand-drawn component images (see Fig. 18c).

Conclusion and future scope

In this work, the need for preparing a dataset for classifying hand-drawn electrical and electronic circuit components has been established. Hence, the focus is mainly given to the methods implemented to prepare a comprehensive and diverse dataset, called JUHCCR-v1, which has 20 different most commonly used circuit components in electrical or electronics circuit diagrams. For benchmarking the results on this dataset, a deep learning-aided method has been designed. In this model, first, various basic CNN models are evaluated on the dataset, out of which, DenseNet-121 produces the best results in terms of some standard evaluation metrics. In order to further improve the classification performance, a CBAM attention layer has been added to the DenseNet-121 model. Following that, top-3 performing snapshots out of five snapshots have been taken into consideration to design a weighted averaging ensemble method to generate final the output, achieving an accuracy of 91.15% on JUHCCR-v1.o, 87.55% on JUHCCR-v1.a and 78.88% on JUHCCR-v1.oa.

However, there are some limitations of this work. The main limitation is that we have considered individual symbols. But, for real-life applications, we have to segment an entire circuit diagram into components to classify

each of them. Here, erroneous segmentation of components leads to erroneous full circuit recognition, and therefore, it requires further attention while designing a competent segmentation model. Another limitation is that the benchmarking model is heavy, and therefore, it takes more inference time as it uses CBAM and a snapshot ensemble approach on the top of DenseNet-121 model. Hence, for ease of deployment in practical scenarios, we need to design a lightweight CNN model. In the future, we will increase more samples in each class to add more variety to the dataset, which would, in turn, make the developed systems more robust. Though considering the complexity of the research domain, we can say that the obtained results are reasonable in its current state. However, to make it useful in real-life scenarios, we have to design more sophisticated and lightweight models that can deal with the extreme drawing variations of any particular symbol, and the misclassification among similarly shaped symbols, and thus reduces overall classification error.

Data Availability

All the datasets developed and used in this work will be made publicly available to the research community. The codes and some sample images are available in the link:https://github.com/AyushRoy2001/Circuit-Component-Analysis.

Received: 29 May 2025; Accepted: 29 September 2025
Published online: 04 November 2025

References

1. Roy, S., Bhattacharya, A., Sarkar, N., Malakar, S. & Sarkar, R. Offline hand-drawn circuit component recognition using texture and shape-based features. *Multimed. Tools Appl.* **79**, 31353–31373 (2020).
2. Okazaki, A., Kondo, T., Mori, K., Tsunekawa, S. & Kawamoto, E. An automatic circuit diagram reader with loop-structure-based symbol recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **10**, 331–341. https://doi.org/10.1109/34.3898 (1988).
3. Mathur, A. & Achar, R. Hand-drawn circuit schematic digitization and netlisting using machine learning with emphasis on signal integrity applications. In *2024 IEEE 33rd Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, 1–3 (IEEE, 2024).
4. Hsu, C.-Y. et al. Computer-aided design of an electrode based on the stone-wales defected zno graphene-like monolayers for sodium-ion batteries. *Mater. Sci. Semicond. Process.* **185**, 108997 (2025).
5. Sun, Y., Li, X. & Sha, Z. Large language models for computer-aided design (llm4cad) fine-tuned: Dataset and experiments. *J. Mech. Des.* 1–19 (2025).
6. Pavithra, S., Shreyashwini, N., Bhavana, H., Nikhitha, G. & Kavitha, T. Hand-drawn electronic component recognition using orb. *Procedia Comput. Sci.* **218**, 504–513 (2023).
7. Chattopadhyay, D. *Electronics (fundamentals and applications)* (New Age International, 2006).
8. Mathur, A. & Achar, R. Recognition of electronic component orientations from hand-drawn circuit schematics through a two stage machine learning system. In *2024 22nd IEEE Interregional NEWCAS Conference (NEWCAS)*, 26–29 (IEEE, 2024).
9. Bohara, B. & Krishnamoorthy, H. S. Deep learning-based framework for power converter circuit identification and analysis. *IEEE Access* (2024).
10. Agrawal, V., Jagtap, J. & Kantipudi, M. P. An overview of hand-drawn diagram recognition methods and applications. *IEEE Access* **12**, 19739–19751 (2024).
11. AlMughrabi, A. & Hiary, H. Hand-drawn electric circuit diagrams recognition using deep learning. In *2024 28th International Conference on Information Technology (IT)*, 1–4 (IEEE, 2024).
12. Malakar, S. et al. Handwritten word recognition using lottery ticket hypothesis based pruned cnn model: A new benchmark on cmaterdb2 1.2. *Neural Comput. Appl.* **32**, 15209–15220 (2020).
13. Moetesum, M., Younus, S. W., Warsi, M. A. & Siddiqi, I. Segmentation and recognition of electronic components in hand-drawn circuit diagrams. *EAI Endorsed Trans. Scalable Inf. Syst.* **5**, e12–e12 (2018).
14. Dey, M. et al. A two-stage CNN-based hand-drawn electrical and electronic circuit component recognition system. *Neural Comput. Appl.* **33**, 13367–13390 (2021).
15. Günay, M., Köseoğlu, M. & Yıldırım, Ö. Classification of hand-drawn basic circuit components using convolutional neural networks. In *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 1–5 (IEEE, 2020).
16. Yang, C., Wang, J., Yang, L., Shi, D. & Duan, X. Intelligent digitization of substation one-line diagrams based on computer vision. *IEEE Trans. Power Deliv.* (2023).
17. Bhutra, H., Tanishq, D. S., Samidala, B., Venugopal, V. & Neelima, N. Enhanced recognition of hand-drawn circuit components using fast r-cnn and faster r-cnn. In *2024 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*, 301–306 (IEEE, 2024).
18. Ahmed, N. *et al.* Digitize-hcd: A dataset for digitization of handwritten circuit diagrams. *Data Brief* 111315 (2025).
19. De Jesus, E. O. & Lotufo, R. D. A. ECIR-an electronic circuit diagram image recognizer. In *Proceedings SIBGRAPI'98. International Symposium on Computer Graphics, Image Processing, and Vision (Cat. No. 98EX237)*, 254–260 (IEEE, 1998).
20. Dewangan, A. & Dhole, A. Knn based hand drawn electrical circuit recognition. *Int. J. Res. Appl. Sci. Eng. Technol.* **6**, 1–6 (2018).
21. Rachala, R. R. & Panicker, M. R. Hand-drawn electrical circuit recognition using object detection and node recognition. *SN Comput. Sci.* **3**, 244 (2022).
22. Amraee, S., Chinipardaz, M., Charoosaei, M. & Mirzaei, M. A. Handwritten logic circuits analysis using the yolo network and a new boundary tracking algorithm. *IEEE Access* **10**, 76095–76104 (2022).
23. Bailey, D., Norman, A., Moretti, G. & North, P. *Electronic schematic recognition* (Massey University, 1995).
24. Rabbani, M., Khoshkangini, R., Nagendraswamy, H. & Conti, M. Hand drawn optical circuit recognition. *Procedia Comput. Sci.* **84**, 41–48 (2016).
25. Mondal, R., Malakar, S., Barney Smith, E. H. & Sarkar, R. Handwritten english word recognition using a deep learning based object detection architecture. *Multimed. Tools Appl.* **81**, 975–1000 (2022).
26. Chen, Y. et al. Yolo-ms: rethinking multi-scale representation learning for real-time object detection. *IEEE Trans. Pattern Anal. Mach. Intell.* (2025).
27. Rekavandi, A. M. et al. A guide to image-and video-based small object detection using deep learning: Case study of maritime surveillance. *IEEE Trans. Intell. Transp. Syst.* (2025).
28. Kononenko, I., Robnik-Sikonja, M. & Pompe, U. Relieff for estimation and discretization of attributes in classification, regression, and ilp problems. *Artif. Intell. Methodol. Syst. Appl.* 31–40 (1996).

29. Ghosh, M., Malakar, S., Bhowmik, S., Sarkar, R. & Nasipuri, M. Memetic algorithm based feature selection for handwritten city name recognition. In *Computational Intelligence, Communications, and Business Analytics: First International Conference, CICBA 2017, Kolkata, India, March 24–25, 2017, Revised Selected Papers, Part II*, 599–613 (Springer, 2017).
30. Malakar, S. et al. A holistic approach for handwritten hindi word recognition. *Int. J. Comput. Vis. Image Process. (IJCVIP)* **7**, 59–78 (2017).
31. Dai, Y.-Y. & Braytont, R. K. Circuit recognition with deep learning. In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 162–162 (IEEE, 2017).
32. Feng, G., Viard-Gaudin, C. & Sun, Z. On-line hand-drawn electric circuit diagram recognition using 2d dynamic programming. *Pattern Recogn.* **42**, 3215–3223 (2009).
33. Lakshman Naika, R., Dinesh, R., an approach based on finite state machine. Handwritten electric circuit diagram recognition. *Int. J. Mach. Learn Comput.* **9**, 374–380 (2019).
34. Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779–788 (2016).
35. Shrivastava, A., Gupta, A. & Girshick, R. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 761–769 (2016).
36. Ren, S., He, K., Girshick, R. & Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems***28** (2015).
37. Hu, W., Zhan, X. & Tong, M. Parsing netlists of integrated circuits from images via graph attention network. *Sensors* **24**, 227 (2024).
38. Said, A. et al. Circuit design completion using graph neural networks. *Neural Comput. Appl.* 1–13 (2023).
39. Loong, C. N., San Juan, J. D. Q. & Chang, C.-C. Image-based structural analysis for education purposes: A proof-of-concept study. *Comput. Appl. Eng. Educ.* (2023).
40. Al Mehdawi, M., Dreesbach, T., Gösling, H. & Brinkmann, M. Investigating the use of augmented reality and machine learning in electrical engineering courses. In *Wirtschaftsinformatik*, 168 (2023).
41. Roy, S., Sarkar, D., Malakar, S. & Sarkar, R. Offline signature verification system: A graph neural network based approach. *J. Ambient Intell. Human. Comput.* 1–11 (2021).
42. Sarkar, D., Roy, S., Malakar, S. & Sarkar, R. A modified gnn architecture with enhanced aggregator and message passing functions. *Eng. Appl. Artif. Intell.* **122**, 106077 (2023).
43. Sarkar, R., Malakar, S., Das, N., Basu, S. & Nasipuri, M. A script independent technique for extraction of characters from handwritten word images. *Int. J. Comput. Appl.* **1**, 83–88 (2010).
44. Malakar, S., Sarkar, R., Basu, S., Kundu, M. & Nasipuri, M. An image database of handwritten Bangla words with automatic benchmarking facilities for character segmentation algorithms. *Neural Comput. Appl.* **33**, 449–468 (2021).
45. Bhowmik, S. et al. A holistic word recognition technique for handwritten Bangla words. *Int. J. Appl. Pattern Recognit.* **2**, 142–159 (2015).
46. Ghosh, M., Malakar, S., Bhowmik, S., Sarkar, R. & Nasipuri, M. Feature selection for handwritten word recognition using memetic algorithm. *Adv. Intell. Comput.* 103–124 (2019).
47. Bhattacharya, A., Roy, S., Sarkar, N., Malakar, S. & Sarkar, R. Circuit component detection in offline handdrawn electrical/electronic circuit diagram. In *2020 IEEE Calcutta Conference (CALCON)*, 80–84 (IEEE, 2020).
48. Barua, S., Malakar, S., Bhowmik, S., Sarkar, R. & Nasipuri, M. Bangla handwritten city name recognition using gradient-based feature. In *Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications: FICTA 2016, Volume 1*, 343–352 (Springer, 2017).
49. Bhowmik, S. et al. Off-line Bangla handwritten word recognition: A holistic approach. *Neural Comput. Appl.* **31**, 5783–5798 (2019).
50. Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708 (2017).
51. Dey, S., Roychoudhury, R., Malakar, S. & Sarkar, R. Screening of breast cancer from thermogram images by edge detection aided deep transfer learning model. *Multimed. Tools Appl.* **81**, 9331–9349 (2022).
52. Dey, S., Roychoudhury, R., Malakar, S. & Sarkar, R. An optimized fuzzy ensemble of convolutional neural networks for detecting tuberculosis from chest x-ray images. *Appl. Soft Comput.* **114**, 108094 (2022).
53. Paul, A., Pramanik, R., Malakar, S. & Sarkar, R. An ensemble of deep transfer learning models for handwritten music symbol recognition. *Neural Comput. Appl.* **34**, 10409–10427 (2022).
54. Woo, S., Park, J., Lee, J.-Y. & Kweon, I. S. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, 3–19 (2018).
55. Huang, G. et al. Snapshot ensembles: Train 1, get m for free. arXiv:1704.00109 (2017).
56. Majumder, S., Ghosh, S., Malakar, S., Sarkar, R. & Nasipuri, M. A voting-based technique for word spotting in handwritten document images. *Multimed. Tools Appl.* **80**, 12411–12434 (2021).
57. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778 (2016).
58. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520 (2018).
59. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826 (2016).
60. Zoph, B., Vasudevan, V., Shlens, J. & Le, Q. V. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8697–8710 (2018).
61. Dey, S., Bhattacharya, R., Malakar, S., Mirjalili, S. & Sarkar, R. Choquet fuzzy integral-based classifier ensemble technique for covid-19 detection. *Comput. Biol. Med.* **135**, 104585 (2021).

Acknowledgements

The authors acknowledge the support provided by the Center for Microprocessor Applications for Training Education and Research (CMATER) Laboratory, Department of Computer Science and Engineering, Jadavpur University, Kolkata, India. This research work was supported by the ongoing FIST program of the Department of Computer Science and Engineering, Jadavpur University, titled 'FIST Engineering Sciences Level B/C/D Project'. The program is funded by the Department of Science and Technology, Government of India, under reference number SR/FST/ET-I/2022/1059(C).

Author contributions

A.R., S.P., S.M., R.S. conceived the idea and analysed the supporting theory. R.P., S.P. and S.M. designed the experimental protocol. R.P. and S.P. conducted the experiments. S.M. and R.S. analyzed the implications of the outcomes. S.M., E.C., M.P. and R.S. supervised the entire work. All authors reviewed and approved the final manuscript.

Funding
There is no funding involved with this research.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to E.C.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.