Hand-drawn Circuit Component Recognition using Deep Learning

Apurva Umredkar[1*]

[1*]Department of Electronics and Communication Engineering, Visvesvaraya National Institute of Technology, Nagpur, 440010, Maharashtra, India.

Corresponding author(s). E-mail(s): apoorv.umredkar@students.vnit.ac.in;

Abstract

A circuit diagram is a necessary tool for the development of many electrical products. With the birth of the era of automation, deep learning algorithms can reduce the effort necessary to reproduce hand-drawn circuit designs into simulation tools. A custom convolutional neural network capable of detecting hand-drawn circuit components with a classification accuracy of 94.94 percent is described in this article. The model features a simple design that may be used on devices with limited computing power. Additionally, the model may be utilised as a foundation for segmenting all accessible components in a comprehensive circuit layout, as well as tracing connections and generating a simulation-ready schematic.

Keywords: circuits, components, computer vision, convolutional neural networks

1 Introduction

A circuit diagram is a visual representation of an electrical circuit made up of several components represented by symbols that are widely known in the industry. It is the most basic tool used by engineers, and it is frequently employed as the first stage in the design of any product, ranging from a simple light switch to complex applications such as microprocessor development. When prototyping, the most usual approach is to design the circuit diagram by hand on paper before utilizing a simulation software such as NI Multisim or Proteus Design Suite.

With the start of the automation era, strong techniques such as machine learning (ML) are becoming increasingly useful in decreasing human labor. Advanced methods for detecting and recognizing objects in images have been developed in the field of computer vision (CV). Individual components in hand-drawn circuit designs can be recognized by the program, which can then be automatically supplied into a simulation tool. A method like this helps minimize human errors.

This study describes a simple convolutional neural network architecture capable of detecting 10 industry-standard circuit symbols with a classification accuracy of 94.94 percent, which was inspired by the work provided by [1–5]. The architecture is designed in such a way that it may be used on devices with less computing capabilities. The dataset generation is covered in section 2, while the proposed neural network design is covered in section 3. In the section 4, the results are presented and described.

2 Dataset

For this investigation, a dataset of images of 10 different circuit component symbols was created. Hand-drawn images were collected from multiple subjects using various drawing styles and lighting circumstances. This contributed to the dataset diversification required for building a robust machine learning algorithm. Samples of the hand-drawn images have been presented in Fig 1.

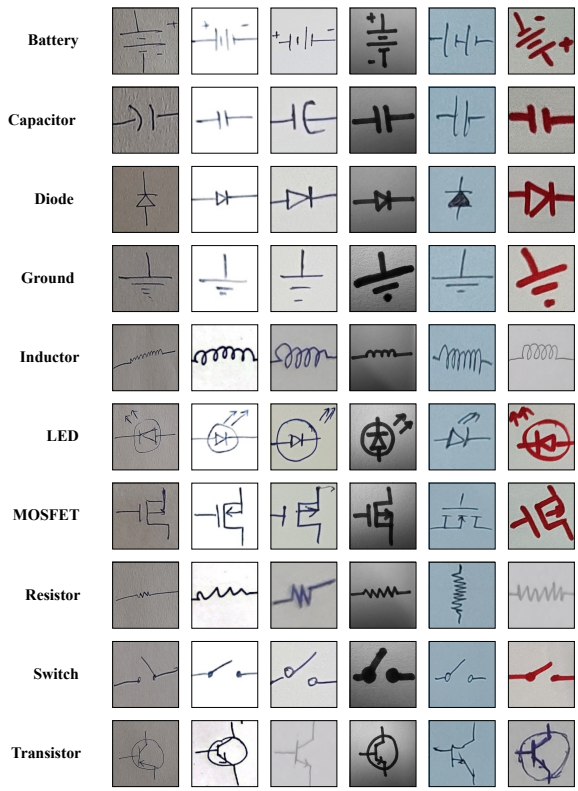


Fig. 1: Samples from originally captured images for each circuit component class.

2.1 Preprocessing

The acquired images were passed through a preprocessing pipeline to make them suitable for the deep learning model. The images were subjected to binarization using adaptive thresholding functions and then brought to a uniform size of 84×84 pixels. The preprocessing pipeline has been illustrated in Fig 2.

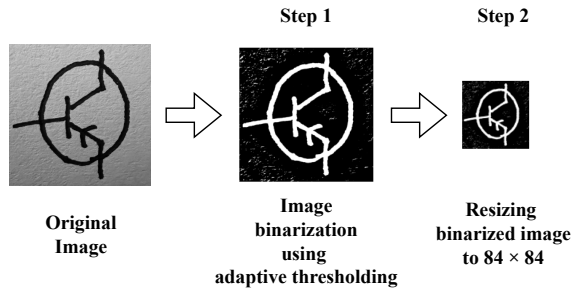


Fig. 2: Image preprocessing pipeline.

2.2 Data augmentation

In order to increase the size of the dataset, the images were later augmented using the following transforms:

1. Horizontal flipping
2. Rotation by 10°& 30°in clockwise & counter-clockwise directions
3. Addition of noise:
   (a) Gaussian
   (b) Salt & pepper

The image count before and after augmenting the dataset has been summarized in Table 1.

Table 1: Number of images per class available in the dataset before and after augmentation.

Class	Pre-augmentation	Post-augmentation
Battery	69	2070
Capacitor	69	2070
Diode	69	2070
Ground	66	1980
Inductor	70	2100
LED	66	1980
MOSFET	65	1950
Resistor	72	2160
Switch	68	2040
Transistor	68	2040
Total	**682**	**20460**

It can be seen that the dataset has an augmentation factor of 30, implying that 29 new images are created from a single existing image.

The final dataset contains 20,460 binary images of the 10 different circuit component classes, each measuring 84×84 pixels, which were put into the training program using the PyTorch ML framework's ImageFolder function. The dataset was shuffled and then split with a

ratio of 80-20% for training and testing the deep learning model.

3 Proposed Approach

A convolutional neural network was designed for feature extraction from the dataset and circuit component classification.

3.1 Feature extraction

Two-dimensional convolution operation uses a kernel matrix, most often of the shape 3×3 or 5×5. The values of these kernel matrices establish a relationship between the input and the output of the model and are the parameters that are constantly updated during the training process such that the model can make predictions with minimum error.

Pooling layers are used to reduce the dimensions of the input feature map and hence no learnable parameters are involved. There are two types of pooling layers commonly used: maximum pooling and average pooling. Maximum or max-pooling calculates the maximum value in the patch for down-sampling the dimensions whereas average pooling calculates the average of all the values present in the patch.

The proposed neural network architecture consists of four convolutional layers, two of them with a kernel size of 3×3 and the other two with 5×5. Each layer has been activated using the non-linear Rectified Linear Unit (ReLU) function. Each convolutional layer is followed by a max-pooling layer (except the last layer) with a default patch size and strides value of 2×2. The network has been illustrated in Fig 3 and the details of every layer of the CNN have been tabulated in Table 2.

3.2 Classification

The vectors are flattened to one dimension and processed through a series of fully connected layers after features are retrieved from the input image. The ReLU activation function is used to introduce non-linearity into the weights of network nodes or neurons. The Softmax activation function is used to activate the last layer, which yields the probability distribution of each class's prediction confidence.

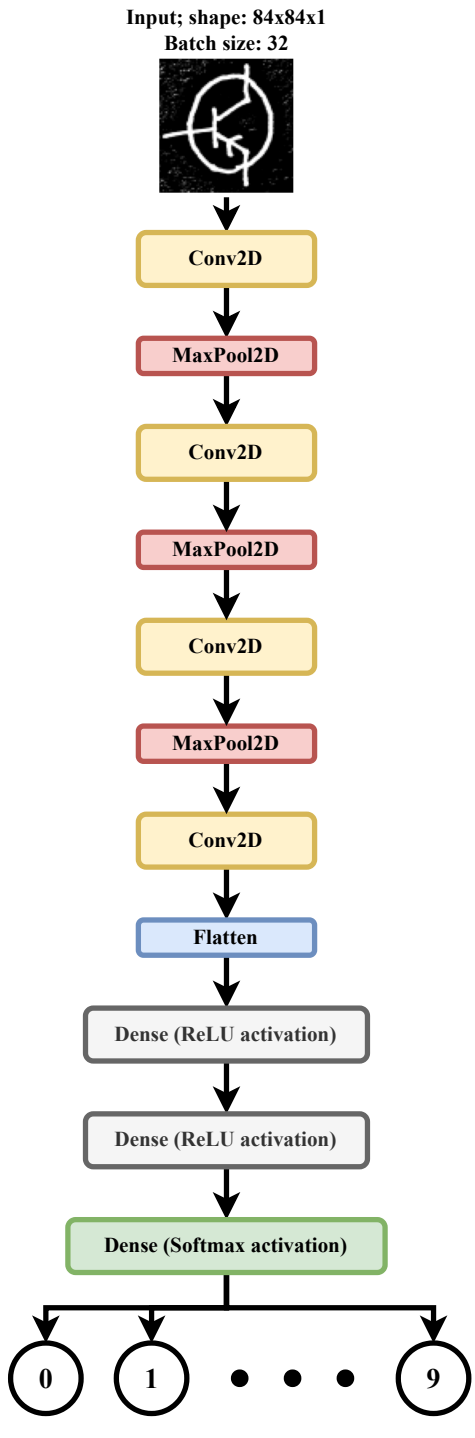


Fig. 3: Architecture of the proposed convolutional neural network.

Table 2: Layer details of the proposed architecture

#	Layer	Number of filters	Kernel size	Units	Layer settings	Output dimensions	No. of trainable parameters
1	Input	1	-	-	-	84 x 84 x 1	0
2	Conv2D	6	5	-	Activation : ReLU Padding : Valid	80 x 80 x 6	156
3	MaxPool2D	-	2	-	Strides : 2	40 x 40 x 6	0
4	Conv2D	16	3	-	Activation : ReLU Padding : Valid	38 x 38 x 16	880
5	MaxPool2D	-	2	-	Strides : 2	19 x 19 x 16	0
6	Conv2D	24	5	-	Activation : ReLU Padding : Valid	15 x15 x 24	9624
7	MaxPool2D	-	2	-	Strides : 2	7 x 7 x 24	0
8	Conv2D	32	3	-	Activation : ReLU Padding : Valid	5 x 5 x 32	6944
9	Flatten	-	-	-	-	800	0
10	Dense	-	-	320	ReLU activation	320	256320
11	Dense	-	-	80	ReLU activation	320	25680
12	Dense	-	-	10	Softmax activation	10	0
Total number of trainable parameters							**300,414**

3.3 Hyperparameters

Aside from its architecture, the hyperparameters used to train the deep learning model have an impact on its training performance and accuracy. After multiple hit-and-trial efforts, the optimal combination of these hyperparameters is determined.

The Adam optimizer was chosen since it has been proven to be the most effective adaptive optimizer for gradient descent algorithms. The optimizer's learning rate was set to an optimum 0.001 to avoid difficulties with exploding and vanishing gradients [6, 7].

The number of training samples used in one iteration of forward propagation through the network is referred to as batch size. Small batch sizes may not guarantee convergence to the global minima of the convex objective function [8], while large batch sizes often result in poor generalization. For this experiment, a batch size of 32 was chosen.

The categorical crossentropy loss function was used, which is common for multi-class classification applications.

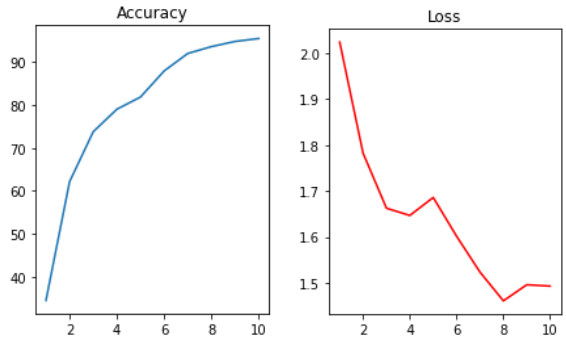
Table 3: Hyperparameters used for training the proposed CNN.

Hyperparameters	**Values**
Optimizer	Adam
Learning rate	0.001
Number of epochs	10
Batch size	32
Train-test split ratio	80-20%
Loss function	Categorical CrossEntropy

4 Results

A deep model extracts more information compared to a shallow model but consequently consumes more time for its training. The training process can be accelerated using graphics processing units (GPU) [9], which are capable of carrying out extensive computations such as matrix calculations required in the convolution operation, at a much faster rate than a CPU due to their parallel processing architecture. An entry-level Nvidia GeForce MX130 GPU was used for this study.

The variation of the training accuracy and loss throughout the training process was recorded and can be visualized using the graphs presented in Fig 4.



(a) Training accuracy (b) Training loss

Fig. 4: Variation of training accuracy and training loss over the epochs.

It can be seen that with each passing epoch, the training accuracy improves and the loss converges to the convex function's global minima. The model's performance metrics have been reviewed in Table 4.

Table 4: Performance evaluation of the neural network.

Performance Metrics	
Training time (with GPU)	314.59 seconds
Training accuracy	95.14%
Training loss	1.463
Test accuracy	94.94%
Test loss	0.056

A confusion matrix, as shown in Fig 5, can be used to determine a supervised learning algorithm's classification capabilities. While the neural network is capable of successfully recognizing most components, it fails to distinguish between components that have similar shapes, such as batteries and capacitors, resistors and inductors, MOSFETs and transistors.

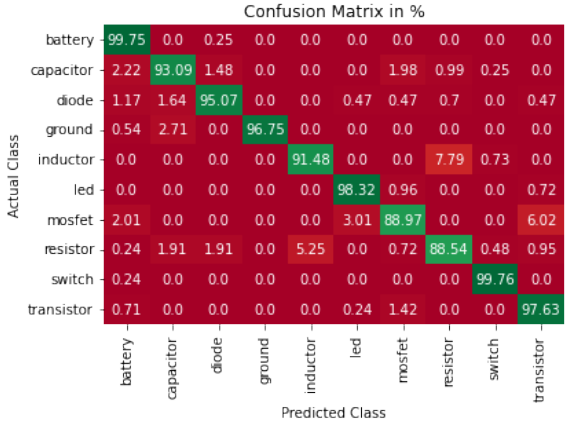


Fig. 5: Confusion matrix

5 Discussion & Future Scope

The proposed model can be developed further to compete with previous works, as evidenced by the results given above (section 4). Extra stages can be added to the preprocessing pipeline to entirely eliminate the noise from the raw photos. Rather of relying on data augmentation, the size of the dataset can be raised by gathering new samples. New component classes, such as OP-AMPS, logic gates, flip-flops, and so on, must be added to the dataset. To increase classification accuracy, the deep learning architecture can be updated and combined with additional computer vision techniques, which should solve the inability to distinguish component symbols with identical shapes.

Individual components can be segmented from a comprehensive circuit diagram using the recognition model. The automation pipeline should be able to trace the circuit diagram's connections and possibly output a SPICE code that can be fed into simulation tools.

6 Conclusion

A simple and efficient convolutional neural network for distinguishing hand-drawn circuit components is provided in this paper. The dataset was custom-made by collecting sample photographs from a variety of subjects, each with a different sketching style and lighting situation, with the goal of adding diversity to the dataset. The dataset was preprocessed by adaptive thresholding to binarize the images and then reduced to a reasonable resolution so that the network could be trained on a device with less processing capacity. To make the dataset bigger, image transforms such flips, rotations, and noise addition were used. The PyTorch ML framework was used to construct, train, and evaluate the deep learning model. The model had a classification accuracy of 94.94 percent and a recall of 94.94 percent and a 0.056 loss. The model's performance was analyzed, and areas for improvement were identified.

References

[1] Mrityunjoy Dey, Shoif Md Mia, Navonil Sarkar, Archan Bhattacharya, Soham Roy, Samir Malakar, and Ram Sarkar. A two-stage cnn-based hand-drawn electrical and electronic circuit component recognition system. *Neural Comput. Appl.*, 33(20):13367–13390, oct 2021. ISSN 0941-0643. doi: 10.1007/s00521-021-05964-1. URL https://doi.org/10.1007/s00521-021-05964-1.

[2] Mahdi Rabbani, Reza Khoshkangini, Nagendraswamy Siddappa, and Mauro Conti. Hand drawn optical circuit recognition. *Procedia Computer Science*, 84:41–48, 12 2016. doi: 10.1016/j.procs.2016.04.064.

[3] Soham Roy, Archan Bhattacharya, Navonil Sarkar, Samir Malakar, and Ram Sarkar. Offline hand-drawn circuit component recognition using texture and shape-based features. *Multimedia Tools and Applications*, pages 1 – 21, 2020.

[4] Rachala Rohith Reddy and Mahesh Raveendranatha Panicker. Hand-drawn electrical circuit recognition using object detection and node recognition. *ArXiv*, abs/2106.11559, 2021.

[5] Haiyan Wang, Tianhong Pan, and Mian Khuram Ahsan. Hand-drawn electronic component recognition using deep learning algorithm. *Int. J. Comput. Appl. Technol.*, 62 (1):13–19, jan 2020. ISSN 0952-8091. doi: 10.1504/ijcat.2020.103905. URL https://doi.org/10.1504/ijcat.2020.103905.

[6] George Philipp, Dawn Xiaodong Song, and Jaime G. Carbonell. The exploding gradient problem demystified - definition, prevalence, impact, origin, tradeoffs, and solutions. *arXiv: Learning*, 2017.

[7] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–1310–III–1318. JMLR.org, 2013.

[8] Ibrahem Kandel and Mauro Castelli. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, 6, 05 2020. doi: 10.1016/j.icte.2020.04.010.

[9] Amr Kayid, Yasmeen Khaled, and Mohamed Elmahdy. Performance of cpus/gpus for deep learning workloads. 05 2018. doi: 10.13140/RG.2.2.22603.54563.

[10] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.