



# Graph-Based Object Detection Enhancement for Symbolic Engineering Drawings

Syed Mizanur Rahman<sup>1,2</sup>, Johannes Bayer<sup>1,2(✉)</sup>, and Andreas Dengel<sup>1,2</sup>

<sup>1</sup> Smart Data & Knowledge Services Department,  
DFKI GmbH, Kaiserslautern, Germany

{mizanur\_rahman.syed,johannes.bayer,andreas.dengel}@dfki.de

<sup>2</sup> Computer Science Department, TU Kaiserslautern, Kaiserslautern, Germany

**Abstract.** The identification of graphic symbols and interconnections is a primary task in the digitization of symbolic engineering diagram images like circuit diagrams. Recent approaches propose the use of Convolutional Neural Networks to the identification of symbols in engineering diagrams. Although recall and precision from CNN based object recognition algorithms are high, false negatives result in some input symbols being missed or misclassified. The missed symbols induce errors in the circuit level features of the extracted circuit, which can be identified using graph level analysis. In this work, a custom annotated printed circuit image set, which is made publicly available in conjunction with the source code of the experiments of this paper, is used to fine-tune a Faster RCNN network to recognise component symbols and blob detection to identify inter-connections between symbols to generate a graph representation of the extracted circuit components. The graph structure is then analysed using graph convolutional neural networks and node degree comparison to identify graph anomalies potentially resulting from false negatives from the object recognition module. Anomaly predictions are then used to identify image regions with potential missed symbols, which are subject to image transforms and re-input to the Faster RCNN, which results in a significant improvement in component recall, which increases to **91%** on the test set. The general tools used by the analysis pipeline can also be applied to other Engineering Diagrams with the availability of similar datasets.

**Keywords:** Graph convolutional network · Circuit diagram · Graph refinement

## 1 Introduction

Graph-based symbolic engineering drawings (like circuit diagrams or piping and instrumentation diagrams) use graphical symbols and line segments to represent the components of technical facilities or devices as well as their interconnections.

In addition, these images can contain texts to provide further information about individual components. The digitization of such images implies the extraction of these information - components, interconnections and texts to extract a complete graph description of the digitized source document. An early attempt for such an extraction is described in [20]. Symbols connected through lines or multiple segments are a feature of many engineering diagrams such as mechanical engineering diagrams and Piping and Instrumentation Diagrams (P&IDs).

In this paper, The GraphFix framework<sup>1</sup> is proposed as an approach to digitize circuit diagrams, but is envisioned to be applicable to other similar document types with the help of suitable datasets. GraphFix is a multi-stage information extraction framework to identify different types of information in an engineering document. First of all, a Faster RCNN [15] is trained to identify component symbols, which form the node of the extracted graph. Based on that, blob detection is used to predict the connections (wiring) between the components, which make up the graph's edges.

The resulting graph structure is amenable to the application of graph refinement and error detection algorithms. There are two main types of errors in the component proposal list: false positives - component proposals in regions of the diagram, which do not have any identifiable symbol and false negatives - which can either result from a symbol in the input image being misclassified or not recognised as a symbol region, the latter type are referred to as Unmarked False Negatives (UFNs) in this work and result in graph anomalies, which can be detected with the help of Graph Convolutional Networks (GCNs) or with node degree comparison.

An attempt is also made to refine symbol labels in the component list generated by the Faster RCNN, using graph level features and symbol characteristics such as position and size. Some methods achieve up to **60%** recall@1, but the refinement does not help achieve any improvement in the overall recall of the framework when these predictions are combined with the Faster RCNN results. However, using graph anomalies, UFNs can be detected, which results in an improvement in recall@1 of up to **2-4%**.

This paper is further structured as follows: Sect. 2 describes related work in digitization of engineering diagrams (EDs) and circuit diagrams in particular. This section also briefly introduces topics on graph refinement and other concepts touched upon in this work. Section 3 provides information on the printed circuit ED dataset used to train and test GraphFix as well as the different data augmentation techniques used to improve the object recognition module's performance. The different processing steps required by GraphFix are explained under Methodology 4 and a review of the overall performance and of the different refinement and error detection techniques is presented in the Results Sect. 5. In conclusion, Sect. 6 summarises the salient contributions of this work as well limitations of the framework, which can be addressed in future research.

---

<sup>1</sup> <https://github.com/msyed-unikl/GraphFix>.

## 2 Related Work

Digitization of document types such as electric circuits, floor plans and P&IDs share many common features. [12] lists the identification of symbols, interconnections and text and some of main challenges in the digitization of engineering diagrams such as circuit diagrams. Circuit diagrams can utilize different standards and conventions to represent symbols for circuit components, resulting in difficulty in producing a well defined dataset, which captures variations in symbol style, pose and scale [12]. Initial attempts to identify symbols in circuit components such as [2] extract graphical primitives such as lines and circles from images and apply rule based templates to identify component symbols. [11] proposes a system to identify symbols in hand drawn engineering diagrams based on subgraph isomorphism by representing symbols and drawings as relational graphs using which the system could also learn to identify new symbols.

More recent approaches such as [13] extract graphical features from hand drawn circuits and inputs them to a neural network for symbol recognition. Other notable attempts utilizing machine learning based approaches as opposed to rule matching include [1], which proposes a probabilistic-SVM classifier using Histogram of Oriented Gradients (HOG) and Radon Transform features and [4], which used geometric analysis to analyse vertical, horizontal and circuit space features to identify symbols in electric and electronic symbols. Direct comparisons of the effectiveness of the different methods is not possible as these systems are trained on different (and often private) datasets with varying sets of symbols despite the availability of a standard dataset for electrical circuits [18].

Alternate approaches to symbol detection using Convolutional Neural Networks (CNNs) were proposed by [6] and more recent attempts in this direction include [17] that combines deep learning based Faster RCNN [15] and semantic segmentation with other statistical methods for such as morphology and component filtration for the vectorization of floor plans. [14] employs a VGG-19 based Fully Convolutional Neural Network to identify symbols in P&ID diagrams. [22] uses a Region based Convolutional Network (RCNN) to generate region proposals for ‘symbols’ and ‘dummy’ regions in P&ID images. [21] also proposes the use of RCNNs for the identification of symbols in P&ID images. GraphFix follows a similar approach by training a Faster RCNN module to train electric component symbols in the custom printed circuit ED dataset.

Multiple computer vision based approaches have also been applied task of identifying connections between symbols in EDs. [4] applies a Depth First Search by considering darkened pixels as nodes and introducing edges between nodes for adjacent pixels. [14] applies Hough Lines Transform to identify pipelines between symbols in P&IDs. GraphFix uses blob detection to identify connections between identified component symbols. Blobs are regions in an image, which differ from their surroundings in terms of image features such as the pixel colour [19]. Blob detection can be carried out with a number of algorithms such as Laplacian of Gaussian (LoG) or Difference of Gaussian (DoG) [19].

Symbol extraction as well as connection identification from ED images are associated with potential errors such as misclassification (for symbol extraction

only), false negatives and false positives. Some works have attempted to use graph or network level features to identify errors in the extraction process. The use of graph based rules to modify graph structures extracted from P&ID diagrams is proposed in [3]. [14] creates a forest structure with the extracted components and uses properties of P&ID diagrams to detect false positive pipelines identified by the Hough Transform algorithm.

[5] applies morphological operations to the shape-graph space of a tree of connected components extracted from maps to filter out components not corresponding to expected layers. In [16], graph features are used to detect node labels for regions in floorplan images by converting floorplans to Region Adjacency Graphs and using Zernike moments as node attributes. Graph Neural Networks and Edge Networks proposed in [16] achieve up to **100%** accuracy on ILIPso dataset [8] in predicting node labels. Experiments attempting to predict node labels (component symbols) using graph or regional features in this work (Graph Refinement) showed limited success and have much lower accuracy compared to the Faster RCNN module. The GraphFix pipeline achieves its improvement over the Faster RCNN’s recall by identifying regions in the circuit image with symbols that have been missed by the Faster RCNN module and this is done by identifying anomalies in the graph extracted from the circuit images using node degree comparison and Graph Convolutional Networks (GCNs).

GCNs provide a semi supervised graph based approach to predict node labels [9]. The Fourier basis on a graph is defined as the eigenvalues of the graph Laplasian, which is defined as  $D - A$  – where  $D$  is the diagonal degree matrix (diagonal elements are the degrees of the nodes and other elements are 0) and  $A$  is the adjacency matrix for the Graph [10].

Graph convolutions can be defined on the Fourier basis, but such an approach is prohibitively expensive for large graphs as the calculation of eigenvector matrix is  $O(N^2)$  where  $N$  is the number of nodes in the graph. Localised spectral filters to make spectral convolutions computationally feasible were proposed in [7]. [7] also put forward the use of a truncated expansion of Chebyshev polynomials as an approximation of the eigendecomposition of the Laplasian. [9] further develops this model to propose a GCN model which can be represented by

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta$$

In this equation from [9],  $\tilde{D} = \Sigma_{ij} \tilde{A}_{ij}$ . Where  $\tilde{A} = A + I_N$ .  $A$  represents the adjacency matrix for the graph and  $X \in \mathcal{R}^{N \times C}$  is the input signal (with  $C$  real input parameters for  $N$  nodes).  $\Theta \in \mathcal{R}^{C \times F}$ , where  $F$  is the number of filter maps. The GCN layer is 1-localized convolution and multiple GCN layers can also be stacked, which approximates higher level localization convolutions [9].

### 3 Printed Circuit ED Dataset

Computer Aided Design (CAD) software used to generate circuit diagrams can use different symbols and standards to represent symbols. A dataset used to train an object detection module to identify various types of symbols for component

types should consist of diagrams from various sources, to capture component symbols differing in style, pose, and other visual features and characteristics. Hand drawn circuits are not included in the ground truth to avoid extreme heterogeneity<sup>2</sup>.

The ground truth needed to run training, validation and testing for the Faster RCNN module and the graph algorithms is generated by scraping an online source<sup>3</sup> for circuit diagram images, which are converted to a standard image format. Component symbols in these images are then manually annotated<sup>4</sup> to mark symbols in the image with the corresponding component labels. Certain conventions and rules are necessary in order to maintain consistency in the use of labels for components across multiple diagram standards. Some of the important labelling conventions are:

- Circuit components are labeled such that there are no overlapping components.
- Wires are not annotated. However, wire crossings and overlaps are labelled as symbols
- Simple variations in symbol style, line colour and spatial orientation are tolerated and grouped under the same label. However, when a different symbol type is used constantly with a qualified component symbol (such as a ‘ground’ and ‘digital ground’), separate label categories are created
- If a composite component (consisting of multiple smaller components) is repeatedly encountered, then the composite symbol is treated as a label category. For example, a rectifier bridge is labelled as one rectifier bridge as opposed to four diodes

Ground truth graphs for the electric circuits, component regions or bounding boxes from the manually annotated symbol list are input to blob detection to identify interconnections. The resulting graphs carrying symbols and connecting edges then comprise the ground truth for graph based methods.

In total, the ground truth consists of **218** annotated printed circuit EDs. These are divided into **182** images for training, **18** diagrams for validation at the time of training and **18** images are set aside for testing the framework. **85** symbol categories have been identified and the dataset has **8697** annotations. For each image in the training set, **8** variants are generated by randomly applying image transforms such as scaling, horizontal flipping, colour inversion and Gaussian noise (Fig. 1).

## 4 Methodology

GraphFix proposes a modular extraction of information from diagram images and their refinement. After image pre-processing (such as format conversion), the Faster RCNN object recognition module identifies component symbols in

<sup>2</sup> <https://github.com/msyed-unikl/printer-d-circuit-ED-dataset>.

<sup>3</sup> [discovercircuits.com](https://discovercircuits.com).

<sup>4</sup> LabelImg - <https://github.com/tzutalin/labelImg>.

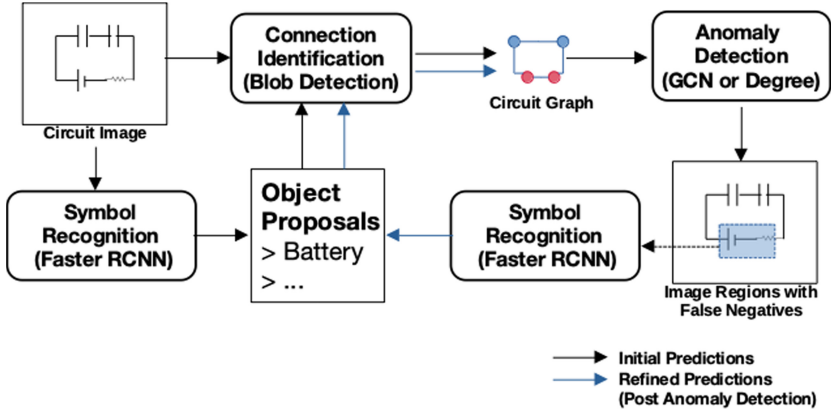


Fig. 1. Circuit digitization workflow proposed by GraphFix.

the image and their locations. This information is then used to identify connections between components using blob detection. The graph generated with the extracted symbols and connections can be subject to graph refinement, where graph anomalies are detected and used to identify component symbols missed by the object recognition module. Finally, select regions of the diagram are subject to image augmentation (cropping and scaling) and input again to the Faster RCNN module to identify these missing symbols.

#### 4.1 Symbol Recognition

The classification head of a pre-trained (COCO Dataset) Faster RCNN module with a ResNet 50 backbone<sup>5</sup> is replaced with a new head to classify component symbols and the model is retrained (or fine-tuned) on the training dataset.

Object proposals from the Faster RCNN consist of bounding box, label and confidence scores. Multiple proposals with different confidence scores can be generated for the same component symbol. GraphFix merges overlapping bounding boxes to create a ‘prediction cluster’ with labels sorted by the confidence score. The list of prediction clusters are then compared with components in the ground truth to generate precision and recall metrics for the symbol detection task.

With grid search, training parameters such as the optimiser algorithm, training batch size and learning rate can be adjusted to maximise recall@1 on the validation set (see Table 1).

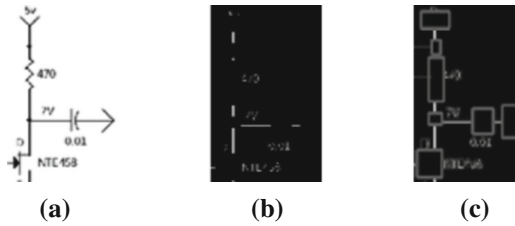
#### 4.2 Connection Identification

After component symbols and their locations are identified, connections between components can be extracted using blob detection and some simple and intuitive

<sup>5</sup> <https://pytorch.org/vision/stable/models.html#object-detection-instance-segmentation-and-person-keypoint-detection>.

computer vision related operations. First, all component regions are blanked out by merging them into the background. After this, the image is converted to grayscale, colours are inverted and a threshold is applied to remove noise. A blob detection algorithm is then applied to identify unique blobs and a bounding box is identified for each blob.

The corners of these bounding boxes are checked for proximity to two or more symbol components and blobs in proximity of two or three symbols are shortlisted. In case the blob bounding box is in proximity to two symbols, then a connection between the symbols (a graph edge between the symbol nodes) is identified. If three symbols are in proximity to the blob bounding box, two of the three symbols are shortlisted such that the area of the rectangle between the two symbol ‘touch points’ is the maximum (Fig. 2).



**Fig. 2.** a. Section of circuit for connection identification b. After symbol removal, thresholding and colour inversion c. Identification of blobs with vertices in vicinity of component symbols ([discovercircuits.com](http://discovercircuits.com)).

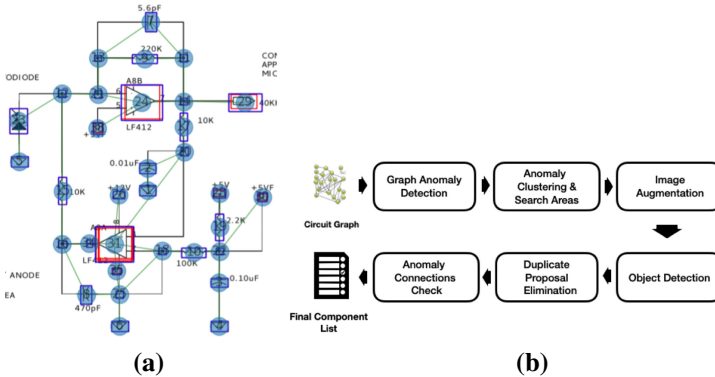
There are some drawbacks to this technique such as:

- Component symbols must be erased before blob detection is applied. This is easily achieved with the ground truth as symbols are identified manually. However, for testing, the Faster RCNN object proposals can completely miss symbols (UFNs). This results in missing connections linked to these symbols and also, ‘overshot’ connections, which is noticed when symbols such as wire crossings or junctions are missed. In the latter case, the symbol is misidentified as a part of a wire connection in the image and incorrect connections can be added to the graph.
- The proposed method also misses complicated wire representations which loop around a component symbol.

Despite these drawbacks, the method identifies connections with a high precision and recall. However, these parameters cannot be quantitatively measured with the current printed circuit ED dataset as it does not list connections between symbols and hence the algorithms’s output is suitable for manual assessment. Blob detection can also be used to identify text annotations and on circuit images in the printed circuit ED dataset, it performs on par with the EAST deep learning based text recognition algorithm [23]. However, the application of OCR systems to image regions with text results in inaccurate character and symbol recognition and thus not considered for graph refinement.

### 4.3 Anomaly Detection

Most errors seen in the output of the block detection based connection identification stem from symbols missed by the Faster RCNN module. Identifying these Unmarked False Negatives (UFNs) can lead to the correction of a number of such errors. These false negatives do not have any corresponding node in the extracted electric graph as no object proposals corresponding to the ground truth object exist in the Faster RCNN output (in contrast there are also other false negatives in the ground truth, which have overlapping object predictions with incorrect labels). When blob detection is carried out on output from the Faster RCNN, connections between UFNs and nodes connected to UFNs in the ground truth will be missed or misidentified by the blob detection. These nodes connected to the UFNs in the ground truth are present in the Faster RCNN output (ignoring the possibility of two UFNs connected to one another) and are termed anomalies. Two methods - node degree comparisons and GCNs are applied to the task of identifying anomalies (Fig. 3).



**Fig. 3.** a. Sample graph generated from symbol and connection identification steps ([discovercircuits.com](https://discovercircuits.com)). b. Unmarked False Negative (UFN) Identification Process with Image Augmentation.

### 4.4 Training Set for Anomaly Detection

To train models to identify anomalies, training set images are input to the Faster RCNN model and blob detection to generate graphs for the circuits. The anomalies in these graphs can be identified by comparing them with the ground truth and identifying nodes connected to UFNs. For Faster RCNN trained on the basic dataset up to **5%** of nodes in ground truth are UFNs and **8%** of nodes in the Anomaly Detection training set are anomalies. This results in under-sampling of anomaly nodes, which has to be handled while training models.



### 4.5 Anomaly Detection Techniques

**Node Degree Comparison.** Using the Anomaly Detection training set, two discreet probability distributions of the observed degree for each symbol type can be calculated. The first distribution is for instances where the symbol is encountered as an anomaly and the second distribution is for symbol instances that do not occur as anomalies.

These two distributions for a symbol type are represented as  $\mathbf{P}(\mathbf{D}_n = \mathbf{d}_n | \mathbf{A}_n = \mathbf{True})$  and  $\mathbf{P}(\mathbf{D}_n = \mathbf{d}_n | \mathbf{A}_n = \mathbf{False})$ , which are the probabilities that the degree  $\mathbf{D}$  of the node  $\mathbf{n}$  is an observed natural number  $\mathbf{d}_n$  given  $\mathbf{A}_n$  is **True** indicating the node  $\mathbf{n}$  is anomalous or **False**, which denotes that node  $\mathbf{n}$  is not an anomaly.

These distributions are then used to predict if node occurrences in the test set graphs are anomalies.

**$\mathbf{A}_n$  is True if**

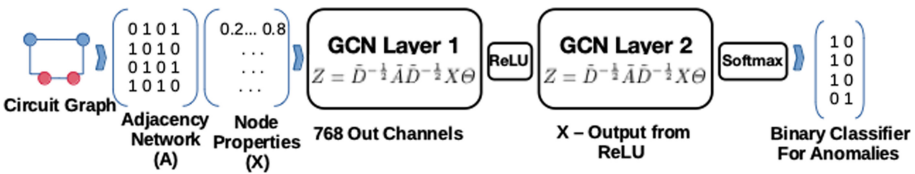
$$P(D_n = d_n | A_n = True) > P(D_n = d_n | A_n = False)$$

Using Bayes Theorem, an alternate can also be suggested where instead of using  $\mathbf{P}(\mathbf{D}_n = \mathbf{d}_n | \mathbf{A}_n = \mathbf{True})$  and  $\mathbf{P}(\mathbf{D}_n = \mathbf{d}_n | \mathbf{A}_n = \mathbf{False})$ ,  $\mathbf{P}(\mathbf{A}_n = \mathbf{True} | \mathbf{D}_n = \mathbf{d}_n)$  and  $\mathbf{P}(\mathbf{A}_n = \mathbf{False} | \mathbf{D}_n = \mathbf{d}_n)$  can be estimated with prior probabilities  $\mathbf{P}(\mathbf{A}_n = \mathbf{True})$  and  $P(\mathbf{A}_n = \mathbf{False})$  and the condition becomes:

**$\mathbf{A}_n$  is True if**

$$P(D_n = d_n | A_n = True) \cdot P(A_n = True) > P(D_n = d_n | A_n = False) \cdot P(A_n = False)$$

In practice,  $\mathbf{P}(\mathbf{A}_n = \mathbf{True})$  is usually very small (less than **0.1**) and therefore anomaly detection with Bayesian priors have an extremely low recall.



**Fig. 4.** A two layer GCN network to detect anomalies in circuits. In a three layer GCN, the output of the second layer (768 out channels) is input to a second ReLU activation and then input to a third GCN layer followed by a softmax function

**GCNs:** The anomaly training dataset can also be used to train a GCN based network to identify anomalies in test set graph outputs from the Faster RCNN and blob detection. In addition to anomalies in the Faster RCNN output in the training set, nodes are artificially dropped in the training set graphs, to generate additional anomalies. All neighbours previously connected to a dropped node

can be marked as anomalies. Multiple variants of each training set circuit graph are generated and nodes are dropped randomly. The data is then used to train GCN networks to identify anomaly nodes. Multiple architectures are trained by varying the training parameters such as number of GCN layers, probability of a node being dropped and the number of variants for each electric graph to maximise recall for anomaly detection (Fig. 4).

The performance of the two anomaly detection methods is presented in the Results section.

#### 4.6 Detection of False Negatives

After identification of anomalies in the Faster RCNN output in electric graphs, GraphFix proceeds to use this information to identify the UFNs causing the anomalies by using the following observations:

- Anomalies are connected to the UFN in the ground truth, so a UFN is expected to be in the proximity of anomalies in most cases
- There should be a connection between an anomaly and new object proposals for UFNs. This is useful in dropping false positive UFN proposals
- A UFN can be connected to one or more anomalies in the ground truth - this assumption allows for the clustering of anomalies in close proximity

Some further assumptions are made to cluster anomalies and to identify suitable search areas in which UFNs may potentially be located:

- Distances between components in the circuit are distributed normally with mean *avg distance* and standard deviation *std dev* - then the distance between the anomaly and a UFN can be represented in these terms
- *avg distance* and *std dev* can also be used to identify anomaly clusters - since anomalies in a cluster should be connected to the same UFN. GraphFix uses a heuristics based method to cluster anomalies by first defining  $D_c = \alpha_c * avg\ distance + \beta_c * std\ dev$ . With  $\alpha_c$  and  $\beta_c$  set based on experimentation. Each node is assigned to a unique cluster initially. Two clusters are aggregated if the minimum distance between nodes of the two clusters is below  $D_c$ . The previous step is repeated till no more clusters can be aggregated. If an anomaly cluster has more than one nodes, the UFN is expected to be located 'near the center' of these anomaly nodes

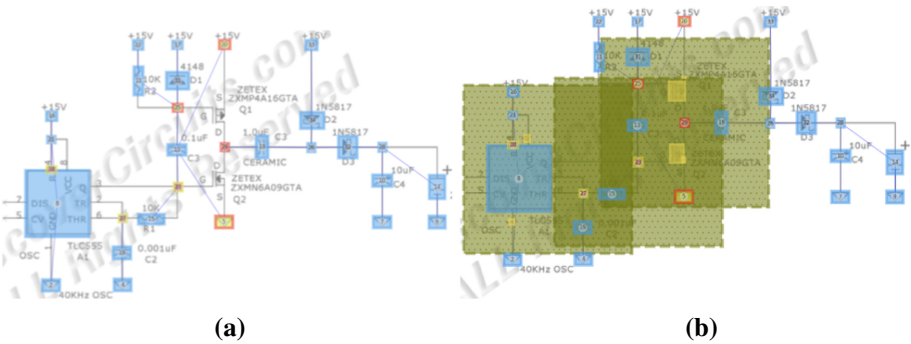
An anomaly cluster with a single node can result from a UFN positioned either above or below or to the left or right of the anomaly. Hence a square search area with side  $S_c = 2 * D_p$  where  $D_p = \alpha_p * avg\ distance + \beta_p * std\ dev$  with  $\alpha_p$  and  $\beta_p$  values set using trial and error is assigned to such single node clusters. Clusters with two anomalies result in a rectangular search area which is centered at the centroid of the two anomaly components and both the length and breadth of the rectangle are set to  $S_c$ .

For clusters with more than two anomalies, the search area is chosen as the minimum rectangle that covers all the anomalies in the cluster - assuming that the anomalies are resulting from one UFN surrounded by the anomalies.

After finalizing search areas, two alternate methods are tested to identify UFNs within the selected regions:

- By default, Faster RCNN outputs proposals with a confidence score greater than **0.05**, the threshold for Faster RCNN object proposals is lowered to **0.01** and the object proposals with confidence scores between **0.05** and **0.01** are stored in a separate list. This list can be searched for object proposals in the search areas
- Search areas can be cropped and subject to image transform such as stretching. The transformed image is input to the Faster RCNN module to identify object proposals. The new proposals have to be converted back to the coordinates of the original circuit image

Experiments on the printed circuit ED dataset show that image augmentation yields better results than lowering the Faster RCNN confidence score and was selected for further processing. Proposals identified using image augmentation are compared with the original output of the Faster RCNN to identify and delete repeats corresponding to components already identified. Finally, blob detection is carried out to identify connections between anomaly nodes and the new object proposals. Any new object proposals not connected to anomalies are dropped as false positives to arrive at the final list of extracted components.



**Fig. 5.** a. Correctly identified anomalies (red borders with red fill) and false negative anomalies (red borders with yellow fill) b. Search areas (green dashed borders) and detected Unmarked False Negatives UFNs (green fill) on a sample circuit ([discovercircuits.com](http://discovercircuits.com)). (Color figure online)

## 4.7 Graph Refinement

Representing electric circuits as graphs with components as nodes and interconnections as edges allows the possibility to apply graph refinement techniques to the extracted information. Graph refinement can be applied for a number of tasks in graphs such as (i) Error Detection (identify wrong node labels or erroneous connections) (ii) Identification of missing edges

For graphs generated from engineering diagrams, the detection of incorrectly labeled nodes is of special interest, as this can potentially improve symbol recall. Two techniques (symbol location and size based prediction, and GCNs) to identify node labels using graph and image features have been presented here. The methods are trained on ground truth graph data and tested on the graphs generated using Faster RCNN output on test set diagrams.

**Symbol Location and Size Based Prediction.** Circuit diagrams utilise certain conventions such as the placement of voltage sources at the top and representing ground connections at the bottom of diagrams. In addition, the dimensions of a symbol bounding box are a good indicator of the label type. This information is normalised for components in images and is used to train a neural network to predict the symbol labels. The trained neural network is then used to predict node labels based on the same information from the Faster RCNN output on the test set.

**Graph Convolutional Networks.** Both the node embeddings and symbol location and size based label prediction methods do not make use of all graph features available, therefore, a 2 layer GCN is trained to predict the label of a node taking the graph adjacency matrix and node data as input. Two versions of the GCN are tested, in one version, symbol position and size data is included as node data. In a second version, Faster RCNN prediction scores for the labels are also appended to the node data.

## 5 Results

Extraction results for the GraphFix framework are presented under four sections - results from fine-tuning the Faster RCNN module with different approaches, anomaly detection and the improvement in extraction after anomaly detection and finally, the results from other graph refinement techniques. The precision@1 and recall@1 metrics are used to quantify the performance on symbol detection and anomaly detection tasks.

### 5.1 Fine-Tuning of Faster RCNN

The Faster RCNN object detection module provides a good baseline for symbol extraction task. Data augmentation and parameter optimization also help improve the performance even further and a recall@1 of **89.47%** is achieved with a combination of these techniques (see Table 1).

**Table 1.** Results of faster RCNN training on test set

Model	Dataset	Precision	Recall	F-Measure	Max Recall
Faster RCNN	Basic	83.82%	83.16%	83.49%	85.44%
Faster RCNN	Augmented	86.91%	87.37%	87.14%	<b>89.47%</b>

## 5.2 Anomaly Detection

To compare node degree and GCN based anomaly detection, Recall@1 and Precision@1 metrics are calculated by comparing the list of anomaly nodes in the anomaly dataset with the list of anomalies predicted for the two methods for the test set (see Table 2).

These metrics vary from one GCN training to another, therefore the results for the model with the best recall in **3** training attempts are reproduced below. Additionally, the number of variants for a model is decided by increasing the number till no further improvement in recall is observed. In contrast, these metrics remain constant for the node degree comparison model unless the underlying Faster RCNN model is retrained (Table 3).

**Table 2.** Anomaly detection using node degree based methods

Anomaly detection method	Precision@1	Recall@1	F-Measure
Node degree	30.60%	58.95%	40.29%

**Table 3.** Best results for anomaly detection using GCNs

GCN	Node Drop %	Variants	Precision@1	Recall@1	F-Measure
2 layer GCN	20.00%	8	53.49%	27.38%	36.22%
2 layer GCN	25.00%	8	52.00%	30.95%	38.80%
2 layer GCN	33.33%	6	49.02%	30.86%	37.88%
2 layer GCN	50.00%	6	41.82%	27.38%	33.09%
3 layer GCN	25.00%	10	45.90%	33.33%	38.62%
3 layer GCN	30.00%	8	<b>46.03%</b>	<b>34.52%</b>	<b>39.46%</b>
3 layer GCN	33.33%	6	42.42%	33.33%	37.33%
3 layer GCN	40.00%	4	42.42%	33.33%	37.33%

GCN based anomaly detection offers higher precision than the node degree comparison, but the node degree method delivers higher recall. Additionally, some observations can be made about the low F-measure for both models:

- Errors from previous processing steps i.e. Faster RCNN and blob detection for connection detection result in errors both in the training and test datasets for anomaly detection

- The node degree comparison model outputs many false positives, which correspond to issues resulting in the graph structure due to problems other than UFNs such as false positives in Faster RCNN output, incorrect output from blob detection as well as use of infrequent circuit symbols or wire configurations
- Improvements in GCN precision and recall with data augmentation (node drops in variants) leads to the hypothesis that the method can achieve better results with additional training data

Since anomaly detection is carried out to detect UFNs, node degree comparison is chosen for further experiments because of its higher recall. This is because, more search areas increase the chances of detecting UFNs.

### 5.3 Improving Faster RCNN Recall Using Anomaly Detection

Using anomaly predictions from node degree comparison, a fixed image transformation (scaling by a factor of **1.15**) is applied to search areas to identify new object proposals. New proposals overlapping with existing components or without connections to anomalies are dropped and precision@1 and recall@1 metrics for the expanded component list are calculated (see Fig. 5).

The anomaly detection method is applied to multiple Faster RCNN models to observe the consistency in the improvement of results (see Table 4).

**Table 4.** Improved symbol recall with UFN & anomaly detection

Model	Dataset	Faster RCNN results			Results after anomaly detection		
		Precision	Recall	F-Meas	Precision	Recall	F-Meas
Faster RCNN I	Basic	83.82%	83.16%	83.49%	75.09%	86.84%	80.54%
Faster RCNN II	Augmented	85.79%	87.37%	86.57%	79.67%	89.34%	84.23%
Faster RCNN III	Augmented	85.60%	89.21%	87.37%	80.93%	<b>91.05%</b>	85.69%

Anomaly detection based identification of UFNs results in a consistent increase improvement in the recall metrics. A slight drawback is the drop in precision, which is largely a result of rectangular corners and segments of wires being identified as wire connections after image augmentation. Simple domain specific rules can eliminate many of these wrong proposals. However, this is not implemented to keep the proposed framework general.

### 5.4 Graph Refinement

For graph refinement techniques, two parameters are tested - recall@1 for the refinement method itself and the recall of a combined model which adds confidence scores of the refinement method and the Faster RCNN model trained on the basic dataset (see Table 5).

**Table 5.** Recall for graph based label prediction methods. Recall for Faster RCNN model - **83.55%**

Refinement method	Recall	Recall combined with Faster RCNN
Neural network with component position and size	58.95%	80.39%
2 layer GCN	41.05%	81.47%
2 layer GCN with Faster RCNN confidence scores	80.06%	81.84%

## 6 Conclusion

GraphFix successfully demonstrates the use of graph level features to improve symbol and connection identification in circuit diagrams images. The graph level abstraction of the proposed enhancement techniques allows for the framework to be applied to other symbolic engineering document types such as P&IDs, subject to availability of suitable datasets, providing further scope for research in this topic, additionally, the framework potentially supports symbol extraction approaches other than Faster RCNN, which provide symbol class and position information as well as other connection identification methods.

Although GraphFix achieves a high recall on the tested dataset, multiple improvements such as identifying connection leads for components, improving the OCR recognition and identifying improvements to the graph refinement stage can improve the accuracy of the digitization and increase the practical utility of the extraction. Additionally, graph level features can also be extracted from available electronic versions of engineering diagrams (such as circuit netlists), providing for a more ground truth for the graph based refinement models.

## References

1. Agarwal, S., Agrawal, M., Chaudhury, S.: Recognizing electronic circuits to enrich web documents for electronic simulation. In: Lamiroy, B., Dueire Lins, R. (eds.) GREC 2015. LNCS, vol. 9657, pp. 60–74. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-52159-6\\_5](https://doi.org/10.1007/978-3-319-52159-6_5)
2. Bailey, D., Norman, A., Moretti, G., North, P.: Electronic schematic recognition. Massey University, Wellington, New Zealand (1995)
3. Bayer, J., Sinha, A.: Graph-based manipulation rules for piping and instrumentation diagrams (2020)
4. De, P., Mandal, S., Bhowmick, P.: Recognition of electrical symbols in document images using morphology and geometric analysis. In: 2011 International Conference on Image Information Processing, pp. 1–6 (2011)
5. Drapeau, J., Géraud, T., Coustaty, M., Chazalon, J., Burie, J.-C., Eglin, V., Bres, S.: Extraction of ancient map contents using trees of connected components. In: Fornés, A., Lamiroy, B. (eds.) GREC 2017. LNCS, vol. 11009, pp. 115–130. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-02284-6\\_9](https://doi.org/10.1007/978-3-030-02284-6_9)

6. Fu, L., Kara, L.B.: From engineering diagrams to engineering models: visual recognition and applications. *Comput. Aided Des.* **43**(3), 278–292 (2011)
7. Hammond, D.K., Vandergheynst, P., Gribonval, R.: Wavelets on graphs via spectral graph theory. *Appl. Comput. Harmon. Anal.* **30**(2), 129–150 (2011)
8. Héroux, P., Le Bodic, P., Adam, S.: Datasets for the evaluation of substitution-tolerant subgraph isomorphism. In: Lamiroy, B., Ogier, J.-M. (eds.) GREC 2013. LNCS, vol. 8746, pp. 240–251. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44854-0\\_19](https://doi.org/10.1007/978-3-662-44854-0_19)
9. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint [arXiv:1609.02907](https://arxiv.org/abs/1609.02907) (2016)
10. Maier, A.: Graph deep learning — part 1. <https://towardsdatascience.com/graph-deep-learning-part-1-e9652e5c4681>
11. Messmer, B.T., Bunke, H.: Automatic learning and recognition of graphical symbols in engineering drawings. In: Kasturi, R., Tombre, K. (eds.) GREC 1995. LNCS, vol. 1072, pp. 123–134. Springer, Heidelberg (1996). [https://doi.org/10.1007/3-540-61226-2\\_11](https://doi.org/10.1007/3-540-61226-2_11)
12. Moreno-García, C.F., Elyan, E., Jayne, C.: New trends on digitisation of complex engineering drawings. *Neural Comput. Appl.* **31**(6), 1695–1712 (2018). <https://doi.org/10.1007/s00521-018-3583-1>
13. Rabhani, M., Khoshkangini, R., Nagendraswamy, H., Conti, M.: Hand drawn optical circuit recognition. *Procedia Comput. Sci.* **84**, 41–48 (2016). *Proceeding of the Seventh International Conference on Intelligent Human Computer Interaction (IHCI 2015)*. <http://www.sciencedirect.com/science/article/pii/S1877050916300783>
14. Rahul, R., Paliwal, S., Sharma, M., Vig, L.: Automatic information extraction from piping and instrumentation diagrams. arXiv preprint [arXiv:1901.11383](https://arxiv.org/abs/1901.11383) (2019)
15. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**(6), 1137–1149 (2016)
16. Renton, G., Héroux, P., Gaüzère, B., Adam, S.: Graph neural network for symbol detection on document images. In: 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW), vol. 1, pp. 62–67. IEEE (2019)
17. Surikov, I.Y., Nakhatovich, M.A., Belyaev, S.Y., Savchuk, D.A.: Floor plan recognition and vectorization using combination UNet, faster-RCNN, statistical component analysis and ramer-douglas-peucker. In: Chaubey, N., Parikh, S., Amin, K. (eds.) COMS2 2020. CCIS, vol. 1235, pp. 16–28. Springer, Singapore (2020). [https://doi.org/10.1007/978-981-15-6648-6\\_2](https://doi.org/10.1007/978-981-15-6648-6_2)
18. Valveny, E., Delalandre, M., Raveaux, R., Lamiroy, B.: Report on the symbol recognition and spotting contest. In: Kwon, Y.-B., Ogier, J.-M. (eds.) GREC 2011. LNCS, vol. 7423, pp. 198–207. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-36824-0\\_19](https://doi.org/10.1007/978-3-642-36824-0_19)
19. Xiong, X., Choi, B.J.: Comparative analysis of detection algorithms for corner and blob features in image processing. *Int. J. Fuzzy Logic Intell. Syst.* **13**(4), 284–290 (2013)
20. Yu, B.: Automatic understanding of symbol-connected diagrams. In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 2, pp. 803–806. IEEE (1995)
21. Yu, E.S., Cha, J.M., Lee, T., Kim, J., Mun, D.: Features recognition from piping and instrumentation diagrams in image format using a deep learning network. *Energies* **12**(23), 4425 (2019)



22. Yun, D.Y., Seo, S.K., Zahid, U., Lee, C.J.: Deep neural network for automatic image recognition of engineering diagrams. *Appl. Sci.* **10**(11), 4005 (2020)
23. Zhou, X., et al.: East: an efficient and accurate scene text detector. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5551–5560 (2017)