Via Modeling on X-Ray Images of Printed Circuit Boards Through Deep Learning

David Selasi Koblah, Ulbert Botero, Fatemeh Ganji*, Damon Woodard, Domenic Forte
University of Florida, Gainesville, FL, USA
*Worcester Polytechnic Institute, Worcester, MA, USA
(dkoblah, jbot2016)@ufl.edu, fganji@wpi.edu, (dwoodard, dforte)@ece.ufl.edu

***Abstract*—A widely-regarded approach in Printed Circuit Board (PCB) reverse engineering (RE) uses non-destructive X-ray computed tomography (CT) to produce three-dimensional volumes with several slices of data corresponding to multi-layered PCBs. The noise sources specific to X-ray CT and variability from designers make it difficult to acquire the features needed for the RE process. Hence, these X-ray CT images require specialized image processing techniques to examine the various features of a single PCB to later be translated to a readable CAD format. Previously, we presented an approach where the Hough Circle Transform was used for initial feature detection, and then an iterative false positive removal process was developed specifically for detecting vias on PCBs. Its performance was compared to an off-the-shelf application of the Mask Region-based Convolutional Network (M-RCNN). M-RCNN is an excellent deep learning approach that is able to localize and classify numerous objects of different scales within a single image. In this paper, we present a version of M-RCNN that is fine-tuned for via detection. Changes include polygon boundary annotations on the single X-ray images of vias for training and transfer learning to leverage the full potential of the network. We discuss the challenges of detecting vias using deep learning, our working solution, and our experimental procedure. Additionally, we provide a qualitative evaluation of our approach and use quantitative metrics to compare the proposed approach with the previous iterative one [1].**

***Keywords*—printed circuit board; machine learning; x-ray computed tomography; mask region-based convolutional neural network; transfer learning.**

I. Introduction

In October of 2018, Bloomberg published an article entitled "The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies." The story claimed that servers in companies like Apple and Amazon had been compromised by secretly embedded spy chips. Although the companies mentioned in the article refuted its claims, the possible existence of such a threat highlighted the need for more research in the field of hardware security and assurance. Access to hardware designs allows adversaries to not only alter existing designs, but also replicate them. These counterfeit designs penetrate various markets and industries. The country's military supply chain is especially vulnerable to counterfeit products, which affects the safety of service members and mission success. According to a report submitted to Congress by the Department of Defense (DoD) in 2017, the United States of America is struggling to keep up with the global printed circuit board (PCB) market. It notes that 90% of global manufacturing occurs in Asia, with the United States accounting for only 5%. The DoD depends on these foreign-produced PCBs to meet military demands, which raises additional security concerns.

Two issues that are often encountered in the quest for seamless hardware security and assurance frameworks are the time and manual labor involved in reverse engineering (RE). An example, which is especially relevant to this paper, is assessing PCB designs at the final stages of the supply chain. Experts are required to manually examine the devices, with each sample taking a considerable amount of time. Any deviations from the original blueprints must be flagged and investigated. Even with a task force numbering 20, a 5-minute check will yield just above 1900 verified devices during an 8-hour shift. For companies that ship or accept thousands of PCBs per day, that is very unsatisfactory.

It is also worth mentioning the destructiveness that comes along with PCB RE for assurance. Destructive methods involve physically delayering a PCB and imaging it layer-by-layer with a digital camera or optical microscope [1]. More recent innovations have focused on non-destructive options like PCB RE using X-ray Computed Tomography (CT). X-ray CT non-destructively reproduces a 3D point cluster of the PCB sample followed by reconstruction to produce a 3D volumetric model. In [2], the authors combine image-stitching and reconstruction with filtering and segmentation to produce a model that can be examined along all three-dimensional axes. The images across layers in the top-down direction reveal via and trace information for the PCB undergoing examination. Vias are the vertical interconnects between layers for multi-layered PCBs, while traces are horizontal interconnects within layers. Although this framework is non-invasive, noise can impait its effectivness as it affects the quality of vias and traces identified within the layers and restricts the extent of automation [2].

Vias mostly appear as circles in the X-ray images, while traces look like sets of lines running in different directions. Smaller vias and thinner traces are hard to identify in the presence of the noise that accompanies the parameterized X-ray process [2]. To address this issue for vias, we initially proposed a via-focused detection approach that started with Hough Circle Transform and iteratively removed false-positive detections [3]. By incorporating frequency-based filtering, we mitigated the noisy results of the X-ray CT and automated the detection pipeline. We began with vias because they are the simplest category we can model in the RE process. In spite of its remarkable performance on the test set, one drawback is that achieving this accuracy requires a priori information such as via radii range and a number of sizes. The generalized

[1]DISTRIBUTION STATEMENT A. Approved for public release: distribution is unlimited.

alternative takes a longer time to remove false positives during the radial template matching process. This is because the process requires time to learn the characterization of undefined via sizes. The lack of pre-defined sizes makes creating templates more difficult. We overcome this generalization issue in this paper by taking advantage of a deep learning approach that captures feature extraction at varying scales. We are able to identify vias from different PCB designs without altering any parameters. We evaluate our approach by conducting experiments on a custom-designed 6-layer PCB (Test PCB), as well as a commercial-off-the-shelf (COTS) Xilinx Spartan 3 PCB.

In the next section, we discuss the R-CNN family and the metrics commonly applied to evaluate object detection. Section III explains our modifications to the out-of-box Mask R-CNN implementation. We emphasize the steps taken to train and how important they are to tackle such a nuanced problem. Section IV provides quantitative results and discussion. The final section concludes the paper and provides directions for future work.

II. Background

A. Region-based Convolutional Neural Network

The Region-based Convolutional Neural Network (R-CNN) method implements bounding-box object detection by proposing candidate object regions and applying convolutional networks on each region of interest [4], [5]. Faster R-CNN [5] builds on the original R-CNN by using a Region Proposal Network (RPN) to propose candidate object bounding boxes. Its second stage extracts features using Region of Interest pooling (RoIPool) from each candidate box and performs classification and bounding-box regression. For each possible object, it outputs a class label and a bounding box offset. The class label is the name of the object, while the bounding box indicates its location. Based on the framework of Faster R-CNN, Mask R-CNN adds a third branch for predicting an object mask (captures form) in parallel with the existing branches for classification and localization. A block diagram of Mask R-CNN is shown in Fig.1.

B. Transfer Learning

Previously-learned tasks are the very foundation of transfer learning. Essentially, the knowledge gained from solving one machine learning problem can be used to address another one. Therefore, the main motivations and benefits of utilising transfer learning are faster training times, increased accuracy, and eliminating the need for large amounts of data. Pan and Yang [6] use domain, task, and marginal probabilities to describe transfer learning.

Take a domain D, a two-element tuple with feature space χ and marginal probability $P(X)$. X is a sample data point, and the entire domain is represented as

$$D = \{\chi, P(X)\}. \tag{1}$$

Now consider a task T, another two-element tuple of label space γ and predictive function η. The objective function can

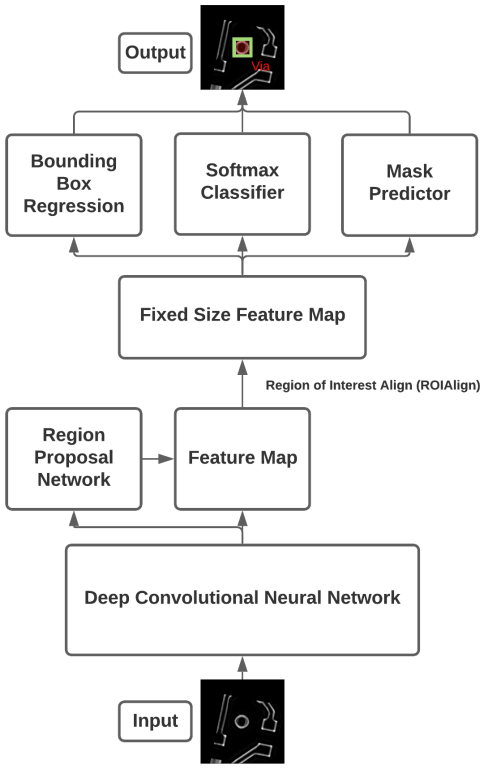

Fig. 1: Overview of Mask R-CNN Framework

be denoted as P(γ| X). Also, Y is a corresponding label point for X. Therefore,

$$T = \{\gamma, P(Y|X)\} = \{\gamma, \eta\}, \tag{2}$$

A source domain can be represented as D_s, source task as T_s, target domain as D_t, and target task as T_t. Hence, the aim of transfer learning is to learn the target conditional probability function η_t or P($Y_t|X_t$) in D_t with information gained from D_s and T_s.

C. Structural Similarity Index

In order to measure the decrease in the quality of an image during processing, the Structural Similarity Index (SSIM) was created. Two visually-identical images from the same shot are compared using this metric. It deduces perceptual differences between images [7]. For two images of a and b with common size NxN, the SSIM score is computed as follows:

$$SSIM(a,b) = \frac{(2\mu_a\mu_b + c_1)(2\sigma_{ab} + c_2)}{(\mu_a^2 + \mu_b^2 + c_1)(\sigma_a^2 + \sigma_b^2 + c_2)} \tag{3}$$

where μ_a and μ_b are the means of a and b respectively. Both σ_a^2 and σ_b^2 represent their individual variances, while σ_{ab} indicates their covariance [7].

D. Intersection Over Union

Intersection over Union (IoU), also known as the Jaccard index, is the most popular evaluation metric for object detection. Two sets of bounding boxes, the ground truth box, and the predicted box, are required to compute the IoU score [8]:

$$IoU = \frac{M_p \cap M_{gt}}{M_p \cup M_{gt}} \tag{4}$$

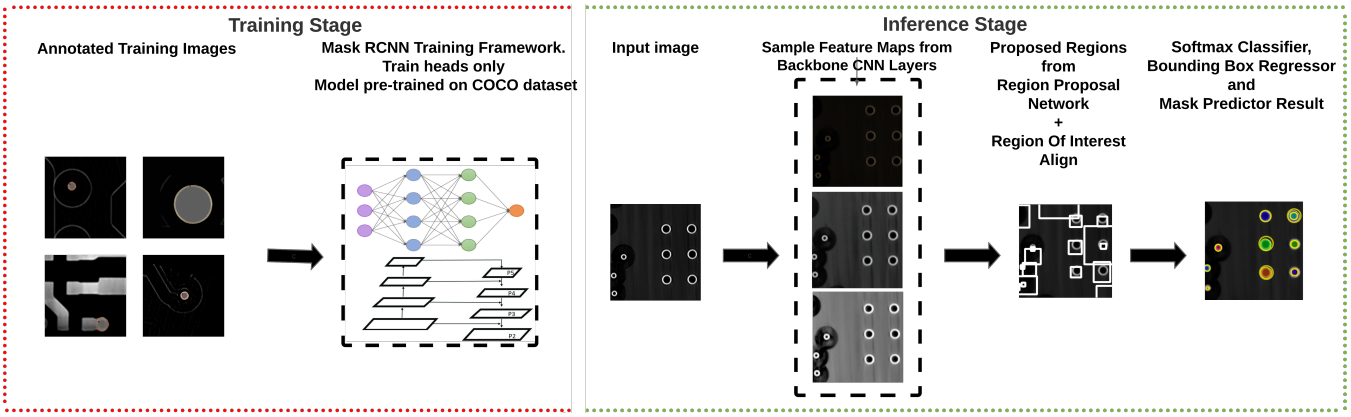

Fig. 2: Visualization of Mask R-CNN Applied to Via Detection

where M_p is the predicted mask and M_{gt} is the ground truth mask. For this task, the IoU is computed for both the foreground and background separately and then averaged to provide a mean score.

E. DICE Coefficient/ F1 Score

The DICE coefficient, also referred to as F1-Score, is computed similarly to IoU, and we once again prefer the mean DICE score result due to class imbalance between foreground and background classes in the segmentation masks [9]. DICE is computed as follows

$$DICE = \frac{2 * M_p \cap M_{gt}}{\text{Total \# of Pixels}} \tag{5}$$

DICE coefficient behaves similarly to IoU, and both are positively correlated with one another. In other words, they are often in accordance with one another when evaluating model performance. However, an important difference is that IoU penalizes instances of misclassified pixels harsher than DICE.

III. Methodology

Deep learning has a significant role to play in in PCB RE process. In the publication by Botero et. al. [3], the unaltered Mask R-CNN implementation is compared to the unsupervised, specialized Hough Circle Detection method proposed in their work. Although it performed poorly by missing many obvious vias, it nevertheless presented an opportunity for further exploration. In this paper, we seek to not only improve the Mask R-CNN method mentioned earlier, but to supersede the unsupervised Hough Circle Detection method. Our framework assures faster via detection time and does not require prior knowledge of via radii sizes for each design. Since it is a feature-based learning solution, we overcome noisy x-ray images by adding noisy samples to the training dataset.

A visual representation of the workflow is shown in Fig. **??**. Like most deep learning problems, the pipeline is divided into two main stages; the training phase and the inference/application phase. The training stage itself is very straightforward. Single via images are annotated for the training process. Once the training step is completed, the model is available for the inference stage. Depending on the size and complexity of the test image, the process takes 1-3 minutes. The details of each part are enumerated in the following subsections.

A. Training Stage

The Mask R-CNN implementation we build upon [10] provides weights pre-trained on the MS COCO [11] dataset. The original dataset is made up of 80 different categories of everyday objects. The pre-trained weights allow us to utilize transfer learning (see Sec. II-B). We adjust only the prediction heads and freeze the main backbone layers for training. This is because the backbone layers already work for feature extraction at different levels. We only require the heads to be able to effectively classify, localize and segment a via object.

1) Annotated Training Images: The annotation style used by the collaborators of the implementation [10] is a single polygon of data-points surrounding the object of interest. During earlier experiments, it was realized that this method significantly under-performed during the via-object inference stage. One reason is the nature of vias on the X-ray CT images. Different X-ray conditions produce vias that do not appear as copper circles. Some open vias appear to be closed on raw x-ray CT images due to the presence of high-Z solder material. Open vias also have image background within their inner circumferences. Hence, we annotate both inner and outer regions of visibly open vias. All these have to be accounted for when the model is learning how to identify a via. More annotation points provide more descriptive points for the model to learn. The VGG Image Annotator (VIA) [12] is the annotation tool used to define the vias in each training image. Each training image contains a single via obtained from the Test board or Xilinx Spartan 3 PCB. They are a randomly chosen set of all via sizes available. Single via images reduce the likelihood of overfitting. They also prevent group localization during the inference stage. Although we test on the same PCBs, it must be emphasized that the single via images are different from the entire PCB X-ray CT images. The test images (see Section IV) are significantly more complex than the training set (see Fig. 2).

2) Backbone CNN Layers (ResNet101 + FPN): The deep convolutional neural network used is a version of ResNet,

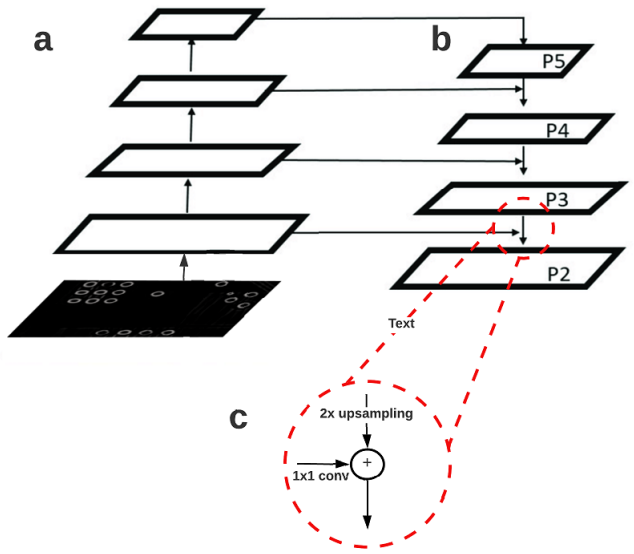

Fig. 3: Feature Pyramid Network a) Bottom-Up Pathway b) Top-Down Pathway c) Lateral Connection

whose core idea is an "identity shortcut connection" that skips one or more layers during training [13]. The ResNet101 model can successively be trained for 1000 different categories of objects. Since only the layers after ResNet are adjusted, the Feature Pyramid Network (FPN) is our primary focus. It generates multi-scale feature maps with essential information. It combines low-resolution, semantically strong features with high-resolution, semantically weak features via a top-down pathway and lateral connections [14]. The low-resolution features are the result of the bottom-up pathway, which is the ResNet101 model in this case. One pyramid level is set for each stage of the network. The output of the last layer of each stage is used as the reference set of feature maps for the top-down pathway. The connection is made laterally. On the other hand, the top-down pathway involves upsampling the higher levels of the pyramid. The feature maps generated are semantically strong. As seen in Fig. 3, the feature maps from the bottom-up pathway (after 1x1 convolution) and the top-down pathway are merged by element-wise addition. Note that the final feature map is generated by a 3x3 convolution on each merged map.

3) Region Proposal Network (RPN) + ROIAlign: The input of this stage is the feature maps from the FPN. Its output is a number of regions of interest (ROIs) that will be examined by a classifier, regressor, and mask branch to eventually check the occurrence of objects. The steps in the RPN step are:

1) Generate anchor boxes.
2) Classify each anchor as foreground or background.
3) Learn shape offsets for anchor boxes to fit them for objects.

Every point in the feature map generated by the backbone is an anchor point. Each anchor point has anchor boxes generated around it. The parameters used to create the boxes are scale and aspect ratio. After anchor box generation, boxes are classified into the foreground or background images. During the training stage, offsets are learned for the foreground boxes to adjust for fitting the objects. The foreground objects are found by calculating IoU between anchors and ground truths, while the background objects are computed using the difference in the coordinates. The difference in the coordinates is calculated and learned as targets by a regressor. Some portion of the foreground and background boxes are considered according to confidence scores. The offsets are applied to those boxes to get the actual ROIs to be processed further. Due to the bounding box refinement step in the RPN, the ROI boxes may have different sizes. This can cause misalignment. ROIAlign prevents this by accurately mapping an ROI from the original image onto the feature map without rounding up to integers. This is achieved using bilinear interpolation.

4) Softmax Classifier, Bounding Box Regressor, and Mask Predictor: The softmax classifier network is deeper and has the capacity to group regions to specific classes. Bounding box regression is similar to the mechanism in the RPN, and its purpose is to further refine the location and size of the bounding box to encapsulate the object. The mask predictor branch is based on the FCN Semantic Segmentation algorithm proposed by Long et al. in 2015 [15]. It takes the positive regions selected by the ROI classifier and generates masks for them. During training, we scale down ground truth masks to 28x28 to compute the loss, and during inferencing we scale up the predicted masks to the size of the ROI bounding box. The multi-task loss function of Mask R-CNN is calculated as follows:

$$L = L_{cls} + L_{box} + L_{mask} \tag{6}$$

where L is the total loss, L_{cls} is the classifier loss, L_{box} is localization loss and L_{mask} is the mask predictor loss.

B. Inference Stage

This is the test of the completely-trained model. It is a sequence of events that ends with bounded, classified and masked object(s). Although Mask R-CNN is reasonably scale-invariant, their features must be recognizable. The original implementation provides a bounding box with the predicted class. The code is refactored to provide a bounding circle for each detected via. A database of 2D coordinates and radii sizes is also generated for each image under analysis.

IV. Results and Discussion

A. Experimental Setup

The pre-trained weights on COCO allow the use of a smaller dataset to facilitate training. The best-forming training set size is found to contain 240 single via images. The images are from X-ray CT results of the 6-layer PCB and the Xilinx Spartan 3 PCB. The 6-layer PCB has physical dimensions of 15mm by 15mm and 3-D referred stack dimensions of 1357×1286×157 slices. The Xilinx Spartan 3 PCB has estimated dimensions 280mm by 210mm and 3-D stack dimensions of 1871×1858×58 slices. All training images were 192×192 pixels. This was to minimize training time and conform to the default dimension requirement of each side divisible by 64 created by the original programmers. As mentioned in the previous section, the VGG Image Annotator (VIA) is the annotation tool used to define the vias in each

Training Parameter	Value
Number of Epochs	30
Steps Per Epoch	100
Batch Size	8
Top Down Pyramid Size	256
RPN Anchor Scales	(8, 16, 32, 64, 128)
RPN Anchor Ratios	(0.5, 1, 2)
Minimum Image Dimension	192
Maximum Image Dimension	192

TABLE I: Main Parameters Used In Matterport Mask R-CNN Implementation

training image. To supplement the existing Mask R-CNN infrastructure, we extend the Config class to modify the training parameters, and the Dataset class to access training images, test images, and annotations for the vias. Like the original annotations used on the COCO dataset, polygons are used. We keep the minimum detection confidence value at 5% to allow all possible vias to be detected. The maximum possible detection value is also increased from 100 to 6000, specifically to accommodate the Spartan Board's 800 via count. Although a greater value may increase the number of false positives, it is advisable to use the largest one possible and eliminate false positives after inference. This will improve the chances of finding marginally-defined vias in preparation for the vast amount of possible via density in a design. The minimum and maximum image dimensions are also changed from 800 and 1024 to 192 and 192 to suit our prepared dataset. Parameters such as the number of epochs and batch size remain at 30 and 8, respectively. The most important parameters are listed in Table I.

We implemented our experiments using Python version 3.8.5. The Mask R-CNN implementation uses Tensorflow and Keras tools. To run the experiments, we utilize 2 GPUs in the HiperGator Research Computing environment at the University of Florida. Each GPU is an NVIDIA GeForce RTX 2080ti with 11GB of dedicated memory.

B. Acccuracy

We measure the accuracy of our design's results using the ground truth set. We hand-label the annotations using the software in [16] for both the Xilinx Spartan PCB and the Test Board layers to accomplish this. The ground truth files are lists of vias with their 2D center coordinates and corresponding radii. This addresses both localization and classification requirements like any standard object detection model. The methods we select to measure accuracy are IoU (Section II-D), DICE coefficient (Section II-E), and SSIM (Section II-C).

Our results using these metrics are stated in Table II. The previous unsupervised method begins by selecting suitable via candidates. After going through pre-processing, the Hough Circle Detector is used to select all possible vias in the image. The two scenarios used for the parameter-based method are prior knowledge of the specific via radii ranges (referred to as "Known" in the table), and a non-specific upper and lower bound value for small, medium, and large ranges (referred to as "General" in the table) [3]. For both sets of instances, there is a false-positive removal step, followed by a concentricity check to eliminate multiple detections of the same via. We compare

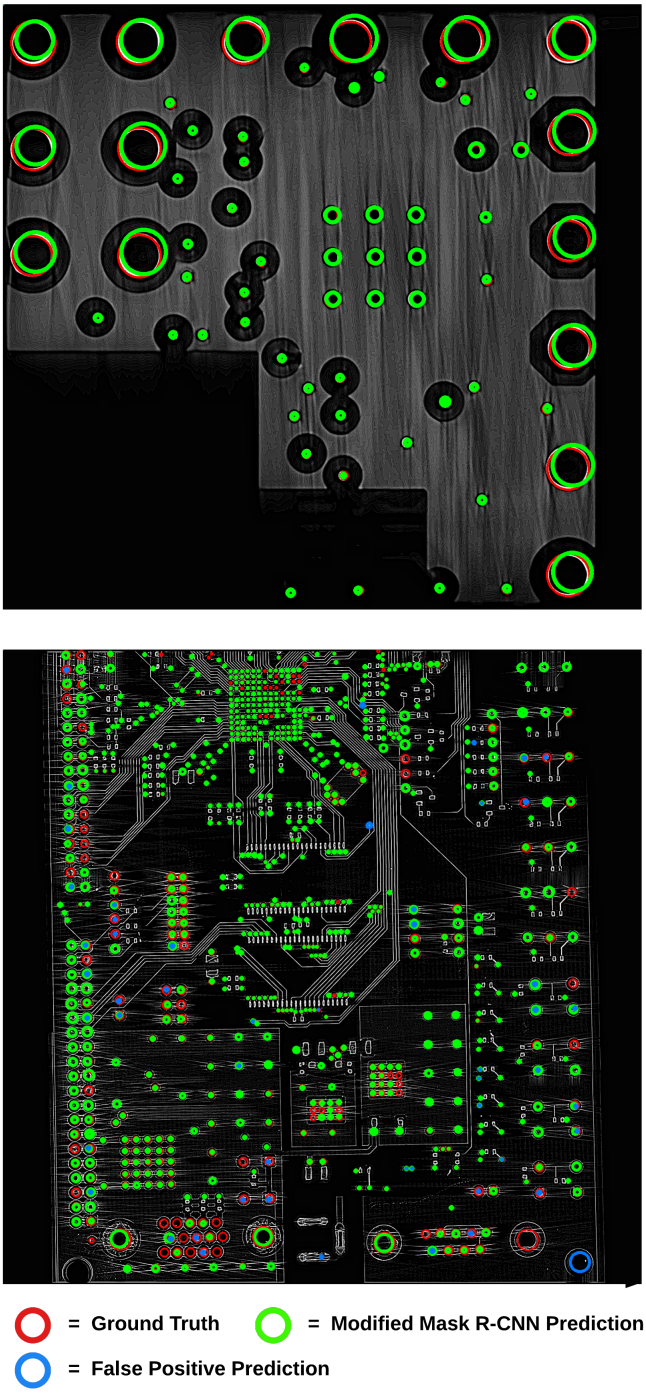

Fig. 4: Sample Results of Modified Mask R-CNN model on Test Board (Top) and Xilinx Spartan 3 Board (Bottom)

our accuracy results to both known and generalized scenarios. We also add the Mask R-CNN implementation used as a benchmark in the publication. All layer images used in the experiments presented in the previous paper are also reproduced. The framework has minor issues with some vias across both sets of X-ray CT images. Vias are classified reasonably well on the Test board, but the localization is skewed, as seen in Fig. 2. This is one issue that must be addressed in the next version

Layer	IoU				DICE				SSIM			
	Known	General	OMRCNN	MMRCNN	Known	General	OMRCNN	MMRCNN	Known	General	OMRCNN	MMRCNN
T-0	.8623	.8521	.4872	.6313	.9209	.9140	.4935	.7126	.9732	.9716	0	.9665
T-1	.8860	.8722	.5816	.6579	.9362	.9274	.6533	.7437	.9734	.9715	.9567	.9690
T-2	.8832	.8407	.5831	.7202	.9345	.9065	.6550	.8082	.9726	.9691	.9548	.9711
T-3	.8800	.8495	.6591	.7225	.9325	.9124	.7484	.8103	.9724	.9698	.9597	.9714
T-4	.8863	.8579	.5835	.7295	.9364	.9182	.6561	.8168	.9735	.9695	.9569	.9718
T-5	.8620	.8253	.4875	.7254	.9206	.8955	.4937	.8130	.9728	.9674	0	.9715
S-0	.7190	.6947	.5062	.6574	.8108	.7891	.5865	.7437	.9477	.9410	.8694	.9431
S-1	.7361	.7052	.5382	.6809	.8261	.7992	.5753	.7691	.9487	.9403	.8912	.9497
S-2	.7342	.6976	.4903	.6606	.8244	.7921	.5380	.7472	.9498	.9411	.7815	.9452

TABLE II: Comparing accuracy scores for the unsupervised Hough Circle Detection method (previously -known radii sizes versus generalized radii sizes for each layer) against the out-of-box Mask R-CNN (OMRCNN) presented in [3]. The final column under each header shows the results of our modified version (MMRCNN). Test board layer is marked with T, while Spartan board layers are S.

of the tool. It is prevalent in the largest-sized vias. During the detection pipeline, the RPN must generate initial bounding boxes that contain the required objects. Several boxes based on the anchors sizes controlled by the entered scales and ratios (as in Table. I) are generated for each proposed object, and non-maximum suppression eliminates the surplus. There is no issue with the sizes, but the localization itself is not accurate. In Table II, IoU results are lower due to this disparity. There are also missed and false detections among layers of the Test board. One problem we faced is the accurate extraction of features, especially in the presence of noise. Features on a populous image may be missed or incorrectly identified, and this can be seen in our results. The Xilinx Spartan 3 PCB is significantly more challenging, and the obstacles faced with the Test PCB are multiplied many times over. Some vias look like pads and contacts. Although we also test on filtered versions of the data, the smaller sizes increase the likelihood of false-positive and -negative detection (See Table III). Like the unsupervised Hough Circle detection method, certain pre-processing techniques may help with this. One option to consider is the parameter-free edge segment detector (EDPF). It is able to make required edges more conspicuous, especially in the presence of noise [17]. This is vital for identifying via shapes.

One other drawback of the deep learning approach is the time factor. It is well-known that training deep learning models takes time, but the inference step leaves much to ponder over. The average time inference runtime on a Test Board layer is 2 minutes, whilst a Spartan layer, with 12 times the number of vias on the Test Board, takes 5 minutes. In a future implementation, a method that allows early exiting when the required features are identified will reduce runtime considerably.

Layer	False Negative Count	False Positive Rate
Test 1	10	0.0156%
Test 2	8	0.0156%
Test 3	0	0.0156%
Test 4	1	0.0156%
Test 5	0	0.0156%
Test 6	0	0.0156%
Spartan 1	66	0.0649%
Spartan 2	171	0.2500%
Spartan 3	219	0.0858%

TABLE III: False positive rate and false negative (missed detection) count for each layer

V. Conclusion

We present an application of deep learning to PCB via detection in X-ray CT images. As a vital step in the PCB reverse engineering process, an automated mechanism provides the foundation for solving other detection problems. The addition of refined pre-processing techniques and optimization of the pipeline will make the current design more potent. After achieving sufficient potency with the current method, we can apply the framework to other parts of the PCB design, such as traces.

References

[1] J. Grand, "Printed circuit board deconstruction techniques," in *USENIX WOOT*, San Diego, CA, 2014.

[2] N. Asadizanjani, M. Tehranipoor, and D. Forte, "PCB Reverse Engineering Using Nondestructive X-ray Tomography and Advanced Image Processing," *IEEE Trans Compon Packaging Manuf Technol*, vol. 7, no. 2, pp. 292–299, 2017.

[3] U. J. Botero, D. Koblah, D. E. Capecci, F. Ganji, N. Asadizanjani, D. L. Woodard, and D. Forte, "Automated via detection for pcb reverse engineering," in *ISTFA*. ASM International, 2020.

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," Tech. Rep. [Online]. Available: https://github.com/

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Tech. Rep., 2015. [Online]. Available: https://github.com/

[6] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans Knowl Data Eng*, vol. 22, no. 10, pp. 1345–1359, 2010.

[7] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multi-scale structural similarity for image quality assessment," in *Asilomar Conference on Signals, Systems & Computers*, vol. 2, 2003, pp. 1398–1402.

[8] D. M. W, "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation," *J. Mach. Learn. Technol.*, vol. 2, no. 1, pp. 37–63, 2011.

[9] F. Ge, S. Wang, and T. Liu, "New benchmark for image segmentation evaluation," *J. Electron. Imaging*, vol. 16, no. 3, p. 033011, 2007.

[10] W. Abdulla, "Mask R-CNN for object detection and instance segmentation on keras and tensorflow," 2017. [Online]. Available: https://github.com/matterport/Mask_RCNN

[11] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*. Springer, 2014, pp. 740–755.

[12] A. Dutta, A. Gupta, and A. Zissermann, "VGG image annotator (VIA)," http://www.robots.ox.ac.uk/ vgg/software/via/, 2016.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[14] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016.

[15] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014.

[16] A. Dutta and A. Zisserman, "The VIA Annotation Software for Images, Audio and Video," in *ACM Multimedia*. ACM, 10 2019, pp. 2276–2279.

[17] C. Akinlar and C. Topal, "Edpf: a real-time parameter-free edge segment detector with a false detection control," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 26, no. 01, p. 1255002, 2012.