PRINTED CIRCUIT BOARD (PCB) DEFECT DETECTION AND CLASSIFICATION

A PREPRINT

Amirhossein Nazeri*
Dept. of Automotive Engineering
Clemson University
anazeri@clemson.edu

Chirag Mutha
Dept. of Automotive Engineering
Clemson University
cmutha@clemson.edu

Haidar Alsalih
Dept. of Automotive Engineering
Clemson University
halsali@clemson.edu

August 22, 2022

ABSTRACT

Printed Circuit Board (PCB) manufacturing and testing is a topic that is gaining interest due to the high demand for consumer electronics.This high demand has therefore lead to rapid production which lead to increase amount of defects in the PCB. These defects range from excess copper to open and short circuits which would use unnecessary material and sometimes render the PCB useless. With the high demand, manual PCB inspections based on statistically timed intervals are becoming more and more impossible to keep up with the demand and therefore more automatic methods are needed. Also, automated visual inspection is a popular way of detecting many kinds of defects in PCB without intervening the manufacturing process. In this paper, we propose an optimal method for automatic anomaly detection of PCBs where a U-NET performs a semantic segmentation and localizes individual defects. This method would not only notify the user of a defect, but also classify the type of defect by labeling the pixel of the detected defect. The proposed method is validated by comparing the predicted images with the ground truth images. We have added low weight based pruning approach to tackle the low speed, high complexity and huge size of conventional deep neural networks.

Keywords U-NET · Deep Learning · Object Segmentation · PCB Defects

1 Introduction

1.1 Background

The fast development of consumer electrical gadgets has drawn more and more attention to PCB fabrication. It is necessary to use a PCB because it acts as a support and an electrical connection between various electronic components. Conductive pathways, signal traces, or tracks etched from copper sheets, laminated onto a non-conductive substrate, are used to achieve these electrical connections. Adding these to the hundreds of components and thousands of solder connections makes the PCBs complex in designing and manufacturing. Because of all these complexities there is a possibility of defects in PCB manufacturing in these connections eur [2020] which cannot be neglected as many of the applications which uses PCBs have lives at stake. For instance, PCBs are mostly used in almost all electronic devices for instance consumer electronics, medical devices, industrial equipments, aerospace industry, etc.

To overcome these defects, various kinds of inspection methods are used depending on the complexities of a PCB. In general, manual visual inspection is one of the most expensive aspects of PCB production, and it always has the potential to introduce human error in detecting defects.

Another method is non-contact method for detection which is gaining interest. Recent research, such as Ray and Mukherjee [2015], Indera Putera and Ibrahim [2010], proposes processing both a defect-free template and a defective

*All authors equally contributed.

tested image to localize and classify defects on the tested image. Non-contact PCB defect detection tests include Automated Optical Inspection (AOI) and X-ray Inspection (AXI). However, AOI is limited to "line of sight," i.e., unable to inspect hidden connections underneath Ball Grid Arrays (BGAs) and other packages. Similarly, AXI can be used only in PCBs having high board density as this method requires high initial investment. This initial high investment risk has created a way to introduce the use of Artificial Intelligence in the PCB defect detection. A sample PCB is shown in Fig. 1.

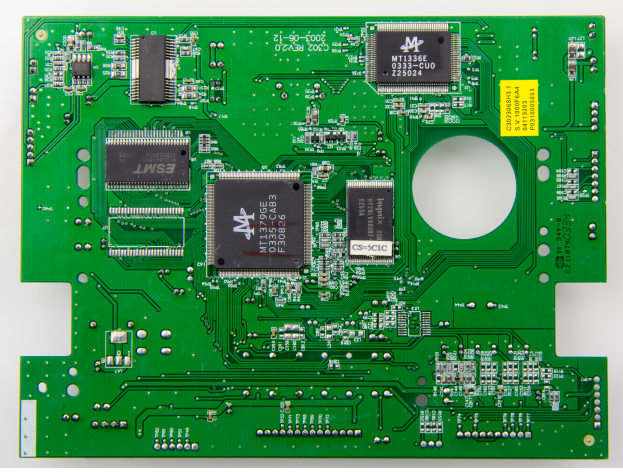

Figure 1: Sample PCB

1.2 Problem Description

To detect all the Deep PCB defects using Semantic Segmentation (U-NET Network) and classify them into one of the six defect types: mousebite, spur, pin-hole and spurious copper. The input template and the output manufactured PCB are both shown in Fig. 3.

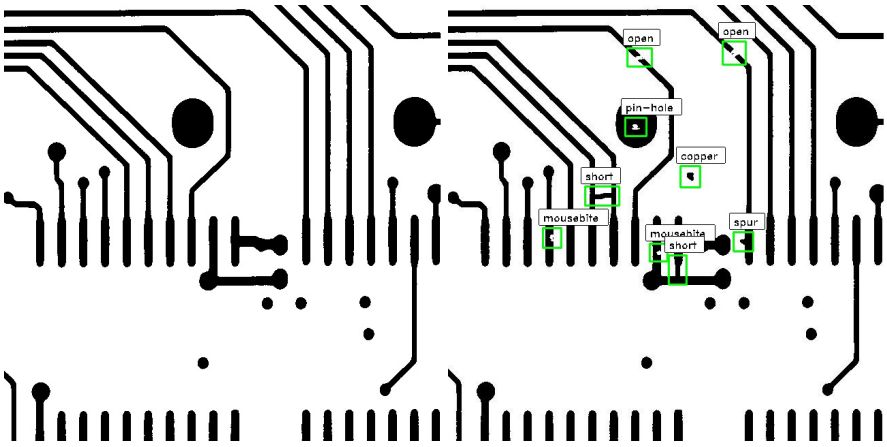

Figure 2: Template vs. Defect PCBs Tang et al. [2019]

1.3 Existing Methods

Authors in Indera Putera and Ibrahim [2010] proposed a hybrid algorithm to detect defects of PCBs based upon morphological segmentation and an image processing technique. Authors in Londe and Chavan [2014] presented an image subtraction method along with a KNN classifier. They have used the MATLAB toolbox to detect and classify 14 different types of defects, namely, Breakout, Pin Hole, Open Circuit, Mousebite, Missing hole, Spur, Short Circuit, and so on. However, they have not reported the performance of learning and execution. Besides the template is so simple and algorithm might fail due to more complicated images. Authors in Prathiba et al. [2017] developed a data-mining scheme

to detect PCBs detects. Their proposed method falls into three main stages (i) data pre-processing (ii) feature selection and reduction, and (iii) Classification. They have also implemented a genetic algorithm (GA) technique to enhance the features selection. Then work is done in MATLAB toolbox that might not be of great interest for many industrial implementations. Nevertheless, the main drawback of all these works, so far, is that they may fail because of defect pattern complexity, an unwanted offset between the ground truth image and the tested image, and more importantly the kernel size mismatch, along with image resolution failure. In recent years, deep neural networks have drawn researchers' attention due to their inherent computational power. In Wang et al. [2017] authors have implemented a Convolutional Neural Network (CNN) algorithm to detect defects in the product quality control section of a manufacturing line. The proposed CNN is made of 11-layer of neurons that accept input images with the size of 128×128 pixels. Fast R-CNN is another promising method to classify different objects that has reduced the running time for learning and detection Ren et al. [2017a].
As mentioned earlier, the biggest problem in defect detection algorithms is the size mismatch between truth ground and tested images during the learning and test process. Authors in Lin et al. [2017] developed a feature pyramid network to tackle this problem by scaling up and down the features of various images having different resolution. Authors Tang et al. [2019] have produced a large-scale dataset of images of the defective and the template PCBs with specifying the types of defects and annotation of their position. Then they proposed a new module named Group Pyramid Pooling (GPP) to take care of images with different sizes and scales. They also evaluated a few different neural networks like SSD Liu et al. [2016], YOLO Redmon et al. [2016], and Faster R-CNN Ren et al. [2017b], on their dataset to validate the strength of their method. Although the methods like GPP and R-CNN are faster than traditional CNN but their performances are not comparable with CNN. In this paper we try to implement Pruning technique, that has shown its strength in performance enhancement, combined with a few CNN architectures like YOLO to improve the computational performance greater than GPP and other existing architectures.

2 The Proposed Network Architecture

2.1 U-Net model

Semantic Segmentation is a part of Computer Vision which also deals with gaining high-level of understanding from digital images and videos. To gain these high-level understanding from images and videos there are various levels of granularity which a computer has to pass from. The granularity of understanding starts from coarse to fine grained understanding. To understand Semantic Segmentation we need to know some pre-processing done before reaching semantic segmentation.

Image Classification - First level is image classification which is the fundamental building block in Computer Vision. A discrete label is expected as an output for the main object in the image.

Localization of classified image - In localization we expect the computer to locate the object along with the label in an image. The localization is implemented using a bounding box across the labeled object. (Note - Image classification and localization have only one object per image)

Object detection - In object detection the image is free from constraint of having one object per image. Object detection can contain multiple objects in one image and localization is done on all these objects in this step.

Semantic Segmentation - It is a pixel level image classification meaning to label each pixel of an image with a corresponding class of what is being represented. This labelling of each pixel is done because in semantic segmentation we predict every pixel in the image making it a dense prediction. The output is a high resolution image with each pixel classifying a particular class.

Instance Segmentation - In instance segmentation the computer is expected to classify the each instance of a class separately i.e. classify the object with pixel level classification.

From the below image of mini-U-NET architecture which is in the form of 'U' is the reason behinds it's name. The architecture is divide into two paths/steps. First path is the down sampling (encoding step on left side) which is used to capture the contect in the image. The encoding step consists of stacks of convolutional layers followed by max pooling. The second path is the up-sampling (decoding step on right side) which is used to apply localization using transposed convolutions. Hence, this make the U-NET an end to end fully convolutional network i.e. it contains only convolutional layers and does not have any dense layers because of which the network can accept image of any size.

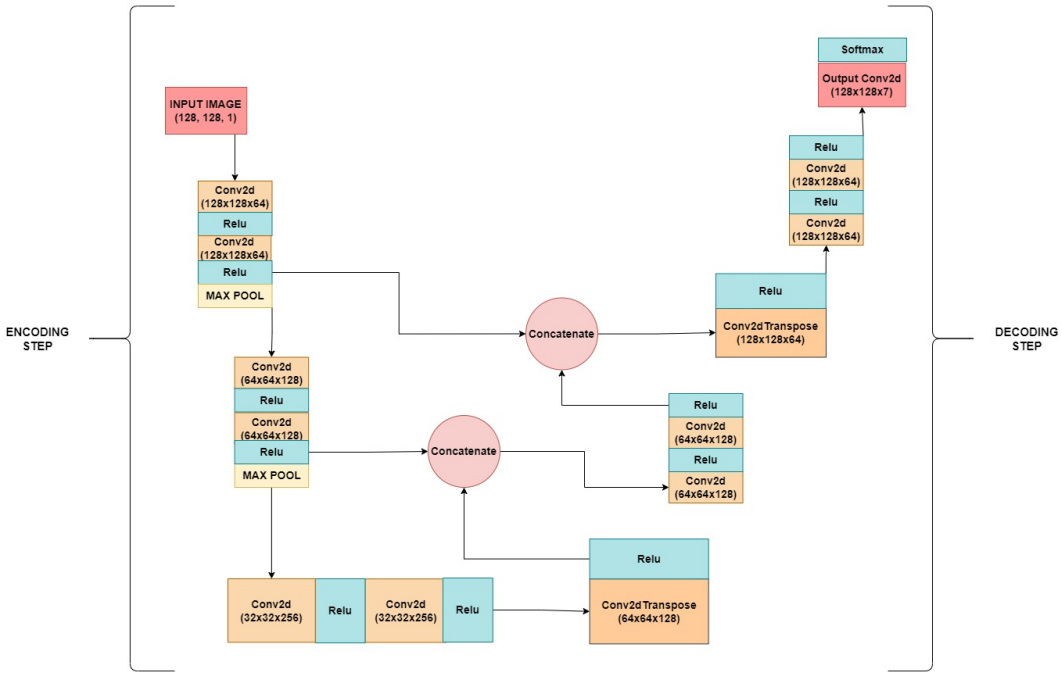

Figure 3: U-NET Model Architecture

We start the network by providing an input image of some size/resolution we want. Then we apply two consecutive convlutional layers and then max pooling after the second convolutional layer. This is the encoding step where the image gradually reduces while the depth gradually increases starting from 128x128x1 to 32x32x256. In simple words the encoding steps means the networks learns the "What" information in the image while it looses the "Where" information.

Then we apply transposed convolutional layer along with regular convolutional layers starting the decoding step. In the decoding step the image size gradually increases and the depth gradually decreases starting from 32x32x256 to 128x128x7. The "7" at last denotes "6 classes" and "7 is background" summing up to "7". The decoder recovers the "Where" information lost while encoding by gradually applying decoding.

To better locate and get the precise locations at every step of the decoder we use skip connections by concatenating the output of the transposed convolutional layers with the feature maps from encoder at the same level (just before max pooling). After every concatenation we apply two consecutive regular connvolutions to help the model learn to assemble a more precise output. All these steps gives the architecture of the network a symmetric U-shape.

High-level relationship representation of network -

$$\text{Input}(128\text{x}128\text{x}1) \Rightarrow \text{Encoder} \Rightarrow (32\text{x}32\text{x}256) \Rightarrow \text{Decoder} \Rightarrow \text{Output}(128\text{x}128\text{x}7)$$

2.2 Pruning Method

Although classical deep neural networks offer high accuracy and low loss, they mainly suffer from low training speed and high model complexity that make them inefficient for real-world applications where the commercial devices can handle low computational burden. Recently, Pruning has shown to be an effective technique to shrink the conventional Neural Network and reduce the computational costs. Deep learning architectures like DNN and CNN in combination with the Pruning technique might be a suitable choice to enhance the learning speed of a neural network while maintaining high accuracy and reducing the overal model size. Pruning removes the unnecessary and trivial elements of an artificial neural network and can enhance the overall efficiency and reduce the computation cost of the neural network. This is of great importance to researchers and companies because it will reduce computational time and therefore reducing costs, particularly for those companies that use online virtual machines like Google Cloud or Amazon Web Service(AWG) GPUs.

There are two main methods in Neural Network Pruning (NNP) that remove either the unnecessary neurons or unnecessary weights from a trained model. The different ways to prune a neural network are - Firstly, one can prune

weights by setting them to zero while keeping the architecture same, which will lower the number of parameters in the mode making the network sparse. In the second method, one can remove trivial nodes from the network, which would change the network architecture while aiming to keep the accuracy of the initial larger network. Fig. 4 flowchart of Pruning implementation in a neural network.

In this paper we employed the *prune_low_magnitude* from Keras library to implement the low-weight pruning. We can apply *prune_low_magnitude* to either whole model at once or perform it layer-wise.The *prune_low_magnitude* method is to set the near-zero weight values of each layer to zero after each iteration in the training. we fine-tuning the model, by retraining the model after pruning to restore accuracy. The pruning parameters are initial sparsity, final sparsity, begin and end steps, and frequency that are defined in the pruning scheduler. We used Polynomial Decay function to fine-tune from sparsity of 50% to 90%. Figure 4 demonstrates the pruning procedure along with the parameters fine-tuning.

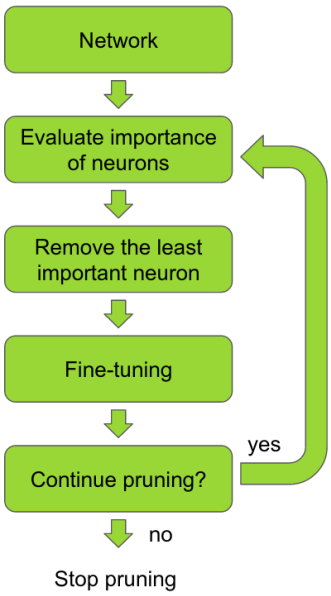

Figure 4: Pruning implementation

3 Training the Model

3.1 Pre-Processing

The Dataset that is provided consists of template and defected images. This Dataset was provided in part to introduce a new defect detection algorithm Tang et al. [2019]. In this paper, the authors matched the positions of the template and defected pictures. With this they were able to subtract the two images. Subtracting the defected and template images results in a mostly zero-filled image with a few white pixels indicating the occurrence of a defect. The total Dataset consisted of 1000 training images (alongside it's templates), 1000 training text files containing the position and type of defect, 500 testing images (alongside it's templates), and 500 text files containing its corresponding defect type and position. Due to some system limitations, we decided to downscale the images from $640X640$ pixels to $128X128$ pixels. Then, the images were only limited to black and white since some images mistakenly were captured with RGB. The images were then scaled by dividing them by the total number of possible output, 255. Making the pixel number range between 0 and 1. The final size for each image became $128X128X1$.

As for the labels, we had to choose between one-hot encoding and defect label representation for every pixel. We chose the latter due to memory efficiency concerns. The final size of the labels became $128X128X1$, where every pixel is in the range from 0 being a background (no defect) to 1-6 defect types. With our approach, we decided that our model should be able to process a new image without any templates to compare it to, and so no logical subtraction preprocessing was done.

3.2 Experiments

The training was conducted on an NVIDIA RTX 3080 GPU. Each batch contained a total of 16 images, we used an ADAM optimizer with a learning rate of 1×10^{-4}, 300 epochs, best model saving (in terms of validation loss), and a training time of 1 hour (compared to the original Dataset paper's 0.5 day training). The loss function is set as *SparseCategoricalCrossentropy*. The learning is stopped and the optimal model is saved before the over-fitting occurs. The saved pruned and unpruned models sizes are 2.4 MB and 6.3 MB, respectively that indicates the pruned model is approximately three times smaller than the original size.

As for the recorded metrics, we chose to record the precision, recall, and F1 scores alongside Intersection over Union.

4 Results

We can see an example of our model in Fig. 5. In this example, we can see that our method has an excellent job classifying different types of defects at a single time. Here, we were able to detect spurs, open circuit, short circuit, copper, and missing hole correctly.

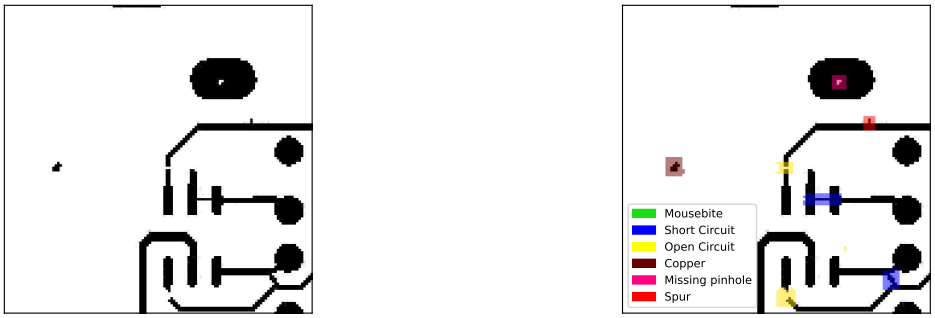

Figure 5: Example 1: U-NET Prediction

As shown in a Fig. 6, our model can do a fairly good job detecting abnormalities such as extra copper in the form of splashes or spurs. However, it tends to ignore a certain size of large missing pinholes. A solution of this problem would be to increase the resolution of these input images to have it detect rough edges: designed pinholes are usually fully circular while defects have a rough edge.

Figure 6: Example 1: U-NET Prediction

Another example is shown in Fig. 7. Our model detected open and short circuits well. While not detected spurs that look like short circuits. Overall, these results might be overshadowed by the random detected missing pinhole in the empty white space.

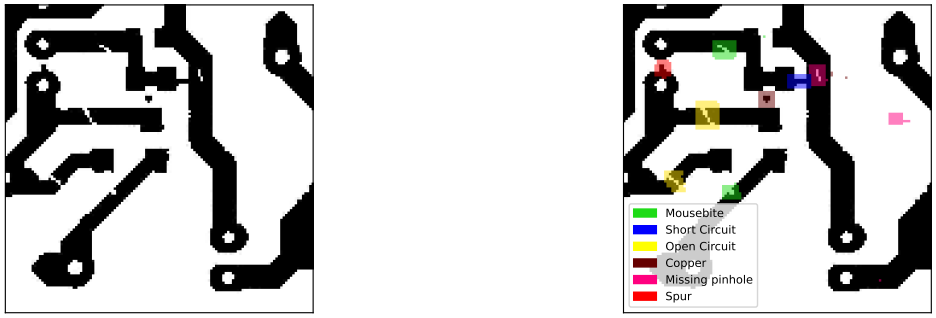

Figure 7: Example 2: U-NET Prediction

Our model has problems classifying missing pinholes as our model detects missing pinholes in an empty space while ignoring pinholes above a certain size.

The training and validation losses are depicted with respective to 225 epochs in Fig. 4 and 4, respectively.

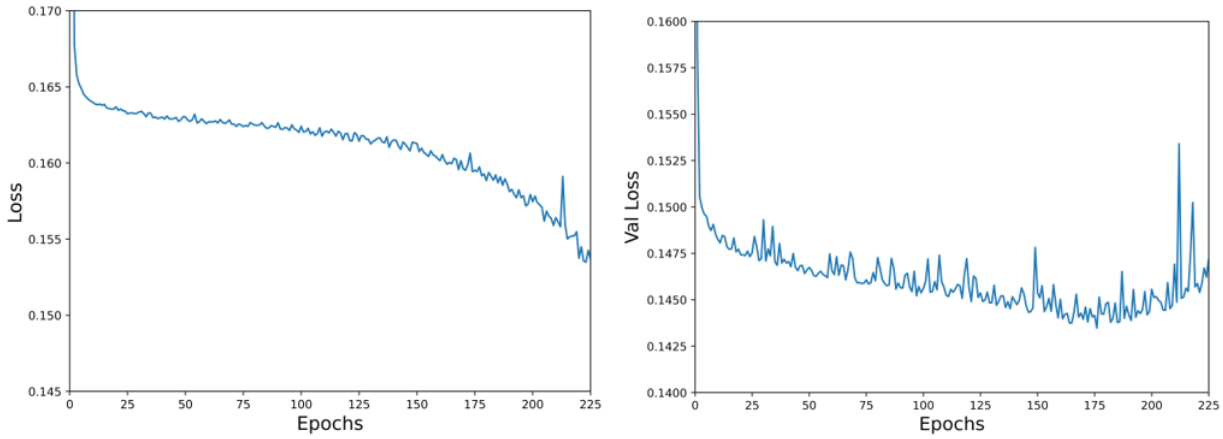

Figure 8: Model Losses

Listed in table 1, we can view our U-NET approach when compared to other methods. As seen, we perform slightly worse than GPP Tang et al. [2019], and about the same as the SSD-FPN Liu et al. [2016].

Table 1: Comparison between the current benchmarks and the proposed work

Method	Precision	Recall	F-Mean
GPP Tang et al. [2019]	98.2	98.1	98.2
SSD-FPN Liu et al. [2016]	94.9	96.8	95.8
Our U-NET	95.3	96.7	96.0

5 Conclusion

In this paper, we proposed a U-NET Neural Network architecture based upon Semantic Segmentation approach to classify six types of PCB defects from DeepPCB dataset. The dataset includes 1500 images containing the six common types of PCB defects. To increase the training speed and reduce the model size we applied a layer-by-layer low weight-based pruning method to our model. This helps us to reduce the size of our model by 3 times compared to its original size. The proposed method is validated by comparing the predicted images with the ground truth images. We

compared our model precision, recall and f-mean score with other models and found out that our model's performance is as good as other models.

References

What is a pcb or printed circuit board?, Jun 2020. URL `https://www.eurocircuits.com/pcb-printed-circuit-board/`.

Swagata Ray and Joydeep Mukherjee. A hybrid approach for detection and classification of the defects on printed circuit board. *International Journal of Computer Applications*, 121:42–48, 07 2015. doi:10.5120/21595-4691.

S.H Indera Putera and Z. Ibrahim. Printed circuit board defect detection using mathematical morphology and matlab image processing tools. In *2010 2nd International Conference on Education Technology and Computer*, volume 5, pages V5–359–V5–363, 2010. doi:10.1109/ICETC.2010.5530052.

Sanli Tang, Fan He, Xiaolin Huang, and Jie Yang. Online pcb defect detector on a new pcb defect dataset, 02 2019.

Prachi P Londe and S A Chavan. Automatic pcb defects detection and classification using matlab. *International Journal of Current Engineering and Technology*, 4(3):2119–2123, Jun 2014.

Vaddin Prathiba, Prof. M. Nagendra, and Prof. M Hanumantappa. Pcb defect detection using genetic algorithm based data mining approach. 2017.

Tian Wang, Yang Chen, Meina Qiao, and Hichem Snoussi. A fast and robust convolutional neural network-based defect detection model in product quality control. *The International Journal of Advanced Manufacturing Technology*, 94 (9-12):3465–3471, 2017. doi:10.1007/s00170-017-0882-0.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017a. doi:10.1109/tpami.2016.2577031.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. doi:10.1109/CVPR.2017.106.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46448-0.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. doi:10.1109/cvpr.2016.91.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017b. doi:10.1109/tpami.2016.2577031.