Image Based Anomaly Detection - PCB Defect Localization and Identification

Introduction

Printable circuit boards (PCB) are fundamental components to many electronic devices where the quality of the device can meaningfully impact performance. With increasing PCB component density and complexity, manual inspection for defects becomes more and more inaccurate. This project focuses on using image processing and machine learning to automatically detect and classify defects in PCBs with the goal of being more accurate than manual inspection.

This project can be split into two core aspects, image pre-processing and localization where PCB images are standardized and split into features for classification, and classification where PCB features are actually identified by a specific defect type.

Related (prior) art

Detection of PCB Defects Via YOLO CNN Architecture
One paper makes use of the YOLO architecture in detecting PCB defects as YOLO can classify more than one object per image, which is critical to this problem as multiple defects can be found on a single PCB. With 11,000 images and an architecture with 24 convolutional layers and 2 fully connected layers, this paper achieves a classification accuracy of 98.82% via YOLO.

This paper makes use of a dataset with data collected from an AOI machine, with labeled data from QA engineers. YOLO works by predicting several bounding boxes with class probabilities using a CNN, which handles both the localization and classification aspect of this problem. (No specific cropping for features or pure image processing is used for localization of defects/features)

Via YOLO based architectures, both localization and classification can be accomplished with a high accuracy. As an expansion, there are many algorithms that generally outperform YOLO, such as fast RCNN as it generally has less localization errors than YOLO.

HRIPCB: a challenging dataset for PCB defects detection and classification

Another paper makes use of a two-stage approach in detecting PCB defects by utilizing extensive preprocessing in conjunction with a morphological operator for defect localization and a CNN for classification. The preprocessing steps behind this detector are registration and binarization. Registration is the process of identifying parts of the image; in this case it would be identifying the components of the PCB board such as the soutering joints and the wires between components. In this implementation, registration was accomplished by the usage of an algorithm called speeded up robust features (SURF) which extracts features from the test image and template image. The features will then be matched up so that a transformation can be determined to rotate and shift the test image to the same orientation as the template image. Registration is then followed by binarization which utilizes an "adaptive threshold segmentation algorithm" that determines the values needed to place a pixel as light or dark based on the luminance of a small region surrounding the pixel. The purpose behind binarization is to decompose the image into a simpler format which would outline the geometry of the PCB board as we are uninterested in the coloring or parts of the image not relevant to the PCB (like the background).

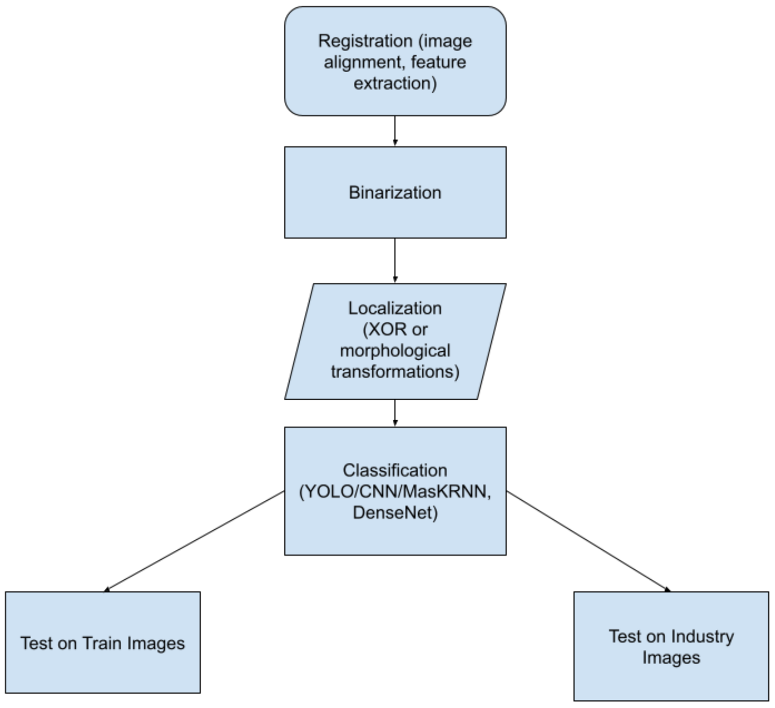
The next step of the architecture is the localization of defects performing an XOR on the test image and template. This step ensures that any features on the test image that are not present in the undefective template are only present. However, since the XOR result is typically noisy, additional median filtering and mathematical morphological processing are utilized.

The actual CNN classification that is then performed utilizes an architecture-style inspired by *Densenet* where a shortcut is present that connects later layers to earlier layers. The purpose of this is to prevent the gradient from approaching from the starting location again as is often characterized in deep neural networks. The architecture has two "blocks" with six convolutional layers that take all previous outputs of the block as an input. In between the two blocks there is a convolution and pooling layer to half the feature map, in addition to there being a convolution and pooling layer before the first block. The output of the last block will then go to a linear layer that classifies the defect region being examined.

The result of this architecture demonstrates it is highly effective on the synthetic PCB defect dataset it was trained and tested on. On the test data, the localisation accuracy was 0% while the classification accuracy was 97.74%. Additionally, the time required for the algorithm to analyze a single PCB image was 0.9899s with registration by far being the slowest step in the algorithm.

Project scope and major ideas/functionalities

The purpose being this project is to create an algorithm for the localization and classification of PCB defects. Unlike previous years, the primary objective behind this project is to build an algorithm that works accurately with industry-pictures of a PCB instead of synthetically altered images. The difference is that the synthetic PCB images are two dimensional and do not vary much in lighting and have a high picture quality. The industry-pictures of a PCB board face several challenges not present in these synthetic images: the three dimensional aspect introduces complexities such as lighting and reflection within the images, the low picture quality introduces a greater level of noise within the data, and less amount of data for the algorithm to be trained on compared to the synthetic PCB dataset. The algorithm for this year's project will utilize the ideas of preprocessing, localization, and classification that were present in previous projects but must modify these ideas to make them compatible with the challenges present when utilizing industrial images of a PCB. The goal of this project is to ensure the same accuracy and efficiency that was present in previous projects for synthetic PCB images but applied to actual industrial PCB images.

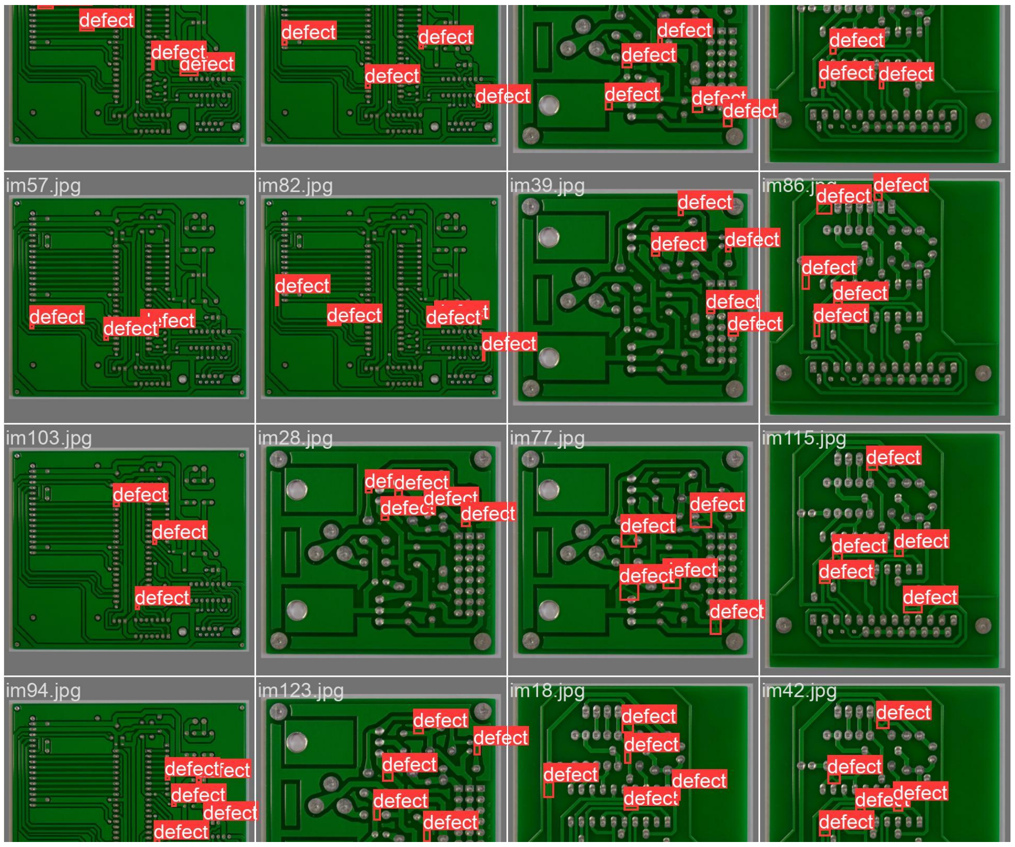

Registration (image alignment, feature extraction)
Binarization
Localization (XOR or morphological transformations)
Classification (YOLO/CNN/MasKRNN, DenseNet)
Test on Train Images
Test on Industry Images

Main approaches and data sources etc.

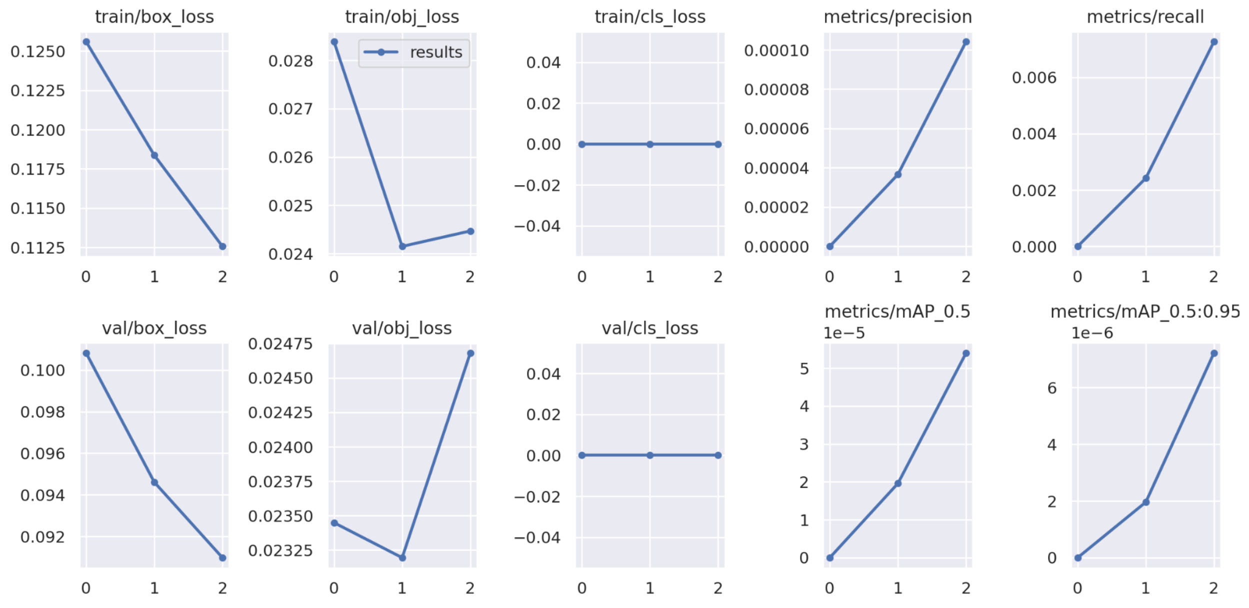
Approach 1: Semantic Segmentation for Localization + Classification via YOLO + Style Transfer

YOLO Training on Kaggle
With this approach, we use an algorithm capable of drawing bounding boxes with a classification (with classification accuracy) on defect features. We decided to use YOLOv5 as this approach does not depend on a template to extract differences, but relies on pure train data where defect properties are learned, not the differences between a test and its respective template.

This approach yielded in a bounding box loss of .1 and an object loss of .02 on the validation set. However, this model is only trained to detect the existence of a defect, not its specific classification as we want to generalize this to the Aerospace Corporation dataset.



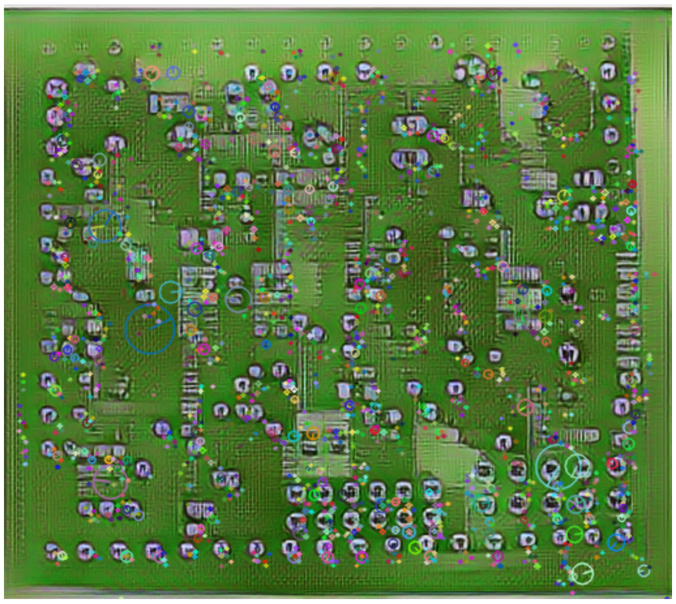
Labels on batch of validation images



Results graphs from training model

Generalization via Style Transfer

One big issue with this approach however is that the Aerospace Corporation data and Kaggle data images look very different and focus on different defect types. One approach we attempted was to use a style transfer model to transfer the aerospace data into an image that looks like the Kaggle data.



Aerospace PCB with Style transfer applied

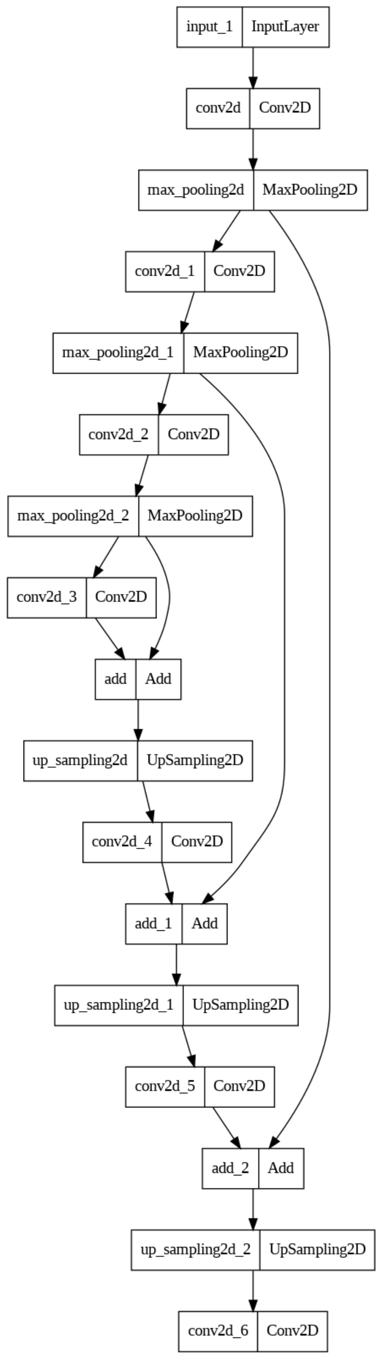
However, even with this approach, we were not able to classify any defects on the style transferred images using the YOLO model. We think this is due to the vastly different defect types in the Aerospace data and the Kaggle data.

Data Sources:

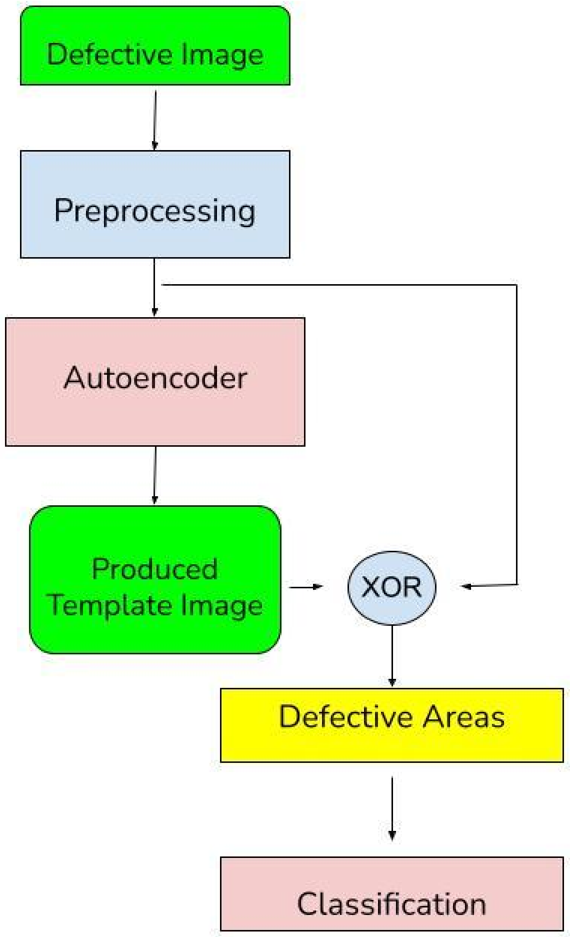
1) PCB Synthetic dataset with 1386 images with 6 types of defects https://www.kaggle.com/datasets/akhatova/pcb-defects
2) Aerospace Corporation dataset

Approach 2: Denoising Convolutional Skip Autoencoder
In this approach, we use an autoencoder directly on the Aerospace Corporation dataset to generate a non-defect image from a defect image. Performing a bitwise XOR operation on the image would then yield the locations of defects. The initial version of this approach attempted to train two autoencoders where one autoencoder was trained on PCB images with defects and the other without them. Then, the outputs from each of the autoencoders would be compared to for defect localization. However, this approach was determined to be pointless considering both autoencoders learned the identity function. Consulting scientific literature, a common approach to training autoencoders for anomaly detection is utilizing a defective image as the input and attempting to reconstruct a paired nondefective image (Khallian, Kim). This approach was found to also lead to learning the identity function when trained. However, this may be because the proportion of defective to nondefective images for which the autoencoder was trained on is quite low. This is because the training was performed on 1200x1600 images which were segmented into 400x400 images (to enable faster training) but the majority of the segmented images corresponded to nondefective images.

The current architecture of the autoencoder involves a 3-layer encoder and decoder with maximum pooling to decompose a binarized 400x400 image into a latent space of 50x50x8. Skip connections were added to the decoding layers to prevent gradient decay in the deeper layers of the autoencoder as it was initially observed when the encoder was built and used in literature (Kim).


input_1 | InputLayer
conv2d | Conv2D
max_pooling2d | MaxPooling2D
conv2d_1 | Conv2D
max_pooling2d_1 | MaxPooling2D
conv2d_2 | Conv2D
max_pooling2d_2 | MaxPooling2D
conv2d_3 | Conv2D
add | Add
up_sampling2d | UpSampling2D
conv2d_4 | Conv2D
add_1 | Add
up_sampling2d_1 | UpSampling2D
conv2d_5 | Conv2D
add_2 | Add
up_sampling2d_2 | UpSampling2D
conv2d_6 | Conv2D

Autoencoder Architecture



High Level Model Architecture

Data Sources:

1) Aerospace Corporation dataset
2) Potentially DeepPCB (dataset with defect and non-defect image pairs) if we explore classification types (would run on XORed image output)

Approach 3: Defect Localization using XOR + Classification via CNN
The XOR operator can be used in this situation where we have an image of what the PCB should look like and an image of the actual PCB. Taking the XOR of these images yields a resulting image that highlights the differences between the two. Ideally, this would mean the defects are more pronounced in the final image where we can conduct further analysis to localize the defect. The classification portion would be handled using a simple CNN. The number of layers and types of layers would be adjusted based on the need for classification. We could evaluate the accuracy of different neural network architectures and determine which one produces the best accuracy.

Data Sources:

1) This paper outlines the use cases for XOR in regards to image processing
https://www.iosrjournals.org/iosr-jce/papers/Vol17-issue2/Version-5/B017250715.pdf
3) Deep PCB with 1,500 image pairs with templates and aligned test images – useful if a template approach is taken where the differences are inputted into the model
https://github.com/tangsanli5201/DeepPCB

Next Steps

We plan to explore training with more balanced datasets. Our current dataset has a larger percentage of non-defective images compared to defective images. This makes training our autoencoder particularly difficult as there are less instances of PCB defects. We plan to generate a more balanced dataset by capturing our own images of the defective PCB's and syntheically altering images of defective PCBs to eliminate defects.
We also plan to explore multi-resolution processing. Essentially, we plan on separating an image into varying resolutions - which produce the original image when added back together - and supplying the separated images as training data for our autoencoder to see if we are able to generate better results. The goal here is to train multiple autoencoders that work at different resolutions to capture defects at multiple scales in an image to enhance localization.

Main tasks, milestones, roles of each team member, and schedule

1. Image pre-processing **(4/1/23)**
    a. Define area of PCB
    b. Binary Mask to standardize all images
2. Image Enhancement **(4/18/23)**

    a. Sharpen PCB images to emphasize possible errors
    b. Filter out noise that could hinder predictions
3. Use one of the main approaches to determine location and type of defect **(4/29/23)**
4. Analysis of Results **(5/9/23)**

These four steps are the main milestones we need to accomplish in order to be successful. We have allocated the most amount of time to image pre-processing as we believe that this will be the most complicated and time consuming task. Additionally, the thoroughness of our image pre-processing will have a direct correlation to the success of our final algorithm to detect PCB defects. It is also important to note that the dates provided are tentative and are subject to change based on project needs.
Specific roles of each team member will be added when the project tasks become more apparent.

References

Adibhatla, V.A.; Chih, H.-C.; Hsu, C.-C.; Cheng, J.; Abbod, M.F.; Shieh, J.-S. Defect Detection in Printed Circuit Boards Using You-Only-Look-Once Convolutional Neural Networks. Electronics 2020, 9, 1547. https://doi.org/10.3390/electronics9091547

Huang, W., Wei, P., Zhang, M. and Liu, H. (2020), HRIPCB: a challenging dataset for PCB defects detection and classification. The Journal of Engineering, 2020: 303-309. https://doi.org/10.1049/joe.2019.1183

Khalilian, Saeed, et al. "PCB Defect Detection Using Denoising Convolutional Autoencoders." *ArXiv.org*, 28 Aug. 2020, https://doi.org/10.48550/arXiv.2008.12589.

Kim, J., Ko, J., Choi, H., & Kim, H. (2021). Printed Circuit Board Defect Detection Using Deep Learning via A Skip-Connected Convolutional Autoencoder. Sensors (Basel, Switzerland), 21(15), 4968. https://doi.org/10.3390/s21154968

Singh, Anurag and Namrata Dhanda. "DIP Using Image Encryption and XOR Operation Affine Transform." (2015). https://www.iosrjournals.org/iosr-jce/papers/Vol17-issue2/Version-5/B017250715.pdf

Image Preprocessing:
The first crucial step to implementing our algorithm would be to process the PCB images. Meaning, we needed to generalize the images to ensure the input to our defect detector had the least amount of erroneous noise as possible. After experimenting with many different procedures such as simply converting the images to gray scale and