Deep Learning for Locating Defects in Manufacturing (QC)

Akshay Devadiga, Arun Mani Sam, Guru Prasad Poornam, John Rekesh

Abstract: We present a new approach for detecting defects on PCBs, including their spatial location and extent. The method extends to other manufacturing scenarios. We use a hierarchical multi-resolution approach to image analysis. The idea is that different levels of image resolution can provide different kinds of information about the target. Each such level can be processed at an acceptable resolution for deep learning, and we proceed deeper hierarchically, maintaining similar image resolutions where possible. The end result from such a hierarchical exploration can be synthesized for overall analysis. This enables the processing of very high resolution images, which otherwise require extremely high computational time and cost investments for neural network training. We also present an approach that spatially locates defects (location, size and shape) on PCBs using only unsupervised training with sane or good images.

We further discuss extensions to our approach to widen its scope

Introduction

There are well known methods in AI for detecting defective products in a manufacturing pipeline using say, images from a camera. What is not so well known is the question of locating and characterizing those defects, in terms of their spatial location and extent. Where exactly (in spatial coordinates) on the product does that defect exist, and what does that defect look like?

Take Printer Circuit Board (PCB) manufacturing for example. There could be many PCB problems to look for, such as open solder joints, missing components, misaligned components, wire breaks, bent pins, fluid leaks, shorts and so on. For each of these, there is a whole class of defect types. The National Physical Laboratory (NPL) has an industrial defect database for PCBs which document common problems. But it is not sufficient to detect that a component may have a type of problem. It is also required to locate the exact area, shape and spread of a defect to be able to fix potential issues in the manufacturing process.

Approaches & Problems

Common object detection models such as YOLO, SSD, R-CNN and others can be trained to detect different types of defects on a PCB. These models can identify a bounding box on an input image where a specific defect is contained. Then there are image segmentation models such as U-net, Mask R-CNN, and Deep Lab which can be used to assign pixels in an image to an object instance or type, in this case a specific instance of a defect. These approaches are supervised learning models requiring extensive examples of all types of defects.

Another problem one may face, for example in the cases of PCBs, is the very high resolution of the images needed to be able to work with very small components and defects. Image resolution required in these cases could be at micrometer scales. Large image sizes are harder to train on neural networks, with an equivalent rise in computational costs.

Combining Object detection & Autoencoders

We arrived at a novel approach to detecting and locating defects on a PCB, which drastically reduces the number of training images required, and can handle higher resolution images without a corresponding rise in training costs. The approach does not apply to every type of defect but does address a good subset.

In our approach there are two stages in the inference pipeline. The first stage identifies various components on a PCB using a lower resolution version of a given high resolution PCB image. This means down sampling say a 4k x 4k resolution image to 1k x 1k resolution. We may lose defect information in this process but the components on the PCB are still identifiable. Next we run an object detection model such as Faster R-CNN to locate all the components on the PCB. Training such an object detector does not require as many images of components as would be needed if we were looking for defects. These networks generalize, so a damaged IC or capacitor still shows up as one.

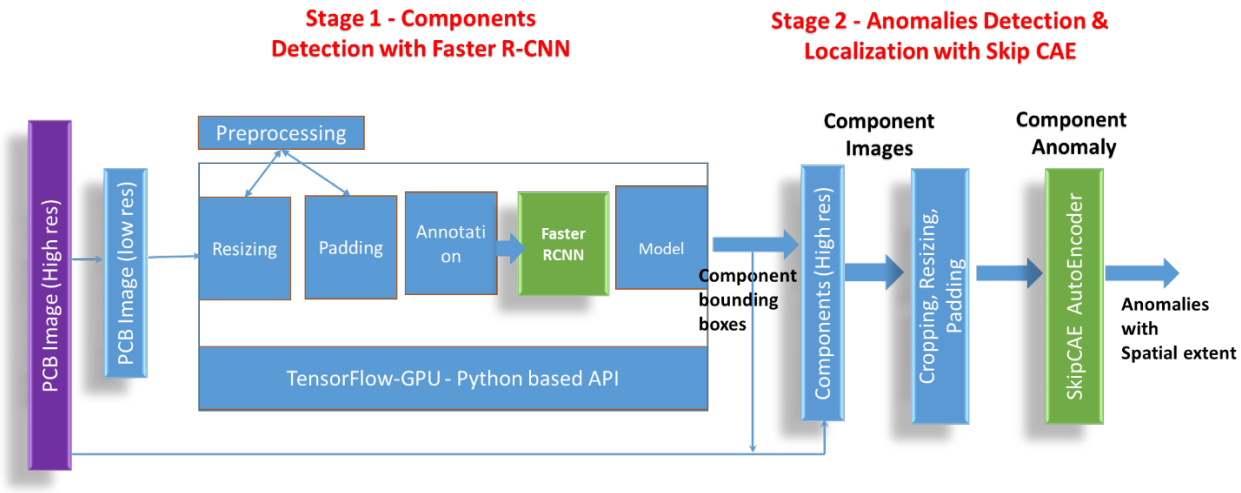

Fig 1. Inference Pipeline

In stage 2, we move to defect detection and localization on these components. For this we identify the bounding boxes of each detected component from stage 1, and map those bounding boxes to the original high resolution PCB image (4k x 4k in this example). Then we extract the high-resolution images of those specific components. Since most components are tiny, these images are relatively small but contain defect information at high resolution.

Next we run these high resolution component images through a Convolutional Auto-Encoder (CAE) with skip connections, similar to those used in U-Net. There is one CAE for each type of component. A CAE in this case is trained to reproduce the input image of a component as its output (to be specific, it is not generating a segmentation map as in U-Net, but the input image itself). Because of the bottle-neck layer present in the CAE, the encoder and decoder portions of the network are forced to learn features of a component image, then compress and decompress to regenerate the image.

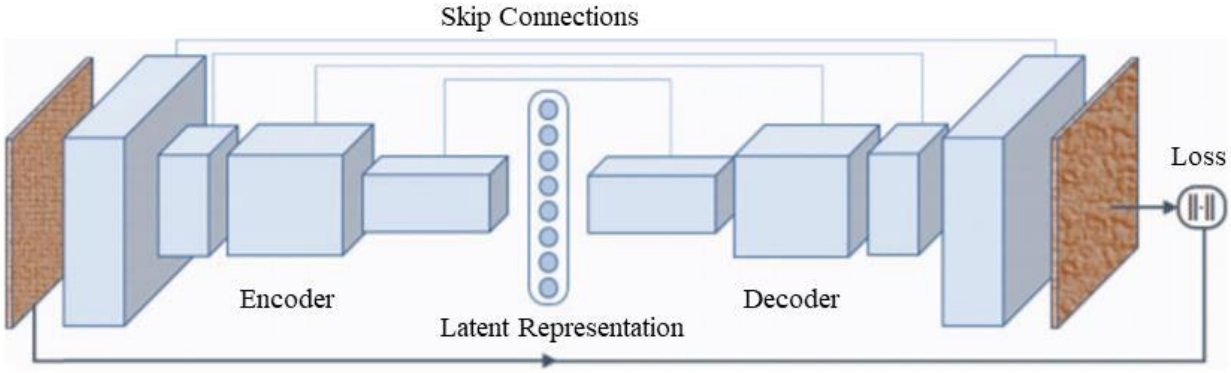

Fig 2. Convolutional AutoEncoder (CAE) with skip connections

The reconstruction loss for an input image that is fed to a trained CAE is used to determine if there is an anomaly in the image. For this it is sufficient to subtract the output image of the trained CAE from its input image (use an XOR), or look at the reconstruction loss heat map to find the difference between the ideal/expected component image and the actual one. In our experiments, **this heatmap corresponds very closely to the location and spatial extent of the defects on the component**. Some example images are given below.

There are multiple versions of CAEs that could be used. For example, one uses skip connections, another uses an inception like network, yet another may use resnets. These networks ultimately do much the same thing, though speed of convergence and computational requirements do vary.

Results

We have uses a hierarchical multi-resolution approach to image analysis. The idea is that different levels of image resolution can provide different kinds of information about the target. Each such level can be processed at an acceptable resolution, and we proceed deeper hierarchically, maintaining similar image resolutions where possible. The end result from such a hierarchical exploration can be synthesized for overall analysis.

One prime difference with other approaches is that we have used only sane or good images of components to train CAEs for detecting defects. There is no reliance on any sort of large datasets that contain all possible defect types and shapes. Such datasets are also very hard to come by. The CAEs are able to accurately find the location, size and shape of defects.

For locating components on a PCB, we tried multiple models. The aim was to identify really tiny objects, given a low resolution image of a PCB. We settled on Faster R-CNN after trying multiple others. Specifically Faster R-CNN with Resnet-50 V1 Object detection model scaled for 1024 x 1024 resolution has been used. Faster RCNN combined with Resnet known for its skip connections during the backward propagation, has shown good success for tiny objects in remote sensing images. Resnet architectures are known for avoiding both vanishing and exploding gradient problem. Training of the model was done through use of component images from the FICS-PCB dataset which is recommended for classification models. The dataset includes full PCB images in addition to components. We found that our Faster R-CNN model did a good job of detecting ICs, resistors, capacitors and other components.

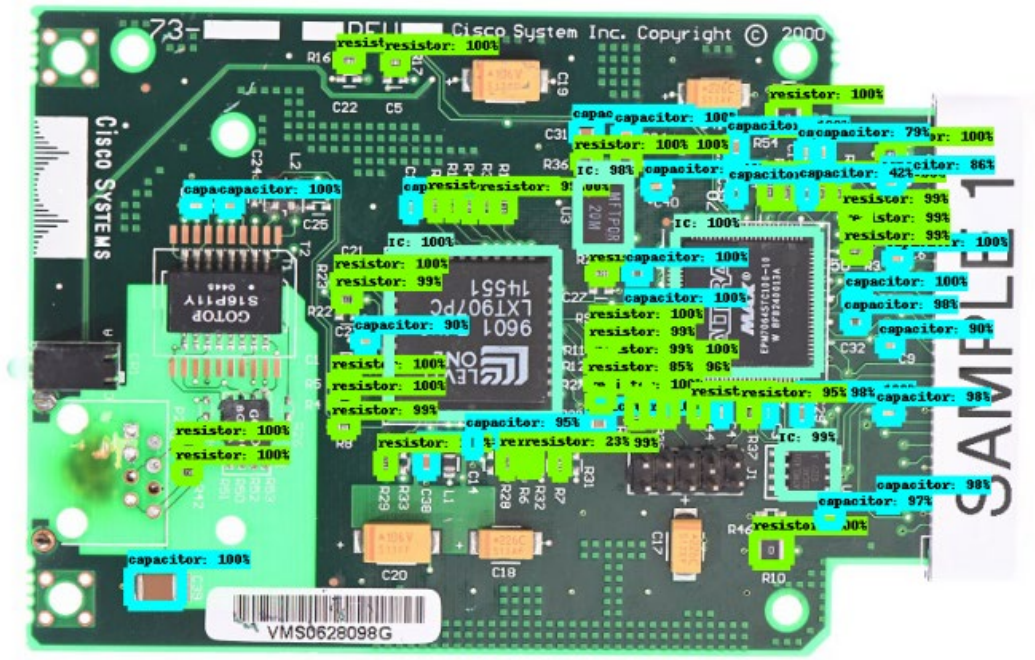

Fig 3. Components detection on PCBs

The autoencoders (one per component type) were trained with images of good components, i.e. without any defects on them. We may call this unsupervised training, as no labeling and classification were required on these images. Each CAE learned to reproduce its input, test, and validation images with very little loss. After training, if images with defects are given to a trained CAE, reconstruction of those images will not be accurate. The actual defects on a component can be located by inspecting these reconstruction losses. In the examples below we also introduced synthetic defects on component images to see how the autoencoders perform.

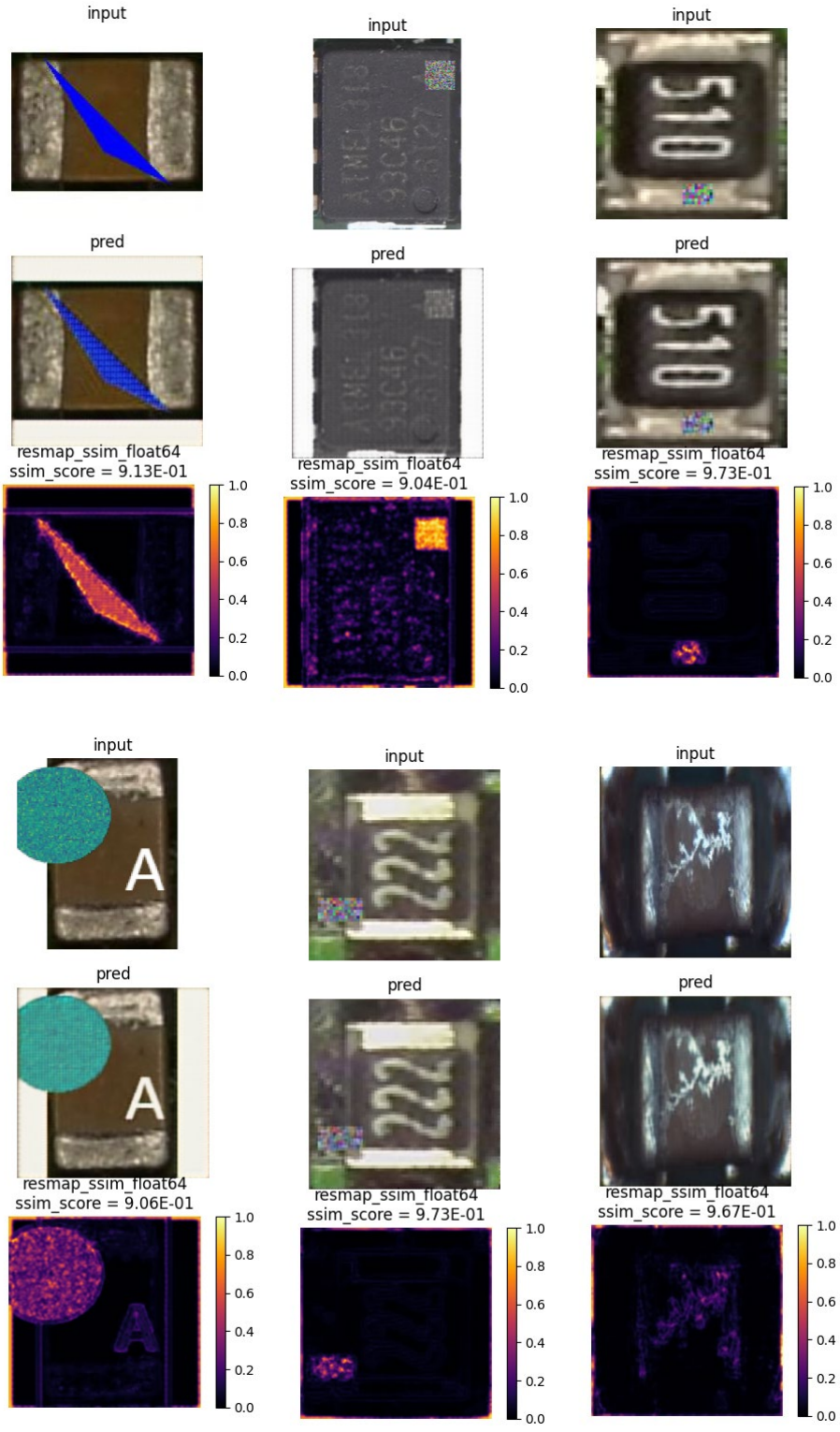

Fig 4. Heatmaps of synthetic defects introduced on PCB components

Going Further

We were inherently limited (even severely) in our access to datasets on PCB images, and therefore this work needs to be taken in with some caution.

Here are some straight forward extensions to increase the scope and accuracy of our approach, in addition to using larger datasets.

1. **Missing Components**: Object detection can provide a simple inventory count of the components on a given PCB image, to identify what specific components are missing if any, and from where. This can be done against a master inventory and template for the PCB.
2. **PCB-specific Learning**: One may also train the models mentioned above for specific PCBs being manufactured, so that the accuracy of detection on those PCBs can be higher. This must be done through a transfer learning approach where a model originally trained on a larger data set and has learnt general features is then fine-tuned for a given PCB.
3. **Classifying Defects**: The heatmaps generated by the CAEs will have different shapes and sizes depending on the types of defects that were detected. An image classification model trained on these heatmaps may be used to also identify what kind of defects they represent (in technical terms).
4. **Alignment Problems**: Component alignment issues are not directly addressed by this approach. But the CAEs can also be trained to generate segmentation maps. A border/contour detection algorithm or template matching algorithm can be used to check if the segmented components are misaligned vs a master template. Another approach may use image subtraction, as is common.
5. **Beyond Components**: Nothing restricts a given CAE to be trained for just one component. CAEs can be trained with multiple components in its image scope, or a larger “bleed” area around a given component. This can be used to locate defects that occur in between components, especially when trained on a specific PCB (using transfer learning). Training CAEs on whole PCBs require large datasets and lots of computing time & power because of the high resolutions involved. One can always find a good middle ground between individual components and full PCB images.
6. **3D Inspection**: 3D is another interesting area. The human neuronal network constructs 3D visualizations from 2D stereo images provided by the eyes. Note that a convolutional network takes an image volume (e.g. RGB planes) as input. Images from multiple fixed perspectives can be added to this volume, even in grayscale where possible. For a CAE, the losses or heatmaps generated for individual planes need to be combined into a 3D mapping to visualize the defects. There are several techniques and papers extant on using 2D images to create 3D visualizations.
7. **Hierarchical multi-resolution**: Even though we have used only 2 levels of image hierarchy in our analysis, one could very well define multiple levels between whole PCB images and individual components. This is left for future work.

Conclusions

We realize that the work presented here is limited, mainly because of our lack of large datasets, and also from not being insiders in this specific industry (PCB quality control). However, we believe that the approaches outlined here, and their extensions would be of sufficient value to make a difference in manufacturing quality control and possibly other areas as well. The current work demonstrates the right direction towards a robust and reliable image anomaly detection system for PCBs.