# Visual Analysis of Printed Circuit Boards

**Marco Günther**
Bachelor's Thesis

**Examiner:** Prof. Dr. Heiko Neumann
**Supervisors:** Dr. Bastian Könings, Fabian Weber, Heiko Ehret
**Submission Date:** November 5, 2020

Issued: November 5, 2020

I hereby declare that this thesis titled:

**Visual Analysis of Printed Circuit Boards**

is the product of my own independent work and that I have used no sources or materials other than those specified. The passages taken from other works, either verbatim or paraphrased in the spirit of the original quote, are identified in each individual case by indicating the source.

Ulm, November 5, 2020

Marco Günther, student number

# Abstract

Sophisticated object detection is used in automated systems throughout many areas of application. This work elaborates the possibilities of utilizing a modern object detection approach for visual analysis of printed circuit boards (PCB) in the context of hardware-related security assessments. In detail, the task is to reliably localize and classify different types of integrated circuits and other components on image data depicting electronic devices.

Faster R-CNN, an object detection architecture based on modern convolutional neural networks, is selected for this purpose. In order to effectively train the detector a variety of datasets are collated that contain suitable annotations for PCB components. In parallel, an internal image acquisition process further complements the available datasets. All dataset are extended by introducing six new sub-categories for integrated circuits. This approach of IC class breakdown is unique and not present in any of the investigated datasets so far. In addition, four new categories for passive and miscellaneous components are also introduced into the data. Finally, the object detection architecture is appropriately configured and several models are trained with different combinations of the available image data.

The best model yields an overall detection performance of 0.57 mAP. Detection scores of individual classes tend to be more vulnerable to small component sizes and high intra-class variance than to class imbalances of the datasets. Acceptable performance scores above 0.6 AP are reported for four out of ten classes, two even reach 0.75 AP.

# Contents

# 1 Introduction

The developments in industry automation and telematics have lead to an increased deployment of highly sophisticated control systems in form of embedded hardware. Recently, these devices are getting more and more interconnected and form complex systems that are potentially vulnerable to malicious attacks. Unsecured communication channels, programming flaws and insufficient assessments before deployment form security vulnerabilities that can be exploited in order to gain unauthorized access, disrupt functionality or even cause permanent damage to affiliated systems. Fortunately, the awareness for this threat scenario is rising. For example, the specialized research field of automotive security has emerged from general IT security. Furthermore, the demand for corresponding assessments is growing and answered accordingly by the service industry. SCHUTZWERK GmbH in Ulm is such a company that offers specialized security assessments for embedded systems [36].

## 1.1 Problem definition

In black box penetrations tests the security analysis of an embedded system is performed without in-depth knowledge or internal design specifications from the manufacturer [44]. Therefore *reverse engineering* is performed to determine the design and functionality of the hardware in question. In general, this process has a broad application scope, ranging from military or commercial espionage, interfacing, obsolescence to security analysis and more [45].

In detail, several consecutive steps have to be taken in order to perform hardware-dependent security analysis. At the beginning, relevant integrated circuits (IC), like micro-controllers and memory modules, are identified on the *printed circuit board* (PCB) that holds these components. If available, documentation and manuals for the parts are consulted. The interconnections between the components are traced visually (if possible) or revealed by resistance measurements between the individual component pins. This results in exposed data interfaces that are analyzed with respect to control signals and communication protocols. From there on, detailed security assessments are performed for the specific protocols in use. All described steps are currently performed manually, especially the early steps are time-consuming and tiring.

The ongoing development in the areas *Computer Vision, Machine Learning* as well as *Rapid Prototyping* bear potential for the initial steps of visual analysis and interconnection search. Especially recent advances in the field of object classification and detection achieved by *convolutional neural network* (CNN) based detector models seem optimally suited for an automated visual analysis of PCBs. Furthermore, a master thesis at SCHUTZWERK has already assessed the possibility of constructing a *flying probe tester* that is able to perform interconnection search and signal analysis with movable measuring tips [43].

It is the belief that a combination of both approaches can yield a soft- and hardware based solution that is able to automatically perform the described process steps for PCB component localization, chip identification as well as interconnection and signal analysis.

## 1.2 Thesis contribution

Besides constructing a flying probe tester, the progenitor work has already conducted preliminary experiments with CNN based object detectors. The evaluation of IC and pin localization has been the primary focus. All detection models have been trained with class-agnostic PCB datasets that do not offer distinct labels for the contained ICs. Therefore, classification has been postponed to a future work.

The purpose of this thesis is to continue the previous work with focus on refining PCB component detection. In detail, localization and classification of PCB components are to be performed. In the following, this combined effort is commonly denoted as *object* or *component detection*. It is the idea that this task can be jointly solved by utilizing a state of the art CNN-based object detector model. In order to achieve this goal, IC sub-categories have to be devised and the datasets have to be re-labeled. In addition, new classes for passive PCB components are also introduced that are of special interest for security assessments.

Furthermore, the available image data should be increased as CNN-based models generally require sufficient data for training. Further public and internal image sources are to be utilized. This new data has to be adjusted likewise with respect to the newly introduced categories.

The next step is to deploy a suitable object detection model and fine-tune its configuration with respect to the scope of application. Finally, the object detector has to be trained and evaluated on the available datasets. Overall, open-source software is to be utilized if possible and extended as necessary.

In order fulfill the stated tasks, the following chapters of this thesis are organized as follows: Chapter 2 presents related work in the context of PCB analysis with focus on component classification and detection. A selection of classical CV- and modern CNN-based solutions are presented that help to understand the decision for utilizing a CNN-based object detector during the rest of this work. Subsequently, the chosen object detection model is presented and its concepts described thoroughly. Finally, existing datasets within the scope of PCB component detection are presented. Chapter 3 describes in its first part the decisions, concrete adjustments and deployment for the object detection model. The second part elaborates the chosen datasets for this work and their adjustments. Chapter 4 contains the experimental results and provides detailed analysis for them. Chapter 5 concludes this work with a summary of the achieved tasks, results and an outlook with respect to future work.

The described object detection architectures are generally independent from the underlying CNN feature extractor. Therefore, it is not part of this work to provide an in-depth analysis of the various mentioned CNNs. A basic understanding of the concepts and common structural elements of CNNs, like convolutional or max pooling layers, should be sufficient for this work.

# 2 Related work

## 2.1 Academic publications on PCB analysis

### 2.1.1 Classification

This part briefly describes approaches that solely deal with the classification of PCB components. One classic CV as well as one CNN based is presented.

#### 2.1.1.1 Automatic classification of SMD packages using neural network

Youn et al. [50] present a classification method for SMD components on PCBs for automated optical inspection, assisting manufacturing processes and enabling subsequent quality assurance. The authors approach extracts color and edge information during optical inspection and utilizes a neural network for classification of the parts. The ground truth data and positioning information for each component on a board are provided by design specifications (e.g. Gerber data) and supplemented by an image of the error-free PCB version.

In order to extract color information the input image is preprocessed by converting the RGB colors to the HSI (Hue-Saturation-Intensity) color space to compensate for varying lighting conditions during image acquisition. For each image section containing a SMD component, histograms for the Hue and Saturation channels are generated, representing distributions of the channel values. Subsequent binning into a fixed number of bins and binarization with a manually chosen threshold yields a fixed sized 0/1 valued vector for each channel.

The extraction of edge information works in a similar fashion. First, a Canny edge detector is applied on a gray-scale version of the image section, yielding interconnected edges (hysteresis) with minimal representation (non-maximum suppression) while emphasizing strong and discarding insignificant edges. The required internal thresholds for the Canny algorithm are acquired by applying Otsu's threshold method on the original input. Vertical and horizontal projections of the edges are generated by applying row- and column-wise summations of the Canny image, followed by binning and binarization, yielding fixed sized vectors for horizontal and vertical edge information.

The vectors are concatenated and used as input for a Multilayer Perceptron (MLP) network with back propagation as supervised learning strategy and sigmoid activation function for the neurons. The input vector's length defines the number of neurons in the input layer while the last layer consists of five output neurons due to the objective of discriminating five different SMD components. The number of hidden layers is set to 10, the quantity of neurons per hidden layer is not stated.

The evaluation consists of two experiments on 83 component images for training and 154 for testing. The first experiment uses color information only and yields 0.753 average accuracy over all classes for correct classification. Good results are archived for three classes with distinguishable color profiles while the remaining two show significant confusions. The lowest class-specific accuracy is 30.8% and thereby below chance level. The second experiment utilizes color and edge information, yielding 97.6% average accuracy and 92.1% lowest class-specific accuracy. These results demonstrate the importance of including both color and edge information in order to successfully discriminate between the given classes of SMD components.

The described method is exemplary for a classical approach in Computer Vision without the usage of Convolution Neural Networks. The pipeline captivates through its straightforward workflow and simplicity of its parts. The results with combined color and edge information convince at first glance. But so far, the endeavor only shows results for discriminating five classes of SMD components, all with similar and small image dimensions as well with only two solder joints. The authors itself state that a follow up study has to determine if the method is applicable to other types.

#### 2.1.1.2 SMD Classification for Automated Optical Inspection Machine Using Convolution Neural Network

The approach by Lim et al. [17] uses deep learning techniques in order to realize the classification of SMD components on a PCB during automated optical inspection. The approach utilizes a multistage process with semantic segmentation in the first and classification in the second stage for 64x64 images of PCB components. Ground truth position and dimensional information of the components are retrieved from design specifications. The first stage consists of a Fully Convolutional Network (FCN) [22] that is capable of learning pixel-wise classification and is used to distinguish between the SMD component and background. A FCN is a special variant of a Convolutional Neural Network (CNN) which has been stripped of its final fully connected layers for classification. They are replaced by an up-sampling layer that uses backwards strided convolution (also called deconvolution or more correctly transposed convolution) to generate an output map with same size and width as the original input image containing the predictions for semantic segmentation. It can be further improved by adding additional skip connections, thus incorporating intermediate results from earlier layers of the network into the final output map. Ground truth masks for training the FCN seem to originate from the design specifications.

Afterwards, the largest region is identified and small ones are discarded. Morphology operations close small gaps and remove frayed boundary parts. The resulting binary mask is used to cut out the components from the input image, removing all background information like surrounding silk screen printings or circuit patterns.

In the second stage a CNN is trained in order to discriminate 14 different classes of PCB components. 7659 training and 4822 testing images with a size of 64x64 pixels are used. Two experiments are conducted, one without the usage of the first stage and the other with both stages. Without prior segmentation, the average accuracy is 79.2% with lowest class specific accuracy of 0.0% for MELF capacitors. The combined approach yields 90.8% average and 50.0% lowest class specific accuracy. The previous poor result for MELF capacitors is now raised to 100.0% while four other classes drop slightly.

The work shows that the performance of Convolutional Neural Networks can be effectively increased through proper segmentation as a preprocessing step. The described two stage pipeline is comprehensive. Especially the first part is reasonably explained. For the second stage the authors do not describe what kind of CNN is used or if it has been pretrained. The evaluation only shows the final results for classification. The FCN's performance and the quality of intermediate segmentation are not analyzed. This leaves room for interpretation whether the classification stage is hindered by poor segmentation or needs further improvement (e.g more training data for generalization or other adjustments) to explain the decrease of accuracy for individual classes between the first and second experiment. The used dataset consists of a large variety of images but only three classes represent integrated circuits (IC) with more than four solder joints, thus concentrating on more simple PCB components for the purpose of quality control during automated inspection.

### 2.1.2 Object detection

This part briefly describes approaches that perform object detection of PCB components. Similar to the previous part, one classic CV- and one CNN-based method is presented. Especially the first description outlines the intricate efforts that are necessary if CNN-based solutions are omitted.

#### 2.1.2.1 Localizing components on printed circuit boards using 2D information

Motivated by the necessity of PCB waste recycling, Li et al. [16] present a modular analysis pipeline for the localization of mounted components in PCB images that combines complementary information sources with a diversification strategy in order to propose promising candidate regions for PCB components.

Their first step is preprocessing the images by removing noise with a median filter and uneven illumination with comprehensive color normalization. Afterwards, a scale-space representation is created by applying a non-linear scale space implementation, creating multi-scale images with various resolutions that cover possible components on all scales. All subsequent operations are performed for each scale, leaving it to the final classification stage to learn and select results on the correct scale and discard non-relevant ones.

The next step covers the extraction of features from an image. The authors focus on intensity from gray values, color from different color spaces (RGB, HSV, CIELAB and CIELUV), edges and textures. Structured random forests [4] are used for edge detection and cut-off windows [26] for extracting local texture features. The gained information are used in the next stage for generating region proposals of PCB components.

Various pixel grouping strategies are utilized that exploit local smoothness and similarity in order to create meaningful sub-regions, effectively performing segmentation by partitioning an image. For this task the authors employ the graph-based methods graph cuts (GC) [1] and efficient graph-based image segmentation (FH) [6] as well as the local distribution-based mean shit segmentation (MS) [3] on the extracted feature spaces. The results are labeled feature maps with proposals of regions that segment the input image. In order to prevent over-segmentation, a subsequent merging step is considered by applying data compression-based algorithms on the texture features of the generated regions. Another approach for merging is selective search which hierarchically groups regions based on color information and similarity measures, forming merged region candidates at different scales. Finally, the proposal generation stage is complemented by a background removal process in order to reject false region candidates. Substrate regions of the printed circuit boards are identified solely by their distinct color information. The *mean shift* algorithm is applied on a 3D histogram of the corresponding color space of an image, clustering the space into disjoint regions. Subsequent joining of neighboring and small clusters with connected-components analysis reduces the number of relevant clusters. The input image is divided into regions by mapping each distinct color to the color of its nearest cluster center. The authors finalize the background removal by training a *weak classifier* that exclusively recognizes substrate regions in a flexible manner for different kinds of PCB images. The overall result of this stage are segmented images with region proposals for PCB components free of background/substrate areas.

The final step of the pipeline, called *validation*, performs classification of the generated region proposals. Locally extracted features from SIFT and SURF algorithms are used to create a Bag-of-Visual-Words (BoVW), a vocabulary consisting only of the cluster centers (also called *visual words*) of all collected feature vectors in this stage. The concept is applied during training by collecting a large quantity of local SIFT and SURF features from known ground truth regions. The resulting vocabulary consisting of $N$ cluster centers and label information from the ground truth data is used to train a final classifier.

Their following evaluation thoroughly describes the performance of various stages, the optimal combination of the complementary methods for segmentation and candidate generation as well as final choices for used descriptors and machine learning models. The used dataset consists of 31 images of PCBs with 1124 labeled components with manually created region masks in form of rectangular bounding boxes. At first, only the performance of single pixel-grouping strategies (GC, FH, MS) with and without background removal is observed. All variants perform better with the proposed background removal and the best scores are obtained with FH (0.709 recall, 0.163 prec.) and GC (0.62 recall, 0.178 prec.). The previously mentioned merging processes, aimed to prevent over-segmentation, are dismissed from the workflow due to extreme computational demands or bad performance results. The next step in evaluation, called the *diversification strategy*, concentrates on assessing optimal parameters for the pixel grouping strategies and which combination of these complementary method yields the best results. While the single methods show marginal performance differences regarding the used scale and parameter settings, the overall result is, that a combination of all three methods produces the best recall score. The final classification process for the remaining region candidates is evaluated with SIFT and SURF features with random forests and support vector machines as classifiers. The combination of SIFT features and random forest yield the best results for effectively canceling false candidates. The overall localization performance with the diverse usage of all three grouping strategies, with background removal, without region merging processes and with final validation of region proposals via machine learning is given with a recall score of 0.808 and precision of 0.510.

The presented pipeline captivates through the combination of sophisticated computer vision approaches in all phases. The authors utilize state-of-the-art methods for preprocessing, scale-space representation, feature extraction, pixel grouping strategies for segmentation and candidate generation. The approach is complemented by the extraction of modern feature vectors and their evaluation by popular and well-established machine learning algorithms without the need for extensive training of neural networks. Final classification performance seems promising, the moderate precision score due to increased false positives might be quite sufficient for PCB waste recycling. It is not explicitly stated if the system is performing multi- or single-/one-class classification. The presented colored label maps after candidate generation and validation could indicate the former, but the lack of per class performance metrics suggests the latter. Some ambiguities remain, for example the background removal process. It does not become actually clear how this step is really distinguished from the general candidate generation and is able to produce background-only results, except for limiting it to color spaces, applying only MS segmentation and training a separate *weak classifier* which is not specified concretely. Another unclarity is how many SIFT/SURF feature vectors are typically generated per region candidate and how a combination of them is precisely evaluated by the final classifier after applying the BoVW model. Some kind of voting process could be applied if not all vectors yield the same conclusion, leaving this open for discussion. Finally, almost all methods are depending on one or multiple parameters, especially candidate generation and classification. Although the authors emphasize on the evaluation of various parameter settings, they do not state any values or viable ranges for their parameters – not even the number of suitable clusters for the BoVW approach. This caveat makes it challenging to reproduce the presented work, considering that the internal evaluation process probably has to be re-created likewise.

#### 2.1.2.2 Application Research of Improved YOLO V3 Algorithm in PCB Electronic Component Detection

A more recent development for PCB component detection is presented by Li et al. [15]. The authors propose an improved version of the You Only Look Once (YOLO) v3 object detector model [32] in conjunction with a joint dataset of real and synthetically generated PCB images. The used algorithm is a CNN-based one stage detector which generates a fixed number of predictions with simultaneous localization and class prediction. Not only the last layer is used but results from several intermediate feature maps of different spatial resolutions are diverted to the so called YOLO-layers where classification and localization is performed. Similar to a FCN, the information of intermediate YOLO layers are enhanced by combining it with an up-sampled version of a deeper (thus smaller) feature map. This concept is called *feature pyramid network* (FPN) [19]. In YOLO v3, three extracted layers represent the processed input image with 1/8, 1/16 and 1/32 of the original resolution, with an initial image input size of 416x416 pixels. Due to the dimensionality reduction present on the level of 1/8 resolution, the authors propose the implementation of a 4th YOLO layer which utilizes the features from the 1/4 resolution backbone layer concatenated with the up-sampled information of the 1/8 YOLO layer. This step is motivated by the demand for detecting SMD components of small spacial dimensions, like resistors and capacitors.

Like other object detectors, YOLO v3 uses anchor boxes for the prediction of bounding boxes. They serve as templates in order to learn and predict the exact width and height of bounding boxes for the sought objects. The model is capable of accepting custom anchor boxes that can be pre-computed. K-means clustering is utilized on the ground truth bounding boxes of used datasets in order to extract a fixed set of the most common dimensions for the anchor boxes (9 cluster centers for the original and 12 for the enhanced model). Each YOLO layer receives three anchor boxes with respect to its resolution, e.g the three largest anchors are assigned to the deepest layer (1/32) which represents the coarsest resolution.

The authors present a joint dataset of real PCB images from [12] combined with synthetic ones generated with Altium Designer [1], a PCB design software. The resulting dataset contains 50 images with 9145 electronic components grouped in 29 different categories. An extensive augmentation process enlarges the dataset to 20 times of its original size. Afterwards, a split with ratio of 4:1 is performed, resulting in a training set with 800 examples and a test set with 200. The distribution of classes and variation with respect to the number of contained objects per image are presented, showing that some images contain up to 800 objects.

Experiments and evaluation are conducted on four different networks. The first two depict the impact of used training data, YOLO v3 trained on COCO dataset vs. the authors own, resulting in 0.7714 mAP and 0.7988 mAP respectively for the test

[1] Altium Designer: `https://www.altium.com`

set. The latter two show the effect of an increased number of maximum bounding box proposals used in the regression and classification layers for the final prediction, a parameter that is usually set to 120 and now increased to 800. Also, the last network is the enhanced version described before. While the standard YOLO v3 with 800 bounding box proposals does not yield better results (0.7643 mAP), yet the enhanced version in combination with increased proposals yields 0.9307 mAP. A detailed table with per class AP scores for all four networks rounds up the analysis, showing how detection accuracy is raised especially for resistors, capacitors and other small SMD components.

The described approach in the paper is clear and comprehensive. A well-established object detector based on a modern convolutional network is enhanced with reasonable effort. The concept of extending existing datasets with synthetic data sounds promising in order to circumvent the sparseness of freely available datasets for PCB component detection. The reported final performance score of 0.9307 mAP is an impressive result that succeeds comparable works (Kuo et al.[12]: 0.653 mAP). However, this success could be induced by the manner of data preprocessing in order to increase the available images. The authors employ augmentation and shuffling before splitting the dataset into training and test parts, which leads to the strong possibility that augmented and original versions of the same image are present in both sets. This practice should be avoided in order to receive meaningful and reliable results from evaluation and testing.

### 2.1.3 Previous work at SCHUTZWERK

Exploring the possibilities for automation in the context of embedded system security analysis has been the topic of the progenitor work at SCHUTZWERK [43]. It consists of two distinct parts. The first elaborates the detection of chips and their respective pins with the help of object detectors based on CNNs. The second, major part deals with the development of a hardware platform for automated electrical contacting, similar to a flying probe tester used for quality control by PCB manufacturers. An in-depth discussion for the mentioned datasets below is provided in Section 2.3.

The chip detection part utilizes two models from TensorFlow's Object Detection API (ODA) [11]. The first is a Single Shot Detector (SSD) [21] implementation with a pre-trained ResNet-50 feature extractor [9]. Like YOLO v3, the SSD meta-architecture is a one-stage detector. In contrast to YOLO v3, it employs the concept of *pyramidal feature hierarchy* [18]. This is realized by additional convolutional layers that replace the fully connected layers of a classical CNN. Each layer produces a further downsampled feature map that is used to predict class labels and bounding boxes simultaneously at each position. For the latter, the already described concept of anchor box proposals is used. The SSD implementation of ODA processes images with a fixed resolution of 640×640 px. Its main advantages are short inference time and low requirements with respect to computational power. The downside is lower performance on detecting small-scale objects.

The second object detection model used in the previous work is a Faster R-CNN [33] implementation with a pre-trained Inception-ResNet-v2 feature extractor [39]. It is a two-stage meta-architecture, that is described in more detail in Section 2.2. In contrast to the SSD model, the Faster R-CNN implementation of the framework can process images with variable resolution. In its standard configuration the aspect ratios are preserved by limiting the larger dimension of input images to 1024 px and allowing a flexible shorter dimension.

Both models were trained either with the dataset PCB-DSLR or SW PCB-Google. In addition, Faster R-CNN was also trained with both datasets. Cloud-based training was performed free of charge on the Google Colab platform, which provided a NVIDIA Tesla K80 GPU with 12 GB memory for accelerated learning. In order to preserve confidentiality, the models were evaluated locally on the SW PCB-Custom dataset.

Average precision, based on the Pascal VOC metrics, was used as a metric for evaluation. The models yielded widely different performance scores depending on the utilized training datasets. For PCB-DSLR only, the SSD model reached 0.421 AP while Faster R-CNN performed with 0.571 AP. Both models trained with SW PCB-Google only, performed better and obtained 0.815 AP (SSD) and 0.874 AP (Faster R-CNN). A Faster R-CNN model trained on both datasets yielded 0.159 AP on the test dataset. A comparable experiment with the SSD model was not reported.

The overall result of the work is, that modern CNN based object detectors are a viable approach for single-class chip detection. In detail, Faster R-CNN outperforms the SSD model more or less depending on the used training datasets. Therefore, the importance seems to lie on the available image data and its properties, as it results in the greater differences of the performance scores.

## 2.2 Faster R-CNN model

The common ground for all CNN based object detector architectures is the usage of features from one or multiple feature maps computed by and underlying convolutional base network. Faster R-CNN is no exception to that and follows the concept of *single feature map* processing [18]. Furthermore, it is a two-stage meta-architecture, that uses features from the last layer of the base network for generating class-agnostic region proposals in its first stage. For each generated proposal, the second stage performs *region of interest pooling*, effectively cropping out the respective portion of the feature map and resizing it to a fixed size. This intermediate result is processed by two sibling fully connected networks which perform class prediction and final bounding box refinement. Therefore, both stages utilize the same convolutional layers and benefit from shared computations of the CNN feature extractor.

The first stage is known as *region proposal network* (RPN) and the second is commonly referred to as Fast R-CNN [7]. From a chronological perspective the

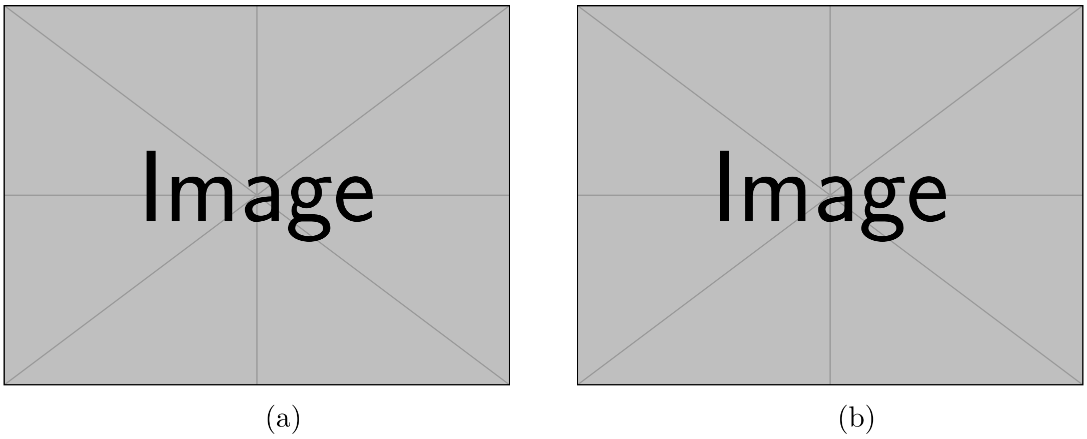


**Figure 2.1:** (a) General architecture of Faster R-CNN with RPN branch and Fast R-CNN network on top [33]. (b) shows the design and structure of the layers unique to RPN with corresponding anchor box proposals [33]. (Images removed due to copyright reasons.)

second stage was developed prior to the first. Moreover, only the combination of both stages and their joint training is called Faster R-CNN.

### 2.2.1 First stage – Region proposal network

Utilizing a CNN in order to compute region proposals for the Fast R-CNN detector was introduced by Ren et. al in 2015 [33]. Up to that point, the Fast R-CNN [7] architecture had been dependent on external proposal algorithms like *Selective Search* [41]. These approaches were often severe slow and consumed additional computational power. Therefore the authors introduced the concept of *Region Proposal Network* (RPN) with anchor box proposals.

A RPN utilizes the output feature map of the CNN that is also used for the Fast R-CNN part in order to create appropriate proposals for the latter. It is constructed on top of the final convolutional layer as a branch to the existing detection network. The first step is typically a 3×3 convolutional layer with 256 or 512 filters (depending on the underlying feature extractor). This acts like sliding window mechanism for each 3×3 patch on the last feature map of the base network. Afterwords, two sibling convolutional layers are applied, one for class membership (*cls*) and the other for bounding box regression (*reg*). The purpose of the *cls* layer is to generate $2 \times k$ probabilities that predict for each of $k$ anchor box proposal if its region covers an actual object or background. According to the authors, the *cls* layer is implemented as a two-class softmax layer in order to generate the probabilities. Altogether, this is referred to as class-agnostic prediction or objectness score. The *reg* layer produces $4 \times k$ outputs that represent the predicted locations $(t_x, t_y)$ and dimensions $(t_w, t_h)$ of objects based on $k$ anchor box proposals. More precisely, the predictions are relative offsets to the locations and dimensions of the anchor box proposals and referred to as parameterized coordinates [7] [8]. Altogether, for each sliding window both sibling layers predict the probabilities of up to $k$ possible objects with different extents. Both layers are realized with 1×1 convolutional filters, the amount is $2k$ for *cls* and $4k$ for *reg*.

Anchor box proposals are designed with different scales and aspect ratios. The reference implementation in [33] uses $128^2$, $256^2$ and $512^2$ px with the aspect ratios 2:1, 1:1 and 1:2, resulting in $k = 9$ anchor box proposals. This is referred to as *pyramid of anchors* by the authors and effectively circumvents the need for image or feature pyramids.

In order to train RPNs, only relevant anchor box proposals are taken into account that are labeled with the following criteria: Anchors with an Intersection over Union (IoU) $> 0.7$ with any ground-truth box are labeled positive. If this criterion yields no results, the fallback is to label one or multiple anchors as positive that have the highest IoU overlap with a ground-truth box. Anchors that have IoU $< 0.3$ with all ground-truth boxes are marked as negative. All others are not considered during optimization of the multi-task loss function

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

that is applied for each mini-batch (256 labeled anchors) of an input image. In detail, the stated terms have the following meaning for each anchor with index $i$:

- $p_i$ is a result from the *cls* layer and represents the predicted probability for the anchor to cover an object.
- $p_i^*$ stands for the positive (1) or negative (0) label from the anchor labeling process before.
- $t_i$ is a 4-element vector containing the parameterized coordinate prediction from the *reg* layer and consists of

  $$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a, \quad t_w = log(w/w_a), \quad t_h = log(h/h_a)$$

  where $x_a, y_a, w_a, h_a$ denote the anchor's coordinates, width and height. $x, y, w, h$ refer to the actual bounding box that is to be predicted. Altogether, $t_i$ represents a scale-invariant translation with log-space height and width shift.
- $t_i^*$ represents the relative coordinate offset for an anchor with its associated ground-truth bounding box and is given by

  $$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a, \quad t_w^* = log(w^*/w_a), \quad t_h^* = log(h^*/h_a)$$

where $x^*, y^*, w^*, h^*$ refer to coordinates, width and height of the ground-truth box.

- $L_{cls}$ is the log loss / binary cross-entropy and given by

$$L_{cls}(p_i, p_i^*) = -\big(p_i^* \cdot log(p_i) + (1 - p_i^*) \cdot log(1 - p_i)\big)$$

for the softmax probabilities $p_i^*$ of the *cls* layer.

- $L_{reg}$ refers to the regression loss between predicted and ground-truth coordinate offsets with

$$L_{reg}(t_i, t_i^*) = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t_i - t_i^*)$$

and realized with the smooth $\text{L}_1$ loss given by

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| \leq 1 \\ |x| - 0.5 & \text{otherwise.} \end{cases}$$

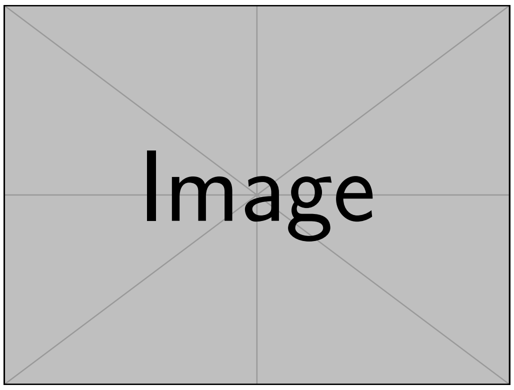
Due to $p_i^* L_{reg}$, the regression loss only contains positive anchor box regressions. Furthermore, both parts of the loss function are normalized by $N_{cls} = 256$, $N_{reg} \sim 2400$ and kept in balance by $\lambda = 10$. A detailed explanation for the chosen values is given in the reference implementation [33].

The approach effectively trains $k$ bounding box regressors simultaneously, with each regressor concentrating on one scale and one aspect ratio. Furthermore, for each image a mini-batch of 256 labeled anchors is randomly sampled. In order to prevent bias towards negative anchors, a positive-negative ratio of up to 1:1 is aimed for.

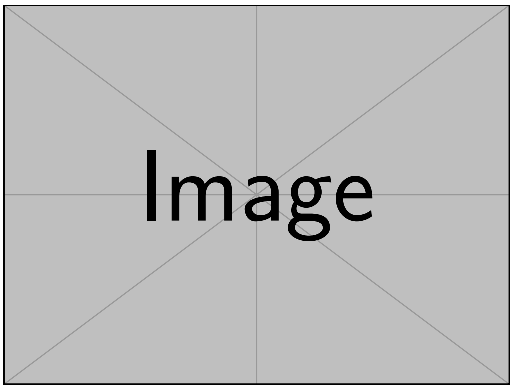
During test-time, non-maximum suppression (NMS) is as a post-processing step and used to filter out highly overlapping bounding box proposals: All boxes are compared with each other. If the IoU between two boxes exceeds 0.7, the box with lower *cls* score is removed from the result set. This process reduces the amount of region proposals to around 2000 per image. Finally, the top-$N$ highest score ranking proposals are reported, typically with $N = 300$.

### 2.2.2 Second stage – Fast R-CNN

The actual object classification and bounding box refinement is performed by the Fast R-CNN object detection algorithm that was introduced by R. Girshick in 2015 [7]. The author's motivation was to overcome certain drawbacks of region-based convolutional networks (R-CNN) [8]: Its pipeline consists of multiple stages that have to be trained consecutively. This process is expensive with respect to time and computational resources. Furthermore, object detection during inference is very slow



(a)



(b)

**Figure 2.2:** (a) Fast R-CNN architecture with RoI pooling and subsequent fully connected layers with classification and bounding box regression branch [7]. (b) shows an example for RoI pooling a $5 \times 5$ RoI window with a $2 \times 2$ non-uniform grid and max-pooling results for each RoI window cell [48]. (Images removed due to copyright reasons.)

because each proposal is evaluated separately by the feature extractor utilized in the R-CNN approach. In contrast, a Fast R-CNN network processes an image once with its backbone CNN but still relies (like R-CNN) on rectangular object proposals from an arbitrary upstream proposal algorithm.

The Fast R-CNN approach utilizes a pre-trained CNN that is stripped of its last pooling layer. In its place, a *region of interest* (RoI) pooling layer is set up that resizes features from a corresponding object proposal into a fixed-sized feature map. In detail, the input proposals are projected onto the last feature map by a process called RoI projection [48]. It utilizes the sub-sampling ratio of the specific feature extractor in use and results in a RoI window, an area on the last feature map that corresponds to original proposal. Afterwards, the process of RoI pooling is accomplished by dividing the RoI window into a $H \times W$ grid and max-pooling the values of each RoI window cell. The process is applied for each feature map channel separately.

The result of the RoI pooling layer is flattened and processed by subsequent fully connected layers that branch into two sibling networks. The first is responsible for predicting the correct class of each RoI. This is accomplished by applying the softmax function over $K + 1$ outputs (for $K$ classes + background) of the last FC layer. The other sibling network performs regression for bounding box offsets relative to an object proposal for each of the $K$ classes. This approach is similar to the bounding box regression of the previous RPN and results in parameterized coordinates as described in [7] [8]. In contrast to the former, the offsets are relative to the coordinates and extents of the object proposal instead of anchors.

In order to train the classification and regression branches of a Fast R-CNN model, each RoI is labeled with the ground-truth class $u$ and the bounding box regression target $v$. In detail, proposals that have an IoU $\geq 0.5$ with a ground-truth bounding box are labeled with the respective ground-truth class ($u \geq 1$). Proposals with IoU $\in [0.1, 0.5)$ are labeled as background ($u = 0$), all others are ignored. The multi-task loss

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda \, [u \geq 1] \, L_{reg}(t^u, v)$$

is used in order to mutually train classification and regression. In detail, the stated terms have the following meaning:

- $p = (p_0, \dots, p_K)$ is the discrete probability distribution obtained by the softmax function over outputs of the classification branch.
- $u$ is the ground-truth class label.
- $t^u = (t^u_x, t^u_y, t^u_w, t^u_h)$ is the predicted tuple of parameterized coordinates for class $u$ from the regression branch.
- $v = (v_x, v_y, v_w, v_h)$ is the ground-truth regression target for a RoI. The set of all regression targets is normalized so that the individual components $v_i$ have zero mean and unit variance. (It is not explicitly stated that targets are calculated as offsets between the proposal and its corresponding ground-truth box, but it is safe to assume.)
- $L_{cls}(p, u) = -log(p_u)$ refers the log loss of the prediction $p$ for the true class $u$. This special case is also known as categorical cross-entropy loss and consists of a single term due the fact that only one class is true (*one-hot*).
- $[u \geq 1]$ is the Iverson bracket indicator function and ensures that the regression loss is only evaluated for foreground classes and ignored for object proposals that do correspond to background.
- $L_{reg}$ refers to the regression loss between predicted and ground-truth coordinate offsets with

  $$L_{reg}(t^u, v) = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t^u_i - v_i)$$

  and realized with the smooth L$_1$ loss given by

  $$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| \leq 1 \\ |x| - 0.5 & \text{otherwise.} \end{cases}$$

- $\lambda$ is a hyper-parameter that can be used to balance both parts of the loss function but is usually set to 1.

Mini-batches for training are utilized and typically consist of $2 \times 64$ proposals from two different images. The proposals are randomly sampled and a fraction of $1/4$ foreground samples ($u \geq 1$) is aimed for.

Post-processing during test-time consists of non-maximum suppression (analogous to the first stage RPN) that is applied independently for each class. Finally, the top-$N$ ranking detections from the union of all classes are returned as the final result.

### 2.2.3 Mutual training of both stages

The authors of Faster R-CNN propose a 4-step alternating training in order to learn the combined network consisting of RPN and Fast R-CNN with shared convolutional layers [33].

1. The backbone CNN of the RPN is initialized with weight from a pre-trained classification model and the RPN's unique layer are initialized with a zero-mean Gaussian distribution with standard deviation 0.01. It is trained end-to-end for the region proposal task independently from the Fast R-CNN stage.

2. The Fast R-CNN detector is also trained independently by initializing its backbone with a pre-trained classification model and its unique layers according to [7]. It is trained end-to-end using proposals from the RPN of the first step. So far, both network do not share convolutional layers.

3. The trained Fast R-CNN network from step 2 is used to initialize the RPN backbone. The weights of the convolutional layers are now frozen and only the unique layers of the RPN are fine-tuned. Both networks now share the same backbone.

4. With still fixed convolutional layers, the unique layers of the Fast R-CNN network are fine-tuned with proposals from the updated RPN of step 3.

Altogether, the applied steps result in a jointly trained and unified Faster R-CNN object detection network. The used implementations adopts this training procedure with only minor discrepancies [10]. Greater differences are actually noticeable for the actual realization of the RoI pooling by a special crop-and-resize operation that performs a more sophisticated bilinear interpolation.

## 2.3 Datasets for PCB analysis

In recent years the development of specialized datasets for PCB analysis is fanned, especially due to the “hunger for data” of modern machine learning approaches. This enables prototyping, testing and deploying of new models and algorithms without the necessity of gathering own datasets from scratch - or at least not depend solely on it. Moreover, self-collected data has to be preprocessed and labeled, which can be time-consuming and error-prone. Finally, publicly available sets can function as a standard for the research community to compare the effectiveness and performance of their newest methods. This part presents the findings of publicly available datasets concerning PCB component analysis during the literature research phase as well as two self crafted collections from the previous work. The requirements are that images contain printed circuit boards and at least have labels and positional annotations for integrated circuits. Datasets that solely contain images of cropped PCB components (eg. for defect detection and classification) are omitted.

### 2.3.1 PCB-DSLR

Pramerdorfer and Kampel [30] presented in 2015 one of the first publicly available datasets for vision-based PCB analysis, application-related to waste recycling. The authors describe briefly, how scientific publications at that time have to rely on datasets that are either not accessible, very small, have insufficient information (labeling, general description) or consist of low-quality images. In order to leverage this situation, a collection of 165 different printed circuit boards are extracted from a waste conveyor belt. The PCBs are sampled randomly and originate mostly from consumer electronics (e.g. PC motherboards, displays) but also some industrial embedded hardware. The image acquisition system consists of a Nikon D4 DSLR camera mounted above the belt at a height of 107 cm, using a 60mm f/2.8 lens with polarization filter. Opaque curtains block outside light and white LEDs with diffusion and polarization filters illuminate the scene accordingly, in order to suppress reflections on the boards. Each board is represented by multiple images with 3 to 5 different orientations (most of the time 0°, 90°, 180° and 270°), resulting in a total of 748 images with a fixed resolution of $4928 \times 3280$ pixels (3:2 aspect ratio). For each image segmentation information for the depicted PCB itself as well as orientated bounding boxes for ICs are supplied. This nets to a total of 9313 annotations for all 748 images, but reduces to 2048 unique labeled ICs with respect to the 165 original images. The authors provide a histogram of the amount of ICs per PCB, revealing that most boards contain 20 or less components but with exceptions up to 135 possible objects.

Although 1740 of the 9313 object samples have additional textual information (e.g. manufacturer identification or part number) the collection itself has to be considered a *single class* dataset. Furthermore, the set is redundant due to its pre-augmented nature with respect to rotation which could be achieved by other pre-processing means if required. Due to its origin, the depicted boards can be polluted by dust deposits – especially on the pins of smaller ICs – which could pose an obstacle for closer analysis. In addition, a certain portion of components are defect, especially connectors with bend and twisted pins are common. Finally, the dataset primarily consists of scrapped legacy hardware from the 2000s and early 2010s years, for example a substantial amount are older THT ICs. But still, the dataset is a first central source to build upon for PCB analysis.

### 2.3.2 PCB-METAL

Mahalingam et al. introduced in 2019 a new PCB dataset for computer vision based analysis called *Micro Electronics Taken Apart Logically*. The authors' motivation is identical to that of Pramerdorfer and Kampel [30], yet their field of application is automatic optical inspection during manufacturing processes. Unlike the PCB-DSLR data, their goal is to deliver image data clear of dust or other debris in order to train models that produce accurate detections. Their images are taken from a wide variety of electronic devices, from computers to cellphones. The image acquisition system consists of a Canon EOS 5D Mark II DSLR camera in combination with a sophisticated lightning system above a fixed white board, the height is not stated. The obtained dataset consists of 123 unique PCBs, photographed from the front and back side with 4 different orientations (always 0°, 90°, 180° and 270°), resulting in a total of 984 images. The authors provide bounding box annotations for 12240 objects that emerge from four different classes of PCB components, in detail 5844 ICs, 3175 capacitors, 2679 resistors and 542 inductors on all 984 images. Statistics for all SMD components are provided, most PCBs contain 20 or less ICs with some exceptions between 60 and 70. Histograms for the remaining classes follow the same distribution pattern with different numbers of occurrences.

Introducing a multiclass dataset is a move in the right direction for modern PCB analysis. The authors follow a fourfold principle of multiple rotated images per board similar to the PCB-DSLR dataset, arguing that a complete visualization of all components from all sides should be provided. This could indeed offer a more complete view on components that are located on the outer edges or have an increased height due to the small but still existing effect of perspective projection during image acquisition. Surprisingly no resolution information for the acquired images is stated, leaving it to the reader, that the used camera system is likely to produce images up to $5616 \times 3744$ pixels.

Until now, the dataset is not available despite the authors' claim to provide an open access dataset – and formal inquiries during this work still remain unanswered.

### 2.3.3 FICS-PCB

Lu et al. from the Florida Institute for Cybersecurity (FICS) at the University of Florida presented in 2020 another dataset for the automated visual inspection of PCB boards [24]. Due to the fact that still only few large-scale datasets are publicly available, the authors proposed a new general purpose dataset for visual inspection of PCBs. One of the goals is to support evaluation on different challenging tasks, like defect detection and component classification. A total of 31 PCBs are used for creating the dataset, ranging from hard drive controllers to display boards and covering a variety of various substrate colors. Therefore, two kinds of image data are gathered, using different acquisition systems, a Leica DVM6 digital microscope for close up, highly magnified images and a Nikon D850 DSLR camera for general purpose footages. The digital microscope has a fixed lens with a movable stage, taking several images per board, each image with a size of $1600 \times 1200$ px. All shots are collected with 3 different illumination settings applied to the build-in ring light and three different scale settings ($1\times$, $1.5\times$, $2\times$) that control the microscope's *field of view*. By this means a total of 8634 close-up images are collected. The DSLR camera is equipped with a $105mm$ macro lens and mounted on a tripod. Instead of using any kind of zoom, images are taken with variable distance to ensure that most boards are captured in one image. For some larger PCBs several shots with board shifts are taken to ensure the accurate capturing of small size components. Boards are placed on a white underground, no additional lightning system or special polarization filters are used in the DSLR setup. Images of the back-side that do not contain any sought components are excluded, resulting into a total of 51 images with a resolution of $8256 \times 5504$ pixels. The authors provide annotations for each image, including bounding box coordinates, component type, optional text on component and logo if available. The sought components are grouped into seven classes: ICs, capacitors, resistors, inductor, transistor, diode and other. For the entire dataset a total of 75965 labeled objects are listed, with 2971 ICs and a majority of capacitors and resistors (each $> 30$k). With respect to the DSLR subset, exact numbers per category are not provided and the total object count is approximately 6834 derived by the mean component density of 134. 11 boards show more than 200 components and one contains up to 478.

The authors successfully introduced a new multiclass dataset with PCBs of modern electronic devices with higher component density, containing more categories than previous attempts. Furthermore the double tracked approach with two subsets is intriguing and should meet the needs for different areas of application. Contrary to other datasets, the provided annotations omit the surrounding pins. This could lead to inconsistent training and inference with respect to bounding box predictions, if mixed with other datasets. Despite the height-adjustable setup, various images of the DSLR subset contain a considerable amount of empty, white border areas at the periphery, which could be prevented by applying zoom or cropping. As of 16. Juli 2020 the dataset is publicly available on Trust-Hub [23].

### 2.3.4 PCB WACV-2019

Kuo et al. introduced with [12] a new dataset for training and evaluation of their approach on PCB component detection in context of the *Winter Conference on Applications of Computer Vision*. According to the authors, 15 images are sampled from the internet and 33 recordings originate from an unspecified DSLR or industrial camera, the released dataset has one less image (47 in total). All images are captured from 29 unique PCBs with a top-down view from above without rotational variants, 15 boards are photographed from both sides. The images differ with respect to various conditions, like lighting and resolution, still the overall quality – as well as the topicality of the hardware – is comparable to FICS-PCB [24]. The image resolution ranges from 600 × 600 up to 6000 × 4000 with an average of 3268 × 2253 pixels.

The novelty of the dataset is its vast amount of differently labeled components with a total of 31 classes, ranging from small parts (capacitors, resistors) up to large components (displays, heatsinks, transformers), covering a so called *bill of components/materials*. A total of 62168 different parts are labeled with bounding box annotations in Pascal VOC format, resulting in an average number of 500 components per image. The authors emphasize on the skewness of the dataset due to its highly imbalanced class occurrences. The information presented here are only available through the additional supplementary materials via arXiv [13] and the corresponding homepage [14].

### 2.3.5 IU-PCB

Rezaa et al. at Indiana University mentioned only incidentally the used dataset during their work with respect to IC detection on PCBs [34]. It is stated, that 483 images are collected through Google Image Search and subsequently labeled, resulting in about 5000 annotated ICs with bounding box locations. No further information is provided.

The actual dataset contains 599 images (with 7426 labeled ICs) of various consumer electronics (PC mainboards and graphic cards), but also older industrial hardware and power supplies with a large number of THT components. The image resolution ranges from 400 × 229 up to 8920 × 4647 with a mean of 1765 × 1310 pixels. A closer look reveals that the first 165 images actually originate from PCB-DSLR [30]. Nevertheless, the dataset can be seen as an acceptable supplement to the scarce set of available datasets.

### 2.3.6 SW PCB-Google and PCB-Custom

On the matter of chip detection, the preceding work at SCHUTZWERK [43] devised two new datasets. The first, called PCB-Google, is designed to amend the training process, due to the lack of publicly available data (only PCB-DSLR accessible at that time). By utilizing Google Image Search and subsequent manual filtering, 190 suitable images are collected. The depicted boards differ substantially with respect to hardware types (consumer electronics, embedded hardware and developer boards), camera perspectives and PCB substrate colors. Image dimensions range from 201 × 180 to 5352 × 3568 with an average of 893 × 638 pixels. A total of 1100 ICs are labeled (single class) and annotated with polygon location information, that enclose the components more precisely than bounding boxes. For this task, the web-based demo-version of the annotation software Labelbox [2] is used.

The second dataset is called PCB-Custom and as a test set its sole purpose is to evaluate the performance of the proposed object detection methods. It consists of 15 selected PCBs. In contrast to other datasets, PCB-Custom also contains back-sides of PCBs that do not contain any components in order to test explicitly for false detections. Due to their sensible nature, they are not used in cloud-based training during the work but instead for local evaluation only. The resolution of the resulting images is $6000 \times 4000$. Due to confidentiality reasons, the usage of web-/cloud-based labeling tools is not applicable. So with the help of the open-source software LabelImg [3], a total of 53 ICs are labeled and annotated with bounding box location information. All things considered, the dataset consists of relevant images of modern embedded hardware that cover precisely the application's scope, its quality is comparable to FICS-PCB and PCB WACV-2019.

[2]LabelBox: `https://labelbox.com/`

[3]LabelImg: `https://github.com/tzutalin/labelImg`

| **Name** | **Year** | **#PCBs** | **#Images** | **Resolution(s)** | **#Objects** | **#Classes** | **Annotation format** | **Remarks** |
|---|---|---|---|---|---|---|---|---|
| PCB-DSLR | 2015 | 165 | 748 | $4928 \times 3280$ | 9313 | 1 | Text, oriented bbox | legacy hardware |
| PCB-METAL | n/a | 123 | 984 | n/a | 12240 | 4 | n/a, bounding box | not available |
| FICS-PCB | 2020 | 31 | $8634^{*}$<br>$51^{\dagger}$ | $1600 \times 1200$<br>$8256 \times 5504$ | 6834<br>69131 | 7 | CSV, bounding box | modern PCBs, pins excluded from bboxes |
| WACV-2019 | 2019 | 29 | 47 | from $600 \times 600$ to $6000 \times 4000$ | 62168 | 31 | XML, bounding box | modern PCBs, vast amount of classes |
| IU-PCB | 2020 | n/a | 599 | from $400 \times 229$ to $8920 \times 4647$ | 7729 | 1 | XML, bounding box | legacy hardware, mixture of [30] and other sources |
| SW-Google | 2019 | n/a | 190 | from $201 \times 180$ to $5352 \times 3568$ | 1100 | 1 | JSON, polygon | obtained through search engine, used for training[43] |
| SW-Custom | 2019 | 15 | 15 | $6000 \times 4000$ | 53 | 1 | XML, bounding box | internal dataset for test[43] |

**Table 2.1:** Summary and key points of presented datasets. $^{*}$ refers to microscope and $^{\dagger}$ to DSLR subset.

# 3 Methodology

## 3.1 Object detection architecture

### 3.1.1 Selected framework and model

At least two well-established open-source frameworks for object detection based on CNNs are currently available, namely *Detectron2* [46] and the *TensorFlow Object Detection API* (ODA) [10][11]. These projects offer efficient and tested implementations of modern object detector meta-architectures, a variety of models pre-trained on large scale datasets and sophisticated interfaces for configuration, data input, augmentation and evaluation.

The first framework is a relatively new ground-up rewrite of the *Detectron* object detection framework, with PyTorch instead of Caffe2 as its underlying deep learning framework. As of now, it offers 9 different classes of object detectors with at least 50 pre-trained models in its model zoo.

The second framework, ODA, is one of the 16 sub-projects under the research-tree of the official *Model Garden for TensorFlow*[1]. The maintainers of ODA provide implementations for 10 different object detectors in form of TensorFlow 1.x implementations and 6 different detectors for TensorFlow 2, altogether with more than 80 pre-trained models. This split is due to the transition within the ODA framework to the new major version of the underlying deep learning framework [31] , a process that went on till July 2020.

A third possible framework with object detection capabilities is the *Microsoft Cognitive Toolkit* (CNTK)[2]. It offers two implementations of object detection algorithms (Fast and Faster R-CNN) with at least two pre-trained models. Documentation is limited to small sections under tutorials/examples and is unchanged since 2017.

From all the mentioned object detection frameworks, ODA and its TensorFlow 1.13 implementation has been selected for the following reasons:

- It builds upon the TensorFlow machine learning platform and programming concepts, by which the author is familiar with.
- The experience collected during the progenitor work, which also used this framework and its version, can be benefited from and built upon. Results can be

[1]Model Garden for TensorFlow: `https://github.com/tensorflow/models`
[2]Microsoft Cognitive Toolkit: `https://docs.microsoft.com/en-us/cognitive-toolkit/`

compared appropriately between both works.

- At the beginning of this work, a TensorFlow 2 version of the desired Faster R-CNN object detector was not available in ODA.

There are also some drawbacks to this decision. The framework itself is not light-weighted, it consists of about 300 Python modules with more than 350 classes (including abstract and test-driven ones). Documentation is limited to tutorials and examples and complemented by source code comments, Github issues within the framework's repository and discussions on Stack Overflow. Finally, multi-GPU training is not functioning appropriately in version 1.13 of ODA.

From all the object detectors available in the ODA model zoo, the Faster R-CNN model has been chosen. It often served as a baseline for comparison between architectures in scientific works with good detection performance ([12][15][25][34]) and as influence for follow up works (see [10] for a detailed listing). The progenitor work [43] also reported best results (0.874 AP[3]) with it on the task of single class chip detection.

In detail the *Faster R-CNN Inception-Resnet-v2-atrous* model is selected for this work. It was pre-trained and evaluated on the COCO dataset with 0.37 mAP[4] by the authors of ODA. A thorough analysis and comparison with respect to performance, accuracy vs. time trade-offs as well as memory consumption is available in [10]. The important keynotes are, that the performance of object detection correlates with the classification performance of the used feature extractor while time and memory consumption rise for more complex models. Due to non-existing constraints of a real-time application, time for training or inference is not an issue for this work. However memory consumption is a factor that should be considered carefully due to the GPU memory limitation of the available hardware. The used feature extractor *Inception-Resnet-v2* is described with a total of 54,336,736 parameters and memory usage of the Faster R-CNN model is stated between 2 and 9.5 GB on low resolution image data (300×300 px). Its sub-sampling ratio from input to the last convolutional feature map is $^1/_8$.

### 3.1.2 Custom parameter choices

The ODA uses a central configuration file which describes the object detection pipeline for a training and evaluation process. Most parameters have default values but can be set explicitly to ones needs. All possible parameters as well as their default values are spread out in 28 different configuration files and are overwritten by the choices made in the central configuration file. The final file consists of five sections that will

[3]single-class average precision, AP@50

[4]mean average precision, mAP@50:5:95 over 10 IoUs

be described briefly for the chosen model. The listed parameters within the section are only an excerpt but represent the more notable changes with respect to standard configurations that are provided as examples by the framework.

**Model** This section defines what kind of object detection architecture (Faster R-CNN, SSD, etc.) and underlying feature extractor will be used. Furthermore, various parameters for fine-tuning the different stages of an object detector can be set. Also the number of classes is defined in this section. The more influential configuration choices for this section that have been adjusted in this work are as follows:

- `image_resizer` allows to resize input images to a fixed target size or a range of sizes for the input layer of the CNN. The 2nd choice (called `keep_aspect_ratio_resizer`) effectively preserves the aspect ratios of the input images while guaranteeing a minimal length for the smaller image dimension and a maximal length for the larger. Variable sizes of input images result in varying spatial dimensions of the model's feature maps, while the amount of convolutional filters stay the same (and therefore the number of resulting channels of the feature maps). For large inputs, this results in high memory requirements for TensorFlow's underlying computational and derivative graphs required for back-propagation.
  This work adapts the choice from the COCO pre-training configuration with `keep_aspect_ratio_resizer`, re-scaling images to a minimal dimension of 600 px and a maximum of 900.[5]

- `1st_stage_anchor_generator` defines size, stride, scales and aspect ratios of the generated anchor box proposals for the 1st stage of the Faster R-CNN model. Default values for size ($256 \times 256$ px) and stride (8 px)[6] are adopted from the example configuration for pre-training with COCO. Adding more scales is beneficial for covering wider differences in object sizes and more aspect ratios will help with elongated objects. Therefore, the given aspect ratios $[1/2, 1, 2]$ are amended by $[1/4, 4]$, and $1/8$ is added to the default scale factors $[1/4, 1/2, 1, 2]$. The combinatorial growth of anchor box proposals results in increased memory usage.
  Extending aspect ratios with $[1/8, 8]$ is possible in conjunction with reducing anchor box scales or adjusting the used input image resizer due to limited memory of the available GPUs (Section 3.1.5).

- `first_stage_max_proposals` is the maximum number of proposals generated by the 1st stage and defaults to 300.

[5]Aspect ratio as well as the the max. dimension restriction will be kept at all cost, resulting in a min. dimension lesser than 600 px if necessary.

[6]The 8 px stride setting is dictated by the earlier mentioned sub-sampling ratio $1/8$ of the feature extractor and should not be altered.

- max_detections_per_class and
  max_total_detections steer the final amount of detections per class and across all classes by the 2$^{nd}$ stage box predictor of the Faster R-CNN architecture. Both default to 100 and have to be set accordingly to the amount of expected objects. in this case 200 for detections per class and 200 for the overall number of possible objects.

**Train-config** Detailed configuration for the object detection pipeline during training is set in this section.

- optimizer can be set to AdamOptimizer, RMSPropOptimizer or Momentum Optimizer. All examples and tutorials of the ODA as well as the progenitor work use the 3$^{rd}$ option with 0.9 as momentum value. The same configuration is also used in this work.

- learning_rate is an additional parameter for the optimizer. Four different schedulers, Constant, ExponentialDecay, ManualStep and CosineDecay are available with respective configurations. Following the configuration of the progenitor work, a ManualStep scheduler is chosen. The initial learning rate is $3.0 \times 10^{-4}$, decays after 25000 steps to $3.0 \times 10^{-5}$ and after 35000 steps to $3.0 \times 10^{-6}$.

- fine_tune_checkpoint specifies the checkpoint file that contains a pre-trained detection model which is used for transfer learning. It is pointed towards the COCO pre-trained model described in section 3.1.1.

- data_augmentation is explained in detail in section 3.1.3.

**Eval-config** Configuration that is used during evaluation cycles and results in extended event logs and information summaries that can be queried via TensorBoard, TensorFlow's visualization toolkit[7].

- metrics_set defaults to COCO detection metrics and is kept this way.

- include_metrics_per_category is set to true in order to get detailed evaluation metrics per object class. Without this option only mAP scores over all categories are reported (see Section 3.1.4 for detailed explanation).

**Train- and eval-input-reader** Both file sections describe configurations for the actual input readers that read the image and annotation data. They are described in one paragraph due to their identical structure.

[7]TensorBoard: https://www.tensorflow.org/tensorboard/

- `label_map_path` defines the path to an appropriate label map file, that specifies the mapping from textual class labels to integer ids internally used by the model. This file is created based on available class information of the used datasets.

- `num_readers` specifies the number input files to read in parallel and defaults to 64. We reduce the amount to 10 readers due to our limited amount of TF-Record files.

- `max_number_of_boxes` determines the maximum amount of bounding boxes read per image sample and is set to a default value of 100. This variable is not set in any example configuration and only brought to attention by carefully examining the `input_reader.proto` file where it is defined, or by inadvertently finding a corresponding Stack Overflow discussion [27]. The value should be set to the maximum amount of boxes in the input data, therefore it is set to 200 for this work. If left unchanged, all other corresponding changes in the model configuration will have no effect in detecting more than 100 objects per image.

- `tf_record_input_reader` holds the path(s) to TF-Record files that hold image and annotation data. Multiple paths and wildcards are supported, allowing to effectively combine multiple datasets without the need for manual merging.

In order to utilize the full potential of the Faster R-CNN object detector various parameters can be adjusted in order to satisfy the needs of specific use cases. As indicated above, only the most vital options for successfully instantiating a Faster R-CNN object detector with transfer learning for our use case are depicted. Still, various parameters are adjustable (e.g weight initialization, mini-batch sampling and NMS postprocessing) but kept to the ODA's dafaults which are in turn adopted from the reference implementations described in [7], [33] and [10]. An exception is the RoI pooling target size which is set to 17×17 instead of 7×7.

The matter of transfer learning deserves one closer look. Normally, the weights of a pre-trained model are adopted and immediately fixed. Then, only the unique layers (fully convolutionals in RPN, FCs in Fast R-CNN) are fine-tuned with new training data. The authors of ODA seem to diverge from this strategy, generally allowing adjustments for all layers during training [38]. It is stated, that freezing the lower shared convolutional layers could be enforced, but it is not recommended. The experience seems to be that it is neither beneficial for training time nor detection accuracy. Therefore, this work adopts the recommendation and omits any weight fixation of the pre-trained COCO model.

### 3.1.3 Preprocessing and augmentation

**Preprocessing** In order to work with large datasets of image and annotation data, it is advisable to utilize a binary file format that can significantly boost performance for the input pipeline of a deep learning model. Binary data uses less space, can be read faster and more efficiently from disks. As a consequence, training and evaluation time of a model can be optimized if the right amount of data is delivered without delay.

TensorFlow offers its own binary file format called *TFRecord* which is seamlessly integrated into data readers and input pipelines of the TensorFlow framework. Naturally, ODA adopts this approach and provides conversion mechanisms from various standard dataset formats (e.g. Pascal VOC, COCO) into TFRecords. Consequently, the previously mentioned train- and eval-input-readers expect this format for training.

Common practice is to shard data across multiple TFRecord files in order to benefit from parallel I/O capabilities of modern systems. A rule of thumb is to generate at least 10 times as many files as there are parallel TensorFlow sessions reading. At the same time, each TFRecord file should have at least 10 MB and ideally more than 100 MB in order to fully benefit from I/O prefetching [40]. The current work follows this advice and incorporates an additional re-scaling operation within the ODA converter mechanism for COCO annotation format to TFRecords. All images and bounding box information are re-scaled near the target input dimensions (largest dimension set to 900) while maintaining aspect ratios. This leverages problems with huge TFRecord file sizes as well as subsequent computational cost and memory consumption for resizing operations during training due to large input images (e.g. 6000×4000 px).

**Data augmentation** The performance of CNNs is closely related to the amount and variety of image data used for model training. This poses a thread to applications with limited access to image data – even if transfer learning is used. If the amount of provided data is too small the algorithm will eventually learn these few and specialized samples well but cannot generalize to new and unseen data. This behavior is called *overfitting* and can be detected by closely monitoring the validation or testing performance after each epoch of training [37].

The process of data augmentation is used to circumvent this problem. It aims at generating new instances of image data by applying automated image processing algorithms on existing datasets. Thus, the amount of available images for training is increased, effectively enlarging smaller datasets. Large datasets also benefit from data augmentation by bringing in distortions and variations that make the learned model more robust.

Instead of performing the desired transformations manually, the task is delegated to the ODA framework. It offers a variety of augmentation options that can be enabled and adjusted through the pipeline configuration file. Moreover, these built-ins will not only take care of the input images but will also apply any transformation to the annotation information of the contained objects as well. A brief summary of the chosen methods is depicted in Table 3.1.

| Method | Description |
|---|---|
| `random_horizontal_flip` | Horizontally flips image and objects |
| `random_vertical_flip` | Vertically flips image and objects |
| `random_rotation90` | Rotates image + objects by 90 deg. counter-clockwise |
| `random_crop_image` | Randomly crops image and bounding boxes:<br>- crop must contain at least one object<br>- aspect ratio must lie within $[1/2, \ldots, 2]$<br>- size/area must be $[1/4, \ldots, 1]$ of orig. image<br>- orig. and cropped bboxes must overlap by 0.3<br>- applied 50% of the time (explicitly set) |
| `random_image_scale` | Resizes image + objects by $[1/2, \ldots, 2]$ (aspect ratio preserved) |
| `random_adjust_brightness` | Changes brightness by up to 0.2 (clipped at $[0, 1]$) |
| `random_adjust_contrast` | Scales contrast by a value between $[0.8, \ldots, 1.25]$ |
| `random_adjust_hue` | Alters hue by a value of up to 0.02 |
| `random_adjust_saturation` | Changes saturation by a value between $[0.8, \ldots, 1.25]$ |

**Table 3.1:** Manually enabled augmentation methods for pipeline configuration of Faster R-CNN object detector. Every transformation has a 50% chance to be applied for each input image.

A 45 degree rotation augmentation is not available but would be a valuable asset in order to reliable detect rotated PCB components. An option for pure translation augmentation is also not available, but the first three augmentations in combination with random cropping should be able to produce a similar effect. Random changes of brightness and contrast should enhance robustness with respect to different illumination and lighting conditions of the used datasets. Finally, hue and saturation augmentations are selected in order to cope with varying background colors, in particular with those of the PCB substrate.

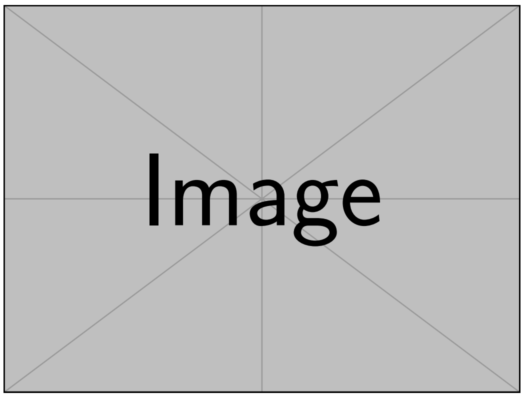
Augmentation operations are performed by the ODA exclusively with CPU resources and therefore do not impact GPU-based training performance.

### 3.1.4 Evaluation metrics

In order to asses the performance of an object detection approach, it is necessary to compare its results with available ground-truth data and determine a meaningful score. By counting correct and wrong results, metrics can be established that allow comparison for different training runs or with other models. Large scale dataset competitions, like Pascal VOC and COCO introduces their metrics and offer standard APIs for their calculation that allow rating and comparison of the submitted algorithms. This work will make use of the Python COCO API[8] and its metrics. The ODA is already intertwined with this API and makes it a prerequisite to install.

In order to understand the used performance indicators *per class average precision*

[8]COCO API: `https://github.com/cocodataset/cocoapi`


(a)

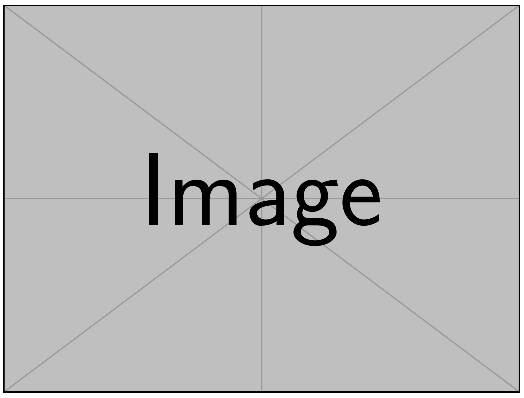

(b)

**Figure 3.1:** Visualizations for the concept of Intersection over Union [35]. (a) shows an example of predicted and ground-truth bounding boxes and (b) illustrates the calculation of IoU metric.

(AP) and *mean average precision* (mAP) the following section will explain its key concepts.

**Intersection over Union** The result of an object detector consists of a list of confidence levels, class labels, and bounding boxes for its predictions. While comparing class labels is straight forward, the comparison of bounding boxes is slightly more challenging. It is unlikely that a predicted bounding box matches exactly its ground-truth counterpart (Fig. 3.1a). Therefore, a standard metric is needed in order to measure the similarity between a predicted region and a ground-truth bounding box. A common approach for this is the *Intersection over Union*(IoU), also known as the Jaccard index. It is defined as the ratio between overlap (= intersection) and union of two areas (Fig. 3.1b), in case of this work rectangular boxes for prediction ($p$) and ground-truth ($gt$).

$$\mathrm{IoU}(\mathrm{bbox}_p, \mathrm{bbox}_{gt}) = \frac{\mathrm{area}(\mathrm{bbox}_p \cap \mathrm{bbox}_{gt})}{\mathrm{area}(\mathrm{bbox}_p \cup \mathrm{bbox}_{gt})} \quad \in [0, 1]$$

In combination with a chosen threshold $T \in [0, 1]$ the IoU allows a distinction between correct and incorrect predictions with the following three definitions:

- **True positive (TP)** is a correct prediction, that is corresponding well with the ground-truth bounding box ($\mathrm{IoU} \geq T$) and has the correct class label.
- **False positive (FP)** is a positive but incorrect prediction, which is
  - not matching any or corresponding inadequately with a ground-truth bounding box ($\mathrm{IoU} < T$)

    - reporting a wrong class label,
    - a subsequent (2nd, 3rd,...) prediction for an already correct predicted object.

- **False negative (FN)** is an undetected ground-truth object with no matching detection.

Due to the fact that there exists an arbitrary amount of bounding boxes that do correspond to non-existing objects, the concept of true negatives is not applied in the context of object detection.

A false positive is referred to as type 1 error and called *false alarm*. False negatives are type 2 errors and mentioned as *misses*. The combination of true positives and false negatives is the ground-truth known as *condition positives*. The sum of true positives and false positives is referred to as *predicted positives* and are all detections reported by the model.

**Precision and Recall** Upon the three basic building blocks TP, FP and FN two further metrics are established that allow closer insights about behavior and performance of an object detector model.

*Precision* is the ability of a model to predict only relevant objects and is also called *positive predictive value*. It is defined as the ratio between correctly detected objects and all positive predictions:

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} = \frac{\text{TP}}{\text{pred. pos}} = \frac{\text{TP}}{|\text{all detections}|} \in [0,1]$$

High precision means that few false positives are reported by the model while a low precision indicates a higher degree of false alarms. Thus, this metric shows how reliable the positive predictions of a model in fact are.

*Recall* is the ability of a model to find all relevant objects and is also called *true positive rate*. It is defined as ratio between TPs and all condition positives:

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{\text{TP}}{\text{cond. pos}} = \frac{\text{TP}}{|\text{ground-truths}|} \in [0,1]$$

High recall indicates few false negatives while a low recall means that the model is not performing well due to missing many objects. Therefore, the metric describes how good a model can truly detect all ground-truth data.

Both metrics have to be analyzed in order to asses the performance of an object detector. A model with high recall might detect almost every object reliably but also produce a high amount of false positives that lower precision. On the other hand, high precision can be coupled with low recall and therefore a higher degree of misses.

**Precision-recall curve** The trade-off between both metrics can be depicted by the *precision-recall curve* that puts them in relationship. First, for each class the predictions are sorted by the reported confidence score in descending order (Table 3.2). Predictions are categorized as TP or FP based on IoU and true class membership. For each row the TPs and FPs are accumulated and allow the calculation of precision and recall for the present course. Afterwards, precision can be plotted as a function of recall (Fig. 3.2).

| Confidence | TP | FP | Σ TP | Σ FP | Precision | Recall |
|---|---|---|---|---|---|---|
| 0.96 | 1 | 0 | 1 | 0 | 1 | 0.2 |
| 0.94 | 1 | 0 | 2 | 0 | 1 | 0.4 |
| 0.87 | 0 | 1 | 2 | 1 | 0.67 | 0.4 |
| 0.83 | 1 | 0 | 3 | 1 | 0.75 | 0.6 |
| 0.80 | 0 | 1 | 3 | 2 | 0.6 | 0.6 |
| 0.79 | 0 | 1 | 3 | 3 | 0.5 | 0.6 |
| 0.75 | 1 | 0 | 4 | 3 | 0.57 | 0.8 |
| 0.64 | 0 | 1 | 4 | 4 | 0.5 | 0.8 |

**Table 3.2:** Example for the process of precision-recall calculation for a specific class. 8 detections in a ranked sequence with 4 TPs and 4 FPs on 5 ground-truth objects are evaluated. For each row precision is calculated by $\frac{\Sigma\,\mathrm{TP}}{\Sigma\,\mathrm{TP}+\Sigma\,\mathrm{FP}}$ and recall by $\frac{\Sigma\,\mathrm{TP}}{|\mathrm{GT}|}$.

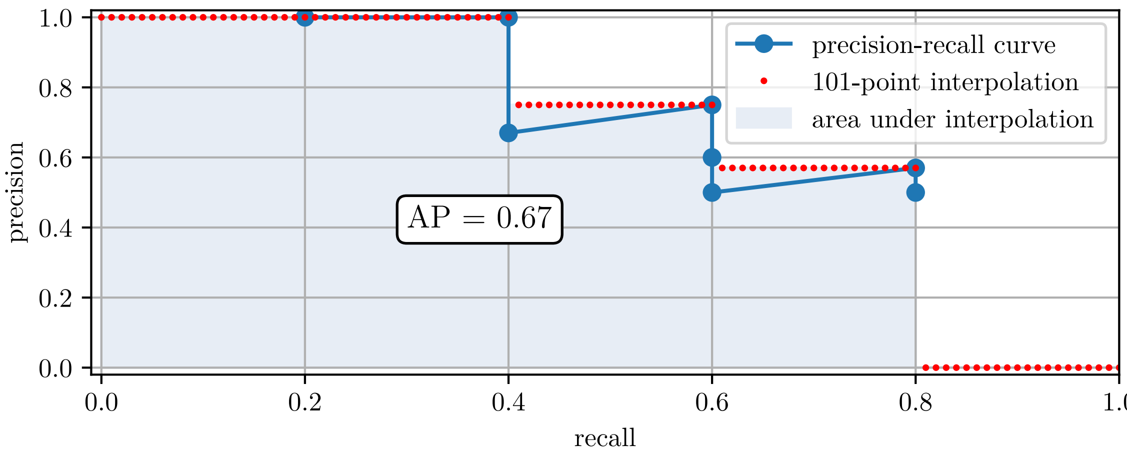


**Figure 3.2:** Visualization for example from Table 3.2. Precision is plotted vs. recall and a 101-point interpolation is applied. The area under interpolation and average precision score complement the example.

**Average precision** After calculation of the PR curve, the *area under curve* (AUC) depicts the *average precision* (AP). In order to ease the area computation, the zigzag-shaped curve is smoothed with an interpolated form. While the Pascal VOC 2008 metric proposes an 11-point (and its 2012 extension an all-point interpolation), the COCO API interpolates at 101 points. At each point, the maximum precision for current or higher recall values is sampled, effectively smoothing out notches of the curve (Fig. 3.2).

$$P_{\text{interp}}(R) = \max_{\tilde{R}:\tilde{R}\geq R} P(\tilde{R})$$

The average precision per class is given by the averaged sum of all interpolated precision values.

$$\text{AP} = \frac{1}{101} \sum_{R\in\{0,0.01,0.02,\ldots,1\}} P_{\text{interp}}(R)$$

Finally, the *mean average precision* (mAP) is calculated in order to measure the performance over all classes.

$$\text{mAP} = \frac{1}{N} \sum_{\text{c}\in\text{classes}} \text{AP}_{\text{c}}$$

The authors of COCO make no distinction between the notions of AP and mAP, assuming the difference is clear from context [20]. In this work class-specific results will be reported with AP and class-spanning ones explicitly with mAP.

So far, the calculation for AP is trimmed to one chosen IoU threshold. Pascal VOC uses only IoU = 0.50 for its prime metric (and 11- or all-point interpolation [28], depending on its version). The COCO challenge devises the averaging of mAP over 10 different IoUs ($[0.50, 0.55, \ldots, 0.90, 0.95]$) and simply refers to this as AP. The approach can be seen as a reward for detection algorithms that provide better localization performance. In order to avoid confusion this metric is explicitly referred to as AP@50:5:95. With this in mind, a direct comparison of Pacal VOC mAP and COCO AP should be done with caution. In order to allow better comparison, the COCO API also provides summaries for AP@50 and AP@75 with fixed IoUs.

Additionally, three *across scales* AP metrics for performance evaluation are provided, that limit the view on three different size ranges for detectable objects. $\text{AP}^{\text{small}}$ refers to objects with pixel area $< 32^2$, $\text{AP}^{\text{medium}}$ to $32^2 \leq \text{area} \leq 96^2$ and $\text{AP}^{\text{large}}$ to area $> 96^2$ [20]. Inconveniently, these metrics all build upon the @50:5:95 approach and are not reported for individual IoUs.

The main drawback of the COCO API is, that only summarized results over all classes are provided. Therefore, evaluation runs of the ODA framework do not support per category metrics and lack the ability to analyze detector performance for each class in detail. This restriction can be overcome by applying a community patch to the COCO API [49] and modifying the metrics receiving functionality of the ODA counter-part. Therefore, in addition to mAP this work can report per class AP with combined @50:5:95 as well as single IoU thresholds @50 and @75. Due to the insights of Huang et. al [10], AP@50:5:95, AP@50 and AP@75 seem to be strongly correlated.

Therefore, this work reports all performance measurements with per class AP@50 and mAP@50 in order to maintain clarity and comparability to the progenitor work.

### 3.1.5 Final setup and deployment

Free cloud-based ML training, for example offered by Google Colab or Amazon AWS, is a convenient and inexpensive way of training ML models that would otherwise exceed computational resources of ones own hardware. Yet, the situation regarding privacy and confidentiality of the uploaded data is often unclear and doubtful when it comes to *ML as a service*.

Up to four GeForce GTX 1080 graphic cards with 8 GB memory are available for training, depending on the utilization by other projects. The appropriate NVIDIA driver (440.82), CUDA- (10.1.168) and cuDNN-library (7.6.5.32-1) as well as Docker[9] (19.03.8) have been installed as a prerequisite for the following software.

In order to isolate the machine learning environment on the host system from other projects, a docker container is utilized. The corresponding docker image is based on the `tensorflow/tensorflow:latest-gpu-py3` base image and extended with TensorFlow-GPU 1.13, the corresponding version of ODA and its necessary dependencies [11]. Additionally, OpenCV (3.4.9.33) is installed for image manipulation tasks during data preprocessing.

Direct access to the physical host system is not granted. Instead, the docker container is made accessible via an internally running SSH daemon. Local port forwarding is used for tunneling specific ports between client and container over the secure connection, enabling access to web applications like TensorBoard or Jupyter Notebook running inside the container.

This kind of isolation is desirable in order to add stability to a high-performance server environment. A rogue TensorFlow session can consume vast amounts of CPU-resources, RAM and even swap for data preprocessing and augmentation. With appropriate resource limitations for the container in place, this has no to little effect on the stability of the host system.

## 3.2 Datasets

### 3.2.1 Acquisition of image data

This section briefly describes the used datasets and their adjustments before any labeling and annotation work takes place. In conjunction with Chapter 2.3 the first part complements the description on public datasets and which parts are used. The second part addresses the adaption of the internal datasets from the previous work as well as their extension by means of continuous image acquisition during this work.

[9]Docker: `https://www.docker.com`

#### 3.2.1.1 Public sources

**PCB-DSLR** The dataset is publicly available and offers a Python API for accessing the annotation information for each image. Among other things, it can be used for automated cropping of the PCBs, thus removing abundant black border areas that are present in all images of the dataset. The focus lies on the first recording of up to five for every board, therefore a pruning of the subsequent rotational duplicates is performed. This is done in order to avoid duplicate labeling work in the following steps. A reduced image dataset consisting of 165 unique images with 2048 annotations for ICs is used in this work.

**PCB-METAL** As stated in Section 2.3.2, the dataset is not publicly accessible at the moment. Therefore, this work cannot utilize it.

**FICS-PCB** The dataset is publicly available and offers two subsets of data, namely microscope and DSLR images. This work focuses only on the latter part which depicts PCBs as a whole. The DSLR subset contains 51 images. From a total of seven classes of PCB components, the annotations for the class IC with 449 bounding box information are extracted.

**PCB WACV-2019** The dataset is not processed due to its late discovery and limiting time constraints.

**IU-PCB** The dataset contains images that origin mostly from older consumer hardware. Therefore, it can be seen as a marginal enhanced version of the PCB-DSLR dataset [30] with 599 images and 7729 annotations for ICs. Due to limited added value and time constraints, this dataset is excluded from the work.

#### 3.2.1.2 Internal sourcing

**PCB-Google** The dataset PCB-Google of the previous work at SCHUTZWERK was sourced with the help of Google Image Search and has not been published so far. Therefore, it can be seen as an internal dataset with 190 PCB images and 1100 annotated ICs that are used in this work.

**PCB-Custom** The second dataset of the previous work is a confidential dataset that contains SCHUTZWERK internal boards and hardware. With its 15 images and 53 annotated ICs, it is merged with new image data acquired during this work and described hereafter.

**SW-Internal** The proposed dataset emerges from PCB-Custom by extending it with newly captured image data from internal projects. The used image acquisition system consists of a Sony $\alpha$6000 mirror-less camera with a 3.8× telephoto zoom lens, all-together mounted on a camera monopod (model 047) from Infuu Holders. This setup is assembled in the electronics laboratory of SCHUTZWERK.

First recording attempts have shown, that perpendicular lighting from above creates reflections on substrate areas of PCBs, even with diffuse light. Therefore lateral lighting is used to illuminate the recorded boards appropriately. Although setup and illumination conditions may not be as professional as described in some related works (e.g. PCB-DSLR with diffusion and polarization filters), they are sufficient in order to avoid reflections on the boards during image acquisition and ensure consistent illumination for all recordings.

87 images from the front and back of 44 unique PCBs have been captured (one image discarded due to insufficient quality). In contrast to all other datasets (except PCB-Custom), images of back-sides without any components are kept in order to test trained models for false detections. The final dataset contains 102 images in total and the resolution of all images is 6000×4000 px.

### 3.2.2 Annotation and labeling

The first part of this sections describes the criteria and decision process for selecting a suitable labeling tool. A thorough description for the PCB component categories introduced in this work is given in the second part.

#### 3.2.2.1 Selected software and customization

Finding a suitable free-of-charge or open-source software for the task of image labeling and annotation is not trivial. While lists like [29] offer a large variety of annotation software, the tools have to meet certain requirements and constrains for this work.

In order to ensure confidentiality for the internal image data of SCHUTZWERK, the software has to be an offline tool that is neither foreign hosted nor storing data in its vendor's cloud. Furthermore, the tool should provide some kind of project administration capability in order to maintain and process datasets independently from each other. Subsequently, the ability of importing datasets with existing annotations as well as exporting in a standard dataset format (e.g. Pascal VOC or COCO) should be part of the software's functionality. If that is not possible, the tool should at least have an internal data structure that is simple enough to allow the injection of existing annotations as well as extracting finished labeling tasks. Finally, the tool should provide the ability to accurately annotate arbitrary shaped objects with polygons instead of standard oriented bounding boxes (e.g. rotated PCB components or objects with perspective views). This requirement can seen as a future investment for

possible works that perform instance segmentation (e.g Mask R-CNN) on the labeled data.
The following listing briefly summarizes the evaluation results of different software tools based with regard to the requirements described above. Furthermore, the final candidate with its necessary adjustments is presented.

- LabelBox, a modern web-based annotation software used by the previous work for creating the PCB-Google dataset, cannot be used due to cloud-based storage of images by its vendor.

- LabelImg, the most rated labeling software on [29], is an offline tool that offers a simple GUI and plain directory based project administration. Annotations are stored in Pascal VOC or YOLO format on a per image basis without the need for an internal data structure. Import or export capabilities for other annotation formats are not included. The tool does not offer polygon based annotation, therefore it is excluded from this work.

- Sloth, an offline tool for image and video labeling offers a promising API for data import and export as well as the capability for polygon based annotations. Due to its outdated state (development ceased 3 years ago) and legacy library dependencies (Python2 in combination with QT4), the software could not be brought into a functional state during the course of this work.

- CVAT, an open source annotation tool specialized in image and video labeling, offers a locally running web-based environment with a sophisticated user interface for labeling, the possibility of polygon based annotations and even collaboration support with appropriate user management. Despite the first impression, tests on an early development version during this work revealed several shortcomings. Firstly, it does not offer the possibility for proper project administration and clear distinction between different datasets. Basic functionality, like adding further data or removal of duplicate images from an existing labeling task is not intended [5]. Despite of the documentation's claim on supporting various import and export formats, tests with valid examples of Pascal VOC, COCO and TFRecord formatted data have failed either for export, import or both. Due to the complexity of the software's code base, the possibility of writing custom import and export interfaces is not pursued. If the development of this software progresses with respect to more reliable project management and import/export interfaces, it could be considered a valid alternative to the chosen solution for this work.

The chosen candidate for this work is Label-Studio, an open-source multi-purpose labeling tool, running as a local web-application. Based on its simple configuration capabilities it can be customized for various labeling tasks with respect to images, videos, text and audio. It offers polygon based annotations for images and distinct project administration for multiple datasets. Import functionalities for pre-labeled datasets are absent, but the software's internal JSON based data structure is simple enough for subsequent modification with Python based scripting. During the course of this work, import functions for the used datasets have been developed, that convert the different annotation formats with various bounding box representations (refer to Table 2.1) into the polygon-based format of Label-Studio. In the current version of the software (v0.7), the export for polygon based annotations into the common COCO format is not functioning properly. Therefore, an export mechanism is developed that extracts and converts the internal information into the target format.

#### 3.2.2.2 Category decisions for multiclass labeling

So far, only the import of datasets with their annotations for the single class IC is accomplished. In order to establish an added value to this work, additional classes are introduced. Their purpose is either to enable sub-classification of ICs or complementing the dataset with additional components of interest. In collaboration with internal security experts at SCHUTZWERK, a selection of active and passive PCB components has been elaborated that is particularly interesting in the context of embedded system security assessments. The focus lies on a balanced trade-off between functional differences of the actual components and visual distinctiveness among the various classes. However, an in-depth discussion about the actual functionality of the different components is not part of this work.

**IC classes** The following categories describe different active components that are responsible for computation, signal processing and memory storage. They emerge from the single class IC used in the available datasets. The color of IC packages usually ranges from dark gray to black with matt or glossy variations. With respect to texture, most packages are imprinted with identification information (e.g. manufacturer, part number, etc.) in white or gray font.

- SOP stands for *Small-outline package* and is a category of surface mounted ICs. The main visual appearance of a SOP can be summarized as a dual rowed, flat chip. It has a variable but equal amount of pins on two opposite sides, varying from 8 to more than 60 pins in total. The structural shape of the chip package is rectangular with variable elongation, depending on the number of outgoing pins. Its component size, pin thickness and the spacing between them are variable and depend both on the chip's purpose and its technological era. In summary, it is the predominant type of IC encountered on most boards during this work.

- QFP is the abbreviation for *Quad-flat package* and is the four-sided equivalent to the SOP class. Also surface mounted, its main visual appearance is quadratic with equal amounts of pins on all sides, although there exist rare rectangular variants from early eras (e.g. in PCB-DSRL). The number of pins can range from 40 up to over 350 and primarily determines the package's size in conjunction with variable spacing between the pins. Generally, this class represents the 2nd most common type of IC in the described datasets.

- SON is the acronym for *Small-outline no-leads* (also known as *Dual-flat no-leads*) and is a dual-sided, smaller variant of SOP. Instead of pins that extend from the package sides, so called terminal pads on the lateral bottom of the chip provide electrical connectivity. A part of the soldering tin typically stands out from the sides to the solder pads of the board and is visible from a perpendicular view. Typically, SONs are significantly smaller than SOPs and have less connections, ranging from 4 to 14 in total. Depending on the number of pads, the components appear in varying rectangular shapes, typically less elongated than SOPs and in some cases almost quadratic.

- QFN is short for *Quad-flat no-leads* and represents the four-sided version of a SON package or a modern variant of a QFP without extending pins. With respect to connectivity, the previous statements for the SON class also apply for components of this category. The amount of pads on all sides can range from 8 to over 80 and influences the component's size. The appearance is consistently quadratic.

- BGA stands for *Ball Grid Array* and is the last type of the surface mounted component in this listing. In contrast to the previous described packages, a BGA's connection pads are exclusively located on the bottom surface. This allows for more interconnections than a dual- or quad-row package but conceals all soldering joints. BGAs are flat and have rectangular or quadratic shapes, their size varies and usually resides between larger SOPs and medium sized QFPs. Although direct electrical contacting is not possible, the classification information generated during a visual inspection is considered valuable in order to avoid unnecessary attempts of automated contacting.

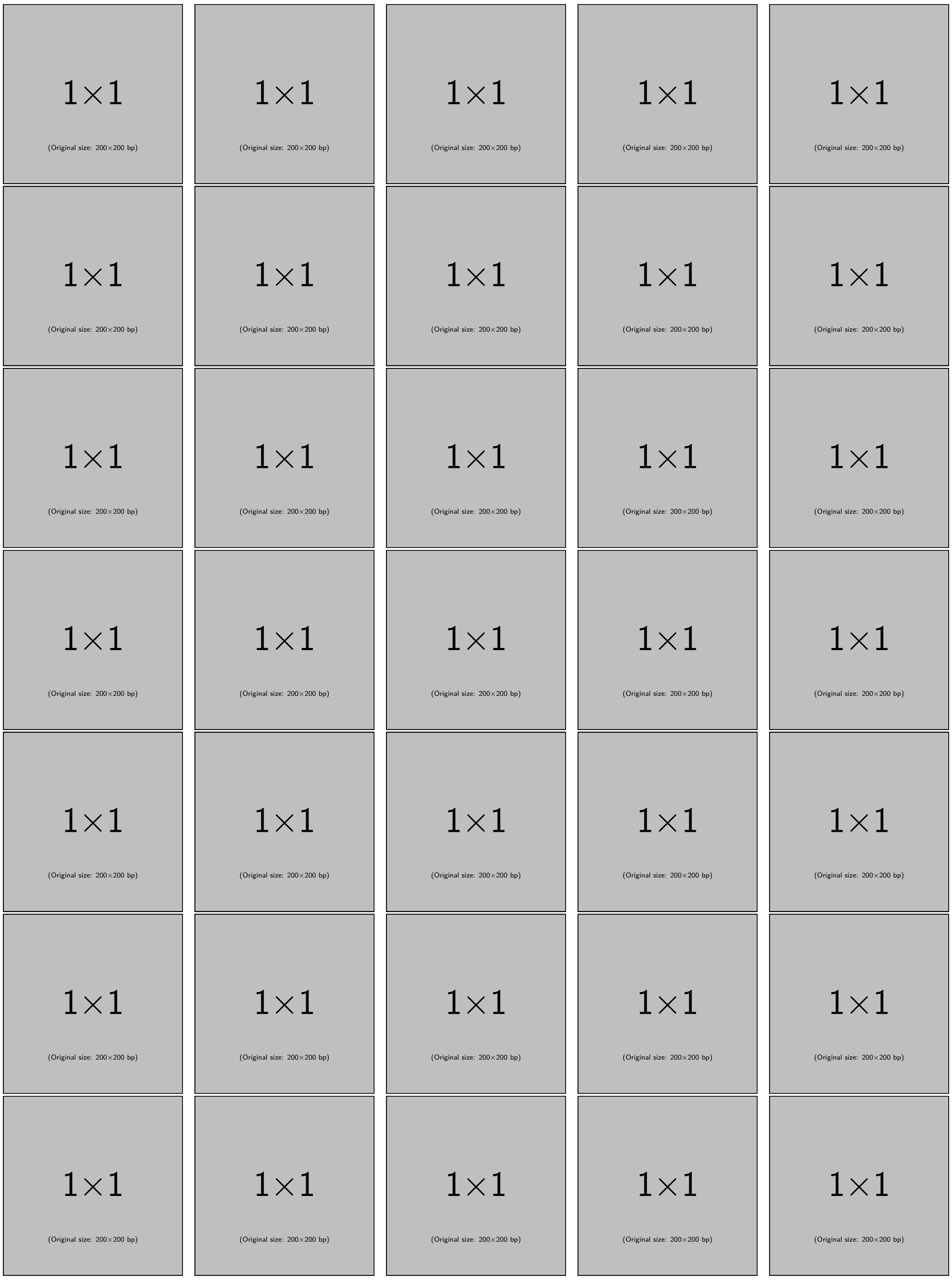
- THT is the abbreviation for *Through-hole technology* and refers to the mounting of chips by placing the pins into drilled holes and soldering them on the opposite side of the PCB. Alternatively, a socket holder can be installed first, in which a removable THT chip can be fitted in. THT based ICs have been virtually superseded by equivalent SMD components but can still be found occasionally. Typical THT chips are dual-rowed with an equal number of pins on both sides, their total amount ranges from 8 to 40. Size correlates with the number of pins. The shape ranges from quadratic to rectangular, in some cases particularly elongated. Their most noticeable visual feature is given by broader but also steeper bend pins. Compared to SMD components their elevated height is noticeable and can cause minor occlusions in the image data.

**Passive components** Besides active components, there exists are large variety of passive components that do not require additional power supply in order to perform their function. While this term also applies to resistors, capacitors and inductors, this work focuses on detecting and identifying connectors and probing points that serve as signal and debug interfaces.

- $\text{CONN}_{\text{THT}}$ is the designator for through-hole technology connector. Although the usage of active THT ICs declines, the respective connectors are still predominant and can be encountered on many PCBs. Female and male versions exist and are called sockets and headers. The latter can optionally be framed by a plastic guide box which functions as a housing. Single- or dual-row versions are most common, but cases of triple-row are occasionally present in the available datasets. The amount of pins per row (or pits for a female version) is versatile and usually ranges from 1 to 20 (dual-row) or even up to 40 (single row). Therefore, these components are highly variable with respect to elongation. Bent versions with a 90 degree angle exist and are usually present at the perimeters of boards. Headers and sockets can be colored arbitrarily, though black and white versions are most frequent.
  Besides the various topside appearances, two additional cases are covered by this category: Soldering joints of THT connectors on the opposite side of PCBs are considered valid probing points. The same applies for areas where soldering pads with drilling holes exist, but actual connectors have not been assembled. DVI and VGA ports are excluded despite being installed in through-hole manner.

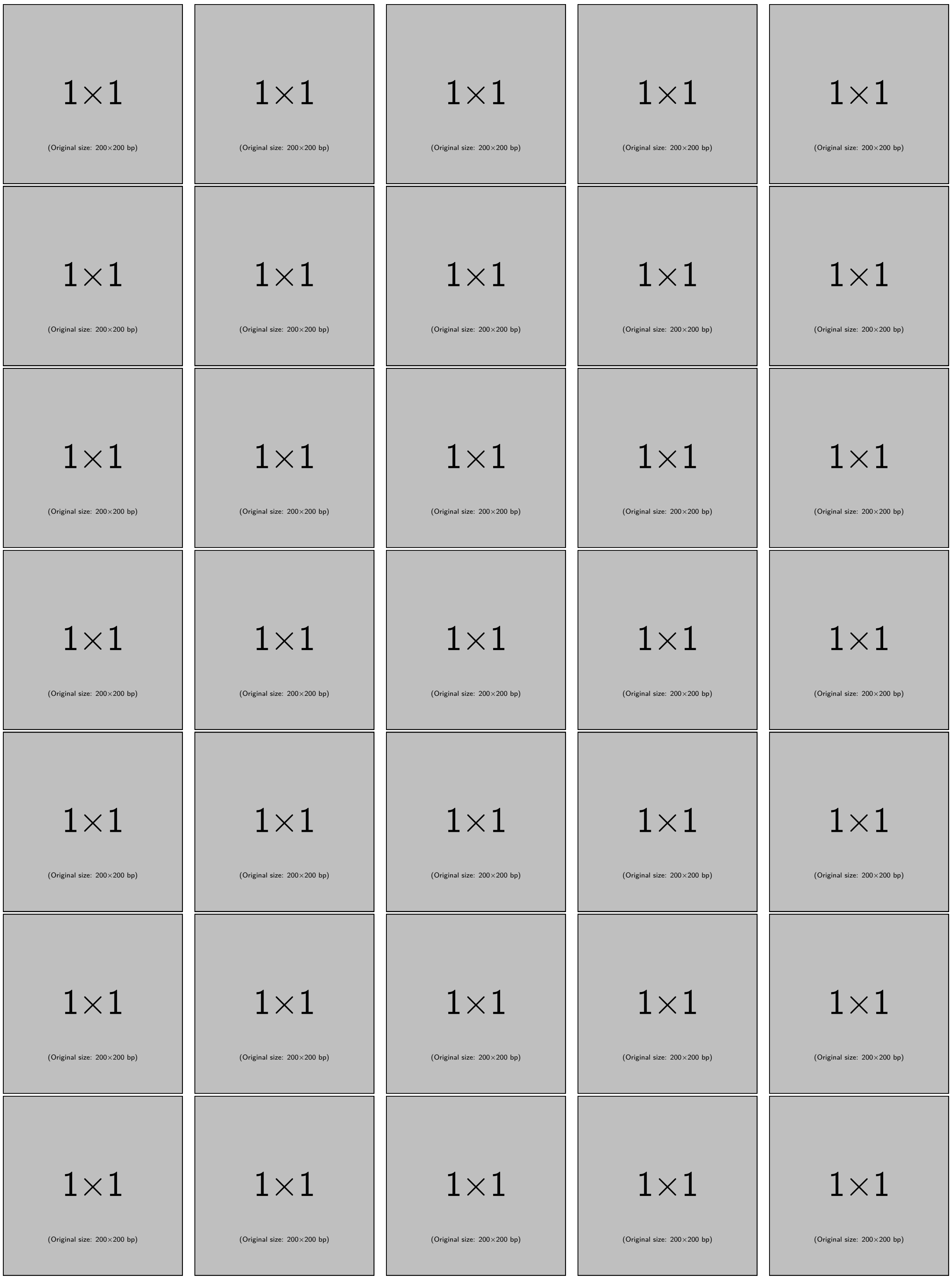
- $\text{CONN}_{\text{SMD}}$ is a surface mounted connector variant. Small pins extend from the sides and are soldered onto pads that reside on the same side of the board as the actual connector. Most remarks for THT connectors also apply to this category with some extensions. The amount of connecting pins on the sides can range up to 80 in total. Moreover, different and versatile plug configurations result in diverse appearances.
  Analogous to THT connectors, areas with unassembled SMD connector are included in this class and standard ports for USB and FireWire excluded.

- $\text{CHIP}_{\text{UNPOP}}$ stands for unpopulated chip. Due to various design and prototyping decisions, additional soldering pads for possible ICs are occasionally placed on the boards. However, a final version of the hardware may not depend on those components and the corresponding pads are left unpopulated. Still, the pads have valid connections to other components and might serve for probing.



(a) SOP (b) QFP (c) SON (d) QFN (e) BGA

**Figure 3.3:** First half of representative examples with 5 of 6 IC classes. (Images removed due to copyright reasons.)


1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)
1×1
(Original size: 200×200 bp)

(a) THT (b) CONN$_{\text{THT}}$ (c) CONN$_{\text{SMD}}$ (d) CHIP$_{\text{UNPOP}}$ (e) MISC

**Figure 3.4:** Second half of visual examples from THT, passives and MISC. (Removed due to copyright reasons.)

Size and appearance of these pad accumulations is governed by the intended chips and their pin configuration. Unpopulated chips are overall scarce compared to their assembled counterparts. Instead of laying out respective classes for each described IC type, all variants of unpopulated chips are gathered by this category.

**Miscellaneous components** The last class devised in this work is named MISC. It stands for miscellaneous components that emerge from the previous labeling works and do not correspond to the described classes. These components have either been labeled mistakenly as ICs or are open for interpretation.

Heatsinks are often marked although they occlude the chips below and therefore cannot be classified as such. Furthermore, small-outline transistors (SOT), diodes (SOD) and other power electronics can have considerable similarity to SOPs but with different pin layout (e.g. odd total amount, uneven or one-sided distribution). Finally, WiFi modules and other complex components with IC resemblance also occur in the pre-labeled data.

Instead of discarding these annotations, they are kept in this supplementary class. If required, the category can be filtered out prior to training and evaluation.

### 3.2.3 Summary and statistics

In order to conclude this chapter, a brief summary for the labeling process results is given.

The existing single-class labeled components from the imported datasets have been assigned to their appropriate classes. Each image is analyzed thoroughly and unlabeled ICs that were missed by the original authors are complemented. Furthermore, the described passive components are localized and annotated. The first pass on each image needs between 10 and 15 minutes. A second pass with temporal distance is highly recommended and takes about 5 to 10 minutes. It often reveals faulty class assignments and still missing components. Especially labeling of unassembled components for the passive classes is time-consuming and error-prone due to their unobtrusive nature.

Additionally, special adjustments for the two datasets are necessary. As described in Section 2.3.3, bounding boxes of FICS-PCB do not cover the extending pins of the ICs. Furthermore, the work on PCB-DSLR revealed a considerable amount of inaccurate bounding boxes that also do not cover the external pins accurately. In order to guarantee consistent training and results, further work is invested to correct these circumstances.

In total, 508 images with 3650 pre-existing annotations have been processed. 4088 additional components have been labeled, the final amount for all classes is 7738. A detailed account per dataset and category is given in Table 3.3. A decline for

THT ICs and an increase of SONs is traceable through the datasets FICS-PCB and SW-Internal, as they contain more modern hardware.

| Class | PCB-DSRL | FICS-PCB | SW-Google | SW-Internal | All datasets |
|---|---|---|---|---|---|
| ICs before | 2048* | 449† | 1100 | 53‡ | 3650 |
| #Images | 165 | 51 | 190 | 102 | 508 |
| MISC | 259 | 180 | 255 | 336 | 1030 |
| SON | 5 | 31 | 15 | 54 | 105 |
| SOP | 1341 | 193 | 543 | 438 | 2515 |
| QFN | 38 | 44 | 102 | 53 | 237 |
| QFP | 346 | 59 | 157 | 42 | 604 |
| BGA | 80 | 19 | 83 | 41 | 223 |
| THT | 260 | 16 | 74 | 0 | 350 |
| $\mathrm{CONN}_{\mathrm{SMD}}$ | 265 | 53 | 75 | 108 | 501 |
| $\mathrm{CONN}_{\mathrm{THT}}$ | 818 | 217 | 514 | 292 | 1841 |
| $\mathrm{CHIP}_{\mathrm{UNPOP}}$ | 165 | 22 | 87 | 58 | 332 |
| All classes | 3577 | 834 | 1905 | 1422 | 7738 |

**Table 3.3:** Absolute component count per class for available datasets. First row shows single-class IC count per dataset prior to multi-class labeling; * refers to unique ICs, †to DSLR subset of FICS-PCB and ‡to original SW-Custom dataset (15 images) prior to the merge with SW-Internal.

In order to give a better understanding for the actual distribution, relative occurrences for each class are reported in Figure 3.5. SOP and $\mathrm{CONN}_{\mathrm{THT}}$ components are the dominant categories throughout all datasets. QFP, QFN and BGA follow with substantial distance while SON being scarcest. The remaining passive components reside in the range between QFP and SON. Overall, the class distribution is skewed and reveals the unbalanced nature of the available data.

Table 3.4 summarizes the density of labeled components per PCB image for the different datasets. While all sets contain images with almost no components, the maximum count varies substantially. PCB-DSLR stands out with its high maximum peak that derives from few large boards with high object count. In contrast, the authors of FICS-PCB recorded each large PCB with several images. The peaks for SW-Google and -Internal are lower but show that a desired object detector should accomplish the detection of up to 100 components. Still, the majority of boards contains less than 20 labeled objects.

Finally, the varying sizes and aspect ratios for components of different classes is visualized in Figure 3.6. In order to allow the comparison of objects originating from different images with varying sizes, the previously mentioned `keep_aspect_ratio_resizer` is applied to all pictures (max. target dimension of 900, also refer to Sec-

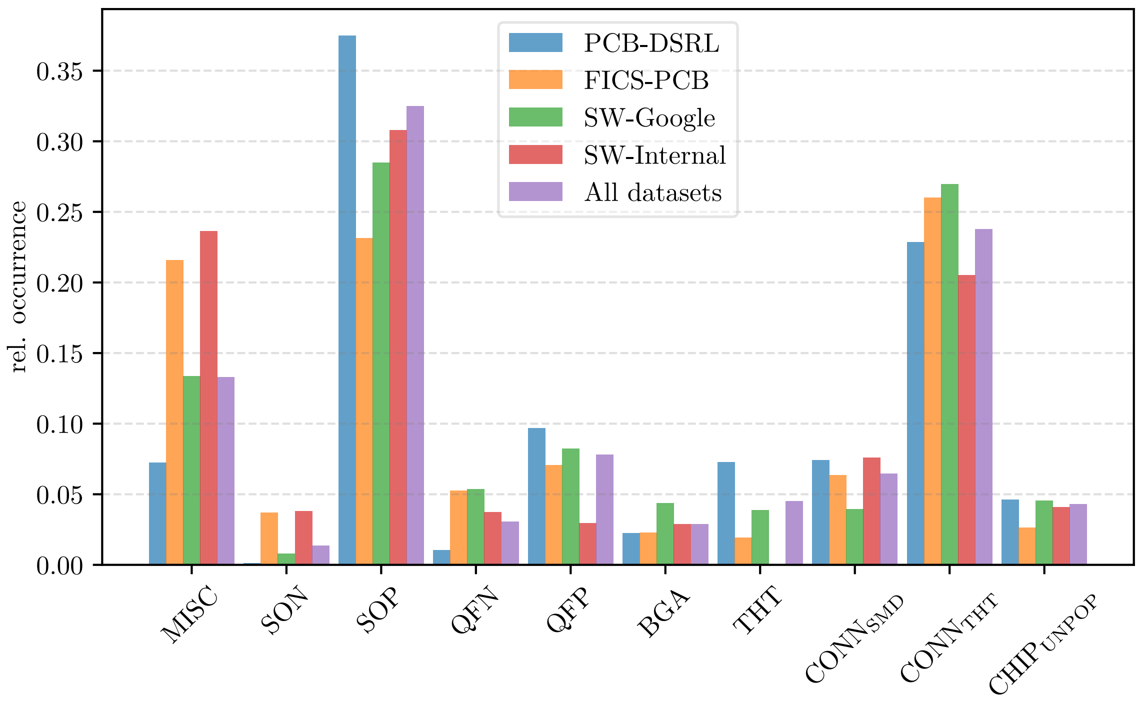


**Figure 3.5:** Relative component occurrences per class for available datasets.

| | PCB-DSRL | FICS-PCB | SW-Google | SW-Internal | All datasets |
|---|---|---|---|---|---|
| Minimum | 1 | 3 | 1 | 0 | 0 |
| Maximum | 171 | 36 | 91 | 88 | 171 |
| Mean | 21.7 | 16.4 | 10.0 | 13.9 | 15.2 |
| Std. deviation | 19.9 | 8.0 | 12.4 | 14.4 | 16.1 |
| Median | 21 | 15 | 6 | 10 | 11 |

**Table 3.4:** Statistics for component density in images of available datasets.

tion 3.1.2). Therefore, all components are resized accordingly and can be compared appropriately.

The areal size concept of COCO metrics is adapted and divides the plot into three distinct dimension categories. Furthermore, integer aspect ratios are depicted and allow a course categorization with respect to object elongation. The small amount of components with extreme dimensions above 350 px have been omitted, otherwise the figure's part for small objects would have been compressed beyond recognition.

SONs are QFNs are almost exclusively concentrated in the lower left part with negligible variance in elongation. The green cloud of SOPs mostly stretches from small to medium sized components with moderate aspect ratios. The same applies for the cyan category of $\text{CHIP}_{\text{UNPOP}}$ with fewer visible examples. QFPs and BGAs mostly reside on the diagonal due to their often quadratic shape and are clearly medium and large sized objects. Both types of connectors are distributed between all sizes with the majority residing in the medium range. Furthermore, their versatile elongations result in more extreme aspect ratios than components of other categories.

As a conclusion, the proposed object detector should focus on smaller dimensions and extended aspect ratios with respect to its bounding box proposals. These insights should approve the described configuration decisions with respect to anchor box proposals in Section 3.1.2.

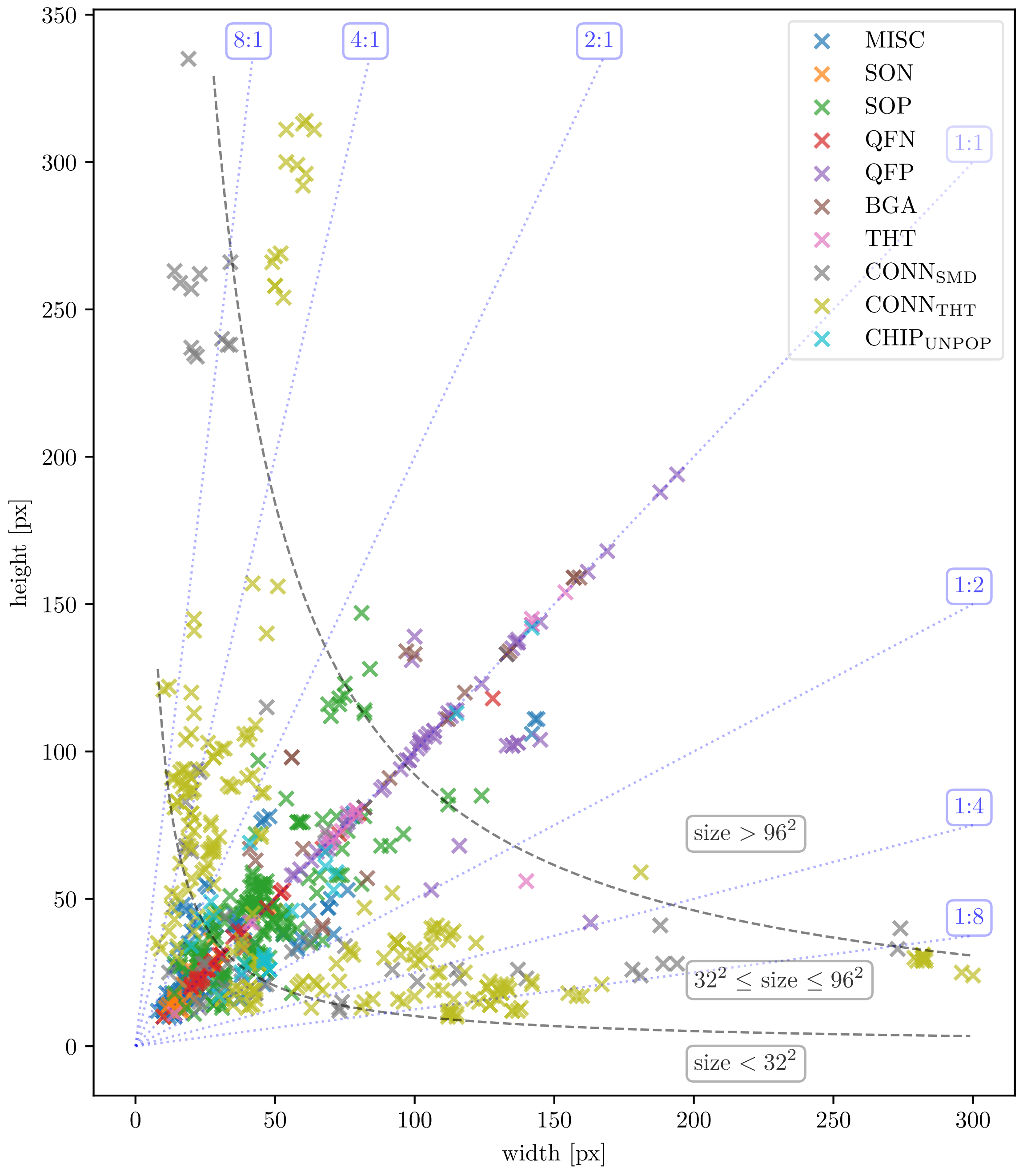


**Figure 3.6:** Visualization of component dimensions after scaling to target size of network input layer. Dashed black lines indicate boundaries between small, median and large area categories. Blue dotted lines represent aspect ratios of possible anchor box proposals. Only the small dataset FICS-PCB is depicted, otherwise the scatter-plot would not be legible.

# 4 Experiments and results

This chapter presents the results from Faster R-CNN models that have been trained with different combinations of datasets. All models follow the configuration described in Section 3.1.2 and 3.1.3.

Due the fact that hyper-parameters and other settings are not tuned during training, this work omits validation sets and consistently uses only training and test sets.

Throughout the first three sections of this chapter the dataset SW-Internal is utilized for testing only. The other datasets are used for training in order to assess their individual and combined usefulness for training the best model for the target set SW-Internal. The experiments in the last section include SW-Internal for training by utilizing $k$-fold cross-validation. The goal is to assess the potential detection performance of a final model trained with all available data.

The test set does not contain any THT ICs and the COCO API consistently reports $-1.0$ AP for this category throughout all experiments. As a result this class is not depicted in the corresponding performance score figures.

Throughout all training runs the models are trained for 50000 steps with continuous evaluation on the test set scheduled every 1000th step. Inspection via TensorBoard shows for all runs, that up to the final training step all AP performance curves for the test data saturate appropriately and degradations do not occur. Therefore, the problem of overfitting does not arise [37] while all phases of learning rate reduction are successfully passed. All results presented in the following parts stem from the final evaluation process after the 50000th training step.

## 4.1 Individual datasets

This part presents the results from training with single datasets PCB-DSLR, FICS-PCB and SW-Google. Figure 4.1 illustrates the achieved AP scores for each class and dataset.

The model trained with SW-Google yields the best overall result of 0.39 mAP. Especially its performance on the classes QFP and BGA stands out with AP above 0.8 and 0.7 respectively Also, the scores for the categories QFN and SOP are leading. Although SW-Google contains twice as much $\text{CONN}_{\text{THT}}$ components as FICS-PCB, the latter seems to provide better samples for the respective connectors that occur in SW-Internal.

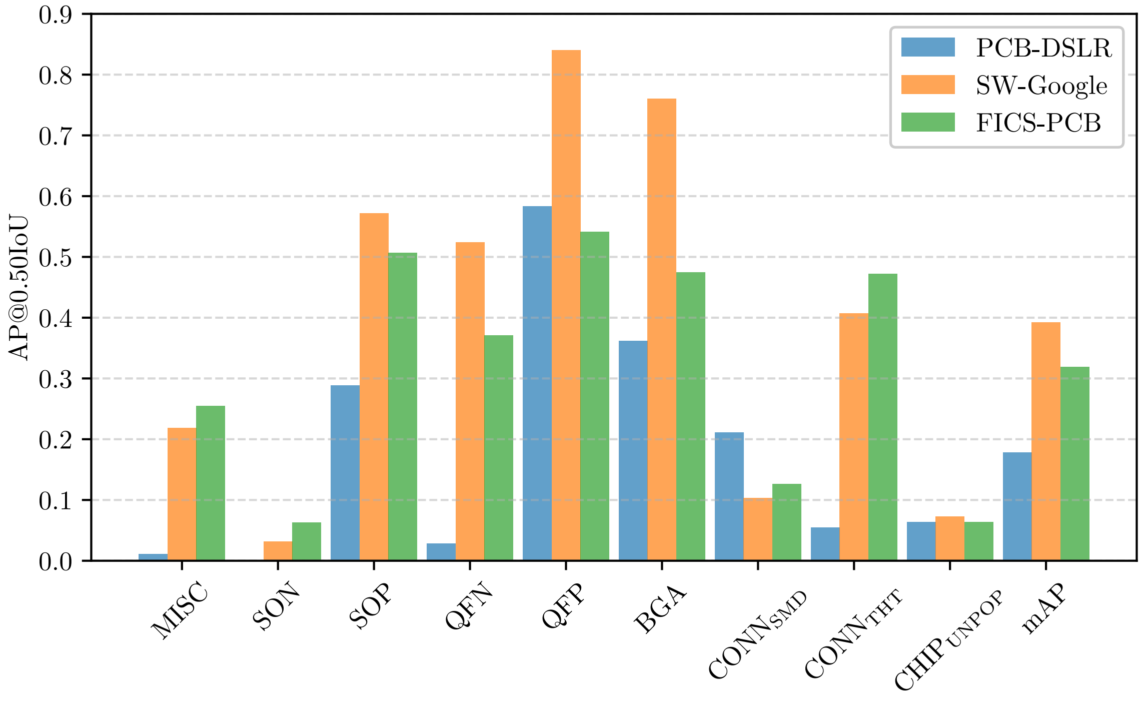


**Figure 4.1:** Per class performances for training with single datasets and evaluation on SW-Internal. Last column reports mean average precision over all categories.

The impact of FICS-PCB on training is also noteworthy with 0.32 mAP. The performance scores are the 2nd best in almost every category and taking the lead for MISC and $\text{CONN}_{\text{THT}}$. Especially SOP and QFN are close to the results yielded by SW-Google. Altogether, this dataset still enables a remarkable performance considering that it contains the smallest amount of images and annotated components of all available datasets.

Finally PCB-DSLR falls short with overall 0.18 mAP. Its performance on QFP is acceptable with 0.58 AP but most other categories are reported below 0.4 AP. Low scores on the IC classes SON and QFN are plausible due to the low amount of samples (5 and 38 respectively) in the dataset. In contrast, the poor performance for the SOP and $\text{CONN}_{\text{THT}}$ categories are likely caused by the origin of its images; a certain proportion of its SOPs are covered with dust and $\text{CONN}_{\text{THT}}$s are often seriously damaged. One exception is $\text{CONN}_{\text{SMD}}$ with leading 0.21 AP. The dataset holds more suitable samples of this class that provides better performance for the test set. Altogether, being the largest dataset with respect to the amount of annotated components does not necessarily enable good test performance on SW-Internal.

The classes SON and $\text{CHIP}_{\text{UNPOP}}$ are below 0.1 AP for all sets. Possible reasons are the low amount of samples and small visual appearances in the images.

## 4.2 Dataset combinations

The next step after single dataset training is to utilize the remaining possible combinations and analyze their impact on training and evaluation. As in the previous part, SW-Internal is hold out and used as test set. The corresponding AP scores are depicted in Figure 4.2. They are more versatile compared to the results from training with single datasets.

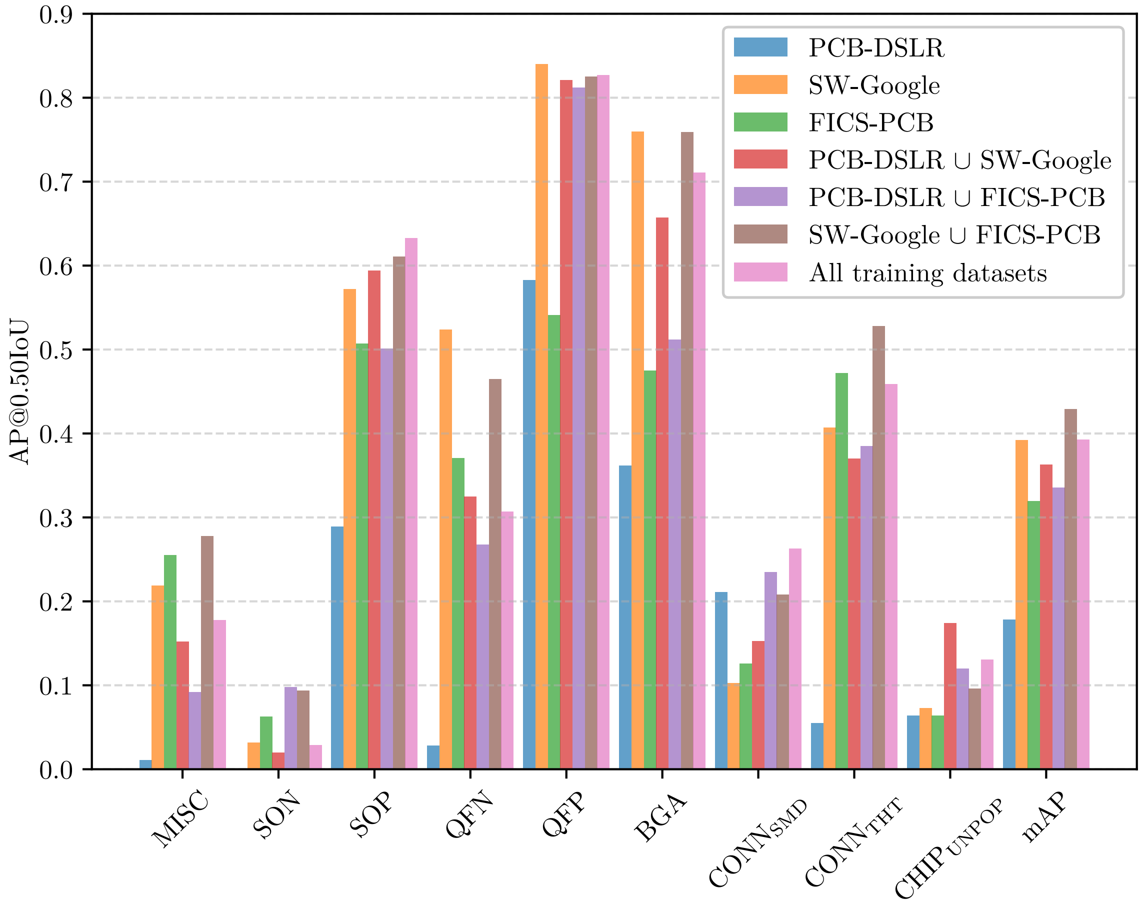


**Figure 4.2:** Per class performances for training with combined datasets and evaluation on SW-Internal. Results from single datasets are depicted for comparison.

The combination of SW-Google and FICS-PCB (brown) leads with an overall performance of 0.43 mAP, followed by the combination of all three and solo FICS-PCB (both 0.39 mAP). In comparison to the individual performances, the leading combination shows improvements for the categories MISC, SON, SOP and especially both connector variants. The classes BGA and QFP do not further benefit from this union and the performance for QFN even decreases compared to the FICS-PCB only result.

Finally, the $\text{CHIP}_{\text{UNPOP}}$ is increased by the combination but still resides below 0.1 AP.

The union of PCB-DSLR and SW-Google (red) falls below the individual performances in almost all categories and can only prevail for the classes SOP, QFP and $\text{CHIP}_{\text{UNPOP}}$; a relatively strong gain to 0.18 AP for the last category can be noted.

PCB-DSLR combined with FICS-PCB (purple) results in a slight improvement with respect to overall mAP performance. The highest SON score and a huge improvement in the QFP category to over 0.8 AP are noteworthy. Slight increases for BGA, $\text{CONN}_{\text{SMD}}$ and $\text{CHIP}_{\text{UNPOP}}$ are also given. On the other hand, MISC, QFN and $\text{CONN}_{\text{THT}}$ do not benefit from this union and decrease compared to the individual sets.

The combination of all three datasets does not provide the best performance. While the categories SOP and $\text{CONN}_{\text{SMD}}$ report the best scores, SON does not benefit at all compared to other combinations. Furthermore, the class performance for QFN drops significantly compared to SW-Google and FICS-PCB. Scores for QFP, BGA and $\text{CONN}_{\text{THT}}$ and $\text{CHIP}_{\text{UNPOP}}$ can be seen as reasonably equal to other combinations. Considering only the IC classes, this combination offers strong performance for medium and large sized components with the downside of a less than moderate ability to detect small ICs.

In summary, the union of SW-Google and FICS-PCB enables the best performance on the test set. However, the gap is only 0.04 mAP compared to the model trained solely with SW-Google. The general observation is that combining multiple datasets does not necessarily result in a strong additive performance gain.

## 4.3 Exclusion of MISC class

The introduction of the category for miscellaneous PCB components described in Section 3.2.2.2 can be seen as controversial. The class functions as a collecting bin for various components that most likely do not offer any benefit in the context of hardware related security analysis at SCHUTZWERK. Therefore, it should be questioned if this category provides any benefit or poses harm. This part addresses the question at hand by providing results for all dataset combinations by excluding all annotations associated with the class MISC from training and test datasets. As in all parts before, SW-Internal is solely used as test set. The respective results are presented in Figure 4.3 and are briefly summarized without detailed comparison between all combinations.

Most noticeable, the combination of all datasets now leads with 0.44 mAP over all categories. Its results for the classes QFN, QFP, BGA and $\text{CONN}_{\text{THT}}$ are increased noticeably. Especially QFN gains 0.11 AP, although results for training with single datasets are now reported with significantly lower scores for this category. $\text{CONN}_{\text{SMD}}$ drops marginally.

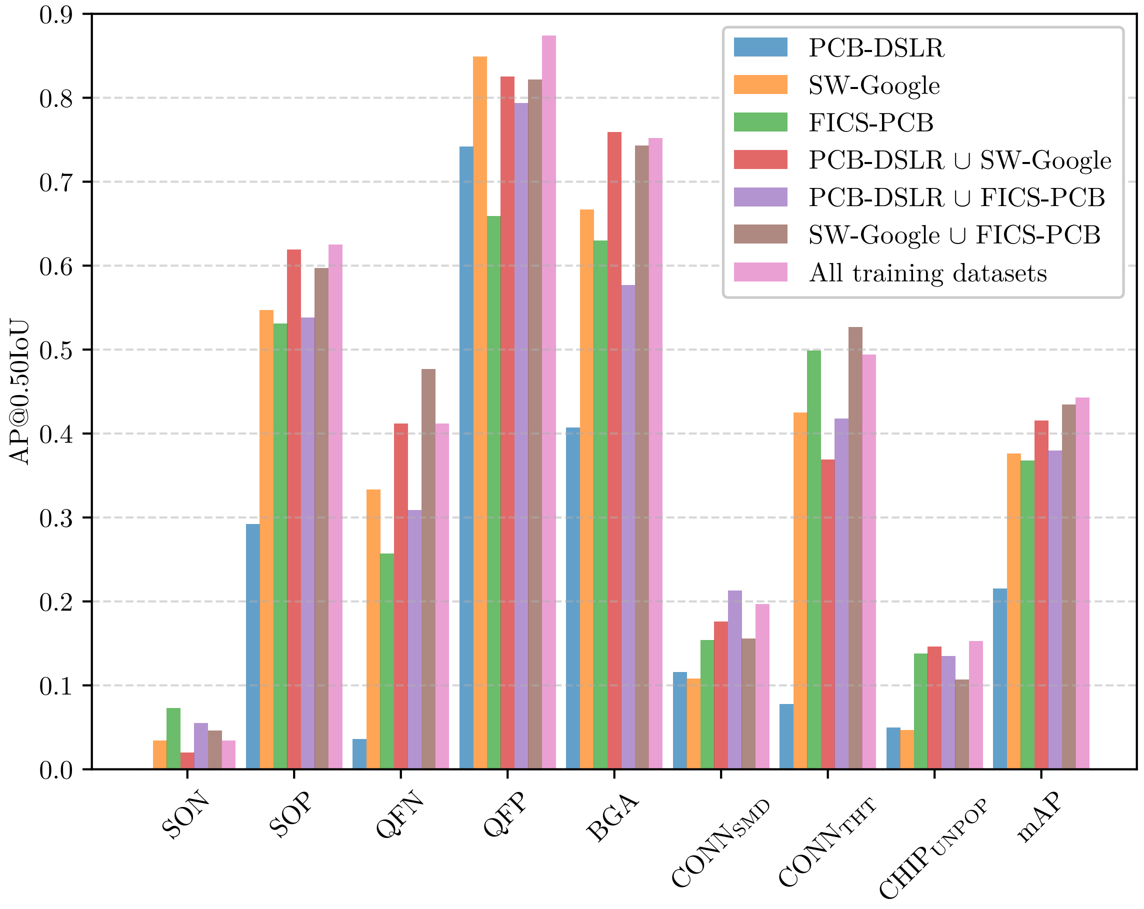


**Figure 4.3:** Detection performances for all datasets with exclusion of the class MISC during training and testing.

For the other datasets and their combinations the results look slightly more different. The former performance in the SON category is not achieved anymore. In general, the results of SW-Google drop for almost every class while PCB-DSLR now enables better performance for QFP and BGA. On the other hand, the former peak performance on $\text{CONN}_{\text{SMD}}$ is no longer present.

The overall impression is, that the removal of MISC samples somehow sets back the detection performance on categories with smaller components (SON and $\text{CONN}_{\text{SMD}}$ in general, QFN for SW-Google) and slightly improves results for medium and large size components throughout all classes. In fact MISC contains a noticeable amount of smaller components (e.g. SOTs and SODs). One possible explanation could be that they represent valuable training samples for the RPN. Without them, the amount of available small sized is reduced significantly and missing during first stage optimization. On the other hand, MISC also contains medium and larger components (e.g. heatsinks) that especially resemble QFPs and BGAs. By omitting this category entirely, it is plausible that confusions between these classes are prevented or at least reduced.

As a result, it can be noted that the combination of all datasets provides the best overall result so far and slightly favors medium and large sized IC components. However, the drawback is that one category is completely dismissed while the gain is only 0.01 mAP compared to the results with all classes. Therefore, MISC will be included for the remaining experiments.

## 4.4 Training including SW-Internal

So far, SW-Internal has only been used as a test set and the combination of SW-Google and FICS-PCB for training has provided the best results. In order to include SW-Internal into the training, a 5-fold cross-validation is performed. The dataset is shuffled and split into five parts with roughly the same amount of labeled components per part (300, 245, 259, 362 and 256). Five train/test combinations are constructed, each with 4/5 of the image data used for training and 1/5 for testing. In general, this procedure enables the evaluation of machine learning models on limited data. Consequently, five Faster R-CNN object detectors are trained and tested with the different combinations. Afterwards, their performance scores are usually averaged. However, the splits do not contain an equal amount of components from each class. Therefore, a weighted mean is applied on the per-category scores over the iterations of the cross-validation process with respect to the varying object count per class for each split.

In the following, three different experiments are presented that utilize the 5-fold cross-validation approach:

1. Five combinations of the datasets SW-Google (5×100%), FICS-PCB (5×100%) and SW-Internal (5×80%) are used for training COCO pre-trained models.
2. The check-point from previous training with SW-Google $\cup$ FICS-PCB (Section 4.2) is utilized and five models are fine-tuned with SW-Internal (5×80%). This approach can be denoted as consecutive fine-tuning of a pre-trained model with multiple training runs.
3. Only SW-Internal (5×80%) is utilized as a training set in order to asses if the previous datasets are necessary at all for achieving good detection performance.

Figure 4.4 shows the results from the 5-fold cross-validation for the described experiments. The averaged scores are complemented by the min-max range of individual AP results from the iterations. This helps to understand from witch range of values each mean is composed of.

The overall performance of the first experiment only yields 0.34 mAP and falls below the model trained with SW-Google $\cup$ FICS-PCB (0.43 mAP, Section 4.2). In

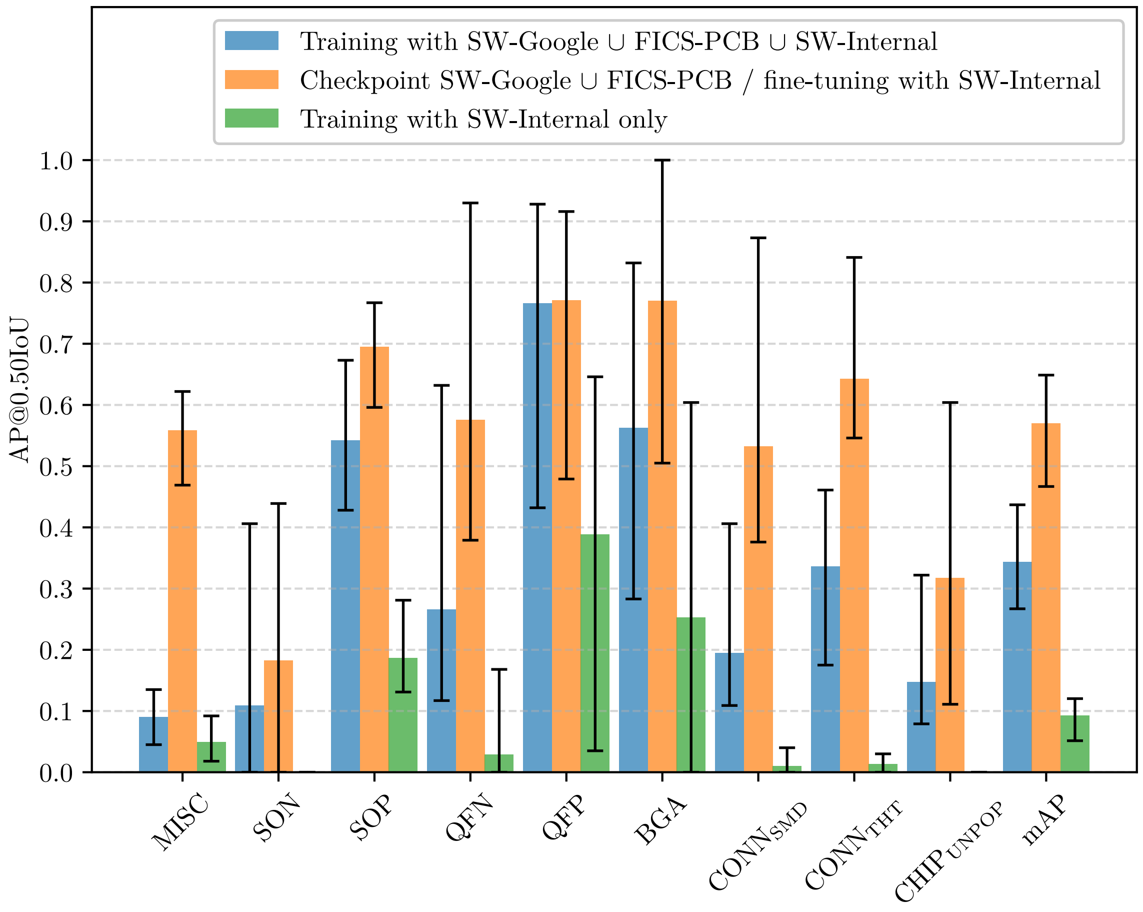


**Figure 4.4:** Test results of the experiments with 5-fold cross-validation. Error bars show maximum and minimum performance scores achieved during the individual iterations of the cross-validation process.

detail, almost every class (except SON, CONN$_{\mathrm{SMD}}$ and CHIP$_{\mathrm{UNPOP}}$) suffers a loss of about 0.1 AP or more compared to the best model trained so far. It is unclear why data, that has been previously tested with acceptable performance, causes inferior results if included during training. A verification of the performance curves via TensorBoard does not reveal abnormalities. But the possibility exists, that the number steps has not been sufficient for training with of all sets combined. A combination of more training steps and a possible further decay of the learning rate could be explored, but is not part of this work anymore.

The second experiment shows an overall performance of 0.57 mAP and succeeds the previous best model (Section 4.2). Almost every category yields better scores except for QFP with only 0.77 AP (SW-Google ∪ FICS-PCB: 0.82 AP). Especially classes with previously weak performances (SON, QFN, CONN$_{\mathrm{SMD}}$ and CHIP$_{\mathrm{UNPOP}}$) benefit substantially and gain up to 0.2 AP. However, SON and CHIP$_{\mathrm{UNPOP}}$ components are still poorly detected (0.18 AP and 0.32 AP respectively).

Finally, the last experiment shows a tremendous cutback with 0.09 mAP. The inferior performance to all other models points out that the internal dataset alone might not enable sufficient training. Compared to results from training with SW-Google only (0.39 mAP) one reason might be that the dataset SW-Internal contains only a fraction of images and components compared to the former (0.53- and 0.75-fold respectively). Additionally, the reduction of available training data caused by the $^4/_5$ split for cross-validation results in even less available data for training (0.43 of images and 0.6 of components compared to all data from SW-Google). This could represent a critical lower bound for training data that should not be undercut. On the other hand, FICS-PCB itself contains even less images and components than the 0.8-fold version of SW-Internal but yields better detection performance if used for training (Section 4.1). Therefore, a comprehensive justification is currently not available.

# 5 Conclusion

## 5.1 Summary and discussion

During the course of this work, four different datasets have been collated that origin from the context of PCB component analysis. Each set has been adjusted and elaborately refined. The datasets' previous single-class annotations with respect to ICs have been re-labeled with six new IC subcategories. In addition, four new classes have been introduced that cover two connector types, areas of unpopulated chips and various other components.

In order to accomplish this task, a suitable open-source label software has been utilized. Adequate import and export interfaces have been developed in order handle the various dataset source formats and allow the export into a uniform target format.

A total of 508 images have been processed and the amount of labeled components has been more than doubled in comparison to the progenitor work. In addition, detailed statistics for the refined datasets have been compiled that describe their properties.

Based on the previous work and insights gained by reviewing related academic publications on the topic of PCB component analysis, the CNN based Faster R-CNN meta-architecture has been chosen for PCB component detection. The TensorFlow Object Detection API and its reference implementation of the detector have been deployed on a dedicated hardware platform. Various options for the object detector's configuration have been investigated and adjusted based on the characteristics of the available datasets.

During the next phase, various experiments have been conducted in order asses the performance of the proposed Faster R-CNN model. In the first two parts of the experiments, different combinations of the datasets have been utilized in order to determine the combinations that produce best detection performances on the target dataset SW-Internal. The evaluation shows that training with the combination SW-Google and FICS-PCB yield the best results, closely followed by a union of all three datasets and SW-Google only. An important intermediate conclusion is that training with multiple datasets does not give rise to substantial higher detection performances. Furthermore, each combination results in a trained model with slightly different strengths and weaknesses for the individual categories.

The last three experiments deal with the transition of including SW-Internal during model training. The first discovery is that a model trained with the union of the previously best dataset combination and SW-Internal does not result in better detection performances. Moreover, another model trained solely with SW-Internal yields inferior performance compared to all other detectors. The previously stated reasons for both results are speculative and the real circumstances for these behaviors have not been discovered yet.

Finally, the second of the last experiments presents a trained Faster R-CNN model with superior performance in almost all aspects. By utilizing a previous training effort, the best model trained with SW-Google and FICS-PCB has been further refined SW-Internal. It depicts a valid approach for consecutive model fine-tuning with newly available data in the future. If this holds true, it would potentially enable continuous learning without the need for training with all data from scratch.

Independent from the previous statement further conclusions can be drawn. The class imbalance with respect to component counts poses a less disadvantageous impact as expected. The two best scoring categories (QFP and BGA both $> 0.75$ AP) are represented with less than 10% and 5% of all object samples. This success seems to derive primarily from their uniform visual appearances and medium to large sizes. In contrast, the three most populated categories (MISC, SOP and $\text{CONN}_{\text{THT}}$) settle below 0.7 AP. These classes contain components with a large variety of visual appearances, elongations and sizes. Especially the components' sizes seem to have far greater influence on the detection performance. Classes that consist primarily of small objects (SON and $\text{CHIP}_{\text{UNPOP}}$) do not reach reasonable scores above 0.4 AP. QFNs seem to represent the right compromise: With a relative component count of less than 5%, overall small size, quadratic appearance and four-sided solder contacts, components of this category still yield a detection performance above 0.5 AP.

Overall, the available data is still limited. The best experiment utilizes 323 images with 3877 objects for training, evaluation occurs on 20 images with 284 component samples. Therefore, the results have to be interpreted with caution. But the general trend towards increasing detection performance is apparent, especially if image data is utilized that represents modern embedded hardware.

## 5.2 Future work

A detailed analysis of false positive detections would yield valuable information about insufficient localization, wrong classification and faulty detections in areas that do not hold any sought components. For example, a preliminary visual analysis has shown, that densely packed and aligned arrays of small-outline transistors (SOT) are falsely classified as one THT connector instead of being individually recognized as miscellaneous components.

Cross-validation experiments should be complemented with all possible dataset training combinations in order to confirm that the currently best model indeed yields the best detection performance.

The extension of available image data by public sourcing should focus on appropriate samples from modern embedded systems instead of deprecated consumer hardware. The dataset PCB WACV-2019 seems to meet these requirements and should be utilized in the future. The same applies to PCB-METAL if it becomes publicly available. Furthermore, the internal acquisition process should be continued as it yields the most valuable data. In addition, synthetic generated image data could further complement the available data without the need for manual labeling work due to already available design specifications [34].

With respect to technical aspects, the update of the ODA framework and transition to TensorFlow 2 should be carried out. This would enable to benefit from its ongoing development process, e.g. multi-GPU training. Furthermore, the concept of profiling has been introduced to TensorFlow 2. This would allow to analysis which parts of a model are most GPU time and memory consuming instead of performing trial and error runs in order to find suitable configurations (e.g. input dimensions, anchor box sizes and aspect ratios) that do not result in out-of-memory scenarios.

Further developments in the field of CNN based feature extractors, like ResNeXt [47], should also be closely observed. The Faster R-CNN meta-architecture would surely benefit from lower memory consumption, better performance as well as faster training and inference times if such pre-trained models become available in ODA.

In order to combat low class sample counts and general class imbalances, new approaches in the field low-/few-shot learning could be investigated [42]. Related work by Kuo et al. utilizes low-shot similarity-based classification in conjunction with a *graph network* for feature embedding [12]. The model achieves good performance with training on the PCB WACV-2019 dataset that consists of only 47 images. With respect to small and densely packet objects, the work of Reza et al. [34] introduces *loss boosting* in order to improve bounding box regression at last stage of object detectors in general. These approaches are currently of an academic nature and have not yet found their way into established object detection frameworks.

Meanwhile, a near-term solution for small-scale object detection could be achieved by adapting a sliding window approach on the available image data. In addition to performing detection on whole but down-sized images, the analysis of overlapping patches (similar to the microscope subset of FICS-PCB) could enable reliable detection of small-scale PCB components. Such an approach would very well coincide with the planned image acquisition system of the future hardware platform for automated contacting. Furthermore, the full potential of the available high-resolution image data could be utilized immediately.

Altogether, this work provides valuable insights and can be seen as groundwork for the upcoming tasks of chip identification, pin detection as well as signal and protocol analysis between the detected PCB components. These tasks will be accompanied by the currently undergoing upgrade of the flying probe tester platform. It is hoped that the transition to automated security assessments at SCHUTZWERK can be accomplished in the near future.

## 5.3 Acknowledgment

This work has been supported by the Federal Ministry of Education and Research of Germany (BMBF) in the context of the SecForCARs project [2]. Furthermore, the author thanks all employees at SCHUTZWERK for their support, guidance and patience during this work.

# Bibliography

[1] Y. Boykov and V. Kolmogorov. “An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 26.9 (2004), pp. 1124–1137.

[2] Bundesministerium für Bildung und Forschung. *SecForCARs - Sicherheit für vernetzte, autonome Fahrzeuge.* [Online; accessed 03-Nov-2020]. Apr. 2018. URL: `https://www.forschung-it-sicherheit-kommunikationssysteme.de/projekte/sicherheit-fuer-vernetzte-autonome-fahrzeuge`.

[3] D. Comaniciu and P. Meer. “Mean Shift: A Robust Approach Toward Feature Space Analysis”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 24.5 (2002), pp. 603–619.

[4] P. Dollár and C. L. Zitnick. “Fast Edge Detection Using Structured Forests”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 37.8 (2015), pp. 1558–1570.

[5] C. Escarmonacalpe. *Add/Delete images/video to existing task - Issue #95.* [Online; accessed 30-Oct-2020]. Sept. 26, 2018. URL: `https://github.com/openvinotoolkit/cvat/issues/95`.

[6] P. F. Felzenszwalb and D. P. Huttenlocher. “Efficient Graph-Based Image Segmentation”. In: *Int. J. Comput. Vis.* 59.2 (2004), pp. 167–181.

[7] R. B. Girshick. “Fast R-CNN”. In: *ICCV.* IEEE Computer Society, 2015, pp. 1440–1448.

[8] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *CVPR.* IEEE Computer Society, 2014, pp. 580–587.

[9] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016.* IEEE Computer Society, 2016, pp. 770–778.

[10] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. “Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017.* IEEE Computer Society, 2017, pp. 3296–3297.

[11] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. *Tensorflow Object Detection API.* [Online; accessed 23-Sep-2020]. Jan. 19, 2019. URL: `https://github.com/tensorflow/models/tree/r1.13.0/research/object_detection`.

[12] C.-W. Kuo, J. Ashmore, D. Huggins, and Z. Kira. "Data-Efficient Graph Embedding Learning for PCB Component Detection". In: *IEEE Winter Conference on Applications of Computer Vision, WACV 2019, Waikoloa Village, HI, USA, January 7-11, 2019.* IEEE, 2019, pp. 551–560.

[13] C.-W. Kuo, J. Ashmore, D. Huggins, and Z. Kira. *Data-Efficient Graph Embedding Learning for PCB Component Detection, supplementary materials.* [Online; accessed 03-Sep-2020]. 2019.

[14] C.-W. Kuo, J. Ashmore, D. Huggins, and Z. Kira. *WACV-2019 Graph PCB Detection.* [Online; accessed 03-Sep-2020]. 2019. URL: `https://sites.google.com/view/graph-pcb-detection-wacv19/home`.

[15] J. Li, J. Gu, Z. Huang, and J. Wen. "Application Research of Improved YOLO V3 Algorithm in PCB Electronic Component Detection". In: *Applied Sciences* 9.18 (2019), p. 3750.

[16] W. Li, C. Jiang, M. Breier, and D. Merhof. "Localizing components on printed circuit boards using 2D information". In: *IEEE International Conference on Industrial Technology, ICIT 2016, Taipei, Taiwan, March 14-17, 2016.* IEEE, 2016, pp. 769–774.

[17] D. Lim, Y. Kim, and T. Park. "SMD Classification for Automated Optical Inspection Machine Using Convolution Neural Network". In: *3rd IEEE International Conference on Robotic Computing, IRC 2019, Naples, Italy, February 25-27, 2019.* IEEE, 2019, pp. 395–398.

[18] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. "Feature Pyramid Networks for Object Detection". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017.* IEEE Computer Society, 2017, pp. 936–944.

[19] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. "Focal Loss for Dense Object Detection". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017.* IEEE Computer Society, 2017, pp. 2999–3007.

[20] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. *Microsoft COCO: Common Objects in Context - Detection Evaluation.* [Online; accessed 24-Sep-2020]. 2014.

[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. "SSD: Single Shot MultiBox Detector". In: *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I.* Ed. by B. Leibe, J. Matas, N. Sebe, and M. Welling. Vol. 9905. Lecture Notes in Computer Science. Springer, 2016, pp. 21–37.

[22] J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2015, pp. 3431–3440.

[23] H. Lu, D. Mehta, O. P. Paradis, N. Asadizanjani, M. M. Tehranipoor, and D. L. Woodard. *FICS-PCB: A Multi-Modal Image Dataset for Automated Printed Circuit Board Visual Inspection.* [Online; accessed 03-Sep-2020]. 2020. URL: `https://www.trust-hub.org/data`.

[24] H. Lu, D. Mehta, O. P. Paradis, N. Asadizanjani, M. M. Tehranipoor, and D. L. Woodard. "FICS-PCB: A Multi-Modal Image Dataset for Automated Printed Circuit Board Visual Inspection". In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 366.

[25] G. Mahalingam, K. M. Gay, and K. Ricanek. "PCB-METAL: A PCB Image Dataset for Advanced Computer Vision Machine Learning Component Analysis". In: *16th International Conference on Machine Vision Applications, MVA 2019, Tokyo, Japan, May 27-31, 2019.* IEEE, 2019, pp. 1–5.

[26] H. Mobahi, S. R. Rao, A. Y. Yang, S. S. Sastry, and Y. Ma. "Segmentation of Natural Images by Texture and Boundary Compression". In: *Int. J. Comput. Vis.* 95.1 (2011), pp. 86–98.

[27] Nuntipat Narkthong. *Problem detecting large number of objects in single image with Tensorflow Object Detection API.* [Online; accessed 24-Sep-2020]. Dec. 31, 2019. URL: `https://stackoverflow.com/questions/59547775/problem-detecting-large-number-of-objects-in-single-image-with-tensorflow-object/63084364%5C#63084364`.

[28] R. Padilla, S. L. Netto, and E. A. B. da Silva. "A Survey on Performance Metrics for Object-Detection Algorithms". In: *2020 International Conference on Systems, Signals and Image Processing, IWSSIP 2020, Niterói, Brazil, July 1-3, 2020.* IEEE, 2020, pp. 237–242.

[29] N. Plesa. *Annotation tools for building datasets.* [Online; accessed 27-Sep-2020]. July 2020. URL: `https://www.datasetlist.com/tools/`.

[30] C. Pramerdorfer and M. Kampel. "A dataset for computer-vision-based PCB analysis". In: *14th IAPR International Conference on Machine Vision Applications, MVA 2015, Miraikan, Tokyo, Japan, 18-22 May, 2015.* IEEE, 2015, pp. 378–381.

[31] V. Rathod and J. Huang. *TensorFlow 2 meets the Object Detection API.* [Online; accessed 03-Sep-2020]. July 10, 2020. URL: `https://blog.tensorflow.org/2020/07/tensorflow-2-meets-object-detection-api.html`.

[32] J. Redmon and A. Farhadi. "YOLOv3: An Incremental Improvement". In: *CoRR* abs/1804.02767 (2018).

[33] S. Ren, K. He, R. B. Girshick, and J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada.* Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. 2015, pp. 91–99.

[34] M. A. Reza, Z. Chen, and D. J. Crandall. "Deep Neural Network–Based Detection and Verification of Microelectronic Images". In: *Journal of Hardware and Systems Security* 4 (2020), pp. 44–54.

[35] A. Rosebrock. *Intersection over Union (IoU) for object detection.* [CC BY-SA 4.0, Online; accessed 25-Sep-2020]. Nov. 7, 2016. URL: `http://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection`.

[36] Schutzwerk GmbH. *Embedded Systems Assessment.* [Online; accessed 04-Nov-2020]. 2020. URL: `https://www.schutzwerk.com/en/107/Embedded-Systems-Assessment.html`.

[37] C. Shorten and T. M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". In: *J. Big Data* 6 (2019), p. 60.

[38] S. Siriwardana. *How transfer learning happening in tensorfow object detection API?* [Online; accessed 30-Oct-2020]. Aug. 13, 2017. URL: `https://github.com/tensorflow/models/issues/2203`.

[39] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.* Ed. by S. P. Singh and S. Markovitch. AAAI Press, 2017, pp. 4278–4284.

[40] Tensorflow community contributors. *TFRecord and tf.train.Example.* [Online; accessed 24-Sep-2020]. Sept. 19, 2020. URL: `https://www.tensorflow.org/tutorials/load_data/tfrecord`.

[41] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. "Selective Search for Object Recognition". In: *Int. J. Comput. Vis.* 104.2 (2013), pp. 154–171.

[42] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. “Generalizing from a Few Examples: A Survey on Few-shot Learning”. In: *ACM Comput. Surv.* 53.3 (2020), 63:1–63:34.

[43] F. A. Weber. “Automatisierung der hardwarenahen Sicherheitsanalyse von eingebetteten Systemen”. MA thesis. Hochschule Kempten, Sept. 16, 2019.

[44] Wikipedia contributors. *Black-box testing — Wikipedia, The Free Encyclopedia.* [Online; accessed 04-Nov-2020]. July 1, 2020. URL: `https://en.wikipedia.org/w/index.php?title=Reverse_engineering&oldid=986651339`.

[45] Wikipedia contributors. *Reverse engineering — Wikipedia, The Free Encyclopedia.* [Online; accessed 04-Nov-2020]. Nov. 2, 2020. URL: `https://en.wikipedia.org/w/index.php?title=Reverse_engineering&oldid=986651339`.

[46] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. *Detectron2.* [Online; accessed 04-Sep-2020]. 2019. URL: `https://github.com/facebookresearch/detectron2`.

[47] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. “Aggregated Residual Transformations for Deep Neural Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017.* IEEE Computer Society, 2017, pp. 5987–5995.

[48] A. Yelisetty. *Understanding Fast R-CNN and Faster R-CNN for Object Detection.* [Online; accessed 31-Oct-2020]. July 13, 2020. URL: `https://towardsdatascience.com/understanding-fast-r-cnn-and-faster-r-cnn-for-object-detection-adbb55653d97`.

[49] G. Y. Yeong. *COCO api evaluation for subset of classes.* [Online; accessed 23-Sep-2020]. June 4, 2019. URL: `https://stackoverflow.com/questions/56247323/coco-api-evaluation-for-subset-of-classes/56442781#56442781`.

[50] S. Youn, Y. Lee, and T. Park. “Automatic classification of SMD packages using neural network”. In: *2014 IEEE/SICE International Symposium on System Integration.* 2014, pp. 790–795.