Deep Neural Network–Based Detection and Verification of Microelectronic Images

Md Alimoor Reza, Zhenhua Chen & David J. Crandall

Journal of Hardware and Systems Security

ISSN 2509-3428

J Hardw Syst Secur
DOI 10.1007/s41635-019-00088-4

Journal of Hardware and Systems Security
https://doi.org/10.1007/s41635-019-00088-4

Deep Neural Network–Based Detection and Verification of Microelectronic Images

Md Alimoor Reza[1] · Zhenhua Chen[1] · David J. Crandall[1]

Received: 11 June 2019 / Accepted: 18 November 2019

Abstract
The safety and integrity of complex electronic devices depend on their electronic components, many of which traverse a complex global supply chain before reaching the device manufacturer. Ensuring that these components are correct and legitimate is a significant challenge, especially given the billions of electronic devices that we depend upon. One possible approach is to use computer vision algorithms to analyze images of electronic components—either installed on printed circuit boards or in isolation—to try to automatically spot incorrect or suspicious parts or other potential problems. Such an automatic approach could be especially helpful for large-scale collections of devices, for which manual inspection would be prohibitively expensive. In this paper, we consider two specific problems in this challenging area of microelectronic device inspection: (i) electronic component detection and (ii) electronic component verification. First, we introduce a technique for locating integrated circuits (ICs) on printed circuit boards (PCBs). We apply modern computer vision algorithms, specifically deep learning with convolutional neural networks, to this problem, but find that the small and cluttered nature of electronic components is a significant challenge. We introduce techniques to help overcome this challenge. Second, we consider the problem of component verification: given a pair of IC images, we try to determine if they are the same part or not, ignoring variations caused both by imaging conditions and by expected manufacturing variations across legitimate instances of the same part. We learn a deep feature representation automatically for this problem by showing the algorithm pairs of known similar parts and different parts during training. We evaluate these techniques on large-scale datasets of PCB and IC images we collected from the web.

Keywords Electronic component detection · Electronic component verification · Automated visual inspection · IC detection and localization · IC image matching

1 Introduction

Electronic devices form the foundation of modern society, and thus, ensuring that they operate safely, securely, and reliably is of utmost importance. While most work in computer security focuses on software vulnerabilities, recent reports have highlighted potential hardware vulnerabilities of the microelectronic components of these devices:[1] a single incorrect part could make a large electronic system insecure, unreliable, or unstable. Such unauthorized parts could be introduced at numerous points in the complex global electronic supply chain, and could be a result of a range of motivations from manufacturer error, to a supplier substituting an inappropriate part in order to save money, to a nefarious actor trying to purposely compromise a critical system.

A possible solution to this problem is to inspect microelectronic components automatically before they are installed in critical systems. While various approaches have been proposed for doing this, from automated electronic testing to smart fingerprinting techniques that can help ensure devices have not been compromised, many of these solutions are costly, require the cooperation of component manufacturers, cannot be deployed at scale, or have other limitations [1].

✉ Md Alimoor Reza
mdreza@iu.edu

Zhenhua Chen
chen478@iu.edu

David J. Crandall
djcran@iu.edu

[1] Department of Computer Science, Indiana University, Bloomington, IN, USA

[1] The big hack: How China used a tiny chip to infiltrate US companies, Bloomberg, Oct 4, 2018

In this paper, we consider using automated visual inspection for identifying incorrect or suspicious integrated circuits (ICs) on printed circuit boards (PCBs). As a first step, we address two specific (but nevertheless important) problems: (1) detecting and localizing ICs in PCBs and (2) performing fine-grained matching between two IC images (e.g., a known reference image and an unknown sample) to determine if they are the same part or not. In doing this, we leverage recent exciting progress in object detection and fine-grained object recognition that has been achieved over the last few years, largely through the use of deep neural networks. While these techniques have been tested extensively on datasets of consumer-style photographs from the web (such as ImageNet [2] and COCO [3]), performance on other types of images is less well understood. For example, we assumed that IC detection from PCBs would be an easy problem—detecting black rectangles seems much easier than, say, cars or people in cluttered consumer images! Much to our surprise, an off-the-shelf Faster R-CNN [4] trained on this problem delivered Mean Average Precision (MAP) very close to 0!

Thus while detecting and recognizing small, densely distributed objects may not seem to be a very difficult problem compared with other general detection tasks, our experiments show that the domain of microelectronics has unique challenges for computer vision. The small objects themselves do not include many distinctive visual features, and their extremely dense distribution makes the detection of them even harder. Even worse is the fact that the objects are extremely cluttered with many other electrical components that share many of the same features as ICs. We propose several solutions to help address these problems. The primary contribution of our detection algorithm is a module called *Loss Boosting (LB)* that is specifically designed for solving the imbalance issue between easy samples and hard examples within a single image.

For the verification problem—determining if two images are of the same IC part or not—the main challenge is that differences between two distinct parts may be quite subtle, especially in the case of a counterfeit part: the difference may be just a slightly imperfect manufacturer logo, or otherwise somewhat different package markings. Meanwhile, two images of different instances of the same legitimate IC may appear very different due to variations in illumination, scale, rotation, etc. A critical task of IC matching is thus to develop comparison metrics that ignore the image features that do not matter, while cueing on those that do. Towards this end, we develop a deep neural network–based solution for the verification of these IC components, which automatically learns the important visual features based on observing pairs of known similar and dissimilar images. We apply and evaluate our deep network on a large collection of IC images.

2 Related Work

Here we review recent work related to our goal of applying computer vision to analyze images of printed circuit boards and integrated circuit packages. For a comprehensive introduction to counterfeit electronic components and various techniques for detecting and preventing them, please refer to [1]. As a very brief summary, counterfeit or incorrect parts can enter the supply chain in many places and for many reasons. Many of these cases may be due to simple errors, such as a few unrelated parts accidentally included in a large lot of otherwise correct parts. Deliberate causes of incorrect components typically involve saving or making money: parts may be presented as new but actually recycled from discarded devices, or re-labeled as if they were made to a higher specification (e.g., military grade) than they actually were. It is also possible that counterfeit parts could be introduced in order to purposely damage or compromise the devices they will be used in.

While a variety of countermeasures have been explored, none is perfect [1]. For example, electrical tests can identify obviously incorrect parts but not those that work correctly but may fail early or have other subtle problems. Manual visual inspection is often used to spot counterfeit or incorrect parts, by checking that the device packages and markings are as expected. Our goal here is to help automate this visual inspection process.

Our task of detecting and recognizing ICs in printed circuit boards is simply a special case of the general object detection problem. Most work in object detection—identifying the objects in the image and locating where they are—falls into one of two categories: classification-based detectors such as R-CNNs [5] and its many variants (Fast R-CNNs [6], Faster R-CNNs [4], Small R-CNN [7], Mask CNN [8], etc.), and regression-based detectors including YOLO [9], SSD [10], and focal loss [11].

Classification-based detectors are accurate but relatively slow; for example, Mask-RCNN [8] is currently the most accurate detection method on PASCAL VOC, COCO, and ILSVRC datasets. These detectors "convert" localization into a classification problem by identifying image regions that may correspond to target objects, typically using a class-agnostic segmentation algorithm, and then classifying each region individually. For example, R-CNNs consist of the following steps: (1) region proposals are obtained according to objectness, which is evaluated by algorithms like selective search [12] and Region Proposal Networks

(RPN) [4]; (2) a neural network is used to extract features from these object proposals; and (3) a classifier is used to evaluate all these proposals. Fast-RCNNs [6] and Faster-RCNNs [4] are modified versions of RCNNs that improve speed but share the same logic. Mask-RCNN [8], the newest member of the RCNN family, includes an extra task of predicting object segmentation masks based on Faster-RCNN, which helps make the technique less sensitive to overfitting.

On the other hand, regression-based detectors, such as YOLO [13], train a neural network to map image pixels to coordinates of bounding boxes directly. Compared with classification-based detectors, regression-based detectors are more efficient in terms of both speed and memory. In fact, by avoiding the need for extracting and classifying hundreds of proposal windows, these techniques can even run on video in real-time. However, regression-based models do not perform as well as classification-based models, due in part to the foreground-background class imbalance problem [11]. Class imbalance is a classic problem in machine learning, which is usually solved by techniques like oversampling [14]. In object detection, this problem is caused by the fact that a typical detector considers thousands of candidate bounding boxes, but only a small percentage of them contain real objects. One solution is hard negative mining, which keeps difficult examples based on the current model and ignores the easy ones [5, 15], although this is time-consuming. Redmon et al. [9, 13] alleviate this problem by setting tiny weights to the background samples, but these weights must be set manually. Lin et al. [11] propose a much simpler solution called “Focal Loss” which dynamically sets different weights for different samples based on how hard they are for the current detector.

Another problem with many current detectors is hard-easy sample imbalance, which refers to the fact that detectors learn to concentrate on easy object instances while ignoring the difficult ones. This imbalance can cause both recall and precision to decrease. Although class imbalance is handled well by [11], the problem of hard-easy example imbalance is largely ignored by most standard detectors. One possible reason is that the imbalance problem is not severe for standard datasets, which contain only a limited number of objects. But in highly cluttered scenes with many objects of differing size and difficulty, such as the printed circuit boards we consider here, training may avoid learning to recognize these instances because of their small size and rarity. Furthermore, although current detectors achieve impressive performance on the prominent objects typical of most popular image datasets, performance is poor for small ones [7]. The most straightforward reason is that small objects usually contain far less evidence than bigger ones, and typical deep networks contain multiple pooling layers that tend to further obscure this weak evidence.

The identification of integrated circuits requires a fine-grained identification of visual features. Related problems in computer vision include face recognition, for which deep representation learning has been shown to be effective. Yi et al. [16] learned representation for the task of face identification and verification, for example. In face recognition, Schroff et al. [17] learned a distance metric using a Siamese Triplet network. The work of Bell et al. [18] learned visual similarity between images of two different domains of the same product with a convolutional neural network (CNN). In the microelectronics domain, the work of Wu et al. [19] addresses the task of microelectronic component *detection* in a printed circuit board (PCB) using a graph embedding network.

3 Methods

Microelectronic image analysis is a relatively new problem with many challenges to be solved. In this paper, we present techniques for two specific tasks: component detection and component verification.

In the *component detection* task, we aim to detect integrated circuits on printed circuit boards. A typical PCB may have dozens of integrated circuits (IC) of varying sizes, and many of them may be quite small. We developed an object detection algorithm to detect these small IC components with a novel “Loss Boosting (LB)” technique, which is specially designed for solving the imbalance issue between easy samples and hard examples within an image. Solving this detection problem allows us to identify the components that might be present in a PCB image.

In the *component verification task*, the goal is to identify whether two images (e.g., a detected IC on a PCB and a known reference sample) actually correspond to the same part or not. PCBs might contain erroneous or even counterfeit components, which could cause the device to malfunction, fail early, or open security vulnerabilities. This component verification task is thus crucial for ensuring the security and reliability of the microelectronic devices. Towards this end, we learn a deep neural network–based representation from a large collection of IC images, enabling our verification algorithm to verify the identity of a pair of IC images. The key idea is to present the network with pairs of IC images that are known to be different and other pairs that are known to be the same, so that the network can learn which visual features are important for distinguishing ICs (e.g., different silkscreening appearance), and which should be ignored (e.g., minor differences in image alignment).

In the following sections, we describe the details of our detection and verification method.

3.1 Microelectronic Component Detection

While IC component detection is simply a specific case of general object detection, ICs present unique challenges compared with most of the detection problems that have been studied in the literature. Unlike most objects in consumer images like cars, people, and airplanes, ICs are often densely packed, are quite small, and do not have distinctive visual features. We want to use deep neural networks to take advantage of their powerful approximation ability. However, very deep networks also lose information about small objects, because of repeated pooling layers. To address this problem, we combine two major ideas: loss boosting to help ensure that small and infrequent object instances are not ignored, and a pre-processing step to help identify boundaries of small and featureless objects. We now describe two techniques in detail.

3.1.1 Loss Boosting

We found that many bounding boxes detected by traditional detectors like Faster R-CNN [4], YOLO-v2 [9], and SSD [10] are not accurate on datasets with small, densely distributed objects. One reason is that small objects are more sensitive to bounding box boundaries. The metric typically used to judge whether a bounding box is correct or not is intersection-over-union (IoU): the area of the intersection of the estimated bounding box and ground truth bounding box, over the area of the union of these two boxes. For small objects, small offsets can have a catastrophic effect on the intersection-over-union score. The other problem is the unbalanced relationship between the easy examples and hard examples within a single image: easy object instances often dominate the total loss at the expense of the loss from harder objects. In other words, the loss from the easy objects keeps decreasing while the loss from the hard ones becomes stagnant. This problem is not severe for many datasets with prominent objects since the IoU threshold is relatively small (IoU = 0.5), and is not as much of a problem for consumer image applications where precise object localizations are often not needed. However, for the scenario of application in the industrial world, both high precision and high recall are important.

We thus propose Loss Boosting (LB) to solve this issue. Specifically, for each training sample, we split all the objects into two groups, easy and hard, according to their IoU in the current iteration of training the loss of easy objects. The weight of each object is dynamic during the training, depending on its current relative contribution to the loss function against all other objects' loss contributions in the same image (as Eq. 2). Loss Boosting is an idea that can be embedded within any type of detector including Faster R-CNN [4], YOLO-v2 [9], or SSD [10]. The final stage of all three types of detectors can be categorized as bounding box regressors. Here we take the regressor of YOLO-v2 [9] as an example. After grouping, the loss function becomes

$$\begin{aligned} loss &= e^{+}\sum_{i=1}^{n}(loss_{obj} + loss_{coord} + loss_{class}) \\ &+ e^{-}\sum_{i=1}^{m-n}(loss_{obj} + loss_{coord} + loss_{class}), \end{aligned} \tag{1}$$

where e^{+}/e^{-} represents an additional coefficient that is larger or smaller than 1. The total number of objects in an image is **m**, while the number of predicted boxes whose IoU is smaller than a given threshold (for example, **TH** < 0.5) is **n**. $loss_{obj}$, $loss_{coord}$, and $loss_{class}$ represent the loss of objectness, loss of coordinates, and loss of classification respectively (please refer to YOLO [13] for more details). For each image in each iteration, we calculate e^{+}, e^{-} as,

$$e^{+/-} = \begin{cases} (\frac{m}{n})^{\alpha} & (IoU <= TH) \\ (\frac{n}{m})^{\alpha} & (IoU > 1 - TH). \end{cases} \tag{2}$$

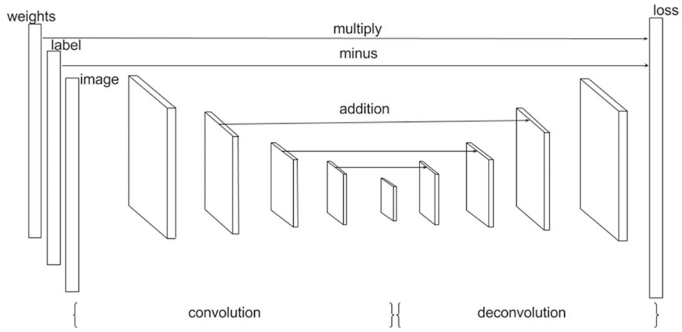
Note that α is a parameter whose value is smaller than 1.

We see from Eq. 2 that the final loss of an object depends on its relative importance compared with all other objects in the same image. Imagine a typical circumstance: early in the training process, most objects' corresponding IoUs are smaller than **TH**, so their losses are unchanged, meaning that they are equally important. After thousands of training epochs, some easy objects will achieve an IoU that is greater than **(1-TH)**, so their loss will be decreased. At the same time, the hard objects whose IoUs are smaller than **TH** have their loss increased. Finally, when the model approaches the optimal minimum, most exemplars should have an IoU which is larger than **1-TH**, and their loss stays unchanged just like in the early training period—i.e., they become equally important again.

Although LB is used to address the issue of easy-hard imbalance in a single image, it can also be generalized to balance different types of loss across the whole dataset. For example, we can use LB to balance different types of loss (objectness loss, coordinate loss, and classification loss). We can think of the loss weights of different loss types as model-level balancing and the loss weights of different objects (LB) as image-wise balancing.

3.1.2 Data Pre-processing

In this step, we extract 9 overlapping patches from the positions of top-left, top-center, top-right, middle-left, the center, middle-right, bottom-left, bottom-center, and bottom-right in a single image. Each patch is resized to the input size. Then, we use an encoder-decoder structure (as Fig. 1 shows) to reconstruct the boundaries of each object. Note that we add some shortcut connections between



(a)

Type	Filters	Stride	Output
Conv1	16	13×13/2	245×245
Conv2	32	3×3/2	122×122
Conv3	64	4×4/2	60×60
Conv4	128	4×4/2	29×29
Conv5	256	3×3/2	14×14
Deconv5+Conv4	128	3×3/2	29×29
Deconv4+Conv3	64	4×4/2	60×60
Deconv3+Conv2	32	4×4/2	122×122
Deconv2+Conv1	16	3×3/2	245×245
Deconv1	3	13×13/2	501×501
Euclidean Loss	N/A	N/A	501×501

(b)

Fig. 1 Neural network architecture for microelectronic components detection and parameters used for pre-processing

some symmetric layers. We make convolutional layers or deconvolutional layers behave like filters and amplifiers through pixel-level labels and pixel-level weights. Note that the "weights" here refer to the relative importance of a label in terms of pixel values. The "label" here is the ground truth boundaries. The loss function is calculated by the Euclidean distance between the labels and the predicted features. Finally, the loss is multiplied by weights for each pixel.

This step is similar to multi-scale training, which helps make an algorithm more robust to scale since it forces the algorithm to "see" multiple scales of input at training time. However, here we use this step to obtain clearer boundaries for the objects. Intuitively, this is like what humans do when they try to find a small object: they take a "closer look" at the image. We benefit from this process not only through using more input images but also by the complementary relationship among patches. In testing, we just need to assemble the results by location and average.

3.2 Identity Verification of Microelectronic Components

Another key task to ensure the reliability and integrity of devices is to verify that their electronic components are the ones that were expected. Towards this goal, we envision developing a method to verify that the appearance of a given microelectronic component (in an image) is consistent with the image of a reference part. Of course, the two images will never be *identical* because of variations in part alignment, lighting, scale, etc., as well as the fact that the two parts are typically different physical instances and there may be small variations due to manufacturing variations, etc. The goal is thus to learn feature representations that can spot visual differences that may indicate an incorrect or counterfeit part, while ignoring the visual differences that are caused by harmless factors like lightning or minor manufacturing variations.

Ideally, we would test our approach using a real dataset of counterfeit electronic parts. Unfortunately, obtaining images of known counterfeits is quite difficult, and the datasets that are publicly available[2] are not large enough for training deep neural networks. We thus study a proxy problem: verifying if two IC images have the same manufacturer (AMD, Intel, etc). This is still a challenging recognition problem, and the system must still automatically learn how to distinguish between the logos and other visual features that are specific to the different companies.

In more detail, we learn this representation from a collection of IC images using deep neural networks. We use a Siamese network architecture that takes pairs of images as training examples and then learns a distance metric in a high-dimensional feature space. For our verification module, we want to learn a distance metric that clusters together images of similar identities while images of dissimilar identities are pushed further apart. Towards this goal, we learn a distance metric with our Siamese network with a joint combination of *Contrastive* and *Classification* losses. Since our goal is to separate ICs by manufacturer, we used the manufacturing company label in the classification loss of our joint loss function.

3.2.1 Network Architecture

A Siamese network contains two copies of the same deep neural network with shared weights between them. During training, pairs of input images are fed into the network at a time, along with a label indicating whether the two images correspond to the same class (manufacturer) or not. Each copy of the shared network thus takes one of the two images as input. Assume that two input images, denoted by X_a and X_b, are fed into the Siamese network and their feature representations in the final layer (before loss layer) are a and b, respectively. A Siamese network can be trained from

[2] http://counterfeitic.org

a batch of N images to learn a distance metric using a contrastive loss function,

$$loss_{contr} = \sum_{i \in N} z_i ||a_i - b_i||^2 + \\ +(1 - z_i) max(0, \beta - ||a_i - b_i||)^2, \qquad (3)$$

where z_i is an indicator that denotes whether the two images are of the same or different class. The *margin* hyper-parameter is denoted by β. We also incorporated two other classification loss terms for the two input images in addition to the contrastive loss, encouraging each branch to correctly identify the class (manufacturer) of its individual input image.

At test time, learned features can be extracted from an image by passing it through a single copy of the network, and then extracting the output of the final layer (before the loss layer). This representation is simply a vector that hopefully captures the important visual properties of the image for the given recognition class (manufacturer detection), while ignoring distracting or irrelevant visual features.

4 Experiments

We have evaluated our techniques through extensive experimentation on several realistic datasets. We briefly describe our dataset and then discuss the experiments for each module in the following sections.

4.1 Microelectronic Component Detection Experiments

4.1.1 IC Detection Dataset

We applied our detection algorithm for detecting integrated circuits on complex printed circuit boards (PCBs). To our knowledge, there are no existing publicly available PCB datasets that are suitable for realistic, large-scale training and testing on this task. For example, Pramerdorfer and Kampel [20] collected a dataset of 165 PCB images, but only a subset is labeled and many of the small ICs are ignored. We thus used our own dataset of 1500 images of PCBs collected from Internet sources (e.g., Google Image Search) by searching for various relevant keywords and image queries [21]. Images of low resolution, poor

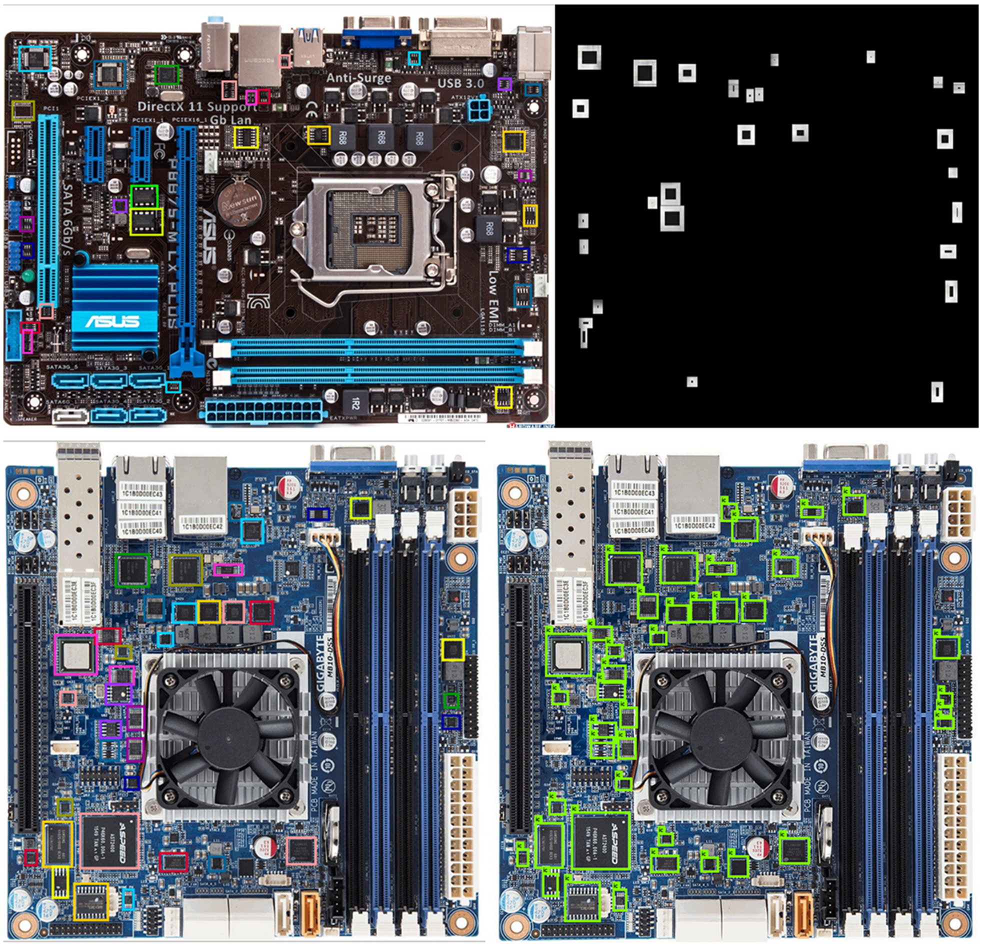


Fig. 2 Top left: a sample input image with ground truth bounding boxes. Top right: the corresponding image with predicted boundaries after pre-processing. Bottom left: another input image with ground truth bounding boxes. Bottom right: the corresponding detection result

Table 1 Results on PCB dataset. "*" means the pre-processing step is applied. "anchors" here means a few pre-defined bounding boxes to take advantage of the prior knowledge of the training set (please check the details in [9])

				Recall		Precision	
Method	Anchors	Backbone	Avg IoU	IoU = 0.9	IoU = 0.8	IoU = 0.9	IoU = 0.8
SSD	Yes	Inception-v2	78.95%	14.79%	32.54%	9.72%	21.37%
YOLO-v2	Yes	YOLO-v2	75.57%	3.11%	47.28%	2.04%	18.30%
YOLO-v2+*	Yes	YOLO-v2	89.49%	62.44%	91.32%	45.77%	66.95%
YOLO-v2+*	No	YOLO-v2	88.93%	59.84%	89.12%	52.98%	78.90%
YOLO-v2+*+LB	Yes	YOLO-v2	89.84%	66.45%	92.88%	47.06%	65.78%
YOLO-v2+*+LB	No	YOLO-v2	89.11%	59.84%	91.84%	52.92%	81.21%

quality, or noisy appearance, etc. were discarded by hand. Four researchers annotated 483 images by hand, with bounding boxes and IC ID labels, yielding around 5000 labeled IC instances, using a customized version of the LabelMe [22] web-based user interface. The data was partitioned into a total of 417 training images and 66 testing images. The PCB images and the annotated is available at http://homes.soic.indiana.edu/IU-PCB-Dataset.

Some examples of the dataset are shown in Fig. 2. This task is important in industrial quality control scenarios where, for example, a manufacturer may wish to verify that all ICs have been properly placed on a circuit board that was produced by a third-party supplier.

4.1.2 Data Preprocessing

We use the network structure shown in Fig. 1. The inputs are raw images, along with ground truth segmentation maps in the form of a binary image, with a pixel value within each ground truth bounding box of 1.0 and the background pixels set to 0 (since these maps can be thought of as probabilities). We set the learning rate to $1e^{-6}$, the momentum to 0.9, and the maximum iteration to 80,000. We conducted the experiments using the Caffe framework [23].

4.1.3 Loss Boosting

After estimating clear boundaries, we use the regression model of YOLO-v2 [9] with loss boosting to predict final bounding boxes. We set $\mathbf{TH} = 0.1$, $\alpha = 0.1$, and we have two classes: IC and background. During training, we vary the input image sizes from 480 × 480 to 800 × 800 randomly, but we only use 800 × 800 during testing. Other meta-parameters are as follows: learning rate $1e^{-3}$ (reduced by multiplying a factor of 0.1 at iteration 40,000 and 60,000), maximum iteration 80,000, momentum 0.9,

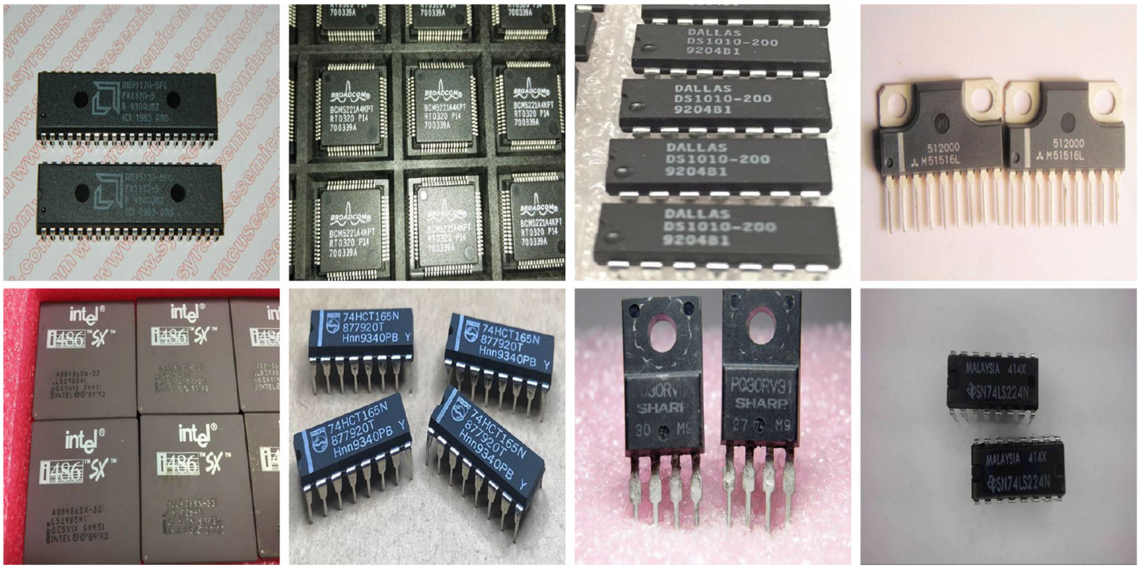


Fig. 3 Test images containing multiple instances of IC images. A random pair was used as positive examples in our verification experiment. Top row (from left to right) shows sample images from *amd*, *broadcom*, *dallas*, and *mitsubishi* manufacturing companies respectively. Similarly, in bottom row, sample images are from *intel*, *philips*, *sharp*, and *texas-instruments* manufacturers

and decay 0.0005. As baselines, we also applied SSD [10] using the TensorFlow object detection API [24]. We use the default parameters except that the input size is modified to 600 × 600 and the learning rate is set to 0.01. We also tried Faster-RCNN [4] on the same dataset, but the results were very poor (a precision and recall close to 0). We obtained this same result from both the Faster R-CNN implementation provided by the TensorFlow object detection API [24] and the official code provided by the authors [4]. We believe the problem is that Faster R-CNN was simply not designed for small, relatively featureless objects such as ICs.

Some detection examples are shown in Fig. 2. Most of the time, objects are detected accurately. Table 1 shows quantitative accuracy measurements for two different Intersection over Union (IoU) thresholds (0.8 and 0.9; higher thresholds require bounding boxes to agree more precisely with the ground truth boxes for them to be considered correct). For example, for an IoU of 0.9, we see that the combination of our techniques, including the pre-processing step and loss boosting, achieves an average IoU of 89.84%, recall of 66.45%, and precision of 47.06%, which is much better than corresponding measurements achieved by YOLO-v2 (75.57%, 3.11%, and 2.04%). The avgIoU refers to the average IoU of all the true positives. Similarly, our model also performs favorably compared with SSD [10], which achieves 78.95%, 14.79%, and 9.72%, respectively.

We also observed failure cases from our detection method. We were often able to categorize them into one of two types.

First, some small ICs are simply not found by our model. These ICs are particularly challenging and could be detected if we assign more weight to these samples in loss boosting, but this could cause a drop in accuracy for easy samples. One possible solution to circumvent this problem would be to detect the larger and smaller samples separately. Small object detection, in general, is a difficult problem. The primary reason for this difficulty could be attributed to the phenomenon that small objects contain less information compared with larger ones. We consider these small objects as difficult objects and detect them by balancing the loss between easy objects and difficult ones. Our experiments show that the detection accuracy is improved after the balanced loss is introduced. However, there are some small objects that can not be detected at all. One possible reason is the generalization gap—the performance difference between the training set and testing set—and this issue could be further softened by incorporating more training images, or enforcing the model to focus more on only small objects.

The second type of failure corresponds to false positives for components that have a very similar appearance to real ICs, such as rectangular transformers or connectors. One possible solution to this or to build classification models for these parts so that they are less likely to be detected as ICs.

4.2 Microelectronic Component Verification Experiments

4.2.1 Integrated Circuit Dataset

While the IC detection algorithm could be evaluated on the PCB dataset above, for verification we needed a larger dataset of cropped IC images with more annotations. As mentioned above, while the technique we develop for IC matching could in principle be used to learn visual properties appropriate to any task, such as verifying if a part is counterfeit or legitimate, here we study the specific proxy problem of checking whether the manufacturers of two given IC images are the same.

We thus collected a large repository of IC images from the Internet using various sources including Google Image Search, eBay product photos, and electronics manufacturer websites. We collected about 10,000 images from more than 30 IC manufacturers. We then manually cropped the IC from the source image, applying a perspective transformation (a homography) if needed to rectify the image so that the angles of the IC were 90°. We removed images that were especially low quality, yielding a total

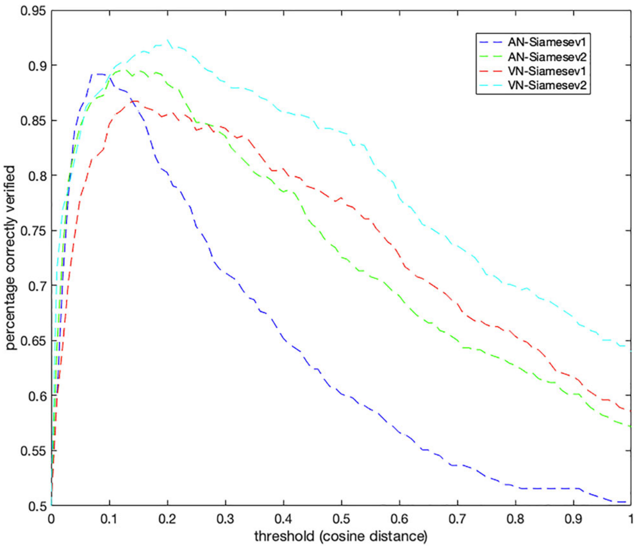


Fig. 4 Accuracies of pairs of IC images against different thresholds. Each curve shows the accuracy of one of our different Siamese network models. The best performance was achieved (shown in cyan) by our VN-Siamesev2 network containing the backbone of VGG16 architecture

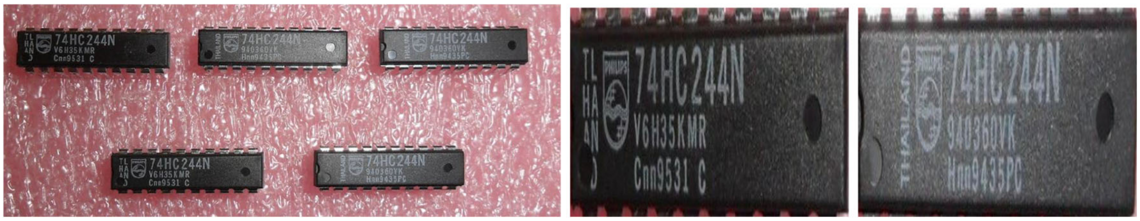


Fig. 5 A sample test image (left) containing multiple instances of ICs. We extract two ICs randomly (middle and right) and then mark the pair as a positive example in our verification experiment

collection of 6387 IC images from 27 manufacturers. The number of ICs per manufacturer varied from about 650 to 50, and the manufacturers were as follows (listed in decreasing order of frequency): NEC, TI, Toshiba, AMD, Sanyo, Hitachi, Fujitsu, Motorola, Intel, Mitsubishi, Philips, Siemens, Dallas, Apple, Oki, Infineon, Qualcomm, Via, Nuvoton, Sharp, Yamaha, Panasonic, Sony, Broadcom, Mediatek, Atheros, and AT&T.

We used over 8000 pairs of IC images from this dataset to train our Siamese networks to learn the features. A set of *286* images containing multiple instances of the same IC chip was selected. We utilized a random pair of two images from these source images for our verification experiment. Figure 3 shows a few sample images used in the verification test.

4.2.2 Siamese Network Training

We trained four different Siamese network architectures from the literature to learn our deep embedding from the dataset of IC images. As is common, we used networks that were pre-trained on the ImageNet [2] large-scale dataset of millions of images. Even though ImageNet consists of consumer images having nothing to do with ICs, much work in transfer learning has shown that this initialization can be helpful when the dataset in the target domain is relatively small. In particular, we tried for variations of the architecture:

- AN-Siamesev1: AlexNet [25] is used as the backbone network and only the fully connected layers are fine-tuned with IC images.
- AN-Siamesev2: AlexNet [25] is used as the backbone network and all layers are fine-tuned with IC images.
- VN-Siamesev1: VGG16 [26] is used as the backbone network and only the fully connected layers are fine-tuned with IC images.
- VN-Siamesev2: VGG16 [26] is used as the backbone network and all layers are fine-tuned with IC images.

The networks were trained in PyTorch framework for at most 200 epochs using a learning rate of 10^{-6} (Fig. 4).

4.2.3 Verification Experiment

Given a pair of IC images, our goal at test time is to determine if the two are similar according to the task that the network was trained on. In the proxy problem in this paper, this task is to determine if the two ICs were made by the same manufacturer. From the *286* images that were held out from training as test images (a few examples as shown in Fig. 3), we randomly cropped two instances of the same IC from a source image and created a pair. This pair was labeled as positive since they were from the same manufacturer. Figure 5 shows an instance of this cropped pair of IC images along with the source image from which it was cropped. We also applied perspective transformation after cropping. Emphasis was given during cropping to preserve the visual characteristics of a chip such as shape, aspect ratio, logo of the manufacturer, and chip identification number. The total number of positive pairs of IC images in the test set was 286. We also included the same number of negative pairs of IC images in our test experiment. For each of the IC images in a positive pair, we randomly selected another IC image from a pair of a different manufacturer. In total

Table 2 Verification accuracy of different architectures. The best accuracy is shown in this table in italics. Accuracy vs threshold trade-off is shown in Fig. 4

Method	Backbone	Trained FC layers only	Best accuracy	Best threshold
AN-Siamesev1	AlexNet	Yes	89.16%	0.07
AN-Siamesev2	AlexNet	No	89.51%	0.12
VN-Siamesev1	VGG16	Yes	86.71%	0.14
VN-Siamesev2	VGG16	No	*92.31%*	0.20

our test set contains *572* pairs of positive and negative images.

Each image from a pair was passed through our trained network to extract a 1024-dimensional feature vector from the last fully connected layer. As a measure of similarity between two features, we compute the cosine distance between these two 1024-dimensional vectors (which should be small if the two images are similar according to the learned distance metric). Based on the distance value between the feature representations, we predict if a pair of images are from the same or different manufacturer by comparing it to a threshold. We compute the accuracy of all the 572 pairs of IC images as quantitative evaluation.

Table 2 presents the results. The best accuracy of *92.31%* was achieved using our *VN-Siamesev2* network, which was trained by learning all the layers from the IC images. *VN-Siamesev2* contains the VGG16 backbone, which is a few layers deeper than *AN-Siamesev1* or *AN-Siamesev2* (which both use AlexNet as their backbones). The best performing model using AlexNet as a backbone was *AN-Siamesev2*, with an accuracy of 89.51%. We also report the threshold used to obtain these best accuracy next to each row in Table 2. For a more comprehensive understanding of the accuracy versus threshold trade-off, we visualized the evolution of accuracy in Fig. 4. We varied the threshold from 0 to 1 with a step size of 0.01. Each curve in Fig. 4 denotes the accuracy of one of the Siamese network models. For example, the *cyan* curve (best viewed in color) denotes the accuracy trade-off when running the experiment with the *VN-Siamesev2* architecture. This model performs better than the rest of the models across all the threshold, and has an accuracy of 64% even at a very high threshold. *AN-Siamesev1*'s performance quickly diminishes as the threshold increases, suggesting that it failed to compute a larger distance between images of dissimilar IC chips.

5 Conclusion

We proposed two computer vision techniques to analyze microelectronic imagery. With a novel loss boosting module, our detection method allows for the identification and localization of different components in a printed circuit board image. We also developed a Siamese network module for the identity verification of the microelectronic component images. We demonstrated the effectiveness of both of our methods with an extensive evaluation on a large corpus of images collected from the web.

Acknowledgments This research used computer facilities donated by Nvidia, Inc., and the Romeo FutureSystems Deep Learning facility, which is supported in part by Indiana University and the National Science Foundation (RaPyDLI-1439007).

Funding Information This research was sponsored by the Naval Engineering Education Consortium (NAVSEA) contract N00174-16-C-0016, with support of NSWC Crane Division in Crane, IN. It was also funded in part by the Indiana Innovation Institute (IN3).

References

1. Tehranipoor M, Guin U, Forte D (2015) Counterfeit integrated circuits: detection and avoidance. Springer
2. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015) ImageNet large scale visual recognition challenge. Int J Comput Vis (IJCV) 115(3):211–252
3. Lin T, Maire M, Belongie SJ, Bourdev LD, Girshick RB, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft COCO: common objects in context, arXiv: 1405.0312. [Online]
4. Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in neural information processing systems (NIPS)
5. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR
6. Girshick RB (2015) Fast R-CNN, arXiv:1504.08083. [Online]
7. Chen C, Liu M, Tuzel O, Xiao J (2016) RCNN for small object detection, ACCV
8. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask R-CNN, arXiv:1703.06870
9. Redmon J, Farhadi A (2016) Yolo9000: better, faster, stronger. arXiv:1612.08242
10. Liu W, Anguelov D, Erhan D, Szegedy C, Reed SE, Fu C, Berg AC (2015) SSD: single shot multibox detector. arXiv:1512:02325
11. Lin T, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection. arXiv:1708.02002
12. Uijlings J, van de Sande K, Gevers T, Smeulders A (2013) Selective search for object recognition, IJCV
13. Redmon J, Divvala S, Girshick R, Farhadi A (2015) You only look once: unified, real-time object detection. arXiv:1506.02640
14. Santos MS, Abreu PH, García-Laencina PJ, Simão A, Carvalho A (2015) A new cluster-based oversampling method for improving survival prediction of hepatocellular carcinoma patients. J Biomed Inform 58(Supplement C):49–59
15. Shrivastava A, Gupta A, Girshick R (2016) Training region-based object detectors with online hard example mining. arXiv:1604.03540
16. Yi S, Chen Y, Wang X, Tang X (2014) Deep learning face representation by joint identification-verification. In: Advances in neural information processing systems (NIPS)
17. Schroff F, Kalenichenko D, Philbin J (2015) Facenet: a unified embedding for face recognition and clustering. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
18. Bell S, Bala K (2015) Learning visual similarity for product design with convolutional neural networks. In: ACM Transaction on graphics (SIGGRAPH)
19. Kuo C-W, Ashmore J, Huggins D, Kira Z (2019) Data-efficient graph embedding learning for PCB component detection. In: Winter conference of computer vision (WACV)
20. Pramerdorfer C, Kampel M (2015) A dataset for computer-vision-based PCB analysis. Machine Vision Applications
21. Chen Z, Wanyan T, Rao R, Cutilli B, Sowinski J, Crandall D, Templeman R (2017) Addressing supply chain risks of microelectronic devices through computer vision. In: IEEE Applied Imagery Pattern Recognition Workshop (AIPR)
22. LabelMe. [Online]. Available: http://labelme.csail.mit.edu

23. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T (2014) Caffe: convolutional architecture for fast feature embedding. arXiv:1408.5093
24. Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, Murphy K (2016) Speed/accuracy trade-offs for modern convolutional object detectors. arXiv:1611.10012. [Online]
25. Krizhevsky A, Sutskever I, Hinton G (2012) Imagenet classification with deep convolutional neural networks. In: Proceedings of International Conference on Neural Information Processing Systems (NIPS)
26. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: International Conference of Learning Representations (ICLR)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.