Automated Defect Inspection in Reverse Engineering of Integrated Circuits

Ann-Christin Bette[12], Patrick Brus[2], Gabor Balazs[12], Matthias Ludwig[12], Alois Knoll[1]
[1] Technical University of Munich, Germany, [2] Infineon Technologies AG, Munich, Germany
ac.bette@tum.de, prename.surname@infineon.com, knoll@in.tum.de

Abstract

In the semiconductor industry, reverse engineering is used to extract information from microchips. Circuit extraction is becoming increasingly difficult due to the continuous technology shrinking. A high quality reverse engineering process is challenged by various defects coming from chip preparation and imaging errors. Currently, no automated, technology-agnostic defect inspection framework is available. To meet the requirements of the mostly manual reverse engineering process, the proposed automated framework needs to handle highly imbalanced data, as well as unknown and multiple defect classes. We propose a network architecture that is composed of a shared Xception-based feature extractor and multiple, individually trainable binary classification heads: the `HydREnet`*. We evaluated our defect classifier on three challenging industrial datasets and achieved accuracies of over 85 %, even for underrepresented classes. With this framework, the manual inspection effort can be reduced down to 5 %.*

1. Introduction

Microchips can be found in all areas of life these days. With ever-growing complexity and increasing outsourcing to third parties, validation of device integrity fosters safe and secure applications. Reverse engineering (RE) enables an in-depth analysis of the structure and functionality of the individual hardware components of microchips, known as integrated circuits (ICs). This process allows searching for potential intellectual property infringements, performing benchmarking studies on competitor products, but also detecting possible malicious modifications and security vulnerabilities [29]. The basic steps of the RE process include [20, 30]:

- *Delayering* to expose the internal components of the chip and analyze the die layer by layer
- *Imaging* with a scanning electron microscope (SEM) to digitize each die layer
- *Feature extraction* of the standard cells, metal tracks and vertical interconnection accesses (VIAs)

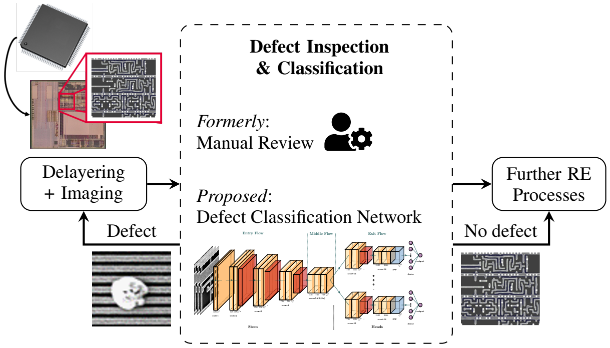


Figure 1: Reverse engineering flow for integrated circuits extended with proposed automated defect inspection.

Moore's Law reveals a major problem of physical RE. With each new generation of technology, the effort of delayering and imaging microchips is increasingly challenging. Consequently, the image quality becomes unreliable and successful feature extraction cannot be guaranteed. The success depends largely on the equipment of the laboratory as well as the skills and experience of the failure analysis (FA) expert: Etching and polishing processes remove the single chip layers down to the level of interest. Exposure of metal layers is particularly difficult with advanced technologies because the layer thicknesses, typically less than $150\,\mathrm{nm}$, require overall planarity. The SEM settings are chosen at the discretion of the expert, matched to the technology to be visualized and the previous delayering methods. Both the ablation of the material and the manual settings on the microscope lead to errors, so-called defects, in the resulting images. These defects propagate unnoticed into subsequent steps and jeopardize the entire RE process. The division into distinct defect classes enables the feedback loop to support the FA and RE experts (Sec. 3.1).
In industrial manufacturing, defect classification using neural networks is very common. These fitted models are applied to consistent data in controlled environments. In this work, we propose a neural network-based defect inspection system for RE (Fig. 1). A particular challenge is the varying input data due to technology differences, as well as man-

ual chip delayering and imaging. The generalization of an adapted model thus appears very difficult. Therefore, we present a network architecture that is extensible, adaptable, flexible in deployment, enables low training overhead, and provides strong control over the learning process.

The key contributions of this work can be summarized as follows:

- To the best of our knowledge, this is the first automatic defect inspection framework for RE of ICs that evaluates defects in the metal layers and uses neural networks to perform the classification task.
- A network architecture focused on the training time and control of training process: the `HydREnet`
- A highly data-centric approach to enable technology-agnostic defect classification, including the identification of unknown defects

2. Related Work

2.1. Reverse engineering

Several articles have been published describing the complete RE process [14,16,21]. Lin *et al.* [15] describe a typical framework of how neural networks are used in RE: SEM images of each chip layer are acquired, stitched, then the features are extracted to finally stack the individual layers. The segmentations for extracting the features are performed by neural networks. The authors use the generative properties of the network architectures [17,22] to correct errors directly in the image. Hong *et al.* [9] address the occurrence of errors directly by categorizing their images according to the degree of noise as a preprocessing step.

We are critical of the error correcting approach because very small or merely synthetic data were used due to the lack of publicly available data. We expect severe overfitting and lack of generalizability to other technologies with this process-centric approach. Many authors reject the use of deep learning altogether due to the lack of training data and restrict themselves to purely rule-based traditional approaches for feature extraction [18,33,34].

Botero *et al.* [1] discuss the problems of using machine learning (ML) methods in the RE of ICs. They mention feature extraction and netlist generation as the two most promising application areas of ML methods. Like us, they demand that the first step is to ensure that the image content is not corrupted by delayering and stitching errors.

Courbon [4] is the first to propose defect control on the way to feature extraction. The rule-based algorithms deal specifically with defects at the standard cell level of the IC. Our tool is optimized for exposing the metal layers, where there are more degrees of freedom overall. The layers are difficult to expose due to the low layer height, and imaging is difficult for modern nodes due to lack of material contrast of copper traces and copper VIAs.

2.2. Industrial defect inspection

Deep Learning based automated surface and texture defect inspection systems [6, 35] are now an integral part of manufacturing. In general, the experiments and neural network architectures are often evaluated against the NEU [13], the synthetic DAGM [32], or the PCB defect [28] datasets.

We see the greatest similarity to our use case in the requirements of wafer surface inspection in semiconductor manufacturing. Imoto *et al.* [12] propose a defect classification based on a convolutional neural network (CNN) system that assists the expert in his decision making and reduces the manual effort to one-third. Images were acquired using an SEM and performance was improved by transfer learning. Cheon *et al.* [3] extend this approach by detecting unknown defects. They realize this with the help of a kNN algorithm. They justify the necessity of the addition with the dynamic manufacturing environments in which new defects arise irregularly. Harada *et al.* [7] also propose an approach to defect classification that is robust to process variations. They realize this by comparing the imaged part of the wafer with a defect-free reference image.

Our use case differs from semiconductor manufacturing in two ways: First, the errors are caused by human hands during the delayering, not by machines during the manufacturing process. Second, we do not have a static environment. The technologies under test change with each analysis, and the SEM settings must be manually adjusted each time.

2.3. Computer vision

Mullapudi *et al.* [19] propose the HydraNets as a dynamic architecture to perform efficient image classifications. HydraNets specialize in computing features for visually similar classes and maintain efficiency by using a gating function to select only a small number of components for inference. We adapt this architecture to gain more control over each training and inference.

Following Tsoumakas *et al.* [31] we basically distinguish two methods for solving multi-label tasks: First, transformation methods that transform multi-label classifications into one or more single-label classifications. Second, algorithm adaptation methods, i.e., methods that adapt single-label classifications to directly process multi-label classifications. We compare these two approaches with our proposed network architecture.

Minority class oversampling by synthetic images (SMOTE) [2] has led to strong performance improvements in many areas for imbalanced datasets. We have so far used this approach only for general data enrichment. In the future, targeted enrichment of individual samples of interest, e.g., through generative image generation with SinGan [24], would be feasible and worth trying.

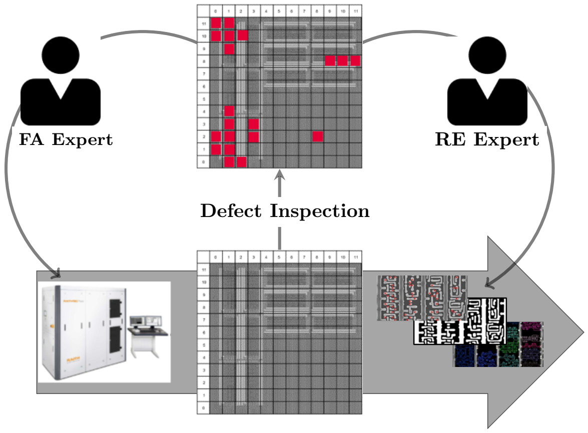
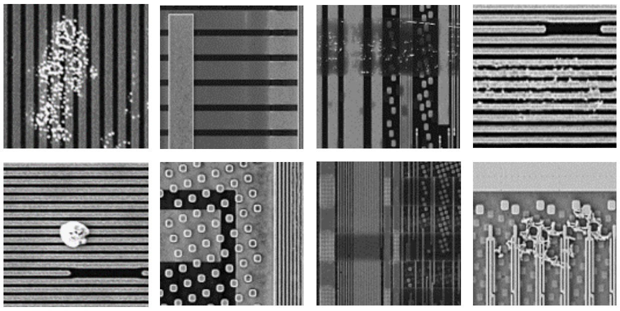


Figure 2: Defect inspection framework which serves as feedback tool for human experts.



(a) Particles on top (b) Underlying layer visible (c) Overlying layer visible (d) Smeared structures

Figure 3: Example images of the four defect classes.

3. Proposed Methodology

In this section we present the individual components of our work: The defect inspection framework, the data used in the process, the network architecture of the proposed defect classifier, and finally details on the training to be performed.

3.1. Defect inspection framework

For a complete IC analysis, the FA expert first delayers the chip to the top metal layer and then digitizes the surface with the SEM. This process is repeated for each metal layer, the transistor level and the active area. The SEM scans the area of interest line-by-line and column-by-column, capturing individual images of a predefined field of view. This results in thousands of individual images per scanned layer, which are stitched together by a predefined overlap. The pixel size needs to be a few nanometers to reveal the smallest structures and their distances from each other, which are defined in the technology node. These images are forwarded to the RE expert to extract the important features using various image processing techniques. Then, the RE expert can continue the analysis in different ways, depending on the objectives.

The FA expert has so far only examined the delayered chip surfaces with an optical microscope for rough defects. The RE expert randomly inspected the obtained layout images. To replace this inadequate process, we are introducing a stand-alone defect inspection tool (Fig. 2) that triggers two different feedback processes: First, the FA expert receives feedback on the status of his chip preparation so that he can make necessary corrections and take appropriate error-reducing measures in the future. Second, the RE expert can assess whether manual error correction is worthwhile and select suitable feature extraction algorithms based on the type of defect. Therefore, the classification into the different types of defects is important for our application.

Each captured image is reviewed for errors using our proposed neural network. Afterwards, the results are presented in the stitched overview picture as a report. This allows a component specific analysis of the chip area. SEM imaging is based on the representation of material and topography contrasts. Therefore, defects can be very inconspicuous if they do not differ in material or stand out from the chip surface. Consequently, histogram and edge detection methods are not sufficient for defect inspection.

3.2. Dataset

Input images and labels The 252 original SEM images have a resolution of $4000\,\text{px} \times 4000\,\text{px}$ and are stored as 8-bit gray scale images. The images are further decomposed into $250\,\text{px} \times 250\,\text{px}$ crops, which serve as input for the neural network. As a result, 256 overlap-free partial images are generated from each original image. We have refrained from rescaling, but will integrate this iteratively in the future. The pixels of each image are normalized to values between 0 and 1. To enable supervised learning, a label must be assigned to each $250\,\text{px} \times 250\,\text{px}$ input image. If a defect is present, the entry in the defect label is set to 1, otherwise it is set to 0.

Defect classes We defined four defect classes (Fig. 3). The first defect class contains *Particles* which are located on the sample surface and often appear very bright because of their non-uniform topography. The second class contains images of the class *Underlying layer visible*. This defect is characterized by the overlap of two metal layers. The falsely visible underlying layer has almost the same gray values as the layer of interest. The third class contains images of *Overlying layer visible* which usually appears as darker areas on top of the metal layer of interest. The final defect class contains defects of *Smeared structures* which result from overly invasive polishing and etching processes.

Table 1: Dataset distribution for a total of 64 270 images.

(a) High class imbalance which is preserved after 70–30 split.

Defect class	Train	Val	Test
Particle	2360	1180	1553
Underlying	2386	1234	1593
Overlying	962	500	641
Smeared	38	20	23
None	24 817	12 364	15 871

(b) Multi-label classification task.

Defects	Images
0	53 052
1	9951
2	1262
3	5
4	0

Data distribution The dataset contains 64 270 partial images and has two issues: First, there is a high class imbalance due to the decomposition of the images (Tab. 1a). *None* indicates the error-free samples, but is generally not a class of its own. Second, multiple defects can occur in one image. Therefore, we refer to this as a multi-label classification task (Tab. 1b).

Data augmentation For the selection of the data augmentations, we relied on the characteristics of the IC structures and the imaging with the SEM. As a baseline, we chose horizontal and vertical reflection, as well as rotation in 90-degree increments. Other rotation angles are not useful because the metal structures in the IC layout images are always horizontal or vertical. According to Sim *et al.* [25], Gaussian noise is the most common type of noise in SEM images. Thus, Gaussian noise in varying strength is added to augment the data. As further augmentation techniques Gamma contrast and brightness are varied in the input images. Elastic deformations [22] are applied to vary the shape of the defects. Motion blur simulates smaller pixel dwell times, thus a faster scanning process. Zooming out by up to 50 % helps to adapt to different pixel sizes of the defects.
All of the above methods improved the performance of the model and were finally used. Only brightness and contrast changes were never made simultaneously, as this led to unrealistic image effects. The best performing strategy was the one that left 10 % of the data unaugmented.

3.3. Network architecture

Backbone CNN We compared different state-of-the-art CNN architectures for the initial multi-label classification task. We evaluated the F_1-score averaged over the individual defect classes (Fig. 4). The ResNet-based approaches [8] achieve solid performances. SEResNeXt [10] performs best, but has twice as many parameters as ResNet50. The EfficientNets [27] based on compound scaling perform comparatively poorly. Overall, the ResNets, DenseNet121 [11], and the Xception network [5] perform similarly well. Our approach is based on the Xception network, but could also be realized with any other architecture.

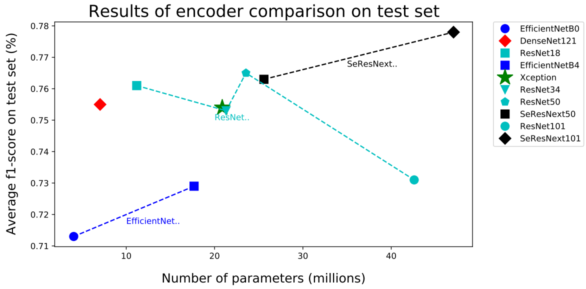


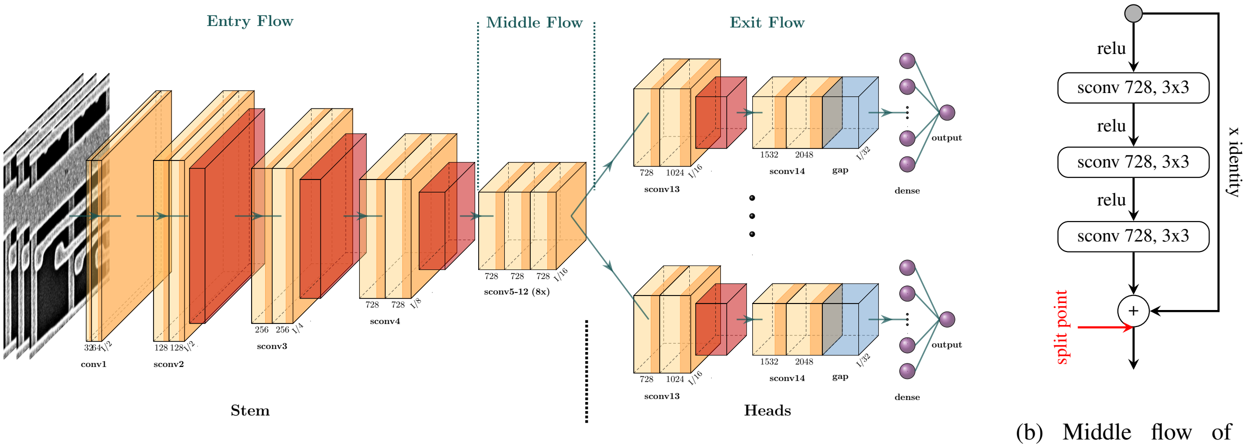
Figure 4: Performance comparison of CNN architectures sorted by their number of parameters. With increasing number of parameters, the training complexity grows.

Table 2: Overview of the evaluated split points.

Middle flow block	Split layer name	layer index	backbone parameters	head parameters	head ratio
sconv_10	add_8	95	10 845 144	10 018 385	0.5
sconv_11	add_9	105	12 459 120	8 404 409	0.4
sconv_12	add_10	115	14 073 096	6 790 433	0.3

HydREnet Our proposed network architecture consists of two components (Fig. 5a): A shared *stem* through which all input images pass. The stem is tasked with extracting low-level features relevant to predicting all classes of defects. Individual *heads* serve as experts for one class at a time. This allows the heads to extract class-specific fine-grained features based on the feature map output from the stem. Unlike Mullapudi *et al.* [19], we do not use a gating mechanism to select the heads to be executed and we do not use a combiner that adds the outputs of each head to make the final prediction.
The 250 px × 250 px gray scale images are stacked three times in a row to be fed into the network as 3-channel input. We decided to set the split point to the individual heads after one of the eight times repeated feature extraction blocks of the middle flow (Fig. 5b). After the separable convolution, the input of the feature extraction block is added to the feature map by means of a skip connection. This feature map serves as output of the stem and is passed on as input to each head. The heads are implemented identically to each other according to the exit flow of the Xception network. The output layer consists of one output neuron per head, which predicts the occurrence of one error class. The size of the heads (Tab. 2) determine the training speed and the proportion of learned class-specific fine-grained features.

Defect localization To interpret and debug the prediction results of the network (Fig. 6) we applied class activation mapping (CAM) [36]. A global average pooling layer must be inserted after the last convolution layer for this technique.



(a) Each input image passes through the shared stem. At the exit of the stem the feature map is passed to each head. There is exactly one head per defect class. Skip connections have been omitted for clarity.

(b) Middle flow of the Xception network with marked hydra split point.

Figure 5: Representation of the proposed `HydREnet` architecture based on the chosen Xception backbone.

3.4. Training

Training of the `HydREnet` is separated into two phases: The initial training of the backbone and the retraining of each individual head. Our approach is based on reusing pretrained weights to thereby increase both training efficiency and prediction quality.

Stem First, the Xception backbone with four output neurons needs to be trained. Second, the stem is initialized with the associated weights of the trained backbone.
We started the backbone training by initializing the network with the weights of ImageNet [23] except for the last layer. ImageNet initialization outperformed random initialization by converging faster. The ImageNet variant with initially frozen layers did not improve the results.
We used Bayesian optimization in our search for the best hyperparameters (Tab. 3). Following Snoek *et al.* [26] the optimization is executed for 14 iterations with $\kappa = 10$ in order to strongly favor exploration over exploitation. We used sigmoid activation with a binary cross entropy loss. The backbone was trained for a total of 70 epochs, and the best model was determined according to the highest validation F_1-score. We did not use an early stopper because the small dataset caused fluctuations in validation loss.

Heads Each head is initialized with the associated weights of the trained backbone. The weights of the dense layer in the backbone are of the form $C \times N_{\text{classes}}$, where in our case $N_{\text{classes}} = 4$. Therefore, each of the four heads can reuse the weights connected with the corresponding output neuron. Thus, the weights of the dense layer of each head have a form of $C \times 1$.
The weights of the stem are frozen so that it recognizes patterns of all defect classes. Each head is further trained separately for 30 epochs with the same hyperparameters as before. Overall, each input image first passes through the stem to create the feature map, and then through the respective head, whose weights are updated by its own loss function.

Table 3: Search space and results of Bayesian hyperparameter optimization after 14 iterations and $\kappa = 10$.

Hyperparameter	Search space	Best
Learning rate	$[1e{-}5, 1e{-}3]$	$2.51e{-}4$
Batch size	8, 16, 32	16
Decay rate	[0.95, 1]	0.95
Decay steps	[1, 3]	2.682
Optimizer	RMSProp, Adam	Adam

Table 4: Evaluation of data sampling strategies for the backbone training. We applied 3-fold cross validation and executed the models on the test set. The averaged results are shown in the table. The training was executed on an NVIDIA Quadro P4000 GPU.

Sampling Strategy	Time per epoch [min]	Class Average (%)				
		P	R	mIoU	F_1	F_2
No sampling	13	**88.0**	78.4	70.8	82.2	77.2
Oversampling	57	83.0	84.3	**71.9**	**83.6**	83.5
Undersampling	7	72.4	86.2	64.9	78.4	82.2
Resampling (ours)	7	74.1	**86.6**	66.5	79.6	**83.9**

Data Sampling The dataset is used in two setups: The backbone training (multi-label classification) and the head training (binary classification). We analyzed four different sampling strategies in terms of performance and training time required (Tab. 4).
No sampling results to high precision (P) due to the predom-

inantly negative examples. Oversampling up to the majority class *None* works especially well for the strongly represented defect classes. Undersampling the majority class to the level of the largest minority class, while oversampling all other classes to that level strongly increases the recall (R). As final implemented strategy, we additionally resampled the majority class *None* samples every five epochs to include previously unused images. In direct comparison, the precision benefits the most, since more negative samples are considered. Overall, this approach has the advantage that we can perform a 70 epoch training in 8 hours and even achieve the highest average F_2-score.

For the binary classification, we adopted the procedure from the multi-label case by lowering the majority class to 50 % and raising the minority class—which represents the particular defect— to this level.

4. Experiments and Results

To demonstrate that the proposed `HydREnet` is capable of handling the real RE process, three datasets of SEM images were compiled. Each of the datasets comes from real IC analyses with typical requirements and deficiencies. Using these datasets, we now systematically test the generalization capability of the proposed `HydREnet`. For this purpose, we address the following problems: The training complexity of the network, the detection of unknown defects, and the extensibility of the network through retraining.

4.1. Training complexity comparison

The proposed `HydREnet` is evaluated in terms of performance and training complexity. We realize binary classification by training four single output neuron Xception networks, where each network predicts one error class. In the first variant, we initialize each network with the weights of ImageNet and train it from scratch for 70 epochs. In the second variant, we initialize the networks with the weights of the trained backbone model (transfer learning) and train it for only 30 epochs.

We realize multi-label classification by training a single four output neuron Xception network for 70 epochs. All hyperparameters are the same as defined in Sec. 3.4.

Dataset-40TT This dataset is used to train and test (TT) the model (Sec. 3.2). It contains layout images of the lower metal layers of four different 40 nm technologies. Thus, the test set is very similar to the training set.

Performance analysis Our proposed `HydREnet` performs similarly well to the four individual binary networks (Tab. 5). Prior initialization with pretrained weights (transfer learning) does not provide any added value.

The `HydREnet` architecture has been evaluated for different splits. The model with split at add_9 achieves the highest average F_1-score. This architecture performs best in predicting poorly represented classes. The CAMs prove the impact of fine-grained expert training in the head (Fig. 6).

Training complexity analysis The training time ratio (Tab. 5) is determined by multiplying the number of trainable parameters by the epochs to be trained. This sum is divided by the corresponding value of the `HydREnet`-add_9 split to obtain a ratio.

The use of the multi-label architecture has the advantage that only one network needs to be trained. However, it is very difficult to control the prediction quality for each class. The binary networks are scalable and allow for individual retraining, but have a very long training time.

4.2. Identifying unknown defects

We evaluate the performance of our trained `HydREnet`-add_9 model on the given dataset-150UD. We show that our model can also recognize new unknown defects. Unknown defects do not belong to the four defined error types. To identify these defects, we discuss the added value of a *Misc* class whose images are manually evaluated by experts. The goal is to collect the unknown errors in this *Misc* class so that the expert will be aware of the insufficiently scanned image areas in the manual review.

Dataset-150UD The imaged technology is fabricated in a 150 nm node. Consequently, the metal tracks are made of aluminum and the VIAs are made of tungsten. Tungsten has a very high electron density and requires different SEM settings than copper based technologies. In this scan, the SEM parameters were not chosen optimally, resulting in new unknown defects (UD). The dataset contains 1280 test images with new unknown defects, as well as *Particles* and *Underlying layer visible*.

Performance analysis The model detects 88.8 % of all *Particles* (Tab. 6). However, only 57.1 % of all visible defects of the *Underlying layer* are found. The low accuracy is mainly due to inconsistent labels. The model does not identify the defect, because the two visible layers have no overlapping area (Fig. 6). So it actually behaves correctly.

Only 33.3 % of the images classified as positive actually have an underlying visible layer. The high false positive rate results from the unknown defects (Fig. 7). These defects arise from a very high brightness setting and would lead to fatal errors in feature extraction with threshold-based segmentation. Overall, our model finds unknown defects reliably, albeit in an incorrectly assigned class. To control this behavior, we introduce a *Misc* class.

Table 5: Evaluation of the different defect classification architectures in terms of performance and the training time required. The number of trainable parameters of the `HydREnet`-add_9 split serves as the base value for ratio calculation.

Architecture	Trainable Parameters		Training time	F$_1$-Score (%) (averaged 3-fold cross validation)				
	n	ratio	ratio	Particles	Underlying	Overlying	Smeared	Class Average
Binary network	83 236 004	1.5	2.4	**87.5**	**90.5**	**82.4**	83.3	85.9
Binary network – transfer learning	104 051 152	1.9	1.6	86.2	89.4	81.1	76.7	83.6
Multi-label network	20 815 148	0.4	0.6	86.5	89.2	80.5	83.8	85.0
`HydREnet`-add_8 split (ours)	47 976 880	0.9	0.9	87.0	90.0	82.0	82.2	85.3
`HydREnet`-add_9 split (ours)	54 432 784	1.0	1.0	87.1	90.1	82.2	**85.7**	**86.3**
`HydREnet`-add_10 split (ours)	60 888 688	1.1	1.1	87.2	90.2	82.1	82.4	85.5
Difference of `HydREnet`-add_9 split to best performing model				-0.4	-0.4	-0.2	0.0	0.0

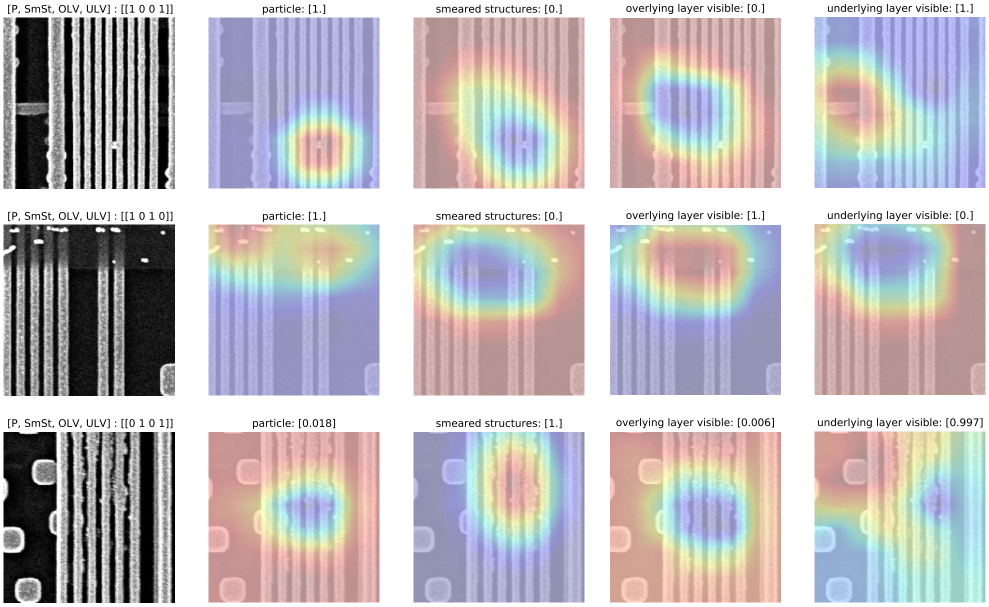

[P, SmSt, OLV, ULV] : [[1 0 0 1]]
particle: [1.]
smeared structures: [0.]
overlying layer visible: [0.]
underlying layer visible: [1.]
[P, SmSt, OLV, ULV] : [[1 0 1 0]]
particle: [1.]
smeared structures: [0.]
overlying layer visible: [1.]
underlying layer visible: [0.]
[P, SmSt, OLV, ULV] : [[0 1 0 1]]
particle: [0.018]
smeared structures: [1.]
overlying layer visible: [0.006]
underlying layer visible: [0.997]

Figure 6: Results of the `HydREnet`-add_9 split architecture: CAMs with confidence score for three different images with two defects each. The red areas indicate the image regions that were crucial for the decision making of the model.

Table 6: Performance evaluation for *Misc* classes of different sizes based on the specified confidence interval.

Defect class	Confidence interval	Manual inspec. (%)	Scores (%)		
			P	R	F$_1$
Particle	[0.1, 0.9]	5.5	89.4	89.4	**89.4**
	[0.2, 0.8]	3.4	86.9	87.8	87.3
	[0.3, 0.7]	2.1	84.3	88.3	86.3
	[0.4, 0.6]	0.5	80.5	88.4	84.3
	none	0.0	79.8	88.8	84.1
Underlying	[0.1, 0.9]	4.7	71.4	62.5	66.6
	[0.2, 0.8]	3.8	70.0	70.0	**70.0**
	[0.3, 0.7]	2.1	50.0	58.3	53.9
	[0.4, 0.6]	0.9	38.1	57.1	45.7
	none	0.0	33.3	57.1	42.1

Misc class and manual effort Low confidence predictions are defined by a sigmoidal activation output close to 0.5, which corresponds to a random decision in the given binary classifications. The *Misc* class is used to capture low confidence predictions and present them to the human expert for manual review. This process improves the accuracy of defect recognition and contribute to the discovery of new types of defects. We evaluate various confidence intervals, expressed as sigmoid activation thresholds that would then fall into the *Misc* class (Tab. 6). In addition, we report the resulting percentage of samples to be manually inspected, with scores referring to the percentage of samples that do not fall into the *Misc* class. By transferring the images within the confidence interval of [0.1, 0.9], the manual effort is reduced to about 5 % of the total images. The remaining incorrect predictions for the images that are not manually checked result mainly from the aforementioned labeling inconsistencies.

4.3. Extensibility through retraining

We first evaluate the performance of our `HydREnet`-add_9 model on the dataset-40SD. With this dataset we

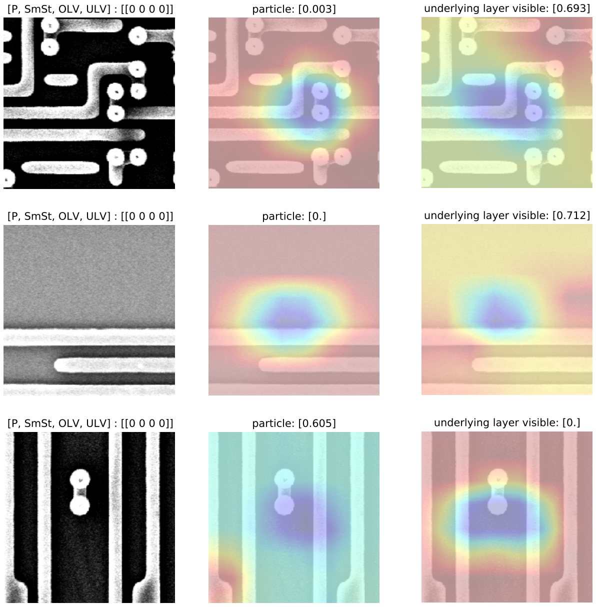

[P, SmSt, OLV, ULV] : [[0 0 0 0]]
particle: [0.003]
underlying layer visible: [0.693]
[P, SmSt, OLV, ULV] : [[0 0 0 0]]
particle: [0.]
underlying layer visible: [0.712]
[P, SmSt, OLV, ULV] : [[0 0 0 0]]
particle: [0.605]
underlying layer visible: [0.]

Figure 7: Examples of unknown defects from dataset-150UD found by our model.

Table 7: Evaluation results of the inital `HydREnet`-add_9 model prediction and the prediction of the new model after further training of the heads with the dataset-40SD.

Dataset	Defect class	Initial (%)			Retrained (%)		
		P	R	F_1	P	R	F_1
40SD	Particle	91.9	92.1	**92.0**	87.6	93.7	90.6
	Underlying	94.9	95.3	95.1	98.5	95.7	**97.1**
	Overlying	97.4	53.6	69.2	87.2	98.6	**92.5**
	Smeared	00.0	00.0	00.0	71.7	89.2	**79.5**
40TT	Particle	85.3	89.2	**87.3**	84.4	90.2	87.2
	Underlying	86.5	94.5	90.3	88.2	93.7	**90.8**
	Overlying	79.4	88.6	**83.8**	79.0	86.3	82.5
	Smeared	80.0	87.0	83.3	77.8	91.3	**84.0**

would like to provoke the failure of the model. We perform a targeted retraining of the heads. Each `HydREnet` head is trained for 10 epochs and the best head is stored according to the best F_1 score of the validation. Finally, we discuss the costs and benefits of our approach.

Dataset-40SD The imaged sample is fabricated in a 40 nm technology node. The images show serious defects (SD) beyond anything the model has seen in training. The dataset contains 1126 training and 282 test images showing all four defect classes.

Performance analysis The initial model performs well on the classes strongly represented in the training. As expected, it fails on the two absolute minority classes (Tab. 7). Briefly continuing to train with the additional new training data solves this problem satisfactorily for the class *Overlying layer visible*. The *Smeared structures* class would need more representatives during training to achieve higher accuracy. The newly trained model also performs well on the original test dataset of 40TT. This indicates that no overfitting to the newly introduced data has occurred.

Cost-benefit-analysis The retraining of the head for the *Particles* class took 134 minutes with a NVIDIA Quadro P4000 GPU. All four heads can be updated within one working day or night. If multiple GPUs were deployed in parallel, training would be complete after a few hours. This short training duration enables iterative, fast and flexible work. Based on new, manually collected data, the model can be easily extended.
Another advantage is the situational use of individual heads. Based on the results obtained, it is reasonable to use the retrained heads for predicting the images of dataset-40SD. For new analyses, reusing the initial heads for classification of *Particles* and *Overlying layer visible* is probably more valuable. The expert can variably handle the differently trained heads. In the long run, this approach prevents a data drift.

5. Conclusion

In this work, a defect inspection framework is proposed that enables automated defect classification in SEM layout images of delayered ICs. The framework serves as a feedback tool for the human experts who plan and execute the RE process. The task of image classification is performed by the proposed `HydREnet`. Each input image first passes through the shared stem to create the feature map, and then through the respective head for the final defect classification. Our approach is based on reusing the pretrained weights of the Xception backbone to increase both training efficiency and prediction quality.
Our network outperforms comparable multi-label and binary networks in terms of training complexity while maintaining high accuracy. Especially the poorly represented classes benefit from the hydra-like network composition. Unknown defects are predicted with a lower confidence level. Therefore, these defects can be collected in a *Misc* class and reviewed manually. Network extensibility can be realized by retraining the heads with new data within a few hours. Subsequently, the expert can variably use and adapt the differently trained heads for future analyses.

Acknowledgement

AI4DI receives funding within the Electronic Components and Systems for European Leadership Joint Undertaking (ECSEL JU) in collaboration with the European Union's Horizon2020 Framework Programme and National Authorities, under grant agreement n° 826060.

References

[1] Ulbert J Botero, Ronald Wilson, Hangwei Lu, Mir Tanjidur Rahman, Mukhil A Mallaiyan, Fatemeh Ganji, Navid Asadizanjani, Mark M Tehranipoor, Damon L Woodard, and Domenic Forte. Hardware Trust and Assurance through Reverse Engineering: A Tutorial and Outlook from Image Analysis and Machine Learning Perspectives. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 17(4):1–53, 2021.

[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. SMOTE: synthetic minority oversampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[3] Sejune Cheon, Hankang Lee, Chang Ouk Kim, and Seok Hyung Lee. Convolutional Neural Network for Wafer Surface Defect Classification and the Detection of Unknown Defect Class. *IEEE Transactions on Semiconductor Manufacturing*, 32(2):163–170, 2019.

[4] Franck Courbon. Practical Partial Hardware Reverse Engineering Analysis. *Journal of Hardware and Systems Security*, 4(1):1–10, 2020.

[5] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[6] Ying Gao, Jiqiang Lin, Jie Xie, and Zhaolong Ning. A Real-Time Defect Detection Method for Digital Signal Processing of Industrial Inspection Applications. *IEEE Transactions on Industrial Informatics*, 17(5):3450–3459, 2021.

[7] Minoru Harada, Yohei Minekawa, and Koji Nakamae. Defect detection techniques robust to process variation in semiconductor inspection. *Measurement Science and Technology*, 30(3), 2019.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition Kaiming. In *Recognition, Proceedings of the IEEE conference on computer vision and pattern*, pages 770–778, 2016.

[9] Xuenong Hong, Deruo Cheng, Yiqiong Shi, Tong Lin, and Bah Hwee Gwee. Deep Learning for Automatic IC Image Analysis. *International Conference on Digital Signal Processing, DSP*, 2018-Novem, 2019.

[10] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[11] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely Connected Convolutional Networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[12] Kazunori Imoto, Tomohiro Nakai, Tsukasa Ike, Kosuke Haruki, and Yoshiyuki Sato. A CNN-Based transfer learning method for defect classification in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, 32(4):455–459, 2019.

[13] K. Song and Y. Yan. Northeastern University (NEU) surface defect database.

[14] Adam Kimura, Jon Scholl, James Schaffranek, Matthew Sutter, Andrew Elliott, Mike Strizich, and Glen David Via. A Decomposition Workflow for Integrated Circuit Verification and Validation. *Journal of Hardware and Systems Security*, 4(1):34–43, 2020.

[15] Tong Lin, Yiqiong Shi, Na Shu, Deruo Cheng, Xuenong Hong, Jingsi Song, and Bah Hwee Gwee. Deep learning-based image analysis framework for hardware assurance of digital integrated circuits. *Microelectronics Reliability*, 123, 2021.

[16] Bernhard Lippmann, Niklas Unverricht, Aayush Singla, Matthias Ludwig, Michael Werner, Peter Egger, Anja Duebotzky, Helmut Graeb, Horst Gieser, Martin Rasche, and Oliver Kellermann. Verification of physical designs using an integrated reverse engineering flow for nanoscale technologies. *Integration*, 71:11–29, 2020.

[17] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[18] Hangwei Lu, Ronald Wilson, Nidish Vashistha, Navid Asadizanjani, Mark Tehranipoor, and Damon L Woodard. Knowledge-based object localization in scanning electron microscopy images for hardware assurance. In *Conference Proceedings from the International Symposium for Testing and Failure Analysis*, volume 2020-Novem, pages 20–28, 2020.

[19] Ravi Teja Mullapudi, William R. Mark, Noam Shazeer, and Kayvon Fatahalian. HydraNets: Specialized Dynamic Architectures for Efficient Inference. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8080–8089, 2018.

[20] Shahed E Quadir, Junlin Chen, Domenic Forte, Navid Asadizanjani, Sina Shahbazmohamadi, Lei Wang, John Chandy, and Mark Tehranipoor. A survey on chip to system reverse engineering. *ACM journal on emerging technologies in computing systems (JETC)*, 13(1), 2016.

[21] Raul Quijada, Roger Dura, Jofre Pallares, Xavier Formatje, Salvador Hidalgo, and Francisco Serra-Graells. Large-Area Automated Layout Extraction Methodology for Full-IC Reverse Engineering. *Journal of Hardware and Systems Security*, 2(4):322–332, 2018.

[22] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9351:234–241, 2015.

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, and Alexander C Berg. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, pages 211–252, 2015.

[24] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. SinGAN: Learning a generative model from a single natural image. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:4569–4579, 2019.

[25] K. S. Sim, V. Teh, and M. E. Nia. Adaptive noise Wiener filter for scanning electron microscope imaging system. *Scanning*, 38(2):148–163, 2016.

[26] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in neural information processing systems*, 25:1–9, 2012.

[27] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:10691–10700, 2019.

[28] Sanli Tang, Fan He, Xiaolin Huang, and Jie Yang. Online PCB Defect Detector On A New PCB Defect Dataset. *arXiv preprint arXiv:1902.06197*, 2019.

[29] M. Tanjidur Rahman, Qihang Shi, Shahin Tajik, Haoting Shen, Damon L. Woodard, Mark Tehranipoor, and Navid Asadizanjani. Physical inspection attacks: New frontier in hardware security. *2018 IEEE 3rd International Verification and Security Workshop, IVSW 2018*, pages 93–102, 2018.

[30] Randy Torrance and Dick James. The State-of-the-art in Semiconductor Reverse Engineering. In *Proceedings of the 48th Design Automation Conference*, DAC '11, pages 333–338, New York, NY, USA, 2011. ACM.

[31] Grigorios Tsoumakas and Ioannis Katakis. Multi-Label Classification. *Database Technologies*, pages 309–319, 2011.

[32] Matthias Wieler and Tobias Hahn. Weakly Supervised Learning for Industrial Optical Inspection. In *DAGM symposium in*, 2007.

[33] Ronald Wilson, Navid Asadizanjani, Domenic Forte, and Damon L Woodard. Histogram-based Auto Segmentation: A Novel Approach to Segmenting Integrated Circuit Structures from SEM Images. *arXiv preprint arXiv:2004.13874*, 2020.

[34] Ronald Wilson, Domenic Forte, Navid Asadizanjani, and Damon L Woodard. LASRE: A novel approach to large area accelerated segmentation for reverse engineering on sem images. In *Conference Proceedings from the International Symposium for Testing and Failure Analysis*, volume 2020-Novem, pages 180–187, 2020.

[35] Xiaoqing Zheng, Hongcheng Wang, Jie Chen, Yaguang Kong, and Song Zheng. A Generic Semi-Supervised Deep Learning-Based Approach for Automated Surface Inspection. *IEEE Access*, 8:114088–114099, 2020.

[36] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning Deep Features for Discriminative Localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.