AI-Driven Anomaly Detection in Oscilloscope Images for Post-Silicon Validation

Kowshic A. Akash*, Tobias Wulf*, Torsten Valentin*, Alexander Geist*, Ulf Kulau†, and Sohan Lal†
*NXP Semiconductors, Hamburg, Germany
Email: {kowshicahmed.akash, tobias.wulf, torsten.valentin, alexander.geist}@nxp.com
†Hamburg University of Technology, Germany
Email: {ulf.kulau, sohan.lal}@tuhh.de

***Abstract*—In the *post-silicon validation* process, different functionalities and boundaries of a *system-on-chip* (SoC) are tested, generating a large amount of data in the form of oscilloscope images, trace data, and log files. Oscilloscope images are used to visualize and analyze the digital I/O signals and play a crucial role in detecting anomalies. However, the debugging process of the oscilloscope images requires a lot of manual data analysis, which is time-consuming, inefficient, costly, and prone to errors. This paper proposes an artificial intelligence (AI) model to automatically detect anomalies in the oscilloscope images. Our proposed model uses a *Convolutional Autoencoder* (CAE), a neural network, which we train on real silicon data obtained from various post-silicon validation projects. While autoencoders have been used for anomaly detection, this is the first use to detect anomalies in oscilloscope images for post-silicon validation. Moreover, the state-of-the-art techniques use *Reconstruction Error (RCE)* as an anomaly detection metric, we show that a combination of *RCE* and *Kernel Density Estimation (KDE)* error metrics greatly reduces the false negatives (68%) for the anomalous category and improves the *recall* metric from 62% to 88%, making our approach 41% better. In addition, our proposed model achieves 99% *precision* in categorizing not anomalous data points. Furthermore, the proposed model has been deployed in the production environment, significantly reducing human effort.**

***Index Terms*—Post-Silicon Validation, Artificial Intelligence, Anomaly Detection, Convolutional Autoencoders**

I. Introduction

With the rapid development of digital technology, especially the emergence of smart and efficient hand-held devices such as smartphones and tablets, the demand for various SoC architectures has surged over the past decade. The emergence of large language models (LLMs) and generative AI is transforming our lives in numerous ways and further driving the demand for SoCs, both for accelerators and hand-held devices. Consequently, there is an immense competition among vendors to bring new SoC products first in the market, putting pressure on reducing time to market metric. The *post-silicon validation* process ensures that SoCs shipped to customers function correctly [1]. It is a complex process but one of the key elements of the modern SoC design life-cycle.

Unfortunately, the validation process generates a large amount of data in the form of oscilloscope images, trace data, and log files. Figure 1 provides an overview of the amount of oscilloscope images generated for various real post-silicon validation projects. The figure shows that millions of

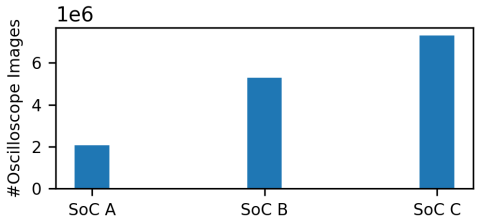


Fig. 1: The amount of oscilloscope images generated from various real SoC post-silicon validation projects.

oscilloscope images are generated during SoCs validation. Manually analyzing and debugging such a large amount of data after each test run is difficult, time-consuming, and error-prone. Moreover, each test is usually repeated multiple times, generating an even higher amount of data, which increases the possibility of missing critical bugs, and anomalies.

This paper proposes an AI powered method to automatically detect anomalies in the oscilloscope images, which provides several benefits such as reducing overall validation time, errors, and time to market. We evaluate two CAE models tailored for anomaly detection in oscilloscope images. The CAE exploits spatial relationships between data points to extract features that have a visual interpretation, separating normal oscilloscope images from anomalous ones. A key advantage of our models is that we train them on real post-silicon validation data obtained from various SoC projects. While autoencoders have been used for anomaly detection in other domains [2]–[5], this is the first work, to the best of our knowledge, that uses autoencoders to detect anomalies in oscilloscope images for the post-silicon validation.

We first examine the model's performance with the RCE metric for different threshold values. While state-of-the-art CAE algorithms use RCE as an anomaly detection metric [6], [7], we show that using RCE alone leads to more false negatives and/or positives. To address this issue, we propose KDE as an additional error metric. The combination of metrics significantly reduces the false negatives (68%) for the anomalous category and improves the recall metric from 62% to 88%, making our approach 41% better. Also, the precision increases from 97% to 99%. Our methodology automates the detection of anomalous oscilloscope images, greatly reducing the human effort for manual debugging.

In summary, we make the following main contributions:

- We propose a novel AI-driven methodology using CAE to automatically detect anomalies in the oscilloscope images. While autoencoders have been used for the anomaly detection, this is the first use to detect anomalies in oscilloscope images for the post-silicon validation.
- We train two CAE models (Model A and Model B) on post-silicon validation data obtained from various SoC projects. The models A and B achieve 98% and 99% precision, respectively, for the non anomalous category.
- While the state-of-the-art CAE algorithms use RCE as an anomaly detection metric, we propose to combine the RCE error metric with KDE metric. The combination reduces false negatives by 68% for the anomalous category and improves the recall metric from 62% to 88%.
- The proposed models have been tested and deployed into production environment, significantly alleviating the human effort for the post-silicon validation of SoCs.

This paper is organized as follows. In Section II-C we present our anomaly detection method. Section III presents the experimental setup. Section IV presents results. Section V presents related work and we draw conclusions in Section VI.

II. Anomaly Detection in Oscilloscope Images

A. Overview of Post-silicon Debugging Process

The post-silicon validation and debugging environment, at a very high-level, consists of a *Validation Application* to run tests on a *DuT (Device Under Test)*, a database where all the results are stored, and a *Test Owner* who implements and debugs the results. Figure 2 shows the result-data relations in the database. Here *Test Run* is an instance of a particular test run containing several parameter *Sweeps* that finally generates the result files. The result files contain 1) *oscilloscope images* recording I/O signals, 2) *trace data files* containing all *sweeps* and measurement data during the execution with successful and failed results for a test run, 3) *log files* with complete log of all activities performed before, during, and after the test run, and 4) other test specific data. A test run is the top-level representation of a test case, aggregating results from multiple parameter variations, such as voltage and frequency sweeps.

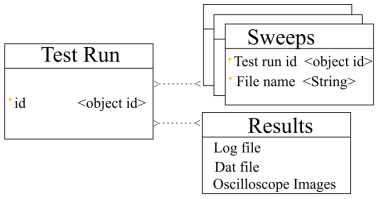


Fig. 2: Structure of test results in a database.

B. Overview of Oscilloscope Images and Possible Anomalies

Oscilloscope images are used to visualize and analyze the electrical waveforms such as voltage and current while running tests on an SoC. Figure 3 shows example oscilloscope images generated for two different test cases: case A (left) and case B (right). These images are generated synthetically [1], however, the underlying concept remains the same.

[1]We could not show real anomalous data due to the confidential agreement.

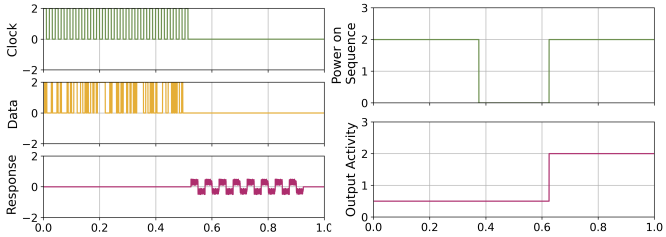


Fig. 3: Oscilloscope images for case A (left) and B (right).

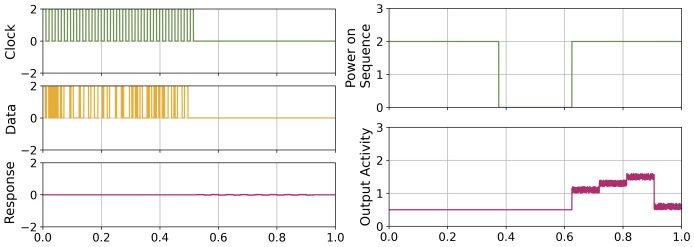
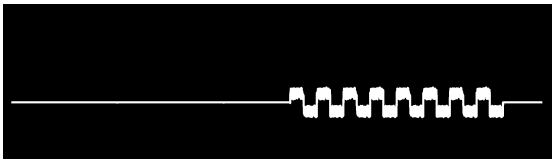


Fig. 4: Example of anomalies in case A (left) and B (right).



(a) Channel segmentation

(b) Whitespace removal

Fig. 5: Preprocessing of oscilloscope images.

The case 'A' tests the communication of the DuT. In this example, the green channel captures the input clock, the yellow channel captures the data sent to the device, and the red channel denotes the current profile of the activity after the data is received. The case 'B' tests the waking-up of the device from a low-power or standby mode. In this example, the green channel represents the power on sequence and the red channel represents the activity of the device after it switches on.

Different kinds of anomalies such as noise, spike, drift, and distortion can occur in a signal [8]. Figure 4 shows example anomalies occurring in test case A (left) and B (right). For the test case A, if the data sent to the device is not received properly, the output response will not follow the normal pattern. The anomalous output pattern means that the communication is not working properly. For the test case B, the output response deviates from the normal pattern indicating that the device did not wake up properly.

C. Data Preparation and Preprocessing

We pre-process data before training the AI models. We briefly describe the pre-processing steps below:

- Resizing: As the oscilloscope images had different shapes and background colors, we re-size images to the same size and background color for better training.
- Thresholding: We employ thresholding to segment and enhance image clarity, selecting a specific intensity value

as the threshold. In our color-based research, we automatically apply thresholding after segmentation.

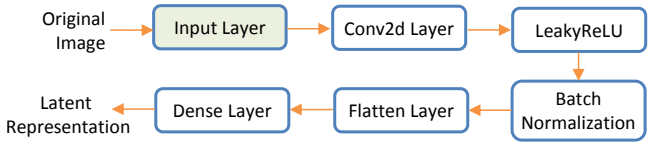
- Channel Segmentation: We employ color-based channel segmentation to separate channel data using color space. Figure 5a shows the output, where the oscilloscope image response is segmented from the original image. This operation can also be applied to other output channels.
- Whitespace Removal: We remove empty space by finding pixel values greater than zero after the channel segmentation as shown in Figure 5b.
- Binary Labeling of Data: Our models only require binary labeling of the data and no further classification of the anomalous behavior. Therefore, we label oscilloscope images as anomalous (1) or not anomalous (0) on the passing/failing criteria of the corresponding test sweep.
- Scaling and Split: Finally, we scale images to make the pixel values between 0-1 for faster training and better error convergence [9]. We split the scaled data into 3 datasets: training, validation, and test datasets for model's training and evaluation (Section III-B).

D. Convolutional Autoencoders for Anomaly Detection

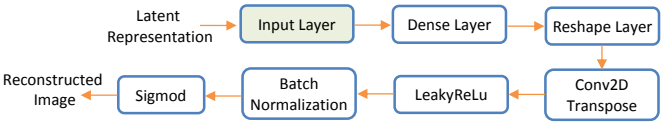
Anomaly detection in image data involves detecting patterns that deviate from the normal behavior. Anomalies can occur in many forms such as rare events or outliers and creating a comprehensive labeled dataset that covers all possible anomalies is very impractical. *Autoencoders* (AE) and specifically CAE for images offer a solution by training a model on normal data without explicit labels for different anomalies. An AE represents the input into a compressed form and then reconstruct it back as similar as possible to the input [10].

An AE reconstructs the input data points after passing them through a compression phase and a decompression phase. The combination of the layers in the compression/decompression phase is referred as an encoder/decoder. The main difference between traditional AE and CAE is that the latter is focused on using spatial relationships between points to extract features that have a visual interpretation. The spatial convolution operations in the intermediate layers achieve this goal [11].

Figure 6a and Figure 6b show examples of typical convolutional encoder and decoder segments of a CAE model, respectively. The encoder part comprises of different layers such as, 2D convolution, pooling, activation function (e.g., Leaky ReLu), and normalization, which extract essential features of the input images. The convolution layer uses a filter called kernel that is responsible for creating the output shape. The final dense layer encodes the input image in the latent space. The decoder attempts to reconstruct the original images from the encoded features. The input layer takes the latent representation from the encoder output and the final output layer gives the reconstructed image. The encoder output in the middle of the network has a lower dimension than the input and output layer forcing the network to learn a compressed representation of the input data.



(a) An encoder model for encoding normal images.



(b) A decoder model for decoding normal images.

Fig. 6: Encoder and decoder stages of an autoencoder.

E. Anomaly Detection Mechanism

An AE learns to reconstruct the input oscilloscope images during the training. When the trained model encounters anomalous images during the inference, it struggles to reconstruct the images properly because anomalous images often deviate significantly from the learned normal patterns. The RCE, which is calculated on the original input image and the reconstructed image, can be used to classify normal vs. anomalous images.

During the AE training, we assess RCE on the training data to set an error threshold. Although this approach performs well on benchmark datasets, it has a few shortcomings. First, the approach is based on the assumption that the training data consists only of normal images. In practice, clean dataset cannot always be guaranteed due to the possibility of errors occurring during data labeling. Therefore, a small number of anomalies contaminating the training might result in an AE to learn to reconstruct anomalous data.

To make the anomaly detection more robust, we combine RCE and the likelihood of an image in latent space [2], which we observe using KDE. When KDE is applied to latent space representations at the encoder output, it identifies the probability distribution of the data points by placing a smooth continuous function such as *Gaussian* and summing up these kernels to create a smooth estimate of the probability density. The anomalous data points in the density distribution plot will be sparse and have a higher standard deviation compared to normal data points. The decision whether data points are anomalous or not is made by identifying a suitable threshold for the RCE and KDE metrics.

F. Key Metrics for Evaluating Model Performance

- Precision: It represents the proportion of predicted positives that are truly positive. In Equation 1, True Positive refers to the samples for which the actual and predicted labels are both 1. False positive refers to the samples for which the true label is 0, but the predicted label is 1.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (1)$$

- Recall: It represents the proportion of actual positives that are correctly classified as defined in Equation 2.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (2)$$

- F1-Score: Classifier tuning often involves a trade-off between precision and recall. The F1-score combines both into a single metric, as shown in Equation 3.

$$F1_{score} = \frac{2(Precision * Recall)}{Precision + Recall} \quad (3)$$

Our goal is to minimize false negatives to catch critical anomalies and ensure reliable SoCs, while keeping false positives low to reduce human effort.

III. Experimental Setup

A. Model Architecture

Table I shows the simplified architectures of the two CAE models (model A and model B) that we evaluate. While model A uses pooling layers, which reduce the input feature size, and the default Relu activation function, model B does not use pooling layers and uses LeakyRelu activation function that solves dead neuron problems. The encoder and decoder parts of model A and B consists of 12 and 14 layers, respectively.

TABLE I: Simplified architectures of Model A and Model B.

Model A	Model B
InputLayer (Encoder Input)	InputLayer (Encoder Input)
Conv2D	Conv2D
MaxPooling2D	LeakyRelu
BatchNormalization	BatchNormalization
Flatten	Flatten
Dense (Encoder Output)	Dense (Encoder Output)
InputLayer (Decoder Input)	InputLayer (Decoder Input)
Dense	Dense
Reshape	Reshape
Conv2D	Conv2DTranspose
Upsampling2D	LeakyRelu
BatchNormalization	BatchNormalization
Conv2D	Conv2DTranspose
Sigmoid (Decoder Output)	Sigmoid (Decoder Output)

B. Dataset and Training

Table II shows the training, test, and validation datasets. We label data using the binary classification as described in Section II-C. We train our models on 8,044 oscilloscope images from various post-silicon validation projects which are considered *normal* after the labeling. We divide the test dataset into two subsets: one containing normal images (2,682) and the other containing anomalous images (338). The training took about 15 hours for 50 epochs on Intel core i5 with an integrated GPU. We train and evaluate our models on the same SoC architecture, but we believe they can be applied to other SoCs for similar test cases, such as communication/wake-up validations. Given the large number of images to be validated, retraining the model for better results is feasible even on a desktop machine. We plan to assess model portability in future work. Additionally, assuming an average time of 20 seconds to debug one image, it takes 695 days of a test engineer to debug 1 million images, showing necessity for the automation.

Figure 7 shows the change in the training and validation loss when model A and B are trained. While the accuracy of the models increase with the number of epochs, there is no overfitting as the validation and training accuracy converges after the training. Moreover, we use callback for early stopping

TABLE II: Datasets used for the experiments.

Dataset	No. of Samples	Image Shape	Label
Training dataset	8,044	80 * 320	Not anomalous
Validation dataset	2,682	80 * 320	Not anomalous
Test dataset	2,682	80 * 320	Not anomalous
Test dataset	338	80 * 320	Anomalous

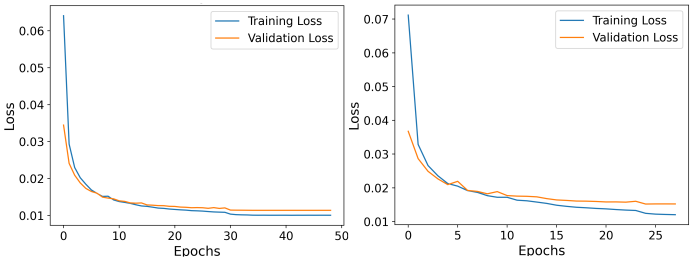


Fig. 7: Training and validation loss of model A (left) and model B (right).

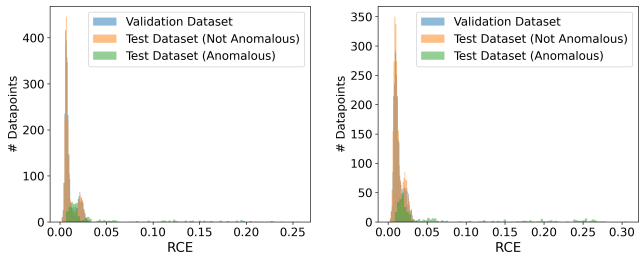
of training to save computational resources and reduce the overfitting as exhibited by Model B which stops at 28 epochs. Table III shows hyper-parameters used for the compilation.

TABLE III: Compilation parameters for the CAE algorithm.

Parameter	Value
Callbacks	[Early Stopping, Reduce Learning Rate]
Optimizer	Adam (learning rate=1e-4, weight decay=1e-5)
Loss/metrics	Mean Squared Error

IV. Results

We first present anomaly detection results using different RCE thresholds and then the combination of RCE and KDE.



(a) RCE histogram for model A. (b) RCE histogram for model B.

Fig. 8: Histogram showing RCE for models A and B.

A. Reconstruction Error

We evaluate CAE models on validation and test (anomalous and not anomalous) datasets. As we train the models on normal images, they will be very good at predicting/recreating the validation and not anomalous test datasets. Therefore, the RCE will be low for these two datasets. In case of validation dataset, the models monitor and evaluate the losses of the validation dataset, however, the weights and biases of the models are not updated based on this evaluation. As soon as the models start to overfit during training, the validation loss will start to increase while the training loss will decrease. In addition, models have never seen the validation, and test datasets. Therefore, if the training was successful, the RCE

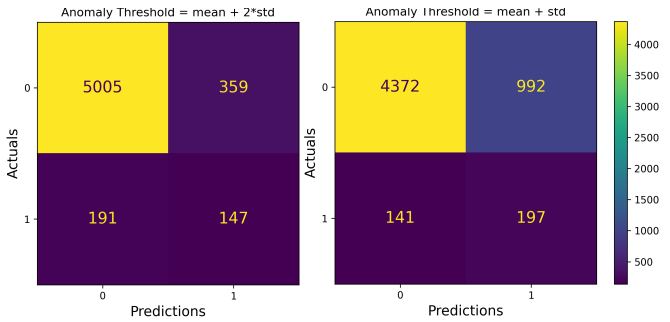


Fig. 9: Classification results by using RCE thresholds for model A. 1 represents an anomaly and 0 no anomaly.

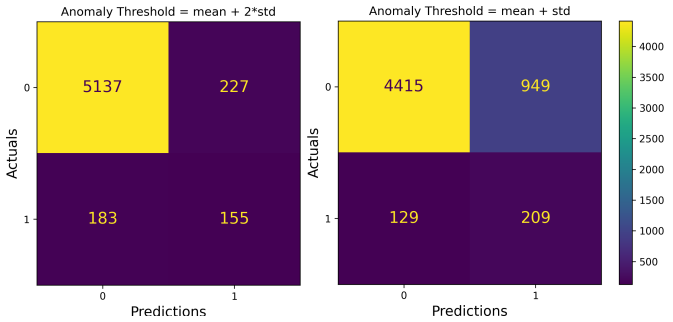


Fig. 10: Classification results by using RCE thresholds for model B. 1 represents an anomaly and 0 no anomaly.

will be high for the anomalous test dataset and low for the not anomalous test and validation datasets.

Figure 8a and Figure 8b show the RCE distribution of validation and test datasets for models A and B. Even though the RCE values are higher for the anomalous dataset, there is an overlapping region between the two classes. While it is possible to choose an RCE threshold to separate anomalous and not anomalous images, the accuracy of the predictions will be lower because of the overlapping.

B. Evaluation using Reconstruction Error Threshold

We evaluate the models on two RCE thresholds shown in Equation 4 and Equation 5 using validation and test datasets together. If a data point has an RCE greater than the threshold, the data point is considered as anomalous. We first experiment with a higher RCE threshold (Equation 4). Lowering the threshold will capture more anomalous points (Figure 8a and Figure 8b), however, this will also categorize some normal data points as anomalies due to the overlapping regions.

$$RCE_{Threshold} = mean_{RCETest} + 2 * std_{RCETest} \quad (4)$$

$$RCE_{Threshold} = mean_{RCETest} + std_{RCETest} \quad (5)$$

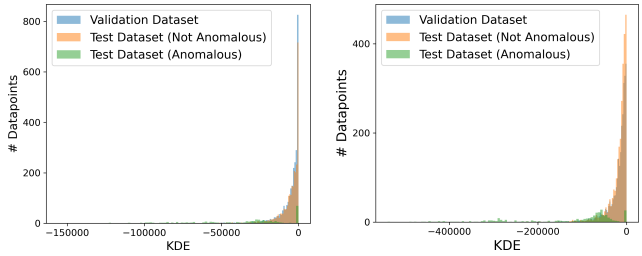
TABLE IV: Performance metrics of model A with RCE.

RCE Threshold	Decision	Precision	Recall	F1-score
mean + 2 * std	Not Anomaly	0.96	0.93	0.95
	Anomaly	0.29	0.43	0.35
mean + 1 * std	Not Anomaly	0.97	0.82	0.89
	Anomaly	0.17	0.58	0.26

Figure 9 and Figure 10 show the confusion matrices of models A and B. While the number of false negatives decreases from 191 to 141 in Figure 9 and 183 to 129 in Figure 10 with the decrease in threshold, the number of false positives increases from 359 to 992 in Figure 9 and from 227 to 949 in Figure 10. As it is more important to detect as many anomalies as possible in the post-silicon validation, even if that results in categorizing some normal data as anomalies. In this regard, the model B provides better results than model A.

TABLE V: Performance metrics of model B with RCE.

RCE Threshold	Decision	Precision	Recall	F1-score
mean + 2 * std	Not Anomaly	0.97	0.96	0.96
	Anomaly	0.41	0.46	0.43
mean + 1 * std	Not Anomaly	0.97	0.82	0.89
	Anomaly	0.18	0.62	0.28



(a) KDE histogram of model A. (b) KDE histogram of model B.

Fig. 11: Histogram illustrating KDE for models A and B.

Table IV and Table V show the change in different performance metrics by changing the RCE threshold for models A and B. The recall of the anomaly dataset increases with the decrease in the RCE threshold. However, this negatively impacts the other performance metrics leading to a trade-off between precision and recall scores. The recall metric also improves for model B (46% to 62%) compared to model A (43% to 58%), reducing the number of false negatives.

The results show that lowering the RCE threshold increases anomaly detection (higher recall, fewer false negatives) but reduces precision (more false positives), indicating RCE alone is insufficient. To mitigate this, we propose using KDE as an additional metric.

C. Kernel Density Estimate Error Metric

Figure 11a and Figure 11b show the KDE distributions for models A and B. We see a clear separation of the *anomalous* and *not anomalous* instances compared to Figure 8a and Figure 8b. Using a combination of RCE and KDE error metrics can improve the anomaly detection mechanism. The images with an RCE value greater than the RCE threshold or a KDE value less than the KDE threshold are considered as anomalous. We keep the higher RCE threshold as shown in Equation 4 and combine it with the KDE threshold as shown in Equation 6. We aim to improve the results we obtain by decreasing the RCE threshold in Table IV and Table V.

$$KDE_{Threshold} = mean_{KDETest} - std_{KDETest} \quad (6)$$

D. Evaluation using Combined Error Metrics

Figure 12 shows the confusion matrix using combined metric for models A and B. Compared to using only the RCE

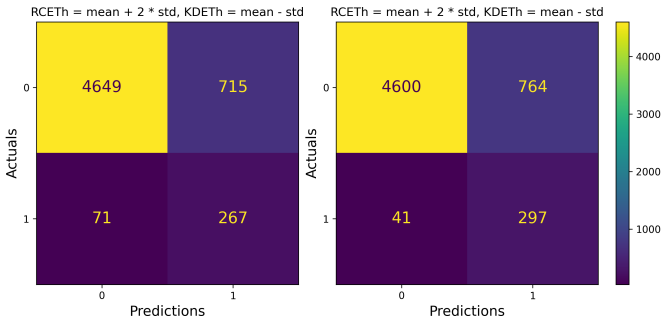


Fig. 12: Evaluation of combined metrics for model A (left) and model B (right).

TABLE VI: Performance of model A with RCE and KDE.

Decision	Precision	Recall	F1-score
Not Anomaly	0.98	0.87	0.92
Anomaly	0.27	0.79	0.40

TABLE VII: Performance of model B with RCE and KDE.

Decision	Precision	Recall	F1-score
Not Anomaly	0.99	0.86	0.92
Anomaly	0.28	0.88	0.42

metric, Figure 12 shows significant improvements with only 71 false negatives for model A (left) and 41 false negatives for model B (right). The combination greatly reduces the number of false negatives (68%) and improves the recall metric from 62% to 88%, making this approach 41% better.

Table VI and Table VII show that the recall metric for the anomalous category increased significantly (from 43% to 79% for model A and from 46% to 88% for model B) compared to Table IV and Table V. Moreover, the precision improves from 96% to 98% and 97% to 99% for model A and model B, respectively. The results can be further improved by fine-tuning both error metrics.

of our knowledge. Additionally, combining RCE with KDE reduces false negatives by 68% and improves recall to 88%.

V. Related Work

Autoencoders have been used for the anomaly detection in various domains [2]–[5]. For example, Beggel et al. [2] use adversarial autoencoders to detect anomalies in handwritten digits from MNIST and Fashion-MNIST. Chong et al. [3] propose a spatiotemporal architecture for anomaly detection in videos, using autoencoders to find representative features. [8] uses a deep transfer learning model, trained on a small set of images of voltage waveforms, to automatically detect power quality disturbances to ensure reliable electrical supply. Some other use cases of machine learning include fault-diagnosis at a circuit level [12], detecting anomalies in data logs to identify the approximate cycle of a bug's occurrence [13], and application of K-means clustering to group rare trace segments during post-silicon debugging [14]. Nassif et al. [15] provides a systematic overview of machine learning techniques to detect anomalies in data in general. Unlike previous work, we use CAE to detect anomalies in oscilloscope images of real post-silicon validation projects, which is the first work to the best

VI. Conclusions

Post-silicon validation is a complex and critical phase in the semiconductor manufacturing process where the functionality and performance of chips are validated, generating a large amount of debug data such as oscilloscope images. Oscilloscope images are used to visualize and analyze the digital I/O signals and play a crucial role in detecting anomalies. However, with the growing complexity of SoCs and pressure to reduce time to market, the manual debugging of huge amount of post-silicon validation data is inefficient and error-prone. Therefore, we propose a novel AI-driven methodology based on CAE to automatically detect anomalies in the oscilloscope images. We train two models on real post-silicon validation data. For the not anomalous category, the two models achieve 98% and 99% precision, respectively. While the state-of-the-art use RCE as an anomaly detection metric, it has shortcomings due to simplistic assumptions. To make anomaly detection more robust, we propose to combine the RCE error metric with the KDE metric. The combination reduces false negatives by 68% for the anomalous category and improves the recall metric from 62% to 88%. The models attain recall scores of 79% and 88% for the anomalous category. The proposed models are being used in production for post-silicon SoC validation, significantly reducing human effort. Our work highlights AI's potential to automate complex processes, enabling faster and more reliable SoCs.

References

[1] P. Mishra, R. Morad, A. Ziv, and S. Ray, "Post-silicon Validation in the SoC Era: A Tutorial Introduction," *IEEE Design & Test*, 2017.

[2] L. Beggel, M. Pfeiffer, and B. Bischl, "Robust Anomaly Detection in Images Using Adversarial Autoencoders," in *Machine Learning and Knowledge Discovery in Databases: ECML*, 2020.

[3] Y. S. Chong and Y. H. Tay, "Abnormal Event Detection in Videos Using Spatiotemporal Autoencoder," in *ISNN*, 2017.

[4] C. Zhou and R. C. Paffenroth, "Anomaly Detection with Robust Deep Autoencoders," in *KDD*, 2017.

[5] D. Thakur, S. Biswas, E. S. Ho, and S. Chattopadhyay, "Convae-lstm: Convolutional Autoencoder Long Short-term Memory Network for Smartphone-based Human Activity Recognition," *IEEE Access*, 2022.

[6] F. Angiulli, F. Fassetti, and L. Ferragina, "Reconstruction Error-based Anomaly Detection with Few Outlying Examples," *arXiv*, 2023.

[7] H. Torabi, S. L. Mirtaheri, and S. Greco, "Practical Autoencoder Based Anomaly Detection by Using Vector Reconstruction Error," *Cybersecurity*, vol. 6, no. 1, p. 1, 2023.

[8] G. Todeschini, K. Kheta, and C. Giannetti, "An Image-based Deep Transfer Learning Approach to Classify Power Quality Disturbances," *Electric Power Systems Research*, vol. 213, p. 108795, 2022.

[9] D. Cheng, Y. Fan, S. Fang, M. Wang, and H. Liu, "ResNet-AE for Radar Signal Anomaly Detection," *Sensors*, vol. 22, no. 16, 2022.

[10] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv*, 2020.

[11] C. C. Aggarwal *et al.*, "Neural Networks and Deep Learning," *Springer*, vol. 10, no. 978, p. 3, 2018.

[12] H. Amrouch, K. Chakrabarty, D. PflÃijger, I. Polian, M. Sauer, and M. S. Reorda, "Machine Learning for Test, Diagnosis, Post-Silicon Validation and Yield Optimization," in *ETS*, 2022.

[13] D. et al., "Machine Learning-Based Anomaly Detection for Post-Silicon Bug Diagnosis," in *DATE*, 2013.

[14] M. et al., "Application of Machine Learning Techniques in Post-Silicon Debugging and Bug Localization," *JET*, 2018.

[15] N. et al., "Machine Learning for Anomaly Detection: A Systematic Review," *IEEE Access*, 2021.