IC SEM Reverse Engineering Tutorial using Artificial Intelligence

Olivia P. Dizon-Paradis*, *Member, IEEE,* David S. Koblah*, *Member, IEEE,* Ronald Wilson, *Member, IEEE,* Domenic Forte, *Senior Member, IEEE,* Damon L. Woodard, *Senior Member, IEEE*
* These two authors contributed equally to this work

***Abstract*—In this tutorial, we will leverage artificial intelligence (AI) techniques to facilitate the reverse engineering of an integrated circuit based on scanning electron microspcopy, contributing to hardware assurance. This work encompasses various subjects, including image processing, computer vision, and machine learning, providing a comprehensive learning experience in these specialized domains. We encourage you to actively participate in this tutorial to execute your own project. The skills and knowledge acquired in this activity can bolster your resume, enhancing your prospects when seeking employment, pursuing further education, or advancing in your current professional role.**

***Index Terms*—artificial intelligence, integrated circuits, reverse engineering, scanning electron microscopy, segmentation, tutorial**

I. Introduction

Reverse engineering (RE) has emerged as a vital method for understanding the intricate structural and functional capabilities of integrated circuits (ICs). This process is of significant interest to both malicious and legitimate stakeholders within the electronics industry. The ultimate goal of RE is to create a product that replicates the utility and functionality of the original IC. Given the complexity of modern ICs, which are designed to meet the computing needs of advanced systems, the RE process requires as much useful information as possible. This necessitates extensive technological support, with a particular emphasis on imaging methods. Scanning electron microscopy (SEM) is one such method, offering macroscale resolution values and a wide range of magnification. The development of artificial intelligence (AI) over the past few decades has further enhanced the RE process, making it more automated and less tedious. Our work provides a blueprint for academic, industrial, and government stakeholders to effectively combine artificial intelligence and scanning electron microscopy for efficient IC reverse engineering.

While the relevance of reverse engineering to stakeholders in the electronics industry has grown over the years, the complexity of ICs has increased. Nonetheless, the resulting product must maintain the functionality and robustness of the original IC. To facilitate this, the process demands comprehensive information to replicate intellectual property with high-precision accuracy. While the applications of RE vary, establishing a functional system for IC analysis is crucial for any stakeholder involved in the electronic design automation (EDA) process. Within the realm of legacy electronic systems, RE addresses the issue of unavailable blueprints for older systems. Despite being used for decades, these systems may require updates or replacements, which can be challenging due to the laborious task of finding exact replicas or the compatibility issues that may arise from using upgraded alternatives. Consequently, obsolescence becomes an imminent threat for stakeholders. Hardware security and assurance is another critical aspect of managing legacy replacements and obsolescence, presenting its own set of challenges. RE helps in identifying vulnerabilities that can be exploited by attackers. It is also essential in distinguishing genuine hardware from counterfeit products, which is crucial for securing the electronics supply chain.

By leveraging AI and SEM, stakeholders can navigate these challenges more effectively, ensuring that the reverse engineering process is both thorough and efficient. This approach not only preserves the functionality of legacy systems but also enhances the security and longevity of modern electronic devices. In today's semiconductor-focused market, having a robust and diverse skill set is crucial. The framework presented in this article is designed to bolster the reader's skillset by providing hands-on experience in the above-mentioned fields within computer science and engineering. All coding scripts referred to throughout this tutorial will be made publicly available in open platforms for easy access, along with accompanying video tutorials[1]. We hope that both sets of information will facilitate development and collaboration among all contributors to the electronics industry.

This tutorial article is organized as follows: Section II provides a brief background on the topics merged into this tutorial. Section III details the framework for IC SEM reverse engineering using AI. Section IV presents the results and discussion. Section V suggests potential extensions. Finally, Section VI concludes the tutorial with key takeaways and provides guidance for future work.

II. Background

The main purpose of IC reverse engineering is to compare two designs and ensure that there is no infringement on intellectual property. The RE process delves into the intricate structure and function of these designs, revealing nanometer-scale features that have evolved over decades of development.

This article and artifacts were produced by the Florida Institute for National Security (FINS), by members of the Electrical and Computer Engineering (ECE) Department at the University of Florida in Gainesville, FL, USA.
Manuscript received Aug 26, 2024; revised Dec 15, 2024.

[1]https://github.com/olivia-dizon-paradis/ic_sem_re_tutorial

While RE is the goal, it is important to emphasize the additional concepts that facilitate the complete framework presented in this tutorial.

A. Artificial Intelligence

Artificial Intelligence (AI), the overarching area depicted in Figure 1, refers to the classification of systems designed to replicate human intelligence and cognitive functions, including problem-solving and learning [1]. Utilizing models for tasks such as prediction, classification, and generation, AI is often employed to enhance and address intricate problems traditionally performed by humans, such as facial and speech recognition, decision-making, and translation. Over the years, the availability of large amounts of data and computational resources has led to a rise in the use of AI in a variety of fields. Although AI and machine learning (ML) are commonly used interchangeably, it is important to understand their distinctions. Therefore, we highlight these differences below, and introduce other sub-fields relevant to this work.

1) Machine Learning: ML is a subset of AI that involves algorithms that automatically learn patterns from data. There are different types of machine learning, e.g., 1) *supervised learning* learns directly from labeled examples, 2) *unsupervised learning* learns in the absence of labeled examples, and 3) *reinforcement learning* learns by trial-and-error. There are other types of ML such as *semi-supervised learning*, which combines supervised and unsupervised learning, but for this tutorial, we are mostly focused on supervised and unsupervised learning. For more information, please refer to [1].

2) Image Processing and Computer Vision: Image processing (ImP) and computer vision (CV) are closely-related fields that both involve image data. There is significant overlap between the two fields, but the differences, however subtle, still hold. While ImP merely transforms an image from one type into another, CV attempts to interpret or recognize patterns in an image. In short, *ImP is image in, image out*, while *CV is image in, knowledge out*. For instance, the task of blurring an image is categorized under ImP, whereas image segmentation is classified as a CV task, given that segmentation involves the classification of foreground and background elements. As CV entails the recognition and interpretation of patterns within visual data, it is widely regarded as a subset of AI, whereas some aspects of ImP may be considered AI while others may not be. Moreover, both CV and ImP may overlap with machine learning in some instances. A thorough explanation of image processing, computer vision, and their scope within the broader context of artificial intelligence is detailed in [2].

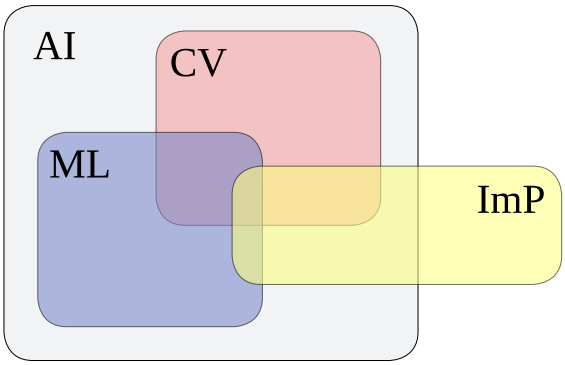


Fig. 1. Artificial Intelligence (AI) with related fields of machine learning (ML), computer vision (CV) and image processing (ImP)

B. Scanning Electron Microscopy

Although there are a variety of imaging modalities that play a crucial role in uncovering the inner workings of these designs, this work primarily focuses on Scanning Electron Microscopy (SEM). The SEM process is initiated with the emission of a beam of electrons from an electronic gun. These electrons traverse electromagnetic fields and lenses that focus the beam onto the sample. Upon impacting the sample, both electrons and X-rays are emitted. Detectors then collect the emitted X-rays and back-scattered electrons, converting them into visible signals. These signals are transmitted to a screen, resulting in the generation of the final detailed image [3]. Although destructive, SEM is renowned for its ability to generate high-resolution three-dimensional images on a nanometer scale, rendering more fine details than X-ray. Moreover, unlike light used in optical microscopy, the beam of electronns in SEM can simultaneously focus on features at different heights, facilitating a thorough assessment of differences in distances between objects and providing a significant advantage with its large depth of field.

The integration of AI has revolutionized IC reverse engineering by automating key aspects. This includes the identification of desirable characteristics that would typically require manual efforts from a subject matter expert. Thus, the intersection between AI and SEM-based IC RE enhances the overall efficiency of the existing RE framework. This tutorial is structured to provide a practical example application of the key aspects of the typical SEM-based IC RE using AI. The combined concepts segue into the comprehensive workflow described in the next section.

III. Methodology & Experimental Set-Up

Given an input IC SEM image (e.g. 130 nm smart card shown in Fig. 2a.), the reverse engineering process seeks to output a binary image with the regions of interest (foreground) distinctly segmented from the rest of the retrieved image (background), similar to the ideal ground truth template (e.g. Fig. 2b). The ground truth image was manually labeled by a subject matter expert, demanding a considerable amount of time, effort, and resources. On an industry scale where multiple intricate ICs must be analyzed quickly in practice, such manual segmentation is slow, expensive, and impractical for humans. To address these challenges, the integration of AI to help automate the segmentation process is described in this tutorial.

A. Workflow and Setup

The IC SEM segmentation pipeline for this tutorial is shown in Fig. 3. First, features are extracted from the input IC SEM

image using image processing (ImP) and computer vision (CV). The core task, segmentation, is then executed using machine learning (ML). The results are then fine-tuned in the post-processing stage using ImP. Finally, the output is evaluated to assess the method's suitability for deployment. If the result is unsatisfactory, the previous steps are re-assessed and revisited as needed.

Here, we use the following Python packages for their accessibility and ease of use: Numpy [5], Scikit-image [6], and Scikit-learn [7]. The experiments were conducted on a 32.0 GB Intel(R) Xeon(R) W-1250P CPU with a 64-bit Windows operating system. To ensure a controlled reproducible computational environment, the default Anaconda base environment with Python version 3.10.13 was used.

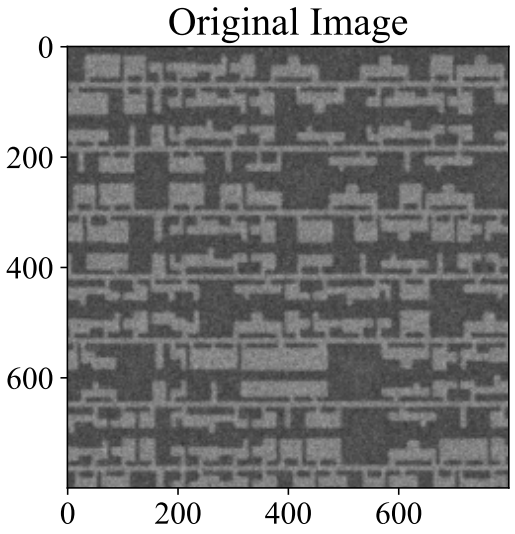


Fig. 2. Sample data from a 130 nm smart card. (a) Input uint8 grayscale image, with values in range [0, 255]; (b) Ground truth boolean image, with the doped silicon defined as foreground (marked in white, True, or 1) and the undoped silicon defined as the background (marked in black, False, or 0). Both images are 800x800 pixels.

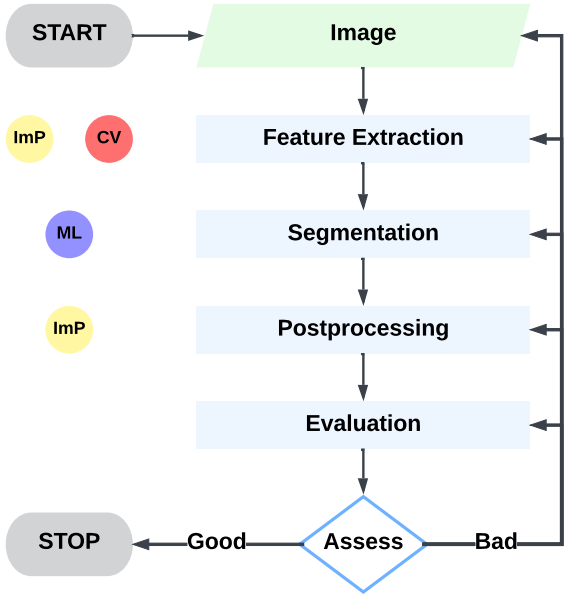


Fig. 3. IC SEM RE framework. An input image is taken and features are extracted using ImP and CV, segmented using ML, then post-processed using ImP. The final result is then evaluated.

B. Feature Extraction

Features extracted from the input image included pixel intensity values, edges, and corners. Pixel intensity values, which represent the color information at each pixel, provide the fundamental data upon which other features are detected. To reduce the effects of noise present in the raw image, Gaussian blurring was used. Edges, or significant local changes in intensity, often correspond to the boundaries of objects within an image. Corners, defined by the intersection of two or more edges, are particularly useful for identifying key points and features that remain invariant under various transformations. Although other image features exist, such as blobs and circles, our focus is directed toward pixel intensity values, edges, and corners because the overwhelming majority of our input image data consists of straight lines and rectangles and very few, if any, other shapes.

To mitigate overfitting, we employed an 80:20 train-test split without shuffling to preserve the visual interpretability of the training and testing data as distinct, non-overlapping images. In other words, the top 80% of the sample image was used as training data for model development and parameter tuning, whereas the bottom 20% of the sample image was used as testing data for performance evaluation. Although this tutorial primarily concerns a single sample image to illustrate the overall workflow, a larger dataset with multiple images would ideally be used in practice. Implementation details are shown in Alg. 1, and examples of the extracted features are shown in Fig. 4.

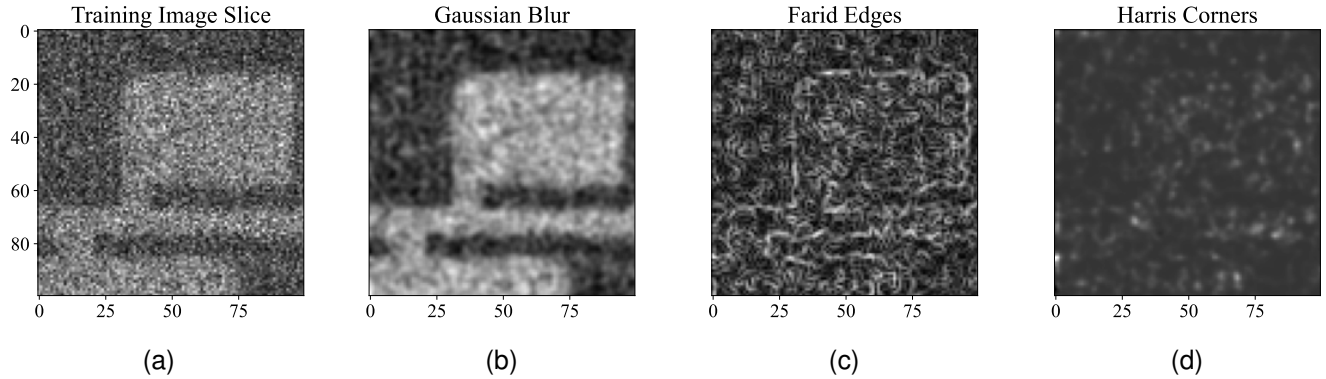


Fig. 4. Image slice of the (a) original IC SEM sample with corresponding (b) Gaussian blur, (c) Farid edge, and (d) Harris corner features.

C. Segmentation

Here, three different ML techniques were explored: KMeans clustering, Decision Tree classification, and Random Forest classification. KMeans clustering is an unsupervised technique that iteratively groups data based on inter-cluster variance [8]. As an unsupervised ML technique, it does not require ground truth information for training. Decision Tree classification is a supervised technique that operates by using hierarchical decision rules (such as greater than or less than) to classify data [9]. As a supervised ML technique, it requires some ground truth information for training. Random Forest classification, another supervised technique, is a meta-estimator that uses an ensemble of decision trees to improve generalization when classifying data [10]. This method leverages collective wisdom, where each tree in the forest is trained on a random subset of the data, and classifications are made by averaging the results. An important parameter for the random forest algorithm is the number of estimators, i.e. "trees in the forest." The final implementation details are shown in Alg. 2.

Algorithm 1 Feature Extraction and Train-Test Split

Require: original image `img`, ground truth image `gt`, test percentage `p_test=0.20`

Import necessary libraries:

```
from skimage import filters, feature
import numpy as np
from sklearn import model_selection
```

Get de-noised intensity features using Gaussian blur:

```
f1 = filters.gaussian(img)
```

Get edge features using Farid filter:

```
f2 = filters.farid(img)
```

Get corner features using Harris corner detection:

```
f3 = feature.corner_harris(img)
```

Stack features together and format:

```
feats = np.dstack((f1, f2, f3))
feats = feats.reshape(800*800, -1)
```

Split the sample data into training and testing images:

```
n_test = int(p_test*800)
feats_train, feats_test, gt_train, gt_test =
model_selection.train_test_split(feats, gt,
test_size = p_test, shuffle=False)
```

return `feats_train`, `feats_test`, `gt_train`, `gt_test`, `n_test`

Algorithm 2 Segmentation

Require: training features `feats_train`, ground truth `gt_train`, testing features `feats_test`, number of test rows `n_test`

Import the necessary libraries, define, and train model:

Alg. 2.a. KMeans Clustering

```
from sklearn import cluster
model = cluster.KMeans(n_clusters=2)
model.fit(feats_train)
```

Alg. 2.b. Decision Tree Classification

```
from sklearn import tree
model = tree.DecisionTreeClassifier(
min_impurity_decrease=0.01)
model.fit(feats_train, gt_train)
```

Alg. 2.c. Random Forest Classification

```
from sklearn import ensemble
model = ensemble.RandomForestClassifier(
n_estimators=10, min_impurity_decrease=0.01)
model.fit(feats_train, gt_train)
```

Predict the segmented image from the testing data:

```
seg_img = model.predict(feats_test)
seg_img = seg_img.reshape(n_test, 800)
```

return `seg_img`

D. Post-processing

Segmentation results were refined using idempotent binary morphological operations: opening and closing. Opening and closing operations build off erosion and dilation operations, which reduce and increase shape sizes, respectively. Opening, or erosion then dilation, was used to remove background noise and separate shapes that were barely connected. Closing, or dilation then erosion, was used to remove foreground noise and connect shapes that were barely disconnected. The execution of morphological operations requires two inputs: the primary image and a structuring element, which serves as a probe for defining the neighborhood around each pixel. The implementation is shown in Alg. 3.

E. Evaluation

The final postprocessed segmentation results were quantitatively compared against the ground truth using the popular

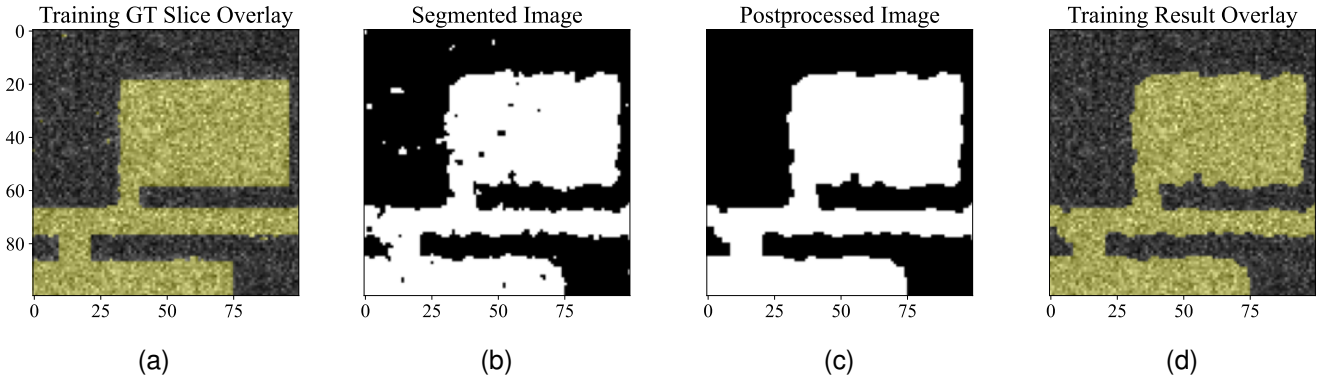


Fig. 5. Image plots of: (a) Ground truth slice overlay, (b) Segmented image, (c) Postprocessed image, and (d) Result overlay for a sample training image

Algorithm 3 Postprocessing

Require: Segmented image `seg_img`

```
Import necessary libraries:
from skimage.morphology import
binary_opening, binary_closing
Define the structuring element:
se = np.ones([3, 3])
Postprocess using binary opening and closing:
post_img = binary_opening(seg_img, se)
post_img = binary_closing(post_img,se)
return post_img
```

intersection-over-union (IOU) score:

$$IOU(S,G) = \frac{|S \cap G|}{|S \cup G|}, \tag{1}$$

where S is the final postprocessed segmentation result, G is the ground truth, and $|S \cap G|$ and $|S \cup G|$ are the area of intersection and the area of union, respectively, of S and G. Due to how the score is defined, the IOU score ranges between 0 and 1, with 1 being the best and 0 being the worst. To assess the stability and robustness of the implemented approaches, each algorithm was trained 30 times with random initialization, and the average IOU scores are reported.

For comparison, results were also obtained for thresholding, a simple segmentation approach using image processing. After a basic gridsearch testing threshold values [50,200] in multiples of ten, the threshold value of 100 was found to yield the highest IOU. To test the effectiveness of the pre- and postprocessing pipelines, results were obtained for: (1)thresholding with no processing and (2) thresholding plus processing.

In addition to quantitative results, qualitative results are crucial for visualizing and interpreting patterns, relationships, and insights. This visualization process is essential for understanding the underlying structures and behaviors that quantitative metrics alone may not fully reveal. By carefully examining intermediate images, such as those depicted in Fig. 5, we can gain a deeper understanding of the model's performance and the specific features it has learned. Through expert observation, we can more effectively communicate the nuances and complexities of the data, facilitating a more comprehensive analysis and enhancing the overall robustness of our findings.

IV. Results & Discussion

The IOU scores of each method on the training and testing data are summarized in Tab. 1. For the stochastic ML methods, the mean IOU values are reported along with their corresponding standard deviations. Overall, the training and testing results were similar, suggesting that the models generalized well to the unseen data. In this tutorial, the Decision Tree classifier performed best. To gain deeper insight into the decision-making process of the ML models, we conducted an interpretation analysis.

A. KMeans Clustering

Although KMeans had the lowest average IOU scores, the standard deviation of the scores was very large. A quick analysis using `numpy.histogram` indicates the IOU scores followed a bimodal distribution, with one peak around 0.023 and the other around 0.882. This observation is consistent with expectation as KMeans is an unsupervised method with an equal chance of clustering foreground as foreground (which is correct) or background (inverted). After flipping the outputs of the inverted models, an analysis of the cluster center locations (found using `sklearn.cluster.KMean`'s `cluster_centers_` attribute) indicated that the normalized foreground center for the Gaussian blur, edge, and corner features, respectively, was located at [0.50146887 0.03532551 0.06544869], whereas the background center was at [0.31454059 0.0279947 0.02885523]. This indicates that the foreground pixels generally had lighter intensity, more edges, and more corners than the background pixels, and that the Gaussian blur features were more important factors for the KMeans models. Although inverted model outputs could be flipped in this two-class classification tutorial, it can be difficult to address in multi-class problems.

TABLE I
Average Training and Testing IOU Scores for Each Method

Method	Training	Testing
Thresholding	0.540	0.579
Thresholding + Processing	0.870	0.864
KMeans Clustering	0.395 ± 4.25E-1	0.401 ± 4.28E-1
Decision Tree Class.	**0.881 ± 2.22E-16**	**0.889 ± 1.11E-16**
Random Forest Class.	0.848 ± 8.41E-2	0.856 ± 8.46E-2

B. Decision Tree Classification

In this tutorial, Decision Tree classification yielded the best IOU scores, with very little variation in performance, indicating model stability and robustness. DT models can be interpreted using `sklearn.tree`'s `plot_tree` function (e.g. Fig. 6). The tree visualization helps in understanding how the model splits the data at each node based on different features to arrive at the final prediction. In this example, the color of each node corresponds to the predicted class (i.e. orange for background, blue for foreground). For internal nodes (i.e. ones with child nodes), lighter colors indicate the class estimated at this point whereas for external leaf nodes (i.e. ones without children), darker colors indicate the final class prediction. In each node, the "samples" percentages indicate the total percentage of the training inputs that reached each node, and should sum to 100% among nodes at similar levels (or roughly 100% due to rounding errors). The "value" arrays indicate the normalized percentage of samples reaching each node for each class and for each output, i.e. the $[.59, .41]$ value in the root node indicates that 59% of the training samples that reached this node were background, whereas 41% were foreground. Nodes are traversed to the left child node if true, and right if false. For example, starting at the topmost root node, if the $faridedges \leq 0.04$ statement is true, we navigate to the left child node. If the following $gaussianblur \leq 0.41$ statement is false, we navigate to the right child node and classify as foreground. For this particular tree, the if-else hierarchy can be simplified to **if** $gaussianblur \leq 0.41$ classify as background; **else** classify as foreground. Simplification may not be possible for other trees in the forest.

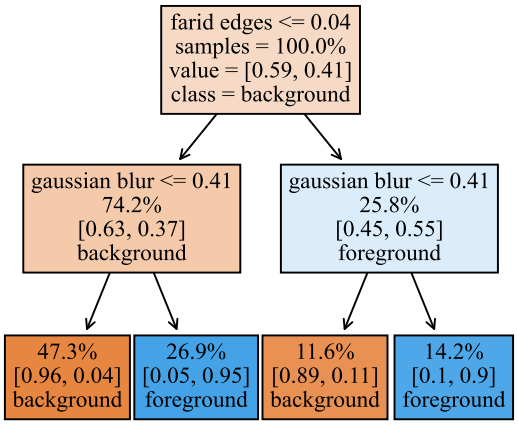


Fig. 6. Visualization of a Decision Tree's predictive process

C. Random Forest Classification

Here, Random Forest (RF) classification also yielded high IOU scores, but underperformed Decision Tree classification and Thresholding+Processing. RF, although intended to boost performance and increase robustness relative to individual decision trees, likely underperformed in this tutorial because there were few informative features available to sample when looking for the best split at each node. In other words, in scenarios where there are only a handful of critical features, since RF randomly samples subsets of features when building each tree, some trees in the forest likely failed to effectively utilize these critical features, resulting in performance degradation. RF's relative feature importances, shown in Fig. 7, can be extracted from `sklearn.ensemble.RandomForestClassifier`'s `feature_importances_` attribute. Feature importances results indicate that the gaussian blur features contributed most significantly to the model's decisions, consistent with the KMeans clustering results. In addition, individual trees from the RL model can be accessed by indexing into `sklearn.ensemble.RandomForestClassifier`'s *estimators_* attribute and visualized using the `plot_tree` function as discussed in the previous subsection.

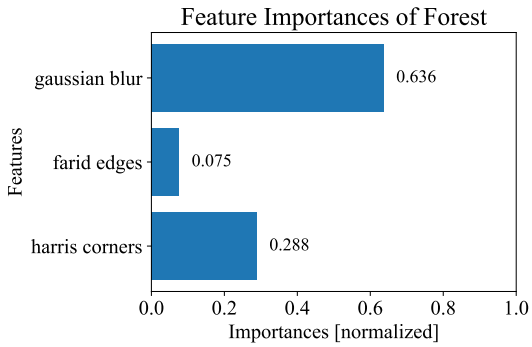


Fig. 7. Visualization of a Random Forest's relative feature importances

D. Discussion

The combination of quantitative results, visualizations, and model interpretation analyses provides a comprehensive understanding of the performance and behavior of the implemented models. These insights are crucial for validating the model's effectiveness, gaining valuable insights, and for guiding future improvements in the predictive algorithm.

Although Decision Tree Classification performed best on the sample image used in this tutorial paper, results may vary on different datasets with different technologies, imaging conditions, and other factors. This methodology presented in this paper is meant to illustrate the general framework, while being extensible to other workflows.

V. Extensions

To supplement the existing process, there are a number of extensions available to the interested reader:

1) For **feature extraction**, we used blurring, edge detection, and corner detection. Readers are encouraged to experiment with the parameters of these methods and explore alternative approaches, such as rectangle detection, leveraging the diverse array of filters and features available in the Scikit-image (skimage) library. For a comprehensive exploration of features and their efficacy, employing feature selection methodologies such as Linear Discriminant Analysis (LDA) could prove beneficial.

In instances where the selected features exhibit widely varying ranges, feature normalization techniques, such as standard scalar, becomes pertinent.

2) For **segmentation**, we used KMeans clustering, Decision Tree classification, and Random Forest classification. Similar to the feature extraction stage, parameter adjustment is encouraged (e.g. with grid search) or the exploration of alternative classifiers such as nearest neighbors, support vector machines, and naïve Bayes. Ensemble methods, including AdaBoost and gradient boosting, present additional avenues for consideration. It is essential to note that the interpretability and explainability methods may vary across different algorithms, necessitating a thoughtful exploration of approaches tailored to the chosen methods.
3) For **postprocessing**, we used binary opening and closing. Users have the flexibility to fine-tune parameters related to the structuring element, including its shape and size. Additionally, alternative morphological operations, such as hole filling, can be explored for potential enhancement.
4) For **evaluation**, we rely on a simple train/test split. However, an alternative and more comprehensive approach involves implementing **k-fold cross-validation**. This technique ensures a thorough assessment of the model's performance by partitioning the dataset into multiple folds, iteratively using different subsets for training and testing.

We highly encourage experimentation, urging the interested reader to observe the effects firsthand and consider exploring methods beyond those explicitly listed in this article. For additional inspiration and guidance, the following resources are recommended: Scikit-image (skimage) [6] and Scikit-learn (sklearn) [7]. These resources offer comprehensive insights and further methods to enhance the interested reader's exploration and understanding of the implemented techniques.

VI. Conclusion & Future Work

Scanning electron microscopy is useful for IC reverse engineering (RE) due to its ability to produce high-resolution images. The addition of artificial intelligence (AI) to the RE process automates crucial aspects, reducing the tedious manual efforts of subject matter experts. AI-enhanced reverse engineering enables quick and accurate recognition of trained patterns, making the overall process more efficient.

The authors hope this tutorial serves as a valuable resource for both teaching and learning the fundamentals of hardware assurance and artificial intelligence. The tutorial is intended to enhance the professional profile of the interested reader, as a valuable resume-building asset. Whether one is seeking employment, pursuing further education, or aiming to advance in one's current role, the skills acquired through this endeavor are poised to contribute to one's prospects.

Looking ahead, the authors are eager to share more knowledge through additional tutorials focused on the intersection between hardware security and AI. Future works cover a broad spectrum, including but not limited to generative AI for hardware security, AI-driven secure hardware design, and anti-RE hardware obfuscation.

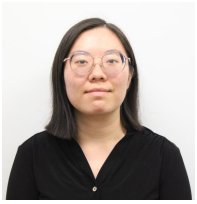
Acknowledgments

The authors would like to thank the following entities and individuals. The National Science Foundation (NSF) for funding this work, through Award # 2131480 SaTC: TTP: Medium: I-C-U: AI-Enabled Recovery and Assurance of Semiconductor IP from SEM Images. The U.S. Department of Defense (DoD) for funding the graduate students working on this tutorial. Mengdi Zhu, Daniel E. Capecci, Gijung Lee for their thorough review of the materials. Pallabi Ghosh, Rabin Acharya, and Alyssa Caples for their involvement on the project.

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[2] R. Gonzalez and R. Woods, *Digital Image Processing*, 4th edition. Pearson, 2017.
[3] A. Mohammed and A. Abdullah, "Scanning electron microscopy (SEM): A review," in *Proceedings of the 2018 International Conference on Hydraulics and Pneumatics—HERVEX, Băile Govora, Romania*, vol. 2018, pp. 7–9, 2018.
[4] "Anaconda — The World's Most Popular Data Science Platform," Anaconda. Accessed: Oct. 27, 2023. [Online]. Available: https://www.anaconda.com/
[5] "NumPy." Accessed: Oct. 27, 2023. [Online]. Available: https://numpy.org/.
[6] S. Van Der Walt et al., "scikit-image: image processing in Python," PeerJ, vol. 2, p. e453, Jun. 2014, doi: 10.7717/peerj.453.
[7] "scikit-learn: machine learning in Python — scikit-learn 1.3.2 documentation." Accessed: Oct. 27, 2023. [Online]. Available: https://scikit-learn.org/stable/
[8] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–08, 1979, doi: 10.2307/2346830.
[9] W.-Y. Loh, "Classification and regression trees," *WIREs Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 14–23, 2011, doi: 10.1002/widm.8.
[10] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.

VII. Biography Section



Olivia P. Dizon-Paradis, Ph.D. is a member of the Florida Institute for National Security (FINS). She is a member of IEEE, Eta Kappa Nu (HKN), and Tau Beta Pi (TBP). Her research interests include applied AI, reinforcement learning, computer vision, and reverse engineering.

David S. Koblah, Ph.D. is a member of the Florida Institute for National Security (FINS). He is an IEEE Graduate Student Member. He is a member of IEEE and Eta Kappa Nu (HKN). His research interests include machine learning and AI for PCB reverse engineering and Security-Aware EDA.

Ronald Wilson, Ph.D. is Co-Director of the Applied AI Group (AAIG), and Research Assistant Professor in ECE Department at the University of Florida. He is also a FINS member. His research interests include AI for vision-assisted hardware assurance on ICs, computational behavioral analytics for identity sciences, computer vision, and NLP.

Domenic Forte, Ph.D. is Associate Director of the Florida Institute for National Security (FINS), and Professor and Steven A. Yatauro Faculty Fellow in the ECE Department at the University of Florida. He is an Senior Member of both the IEEE and ACM. His research interests include multiple aspects of hardware security.

Damon L. Woodard, Ph.D. is Director of the Florida Institute for National Security (FINS), Director of the Applied Artificial Intelligence Group (AAIG), and Professor in the ECE Department at the University of Florida. He is an Senior Member of both the IEEE and ACM. His research interests include multiple aspects of AI.