Autoencoder-Based Data Sampling for Machine Learning-Based Lithography Hotspot Detection

Mohamed Tarek Ismail
mtarek.ismail@gmail.com
Siemens EDA
American University in Cairo
Cairo, Egypt

Hossam Sharara
hossam.sharara@aucegypt.edu
American University in Cairo
Cairo, Egypt

Kareem Madkour
kareem_madkour@mentor.com
Siemens EDA
Cairo, Egypt

Karim Seddik
kseddik@aucegypt.edu
American University in Cairo
Cairo, Egypt

ABSTRACT

Technology scaling has increased the complexity of integrated circuit design. It has also led to more challenges in the field of Design for Manufacturing (DFM). One of these challenges is lithography hotspot detection. Hotspots (HS) are design patterns that negatively affect the output yield. Identifying these patterns early in the design phase is crucial for high yield fabrication. Machine Learning-based (ML) hotspot detection techniques are promising since they have shown superior results to other methods such as pattern matching. Training ML models is a challenging task due three main reasons. First, industrial training designs contain millions of unique patterns. It is impractical to train models using this large number of patterns due to limited computational and memory resources. Second, the HS detection problem has an imbalanced nature; datasets typically have a limited number of HS and a large number of non-hotspots. Lastly, hotspot and non-hotspot patterns can have very similar geometries causing models to be susceptible to high false positive rates. Due to these reasons, the use of data sampling techniques is needed to choose the best representative dataset for training. In this paper, a dataset sampling technique based on autoencoders is introduced. The autoencoders are used to identify latent data features that can reconstruct the input patterns. These features are used to group the patterns using Density-based spatial clustering of applications with noise (DBSCAN). Then, the clustered patterns are sampled to reduce the training set size. Experiments on the ICCAD-2019 dataset show that the proposed data sampling approach can reduce the dataset size while maintaining the levels of recall and precision that were obtained using the full dataset.

CCS CONCEPTS

• **Hardware → Software tools for EDA**; • **Computing methodologies → Cluster analysis**.

MLCAD '22, September 12–13, 2022, Snowbird, UT, USA.

ACM ISBN 978-1-4503-9486-4/22/09…$15.00
https://doi.org/10.1145/3551901.3556479

KEYWORDS

autoencoder; clustering; DBSCAN; data reduction; design for manufacturing; hotspots; machine learning; sampling

ACM Reference Format:
Mohamed Tarek Ismail, Hossam Sharara, Kareem Madkour, and Karim Seddik. 2022. Autoencoder-Based Data Sampling for Machine Learning-Based Lithography Hotspot Detection. In *Proceedings of the 2022 ACM/IEEE Workshop on Machine Learning for CAD (MLCAD '22), September 12–13, 2022, Snowbird, UT, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3551901.3556479

1 INTRODUCTION

The continuous reduction in semiconductor device size has lead to a vast improvement in the functionality and performance of integrated circuits (ICs). However, it has also caused more challenges in the manufacturing process. Specifically, in the lithography step, yield can be affected by certain patterns that are sensitive to process variations. These patterns are commonly known as hotspots (HS). An example is illustrated in Figure 1 where the manufactured metal tracks have deviated from the drawn ones causing a short circuit to occur.

HS patterns must be detected early in the design stage to ensure that the IC can be fabricated with high yield. However, this process is computationally expensive since it requires lithography simulation using rigorous process models, which needs many hours of high-powered computational resources. As the complexity of integrated circuits continues increasing, it is becoming a real challenge to run simulation on a full chip. Therefore, new approaches other than full simulations were introduced in the literature to address this challenge. One of the recent promising methods in hotspot detection is the use of machine learning techniques to tackle the problem in an approximate, yet more efficient manner. These techniques rely on training a machine learning model using a set of known hotspots (HS) and non-hotspots (NHS) on existing designs. The trained model can then be used to detect hotspots on new designs, so that they can be rectified before fabrication.

The process of HS detection using machine learning techniques can be divided into three steps: data generation, model training, and prediction. In the first step, patterns are captured from the input layout and encoded into a set of features that can be used for training. Each feature vector is then labeled as either corresponding

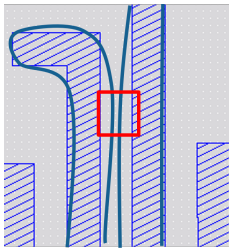

Figure 1: Example of a Hotspot Pattern

to a hotspot (HS) or a non-hotspot (NHS). In the training step, the model iteratively minimizes a pre-defined loss function that measures the deviation from the model predictions during training from the actual labels. The learned model (achieving the minimal loss), is then used to predict patterns on the test layouts to determine whether they correspond to a hotspot or not.

There are three main challenges in the data generation step. The first challenge is that industrial training layouts typically contain millions of unique patterns [10]. It is difficult to use this amount of data to train a ML model due to computational and memory limitations. For example, using the ICCAD-2019 dataset [15] which has around 15k patterns, it would take a few minutes to train a convolutional neural network such as Mobilenet [9]. However, an industrial dataset such as the one in [10] having 20M unique patterns would require several days for training to finish. Hence, data selection is needed to reduce the training set size. The second challenge is the imbalanced nature of the problem; in a typical design, the ratio of hotspots to non-hotspots is very small, which causes a class skew in the generated datasets for training. Machine learning models are sensitive to imbalance in class distribution, which necessitates the employment of a sampling technique to choose a representative dataset for training that is more balanced. The third challenge is the existence of numerous non-hotspot patterns that are structurally and geometrically similar to hotspot patterns, which makes them harder to predict by the machine learning model, and usually get classified as HS. These patterns are described as hard-to-classify (HTC) [15] and can lead to an increase in the false positive rate of the model.

While basic random sampling can be used for data reduction, it can cause many patterns such as the HTC patterns to be lost. Moreover, random sampling does not consider the data imbalance issue that is present in lithography hotspot datasets. Hence, a more sophisticated sampling method is needed to preserve the statistical properties in the original dataset. This method should be able to address the data imbalance issue and ensure that the sampled dataset contains the HTC patterns to help the model learn the subtle differences between HS and NHS.

Various ML methods for HS detection are discussed in the literature. These methods vary from pattern-matching based methods to deep learning methods [1, 11, 14, 17–19]. In addition, the problem of pattern selection has been explored in multiple contexts. In the context of HS detection, Gao et al. have used hierarchical clustering reduce the size of the dataset that is then used to train an SVM classifier [5]. In the context of lithography process modelling, Cho et al. [2] have used image parameters from lithography simulation

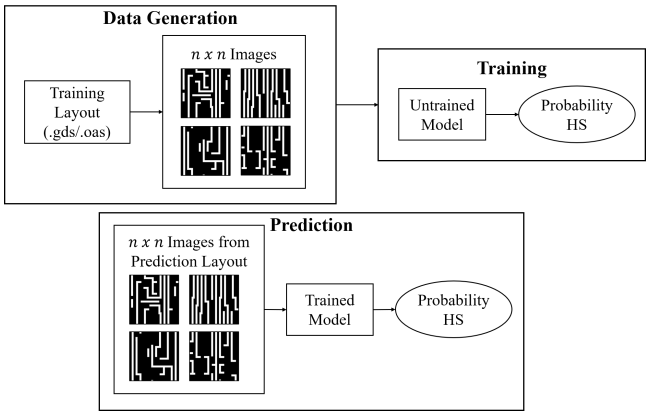

Figure 2: ML Hotspot Detection System

and the DBSCAN algorithm [3] to cluster the patterns and choose a representative set. Moreover, Gai et al. have used hierarchical clustering to reduce the number of patterns needed to calibrate process models [4].

In this paper, we explore the pattern selection problem in the context of HS detection. We propose a new sampling approach that is based on training an autoencoder [7] to project the input training patterns into a lower dimensional space, preserving the patterns' geometrical structure. The autoencoder would learn the latent features that describe the patterns, thus, eliminating the traditional feature engineering step that is done before clustering. The encoded features are then used to cluster the training patterns in the lower dimension space, and finally, for each cluster, a sampling technique is used to select the representative samples to be included in the training dataset. The previously mentioned sampling methods have usually resorted to using the median as the cluster representative. In our proposed method, we describe a more involved selection technique that improves cluster coverage by making sure that HTC patterns are sampled.

The rest of the paper is organized as follows: In Section 2, the problem statement and dataset description are discussed. In Section 3, the proposed sampling method is explained in detail. The following section describes the experiments that were done to evaluate the proposed approach. The last section summarizes the outcomes and presents our recommendations and conclusions.

2 PROBLEM STATEMENT AND DATASET DESCRIPTION

Our problem can be stated as follows: Given a particular training dataset, find the most representative patterns that can reduce the dataset size while maintaining the performance metrics of the ML model that are obtained using the full dataset.

As mentioned in the introduction, the steps for using any machine learning system can be summarized in Figure 2. The inputs to the system are $n - by - n$ grey-scale images of the layout patterns. These images are used to train any machine learning model of choice. Then, in the prediction step, pattern images are generated from the test layout and then input to the trained model, which outputs a probability for each input pattern to be HS or not. The

Table 1: ICCAD-2019 Benchmark Statistics with HS Augmentation

	Number of HS	Number of NHS
Training Dataset	1868 (467 * 4)	15276
Test Set I	1001	14621
Test Set II	64310	65523

metrics that are used to evaluate the model are recall, precision, and F1 Score.

The dataset that will be used in our experiments is the ICCAD-2019 dataset [15], which was introduced to tackle the shortcomings of the earlier ICCAD-2012 dataset. The ICCAD-2012 dataset was shown to have some underlying structural issues that led to inaccuracies in reporting the machine learning models' performance. These issues where addressed and corrected in the ICCAD-2019 dataset, showing that the false positive rates for many of the machine learning models presented in the literature were higher than what they were believed to be. Thus, we resorted to using the ICCAD-2019 dataset as the most accurate benchmark for evaluating ML models. Table 1 presents a brief description of the ICCAD 2019 dataset. One thing to note is that data augmentation was used for generating the training HS patterns. This augmentation was performed by generating 3 different orientations in addition to the original one: reflection along x and y axes, and 180 degrees rotation. Hence, the number of HS would thus be four times the one reported in the original dataset.

3 PROPOSED DATA SAMPLING METHOD

Our proposed approach for sampling the training data is divided into three steps. The first step is a dimensionality reduction step which is performed using an autoencoder. The autoencoder learns the key features that would enable it to reconstruct the input patterns, and is then used to encode all input patterns. In the second step, DBSCAN is used to group the patterns into clusters that share similar features[3]. Finally, the clustered patterns are then sampled to achieve the required data reduction, while ensuring the coverage of the subtle differences needed to correctly predict the hard-to-classify patterns. In the following subsections, each step would be explained in more detail.

3.1 Feature Reduction using Autoencoder

As described in Section 2, the input patterns to our ML system are $n - by - n$ greyscale images (n^2 pixels). Due to the high dimensionality of pixel-based feature vectors, it is difficult to obtain a meaningful clustering. Thus, dimensionality reduction is needed to obtain meaningful feature vector that can be used for clustering and then sampling the data.

The first step in the proposed flow is to perform dimensionality reduction using an autoencoder. An autoencoder can be trained to reconstruct the input images using a lower dimension feature vector. It learns different relations between the data points and can encode the high dimensional data into a lower dimensional space without a significant loss of information.

To select the size of the encoded features, multiple autoencoders with decreasing encoded feature size are trained. Based on the reconstruction loss values, the encoded feature size $d_{encoded}$ can be selected to minimize the error. Each pattern in the training dataset is then converted to the encoded representation. The dimensionality of each data point is thus reduced from n^2 to $d_{encoded}$. The encoded dataset can then be used in the next step of the flow which is clustering of the patterns using the DBSCAN algorithm [3].

3.2 DBSCAN Clustering

To reduce the size of the dataset, clustering of the patterns is done using the encoded features. The DBSCAN algorithm is used [3]. It has been selected because it can find non-linearly separable clusters and the number of clusters does not need to be specified as an input. DBSCAN has two parameters: radius of neighborhood around a point (ϵ) and the minimum number of points required to form a dense region ($minPts$). In the following experiments, $minPts$ will always be equal to one. This means that any point that does not have neighbors will be assigned a cluster on its own. This is done to simplify the flow where only a single parameter is varied. Clusters which have single points will then be handled in the sampling step of the flow. They can either be included in the sampled data or discarded based on the user's preference. The distance measure used is the Euclidean Distance defined over the autoencoder feature space.

Due to the binary nature of the HS detection dataset, there exists three different types of clusters:

- Clusters containing HS only (pure cluster)
- Clusters containing NHS only (pure cluster)
- Clusters containing both HS and NHS (hybrid cluster)

Each type of cluster will be handled in a different way when performing the sampling step. This will be discussed in the next subsection.

3.3 Sampling from Clustered Data

After clustering is done at a certain value for ϵ, sampling of the data is performed. For this purpose, another parameter is introduced to the flow which is called the sampling percentage per cluster (P). This allows the user to control how much data is sampled from each cluster. The goal of the sampling step is to select patterns that maximize the cluster coverage. The proposed sampling technique starts by obtaining the median point of the cluster. Next, the furthest point from the median is obtained. Then, the furthest point from the previous two points is obtained. This is done by summing the distances to all points from these two points and then getting the point which has the largest distance. This process is repeated until the required number of points is obtained. This gives good cluster coverage because the sample contains a pattern from the center of the cluster (median) and then several patterns at the boundary of the cluster. This method can be used for pure clusters. For hybrid clusters, the algorithm is slightly modified to ensure that HTC patterns are included in the output sample. For each point sampled, the closest pattern with the opposite label is obtained. This enhances the sampling by including HS and NHS that are close to each other to allow the model to differentiate between them.

Figure 3 shows an illustration of the algorithm using 2D points. The sampling method for pure clusters is explained in Algorithm 1 while the one for hybrid clusters is explained in Algorithm 2.

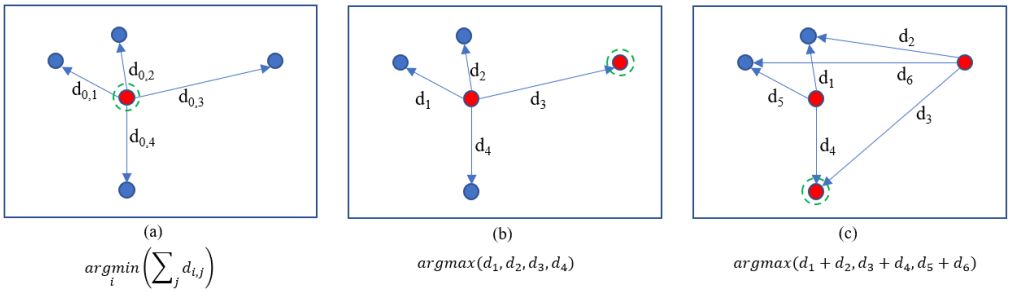

Figure 3: 2D Illustration of the Sampling Algorithm. (a): Median Selection (b), (c): Selection of Next Furthest Point.

Algorithm 1 Sampling Pure Clusters

Input: (Required Number of Samples)
Output: (List of Patterns)
- Get the Median Pattern of the Cluster
- Add Median to Output
while there are still unprocessed patterns and the required number of samples is not obtained **do**
    - Get the furthest pattern relative to the currently selected set of patterns so far
    - Add Pattern to Output
end while

Algorithm 2 Sampling Hybrid Clusters

Input: (Cluster Size (n), Sampling Percentage (P), HS/NHS ratio in the Training Dataset (hs_nhs_ratio), #HS ($input\_hs\_count$), #NHS ($input\_nhs\_count$))
Output: (List of Patterns)
- Calculate the total number of sampled patterns ($total\_count$) = $P \times n$
- Calculate the number of HS (out_hs_count) = $min(hs\_nhs\_ratio \times total\_count,\ input\_hs\_count)$
- Calculate the number of NHS (out_nhs_count) = $min(total\_count - out\_hs\_count,\ input\_nhs\_count)$
- Get the Median Pattern of the Cluster
- Add Median to Output
- Get the furthest pattern relative to the Median
- Add Pattern to Output
while there are still unprocessed patterns and the required number of samples is not obtained **do**
    - Get the furthest pattern relative to the currently selected set of patterns so far
    - Get the closest pattern with the opposite label
    if out_hs_count is not exceeded **then**
        - Add HS pattern to Output
    end if
    - Add NHS pattern to Output
end while

4 EXPERIMENTS AND RESULTS

The experimental procedure will be outlined in this section. The size of the input images for our experiments is 60x60 pixels (3600 features). The first step in the flow is performing feature reduction using an autoencoder. An autoencoder with a single hidden layer is used. The activation function is the rectified linear unit (relu) [6] and the cost function is binary cross-entropy. To find the size of the encoded features, several autoencoders with varying hidden layer sizes are trained using the full ICCAD-2019 dataset. The size of the hidden layer starts at 3600 and is reduced by half on each iteration. A graph of loss against the hidden layer size is then plotted. Figure 4 shows that the loss begins to increase starting around 225 features. Based on this plot and examining the reconstructed images, it was decided to use a feature size of 150 to encode the input patterns. The full dataset is then encoded using this autoencoder. The input feature size is 3600 and the encoded size is 150 which results in 24x reduction in feature size.

Using the encoded dataset, DBSCAN clustering and sampling are then performed. Different reduced datasets are generated by varying the values for ϵ and P. For each reduced dataset, a ML model is trained and the best model for each experiment is chosen based on the highest validation F1 score. The model used for the experiments is the Mobilenet CNN model which is known for its efficiency in Mobile Vision applications [8]. In addition, it has been recently used for HS detection on the ICCAD-2012 dataset and has shown good results [9]. The loss function used for training is the Focal Loss which is suited for problems with data imbalance [13]. The optimizer used during training is the Adam Optimizer [12].

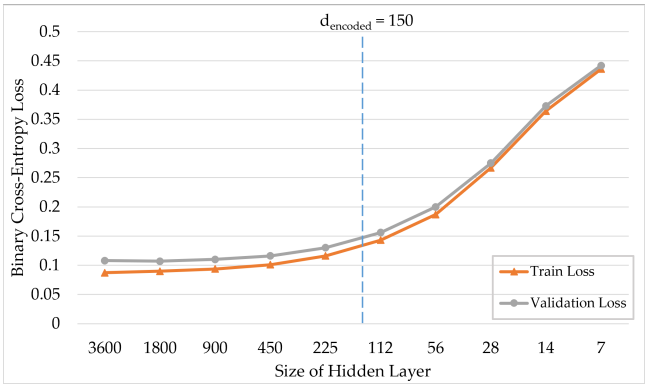

Figure 4: Plot of Binary-Cross Entropy against Hidden Layer Size of the Autoencoder

The training data is split into two parts: 90% for training and 10% for validation. The trained model is then used to run prediction on the ICCAD 2019 Test Sets I and II.

Since the objective is to perform dataset reduction without sacrificing the model performance metrics, the clustering method will be compared to the trivial random sampling method. This will be done by generating reduced datasets using random sampling and then training the same ML model. Then, the model's metrics will be compared against the one that was trained using the clustered data. The metrics used for comparison are HS recall, precision and F1 score. Each random sampling experiment is repeated 50 times and the average statistics are obtained.

Table 2 shows how the size of the reduced data as a percentage of the full dataset size varies with ϵ and P. For example, at (ϵ = 35 and P = 20), 54.5% of the full dataset is used for training. From the table, it can also be seen that the size of the sampled dataset decreases as ϵ increases and P decreases. Moreover, as ϵ approaches infinity, all the patterns are assigned to the same cluster. Hence, at large values of ϵ, the size of the dataset approaches the value of sampling percentage. This is observed at ϵ = 50 where the dataset size is very close to the value of P.

An important point to note is that at small values of ϵ, most of the clusters will contain a single pattern. In that case, it was decided to include the pattern in the output sample. Hence, at small values of ϵ (10-15), the achieved data reduction will be small. Higher reduction can be observed as ϵ increases.

To compare the proposed sampling method against random sampling, Test Set I is examined first. The difference in F1 score between the models trained with clustered data and the ones trained with random data is shown in Table 3. The table indicates an increase in F1 score relative to the random sampling method. The model

Table 2: Dataset Size (%) as a Function of ϵ and P

ϵ/P	10%	20%	30%	40%	50%	60%	70%	80%	90%
10	93.2	93.8	96.1	96.3	96.7	96.9	97.0	99.4	99.9
15	89.0	90.2	93.4	93.8	94.7	95.0	95.5	98.6	99.8
20	83.7	85.6	89.8	90.6	92.1	92.7	93.5	97.7	99.6
25	76.2	79.1	84.2	85.8	88.1	89.3	91.0	96.1	98.9
30	63.9	68.0	73.9	77.2	80.9	83.8	87.2	93.1	97.2
35	48.8	54.5	61.8	67.0	72.3	77.1	82.2	89.6	95.3
40	32.6	40.2	49.1	56.1	63.2	69.9	77.0	85.9	93.4
45	18.8	27.9	37.6	46.4	55.2	63.9	72.7	82.4	91.5
50	13.5	23.1	33.2	42.7	52.2	61.5	71.0	81.1	90.7

Table 3: Difference in F1 Score on Test Set I Between Clustering and Random Sampling Methods

ϵ/P	10%	20%	30%	40%	50%	60%	70%	80%	90%
10	-2.9	-2.2	0.3	2.2	2.5	2.1	1.7	1.4	0.2
15	-1.0	0.2	-1.7	1.1	0.2	0.6	2.4	-0.2	1.0
20	-0.1	1.7	1.5	-0.9	1.8	3.2	-1.0	-1.7	1.4
25	2.9	3.8	0.6	1.9	0.9	-2.8	-1.8	0.1	1.1
30	1.6	2.4	0.0	2.5	1.4	3.2	-0.2	3.3	-1.9
35	5.4	3.4	3.1	3.2	2.2	4.1	2.9	3.5	-1.3
40	2.2	4.3	-0.9	0.9	1.9	-1.7	1.9	0.8	1.4
45	-4.6	-4.0	-4.8	2.0	-0.3	3.3	-1.5	1.4	1.2
50	-0.4	-7.1	-4.6	-1.1	0.6	-0.8	1.0	5.7	1.0

Table 4: Recall on Test Set I

ϵ/P	10%	20%	30%	40%	50%	60%	70%	80%	90%
10	82.1	82.6	75.6	78.9	78.0	72.3	76.4	77.9	81.7
15	80.9	78.3	83.9	76.5	76.5	75.9	73.3	80.5	76.7
20	76.3	77.4	81.0	77.7	82.0	82.0	75.8	81.5	77.5
25	78.5	81.7	82.4	79.1	80.8	80.1	79.1	80.2	87.1
30	79.7	75.9	85.7	76.6	74.0	76.7	78.6	79.2	79.5
35	81.3	76.3	82.5	74.2	73.6	75.2	75.4	83.6	77.2
40	79.3	84.6	82.0	83.6	80.2	80.8	81.4	79.7	79.7
45	59.3	63.4	69.6	82.6	79.2	79.4	81.9	77.4	79.3
50	38.8	62.5	67.5	69.4	70.1	73.2	72.7	80.9	81.9

Table 5: Precision on Test Set I

ϵ/P	10%	20%	30%	40%	50%	60%	70%	80%	90%
10	49.4	48.9	55.9	57.1	59.1	60.2	57.4	57.5	52.7
15	51.9	55.6	51.0	57.8	57.0	57.2	60.9	54.0	58.2
20	53.3	54.8	54.6	54.4	55.6	56.8	54.2	52.1	56.9
25	56.5	55.4	53.4	56.4	54.0	50.8	52.1	56.2	52.1
30	51.7	54.3	48.7	56.5	57.0	58.2	54.3	59.2	52.7
35	53.5	55.0	52.5	57.5	55.8	59.2	59.3	56.4	53.9
40	46.8	48.6	45.9	48.1	52.3	49.0	53.3	54.3	55.9
45	40.1	43.5	42.0	47.8	47.3	53.1	47.9	55.6	55.4
50	37.7	35.2	41.3	48.2	52.0	50.7	55.8	58.9	53.6

Table 6: Difference in F1 Score on Test Set II Between Clustering and Random Sampling Methods

ϵ/P	10%	20%	30%	40%	50%	60%	70%	80%	90%
10	-0.2	-0.1	-0.2	0.1	0.5	0.0	-0.1	0.2	0.2
15	-0.3	0.1	-0.1	-0.4	-0.2	0.0	0.5	-0.2	0.2
20	0.2	0.1	0.0	-0.1	0.0	-0.2	0.0	0.0	-0.4
25	0.0	0.2	-0.1	-0.2	-0.3	0.0	-0.4	0.1	0.2
30	0.0	-0.2	0.1	-0.1	0.0	0.1	0.1	0.6	-0.1
35	0.3	-0.2	0.1	-0.2	-0.2	0.0	0.1	0.4	-0.1
40	-0.8	-0.1	-0.3	0.1	0.1	0.2	-0.3	-0.2	0.2
45	-2.9	-3.8	-1.7	0.2	-0.2	0.3	-0.2	0.1	-0.3
50	1.9	0.0	-1.3	0.2	-1.3	-0.9	-0.8	-0.4	0.1

with the largest positive difference of 5.7% is at (ϵ = 50, P = 80). This corresponds to a dataset size of 81.1% and F1 score of 68.2. The second-best model is at (ϵ = 35, P = 10). It has a difference in F1 score of 5.4%. The dataset size is 48.8% and the F1 score is 64.6.

Tables 4 and 5 show the recall and precision values obtained on Test Set I for each combination of ϵ and P. These tables provide a way to choose different models that suit various applications. For example, when the detection of HS is more important (e.g., defect analysis), models having the highest recall can be used such as the one at (ϵ = 25, P = 90) which has a recall of 87%. For applications where the cost of false alarms is high (e.g., design), then models having high precision can be selected. An example of this is the model at (ϵ = 15, P = 70) which has the highest precision of about 61%. A trade-off between recall and precision can also be made by maximizing F1 score. Hence, clustering and sampling enable customization of the training data to achieve different outcomes.

Table 7 lists several models trained using different subsets of the ICCAD-2019 dataset. The first model was trained using the full ICCAD-2019 dataset. The following models were trained using

Table 7: Comparison of Mobilenet Models Trained with Different Sampled Datasets using ICCAD-2019 Test Set I

#	Description	Dataset Size	Training Time (s)	Recall	Precision	F1
1	Baseline	100%	300 (100%)	86.7	52.2	65.2
2	$\epsilon = 50$, $P = 80$	81.10%	200 (66.7%)	80.9	58.9	68.2
3	Random Sampling	81.10%	200 (66.7%)	80.4	51.2	62.5
4	$\epsilon = 35$, $P = 10$	48.80%	100 (33.3%)	81.3	53.5	64.6
5	Random Sampling	48.80%	100 (33.3%)	84.8	45.4	59.2

reduced datasets obtained via our proposed approach in addition to the random sampling approach. Comparing models 2 and 3, it can be seen that while random sampling lead to a drop in the F1 score metric relative to the baseline model, the proposed method was able to reduce the run time by around 33% while obtaining a higher F1 score than the baseline. Models 4 and 5 confirm these results even further. The large data reduction using random sampling caused the F1 score to drop below 60% while the same reduction using our proposed method achieved almost the same F1 score as the baseline. This shows that clustering and sampling the training dataset results in choosing better representatives, thus, reducing the time needed for training and improving the data balance.

For Test Set II, the difference in F1 score between clustering and random sampling is not significant as shown in Table 6. This proves that the lack of coverage in the training dataset is the dominating factor, hence our proposed method for data selection will not influence the final trained model. Therefore, data enrichment of the training dataset is needed to improve the model and show the value of dataset selection. Synthetic pattern generation can be used to enrich the training dataset. In [16], a new method was introduced where subtle differences between patterns were generated by randomly moving pattern edges. Exploring similar techniques is needed to create richer datasets for ML hotspot detection models.

5 CONCLUSION AND FUTURE WORK

In this paper, we presented an efficient data sampling approach for selecting the most representative samples for machine learning-based hotspot detection. The latent features obtained using autoencoders were used to reduce the dimensionality of the data points. Then, these features were used for pattern clustering. This method allows users to obtain representative training patterns for ML hotspot detection. We showed that this technique gives the ability to reduce the amount of data needed to train ML models while maintaining the levels of recall and precision. This can be attributed to the fact that our data sampling approach strikes a balance between the number of HS and NHS. This, in turn, leads to a better balanced training dataset that results in building improved HS detection models in a shorter period of time.

Finally, as a recommendation for future work, it would be beneficial to build larger benchmark datasets having more pattern variations to test and evaluate this method in an industrial IC design scenario.

REFERENCES

[1] Vadim Borisov and Jurgen Scheible. 2018. Lithography Hotspots Detection Using Deep Learning. In *2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. IEEE, Prague, 145–148. https://doi.org/10.1109/SMACD.2018.8434561

[2] Gangmin Cho, Yonghwi Kwon, Pervaiz Kareem, Sungho Kim, and Youngsoo Shin. 2021. Test pattern extraction for lithography modeling under design rule revisions. In *Optical Microlithography XXXIV*, Soichi Owa and Mark C. Phillips (Eds.). SPIE, Online Only, United States, 23. https://doi.org/10.1117/12.2583916

[3] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press, 226–231.

[4] Tianyang Gai, Ying Chen, Pengzheng Gao, Xiaojing Su, Lisong Dong, Yajuan Su, Yayi Wei, and Tianchun Ye. 2019. Sample patterns extraction from layout automatically based on hierarchical cluster algorithm for lithography process optimization. In *Design-Process-Technology Co-optimization for Manufacturability XIII*, Jason P. Cain and Chi-Min Yuan (Eds.). SPIE, San Jose, United States, 32. https://doi.org/10.1117/12.2514177

[5] Jhih-Rong Gao, Bei Yu, and David Z. Pan. 2014. Accurate lithography hotspot detection based on PCA-SVM classifier with hierarchical data clustering, John L. Sturtevant and Luigi Capodieci (Eds.). San Jose, California, USA, 90530E. https://doi.org/10.1117/12.2045888

[6] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 15)*, Geoffrey Gordon, David Dunson, and Miroslav Dudík (Eds.). PMLR, Fort Lauderdale, FL, USA, 315–323. https://proceedings.mlr.press/v15/glorot11a.html

[7] Geoffrey E. Hinton and Richard S. Zemel. 1993. Autoencoders, Minimum Description Length and Helmholtz Free Energy. In *Proceedings of the 6th International Conference on Neural Information Processing Systems* (Denver, Colorado) *(NIPS'93)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3–10.

[8] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861 [cs]* (April 2017). http://arxiv.org/abs/1704.04861 arXiv: 1704.04861.

[9] Xuanyu Huang, Rui Zhang, Yu Huang, Peiyao Wang, and Mei Li. 2021. Enhancements of Model and Method in Lithography Hotspot Identification. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, Grenoble, France, 102–107. https://doi.org/10.23919/DATE51398.2021.9473963

[10] Aliaa Kabeel, Wael ElManhawy, Joe Kwan, Asmaa Rabie, Mohamed Ismail, Ahmed Khater, and Sarah Rizk. 2020. Machine Learning using retarget data to improve accuracy of fast lithographic hotspot detection. In *Design-Process-Technology Co-optimization for Manufacturability XIV*, Chi-Min Yuan (Ed.), Vol. 11328. International Society for Optics and Photonics, SPIE, 1 – 7. https://doi.org/10.1117/12.2551827

[11] Namjae Kim, KiHeung Park, Jiwon Oh, Sangwoo Jung, Sangah Lee, Jae-Hyun Kang, Seung Weon Paek, Kareem Madkour, Wael ElManhawy, Aliaa Kabeel, Ahmed ElGhoroury, Marwah Shafee, Asmaa Rabie, and Joe Kwan. 2019. Machine learning to improve accuracy of fast lithographic hotspot detection. In *Design-Process-Technology Co-optimization for Manufacturability XIII*, Jason P. Cain and Chi-Min Yuan (Eds.). SPIE, San Jose, United States, 41. https://doi.org/10.1117/12.2515139

[12] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. Focal Loss for Dense Object Detection. *arXiv:1708.02002 [cs]* (Feb. 2018). http://arxiv.org/abs/1708.02002 arXiv: 1708.02002.

[14] Tetsuaki Matsunawa, Shigeki Nojima, and Toshiya Kotani. 2016. Automatic layout feature extraction for lithography hotspot detection based on deep neural network, Luigi Capodieci and Jason P. Cain (Eds.). San Jose, California, United States, 97810H. https://doi.org/10.1117/12.2217746

[15] Gaurav Rajavendra Reddy, Kareem Madkour, and Yiorgos Makris. 2019. Machine Learning-Based Hotspot Detection: Fallacies, Pitfalls and Marching Orders. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, Westminster, CO, USA, 1–8. https://doi.org/10.1109/ICCAD45719.2019.8942128

[16] Gaurav Rajavendra Reddy, Constantinos Xanthopoulos, and Yiorgos Makris. 2021. On Improving Hotspot Detection Through Synthetic Pattern-Based Database Enhancement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2021), 1–1. https://doi.org/10.1109/TCAD.2021.3049285

[17] Moojoon Shin and Jee-Hyong Lee. 2016. Accurate lithography hotspot detection using deep convolutional neural networks. *Journal of Micro/Nanolithography, MEMS, and MOEMS* 15, 4 (Nov. 2016), 043507. https://doi.org/10.1117/1.JMM.15.4.043507

[18] Moojoon Shin and Jee-Hyong Lee. 2016. CNN Based Lithography Hotspot Detection. *The International Journal of Fuzzy Logic and Intelligent Systems* 16, 3 (Sept. 2016), 208–215. https://doi.org/10.5391/IJFIS.2016.16.3.208

[19] Bei Yu, David Z. Pan, Tetsuaki Matsunawa, and Xuan Zeng. 2015. Machine learning and pattern matching in physical design. In *The 20th Asia and South Pacific Design Automation Conference*. IEEE, Chiba, Japan, 286–293. https://doi.org/10.1109/ASPDAC.2015.7059020