JM3 Journal of Micro/Nanopatterning, Materials, and Metrology

RESEARCH PAPER

Geometrical positioning surveying-based features for BEOL line-end-pull-back modeling using regression-based machine-learning

Ahmed Hamed Fathi Hamed,[a,b,*] **Hazem Hegazy**,[c] **Omar El-Sewefy**,[c] **Mohamed Dessouky**,[a] **and Ashraf Salem**[d]

[a]Ain Shams University, Electronics Engineering and Electrical Communications Department, Faculty of Engineering, Cairo, Egypt

[b]Siemens EDA, Calibre Semi-Manufacturing Solutions, Siemens Industry Software Inc., Cairo, Egypt

[c]Siemens EDA, Calibre Design-to-Silicon, Siemens Industry Software Inc., Cairo, Egypt

[d]Ain Shams University, Computer and Systems Engineering Department, Faculty of Engineering, Cairo, Egypt

ABSTRACT. **Background:** Line-end-pull-back (LEPB) is a well-known systematic defect in BEOL metal layers, where a line-end (LE) tip is pulled back from its desired location due to lithography (litho) process effects. Severe LEPB directly affects BEOL connectivity and may lead to partial or total metal-via disconnection.

Aim: LEPB can be characterized through model-based litho simulations but at the cost of high computational resource consumption. This study aims to provide a fast and accurate approximation of computationally expensive litho simulations through regression-based machine learning (ML) modeling.

Approach: LEPB modeling is approached through the LightGBM model. Input features were approached using density pixels, density concentric circle area sampling (CCAS), and geometrical positioning surveying (GPS), which is an edge-based engine that provides a direct one-to-one mapping between model features and geometrical measurements between the LE as a point-of-interest and its surrounding contextual patterns. The importance of LightGBM features by splits was employed to reduce features across the used approaches.

Results: The reduced features of GPS produced almost the same modeling quality (training: RMS $= 0.571$ nm, $\delta_{\text{EWD}} = 0.297$ nm, and $R^2\% = 98.74\%$, and testing: RMS $= 0.643$ nm, $\delta_{\text{EWD}} = 0.344$ nm, and $R^2\% = 98.40\%$) with -22.22% fewer number of features and less feature extraction runtime compared to the full features set of density pixels and density CCAS approaches.

Conclusions: Compared to model-based litho simulations, the obtained calibrated ML models can be used to provide fast, yet accurate predictions of the amounts of pull-back or extensions introduced at LEs near vias, eliminating a major contributor to systematic IC yield loss.

Keywords: design-for-manufacturability; geometrical positioning surveying; directional geometrical kernels; machine learning models; line-end-pull-back modeling; IC layout features extraction

Paper 22075G received Dec. 8, 2022; revised May 27, 2023; accepted Jun. 5, 2023; published Jun. 29, 2023.

*Address all correspondence to Ahmed Hamed Fathi Hamed, ahmed.hamedfatehy@siemens.com

1 Introduction

Machine learning (ML) techniques are dynamic powerful tools that perfectly fit complex big data modeling problems, which is the domain inhibited by most electronic design automation (EDA) problems. Huang et al.[1] comprehensively surveyed incorporated ML applications that approach several EDA tasks, mostly NP-complete tasks, showing how ML techniques are efficiently employed in resolving complex big data problems, replacing traditional approaches. During the past few years, ML techniques have experienced revolutionary adoptions in many EDA domains, especially integrated circuit (IC) design and manufacturing domains, due to hardware utilization and software reusability that provides ease-of-use, as well as ready-for-integration packaged ML solutions. Conventional lithography (litho) simulations, including optical and etching model-based simulations, are one of the most computationally expensive steps during IC manufacturing. With large chip areas, simulation time and consumed memory are increasing exponentially. Line-end-pull-back (LEPB), in which a line-end (LE) tip is pulled back from its desired location due to litho process effects, can be precisely characterized through model-based litho simulations, but at the cost of high computational resources consumption. For specific problems, such as LEPB modeling, ML can provide a fast and accurate approximation of computationally expensive simulations. In Sec. 1.1, we review ML applications in IC design and manufacturing phases, in Sec. 1.2, we identify ML modeling approaches for IC layouts, and in Sec. 1.3, we summarize common IC layout features representations. In Sec. 1.4, we state the aspects of LEPB modeling.

1.1 ML Employment in IC Layout Applications

Wide varieties of ML applications have been incorporated throughout modern chip design and fabrication flows, in both the IC design and manufacturing phases. Our review in this study focuses on ML applications that use features derived from IC layout pattern representations.

1.1.1 *ML applications for IC layouts during IC design-phase*

Routing is the earliest step in the IC design flow in which the IC layout starts to be—its principal characteristic is a nearly final definition of the physical structure of a design. AENEID[2] is a litho-friendly detailed router that uses litho simulation data from hotspot detection (HD) kernels, which is an improved extension of Ref. 3. The AENEID router combines geometrical features of layout patterns fragments with routing path data to perform detailed routing, avoiding potential hotspots (HS) in the routing paths using artificial neural network (ANN) and support vector machines (SVM) models. Xie et al.[4] presented an ML application in IC design to predict design rule checking (DRC) HS and design rules violations counts using a fully connected convolution network (FCN) that takes red-green-blue (RGB) pixelized images of layout clips together with routing information as input features. Liang et al.[5] presented a DRC HS prediction for IC designs in a sub-10 nm process using J-Net, which is a customized convolutional neural network (CNN) based on extending the U-Net architecture[6] and using high-resolution pixelized images for layout clips as an input to the ML model. Pattern matching is one of the most frequently used tools by both IC designers and manufacturers, and one of its significant applications is detection of systematic defects and HS. In advanced technology node fabrications, libraries of HS patterns are shared by IC manufacturers with IC designers, so they can avoid the use of these patterns in their designs. An illustration for such a flow is shown by Selvam et al. in Ref. 7, where the layout clips are represented using density vectors, then CNN deep learning is used to detect process-sensitive patterns. Although the flow is designed to predict litho HS patterns in the manufacturing phase, but it is exported to IC designers to predict problematic patterns during design implementation. Wuu et al.[8] used density grid pixel components of layout clips as pattern representations with two-level classifiers based on SVM ML model to classify patterns and predict HS. Design-for-manufacturability (DFM) checks are part of IC physical verification and can be defined as conservative rules checks beyond DRC that provide additional fabrication process margin to IC designers. Wang et al.[9] incorporated 17 localized geometrical features for metal-via LE patterns with multi-layer perceptron (MLP) neural network (NN) architecture to check for metal-via enclosure DFM violations. The aim of this model is to aid the IC design process and fix potential

Table 1 ML applications use layout extracted features during IC design phase.

Problem domain: IC design	Task	ML model	Layout pattern representation features	Reference
Routing	Litho-friendly detailed router	ANN/SVM	Fragments-based geometrical features	2
Physical verification	Predict DRC HS	CNN	Layout clips: RGB pixelized images	4
		J-Net	Layout clips: pixelized images	5
Pattern matching	HS prediction	SVM	Density features	8
		CNN	Density grid pixels vector	7
DFM	Retarget-aware DFM checking	NN (MLP)	Localized geometrical and density features	9

DFM violations before manufacturing. Table 1 shows a summary of ML applications that use layout-extracted features during the IC design phase, mainly the routing, physical verification, pattern matching, and DFM steps.

1.1.2 *ML applications for IC layouts during IC manufacturing phase*

Several ML applications were proposed to solve manufacturing-related problems in domains such as litho HS detection,[3,10–19] optical proximity correction (OPC),[20–22] sub-resolution assist features (SRAF) insertion,[23,24] and litho simulation.[25] Litho HS detection is one of the most plentiful domains for ML applications, with a wide variety of related layout representation features. Ding et al.[3] presented a litho HS detection in a single layer using fragment-based geometrical features and a hybrid ML model combining ANN and SVM. Yang et al.[10] and Shin et al.[11] approached the problem using a deep CNN to process pixel-based images representing the layout and extract the feature map out of these layout images. Jiang et al.[12] presented binarization approaches in both layout representation and ML model parts to reduce execution time by avoiding floating-point kernels. Layout clips are represented by binarized images and then passed to a binarized neural network (BNN) to mitigate the complexity and reduce execution time. Feature tensors and deep-biased learning were used by Yang et al. in Ref. 13, which starts by dividing the layout clips into sub-blocks and obtaining the feature tensors by encoding on discrete cosine transform (DCT) coefficients of each sub-block. The extracted features tensors are then passed to a CNN model, followed by two fully connected nodal layers to get the classification output. In Ref. 14, Matsunawa et al. segmented layout clips into blocks represented by density features, then projected to lower-dimensional space through principal component analysis (PCA) to maintain the selection of features with high linear separability in lower dimensional space. The selected features are passed to real AdaBoost, a decision-tree-based ML model using adaptive boosting (AdaBoost). In Ref. 15, Matsunawa et al. tackled the HS using a deep-learning approach with DNN model that has a self-features extraction and HS detection functionality. Zhang et al.[16] represented the layout clips by concentric circle area sampling (CCAS) and proposed an ML model that employs smooth boosting. Dawar et al.[17] extracted aerial image and geometrical parameters with a random-forest (RF) classifier to detect litho HS. The extracted aerial image parameters are done with approximations and include: (a) process variation band, (b) normalized image log-slope, and (c) mask error enhancement factor. The extracted geometrical parameters are performed for each edge of a polygon and include: (a) external space and width and (b) densities around an edge. Yang et al.[18] used the adaptive squish as a layout patterns representation with a CNN architected model containing a significant portion of architecture dedicated for features extraction, and in Ref. 19 used a more tailored, deeper CNN architecture named "SquishNet." Table 2 shows a summary of ML applications that use layout extracted features during the IC manufacturing phase, mainly the detection of litho HS, OPC, SRAF placement, and litho simulation steps.

Table 2 ML applications use layout extracted features during IC manufacturing phase.

Problem domain: IC manufacturing	ML algorithm	Layout pattern representation features	Reference
Litho HS detection	ANN/SVM	Fragments-based geometrical features	3
	BNN	Layout clips: binarized images	12
	DNN	Density-based features	15
	CNN	Layout clips: pixelized image	10, 11
		Frequency/DCT domain feature tensor	13
	AdaBoost, DT	Layout clips: density-based features with PCA	14
	Smooth boosting	CCAS	16
	RF	Aerial-image and geometrical parameters	17
	SquishNet/CNN	Adaptive squish	19
	CNN	Adaptive squish	18
OPC	GAN	Layout clips: pixelized image	20
	HBM	CCAS	21
	MLP	Polar Fourier transformation (PFT) basis functions	22
SRAF placement	Logistic regression and DT	CCAS	23
	Supervised online dictionary learning and logistic regression	CCAS	24
Litho simulation	Litho generative adversarial network (GAN)(CNN and GAN)	Layout clips: RGB pixelized images	25

1.2 ML Modeling Approaches for IC Layouts

A general perspective on ML learning approaches was highlighted by Lin et al.[26] Considering the reviewed ML applications that use IC layouts in the previous Sec. 1.1, we reintroduce this perspective with slight modification. ML modeling can follow one of two schemes: (a) traditional approach or (b) deep-learning approach, as shown in Fig. 1. In the traditional approach, the input data represent extracted features from the IC layout, and these features are passed to a predefined architected off-the-shelf ML model from libraries, such as SciKit learn.[27] The main challenges

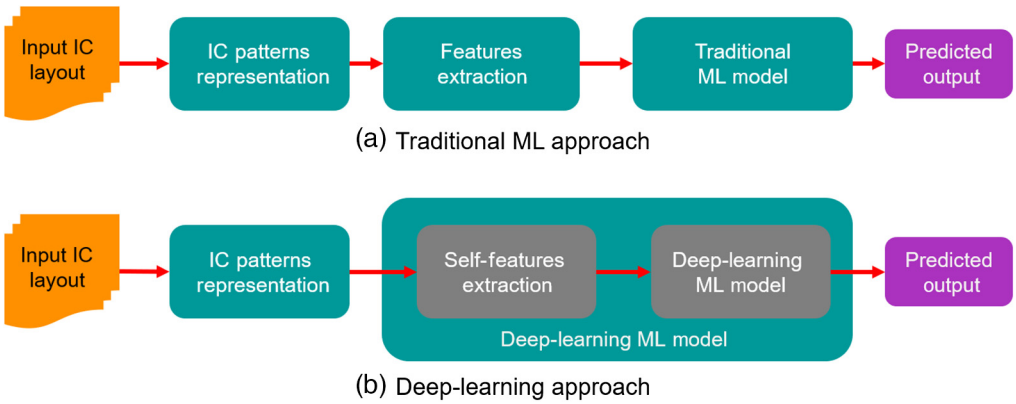


Fig. 1 ML modeling approaches: (a) traditional ML approach and (b) deep-learning approach.

with this approach are (1) choosing an adequate IC layout representation, (2) extracting descriptive features, (3) selecting a suitable off-the-shelf ML model that fits properly with the learning task, and eventually (4) tuning the model's parameters to achieve the required accuracy in training and testing phases and avoid model overfitting. This approach provides a relatively shorter model development turnaround-time (TAT); however, it needs cautious, careful selections throughout the flow. Applications such as Refs. 2 and 8 in IC design phase, and Refs. 3, 14, 16, 23, and 26 in the IC manufacturing phase follow this approach.

In the deep-learning approach, the input IC layout patterns representations are passed to a deep-learning ML model from libraries, such as TensorFlow.[28] In this case, the model contains an automated self-features extraction process in addition to the learning process. The self-features extraction is carefully designed and architected in compromise with the learning process's objectives to extract meaningful features contributing to the model accuracy and avoid overfitting in regression-based learning and false-positives in classification-based learning. For example, in litho HS detection applications, Shin et al.[11] presented self-feature extraction through CNN deep-learning. The input layout is parsed into pixelized images through a sliding scanning window. Scanned images are passed to CNN with four convolutional pools for features extraction. The numbers of connections per image in the four convolutional pools are 1.6×10^8, 1.5×10^9, 8.6×10^8, and 1.5×10^9, respectively, and must be run with GPU assistance to accelerate the convolutional operations. Yang et al. in Refs. 10 and 19 presented deeper CNNs architectures for feature extraction and also ran with GPU assistance. Selvam et al.[7] used density-grid pixels vectors with deep-learning CNN run with GPU assistance as well. Matsunawa et al.[15] approached the feature extraction using density-pixels and DNN, where the input layout is parsed into clips, which are passed to DNN with an input layer (3600: 60×60 nodes) and four hidden layers: (196: 14×14 nodes), (196: 14×14 nodes), (144: 12×12 nodes), and (144: 12×12nodes), respectively. Specifying the appropriate DNN architecture is an iterative trials process[15] objective by minimizing the loss function based on the number of used hidden layers, hyperparameters, transfer functions, and network connections. The main challenges with the deep-learning approach are (1) selecting the proper architecture for the deep-learning model, as there is no global solution that can fit all applications, (2) selecting the proper convolutional kernel sizes for CNN models and the number of hidden layers and the number of nodes per layer for DNN models, (3) selecting the model's activation functions and tuning the hyperparameters to achieve the optimum accuracy, and (4) selecting proper layers connections and dropout ratios between layers to avoid model overfitting. All these steps are experimental and impact the model development TAT. In general, the optimized input clip size and pattern representation play a significant role in the accuracy of the deep-learning models. Moreover, finding the appropriate compromised architecture with proper hyperparameters of the self-features extraction process and the learning process is a significant material of research that might also result in longer model development TAT. The deep-learning approach can be traced in applications, such as Refs. 4 and 7 in the IC design phase, and Refs. 10 to 13, 15, 18, and 19 in the IC manufacturing phase.

In this study, we focus on efficient IC pattern representation with predefined architected off-the-shelf ML models to avoid deep-learning's self-features extraction that requires high computational resources, such as GPU assistance and longer model development TAT.

1.3 IC Layouts Feature Representation for ML Applications

Many pattern representations have been introduced to extract ML features for IC layout patterns. Although fragment-based geometrical features[2,3] provide geometrical information about the context around a fragment of interest, this approach generates a massive amount of data per fragment for a full chip layout that requires equally massive computational and storage resources. Density-based features[7,8,14,15] are one-to-many geometrical encoding representations, as one density value can be mapped to different geometrical shapes, which drops some of the spatial data about the context around a point of interest (POI) based on the selected resolution. To capture dense geometrical information, especially with tight layout dimensions, high resolution with a small grid size should be used, which may require a deep-learning approach with self-features extraction capabilities.[7,15] Although Matsunawa et al.[14] presented an optimized approach in selecting adequate density grid pixels representation (grid area 1.2 μm and grid size 10×10 pixels) for litho HS detection without deep-learning by measuring the distance between HS and NHS

patterns in a lower dimensional space using PCA components, this representation is bounded by a classification problem. For a regression-based problem, we may need higher resolution and accordingly, a larger number of features that may require deep-learning. CCAS was first introduced by Matsunawa et al.[21] and applied on a fragmentation basis with a hierarchical Bayes model (HBM) to reduce the number of iterations in model-based OPC. It considers the fact that diffracted light from the mask patterns is propagating concentrically. CCAS was used in Refs. 23 and 24 for SRAF placement and in Ref. 16 for litho HS detection. Although CCAS provides a compact IC layout patterns representation, such as density-based features, it still drops some of the spatial data about the context around the POI, especially with high resolution representation. Increasing the number of sampling points creates adjacent circles with redundant information. This problem was highlighted by Geng et al.,[24] who tried to eliminate this redundancy through self-adaptive dictionary learning in a supervised feature encoding step, where CCAS features are mapped into a different lower dimensional space as sparse encoded features to eliminate the redundancy. The main problem with this technique is data leakage,[29] where the feature encoding's objective function jointly uses the supervised predicted labeled data to encode features and then passes a piece of prediction data to the ML model's input encoded features. In Ref. 16, Zhang et al. tried to optimize the selection of circles index using mutual information to compute the dependency between circle indexes and classification prediction variable. The training and testing data are extracted using this optimized circle indexes, which can also be considered as a source of data leakage, as the testing data set has prior knowledge about the optimized circle index. Image-pixel-based features[4,5,10,11,20,25] do not provide a direct mapping to features, but need deep-learning models with self-features extraction. This approach is usually employed with CNN models and potentially needs GPU assistance. It should be highlighted that both density-based-pixels and image-based-pixels feature representation quality directly depend on selecting adequate resolutions for the layout critical dimensions (CD) to maintain descriptive details. Moreover, the information gain in every pixel is very high and that challenges features reduction and features merging or synthesis without deep-learning self-feature extraction. Frequency domain/DCT features[13] are not computationally friendly—layout patterns are clipped and subjected to density-based scanning, then transformed into the frequency domain using DCT. Similarly, aerial image features[17] and PFT features[22] need optical simulations to be calculated, which may be suitable for ML-OPC and litho HS detection applications where features calculation is bounded by the downstream application and embedded within the overall flow. Adaptive-squish-pattern[18,19] is an adaptive-grid bitmap representation for a layout pattern that provides a compact lossless representation of layout patterns. However, it does not provide direct one-to-one mapping with geometrical features, but requires a deep-learning feature extraction through CNN models as presented in Refs. 18 and 19. Although Ref. 9 presented a mixture of density features and localized extracted geometrical features to represent layout patterns, the list of extracted features still ignores several descriptive details about the contextual patterns, such as corners and internal and external spacings. To address this problem, we present a novel edge-based approach named geometrical positioning surveying (GPS)[30] to represent the layout POIs and their surrounding context of patterns. Further details are discussed in Sec. 2.

1.4 Line-End-Pull-Back Modeling Aspects

The LEPB problem can be characterized by two main aspects: (1) LE corner rounding and (2) LE shortening. Based on Fraunhofer diffraction patterns and Fourier optics,[31] the band limitations of the optics system filter out high spatial frequencies and impose rounding at the corners[32]; in LEs, this rounding is severe. LE shortening depends on the line width and surrounding context—isolated or semi-isolated LEs whose width is near the resolution limit of the litho process typically suffer from noticeable shortening on the wafer,[31] whereas very dense LEs tend to merge with neighbors. Furthermore, the etching process exacerbates the impact of corner rounding and LE shortening. Consequently, fabrication process variation sensitivity affects both LE corner rounding and LE shortening. Figure 2 shows the difference in corners and LEs between (a) layout designed patterns and (b) simulated patterns through a litho (optical and etching) manufacturing system.

In the conventional approach, LEPB modeling is conducted as a part of OPC modeling[32] and is objective by an edge-placement-error convergence at LEs, which mandates careful tuning for

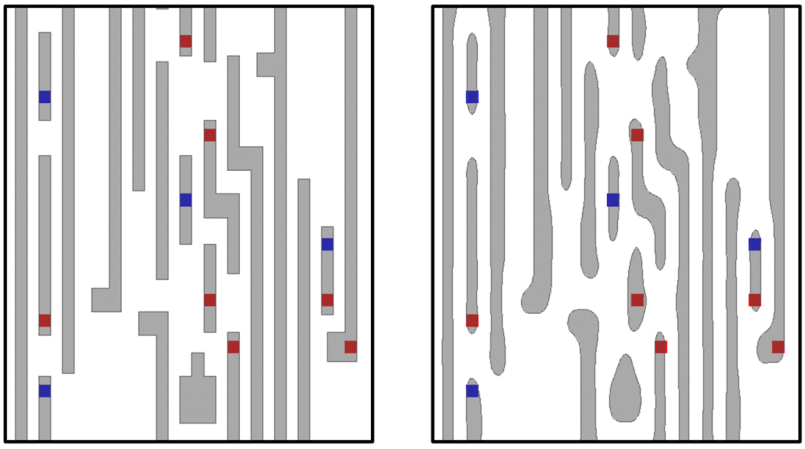


Fig. 2 Differences in corners and LEs between: (a) layout designed patterns and (b) layout lithography simulated manufactured patterns.

LE fragments in the OPC recipes. ML-OPC[20–22,33] and GPU-based OPC[34] have been proposed to accelerate OPC modeling. LEPB amounts can be precisely characterized through calibrated model-based litho (optical and etching) simulations, which require high computational resources consumption. Shin et al.[11] estimated that model-based litho simulation usually takes more than 100 CPU h/mm^2, and it could take several days to prepare accurate calibrated models. We explain the conventional litho simulation flow in detail in Sec. 3.2.

In this paper, we present fast and accurate LEPB modeling through a supervised regression-based ML approach. Input features were approached using density pixels, density CCAS, and GPS, which is a novel edge-based approach that describes an LE as a POI and its surrounding context of patterns in the IC layout. GPS provides a direct one-to-one mapping between input features passed to the ML model and physical measurements between the POI and its surrounding patterns. The remaining sections are organized as follows: Sec. (2) explains GPS, Sec. (3) explains the conventional litho simulation flow used to model LEPB and data collection processes, Sec. (4) demonstrates the ML modeling flow, Sec. (5) demonstrates the experiments conducted on an industrial test case, and Sec. (6) presents our conclusions.

2 Geometrical Positioning Surveying

As discussed in Secs. 1.1–1.3, most layout context analysis focuses on image-based analysis (such as pixel-based image analysis) or density-based analysis, both of which use identical informational density datasets and accordingly, require a self-feature extraction process in deep-learning ML models that mandates both high-computational resources, such as GPU assistance, and (usually) longer model development TAT to select the proper model architecture. Moreover, it limits both the ML model interpretability and the ability to decompose the effect of different contextual polygons on a selected POI, as highlighted in Ref. 15. GPS[30] is an edge-based engine, implemented using the Calibre® standard verification rule format (SVRF), that extracts direct one-to-one directional geometrical features about a POI and its surrounding context of polygons in the layout. These features are measured and extracted through directional geometrical kernels (DGKs), which are the core measuring functions of GPS that use Calibre SVRF edge-based operations of geometrical measurements and processing. Further details about DGKs are provided in Sec. 2.3.

2.1 Constrained Generality Concept in IC Layouts

The functionality of GPS relies on the constraint generality concept (CGC), which is inherent to IC layout designs. In essence, the CGC ensures that patterns in a layout design are not unconstrained, such as in a freeform drawing that may take any shape. Layout designers are restricted to certain types of geometries, such as polygons, as well as being subject to other restrictions

derived by place and route tools, DRC and DFM rules, etc., all of which imply constraints in polygon width, spacing, direction, etc. These constraints not only result in layout design patterns that are a much smaller subset of the freeform population but also support the ability of the edge-based engine to describe an edge's contextual relations and measurements to neighboring polygons.

2.2 GPS Measurement Constructors

As shown in Fig. 3, a central POI is placed at the center of the LE tip of a residential polygon surrounded by contextual polygons in all directions and with different structures and dimensions.

A respective edge is defined based on two main features: topological features and dimensional features.[30] Topological features comprise quantized values restricted to a discrete set that describes a certain geometrical state about the edge, such as orientation (horizontal, or vertical) and corner type (convex, concave, or no corner). Dimensional features represent continuous numerical measurements, such as spaces, widths, enclosures, and angles. Thus, topological features are based on the conditional constraints of the layout design process, and dimensional features expound on the numerically constrained particulars of a given layout design. Consequently, decomposing the quantifiers of edges (which are the basic constructor of patterns) into topological and dimensional features requires GPS to extract direct one-to-one directional geometrical features about a POI and its surrounding context of polygons in the layout.

This description is decomposed into the following constructors:

- Directional relations vector, which describes the border around the POI within the search distance (SD) in predefined directions coordinated by the POI, providing the directionality aspect to measured geometrical properties.
- Orientation relations vector, which describes the orientation of the edge/polygon, such as horizontal or vertical.
- Corner relations vector, which describes the corners and defines corner properties, such as the type of corner and its constituting edges.
- Geometrical relation vector, which describes the geometrical properties of the residential polygon and the contextual polygons, such as widths and spaces.
- Zone relation matrix, which compiles various geometrical properties into different zones according to the distance from the POI.

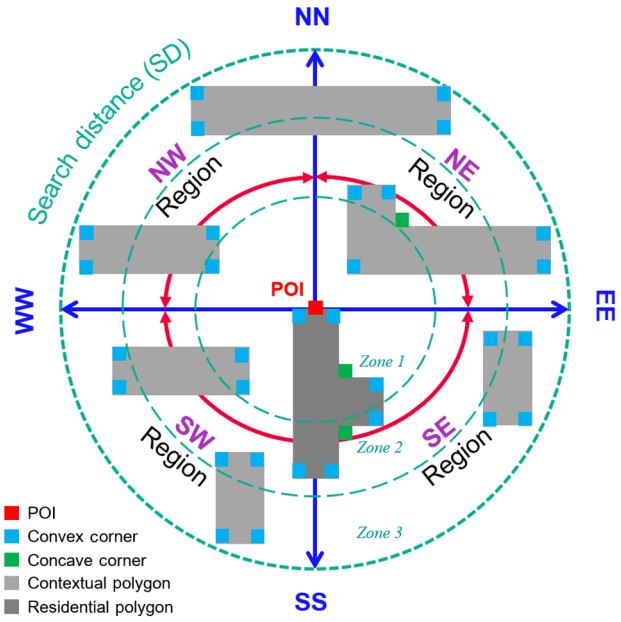


Fig. 3 Layout POI and surrounding context of patterns.

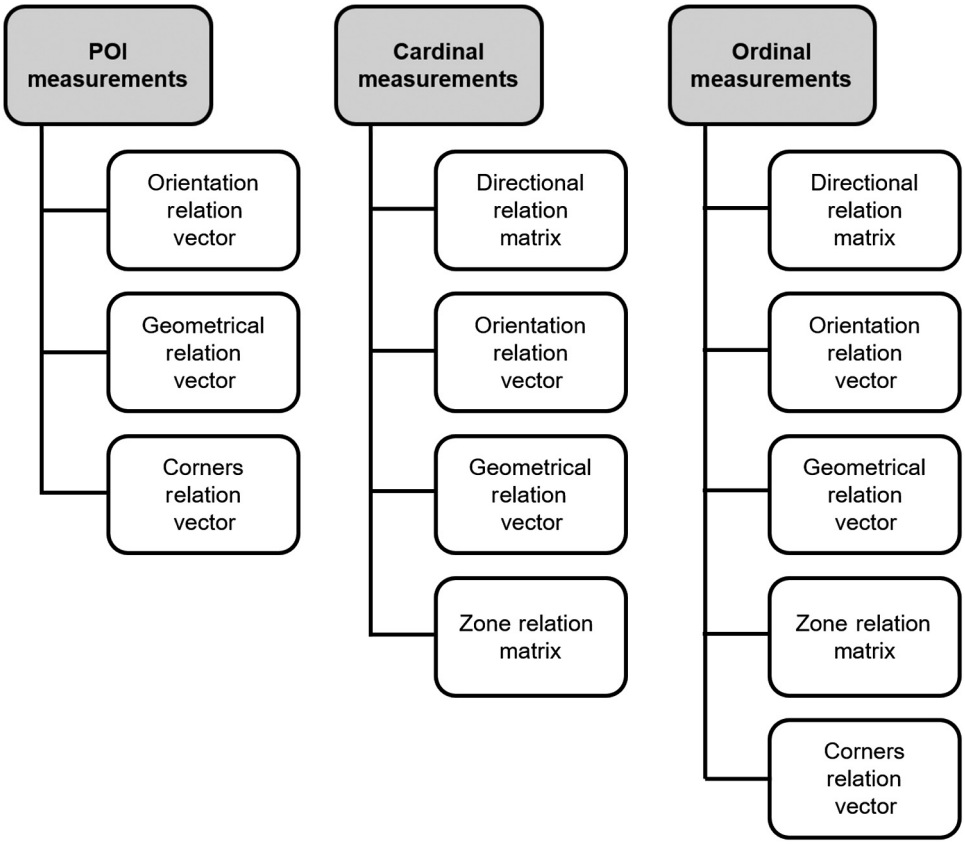


Fig. 4 GPS geometrical measurements constructors.

As shown in Fig. 4, the combinational embodiments structured by these constructors form three categories of descriptive geometrical measurements:

- **POI measurements** for the POI and the residential polygon, which is the polygon where the POI is placed. If the POI is placed on a polygon, then the residential polygon measurements is calculated, and if the POI is placed in a space, it is not calculated.
- **Cardinal measurements** for contextual patterns in east, north, west, and south directions.
- **Ordinal measurements** for contextual patterns in north-east, north-west, south-west, and south-east directions.

2.3 DGKs and GPS Features Vector

DGKs are the core geometrical measuring functions applied per direction in each zone. These measuring functions use the constructors shown in Fig. 4 to calculate POI, cardinal, and ordinal measurements, as explained in the previous section. These core measuring functions are implemented using Calibre SVRF commands that physically extract geometrical measured features. For example, the EXTERNAL command is used to measure distance-related features, and the INTERNAL command is used to measure width-related features. Directional, orientation, corners, and zone relations are implemented using DFM PROPERTY and DFM Functions operations. Then, the measured geometrical features by DGKs are concatenated using DFM PROPERTY commands in a single feature vector entry per each POI. Accordingly, the POI feature vector can be represented by the following equation to reflect the three categories of measurements shown in Fig. 4:

Features vector of the POI and its surrounding context of patterns are given as

$$V(F_{\mathrm{POI}}) = \langle f_P \rangle \oplus \sum_{Z \in \{z_1, z_2, z_3\}}^{\varphi_{\mathrm{SD}}} \langle f_{\mathrm{CC}} \rangle \oplus \langle f_{\mathrm{CO}} \rangle, \tag{1}$$

where

- $V(F_{\mathrm{POI}})$ represents a vector of all features of the POI and its surrounding context of patterns.
- $\langle f_P \rangle$ represents POI measurements, which is a features vector of geometrical measurements for the POI edge and the residential polygon, and it is decomposed into a dimensional vector $\langle D_P \rangle$ (containing numerical continuous data for five features) and a topological vector $\langle T_P \rangle$ (containing discrete finite-states data for eight features).

$$\langle f_P \rangle = \langle D_P \rangle \oplus \langle T_P \rangle$$

- φ_{SD} represents the SD shown in Fig. 3, which is empirically represented by the optical diameter (OD) of the litho system. SD can be configured to fit different applications.
- z_1, z_2, z_3 represent three measurement zones, as shown in Fig. 3. These zones are centered by the POI and extend to the surrounding context of patterns within the OD distance. Same as SD, measurement zones can be configured to fit different applications.
- $\langle f_{\mathrm{CC}} \rangle$ represents the cardinal measurements, which is the features vector of the geometrical measurements in the cardinal directions for the contextual patterns around the POI.
- $\langle f_{\mathrm{CO}} \rangle$ represents ordinal measurements, which is the features vector of the geometrical measurements in the ordinal directions for the contextual patterns around the POI.

Both $\langle f_{\mathrm{CC}} \rangle$ and $\langle f_{\mathrm{CO}} \rangle$ can be decomposed into $\{D_{\mathrm{CC}}\}$, $\{T_{\mathrm{CC}}\}$ and $\{D_{\mathrm{CO}}\}$, $\{T_{\mathrm{CO}}\}$, respectively. Where $\{D_{\mathrm{CC}}\}$, $\{D_{\mathrm{CO}}\}$ represent matrices of dimensional features for the contextual patterns in the cardinal and ordinal directions with the size of (7 features × 4 directions × 3 zones) and (10 features × 4 directions × 3 zones), respectively, $\{T_{\mathrm{CC}}\}$ and $\{T_{\mathrm{CO}}\}$ represent matrices of topological features for the contextual patterns in the cardinal and ordinal directions with the size of (3 features × 4 directions × 3 zones) and (5 features × 4 directions × 3 zones), respectively.

This topological and dimensional features decomposition expands Eq. (1) into the following equation:

Expanded features vector with dimensional and topological features are given as

$$\begin{aligned} V(F_{\mathrm{POI}}) = {} & \langle \langle D_P \rangle \oplus \langle T_P \rangle \rangle \\ & \oplus \sum_{Z \in \{z_1, z_2, z_3\}}^{\varphi_{\mathrm{SD}}} \left\langle \left\langle \sum_{\mathrm{Dir} \in \{E,N,W,S\}} \{D_{\mathrm{CC}}\} \otimes \{T_{\mathrm{CC}}\} \right\rangle \right. \\ & \left. \oplus \left\langle \sum_{\mathrm{Dir} \in \{\mathrm{NE,NW,SW,SE}\}} \{D_{\mathrm{CO}}\} \otimes \{T_{\mathrm{CO}}\} \right\rangle \right\rangle, \end{aligned} \tag{2}$$

where operators $\{\ \}$ and $\otimes$ are matrix construction and concatenation operations, $\langle\ \rangle$ and $\oplus$ are vector construction and concatenation operations, and $\sum$ is an element-wise processing operation. These operations are incorporated using DFM PROPERTY commands, to generate the feature vector $V(F_{\mathrm{POI}})$.

2.4 GPS Topological and Dimensional Features

Based on Eq. (2) for $V(F_{\mathrm{POI}})$, the collected geometrical features of the POI and its surrounding context of patterns can be categorized into

- Topological features set $\langle T_P \rangle$, $\{T_{\mathrm{CC}}\}$, $\{T_{\mathrm{CO}}\}$ that refers to finite-state discrete data representing the topological aspects of the pattern structured by the POI, residential polygon, and the surrounding context of polygons, which represents the topological signature
- Dimensional features set $\langle D_P \rangle$, $\{D_{CC}\}$, $\{D_{CO}\}$ that refers to numerical continuous data that combine with the topological features to compose the pattern structure, representing the dimensional signature.

The GPS topological and dimensional signatures are unlike adaptive squish pattern signatures,[18,19] which are low-level bitmap descriptions of patterns where each bit represents a unique value by its own and requires further feature extraction. GPS signatures directly provide descriptive features that reflect actual geometrical (topological and dimensional) properties. Table 3 shows examples of GPS measured features with features description and feature types.

Table 3 Example of GPS measured features.

Feature name	Feature description	Feature type
POI_Width	Length of POI resident edge	Dimensional
POI_Orientation	Orientation of POI resident edge	Topological
Target_Total_Length	The resident polygon perimeter	Dimensional
Target_Facial_Border	The distance to the facial border from the POI edge	Dimensional
Target_[a]_Zone_[c]	Equals (1) if a contextual polygon exists in the corresponding cardinal direction and zone, otherwise (0)	Topological
Target_[a]_Distance_[c]	Distance from POI to the corresponding polygon	Dimensional
Target_[a]_Corner_Type_[c]	The type of the corresponding ordinal direction corner: (1) convex corner, (2) concave corner, and (3) no corner	Topological
Target_[b]_Distance[c]	Distance from POI to the corresponding ordinal corner	Dimensional

[a]Cardinal direction index, i.e., east, north, west, and south.
[b]Ordinal direction index, i.e., north-east, north-west, south-west, and south-east.
[c]Zone index, i.e., zone 1, zone 2, or zone 3

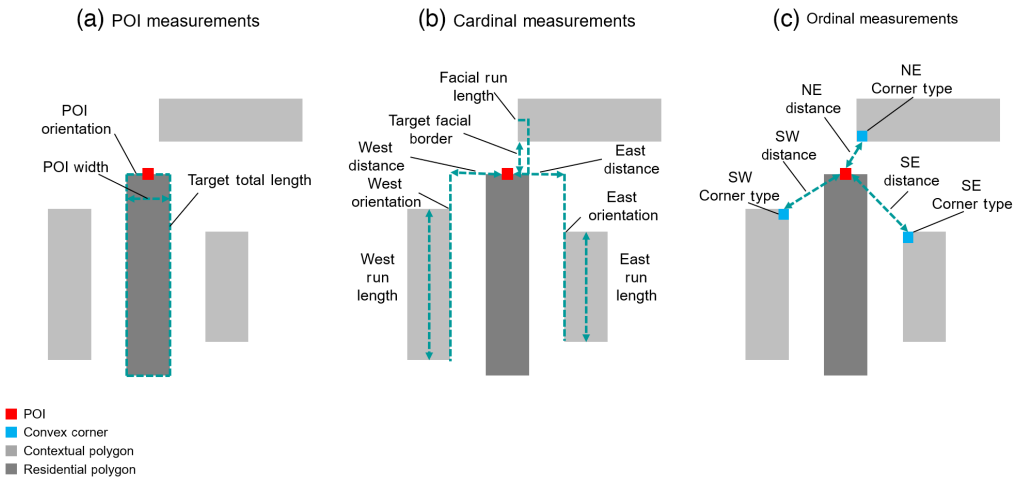


Fig. 5 Illustrative examples of GPS measured features. (a) POI measurements, (b) cardinal measurements, and (c) ordinal measurements.

Illustrative examples for the measured features are shown in Fig. 5 for the same pattern categorized as (a) POI measurements, (b) cardinal measurements, and (c) ordinal measurements. Further details about GPS features and DGKs implementation can be found in Ref. 30.

Metaphorically, GPS is an edge-based camera that snaps geometrical edge-based images for the POI and its surrounding context of patterns, rather than the conventional pixel-based or density-based images.

3 Simulation Flow and Data Harvesting

The test case used in this study is a fully connected industrial layout with electrical functionality and lithographic manufacturing recipes for a single exposure 28 nm technology node. We focus on BEOL interconnect layers stack V1-M2-V2, where M2 is a vertical two-dimensional metal layer with minimum line/space width of 40 nm, and the via layers (V1 and V2) are of size 40 nm.

3.1 POI Definition and Placement

The primary objective of this study is LEPB prediction for LEs that have a nearby via, as these are the most critical LEs in any design, and the most likely locations where a systematic defect can occur that results in partial or total metal-via disconnection. Accordingly, we set the following criteria to select POIs as LE tip with length $\leq$70 nm and has a via (either V1, V2, or both) placed within a distance $\leq$100 nm away from the LE tip.

3.2 Lithographic Manufacturing Simulation Flow

The conventional litho manufacturing simulation workflow is shown in Fig. 6 and the associated processed layers are shown in Fig. 7. The flow is used to precisely calculate the amounts of pullbacks and extensions at each LE, which will be used to train and test the ML models in the following sections.

The input design layout passed the physical verification sign-off process, and input design layers (V1-M2-V2) are shown in Fig. 7(a). The POI placement criteria stated in Sec. 3.1 is applied on a testcase of size 1 mm-by-1 mm, resulting in 248,798 POIs. The placed POIs are shown in Fig. 7(b). The layout M2 layer is then subjected to etch-biasing compensation to generate the retarget layer shown in Fig. 7(c). Hybrid SRAF insertion is used to place assist features using both (1) a model-based approach using calibrated litho model (optical and etching models) and (2) rule-based approach based on design dimensions. The OPC process is performed based on a litho model basis and applied on the placed SRAFs seeds and M2 retarget layers, including (1) process-window OPC, (2) SRAF print avoidance, and (3) mask rules constraints to maintain

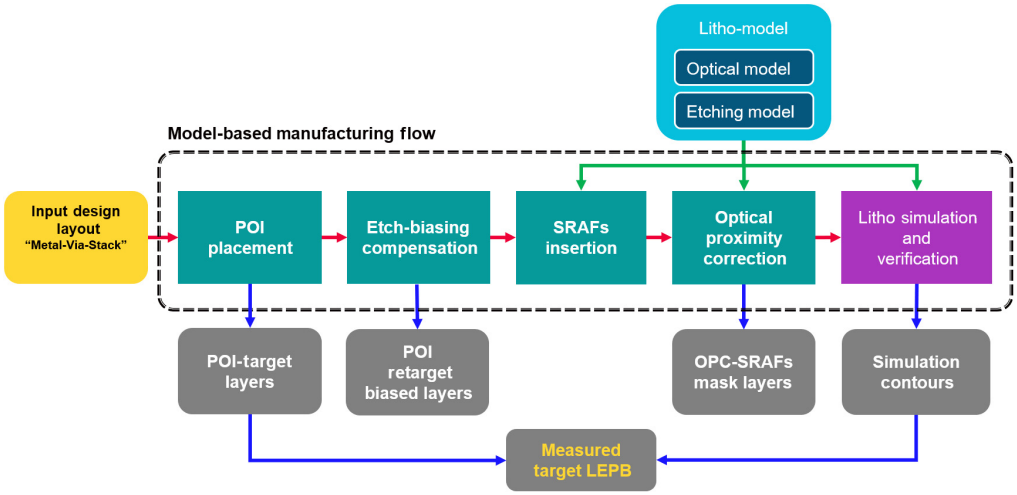


Fig. 6 Model-based lithographic manufacturing workflow to measure target LEPB.

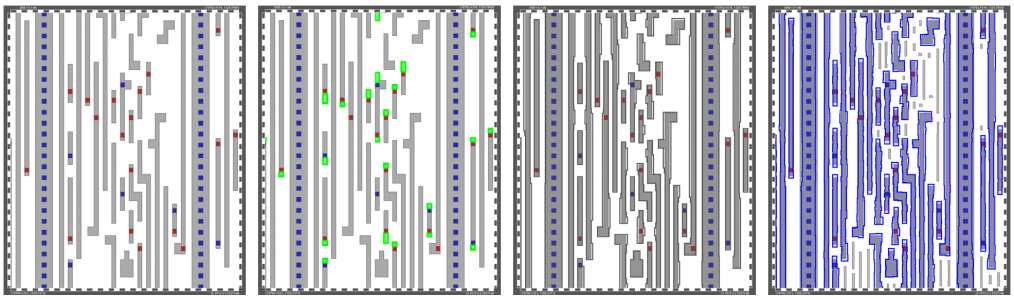
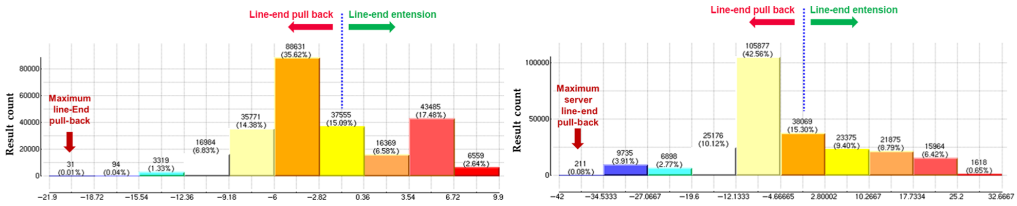


Fig. 7 Lithographic manufacturing workflow associated layers. (a) Origin drawn target V1-M2-V2, (b) placed POIs on selected line-ends, (c) etching biased retarget M2 layer, and (d) model-based OPC and SRAFs placement.



(a) Target LEPB amount (nm) (b) Target LEPB severity amount (%)

Fig. 8 Histograms of (a) target LEPB amount (nm) and (b) target LEPB severity amount (%).

a practical industrial approach. The placed SRAFs and OPC final masks are shown in Fig. 7(d). Generated masks (OPC and SRAFs) are used in model-based litho (optical and etching) simulation and verification to generate the final contours after etching the printed contours at nominal process window conditions. Target LEPB is measured between the final etched contours and M2 input target layer (original drawn) at POIs.

Figure 8(a) shows a histogram of target LEPB amounts (nm), where negative amounts refer to LE pullback and positive amounts refer to LE extensions. Although the maximum LEPB amount is 21.9 nm, the LEPB amount should not be viewed as a standalone value, but as a percentage of the TIP-to-VIA distance to reflect the severity of the pullback, as shown in Fig. 8(b). The maximum target LEPB severity is 42%, with a target LEPB amount of 21 nm and TIP-to-VIA distance of 50 nm.

3.3 Layout ML Features Harvesting

As explained in Sec. 3.2, the LEPB amount at each POI is measured through a conventional litho simulation flow, as shown in Fig. 6. To quantify the geometrical features provided by GPS, we integrated density pixels and density CCAS features as alternative comparative approaches. For consistency comparison between the three implemented approaches, we impose a POI-based analysis and center the SD of each approach around the POI and select adequate resolution within a comparable number of features. The optical system used in this study is a deep-ultraviolet system with an OD of 1.28 μm; accordingly, the SD centered by the POI should be 0.640 μm for all approaches. It should be highlighted that in typical OPC modeling, especially etching process considerations, the SD can be beyond the OD up to 25 μm or even 50 μm to account for long-range effects. However, for consistent comparisons, we set the SD approximately equal to the system's OD for all approaches.

3.3.1 *GPS features vector*

As shown in Eq. (2), the number of measurement zones has a direct impact on the length of the features vector produced by GPS. Increasing the number of measurement zones will accordingly increase the length of the features vector. In this study, we empirically set three measurement zones with equal steps covering the SD centered by the POI up to the OD. This yields 313 features representing the POI and its surrounding context of patterns. Figure 9 shows the distribution between topological and dimensional features, and their counts for: (1) POI and resident polygon features, (2) zone 1 features, (3) zone 2 features, and (4) zone 3 features.

GPS **313** features

Topological (104) features

Features set	T_{POI}	T_{Z_1}	T_{Z_2}	T_{Z_3}
Number of features	8	32	32	32

Dimensional (209) features

Features	D_{POI}	D_{Z_1}	D_{Z_2}	D_{Z_3}
Number of features	5	68	68	68

Fig. 9 GPS topological and dimensional features vectors and their counts.

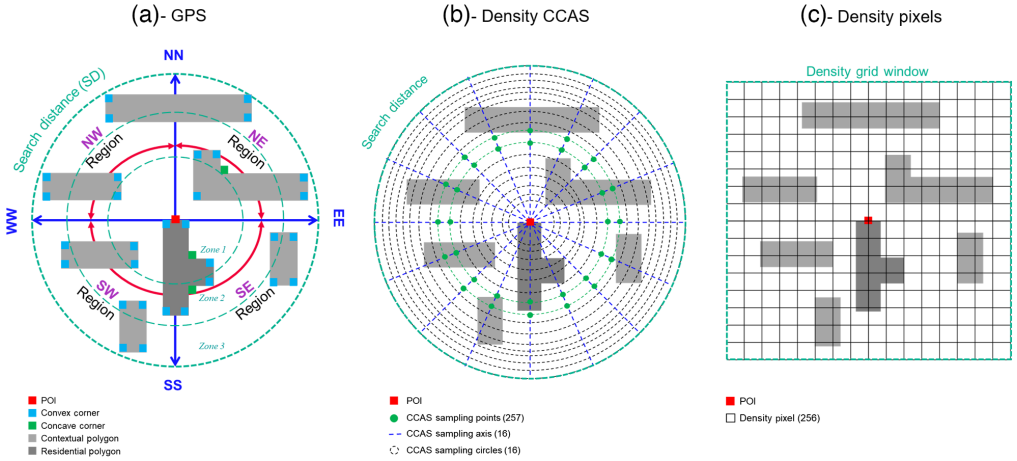


Fig. 10 Incorporated layout ML features: (a) GPS, (b) density CCAS, and (c) density pixels.

Although the number of edges and corners of the context around each POI is different, the extracted GPS features vector length is fixed for all POIs. Figure 10(a) shows the GPS representation of the POI and its surrounding context of patterns.

3.3.2 *Density CCAS features vector*

We implanted the density CCAS representation with 16 sampling circles equally distributed with a radius step of 40 nm (equal to target layer line/space width) covering the search area and 16 sampling axes with a radial angular step of $(\pi/8)$, as shown in Fig. 10(b). We approached sampling points based on 1 nm square pixel in a grid of 1280×1280 pixels covering the SD centered by the POI. This approach results in 16 sampling points for each sampling circle at the intersection points between circles and axes. Eventually, this sampling yields 257 density CCAS features.

3.3.3 *Density pixels features vector*

The density grid pixel approach is shown in Fig. 10(c), where the density grid square window of size 1.280 μm is divided into 16×16 grid pixels of squares with a size of 80 nm (equal to the target layer pitch), and density values are calculated inside each pixel, which eventually yields 256 density pixels features.

3.3.4 *Features extraction evaluations*

We used five test cases belonging to the same manufacturing technology, which was previously described in this section, to quantify the runtime and memory consumption of SVRF scripts that extract features using GPS, density CCAS, and density pixels. Table 4 shows each test case's area (in mm^2) and the number of placed POIs.

All runs were exclusively executed on a single machine with 32 CPUs, Intel® Xeon®, 2.90 GHz, with a system memory of 384 GBs, and using the same CALIBRE running options. For each evaluation run, we report the runtime, the peak memory consumption during the execution of the run, the CPU h/mm^2, and the CPU h/1 M POIs to reflect the needed processing time

Table 4 Features extraction evaluation test cases.

Test case	A	B	C	D	E
Area (mm^2)	1.416	2.214	3.340	2.408	1.321
Number of POIs	248,597	266,935	540,415	443,919	676,121

per 1 million POIs within the processed area (mm^2). Tables 5–7 show the evaluations of GPS, density CCAS, and density pixels feature extraction, respectively.

The average CPU h/mm^2 of GPS is less than the density CCAS and density pixels, and much less than that required for model-based litho simulation, which is estimated by Shin et al. in Ref. 11 to be >100 CPU h/mm^2. In addition, the peak memory required during the run execution by GPS is less than that required for the density CCAS and the density pixels

To predict the target LEPB, ML features representing the POI and its surrounding context are not sufficient on their own. We add localized features, which are layout measurements specific to

Table 5 GPS features extraction runtime and peak memory consumption.

Test case	Real time (H)	Peak memory (KB)	CPU (h/mm^2)	CPU (h/1M POIs)
A	0.05	1670	1.13	6.44
B	0.06	1805	0.87	7.19
C	0.09	2102	0.86	5.33
D	0.07	1875	0.93	5.05
E	0.11	2665	2.66	5.21
			Avg. CPU (h/mm^2)	**Avg. CPU (h/1M POIs)**
			1.29	**5.84**

Table 6 Density CCAS features extraction runtime and peak memory consumption.

Test case	Real time (H)	Peak memory (KB)	CPU (h/mm^2)	CPU (h/1M POIs)
A	0.30	2464	6.78	38.67
B	0.31	2587	4.48	37.16
C	0.64	3146	6.13	37.90
D	0.52	2667	6.91	37.48
E	0.78	3708	18.89	36.92
			Avg. CPU (h/mm^2)	**Avg. CPU (h/1M POIs)**
			8.64	**37.62**

Table 7 Density pixels features extraction runtime and peak memory consumption.

Test case	Real time (H)	Peak memory (KB)	CPU (h/mm^2)	CPU (h/1M POIs)
A	0.92	25,909	20.79	118.43
B	0.94	25,986	13.59	112.69
C	2.44	55,606	23.38	144.48
D	1.86	46,219	24.72	134.08
E	3.35	72,503	81.15	158.55
			Avg. CPU (h/mm^2)	**Avg. CPU (h/1M POIs)**
			32.72	**133.65**

the LEPB prediction problem, such as line-end width, tip to via distance, and via status (a categorical feature that states if the via is a top or bottom via).

4 Machine Learning Modeling Flow

The objective of the ML data modeling flow is to calibrate ML models to predict target LEPB at POI locations based on various features sets. As discussed in Sec. 2, GPS provides direct one-to-one directional geometrical features describing a POI and its surrounding context of polygons in the layout, which paves the way toward using predefined architected off-the-shelf ML models.

4.1 Decision-Trees and IC Layouts

The decision tree (DT) regressor is a supervised ML algorithm that aims to model a continuous prediction target by building a tree structure consisting of decision nodes and branches. A simple explanation of DT is shown in Fig. 11. The decision node contains a subset of the input features, and the decision branches are rules applied to the node's features. Based on the number of features in a parent node and applied decision rules, subsequent child nodes are created through splits. Each of the child nodes inherits a subset of the parent node features with more pivotal branches. In the DT classifier, splits are based on information gain and entropy, whereas in the DT regressor, the split decision is based on information gain and error measurement criteria to evaluate the deviation from the prediction target.

As we explained in Sec. 2.1, the IC creation process is constrained by many rules derived by place and route tools, DRC and DFM rules, etc., all of which imply constraints in polygon width, spacing, direction. Each of these constraints has boundaries and can be constructed in the form of a DT model. Dawar et al.[17] charted an example of a DT model for a simple constrained rule of tip-to-tip. We reintroduce this example in Fig. 12, with slight modifications to explain how a simple IC constrained rule construction can be approached through DT models.

This analogy can be extended to GPS features as well, as the features directly represent directional topological and dimensional measurements for a POI and its surrounding context

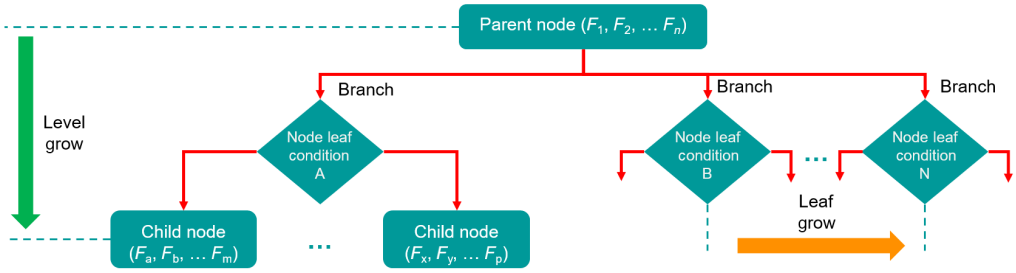


Fig. 11 DT growing with level and leaf.

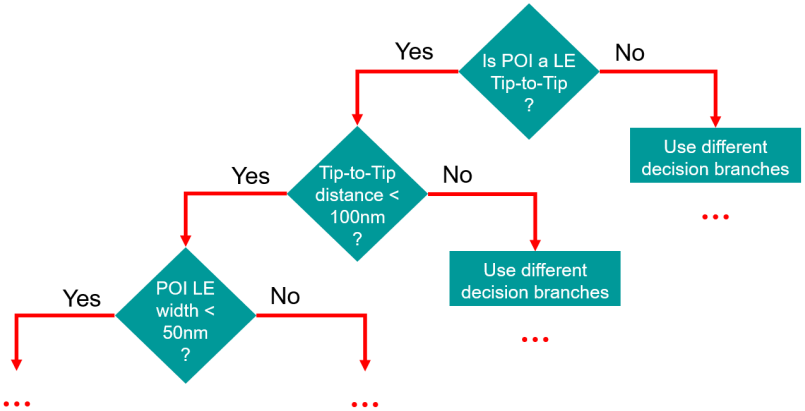


Fig. 12 Example of simple DT with simple constrained rule.

of patterns and can be constructed in the form of a DT model. In addition, DT models have been incorporated with density pixels features,[14] aerial image, and geometrical features[17] for litho HS detection applications, and with CCAS features for SRAF placement.[23]

Initially, we selected random forest (RF)[35] and adaptive-boosting RF (AdaBoostRF)[36] ML models from Scikit-learn ensemble regressors.[37] However, we faced a limitation with the RF model, as it does not natively support categorical features, and GPS topological features are categorical features. We tried to use one-hot-encoding (OHE)[38] to encode GPS topological features, but this approach significantly increased the number of features, because based on the number of values that each categorical feature can take, OHE extends each categorical feature into several extended encoding features with low cardinality, which tends to create unbalanced biased trees.[39] Increasing the number of features mandated the usage of a large number of estimators that roughly fit with input data and make the model more vulnerable to overfitting, especially with AdaBoostRF. Moreover, we discovered that the GPS topological features were located at the bottom of the feature-importance list produced by the RF. This low priority meant the model did not make significant use of the benefits of GPS topological features, due to the lack of native support of categorical features by the RF regressor combined with the low cardinality of GPS topological features after OHE. Other encoding techniques such as target encoding[40] could be used, but they will make the ML modeling process subject to data leakage,[29] whereas the lack of native categorical features support in the RF model remains a challenge. These limitations motivated us to employ a LightGBM[41] model.

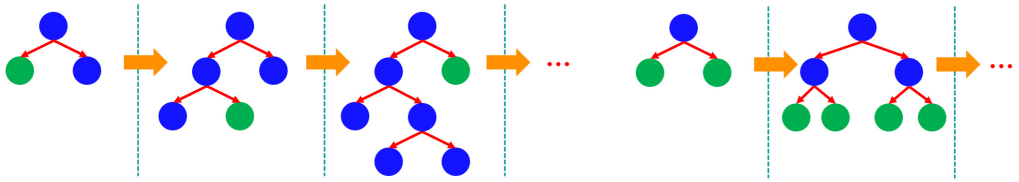
4.2 LightGBM Regressor

The LightGBM[41] regressor is a gradient-boosting DT-supervised ML model that natively supports integer-encoded categorical features by finding the optimal splits over categorical features. A sorted histogram for each categorical feature's values is used to partition the categories into subsets,[41] and a reduced complexity algorithm[42] is used to find the optimum partitions. Then, the optimal split points are located on the sorted histogram based on the training objectives at each split.[39] LightGBM provides two main advantages: (1) gradient-based one-side sampling, which samples the training data on a gradients-basis, and (2) exclusive feature bundling through a greedy algorithm, which provides better performance than OHE.[41] One of the main advantages of LightGBM is that it is fast[41] compared to other DT models, such as XGBoost.[43] This speed is due primarily to growing fewer trees through optimized leaf-wise splits, rather than level-wise growth, as shown in Fig. 13,[44] which significantly speeds up the training process. LightGBM also supports ensemble learning, where a series of base DT models are trained in sequence on versions of the training data, with each model achieving improved learning based on the prediction error (PE) in the previous model. The final prediction is produced through voting across all base models.

A list of LightGBM model parameters that we encountered by tuning throughout our evaluation experiments is found in Table 8.

4.3 GPS Topological Signatures Features

As explained in Sec. 2.2, GPS topological features are categorical features that take finite-state discrete data values to represent the topological aspects of the pattern. The combination of all topological features within a measurement zone is the zone's topological signature, which is a dimensionless integer-encoded number that refers to a specific constellation of topological combinations within the zone. Patterns that have the same topological signature of a given zone have



(a) Tree growth through leaf-wise splits (LightGBM) (b) Tree growth through level-wise splits

Fig. 13 DT growing through splits: (a) tree growth through leaf-wise splits (LightBGM) and (b) tree growth through level-wise splits.

Table 8 LightGBM model parameters.

LightGBM parameters	Description	Parameter value
boosting_type	Select the boosting type	"gbdt": gradient boosting DT
objective	Select the model's objective	"regression": regression with L2 loss
tree_learner	Selects the tree learner	"serial": single machine tree learner
num_leaves	Maximum tree leaves for base learners	128
max_depth	Maximum tree depth for base learners	−1: means no limit
learning_rate	Boosting learning rate	0.1
n_estimators	Number of boosted trees to fit	100
min_child_samples	Minimum number of data needed in a child (leaf)	20
min_child_weight	Minimal sum hessian in one leaf	0.001
max_cat_threshold	limit number of split points considered for categorical features	64

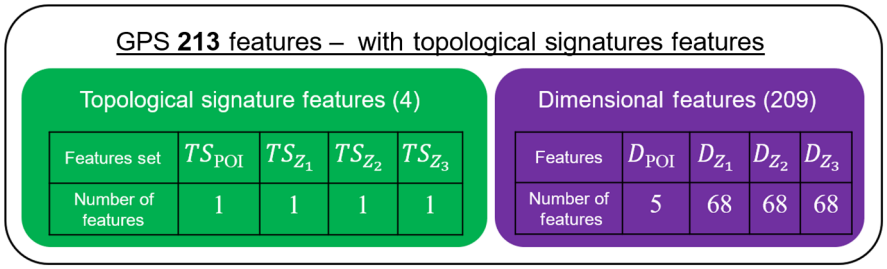


Fig. 14 GPS features with topological signature features.

the same topological aspects in this zone. Consequently, the patterns that have the same topological signatures in all zones have the same topological feature values, and accordingly the same topological aspects across all measurement zones, although they can have the same or different dimensional features. Instead of passing all 104 GPS topological features to the LightGBM model, we use only four topological signatures features for (1) POI and resident polygon, (2) zone 1, (3) zone 2, and (4) zone 3, as shown in Fig. 14. This reduces the GPS features vector length to 213 features instead of 313 features, and, as highlighted in Sec. 4.1, the LightGBM model handles categorical features with integer-encoding values without numerical biasing.

4.4 Experimental Evaluations

Besides collecting measured target LEPB at each POI, the following features are collected: (1) localized features, (2) GPS features, (3) density-pixels features, and (4) density-CCAS features. Three experimental sets of features are formulated, as shown in Table 9. Each of the experimental feature sets is trained with LightGBM to predict target LEPB.

Table 9 ML experimental evaluations features sets.

Features set	Incorporated features	Number of features
DENSITY_PIXELS	Density pixels features (256) + (5) localized features	261
DENSITY_CCAS	Density CCAS features (257) + (5) localized features	262
GPS	GPS features (213) + (5) localized features	218

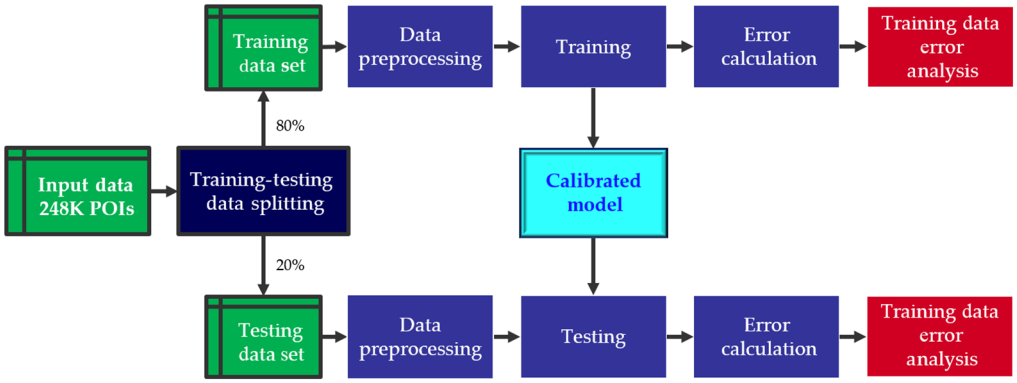


Fig. 15 ML data modeling flow.

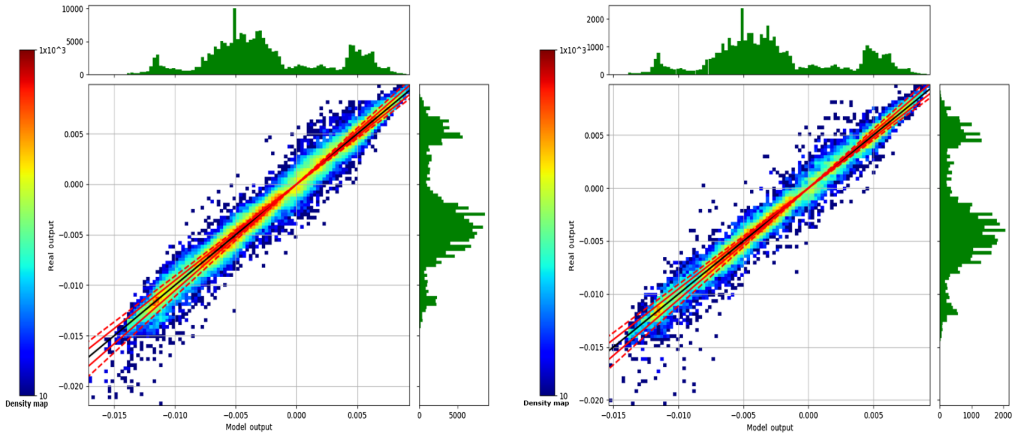
The ML modeling flow shown in Fig. 15 starts by dividing the input data randomly into training and testing data sets, with allocations of 80% and 20%, respectively, to maintain healthy testing on unseen data by the ML models during the training phase to avoid data leakage.[29] The data preprocessing step normalizes input features and target prediction before starting the training phase, and the same preprocessing is applied to the testing data set. The training phase is then triggered, producing training predictions, and the calibrated ML model is passed to the testing block, which produces testing predictions. The error calculation and error analysis steps (which are explained in detail in the next section) are applied to the training and testing predictions and the results compared against the actual measured target LEPBs to determine the quality of the training and testing.

5 Experimental Evaluations Results

To aid navigation through the results of the various experimental features sets listed in Table 9 trained with LightGBM model, we generated a heat-map scatterplot for each set and each model's training and testing predictions. In these plots, the model predictions are on the X axis with a horizontal top histogram, whereas the actual measured target LEPB values are on the Y axis with a vertical left histogram. A solid black 45 deg line is added to mark optimal predictions matching actual targets, and solid and dotted red lines are added to mark 5% and 10% PEs, respectively. The heat map reflects data points density where the color code starts with blue points representing 10 data points, then gradually changes colors toward red points representing 1K data points.

5.1 ML Model Error Quantifiers

The PE at each data point is defined as $\mathrm{PE}_i = (y_i - f(x_i))$, where i represents the data point index, y_i represents the actual measured target value at the data point, and $f(x_i)$ represents the value predicted by the model using a features vector x_i. To quantify the quality of the incorporated models' training and testing predictions across the experimental features sets, we extract the following error quantifiers: (1) predictions coefficient of determination (R^2), (2) root mean squared (RMS) of the PEs, and (3) standard deviation of prediction residuals error wideness distances (δ_{EWD}). R^2 is defined as $1 - (R_{ss}/T_{ss})$, where R_{ss} is the residual sum of squares, T_{ss} is the total sum of squares, and R^2 indicates the quality of model training and testing processes, with values closer to "1" reflecting higher quality. The RMS of predictions errors is defined as $\mathrm{SQRT}(\frac{1}{n} * \sum_i^n (\mathrm{PE}_i)^2)$, where n is the total number of data points, and the result indicates the error in the predicted target, with values closer to "0" reflecting better predictions. To indicate the confinement of predictions around the 45 deg line, we measure δ_{EWD}, which is defined as $\mathrm{SQRT}(\frac{1}{n} * \sum_i^n (\overline{\mathrm{PE}}_i - \mu_{\overline{\mathrm{PE}}_i})^2)$, where $\overline{\mathrm{PE}}_i = |\mathrm{PE}_i|/\sqrt{2}$ and $\mu_{\overline{\mathrm{PE}}_i}$ is the mean of $\overline{\mathrm{PE}}_i$ values. We use $\overline{\mathrm{PE}}_i$ instead of PE_i, which has a mean close to zero and produces a standard deviation value close to the RMS value. The values of δ_{EWD} closer to "0" reflect more confinement predictions around the 45 deg line.



(a) – LightGBM with density-pixels training (b) – LightGBM with density-pixels testing

Fig. 16 LEPB predictions results using LightGBM and density-pixels features set: (a) LightGBM with density-pixels training and (b) LightGBM with density-pixels testing.

Table 10 LEPB PE quantifiers using LightGBM and density-pixels features set.

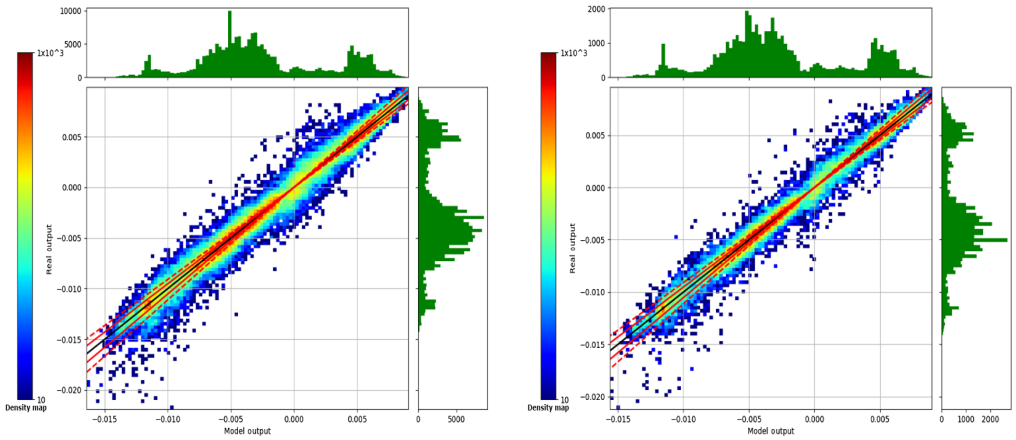
Learning phase	RMS (nm)	δ_{EWD} (nm)	R^2 (%)
Training	0.538	0.284	98.89
Testing	0.616	0.338	98.53

5.2 Modeling LEPB Using Density-Pixels Features

The training and testing LEPB prediction results (μm) of the LightGBM model with the DENSITY_PIXELS features set are shown in Figs. 16(a) and 16(b), respectively. The error quantifiers are listed in Table 10, where the prediction RMS error values are <1 nm, with no observations of overfitting in the testing results, δ_{EWD} results show confined predictions within 0.5 nm, and R^2 results show an overall good fitting of LEPB predictions to the input features set.

5.3 Modeling LEPB Using Density-CCAS Features

The training and testing LEPB prediction results (μm) of the LightGBM model with a DENSITY_CCAS features set are shown in Figs. 17(a) and 17(b), respectively, with close

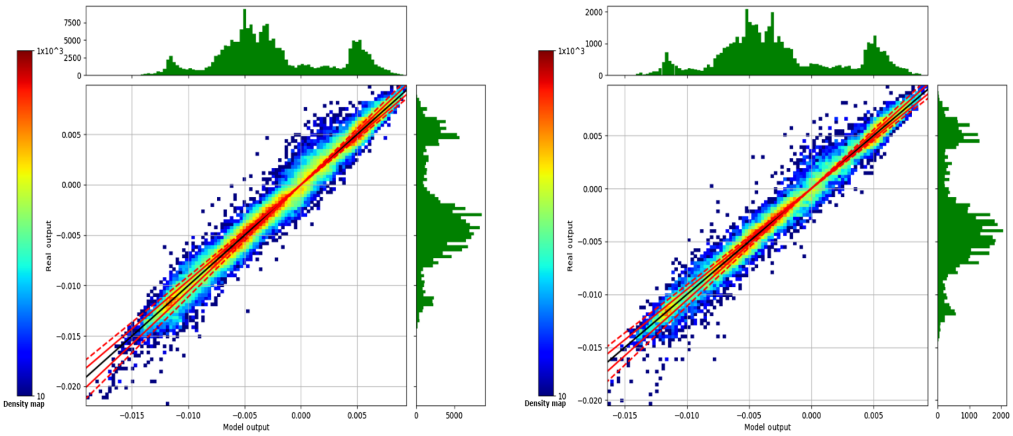


(a) – LightGBM with density-CCAS training (b) – LightGBM with density-CCAS testing

Fig. 17 LEPB predictions results using LightGBM and density-CCAS features set: (a) LightGBM with density-CCAS training and (b) LightGBM with density-CCAS testing.

Table 11 LEPB PE quantifiers using LightGBM and density-CCAS features set.

Learning phase	RMS (nm)	δ_{EWD} (nm)	R^2 (%)
Training	0.589	0.310	98.66
Testing	0.645	0.350	98.40



(a) – LightGBM with GPS training (b) – LightGBM with GPS testing

Fig. 18 LEPB predictions results using LightGBM and GPS features set: (a) LightGBM with GPS training and (b) LightGBM with GPS testing.

Table 12 LEPB PE quantifiers using LightGBM and GPS features set.

Learning phase	RMS (nm)	δ_{EWD} (nm)	R^2 (%)
Training	0.569	0.296	98.75
Testing	0.640	0.343	98.41

behavior correlation to the DENSITY_PIXELS features set. The error quantifiers are listed in Table 11, where the prediction RMS error values are <1 nm, with no observations of overfitting in the testing results, δ_{EWD} results show confined predictions within 0.5 nm, and R^2 results show an overall good fitting of LEPB predictions to the input DENSITY_CCAS features set.

5.4 Modeling LEPB Using GPS Features

The training and testing LEPB prediction results (μm) of the LightGBM model with GPS features set are shown in Figs. 18(a) and 18(b), respectively, with very similar behavior to the DENSITY_PIXELS and DENSITY_CCAS features set. The error quantifiers are listed in Table 12, where the prediction RMS error values are <1 nm, with no observations of overfitting in the testing results, δ_{EWD} results show confined predictions within 0.5 nm, and R^2 results show an overall good fitting of LEPB predictions to the input GPS features set.

5.5 LightGBM Features Reduction

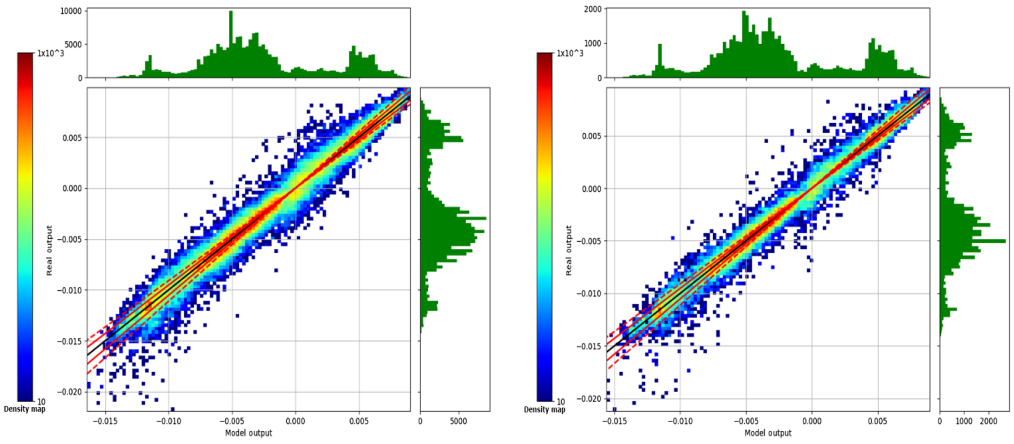
As we observe from the previous results, DENSITY_PIXELS, DENSITY_CCAS, and GPS features sets return very similar modeling results. Accordingly, from a representation efficiency point of view, all three approaches provide good fitting results with good prediction accuracy. However, are the features used by each representation approach as efficient as possible, or there is a redundancy that can be eliminated? LightGBM can produce a features importance list based on the number of splits in which each feature has been used. The more splits obtained, the more important the feature is. This list can be presented in descending order, where the most important

Table 13 Features reduction based on LightGBM splits.

Features set	Full set features count	Zero splits features count	Reduced set features count	Features reduction (%)
DENSITY_PIXELS	261	0	261	0
DENSITY_CCAS	262	13	249	~ −5
GPS	218	15	203	~ −7

features display at the top and the least important features at the bottom. Features with zero splits were not employed by the model in any splits and can be safely removed without affecting the modeling behavior or degrading the accuracy. Table 13 summarizes the feature reduction in each representation approach based on the LightGBM model feature importance by splits.

DENSITY_PIXELS has no zero split features, indicating that for LEPB modeling through a regression-based ML modeling, a high-resolution density pixel is needed. Moreover, this information aligns with the fact that the information gain in each pixel is quite high, and each pixel is employed efficiently by the model, with no possible further feature reductions. DENSITY_CCAS has 13 features with zero splits (−5%), aligning with the fact that CCAS has a type of information redundancy, as highlighted by Geng et al. in Ref. 24, since a dense number of sampling points implies circles and axes closer to each other, which accordingly implies redundancy. Figure 19 shows the training and testing LEPB prediction results (μm), and Table 14 shows the PE quantifiers after applying features reduction to the DENSITY_CCAS features set. These results are almost identical to the results in Fig. 17 and Table 11 using the full set of DENSITY_CCAS features. GPS has 15 features with zero splits (−7%) and all the reduced features are dimensional, which means these dimensional properties features were not important for LEPB modeling. Figure 20 shows the training and testing LEPB prediction results

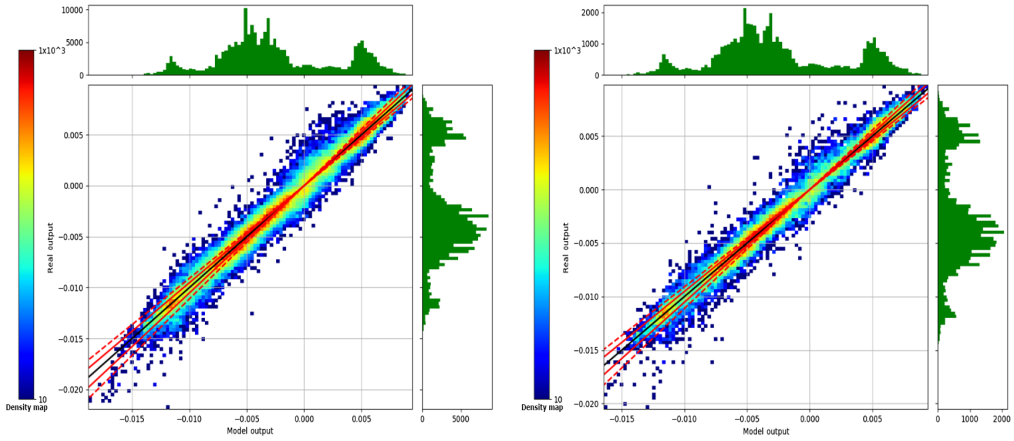


(a) – LightGBM with CCAS reduced features training (b) – LightGBM with CCAS reduced features testing

Fig. 19 LEPB predictions results using LightGBM and Density-CCAS reduced features set: (a) LightGBM with CCAS reduced features training and (b) LightGBM with CCAS reduced features testing.

Table 14 LEPB predictions error quantifiers using LightGBM and density-CCAS reduced features set.

Learning phase	RMS (nm)	δ_{EWD} (nm)	R^2 (%)
Training	0.590	0.310	98.66
Testing	0.640	0.350	98.40



(a) – LightGBM with GPS reduced features training (b) – LightGBM with GPS reduced features testing

Fig. 20 LEPB predictions results using LightGBM and GPS reduced features set: (a) LightGBM with GPS reduced features training and (b) LightGBM with GPS reduced features testing.

Table 15 LEPB predictions error quantifiers using LightGBM and GPS reduced features set.

Learning phase	RMS (nm)	δ_{EWD} (nm)	R^2 (%)
Training	0.571	0.297	98.74
Testing	0.643	0.344	98.40

(μm), and Table 15 shows the PE quantifiers after applying features reduction to the GPS features set. These results are almost identical to the results in Fig. 18 and Table 12 using the full set of GPS features.

As we can observe from the results in Fig. 17, Table 11, Fig. 19, and Table 14 for DENSITY_CCAS representation, and Fig. 18, Table 12, Fig. 20, and Table 15 for GPS representation, there is almost no degradation in either modeling accuracy or error quantifiers after applying the features reduction.

Table 16 Summary of features counts, features utilization, and predictions error quantifiers.

Feature set	Features count	Features utilization (%)	Training phase			Testing phase		
			RMS (nm)	δ_{EWD} (nm)	R^2 (%)	RMS (nm)	δ_{EWD} (nm)	R^2 (%)
Full features sets								
Density pixels	261	100	0.538	0.284	98.89	0.616	0.338	98.53
Density CCAS	262	100.38	0.589	0.310	98.66	0.645	0.350	98.40
GPS	218	83.52	0.569	0.296	98.75	0.640	0.343	98.41
Reduced features sets								
Density CCAS	249	95.40	0.590	0.310	98.66	0.640	0.350	98.40
GPS	203	77.78	0.571	0.297	98.74	0.643	0.344	98.40

Table 16 summarizes: (1) features counts, (2) features utilization referenced to density pixels (since it is the only features set that was not reduced), and (3) PE quantifiers of the LightGBM model. The error quantifiers are very similar in all the evaluation experiments, which indicates good model fitting to the LEPB predictions, and after features reduction, we can see that GPS features produced almost the same modeling quality (training: RMS $= 0.571$ nm, $\delta_{\mathrm{EWD}} = 0.297$ nm, and $R^2\% = 98.74\%$, and testing: RMS $= 0.643$ nm, $\delta_{\mathrm{EWD}} = 0.344$ nm, and $R^2\% = 98.40\%$) with -22.22% fewer features compared to the full features set of density pixels approach, and -22.26% fewer features compared to the full features set of the density CCAS approach.

5.6 Further Discussion

In this study, we presented LEPB modeling using GPS direct one-to-one mapping geometrical features. However, GPS features are not limited only to the LEPB problem and can be incorporated further to model other systematic defects, such as pinching and bridging.

6 Conclusions

In this paper, we presented a novel edge-based engine (GPS) that extracts direct one-to-one directional geometrical features describing a POI and its surrounding context of patterns in the layout, with feature extraction runtime less than density pixels and density CCAS approaches. We also presented efficient LEPB predictions using the LightGBM ML model with density pixels, density CCAS, and GPS as IC layout pattern representation, with a prediction RMS error <1 nm. We employed LightGBM features importance by splits to conduct feature reductions on the used approaches. The reduced features of GPS produced almost the same modeling quality (training: RMS $= 0.571$ nm, $\delta_{\mathrm{EWD}} = 0.297$ nm, and $R^2\% = 98.74\%$, and testing: RMS $= 0.643$ nm, $\delta_{\mathrm{EWD}} = 0.344$ nm, and $R^2\% = 98.40\%$) with -22.22% fewer features compared to the full features set of the density pixels approach, and -22.26% fewer features compared to the full features set of the density CCAS approach. Compared to litho simulations, the obtained calibrated models can be used to provide fast and accurate predictions of the amounts of pull-back or extensions introduced at LEs near vias, eliminating a major contributor to systematic IC yield loss.

Code, Data, and Materials Availability

The data utilized in this study were obtained from Siemens EDA and contains company proprietary information.

Acknowledgments

The authors would like to sincerely thank Shelly Stalnaker for the editorial assistance in this paper.

References

1. G. Huang et al., "Machine learning for electronic design automation: a survey," *ACM Trans. Design Autom. Electron. Syst.* **26**(5), 40 (2021).
2. D. Ding et al., "AENEID: a generic lithography-friendly detailed router based on post-RET data learning and hotspot detection," in *Proc. ACM/IEEE Des. Autom. Conf. (DAC'11)*, ACM, pp. 795–800 (2011).
3. D. Ding et al., "High performance lithographic hotspot detection using hierarchically refined machine learning," in *16th Asia and South Pac. Des. Autom. Conf. (ASP-DAC 2011)*, pp. 775–780 (2011).
4. Z. Xie et al., "RouteNet: routability prediction for mixed-size designs using convolutional neural network," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD 2018)* (2018).
5. R. Liang et al., "DRC hotspot prediction at sub-10nm process nodes using customized convolutional network," in *Proc. ACM Int. Symp. Phys. Des. (ISPD'20)* (2020).
6. O. Ronneberger, P. Fischer, and T. Brox, "U-Net: convolutional networks for biomedical image segmentation," *Lect. Notes Comput. Sci.* **9351**, 234–241 (2015).
7. P. Selvam et al., "Deep learning-based hotspot prediction of via printability in process window corners," *Proc. SPIE* **11614**, 116140X (2021).
8. J.-Y. Wuu et al., "Rapid layout pattern classification," in *IEEE/ACM Asia and South Pac. Des. Autom. Conf. (ASPDAC)*, pp. 781–786 (2011).

9. L. Wang et al., "Retargeting-aware design for manufacturability (DFM) checks using machine learning," *Proc. SPIE* **12052**, 1205211 (2022).
10. H. Yang et al., "Imbalance aware lithography hotspot detection: a deep learning approach," *Proc. SPIE* **10148**, 1014807 (2017).
11. M. Shin and J. Lee, "Accurate lithography hotspot detection using deep convolutional neural networks," *J. Micro/Nanolithogr. MEMS MOEMS* **15**(4), 043507 (2016).
12. Y. Jiang et al., "Efficient layout hotspot detection via binarized residual neural network ensemble," in *Proc. IEEE Computer-Aided Design of Integrated Circuits and Systems* (2021).
13. H. Yang et al., "Layout hotspot detection with feature tensor generation and deep biased learning," in *Proc. ACM/IEEE Des. Autom. Conf. (DAC'17)* (2017).
14. T. Matsunawa et al., "A new lithography hotspot detection framework based on AdaBoost classifier and simplified feature extraction," *Proc. SPIE* **9427**, 94270S (2015).
15. T. Matsunawa, S. Nojima, and T. Kotani, "Automatic layout feature extraction for lithography hotspot detection based on deep neural network," *Proc. SPIE* **9781**, 97810H (2016).
16. H. Zhang, B. Yu, and E. F. Y. Young, "Enabling online learning in lithography hotspot detection with information-theoretic feature optimization," in *IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, pp. 1–8 (2016).
17. R. Dawar et al., "Random forest-based robust classification for lithographic hotspot detection," *J. Micro/Nanolithogr. MEMS MOEMS* **18**(2), 023501 (2019).
18. H. Yang et al., "Detecting multi-layer layout hotspots with adaptive squish patterns," in *Proc. IEEE/ACM Asia and South Pac. Des. Autom. Conf. (ASPDAC'19)* (2019).
19. H. Yang et al., "Hotspot detection using squish-net," *Proc. SPIE* **10962**, 109620S (2019).
20. H. Yang et al., "GAN-OPC: mask optimization with lithography-guided generative adversarial nets," in *Proc. ACM/IEEE Des. Autom. Conf. (DAC'18)* (2018).
21. T. Matsunawa, B. Yu, and D. Z. Pan, "Optical proximity correction with hierarchical Bayes model," *J. Micro/Nanolithogr. MEMS MOEMS* **15**, 021009 (2015).
22. S. Choi, S. Shim, and Y. Shin, "Machine learning (ML)-guided OPC using basis functions of polar Fourier transform," *Proc. SPIE* **9780**, 97800H (2016).
23. X. Xu et al., "A machine learning based framework for sub-resolution assist feature generation," in *Proc. ACM Int. Symp. Phys. Des. (ISPD'16)* (2016).
24. H. Geng et al., "SRAF insertion via supervised dictionary learning," in *Proc. IEEE/ACM Asia and South Pac. Des. Autom. Conf. (ASPDAC'19)* (2019).
25. W. Ye et al., "LithoGAN: end-to-end lithography modeling with generative adversarial networks," in *Proc. ACM/IEEE Design Autom. Conf. (DAC'19)* (2019).
26. Y. Lin et al., "Machine learning for mask/wafer hotspot detection and mask synthesis," *Proc. SPIE* **10451**, 104510A (2017).
27. F. Pedregosa et al., "Scikit-learn: machine learning in Python," *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).
28. M. Abadi et al., "TensorFlow: a system for large-scale machine learning," in *12th USENIX Symp. Oper. Syst. Des. and Implement. (OSDI '16)*, 2016, https://tensorflow.org.
29. S. Kaufman, S. Rosset, and C. Perlich, "Leakage in data mining: formulation detection and avoidance," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discov. and Data Mining*, pp. 556–563 (2011).
30. H. Hegazy, A. Hamed, and O. Elsewefy, "Edge-based camera for characterizing semiconductor layout designs," US Patent 1017147B2 (2021).
31. C. Mack, *Fundamental Principles of Optical Lithography: The Science of Microfabrication*, Chapter 8, John Wiley & Sons, Ltd. (2007).
32. N. B. Cobb, "Fast optical and process proximity correction algorithms for integrated circuit manufacturing," PhD dissertation, University of California at Berkeley (1998).
33. W. Ciou et al., "Machine learning optical proximity correction with generative adversarial networks," *J. Micro/Nanopatterning, Mater. Metrol.* **21**(4), 041606 (2022).
34. H. Lee et al., "Thread scheduling for GPU-based OPC simulation on multi-thread," *Proc. SPIE* **10587**, 105870P (2018).
35. L. Breiman, "Random forests," *Mach. Learn.* **45**, 5–32 (2001).
36. H. M. Gomes et al., "A survey on ensemble learning for data stream classification," *ACM Comput. Surv.* **50**(2), 23 (2017).
37. Scikit-learn documentation, "Scikit-learn ensemble regressors," 2023, https://scikit-learn.org/stable/modules/ensemble.html#.
38. K. Potdar, T. S. Pardawala, and C. D. Pai, "A comparative study of categorical variable encoding techniques for neural network classifiers," *Int. J. Comput. Appl.* **175**(4), 7–9 (2017).
39. LightGBM documentation by Microsoft Corporation, 2023, https://lightgbm.readthedocs.io/en/latest/Features.html#optimal-split-for-categorical-features.

40. D. Micci-Barreca, "A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems," *ACM SIGKDD Explor. Newsl.* **3**(1), 27–32 (2001).
41. G. Ke et al., "LightGBM: a highly efficient gradient boosting decision tree," in *31st Conf. Neural Inf. Process. Syst. (NIPS 2017)*, Long Beach, California (2017).
42. W. D. Fisher. "On grouping for maximum homogeneity," *J. Am. Stat. Assoc.* **53**(284), 789–798 (1958).
43. T. Chen and C. Guestrin, "Xgboost: a scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. and Data Mining*, ACM, pp. 785–794 (2016).
44. LightGBM documentation by Microsoft Corporation, 2023, https://lightgbm.readthedocs.io/en/latest/Features.html#leaf-wise-best-first-tree-growth.

Ahmed Hamed Fathi Hamed received his BSc and MSc degrees in electronics and communications engineering from Cairo University in 2007 and 2015, respectively. He is currently pursuing his PhD in integrated circuits from Ain Shams University, Cairo, Egypt. He has many publications in the fields of RET, OPC, and DFM in addition to a U.S. patent. Ahmed is currently a senior product engineering manager at Calibre Semi-Manufacturing Solutions, Siemens EDA Egypt. He has been a member of SPIE since 2013.

Hazem Hegazy received his Bachelor of Science degree in 1997 followed by MSc degree in 2003 and PhD in 2009 all from Ain Shams University, Cairo, Egypt. Currently, he holds the position of product management director at Siemens EDA Egypt. He has been with Siemens EDA for 23 years. He has published many technical papers and articles in addition to many granted US patents.

Omar El-Sewefy received his BSc and MSc degrees from Ain Shams University, Cairo, Egypt, in 2008 and 2015, respectively. He is a field application engineer at Siemens Industry Software Inc. With 12 years of experience in design physical verification solutions and semiconductor manufacturing solutions, his industry experience covers semiconductor resolution enhancement techniques (RET), source mask optimization, design rule checks, and silicon photonics physical verification. He has been a member of SPIE since 2009.

Mohamed Dessouky received his BSc and MSc degrees in electrical engineering from the University of Ain Shams, Cairo, Egypt, in 1992 and 1995, respectively, and his PhD in electrical engineering from the University of Paris VI, Paris, France, in 2001. He has co-authored 50+ papers in refereed journals and conferences. He was a visiting professor at the University of Paris VI in 2002 and 2004. He holds three US patents.

Ashraf Salem received his BSc (Hons.) and MSc degrees in computer engineering from Ain Shams University, Cairo, Egypt, in 1983 and 1987, respectively, and his PhD in computer engineering from the University of Joseph Fourier, Grenoble, France, in 1992. He is currently a professor with the Department of Computer and Systems Engineering, Ain Shams University.