DeePattern: Layout Pattern Generation with Transforming Convolutional Auto-Encoder

Haoyu Yang
Chinese University of Hong Kong
hyyang@cse.cuhk.edu.hk

Piyush Pathak
Cadence Design Systems, Inc.
ppathak@cadence.com

Frank Gennari
Cadence Design Systems, Inc.
gennari@cadence.com

Ya-Chieh Lai
Cadence Design Systems, Inc.
ylai@cadence.com

Bei Yu
Chinese University of Hong Kong
byu@cse.cuhk.edu.hk

ABSTRACT

VLSI layout patterns provide critic resources in various design for manufacturability researches, from early technology node development to back-end design and sign-off flows. However, a diverse layout pattern library is not always available due to long logic-to-chip design cycle, which slows down the technology node development procedure. To address this issue, in this paper, we explore the capability of generative machine learning models to synthesize layout patterns. A transforming convolutional auto-encoder is developed to learn vector-based instantiations of squish pattern topologies. We show our framework can capture simple design rules and contributes to enlarging the existing squish topology space under certain transformations. Geometry information of each squish topology is obtained from an associated linear system derived from design rule constraints. Experiments on 7*nm* EUV designs show that our framework can more effectively generate diverse pattern libraries with DRC-clean patterns compared to a state-of-the-art industrial layout pattern generator.

1 INTRODUCTION

VLSI layout patterns provide critic resources in various design for manufacturability (DFM) researches, from (1) early technology node development to (2) back-end design and sign-off flows [1]. The former includes perfection of design rules, OPC recipes, lithography models and so on. The latter covers, but is not limited to, layout hotspot detection and correction [2–8]. However, layout pattern libraries are sometimes not large and diverse enough for DFM researches/solutions due to long logic-to-chip design cycle. Even some test layouts can be synthesized within a short period, they are usually restricted by certain design rules and the generated pattern diversity is limited [9, 10].

One state-of-the-art industrial pattern generation solution is migration from older technology node designs [11], where seed patterns are selected from the process weak points in older designs and shrunk by a scale factor to match current design rules. However, such flow is no longer applicable when there are large gaps between design nodes. An example is that 7*nm* EUV designs contain all unidirectional shapes while patterns are still 2D in previous design nodes. Monte Carlo shape generation is another approach that is typically applied industry-wide to generate metal layer patterns for lithography and OPC research. In this method, shapes are randomly created and placed on a given region according to certain geometry constraints, which, to some extent, limits the pattern library diversity created with the flow (see Figure 1(a)).

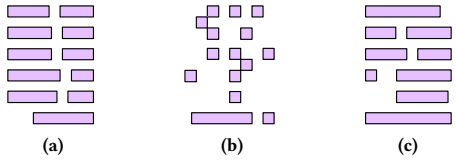


Figure 1: Sample pattern topologies generated from (a) Industry Tool, (b) DCGAN [19], and (c) our proposed TCAE.

In this paper, we study the capability of generative models to synthesize test layout patterns. The most popular generative machine learning model recently is generative adversarial network (GAN) that is first proposed in [12]. GAN consists of two sub-neural networks interacting with each other. One is called the generator, which takes random latent vectors as input and generates images or patterns following some distribution as of the training data set. The other is discriminator, which aims to discriminate real patterns from the generated ones. GAN models have been widely studied on computer vision related tasks like image synthesis [12, 13] and scene transformation [14, 15] by extracting and understanding feature distributions. In EDA area, such architecture and its variations have also been successfully applied including mask optimization [16, 17] and on-chip sensor placement [18]. However, generating layout patterns is a more challenging task as layouts are constrained by design rules which are hard to learn with native GAN training mechanisms, as shown in Figure 1(b). For the example of 7*nm* EUV designs, M2 layer shapes are all unidirectional and cannot appear in adjacent wire tracks.

To overcome these problems, we propose a transforming convolutional auto-encoder (TCAE) architecture that consists of a recognition unit and a generation unit. The recognition unit is built with stacked convolutional layers that convert layout pattern topologies into latent vectors that allow perturbation for certain transformations. The generation unit is expected to capture layout spatial information (e.g. track and wire direction) well and converts latent vectors back to legal pattern topologies with corresponding deconvolution operations. We will show that perturbation on individual latent vector node contributes to pattern transformation in certain layout design rule domain and admits a larger pattern library diversity, as shown in Figure 1(c). The main contributions of this paper are listed as follows:

DAC '19, June 2–6, 2019, Las Vegas, NV, USA

ACM ISBN 978-1-4503-6725-7/19/06…$15.00
https://doi.org/10.1145/3316781.3317795

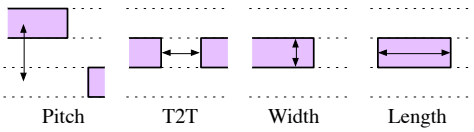


Figure 2: Layout geometry and critical dimensions.

- We propose a pattern generation framework that reduces the challenging pattern generation problem into two simpler subproblems with the aid of efficient squish pattern representation.
- We design a TCAE architecture that aims at generating pattern topologies efficiently.
- We develop linear systems that can provide solutions of pattern geometric information which, combined with generated squish topology, produces DRC-clean patterns.
- We conduct experiments to show that each latent vector node in TCAE comes along with real physical meanings in layout pattern space and transformations on latent vectors can produce additional patterns of interest. Results show that the generated pattern library exhibits larger pattern number and pattern diversity compared to a state-of-the-art industry layout generator.

The rest of the paper is organized as follows. Section 2 introduces basic terminologies and evaluation metrics related to this work. Section 3 shows details of the pattern generation framework including topology generation and legal pattern assessment. Section 4 covers experimental results and Section 5 concludes the paper.

2 PRELIMINARIES

In this section, we introduce some geometry concepts to accommodate layout design rules and the pattern generation flow. Because in this paper we focus on 7*nm* EUV metal layer designs which contains only unidirectional on-track shapes, following terms are accordingly adopted to quantize pattern shapes and positions, as depicted in Figure 2. *Pitch*, denoted as p, measures the distance between two adjacent tracks that contain shapes. *T2T*, denoted as t, measures the line-end-to-line-end distance between two adjacent shapes in a track. Wire *length* l and *width* w measure the shape size along and against the design track, respectively. In order for a pattern to be DRC-clean, these measurements will be constrained by some design rules.

All shape edges in a fixed-size window are aligned with x-axis and y-axis. If we extend all horizontal and vertical edges infinitely into scan lines, more non-overlapping scan lines always come with more complex patterns. We hence define the complexity of a layout pattern as follows.

Definition 1 (Pattern Complexity)**.** The complexity of a pattern in x and y directions (denoted as c_x and c_y) are defined as the number of scan lines subtracted by one along x-axis and y-axis, respectively.

We also introduce the concept of *pattern diversity* (denoted as H) to measure how are the pattern complexities distributed in a given library. A larger H implies the library contains patterns that are more evenly distributed, as in the following definition.

Definition 2 (Pattern Diversity)**.** The diversity of a pattern library is given by the *Shannon Entropy* [20] of the pattern complexity sampled from the library, as shown in Equation (1),

$$\mathrm{H} = -\sum_i \sum_j P(c_{xi}, c_{yj}) \log P(c_{xi}, c_{yj}), \tag{1}$$

where $P(c_{xi}, c_{yj})$ is the probability of a pattern sampled from the library has complexities of c_{xi} and c_{yj} in x and y directions respectively.

With above definitions, the pattern generation problem can be formulated as follows.

Problem 1 (Pattern Generation)**.** Given a set of layout design rules, the objective of pattern generation is to generate a pattern library such that the pattern diversity and the number of unique DRC-clean patterns in the library is maximized.

3 THE FRAMEWORK

Generating layout patterns is extremely challenging for learning machines as design rules are usually not friendly to most machine learning models. We therefore simplify the problem into two stages with the aid of squish patterns [21]. We first deal with the topology generation problem and then establish a linear system to finalize the pattern generation flow with proper geometry vectors.

3.1 Squish Pattern Extraction

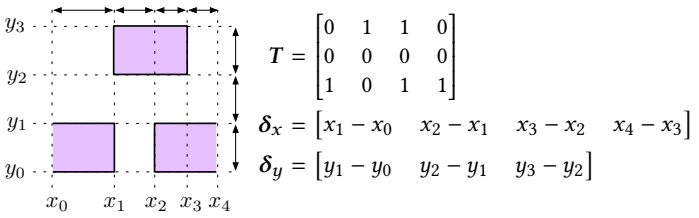
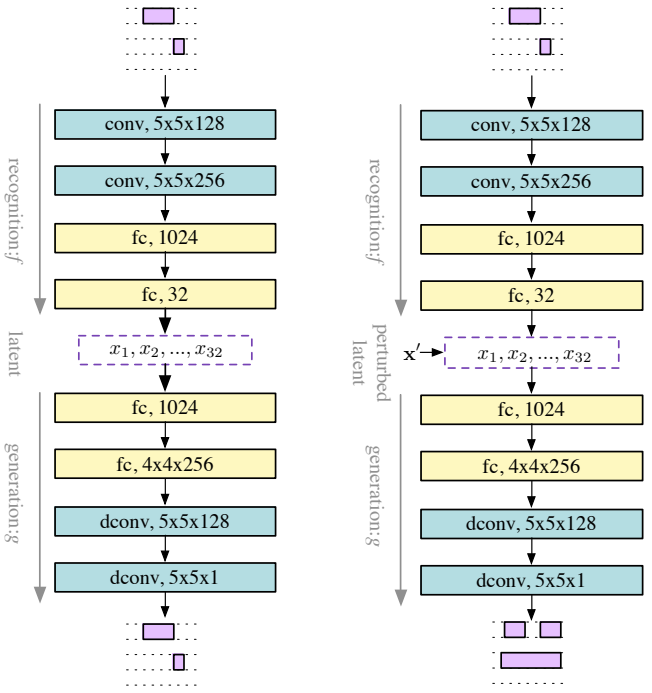


Figure 3: Squish pattern generation.

Squish pattern is a scan line-based representation that each layout clip is cut into grids aligned at all shape edges, as shown in Figure 3. The squish pattern representation of a given layout clip consists of a topology matrix $\boldsymbol{T}$ and two vectors $\boldsymbol{\delta}_x$ and $\boldsymbol{\delta}_y$ that contain geometry information in x and y directions. Each entry of $\boldsymbol{T}$ is either zero or one which indicates shape or space respectively. The geometric information describes the size of each grid. For example, the pattern in Figure 3 can be accordingly expressed as in the right side of Figure 3. Here x_is and y_is are the locations of vertical and horizontal scan lines respectively, and the pattern complexity is accordingly given by $c_x = 4$ and $c_y = 3$. Canonically, (x_0, y_0) is the coordinate of the bottom left corner of the pattern and $x_i < x_{i+1}, y_i < y_{i+1}$. From the squish pattern extraction procedure, we can also easily obtain the following lemma.

Lemma 1. *Squish pattern representation is lossless.*

Now the problem becomes generating legal topologies and solving associated $\boldsymbol{\delta}_x$s and $\boldsymbol{\delta}_y$s that are much easier than directly generating DRC-clean patterns. The advantages of squish patterns are two-fold: (1) Squish patterns are storage-efficient and supports neural networks and other machine learning models. (2) Squish patterns are naturally compatible with the simplified pattern generation flow that will be discussed in following sections.



(a) Training (b) Testing

Figure 4: Architecture of transforming convolutional auto-encoder in (a) training phase and (b) testing phase.

3.2 Topology Generation

3.2.1 Transforming Convolutional Auto-Encoder (TCAE). In this paper, we propose a TCAE architecture that aims at efficient pattern topology $\boldsymbol{T}$ generation. The TCAE is derived from original transforming auto-encoders (TAEs) [22] which are a group of densely connected auto-encoders and each individual, referred as a capsule, is targeting on certain image-to-image transformations. Each capsule employs a recognition unit and a generation unit to capture a pose position and re-synthesize the translated object, respectively. The translation is defined on a regular coordination system that will be added up to the original pose position before fed into the generation unit. In the training phase, neuron weights are updated by backpropagating the differences between the output image and the actual translated image given the translation information. After the neural network is trained, it takes inputs of an image and translation information and outputs the image with desired shifts. Such architecture agrees with the pattern generation tasks in the following aspects. (1) Feature instantiation attains data set domain properties. (2) All capsules contribute together to produce variations of any input objects. Although such architecture can capture the feature characteristics from original object pixel intensities, TAEs cannot be directly applied for pattern generation due to the fact that *transformations are restricted by layout design rules and only very simple pose transformations are supported by original TAEs, which does not satisfy our pattern generation objectives.*

Observe that each capsule unit functions similarly as independent receptive field in convolutional layers, we develop the TCAE architecture for feature learning and pattern reconstruction, as shown in Figure 4. The detection unit in TCAE consists of multiple convolutional layers for hierarchical feature extraction, followed by several

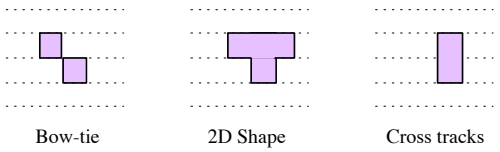


Figure 5: Illegal topology examples.

densely connected layers as an instantiation of the input pattern in the latent vector space, as in Equation (2).

$$\boldsymbol{l} = f(\boldsymbol{T}; \boldsymbol{W}_f), \tag{2}$$

where $\boldsymbol{l}$ is the latent vector, $\boldsymbol{T}$ represents the input topology and $\boldsymbol{W}_f$ contains all the trainable parameters associated with the recognition unit. The latent vector works similarly as a group of capsule units with each node being an low level feature representation. We will show each latent vector node contributes to pattern shape globally or locally in the experiment section.

The generation unit contains deconvolutional layers [23] that cast the pattern object from the latent vector space back to the original pattern space, as in Equation (3).

$$\boldsymbol{T}' = g(\boldsymbol{l} + \Delta\boldsymbol{l}; \boldsymbol{W}_g), \tag{3}$$

where $\Delta\boldsymbol{l}$ is the perturbation applied on the latent vector that allows inputs to conduct transformations. During training, we force the TCAE to learn an identity mapping with the following objectives.

$$\min_{\boldsymbol{W}_f, \boldsymbol{W}_g} ||\boldsymbol{T} - \boldsymbol{T}'||_2, \text{ s.t. } \Delta\boldsymbol{l} = \boldsymbol{0}. \tag{4}$$

The TCAE-based framework differs from TAEs in the following aspects.

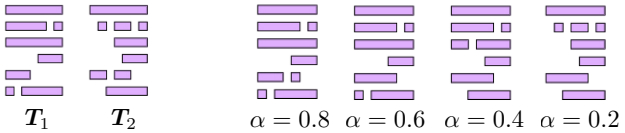
- We replace the group capsules with a simpler latent vector that contains feature nodes that connect to different receptive fields in the early convolutional layers and hence can represent certain part-whole feature instantiation.
- The transformation in our framework applies directly on the latent vector space that promises a much larger diversity of the generated patterns compared to the limited transformation on the coordinate system only in TAEs.
- Identity mapping in the training phase helps the TCAE capture the design rule properties of existing patterns.

Once the TCAE is trained, we can adopt the flow in Figure 4(b) to generate pattern topologies from perturbed latent vector space of existing layout patterns. During the inference phase, we feed a group of squish topologies into the trained recognition unit that extract latent vector instantiations of existing topologies. Perturbation on the latent vector space is expected to expand the existing pattern library with legal topologies. We claim the following assumption.

Assumption 1. The latent vector space offers richer information and each latent vector node represents patten geometry locally or globally, e.g., some vector nodes may control shape sizes while some others may control shape positions.

3.2.2 TCAE-Combine. A straightforward perturbation approach is combining existing topologies in the latent vector space, which is expected to fill certain region of a given pattern library. The combination rule can be defined as Equation (5).

$$\boldsymbol{T}_g = g(\sum_i \alpha_i f(\boldsymbol{T}_i)), \tag{5}$$



(a) Origin **(b)** $g(\alpha f(T_1) + (1-\alpha) f(T_2))$

Figure 6: Combination of existing patterns with latent vectors.

where $0 < \alpha_i < 1, \forall i$ are combination coefficients and satisfy $\sum_i \alpha_i = 1$. However, the diversity topologies generated by such approach might be limited by the existing pattern complexity. We randomly pick two patterns from existing designs and combine them in the latent vector space using TCAE. As shown in Figure 6, even we adjust the combination coefficient with a large step, there are still repeating topologies in the reconstructed topology set.

3.2.3 TCAE-Random. Another approach is introducing random perturbations in the latent vectors. We take Assumption 1 into consideration when generating perturbation vectors to avoid illegal topologies as many as possible. We introduce the concept of feature sensitivity $\boldsymbol{s}$ that statistically defines how easily an legal topology can be transformed to illegal when manipulating the latent vector node with everything else unchanged.

Definition 3 (Feature Sensitivity)**.** Let $\boldsymbol{l} = \begin{bmatrix} l_1 & l_2 & \dots & l_n \end{bmatrix}^\top$ be the output of the layer associated with the latent vector space. The sensitivity s_i of a latent vector node l_i is defined as the probability of reconstructed pattern being invalid when a perturbation $\Delta l_i \in [-t, t]$ is added up on l_i with everything else unchanged.

It can be seen from Definition 3 that a larger s_i indicates the corresponding latent vector node l_i is more likely to create invalid topologies if a large perturbation is applied. We therefore avoid manipulating such nodes when sampling random perturbation vectors from a Gaussian distribution. The s_is are estimated following Algorithm 1, which requires a set of legal topologies and trained TCAE. The sensitivity of each latent vector node is estimated individually (lines 1–2). We first obtain the latent vectors of all topologies in $\mathcal{T}$ and feed them into the reconstruction unit along with certain perturbation on one latent vector node (lines 3–5). Reconstructed patterns are appended in the corresponding set $\mathcal{R}_i$ (line 6). The sensitivity of the latent vector node i is given by the fraction of invalid topologies in $\mathcal{R}_i$ (line 8).

After we get the estimated sensitivity of all latent vector nodes, we are able to sample perturbation vectors whose elements are sampled independently from $\mathcal{N}(0, \frac{1}{s_i})$. These perturbation vectors will be added up to the latent vectors of existing pattern topologies to formulate perturbed latent vectors which will be fed into the generation unit to construct new topologies. Because we focus on EUV metal layers in this work, a topology is illegal if and only if it contains any patterns in Figure 5. That is, illegal topologies can be filtered out by checking whether shapes appear at any two adjacent tracks.

3.3 Legal Pattern Assessment

To generate DRC clean patterns, we need legal $\boldsymbol{\delta}_x$s and $\boldsymbol{\delta}_y$s of all generated topologies. We first detect all critical dimensions listed in Figure 2 in each valid topology and then formulate a linear system

Algorithm 1 Estimating feature sensitivity. $\mathcal{T} = \{\boldsymbol{T}_1, \boldsymbol{T}_2, ..., \boldsymbol{T}_N\}$ is a set of valid pattern topologies, f and g are trained recognition unit and generation unit respectively, t determines the perturbation range, and $\boldsymbol{s}$ is the estimated feature sensitivity.

Input: $\mathcal{T}, f, g, t$.
Output: $\boldsymbol{s}$.
1: $\mathcal{R}_i \leftarrow \emptyset, \forall i = 1, 2, ..., N$;
2: **for** $i = 1, 2, ..., n$ **do**
3: **for** $\lambda = -t : t$ **do**
4: $\Delta \boldsymbol{l} \leftarrow \boldsymbol{0}, \Delta l_i \leftarrow \lambda$;
5: $\mathcal{T}_i \leftarrow g(f(\mathcal{T}) + \Delta \boldsymbol{l})$;
6: $\mathcal{R}_i \leftarrow \mathcal{R}_i + \mathcal{T}_i$;
7: **end for**
8: $s_i \leftarrow$ fraction of invalid topologies in $\mathcal{R}_i$;
9: **end for**
10: **return** $\boldsymbol{s}$.

combining all associated constraints, as in Formula (6).

$$y_{i+1} - y_i = \frac{p}{2}, \qquad \forall i, \tag{6a}$$

$$x_i - x_j = t_{\min}, \qquad \forall (i,j) \in \mathcal{C}_{T2T}, \tag{6b}$$

$$x_i - x_j = l_{\min}, \qquad \forall (i,j) \in \mathcal{C}_W, \tag{6c}$$

$$x_{i+1} - x_i > 0, \qquad \forall i, \tag{6d}$$

$$x_{\max} - x_0 = d_x, y_{\max} - y_0 = d_y. \tag{6e}$$

where d_x, d_y, $t_{\min}$ and $l_{\min}$ denote clip width, clip height, minimum tip-to-tip distance and wire length (refer to Figure 2), respectively. Each index pair $(i,j) \in \mathcal{C}_{T2T}$ indicates that there exists at least one tip-to-tip pattern at scan lines x_i and x_j in the clip. $\mathcal{C}_W$ is defined similarly for wire patterns. x_0 and y_0 define the origin of each clip that can be any value and do not affect pattern complexities. Certain constraint values correspond to the minimum critical dimensions when no defects are found in EUV simulation under a given process window [24]. Note that the system in Equation (6) can be efficiently solved with vast linear programming algorithms or numerical methods. A solution of $\boldsymbol{\delta}_x$ and $\boldsymbol{\delta}_y$ in together with the associated topology matrix $\boldsymbol{T}$ formulates a complete squish pattern representation.

3.4 Overall Flow

We summarize the pattern generation flow as in Figure 7, where key steps include squish pattern extraction, topology generation and pattern generation. In the topology generation phase, we force the TCAE to learn an identity mapping that can capture simple but important design rules. Such strategy also allows us to create a large fraction of new legal topologies by perturbing the latent vectors. In the final pattern generation stage, we search critical dimensions

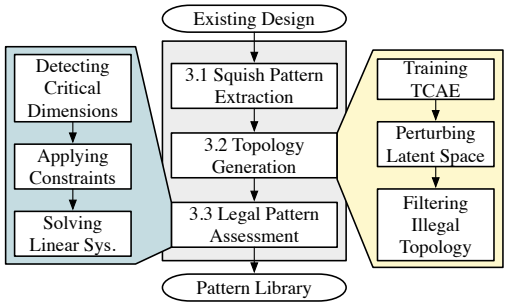


Figure 7: Proposed layout pattern generation flow.

that are defined in the design rules in all generated topologies and formulate corresponding linear systems to obtain legal $\boldsymbol{\delta}_x$s and $\boldsymbol{\delta}_y$s.

4 EXPERIMENTAL RESULTS

4.1 The Dataset and Configurations

We implement the pattern generation flow using `Python` and `Tensorflow` [25] library. The framework is tested on a platform with one Tesla P100 Graphic Card. We train the TCAE on a dataset with 1 million metal-2 layout clips under 7nm EUV design node. The clip size is $192 \times 192nm^2$ and the corresponding squish topology size is zero-padded to 16×16 that will be the input size of the neural networks. The initial learning rate is set to 0.001 and decays by 0.7 every 2000 iterations. The maximum number of training steps is 10000 with a mini-batch size of 64. All neuron weights are initialized with Xavier [26] initializer and regularized with l_2 regularizer. The regularization coefficients for convolution layers are 0.001 and we chose 0.01 for densely connected layers. No data augmentation strategies are employed during training and the model at last training step is picked during inference stage. Note that we mark topologies with $c_x > 12$ and $c_y > 12$ as illegal such that the linear systems associated to legal topologies always *admit at least one* solution under the given window size, which ensures the quantity and quality of the generated pattern libraries. We adopt industrial solver when generating geometry information with Equation (6) for new topologies and only one solution is kept for each topology.

4.2 Understanding Features in TCAE

In the first experiment, we study the relationship between auto-learned features and human understandable layout space. Because the TCAE is trained to reconstruct input topologies as accurate as possible, feature vectors derived from the latent vector layer must attain all geometry informations that include wire tracks, line-end alignments, tip-to-tip distances, shape directions and so on. To show exactly how these auto-learned features affect the topology space, we conduct simple transformations on each individual entry of the latent vector and keep everything else unchanged. The transformed feature vectors are then fed into the reconstruction unit for topology reconstruction. We visualize part of the reconstructed patterns in Table 1, where each row corresponds to the transformation on certain nodes in the latent vector with everything else unchanged. In our case, a small perturbation is added up to a specific entry. We can easily observe that some features extend or pull back line-ends, some features create or destroy geometries and some features controls the

Table 1: Visualizing how are convolutional features reflected in original topology space.

Transformations	Reconstructed Topologies
Extend or pull back line-ends	
Create or destroy shapes	
Control shape directions	

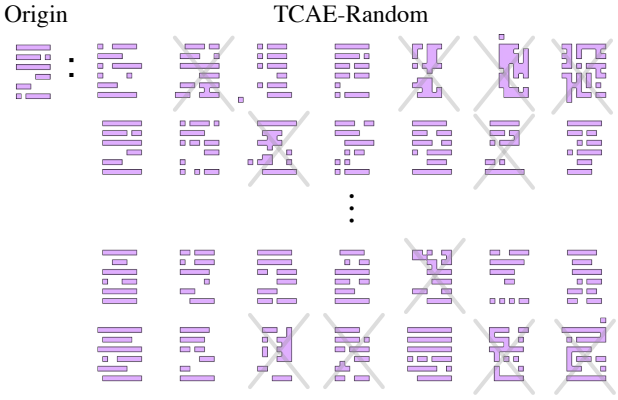


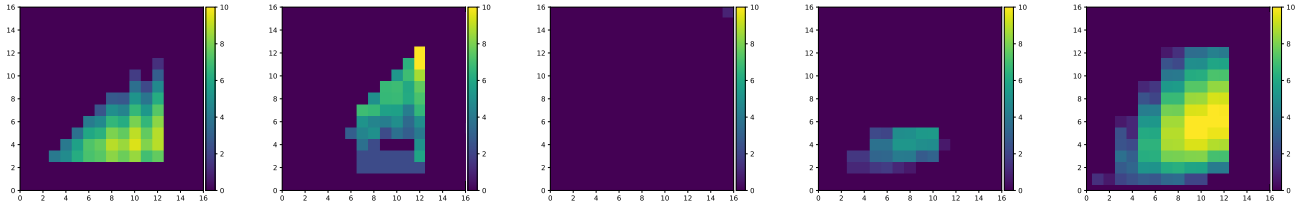
Figure 8: Contribution of Gaussian perturbation on topology reconstruction. 1000 topologies (~400 legal) are created from one topology randomly picked from the existing pattern library.

directions of shapes. Unlike traditional design rules, auto-learned features control layout patterns in a more global point of view that some feature determines spacing, wire length and sometimes geometry direction as a whole.

Here we show how random perturbations on the latent vectors contribute to topology generation. We randomly take one topology from the training set and obtain its feature vector through the trained encoder network. 1000 noise vectors sampled from Gaussian are added up to the feature vector before it is fed into the decoder network for pattern reconstruction. We visualize the generated topologies in Figure 8. We can observe that perturbations create significantly amount of new topologies where a large fraction of them are consistent with EUV pattern rules (e.g. no bow-tie shapes and unique direction single track polygons). Such results also show that deconvolution layers have the ability to learn simple design rules during training. On the contrary, no valid topology will be generated if the noise are directly applied on pattern space.

4.3 Comparison with State-of-the-art

We have shown the manipulation in the latent vector space can generate new topologies. In the last experiment, we will make use of the flow above to augment the pattern space. Pattern library statistics are listed in Table 2. Column "Method" denotes the approach used to generate layout patterns, column "Pattern #" denotes the number of DRC clean patterns that are different from others, and column "Pattern Diversity" corresponds to the Shannon Entropy of each pattern library in terms of pattern complexity. Row "TCAE-Random" corresponds to the details of 1M patterns generated by perturbing the features of 1000 patterns in existing design with Gaussian noise. Row "TCAE-Combine" represents patterns generated from 1M different combinations of 10 test layout clip features. "Industry Tool" shows the cataloged results of a test layout generated from a state-of-the-art industry layout generator. The test layout has similar total chip area (10000 μm^2) as "TCAE-Random" (14807 μm^2). We also implement a DCGAN [19] that has similar number of trainable parameters as the TCAE designed in this paper. 1M patterns are generated by feeding random latent vectors in the trained generator networks. "Existing Design" lists the statistics of a pattern library extracted from an industry layout.



(a) Existing Design (b) Industry Tool (c) DCGAN (d) TCAE-Combine (e) TCAE-Random

Figure 9: Visualization of the distribution of layout libraries: (a) Existing layout pattern dataset. (b) Industrial layout generator; (c) Patterns generated by DCGAN; (d) Patterns generated by TCAE-Combine; (e) Patterns generated by TCAE-Random.

Table 2: Statistics of generated patterns.

Method	Pattern #	Pattern Diversity (H)
Existing Design	-	3.101
Industry Tool	55408	1.642
DCGAN	1	0
TCAE-Combine	1738	2.665
TCAE-Random	**286898**	**3.337**

Perturbation with Gaussian exhibits greatest pattern generation power with around 30% generated patterns are unique and DRC clean. Combination of patterns in feature space shows much less unique pattern count because the generation procedure are restricted by existing pattern space. Combine more patterns will not affect the result much which will significantly reduce the count of DRC clean patterns if any two candidate patterns contain unaligned wires. Most GAN generated patterns fail with bow-tie or 2D wires even the training procedure has reached the equilibrium point because it is very hard to learn layout track information with randomly generated latent vectors.

Figure 9 compares the distributions of generated patterns and existing layout data set with similar pattern count, where x-axis and y-axis denote pattern complexity in each direction and the heatmap value is the total count (`log`-scale) of the pattern with that complexity. We employ *Pattern Diversity* to measure the pattern library distribution. We observe that Random perturbation can efficiently expand the weakly distributed pattern library (large fraction of patterns falls in certain complexities) with $H = 3.337$ while the industrial layout generator are still weakly distributed with $H = 1.642$ compared to existing designs.

5 CONCLUSION

In this paper, we address the pattern library requirements in DFM flows/researches under advanced technology nodes. We propose a transforming convolutional auto-encoder framework that can capture layout design rule characteristics. We show individual element in latent vector instantiation contributes to form the pattern space locally or globally, which inspires a pattern generation flow with perturbation of the latent vector space. The experimental results show that our framework outperforms a state-of-the-art industrial layout generation tool in terms of pattern library diversity, which is promising to facilitate early technology node development and the back-end and sign-off flows.

ACKNOWLEDGMENT

This work is supported in part by The Research Grants Council of Hong Kong SAR (Project No. CUHK24209017).

REFERENCES

[1] C. Tabery, Y. Zou, V. Arnoux, P. Raghavan, R.-h. Kim, M. Côté, L. Mattii, Y.-C. Lai, and P. Hurat, "In-design and signoff lithography physical analysis for 7/5nm," in *SPIE Advanced Lithography*, vol. 10147, 2017.
[2] H. Yang, L. Luo, J. Su, C. Lin, and B. Yu, "Imbalance aware lithography hotspot detection: a deep learning approach," *JM3*, vol. 16, no. 3, 2017.
[3] B. Jiang, H. Zhang, J. Yang, and E. F. Young, "A fast machine learning-based mask printability predictor for opc acceleration," in *Proc. ASPDAC*, 2019.
[4] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Y. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE TCAD*, 2018.
[5] M. Shin and J.-H. Lee, "Accurate lithography hotspot detection using deep convolutional neural networks," *JM3*, vol. 15, no. 4, 2016.
[6] H. Yang, P. Pathak, F. Gennari, Y.-C. Lai, and B. Yu, "Detecting multi-layer layout hotspots with adaptive squish patterns," in *Proc. ASPDAC*, 2019.
[7] H. Geng, H. Yang, Y. Ma, J. Mitra, and B. Yu, "SRAF insertion via supervised dictionary learning," in *Proc. ASPDAC*, 2019.
[8] H. Yang, Y. Lin, B. Yu, and E. F. Y. Young, "Lithography hotspot detection: From shallow to deep learning," in *Proc. SOCC*, 2017.
[9] H. Yang, S. Li, C. Tabery, B. Lin, and B. Yu, "Bridging the gap between layout pattern sampling and hotspot detection via batch active sampling," *arXiv preprint arXiv:1807.06446*, 2018.
[10] Y. Lin, M. Li, Y. Watanabe, T. Kimura, T. Matsunawa, S. Nojima, and D. Z. Pan, "Data efficient lithography modeling with transfer learning and active data selection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.
[11] L. Zhuang, J. Xu, M. Tsai, Q. W. Liu, E. Yang, Y. Zhang, J. Sweis, C. Lai, and H. Ding, "A novel methodology of process weak-point identification to accelerate process development and yield ramp-up," in *Proc. ICSICT*, 2016.
[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. NIPS*, 2014.
[13] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Proc. NIPS*, 2016.
[14] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. ICCV*, 2017.
[15] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.
[16] H. Yang, S. Li, Y. Ma, B. Yu, and E. F. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," in *Proc. DAC*, 2018.
[17] H. Geng, H. Yang, B. Yu, X. Li, and X. Zeng, "Sparse vlsi layout feature extraction: A dictionary learning approach," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2018.
[18] J. Liu, Y. Ding, J. Yang, U. Schlichtmann, and Y. Shi, "Generative adversarial network based scalable on-chip noise sensor placement," in *Proc. SOCC*, 2017.
[19] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. ICLR*, 2016.
[20] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE mobile computing and communications review*, vol. 5, no. 1, 2001.
[21] F. E. Gennari and Y.-C. Lai, "Topology design using squish patterns," Sep. 9 2014, US Patent 8,832,621.
[22] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *Proc. ICANN*. Springer, 2011.
[23] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.
[24] P. Gupta, "What is process window?" *ACM SIGDA Newsletter*, vol. 40, no. 8, 2010.
[25] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. OSDI*, 2016.
[26] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS*, vol. 9, 2010.