Journal of Electronic Testing (2024) 40:603–614
https://doi.org/10.1007/s10836-024-06138-2

RESEARCH

Generating Synthetic Layout Test Patterns using Deep Learning

Adel Mahmoud[1] · **M. Watheq El-Kharashi**[1] · **Cherif Salama**[1,2]

Received: 15 March 2024 / Accepted: 10 September 2024 / Published online: 5 October 2024

Abstract
The testing process of various electronic design automation tools and the verification process of the manufacturability of a developed Design Rule Check (DRC) deck, particularly in very-large-scale integration designs, is highly dependent on the presence of test data that covers a vast range of the design space in a layout design. However, test data extracted from real designs often lacks the diversity necessary to ensure high coverage during the testing process. This creates a need for a synthetic layout test pattern generator capable of producing diverse test clips to ensure robust testing, where a test clip represents a DRC clean segment of a layout design that encapsulates various layout patterns. This study proposes the use of the advanced Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) to generate a vast and diverse collection of synthetic test clips. The proposed workflow includes WGAN-GP for pattern generation, followed by a legalization process, and ending with generation assessment. Experimental results have shown promising outcomes in terms of the quantity and diversity of the generated patterns. The generated patterns were fully legalized, ensuring they are DRC clean. The diversity of the generated test clips, assessed using Shannon entropy, reached 8.4, while the training clips scored 8.1. This indicates a high diversity ratio between the generated and training datasets, outperforming previous methods and demonstrating the effectiveness of the generation approach.

Keywords Automatic layout synthesis · Layout test pattern generation · Synthetic layout patterns

1 Introduction

The testing process for various Very-Large-Scale Integration (VLSI) applications requires patterns in the layout design that comprehensively cover a vast range of the design space. When developing Electronic Design Automation (EDA) tools, such as machine learning-based hotspot detection [1–3] or Optical Proximity Correction (OPC) [4] systems, test engineers must create test clips encompassing diverse variations of patterns found in real designs. In the downstream chip manufacturing process, Design Rule Check (DRC) decks developed for fabrication are qualified by verifying the manufacturability of different layout patterns that pass the DRC rules. A significant challenge in this testing process is achieving extensive coverage of the layout design space by having test clips that include patterns that are DRC clean and diverse enough to cover different scenarios. This is where the synthetic layout test pattern generator becomes crucial. Test engineers employ these generators to produce test clips containing synthetic layout patterns, which are then used to rigorously test the developed EDA tools and qualify DRC decks before actual fabrication. By utilizing synthetic patterns, engineers can ensure robust and comprehensive testing, ultimately enhancing the reliability and manufacturability of VLSI designs.

A synthetic layout test pattern generator is a tool that generates test clips, where a test clip is a square segment that carries DRC clean patterns that can be found as a part of a complete layout design. A good synthetic layout test pattern generator is the one that is capable of generating DRC clean test clips with high diversity in patterns encapsulated in the test clips.

Responsible Editor: L. Cassano

✉ M. Watheq El-Kharashi
watheq.elkharashi@eng.asu.edu.eg

Adel Mahmoud
adelmahmoud1997@gmail.com

Cherif Salama
cherif.salama@aucegypt.edu

[1] Computer and Systems Engineering, Ain Shams University, Cairo, Egypt

[2] The American University in Cairo, Cairo, Egypt

As illustrated in Fig. 1, the synthetic layout test pattern generator is a critical component in both the EDA tools testing flow and the DRC deck verification process. It functions as an automated test scenario generator, significantly enhancing the coverage of the testing process by producing test clips that are both DRC-compliant and highly diverse. Test engineers often face resource constraints when relying only on test clips derived from real designs. To overcome this limitation and achieve broader coverage, the synthetic layout test pattern generator utilizes real design clips as input to create a set of synthetic test clips containing new patterns. The integration of these synthetic test clips with the original real design clips leads to a more comprehensive testing process, ensuring higher coverage and more robust validation.

Early work of synthetic pattern generators was built based on deterministic and parametric techniques [5, 6]. Reddy et al. generated synthetic patterns by varying one or more features like corner-to-corner distances, jogs, and line-end positions [5]. These parametric methods have their own limitations when it comes to the diversity and the quantity of the generated patterns. Therefore, the capabilities of the generative deep learning techniques are being explored as solutions to the pattern generation problem, specially that generative deep learning was shown to overcome the limitations of quantity and diversity of the generated data when addressing other data generation problems. Previous studies on generating layout test patterns using generative deep learning have varied in how patterns are represented, generated, legalized, or evaluated. Each study has highlighted certain limitations, either in the data representation method or the generation approach. The motivation of this work is to introduce a new methodology for generating synthetic layout test patterns, offering a flexible image representation and a robust image generation method. Additionally, it presents the results and limitations of the proposed method, comparing them with those of previous studies.

The main contributions of this paper could be summarized as follows:

1. An introduction to an integrated workflow for generating synthetic test clips using generative machine learning techniques, followed with an evaluation process.
2. A proposal to utilize the Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) to generate diverse and design-like patterns.
3. Adoption of a Convolutional Auto Encoder (CAE) as a means to address the fuzziness in the generated images.
4. Application of morphological operations on the generated patterns for correction and legalization.
5. Demonstration of the remarkable responsiveness of the WGAN-GP and its immunity to the mode collapse issue, commonly encountered by Generative Adversarial Networks (GANs).

The rest of this paper is organized as follows. First, previous work on deep learning approaches to generate synthetic layout test patterns is illustrated in Section 2. The proposed generation methodology is outlined in Section 3, providing a detailed explanation of the approach. We illustrate the adopted evaluation techniques in Section 4. The experiments, their results, limitations, and the comparison with the previous work are discussed in Section 5. Finally, Section 6 states the conclusion of this research.

2 Related Work

In this section, we focus on previous research that has explored the generation of synthetic layout test patterns using generative deep learning techniques. These studies have approached the problem from various perspectives, focusing on data representation during the generation process, generation techniques, post-generation processing, and evaluation methods.

Some work represented the layout patterns using the squish pattern representation [7–9]. A squish representation, as shown in Fig. 2, divides a layout clip into a pattern topology and geometric information. The pattern topology is a binary grid where 1 represents the presence of a polygon and 0 represents absence of a polygon. The geometric information is represented in vectors that hold the height and width of each entry in the topology grid. Despite being efficient and lossless, this representation imposes a limitation on the generation process, where the geometric information is being preserved during the topology generation process.

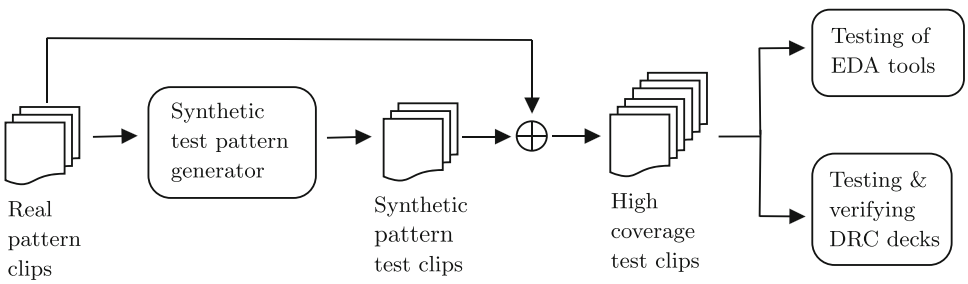

Fig. 1 Integration of the synthetic layout test pattern generator in the VLSI testing process: enhancing EDA tools validation and DRC decks qualification

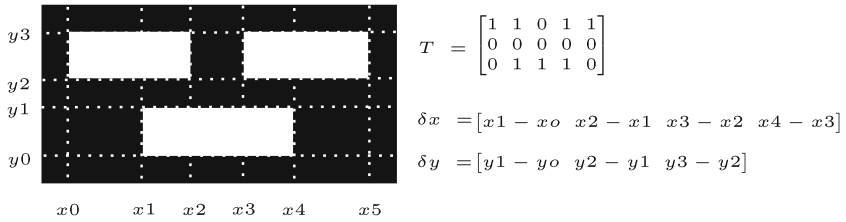

Fig. 2 Squish pattern representation with T as the topology matrix and δx, δy as geometric information along the x- and y-axes

Discrete Cosine Transform (DCT) was also used as a compact representation for pattern clips [10]. This representation adds the flexibility of ignoring the high frequencies in the pattern, enabling more compact representations.

Image Parameter Space (IPS) was adopted in another study as a representation for the pattern clips [11]. IPS consists of minimum intensity (Imin), maximum intensity (Imax), and maximum intensity slope (Islope) measured with target pattern at the center of local aerial image. This representation is popular in assessing pattern diversity [12].

Sequential modelling of the patterns was also utilized as data representation for the pattern clips [13]. Each polygon in the clip is represented as a sequence containing the coordinates of the starting point and the directions and offsets for walking through all the edges in turn from the starting point in a counter-clockwise direction. Each polygon is identified with a pair of tokens indicating the start and the end of the pattern sequence.

Based on the data representation technique, different generation methodologies were adopted.

Transforming Convolutional Auto-Encoder (TCAE), which is originally derived from Transforming Auto-Encoders (TAEs) [14], was used as a pattern generator [7]. TAEs are a group of densely connected auto-encoders and each individual, referred as a capsule, targets on certain image-to-image transformations. During the training phase, the TCAE is forced to learn identity mapping between the input topology and the output topology. Having the TCAE trained, new topologies are generated by perturbing the latent space of the existing layout patterns, where perturbation is expected to expand the existing topologies to new topologies. The perturbation process was approached either by combining the latent vectors of existing patterns or by applying random perturbation on the latent space of an existing pattern.

Variational Convolutional Auto-Encoder (VCAE), which is derived from the Variational Auto-Encoder (VAE) [15], was adopted as a pattern generator as well [8]. In addition to the VAE scheme, which is basically an encoder followed with a decoder where both are formed of a densely connected neural network, the VCAE adds convolutional and residual blocks in order to increase the capability of capturing the distribution of complex topologies. New topologies are generated by applying Gaussian perturbation to the latent vector of the existing pattern topologies.

Generation methodologies that adopted the squish representation during the generation step are restricted with the geometry vectors of the existing topologies during the generation process. Despite the generation methodology used, the generation step is concerned only with the topology part of the squish pattern while the geometry vectors are preserved.

DCT-GAN was introduced as a pattern generator, where the generator and discriminator are multi-layer perceptron network and the generator generates DCT signals on receiving an input of 10 random numbers in [-1, 1] range [10]. On the other side, the discriminator predicts the probability of the DCT signals being generated or from an actual training clip in order to train the generator. In addition to using a lossy representation for the layout patterns during the training process, a vanilla GAN [16] whose architecture was concluded in an exhaustive fashion was used. This may make the training process vulnerable to the well-known GANs problem like mode collapse or not being able to reach an equilibrium during the training process.

An IPS map generator followed by a layout generator were used for the generation process on using IPS representation [11]. The IPS map generator is basically an encoder, decoder, and discriminator whose goal together is to generate an IPS map with the desired IPS values at its center. The encoder along with decoder formulate an auto-encoder while the same encoder along with the discriminator formulate a GAN. The role of the GAN is to train the encoder in order that its latent vector follows a standard normal distribution while the role of the auto-encoder is to minimize the reconstruction error between the input of the encoder and the output of the decoder. Having the IPS map generated, a variant of U-Net [17] is used to transform the IPS map to a layout clip.

Transformers [18] were used for the generation process when using sequential representation for layout patterns due to their ability to capture long term dependencies between inputs [13]. Despite being efficient due to the absence of the need of the post-generation processing steps, this approach may be adding limitations on the generation process, given the increasing demands of curvilinear shapes.

Most of the work listed in this section deals with the test clips generation problem as an image generation problem with different representations for the images. These methods have some limitations that are either due to the data representation during the generation process or due to the

data generation method itself. Our study aims to overcome the limitations identified in previous research by leveraging state-of-the-art deep learning techniques, specifically employing WGAN-GP as the image generator. This approach is chosen due to WGAN-GP's demonstrated ability to produce high-quality, diverse images and its stability during the training process. We will evaluate the quality and diversity of the generated test clips produced in this study and compare them with those from earlier studies.

3 Pattern Generation Methodology

This section details the proposed generation methodology, which encompasses three main stages. First, the WGAN-GP is used to generate a diverse database of layout clips. These layout clips are represented as images during the generation and legalization process. After a layout clip is successfully generated using WGAN-GP, it undergoes a deblurring and fuzziness elimination process by being fed through a CAE. Finally, the deblurred clips undergo a post-processing step to fully legalize them and eliminate any pixel errors. Figure 3 provides a visual representation of the entire process. Each of the three stages is explained in more details in the following subsections.

3.1 Generation Stage

GANs [16] are mainly composed of two models of a multilayer perceptron network, the Generator (G) and the Discriminator (D). The discriminator is trained to determine the probability of an input being from a real dataset. So, the job of the discriminator is to determine whether the current input is real or fake. The generator is trained to learn the distribution of the training set. Having learned the training data distribution, the generator should be able to generate fake data samples that are pretty much similar to the training data by having an input of just a noise variable (z), where a change in the input (z) is expected to cause a change in the generated output. The generator and discriminator are trained simultaneously so that the generator is trying to adjust its output to be able to trick the discriminator while the discriminator is trying not to be tricked.

GANs are known to be hard to train. GANs usually suffer from several problems during training, like mode collapse, vanishing gradient, and lack of evaluation metrics during the training. The generator can simply learn to output a specific output that is enough to fool the discriminator during the training process and just stick to it. This problem is known as mode collapse. When mode collapse occurs, the generator is no longer generating different realistic data samples that represents the training data distribution. Instead, the generator keeps generating the same output that is good enough to fool the discriminator. Since the discriminator and generator are trained simultaneously, we may face another problem when the discriminator learns much faster than the generator that it can filter out the samples generated by the generator. Hence, we fall into the problem of vanishing gradient causing the learning process to stop as generator will not be able to learn anymore. GANs also lack having an objective function that tells us how good the training is going on so far, the loss value is not well correlated with the quality of the generated data. So, it is not easy to decide when to stop. Accordingly, even comparing the performance of different models cannot be done easily.

Wasserstein Generative Adversarial Network (WGAN) was defined to solve the main training problems that GANs face [19]. WGAN uses a reasonable approximation of the Wasserstein Distance or the Earth Movers (EM) distance. Wasserstein distance is a measure of the distance between two probability distributions. WGAN was emphasized to have the ability to go over the problem of mode collapse and showed how useful to make use of the discriminator job of estimating the EM distance continuously for debugging and evaluation. In a traditional GAN, the discriminator predicts the probability of the generated images being real or fake. In WGAN, the discriminator is replaced with a critic that scores the realness or fakeness of generated images. The critic is trained to approximate the Wasserstein distance between the generated data distribution and the training data distribution making use of the fact that the Wasserstein distance is continuous and differentiable so it has a linear gradient even after the

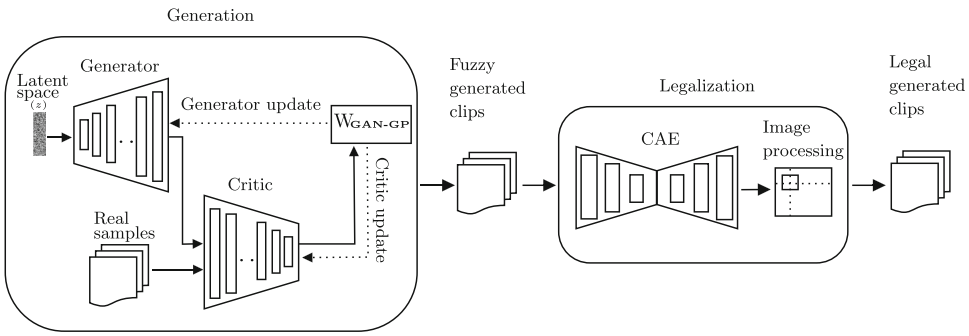

Fig. 3 A full flow of the proposed pattern generation methodology

critic is well-trained, which solves one of the main problems of the traditional GAN.

The WGAN adopted in this work for generation uses a linear activation function in the critic model. The critic model is updated five times more than the generator in each training iteration. The initial work that introduced the WGAN used to clip the critic model weights to a limited range after each mini batch update. Instead, we adopt the gradient penalty instead of weight clipping as weight clipping was demonstrated to being vulnerable to optimization difficulties [20]. The generator is formulated of six convolutional layers with batch normalization and Rectified Linear Unit (ReLU) activation except for the output layer, which has a hyperbolic tangent (tanh) activation and no batch normalization. The input to generator (z) is a vector of 1024 random numbers sampled from a normal distribution of mean 0 and variance 1. The detailed architecture of the generator is illustrated in Table 1.

The selection of the size of the input (z) was experimentally concluded by searching the parameters space of the convolutional auto-encoder demonstrated in Section 3.2 in order to achieve the optimal representation of the latent space of the training data.

The generator is trained to maximize the expected value of the critic's score for the generator's output, as shown in Eq. 1. It can be illustrated as if the generator is trying to minimize the difference between the critic's score for real images (x) and generated images.

$$\min_G -\mathbb{E}(D(G(z))) \tag{1}$$

Table 1 The network configuration of WGAN's generator

Layer	Input Size
ConvTranspose2d	1×1×1024
Batchnorm2d	4×4×512
Relu	4×4×512
ConvTranspose2d	4×4×512
Batchnorm2d	8×8×256
Relu	8×8×256
ConvTranspose2d	8×8×256
Batchnorm2d	16×16×128
Relu	16×16×128
ConvTranspose2d	16×16×128
Batchnorm2d	32×32×64
Relu	32×32×64
ConvTranspose2d	32×32×64
Batchnorm2d	64×64×32
Relu	64×64×32
ConvTranspose2d	64×64×32
Tanh	128×128×1

Table 2 The network configuration of WGAN's critic

Layer	Input Size
Conv2d	128×128×1
InstanceNorm2d	64×64×16
LeakyReLU	64×64×16
Conv2d	64×64×16
InstanceNorm2d	32×32×32
LeakyReLU	32×32×32
Conv2d	32×32×32
InstanceNorm2d	16×16×64
LeakyReLU	16×16×64
Conv2d	16×16×64
InstanceNorm2d	8×8×128
LeakyReLU	8×8×128
Conv2d	8×8×128
InstanceNorm2d	4×4×256
LeakyReLU	4×4×256
Conv2d	1×1×1

The critic is formulated from six deconvolutional layers with instance normalization and leaky ReLU activation except for the output layer, which uses the linear activation. The detailed architecture of the critic is illustrated in Table 2.

Contrary to the generator, the critic is trained to maximize the difference between the scores for real and generated images, as illustrated in Eq. 2.

$$\min_D -[\mathbb{E}(D(x)) - \mathbb{E}(D(G(z)))] \tag{2}$$

Figure 4 shows sample images of the WGAN-GP's generator output. The output of the generation step is showing

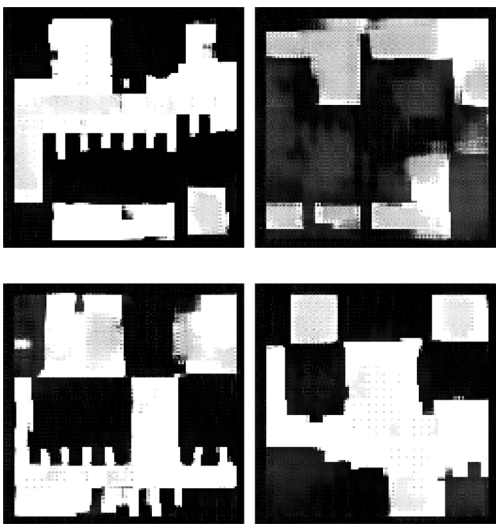

Fig. 4 Sample of the WGAN-GP's generator output after training

good diversity and an acceptable quality, but the generated images include some fuzziness, which needs to be worked on in the upcoming steps.

3.2 Deblurring and Fuzziness Elimination Stage

CAE is an unsupervised learning dimensionality reduction technique composed of an encoder followed by a decoder. The encoder is formed of convolutional layers that work consecutively on reducing the dimensionality of the input image to the lowest dimensions reversible representation, which is called the latent space representation. The decoder is formed of deconvolutional layers that work on up-sampling the images from the latent space representation or the compressed representation to the original full representation.

An auto-encoder is trained to minimize the reconstruction error over the dataset. On achieving an optimum reconstruction error, the encoder is capable of finding the latent space representation that fits the input and the decoder is capable to up-sample the latent space representation again. Making use of this idea, a convolutional auto-encoder is used as a post generation correction or deblurring technique. The WGAN-GP outputs are not perfect ones. The output images of the previous generation step carry a fuzzy pattern distribution over the output clip. The role of the convolutional auto-encoder is to remove the fuzziness of the data distribution of the generated images. Figure 5 shows a sample CAE input and the corresponding output.

For the CAE to achieve this role efficiently, it should learn a good representation for the training dataset that does not over-fit or under-fit on it. The latent space should be sufficient to represent all the features extracted from the input clips. To achieve a reasonable architecture for the encoder, decoder, and the latent vector, a hyper-parameter optimization tool called Optuna was used.

Optuna [21], an open source hyper-parameter optimization framework used to automate the hyper-parameters search, is used to achieve the optimal architecture for the encoder, decoder and the optimal representation for the latent space. The number of layers and filters for both the encoder and decoder are parameterized as well as the size of the latent space. Optuna is used to settle on the best architecture that results in the lowest reconstruction error over the test set, where each architecture tried out is trained till the reconstruction error is minimized on the training set and validated against over-fitting using a validation set.

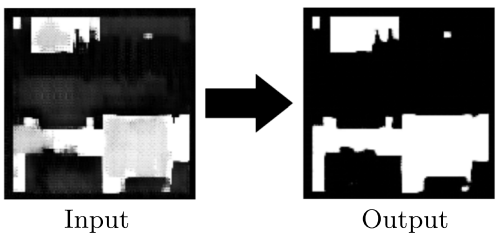

Fig. 5 Sample of the input/output of the CAE

Table 3 The CAE encoder network configuration

Layer	Input Size
Conv2d	128×128×1
InstanceNorm2d	64×64×16
LeakyReLU	64×64×16
Conv2d	64×64×16
InstanceNorm2d	32×32×32
LeakyReLU	32×32×32
Conv2d	32×32×32
LeakyReLU	16×16×4

The best results were obtained with the architecture formulated of three convolutional layers in the encoder and three deconvolutional layers in the decoder with the middle layer size set to 1024. The architecture of the encoder and decoder of the CAE is illustrated in Tables 3 and 4, respectively.

3.3 Post-Processing Stage

Dealing with the layout pattern generation problem as an image generation problem comes with the drawbacks of having a non-perfect image on the pixel level. The generated images may exhibit scattered pixels that are not part of any generated polygons, as well as missing pixels within one or more of the generated polygons. The area of the generated polygons may be insufficient to meet the criteria for a valid polygon. For these reasons, a legalization or a post-processing step is applied to legalize the generated pattern on the pixels level. In this work, a set of morphological operations is applied on each of the generated clips to legalize the clip. Opening is responsible for removing scattered random pixels and small illegal polygons while closing is applied to fill in the missing pixels within a legal polygon. The kernels

Table 4 The CAE decoder network configuration

Layer	Input Size
ConvTranspose2d	16×16×4
InstanceNorm2d	8×8×8
LeakyReLU	8×8×8
ConvTranspose2d	8×8×8
InstanceNorm2d	16×16×16
LeakyReLU	16×16×16
ConvTranspose2d	16×16×16
Tanh	128×128×1

used during the morphological operations should be selected wisely with respect to the size of the clips in order to legalize the polygons without violating the data distribution or the polygons structure within a generated clip. The effect of the post-processing step on the input image is depicted in Fig. 6.

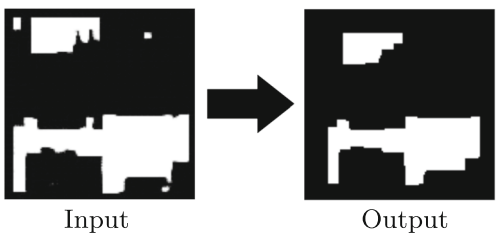

Fig. 6 Sample of the input/output of the post-processing step

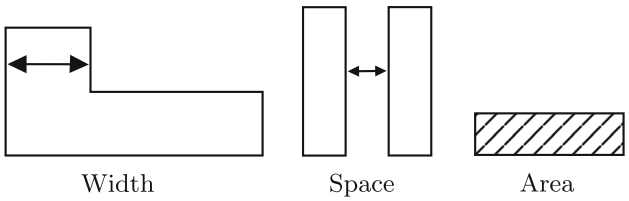

Fig. 7 The geometric design rules constraints on the layout patterns

4 Evaluation Methodology

The generated test clips, like any other generated data, need to be assessed by a well-defined quantitative assessment in order to be able to evaluate the quality of the generation methodology. The generated patterns should be ensured to be legal patterns in terms of the design constraints. Not only the legality of the generated patterns is the concern, but also the diversity of the generated patterns. The generated patterns should be assessed for how diverse they are compared to the training patterns and to the rest of the generated patterns. This ensures that the model actually generated new test clips and these generated test clips are diverse, so the model did not fall into the mode collapse trap. This section illustrates the evaluation metrics that were used to assess the legality and diversity of the generated test clips.

4.1 Pattern Legality

The legality assessment of the generated test clips ensures that the patterns within the clip comply with Design Rule Check (DRC) rules. A robust generation methodology should be able to produce test clips with patterns that meet the DRC requirements of the training layout clips. DRC rules differ from one design to another, with varying complexities. This study focuses on the geometric DRC rules, illustrated in Fig. 7, which are the same DRC rules previously incorporated in earlier research on generating layout patterns using generative deep learning. A *Width* rule constrains the minimum width of any shape in the clip. A *Spacing* rule constrains the minimum distance between any two adjacent polygons. An *Area* rule constrains the minimum area of each polygon in the generated pattern. The generation and legalization methodologies should be able to generate patterns that do not violate these constraints. Any generated pattern set that violates these rules is considered an unsuccessful sample.

4.2 Pattern Diversity

The main goal of the synthetic layout pattern generation is to enrich an already existing database of layout clips by a new and diverse set of generated clips. A successful generation methodology is characterized by its ability to produce a wide range of highly diverse synthetic clips. In line with prior work, the Shannon entropy of the distribution of pattern complexities is utilized as a metric to quantify the diversity of patterns [7, 13]. Pattern complexity (c_x, c_y) is defined in a previous work, where c_x is the number of scan lines subtracted by one along x-axis and c_y is the number of scan lines subtracted by one along y-axis [7]. Scan lines along x-axis are the vertical lines that pass through the unique x coordinates of the polygons. Scan lines along y-axis are the horizontal lines that pass through the unique y coordinates of the polygons. Therefore, the pattern diversity denoted by H is calculated according to Eq. 3, where a larger entropy indicates more diversity.

$$H = -\sum_i \sum_j P(c_{xi}, c_{yi}) \log P(c_{xi}, c_{yi}) \tag{3}$$

where $P(c_{xi}, c_{yi})$ denotes the probability of a pattern with complexity (c_{xi}, c_{yi}) in the set of patterns.

5 Experiments and Results

This section outlines the experimental details of the employed methodology and demonstrates the results. The entire process is implemented using the Python programming language. The deep neural networks are implemented using PyTorch [22]. OpenCV [23] is used for image processing. Gdspy [24] is utilized for manipulating Graphic Design System (GDS) files. Calibre DESIGNrev [25] software serves as the layout viewer, while Calibre DRC [26] software is employed for validation purposes.

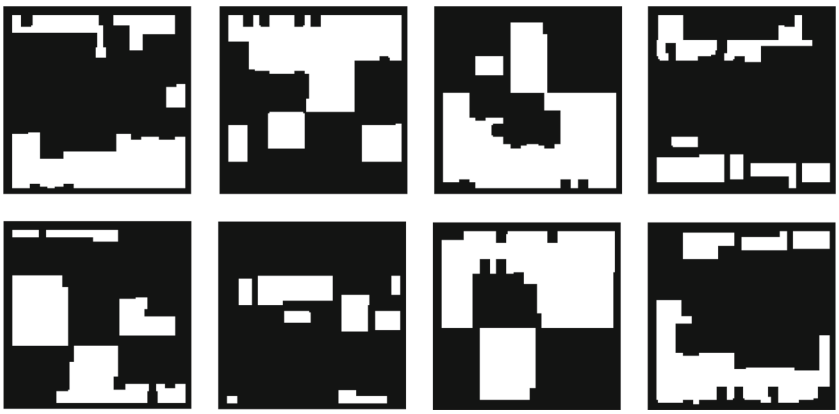

Fig. 8 Sample clips from training dataset

5.1 Dataset

Calibre DESIGNrev [25] software was utilized to extract a specific layer from a real design and save it as a GDS file. Subsequently, the layer within the GDS file was converted into an image format. This image was subsequently divided into smaller clips, each measuring 128×128 pixels in size. The size of the clip was chosen to be carrying a convenient number of polygons per clip. The dataset was then pre-processed to ensure that the splitting operation did not include small snippet of an adjacent polygon in the current clip. The clips were divided into a training set of size 5600 clips and a validation set of size 1400 clips. Figure 8 shows sample clips of the training dataset.

5.2 Pattern Generation

WGAN-GP described in the previous sections is implemented with Pytorch [22]. Having the GAN trained with the training dataset, 100K samples were generated by feeding random inputs sampled from a Gaussian distribution. The manual seed of the random input vector is changed with a randomly generated value to ensure the randomness and variety of the GAN's input. Figure 9 shows samples of the generated clips from the GAN. As observed, the generated clips include fuzziness and violations.

5.3 Pattern Legalization

After generating 100K samples from the GAN, these samples are fed to a sequence of legalization operations to remove the fuzziness of the generated samples and ensure the correctness of these samples.

5.3.1 Deblurring and Fuzziness Elimination

The convolutional auto-encoder, as described in the preceding sections, is implemented using PyTorch framework. The outputs of the GAN are fed to the convolutional auto-encoder. The results are non-fuzzy samples with only simple defects in the spatial domain. Figure 10 illustrates samples of the outputs of the convolutional auto-encoder.

5.3.2 Post-Processing

The post-processing operations, as outlined in the preceding sections, are implemented using the Python programming language with the assistance of OpenCV [23] library. Figure 11 illustrates the output of the post-processing step. As observed, the images are corrected on the pixel level, where all the shapes are perfect Manhattan shapes.

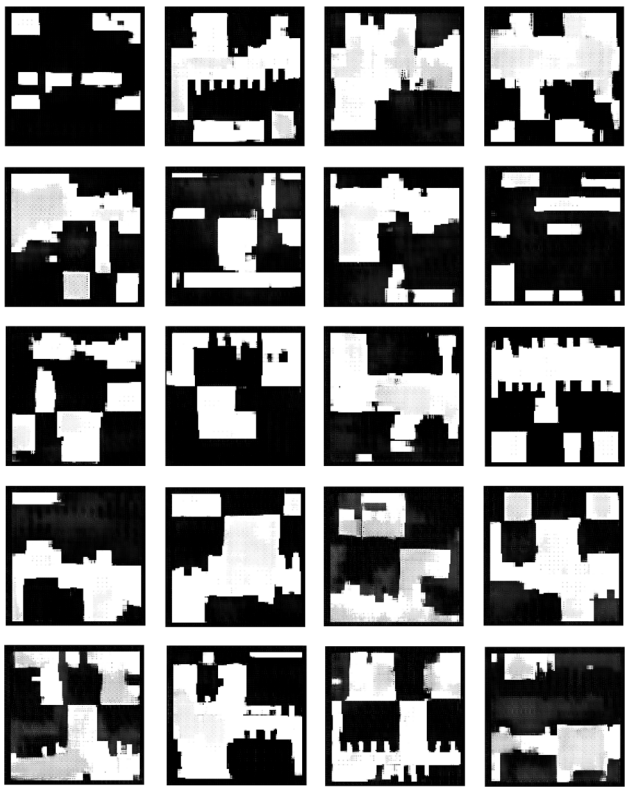

Fig. 9 Sample of the generated clips from WGAN-GP after training

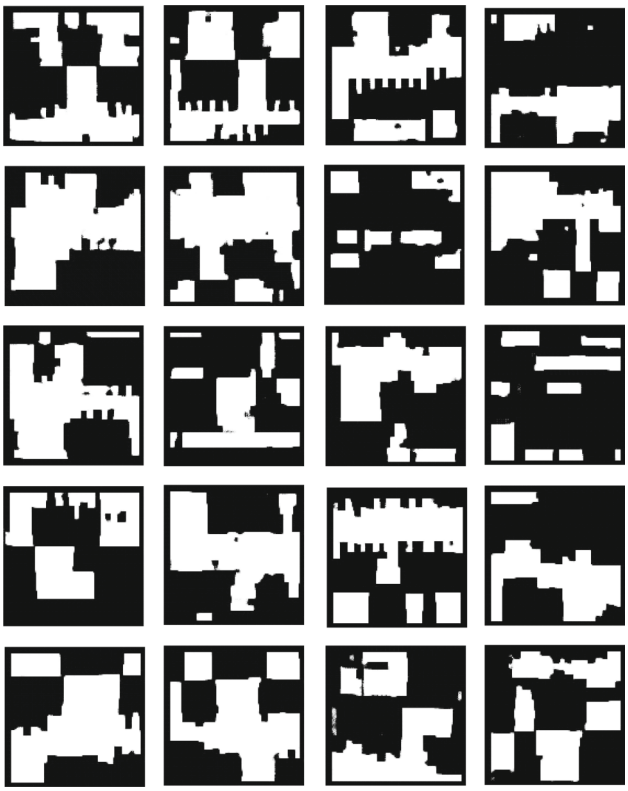

Fig. 10 Sample output of the deblurring convolutional auto-encoder

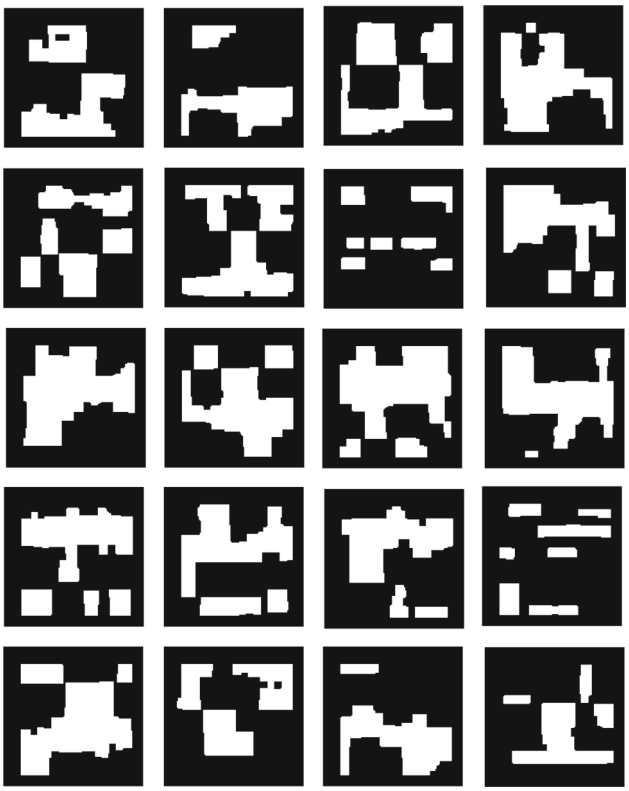

Fig. 11 Sample of the output of the post-processing step

Table 5 Percentage of legal patterns for different generation methods

Method	Percentage of Legal Patterns (%)
CAE	0.019
VCAE	2.126
CAE + LegalGAN	3.74
VCAE + LegalGAN	84.51
LayoutTransformer	89.726
WGAN-GP + Legalization (proposed methodology)	100

5.4 Evaluation

This section presents the results obtained from evaluating the 100,000 generated samples using the evaluation techniques described in the preceding sections.

5.4.1 Pattern Legality Results

Each clip among the 100,000 generated clips is converted to GDS format and passed through Calibre DRC [26] software to ensure that the DRC rules of the training set are not violated in these generated clips. The post-processing step was effective in eliminating all small polygons by utilizing a carefully selected set of hyperparameters. As a result, all of the generated clips successfully passed the DRC checks. Table 5, referencing results mentioned in [13], illustrates the percentage of legal patterns in the total number of generated patterns of this work compared to previous work.

5.4.2 Pattern Diversity Results

Shannon entropy was computed for three datasets: the training set, the generated clips, and the merged dataset comprising both the training set and the generated clips. The Shannon entropy values obtained are as follows: the Shannon entropy of the training set is 8.1, the Shannon entropy

Table 6 Ratio between diversity in generated clips to training clips across generation methods

Method	Diversity Ratio
CAE	0.351
VCAE	0.926
CAE + LegalGAN	0.5395
VCAE + LegalGAN	0.9155
LayoutTransformer	0.9768
WGAN-GP + Legalization (proposed methodology)	1.037

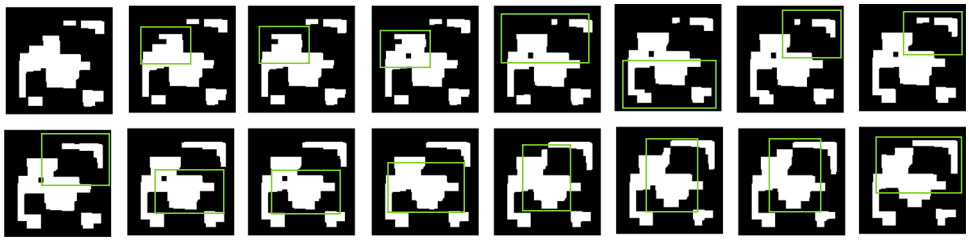

Fig. 12 Latent space morphing experiment: changes observed in the output clips from left to right

of the generated clips is 8.4, and the Shannon entropy of the combined dataset is 8.5.

This emphasizes the diversity of the generated samples where the generation methodology generates diverse samples. Table 6, referencing results mentioned in [13], shows a comparison between the ratio of the diversity of the generated clips to the diversity of the training clips using different methods. The proposed method has shown a higher diversity in the generated samples compared to the diversity of the training samples, whereas all the other methods have shown lower diversity in the generated set compared to the training set.

5.5 Limitations

The proposed methodology has certain limitations due to its high flexibility during the generation step. This flexibility can result in pixel-level defects that must be corrected during the legalization step to produce DRC clean test clips, as illustrated in previous sections. Although the legalization steps in this work have been successful, addressing defects can be particularly challenging when dealing with complex DRC rules, as this requires a more sophisticated legalization process.

5.6 Latent Space Morphing

This section showcases the responsiveness of the adopted WGAN-GP to variations in input. It highlights the generator's capability to produce diverse clips by making minor adjustments to the input, indicating that it does not suffer from mode collapse. It also demonstrates that the generator is not over-fitting on the entries of the training samples in the latent space of the training set. In this experiment, 100 samples from the training set are chosen randomly and embedded to their latent space using the trained convolutional auto encoder introduced in the preceding sections. To explore the diversity of outputs from the GAN, the embedding of each pair is morphed by taking incremental steps to generate 100,000 distinct input vectors from the original 100 image embeddings. These morphings involve small adjustments from one embedding to another, and the resulting outputs from each of these morphings are observed from the GAN. This process allows for the examination of the variations in the generated outputs across the different morphed input vectors. Figure 12 shows how the generator is very responsive to the change in the input vector, where a slight change corresponds to a slight change in the output clip moving from one embedding towards the other. Notice that the output clips are changing slowly from one vector towards the other showing high responsiveness to the small steps between them.

6 Conclusion

The application of generative machine learning techniques for generating synthetic layout patterns was shown to be successful. Despite the challenging legalization difficulties in post-generation legalization steps, the WGAN-GP has shown remarkable capabilities in producing a vast and diverse range of synthetic patterns, while avoiding common issues encountered by GANs, such as mode collapse, overfitting, and training instability.

Future work in this area can focus on enhancing the post-generation legalization steps to better accommodate more complex DRC rules in the generated patterns. Additionally, incorporating additional evaluation metrics to assess the quality and legality of the generated patterns could provide further insights and advancements in this field.

Availability of data and materials Data-sets generated during and/or analyzed during the current study are available from the author upon reasonable request.

Declarations

Competing interests The authors have no relevant financial or non-financial interests to disclose.

References

1. Park JW, Torres A, Song X (2018) Litho-aware machine learning for hotspot detection. IEEE Trans Comput-Aided Des Integr Circuits 37(7):1510–1514. https://doi.org/10.1109/TCAD.2017.2750068
2. Ding D, Wu X, Ghosh J, Pan DZ (2009) Machine learning based lithographic hotspot detection with critical-feature extraction and classification. In: 2009 IEEE International Conference on

IC Design and Technology, pp 219–222. https://doi.org/10.1109/ICICDT.2009.5166300
3. Madkour K, Mohamed S, Tantawy D, Anis M (2016) Hotspot detection using machine learning. In: 2016 17th International Symposium on Quality Electronic Design (ISQED), pp 405–409. https://doi.org/10.1109/ISQED.2016.7479235
4. Abdelghany H, Hooker K, Guajardo M, Lu C-C (2020) Implementing Machine Learning OPC on product layouts. In: Yuan C-M (ed) Design-Process-Technology Co-optimization for Manufacturability XIV, vol. 11328, p 1132805. SPIE, San Jose, California, United States. International Society for Optics and Photonics. https://doi.org/10.1117/12.2552398
5. Reddy GR, Xanthopoulos C, Makris Y (2018) Enhanced hotspot detection through synthetic pattern generation and design of experiments. In: 2018 IEEE 36th VLSI Test Symposium (VTS), pp 1–6. https://doi.org/10.1109/VTS.2018.8368646
6. Reddy GR, Madkour K, Makris Y (2019) Machine learning-based hotspot detection: Fallacies, pitfalls and marching orders. In: 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp 1–8. https://doi.org/10.1109/ICCAD45719.2019.8942128
7. Yang H, Pathak P, Gennari F, Lai Y-C, Yu B (2019) Deepattern: Layout pattern generation with transforming convolutional auto-encoder. In: Proceedings of the 56th annual design automation conference 2019, pp 1–6. https://doi.org/10.1145/3316781.3317795
8. Zhang X, Shiely J, Young EF (2020) Layout pattern generation and legalization with generative learning models. In: Proceedings of the 39th international conference on computer-aided design, pp 1–9. https://doi.org/10.1145/3400302.3415607
9. Gennari FE, Lai Y-C (2014) Topology design using squish patterns. US Patent No. 8,832,621 B1. Filed November 28, 2011; Granted September 9, 2014; Assigned to Cadence Design Systems Inc
10. Kareem P, Kwon Y, Shin Y (2020) Layout pattern synthesis for lithography optimizations. IEEE Trans Semicond Manuf 33(2):283–290. https://doi.org/10.1109/TSM.2020.2982989
11. Kareem P, Shin Y (2021) Synthesis of lithography test patterns using machine learning model. IEEE Trans Semicond Manuf 34(1):49–57. https://doi.org/10.1109/TSM.2021.3052302
12. Filitchkin P, Do T, Kusnadi I, Sturtevant JL, de Bisschop P, Van de Kerkhove J (2009) Contour quality assessment for OPC model calibration. In: Allgair JA, Raymond CJ (eds) Metrology, Inspection, and Process Control for Microlithography XXIII. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 7272, pp 72722. https://doi.org/10.1117/12.814662
13. Wen L, Zhu Y, Ye L, Chen G, Yu B, Liu J, Xu C (2022) Layoutransformer: generating layout patterns with transformer via sequential pattern modeling. In: Proceedings of the 41st IEEE/ACM International Conference on Computer-aided Design, pp 1–9. https://doi.org/10.1145/3508352.3549350
14. Hinton GE, Krizhevsky A, Wang SD (2011) Transforming auto-encoders. In: Artificial neural networks and machine learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21, pp 44–51. https://doi.org/10.1007/978-3-642-21735-7_6
15. Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv:1312.6114. https://doi.org/10.48550/arXiv.1312.6114 [stat.ML]
16. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, vol 27, pp 2672–2680. https://doi.org/10.48550/arXiv.1406.2661
17. Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: Medical image computing and computer-assisted intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, Springer, pp 234–241. https://doi.org/10.1007/978-3-319-24574-4_28
18. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, vol 30.https://doi.org/10.48550/arXiv.1706.03762
19. Arjovsky M, Chintala S, Bottou L (2017) Wasserstein generative adversarial networks. In: International conference on machine learning, pp 214–223. https://doi.org/10.48550/arXiv.1701.07875
20. Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC (2017) Improved training of Wasserstein GANs. In: Advances in neural information processing systems, vol 30. https://doi.org/10.48550/arXiv.1704.00028
21. Akiba T, Sano S, Yanase T, Ohta T, Koyama M (2019) Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining, pp 2623–2631. https://doi.org/10.1145/3292500.3330701
22. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp 8024–8035. https://doi.org/10.48550/arXiv.1912.01703
23. Bradski G (2000) The OpenCV library. Dr Dobb's J Software Tools 25(11):120–122125
24. Gabrielli LH Gdspy Library. https://github.com/heitzmann/gdspy
25. Siemens: Calibre DESIGNrev. Siemens EDA. Available: https://eda.sw.siemens.com/en-US/ic/calibre-design/interfaces/designrev/
26. Siemens: Calibre DRC. Siemens EDA. Available: https://eda.sw.siemens.com/en-US/ic/calibre-design/physical-verification/nmdrc/

Adel Mahmoud received the B.Sc. degree (First Class Hons.) in computer engineering from Ain Shams University, Cairo, Egypt, in 2019. Adel is currently pursuing the M.Sc. degree from Ain Shams University, Cairo, Egypt, focusing on applying deep learning in the development of EDA solutions. His current research interests include the application of machine learning techniques in physical verification solutions.

M. Watheq El-Kharashi received the B.Sc. (First Class Hons.) and M.Sc. degrees in computer engineering from Ain Shams University, Cairo, Egypt, in 1992 and 1996, respectively, and the Ph.D. degree in computer engineering from the University of Victoria, Victoria, BC, Canada, in 2002. He is currently a Professor of computer organization and a chair of the Department of Computer and Systems Engineering, Ain Shams University. He has published 180 articles in refereed international journals and conferences and authored two books and eight book chapters. His current research interests include advanced system architectures, especially networks-on-chip (NoC), systems-on-chip (SoC), and secure hardware. More specific interests include hardware architectures for networking (network processing units) and security; advanced microprocessor design, simulation, performance evaluation, and testability; and computer architecture and computer networks education.

Cherif Salama received his B.Sc. and M.Sc. degrees from the computer and systems engineering department of Ain Shams University (ASU) in 2001 and 2006 respectively. In 2010, he received his Ph.D. degree in computer science from Rice University, Houston, Texas. He worked as assistant professor in the Computer and Systems Engineering Department of ASU and as adjunct lecturer in the EELU, the MIU, and the GUC. He served as unit head of the Computer Engineering and Software Systems program at Ain Shams University (ASU). He is currently an associate professor and associate chair of the Computer Science and Engineering Department at The American University in Cairo (AUC). He was also the key person in the design and development of the Verilog Preprocessor funded by Intel through the SRC. His research interests and publications span a wide spectrum including computer architecture, CAD, hardware description languages, programming languages, parallel computing, and AI.