

## What can Colors and Shapes Tell about Generative Adversarial Networks?

Can Uzun


Altınbaş University, Turkey  
[can.uzun@altinbas.edu.tr](mailto:can.uzun@altinbas.edu.tr)

**Abstract.** The study aims to understand the how's and what's of creating an architectural dataset for generative adversarial nets through the evaluation of the effects of colors and shapes in image datasets on generative adversarial nets. Throughout the paper, six generative adversarial network training sessions are conducted on DCGAN and context-encoder algorithms with three different datasets having different complexities for colors and shapes. Firstly the color and shape complexities are analyzed for datasets. For color complexity, heuristic analyze is applied and for shape complexity, gray level occurrence matrix entropy which gives the textural complexity is utilized. In the end, the complexities and the training results are evaluated. Results show that color complexity has an important role for generative adversarial networks to generate colors correctly. Regularity in shape complexity /gray level co-occurrence matrix entropy distribution facilitates the algorithm training and shape generating processes.

**Keywords:** GAN, Color, Shape, Texture, Architectural Dataset

### 1 Introduction

Generative adversarial network (GAN) is an unsupervised, generative algorithm in which two functions are in a race to fool each other to generate the realistic results mimicing the training dataset (Goodfellow et al., 2014). GAN algorithm has taken its place as one of the most powerful algorithms in visual data generation. With this potential for visual data, the number of research on GAN within visual arts and architecture has increased (Shen et al., 2020; Chan & Spaeth, 2020). Despite numerous studies on GAN in architecture, there are few studies on the evaluation of GAN's working process, its outputs, its relation between the architectural training dataset (Xu et al., 2018; Borji, 2019; Gulrajani et al., 2020). In this paper, the focus will be on colors, shapes, and their relation with GAN efficiency to understand the



nature of dataset creation for design problems. Haralick et al. (1973) says shapes are a part of textural features. Throughout the paper, “shapes” will refer to the textural feature of the image dataset.

Color and textural/shape qualities are two important determinative properties of architectural space. Human perceives and experiences the architectural space with its color and textural features (Zobel, 1995). Besides having experiential effects, these architectural properties give the characteristic of the architecture (Wake& McCulloch, 1991). The textures and color scheme of baroque architecture are different from traditional Japanese architecture. Ottoman architecture has no common textural properties with modern architecture. Color and textural properties are inseparable qualities of architecture. Color gives substance, material information, but it is not enough by itself. Textural information is a complementary feature of architectural space. Gibson (1986) defines texture as centimetric and millimetric precision of a fine structure of the surrounding. According to Gibson (1986), the textural feature has two subclasses; layout texture, and pigment texture. The layout texture is three-dimensional topological features of the object, while pigment texture is insensitive to the three dimensions of the geometry. Pigment texture is the sum of color patterns. In this text, to understand the shape complexity the pigment texture quality of the images will be evaluated.

While creating an architectural dataset for training with GAN; color, and the textural qualities must be considered detailly. Not only do these qualities have the characteristic and perceptual experience of architectural space but also characteristics and perceptual features of the dataset will affect. The dataset is the input for generative networks. Training results/ outputs and efficiency of generative network will be affected by the characteristic of the architectural dataset.

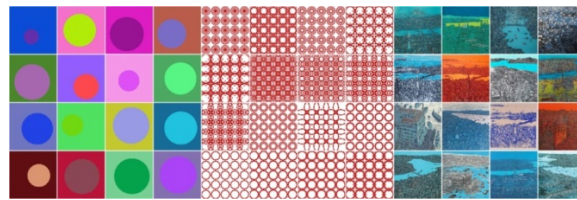
This paper aims to understand the effect of the shape complexity through textural feature complexity and color complexity on generative adversarial networks. So GAN training experiments will be conducted with datasets with varying color and shape complexities.

## **2 Methodology**

This study conducts six different training sessions with three datasets with different colors, shapes, and different textural complexity levels, with two different algorithms. Before the training sessions, the color and shape/ textural complexities of datasets were analyzed. After training the algorithms, the outputs and the training process are compared and evaluated to understand the effects of the color and the shape. Two different algorithms were trained for each dataset to check the reliability of correlation (if there is any correlation) between the algorithm efficiency and the color, shape/textural complexity features.

## 2.1 Dataset

Three datasets are utilized in the training sessions. The datasets are El Korci's (2020) colored circle dataset, red color repeating geometric patterns dataset, and artist Devrim Erbil's Istanbul abstract silhouettes dataset (Figure 1). Each dataset has a shape of  $300 \times 128 \times 128 \times 3$ . In this image shape, 300 denotes the number of data/images in one dataset while  $128 \times 128$  defines pixel resolution of one image, and 3 is for the red, green, blue (RGB) color channel.



**Figure 1.** From left to right: Circle dataset (El Korci, 2020), Pattern Dataset (Source: Author), Devrim Erbil Dataset (Erbil, 2020).

All these datasets look too different to compare. But when it comes to the inner structure of the images, we can analyze and compare these different datasets. First of the comparisons can be done with the colors. Color complexities can be easily compared by merely visual perception. The second feature can be the textural feature of the dataset to understand the shape complexity. But for shape complexity, rather than relying on perception, gray level co-occurrence matrix (GLCM) entropy is utilized. GLCM shows the complexity level of the textures on images.

**Color complexity:** The color complexity of each dataset is pure to perception. I simply count the colors to understand the color complexity of an image. If an image has only two colors then the color complexity is two. But if there are more than two colors, the color complexity defined as more than two. These color complexity definitions are enough to compare these datasets. But if datasets in training sessions were similar, this method could be problematic. On the other hand, to measure the color complexity of an image, I look at the color complexity per image in the dataset and for the entire dataset. A single image in a dataset may have only one color value. In the dataset, each image can be in a different color. This can increase the color complexity of the entire dataset.

The pattern dataset has only one color/red for all single images in the dataset. So the color complexity of the pattern dataset per image and for the entire dataset is the same and one. The color complexity of the Erbil dataset is more than two per image and for the whole dataset. The color complexity of the circle dataset per image is two, while the color complexity of the entire circle dataset is more than two. In descending order, the color complexity per image is Erbil, circle, and pattern dataset. For the color complexity of the entire dataset, Erbil and circle have equal values (more than two color

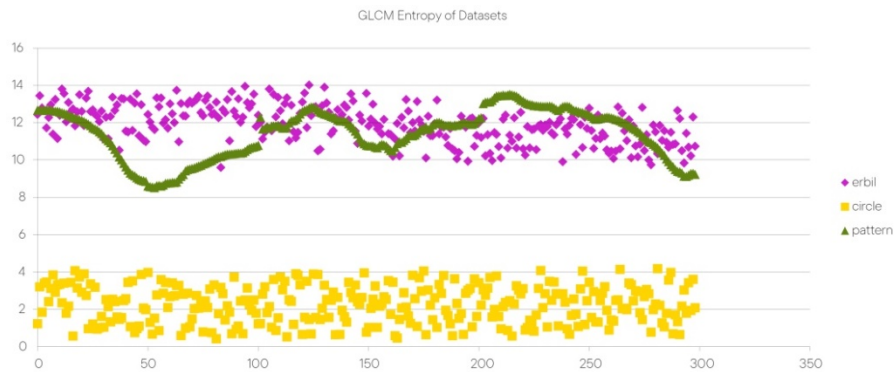
values), and the color complexities of the pattern dataset are the smallest (Table 1).

**Table 1.** The Color Complexity per Image and the Color Complexity for Entire Dataset

	Color Complexity per Image	Color Complexity for Entire Dataset
Circle	2	More than 2
Pattern	1	1
Erbil	More than 2	More than 2

**Shape/Texture complexity:** We may evaluate the complexity levels of shapes in the dataset heuristically. But the inner structure of the shape information may say something different. To be objective on the shape complexity analysis, I used GLCM entropy for all datasets. Haralick et al. (1973) support the idea that the classification of images can benefit from this textural feature of images. Gray level co-occurrence matrix entropy is one of the methods that qualify the complexity level of the textural information of image data (Haralick et al., 1973). To evaluate the shape complexity, the textural features are analyzed with GLCM entropy. GLCM is a calculation result of the distribution of co-occurring pixels in an image. Not only this occurrence is creating the textural feature, but it is the structure of the shapes in the image.

Figure 2 shows GLCM calculation results for all the images in three datasets. The values of GLCM entropy for Erbil and circle datasets values for all images spread all around the graph almost homogeneously. But the numeric values of GLCM entropy for the Erbil dataset are higher than the circle dataset. Pattern dataset has uniquely different values for GLCM entropy. All the GLCM values for the pattern dataset are lined on a curvy path rather than spreading all around the graph homogeneously. But the cumulative numeric values of the GLCM entropy of the pattern dataset are higher than the circle dataset and lower than the Erbil dataset. Numeric GLCM values in descending order for the shape/textural complexity among datasets are Erbil dataset, pattern dataset, and circle dataset.

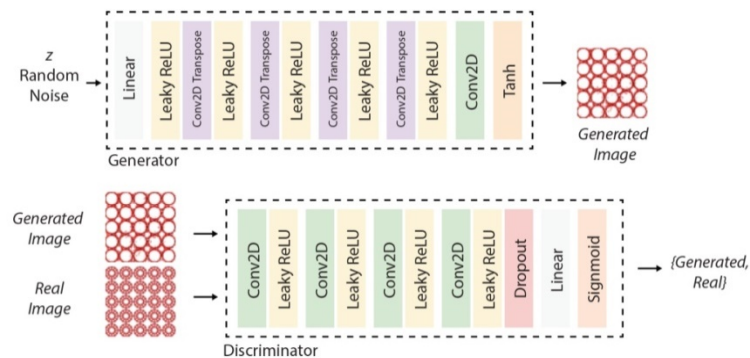


**Figure 2.** Shape Complexity Distrubution with GLCM Entropy. (Source: Author).

## 2.2 Algorithm Architecture

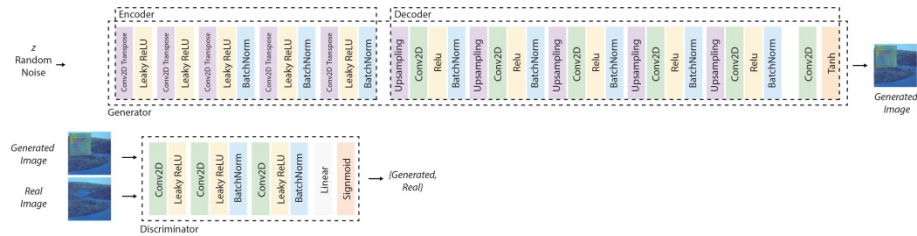
In the training sessions, DCGAN and context-encoder algorithms were utilized. Both these algorithms are generative algorithms that have both generator and discriminator functions. The discriminator function tries to recognize the real and fake images. The generator function tries to generate fake images to fool the discriminator function. If the generator function succeeds, the algorithm can generate fake but indistinguishable images from their originals. DCGAN generates the whole image surface. The context-encoder algorithm works in the masked part of the image and the context-encoder generates an image part coherent to the entire image.

**DCGAN:** The architecture of DCGAN has generator and discriminator functions which are created with convolutional layers. The generator function consists of four Conv2D transpose layers for upsampling the resolution and leakyReLU for activation function, in the last layer there is one Conv2D layer and tanh is used for the last layer activation function. Discriminator consists of Conv2D layers with leaky ReLU activation function with a dropout in the last layer and completed with sigmoid activation function (Figure 3).



**Figure 3.** DCGAN Architecture. (Source: Author).

**Context-Encoder:** Context encoder has generator and discriminator functions. The generator function has an encoder and a decoder part. The encoder part has five conv2D Transpose layers with LeakyReLU activation functions. The last three of these layers have also batch normalization. The decoder part has five upsampling layers and conv2D layer with ReLU activation function and Batch normalization. The last two of these layers have batch normalization. Sigmoid is used for the last layer's activation function (Figure 4).



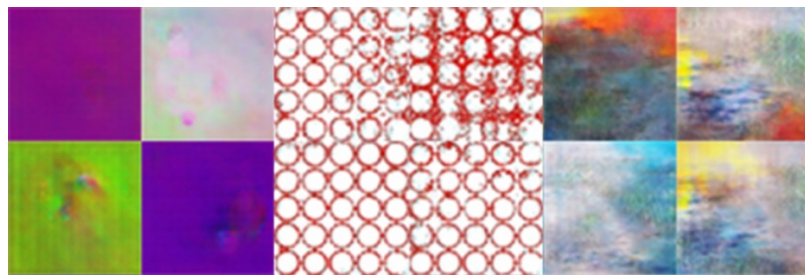
**Figure 4.** Context Encoder Architecture. (Source: Author).

### 2.3 Evaluation Methodology

The evaluation methods of the outputs of GAN algorithm are respectively; heuristic evaluation and Frechet Inception Distance (FID) score evaluation. The heuristic is a subjective and qualitative evaluation method, while FID score evaluation is objective and quantitative method.

## 3 Training Experiments

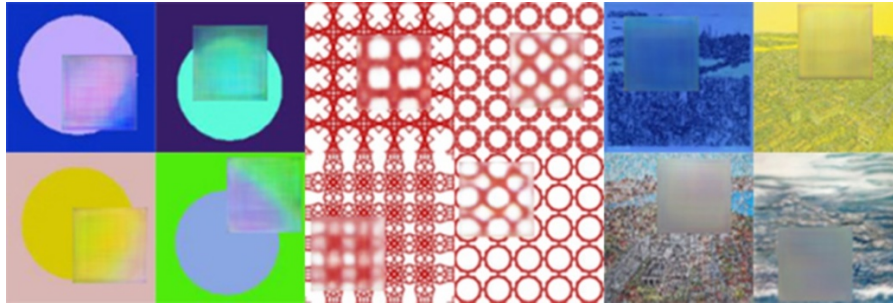
For all training experiments, Google Colab which is a free notebook for executing python codes was utilized. Each datasets were trained on tensor processing unit (TPU) service which Colab provides freely. With the help of TPU, all the training sessions took almost two hours. In both DCGAN and context-encoder training sessions, consecutively circle, pattern, and Erbil datasets are utilized. For each dataset algorithm was trained for 1000 epoch with a batch size of 30. For every ten epoch, generated visuals were saved, and for every epoch discriminator and generator loss values were saved. There was no issue detected as over-fitting, vanishing gradient, or mode collapse, which are the main problems of GAN algorithm. The loss values of discriminator and generator are mostly balanced.



**Figure 5.** DCGAN Outputs (After 1000 Epoch Training). (Source: Author).



Figure 5 and figure 6 show four samples for each training session results by DCGAN and context-encoder. In figure 5, DCGAN generated the entire visual scene while the context-encoder generated only the masked/hidden parts of the images. In figure 6, we can see some blurry square regions. These parts are the prediction of context-encoder for masked regions of the images.



**Figure 6.** Context Encoder Outputs (After 1000 Epoch Training). (Source: Author).

## 4 Colors and Shapes Evaluation

### 4.1 Heuristic Evaluation

Borji (2019) says that qualitative evaluation methods are biased and can be sometimes misleading however; these methods are often used for GAN evaluation as these methods are fast. So heuristic evaluation will be questionable, but it may give some quick perceptual insights for the evaluation of GAN besides the relation between color, shapes, and the GAN algorithm. Rating method is 2 and 3-point likert scale for evaluating and comparing the respectively color and shape outputs of the training sessions. If the algorithm is successful in generating colors, the algorithm gets one otherwise zero. For the success of the algorithm in generating the shapes, the rating scale starts with 0 (the worst session), continues with 1, and ends with 2 (the best session) (Table 2).

**Table 2.** Heuristic Rating for Success of Training Sessions

Heuristic Rating for Training Success	DCGAN		Context Encoder	
	Color	Shape	Color	Shape
<i>Circle Dataset</i>	0	0	1	1
<i>Pattern Dataset</i>	1	2	1	2
<i>Erbil Dataset</i>	0	1	1	0

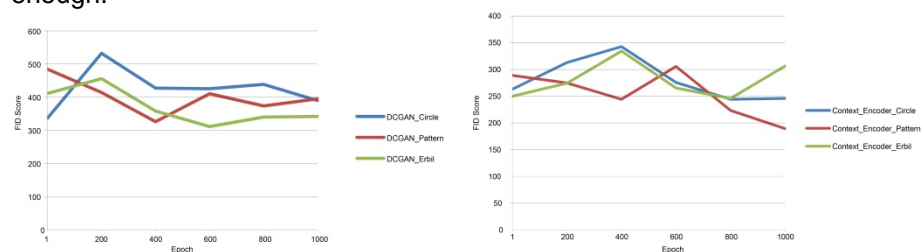
In DCGAN training sessions, the pattern dataset training is the most successful one in generating colors. Pattern dataset is the best one in shape generation too. Circle and Erbil datasets are not successful in generating colors, but the Erbil dataset is better than the circle dataset in generating “shapes. Heuristically in DCGAN training sessions, the best one is the pattern dataset, secondly, the Erbil dataset and the worst one is the circle dataset.

Context-encoder could generate the colors almost right for all datasets. But for the shape feature, the best one is the pattern dataset, and the worst one is the Erbil dataset. In context-encoder training, the best one is the pattern, the second, circle, and the worst one is the Erbil dataset.

We can see that the best training session for both algorithms is pattern dataset training. And the worst one changes accordingly to the training algorithm.

## 4.2 Frechet Inception Distance Evaluation

Frechet Inception Distance (FID) is one of the quantitative evaluation methods for GAN algorithms. FID compares the training data and the data generated by GAN, and this comparison result is a numeric value that shows how big the dissimilarity of generated image and the training data (Borji, 2019). FID algorithm takes two different datasets and calculates a feature probability distribution for each dataset (Brownlee, 2019). The difference between these probability distributions is the FID score. All the features (color values based on pixel locations) in an image-based dataset represent a vector. Training dataset and generated data have these features as vectors. The ‘distance’ between these vectors is called FID. If the FID score is big, the similarity between the two vectors is low. When the FID score is small, two vectors are similar to each other. But for good efficiency in the GAN algorithm, the FID score should not be zero. If FID score is zero, it means that during the training process, an over-fitting issue occurs. So we want FID score to be small enough.



**Figure 7.** FID Scores for DCGAN and Context-Encoder. (Source: Author).

For each training session, the FID scores are calculated for every 200 epoch (Figure 7). We can see that FID scores are quite big for each training session. The small sample size (generated and real data) can cause these big



numbers. But still, we can track the changes in FID scores by epoch in figure 8.

DCGAN\_Erbil training session obtained the least FID score and the DCGAN\_Pattern had the biggest FID score. For context-encoder pattern dataset had the smallest FID score, while Erbil having the biggest FID score. But we should look at the total change of FID score from the first epoch to the last epoch rather than only looking at the last score. Table 3 shows these differences between the first epoch FID and the last epoch FID scores.

**Table 3.** FID Scores by epoch and FID Difference Between First and Last Epoch

Algorithm	Dataset	FID Score by Epoch						FID Difference Between First and Last Epoch
		1	200	400	600	800	1000	
DCGAN	Circle	337,900	534,212	429,215	427,359	440,27	391,230	-53,330
	Pattern	486,226	415,775	327,558	412,028	375,279	396,768	89,458
	Erbil	412,930	457,620	359,923	312,867	341,697	343,409	69,521
Context_Encoder	Circle	262,603	312,473	342,179	275,255	243,686	245,450	17,153
	Pattern	288,268	274,358	243,693	305,042	222,902	188,839	99,429
	Erbil	249,544	273,685	333,749	264,861	245,287	305,929	-56,385

According to FID difference scores, in DCGAN training sessions, the pattern dataset is the best one. Circle training session becomes the worst one, moreover, after 1000 epoch, circle training FID scores increased. In context-encoder training, after 1000 epoch Erbil training session FID score increased. So the worst one is Erbil dataset training in context encoder. The second unsuccessful one is the circle training session, while the best training session is the pattern dataset training.

## 5 Results

Heuristic evaluation and FID scores say that regardless of DCGAN or context-encoder algorithm, the pattern dataset training gets the highest evaluation scores. But the lowest evaluation scores change accordingly to the algorithm type.

*Effects of colors:* In DCGAN training, color complexity affected color generation. It is obvious that if the color complexity is one, the algorithm can be successful in generating colors. If the color complexity is two or more than two per image or the entire dataset it may affect the color generation negatively in DCGAN. In context-encoder, color complexity for the entire

dataset is not that much important, as the algorithm is focusing on the context of the single image. So context-encoder is better than DCGAN in generating colors regardless of the color complexities. We can understand that algorithm has an important role in generating colors, secondly, the color complexity is affecting the outputs too. But it is not obvious how the color complexity per image or entire dataset or both of them affect the algorithms.

*Effects of shapes:* GLCM entropy shows the textural complexity levels. As textural complexity includes a spatial feature (co-occurrence of pixel values) it has information about shapes too. In shape generation, according to heuristic and FID score evaluation, the pattern dataset gets the highest scores in DCGAN and context-encoder sessions, even the pattern dataset has the second biggest GLCM entropy value. GLCM entropy results show that there is no direct relationship between the numeric GLCM entropy values with the algorithm outputs, instead, the distribution of the GLCM entropy values affects the training. GLCM distribution of the pattern dataset has a continuous path one after another that creates a regularity in change. In training sessions, if there is a regular change in the dataset, the algorithm training session became successful.

## 6 Conclusion

Results shows that the color and shapes complexities are critical for selecting the dataset for GAN training sessions. Color complexity has negative correlation between DCGAN outputs success but the correlation is weak in context encoder. For the shape complexity among DCGAN and context-encoder training sessions, rather than the numeric shape complexity values for each image, the regularity of the shapes in the entire dataset is affecting the generative algorithm.

This findings opens up valuable questions in terms of architectural dataset creation. What kind of color complexities do we have in architectural datasets? What is regularity in architecture? What is a regular architectural dataset? Is it possible to create a regular dataset of architecture? Is this regularity a trap for creating the same typological outputs for architecture with GAN algorithms? With the lead of all these results and questions, we can say the problem of complexity in architectural dataset is a potential and important research area for GAN algorithms in architecture.

## References

- Borji, A. (2019). Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179, 41-65.
- Brownlee, J. (August 30, 2019). How to Implement the Frechet Inception Distance (FID) for Evaluating GANs [Web blog]. Retrieved: November 11, 2019, from:

<https://machinelearningmastery.com/how-to-implement-the-frechet-inception-distance-fid-from-scratch/>

- Chan, Y. E., & Spaeth, A. B. (2020). Architectural Visualisation with Conditional Generative Adversarial Networks.-What machines read in architectural sketches. *Education and research in Computer Aided Architectural Design in Europe (eCAADe) Conference 2020*, Berlin, Germany.
- El Korchi, A. (2020). "2D geometric shapes dataset ", *Mendeley Data*, V1, DOI: 10.17632/wzr2yv7r53.1
- Erbil, D. (2020, February 8). *Yağlı Boyalar*. Retrieved October 7, 2021, from <https://www.devrimerbil.com/yagli-boyalar/>.
- Gibson, J. J. (1986). The ecological approach to visual perception. Westport, CT: Greenwood Press
- Gulrajani, I., Raffel, C., & Metz, L. (2020). Towards GAN benchmarks which require generalization. *arXiv preprint arXiv:2001.03653*.
- Haralick, R. M., Shanmugam, K., & Dinstein, I. H. (1973). Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6), 610-621.
- Goodfellow, I. Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2014). Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680.
- Shen, J., Liu, C., Ren, Y., & Zheng, H. (2020). Machine Learning Assisted Urban Filling. *Education and research in Computer Aided Architectural Design in Europe (eCAADe) Conference 2020*, Berlin, Germany.
- Wake, W. K., & McCullough, M. (1991, October). Architectural Tours through Texture Space. *In Realistic and Virtual Reality: ACADIA Conference Proceedings* (pp. 91-53).
- Xu, Q., Huang, G., Yuan, Y., Guo, C., Sun, Y., Wu, F., & Weinberger, K. (2018). An empirical study on evaluation metrics of generative adversarial networks. *arXiv preprint arXiv:1806.07755*.
- Zobel Jr, R. W. (1995). The representation of experience in architectural design. *Presence: Teleoperators & Virtual Environments*, 4(3), 254-266.