IADIS International Journal on Computer Science and Information Systems
Vol. 19, No. 2, pp. 87-101
ISSN: 1646-3692

CASE STUDY IN GENERATIVE ADVERSARIAL NETWORKS FOR TEXTILE PATTERNS GENERATION

Diogo Araújo[1], Rita Gomes[2], Ivan Gomes[2], Luís Romero[1] and Pedro Miguel Faria[1]
[1]Instituto Politécnico de Viana do Castelo, Portugal
[2]CITEVE – Textile and Clothing Technology Centre, Portugal

ABSTRACT

This study focuses on the implementation and evaluation of generative models for the generation of textile designs using Generative Adversarial Networks (GANs). The approach involved developing both unconditional and conditional versions of Wasserstein GANs (WGANs) and Wasserstein GANs with Gradient Penalty (WGAN-GP), as well as adaptations for higher resolution outputs. A diverse dataset of 13,000 textile patterns was compiled, and the models were trained on this data, with architectures designed to optimize image generation in terms of both resolution and feature learning. The training process was analyzed using loss stability assessments, visual evaluation, and accuracy metrics. Results showed that WGAN-GP models demonstrated greater loss stabilization but lower overall accuracy since the discriminator learned faster, while conditional models showed improvement in image fidelity but with some divergence issues during training. Additionally, efforts to upscale output resolution to 256x256 pixels were largely unsuccessful, with significant loss oscillations and poor constructed generated samples. This study concludes with recommendations for further refinement of the model architectures and training strategies to improve the generation of high-quality, high-resolution textile designs.

KEYWORDS

Textile Pattern, Textile Design, Generative AI, GAN, Conditional GAN

1. INTRODUCTION

The work described here is part of the TEXP@CT project, in the context of the work package on Digital Product and particularly Digital Tools for Creativity. The textile industry, often regarded as a bastion of creativity and artistry, is undergoing a transformative shift into the realm of knowledge-driven enterprises (Samia et al., 2022; Karegowda et al., 2024). While the artistic aspect of textile design remains paramount, the industry increasingly relies on technology, allocating substantial resources to stay at the forefront of innovation. The integration of digital tools, crafted by seasoned professionals, has become a defining feature of this evolution (Mishra et al., 2024). In this dynamic landscape, machine learning emerges as a powerful catalyst for change, offering an avenue to understand and replicate intricate design patterns. Machine learning, as a discipline, revolves around the identification of patterns within selected data to predict outcomes or generate new material. Recent advancements in technology and processing power have catapulted machine learning into various applications, spanning data classification, statistical analysis, image recognition, and the development of intricate systems. One notable offshoot of machine learning, deep learning, relies on artificial neural networks to unravel complex patterns (Dwivedi et al., 2016; Li, 2024). This includes specialized neural network architectures such as convolutional neural networks (CNN), generative adversarial networks (GAN) and recurrent neural networks (RNN), each designed to excel in specific applications. Generative Adversarial Networks (GAN) are based on neural networks used for the generation of images (Goodfellow et al., 2014). It mainly consists of two parts, a generator and a discriminator or critic. The generator generates images through latent space and noise as its weights. These generated images are passed onto a discriminator which compares the generated images with the real images and calculates the error/loss value. These loss functions are passed onto the generator and discriminator to fine-tune according to loss function to produce better images and making better predictions.

A method of textile image generation using Wasserstein Generative Adversarial Networks (WGAN) (Weng, 2019) and Wasserstein Generative Adversarial Networks with Gradient Penalty (WGAN-GP) (Gao, 2020) is introduced. According to our knowledge, few algorithms exist that are dedicated to generating proper textile patterns, resulting in the need for testing and evaluation of current models, and fine-tuning them. We have compared the performance of 2 image generative algorithms such as WGANs (Weng, 2019; Arjovsky et al., 2017a), WGANs-GP (Gao, 2020) and added the concept of conditionality through the embedding of categories/classes. These models were chosen because they represented significant advancements in the field of Generative Adversarial Networks (GANs) and have addressed some of the limitations associated with traditional GANs, such as:

- Stability and Convergence: Where WGAN addresses the issue of training instability and mode collapse that can be observed in traditional GANs. It introduces a more stable training procedure by replacing the Jensen-Shannon divergence with the Wasserstein distance in the objective function.
- Wasserstein Distance: The use of Wasserstein distance allows for a more meaningful and continuous measure of the difference between the generator and real data distributions. This helps prevent gradients from disappearing or exploding during training.
- Gradient Penalty: While WGAN improves stability, WGAN-GP further refines the training process by incorporating a gradient penalty. The gradient penalty addresses the problem of weight clipping used in WGAN, which can lead to suboptimal convergence. WGAN-GP penalizes the norm of the gradient of the critic (discriminator) with respect to its input, enforcing smoother convergence.
- Mode Collapse Mitigation: WGAN-GP has been observed to be less prone to mode collapse, when compared to some traditional GAN formulations. Mode collapse occurs when the generator collapses to produce a limited set of similar samples.

This article is divided into five sections: the present one which introduces the study carried out, followed by a section of related work about some utilities of the GANs, possible activities in the range of image manipulation and some case studies of GANs, in the textile pattern generation; then the implementation section describes a prepared dataset and explains the process of building a GAN model and its conditionalization; the next section refers to the analysis of obtained results, through visual and loss stability assessments; and the last section indicates some conclusions about the study presented here.

2. RELATED WORK

Extensive research has been done in the field of GANs from generating simple images as simple as generation of bird images (Huang, 2018) to more complex subjects such as medical studies (Cepa et al., 2023) and complex designs (Liu et al., 2020). In the last few years, the evolution of machine learning algorithms and deep learning methods has resulted in different applications and has shown great results in many industries (Li, 2023). In general, the generative model can perform tasks like reasoning, density estimation, and sampling. The generation model can be separated into unconditional generation and conditional generation, based on the various inputs (Yoshimura and Kasahara, 2016). Image synthesis is an important research area within the field of computer vision (Yi et al., 2023). Its primary purpose is to transform images between different image domains, including applications such as image super-resolution generation, image coloring, image filling, style transformation, and attribute transformation. (Aittala et al., 2016) uses CNN to learn the mapping between input images and output images, and (Ren et al., 2017) decomposed the hidden space of an image into a domain-invariant content space and a domain-specific style space and obtained multimode output. A deep network model based on edge enhancement is proposed for image super-resolution reconstruction, and a deconvolution network is used to achieve the goal of edge enhancement.

Considering textile design, (Yar et al., 2023) introduces the use of Generative Adversarial Networks (GANs), leveraging the image generation capabilities of GANs, the study curates a dataset of 17,000+ textile design images, categorized by design class. The authors employ a Deep Convolutional GAN (DCGAN) to collectively generate patterns and fine-tune it for specific pattern types, such as cheetah. The study claims three main contributions: dataset collection and classification, the introduction of GANs in textile design, and the training of a versatile DCGAN for efficient pattern generation. As for (Fayyaz et al., 2020), the study proposes an innovative approach to automated textile design pattern generation using generative models, claiming that the accuracy of classifying textile design patterns is enhanced by 2%, through data cleaning and pseudo labelling. Performance comparisons among Wasserstein Generative Adversarial Networks Gradient Penalty (WGANs GP), Deep Convolutional GANs (DCGANs), and Convolutional Variational Autoencoders (CVAEs) are conducted on a dataset for individual classes, assessed using the inception score.

3. IMPLEMENTATION

An experimental study was carried out. First, it was necessary to acquire and process data related to the problem of generating textile designs. Then, the desired algorithms were implemented, and an architecture was built, in order to be able to train the models.

3.1 Data Acquisition

Before we started the process of textile patterns image generation, we had the need for a data set about textile patterns. We used a dataset made from different sources such as TexGan (Yar et al., 2023), clothing-pattern-dataset (Alexander et al., 2021) and some hand-picked images from sources such as Freepik, Shutterstock and Vecteezy. The dataset consists of 13,000 images, divided by categories, as exemplified in Figure 1.

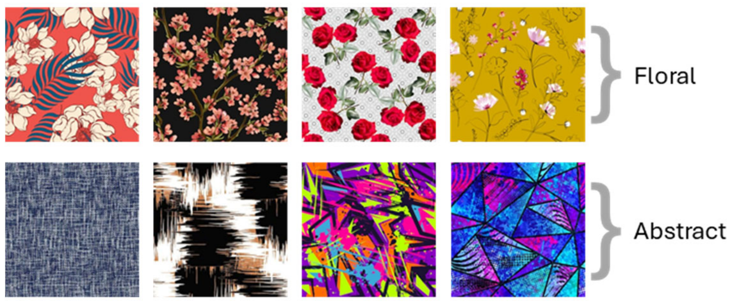

Figure 1. Four pattern examples, from the dataset, for the floral and abstract categories

The dataset was built using images of different resolutions, and we pre-processed all images before feeding the data models. As the images were loaded, we resized them to 64×64 pixels dimension and established the color channels as 3 (RGB), due to the lack of some computational power, to experiment with larger images.

3.2 Base Algorithms

For the purpose of this study, the developed models were based on the proposed algorithms represented in Figure 2, from (Arjovsky et al., 2017b), and represented in Figure 3 from (Gulrajani et al., 2017). Some considerations were also based on (Soumith and Denton, 2017) GAN-hacks.

Following Figure 2, for each step the generator takes, during the training process, the discriminator will take n steps. As in basic GANs, we take the latent noise sample and the dataset sample to compute the gradient. RMSProp is the optimizer used and it is applied to the clipping technique, which clips every weight parameter in the network to be conformed within a certain range. Then the generator is trained, computing the gradient, and using the same optimizer.

Following Figure 3, for each step the generator takes, during the training process, the discriminator will take n steps. Instead of clipping, it is introduced the concept of gradient penalty where it is added the interpolation of the generated sample and the dataset sample, and computed the norm of the gradient, which is desired to be approximately 1 for every interpolation, to satisfy the Lipschitz constraint. The optimizer used in this case is Adam.

Simply put, the biggest differences between the algorithms are the introduction of clipping and gradient penalties that attempt to enforce the Lipschitz condition differently, and the difference between optimizers.

```
Require: : $\alpha$, the learning rate. $c$, the clipping parameter. $m$, the batch size.
    $n_{\text{critic}}$, the number of iterations of the critic per generator iteration.
Require: : $w_0$, initial critic parameters. $\theta_0$, initial generator's parameters.
 1: while $\theta$ has not converged do
 2:     for $t = 0, ..., n_{\text{critic}}$ do
 3:         Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
 4:         Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
 5:         $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
 6:         $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
 7:         $w \leftarrow \text{clip}(w, -c, c)$
 8:     end for
 9:     Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
10:     $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
11:     $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
12: end while
```

Figure 2. WGAN proposed algorithm (Arjovsky et al., 2017b)

```
Require: The gradient penalty coefficient $\lambda$, the number of critic iterations per generator iteration
    $n_{\text{critic}}$, the batch size $m$, Adam hyperparameters $\alpha, \beta_1, \beta_2$.
Require: initial critic parameters $w_0$, initial generator parameters $\theta_0$.
 1: while $\theta$ has not converged do
 2:     for $t = 1, ..., n_{\text{critic}}$ do
 3:         for $i = 1, ..., m$ do
 4:             Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
 5:             $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
 6:             $\hat{\boldsymbol{x}} \leftarrow \epsilon\boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
 7:             $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
 8:         end for
 9:         $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m}\sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$
10:     end for
11:     Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^m \sim p(\boldsymbol{z})$.
12:     $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m}\sum_{i=1}^m -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
13: end while
```

Figure 3. WGAN-GP proposed algorithm (Gulrajani et al., 2017)

3.3 Wasserstein Generative Adversarial Network (WGAN)

Based on the WGAN algorithm structure, we developed a Wasserstein Generative Adversarial Network model, where a specific generator and discriminator architecture was structured, taking into consideration that the generator will enlarge a certain input and try to construct features, while the discriminator will shrink the input, trying to detect features. As defined in the algorithm, the optimizer used was RMSprop and for each discriminator iteration, we clamped the weight clip of 0.01.

3.3.1 Generator Architectures

The generator architecture has 4 deconvolutional blocks (deconvolutional layer, normalization and ReLU activation function) and a final deconvolutional layer with ReLU activation function. Each deconvolutional layer has an input (number of channels in the image), output (produced number of channels), kernel size (convolving kernel size), stride (for cross-correlation), padding and bias. During the generation process, the generator takes and goes through the following number of channels (initial noise, 1024), (1024, 512), (512, 256), (256, 128) and (128, 64). The kernel size, stride, padding and bias are set as 4, 2, 1, false.

3.3.2 Discriminator Architectures

The discriminator architecture has an initial convolutional layer with LeakyReLU activation function, 3 convolutional blocks (convolutional layer, normalization and LeakyReLU activation function) and a final convolutional layer. Each convolutional layer has an input (number of channels in the image), output (produced number of channels), kernel size (convolving kernel size), stride (stride of the convolution), padding and bias. During the discrimination process, the discriminator takes and goes through the following number of channels (64, 64), (64, 128), (128, 256), (256, 512) and (512, 1). The kernel size, stride, padding and bias are set as 4, 2, 1, false.

3.4 Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP)

Based on the WGAN-GP algorithm structure, we developed a Wasserstein Generative Adversarial Network with Gradient Penalty model, where a specific generator and discriminator architecture was structured. As established in the algorithm, the optimizer used was Adam with betas of (0.0, 0.9). and for each discriminator iteration, we calculated the gradient penalty through the interpolated images and respective mixed scores, ending up multiplied by 10 and added to the discriminator loss.

3.4.1 Generator Architectures

The generator architecture has 4 deconvolutional blocks (deconvolutional layer, normalization and ReLU activation function) and a final deconvolutional layer with ReLU activation function. Each deconvolutional layer has an input (number of channels in the image), output (produced number of channels), kernel size (convolving kernel size), stride (for cross-correlation), padding and bias. During the generation process, the generator takes and goes through the following number of channels (initial noise, 1024), (1024, 512), (512, 256), (256, 128) and (128, 64). The kernel size, stride, padding and bias are set as 4, 2, 1, false.

3.4.2 Discriminator Architectures

The discriminator architecture has an initial convolutional layer with LeakyReLU activation function, 3 convolutional blocks (convolutional layer, normalization and LeakyReLU activation function) and a final convolutional layer. Each convolutional layer has an input (number of channels in the image), output (produced number of channels), kernel size (convolving kernel size), stride (stride of the convolution), padding and bias. During the discrimination process, the discriminator takes and goes through the following number of channels (64, 64), (64, 128), (128, 256), (256, 512) and (512, 1). The kernel size, stride, padding and bias are set as 4, 2, 1, false.

3.5 Conditional WGAN and Conditional WGAN-GP

Following the models i.e., basic conditional adaptations of the models were developed. For these, we introduced categories/classes, as each image in the dataset belongs to a class such as

Floral, Geometric, Cultural, Abstract, etc. Concerning the generator and discriminator Architectures, for this case, we kept both models' architectures i.e., but we embedded the existing classes of the dataset and added the respective class of each image, into the generator and discriminator. This way, both Generator and Discriminator will take into consideration this new characteristic.

3.5.1 Generator Architectures

Both models keep the same generator architecture, while implementing little alterations for the embedding. The embedding takes the classes and the embedding dimension and embeds in each image the labels, creating a new tensor, and with the objective of learning to generate more specific images guided by the label. During the generation process, the generator takes and goes through the following number of channels (initial noise + embedding dimension, 1024), (1024, 512), (512, 256), (256, 128) and (128, 64).

3.5.2 Discriminator Architectures

Both models keep the same discriminator architecture, while implementing little changes for the embedding. In this case, the embedding follows basically the same structure as in the generator and tries to associate the label with the extracted characteristics. During the discrimination process, the discriminator takes and goes through the following number of channels (64 + 1, 64), (64, 128), (128, 256), (256, 512) and (512, 1).

3.6 Conditional WGAN with 256x256 Output

According to the Conditional Wasserstein Generative Adversarial Network model i.e., we made some adaptations for the 256x256 resolution output instead of 64x64 as per the other 4 models. Considering that the amount of time needed for training grows exponentially based on the resolution and characteristics/features, we experimented with 2 different sets of values (normal and lowered sets).

3.6.1 Generator Architectures

The generator architecture follows the same pattern as the 64x64 conditional WGAN, with the addition of 2 extra blocks. During the generation process, for the lowered set, the generator takes and goes through the following number of channels (initial noise + embedding dimension, 256), (256, 128), (128, 64), (64, 32), (32, 16), (16, 8) and (8, 4). For the normal set, the generator takes and goes through the following number of channels (initial noise + embedding dimension, 1024), (1024, 512), (512, 256), (256, 128), (128, 64), (64, 32) and (32, 16).

3.6.2 Discriminator Architectures

The discriminator architecture follows the same pattern as the 64x64 conditional WGAN, with the addition of 2 extra blocks. During the discrimination process, for the lowered set, the discriminator takes and goes through the following number of channels (64 + 1, 8), (8, 16), (16, 32), (32, 64), (64, 128), (128, 256) and (256, 1). For the normal set, the discriminator takes and goes through the following number of channels (64 + 1, 16), (16, 32), (32, 64), (64, 128), (128, 256), (256, 512) and (512, 1).

3.7 Training Hyperparameters and Resources

Several hyperparameters were defined for every model with the output of 64x64 resolution. The computational resources consisted of a laptop with a professional mobile graphics chip by NVIDIA: Quadro RTX 4000 MAX-Q Design, a mobile processor with 8 cores: Intel I7-10875H and 32GB RAM. In terms of functional hyperparameters, we defined the necessary ones as follows:

- Learning rate = 5e-5
- Batch size = 64
- Image size = 64
- Color channels = 3
- Noise dimension = 100
- Discriminator characteristics = 64
- Generator characteristics = 64
- Discriminator iterations = 5
- Number of cycles = 200
- Embedding noise dimension = 100

After the initial experiments, we had some issues with the computational resources, so we transferred the models to a new laptop with the Apple M3 Pro chipset model and 32GB RAM. With these new resources we trained a CWGAN model for the 256x256 output resolution. As such, we decided to use the same hyperparameters, introducing only 2 changes:

- Image size = 256
- Number of cycles = 150

4. RESULTS ANALYSIS

Following, the training parameters i.e., for each epoch, the model weights were stored and can be implemented to transfer learning. The results were evaluated through visual and loss stability assessment.

4.1 Unconditional Models

For the base models (unconditional), where the models (WGAN and WGAN-GP) where trained while ignoring any type of labeling, resulting in the generation of random images, we saved and assessed the convergence of the generator and discriminator losses. We also tried to assess the visual output to identify characteristics.

4.1.1 Loss Stability Assessment

Knowing that, in practical terms, optimizers try to minimize loss functions, we can compare the performance of the discriminator and the generator. Therefore, an analysis was carried out on the losses of each model throughout the training process, as observed in Figure 4. The discriminator loss aims to maximize the mean of scores for real samples (encouraging correct classification of real samples) and minimize the mean of scores for synthetic samples (encouraging correct classification of fake samples). The generator loss aims to maximize the mean of discriminator scores assigned to its synthetic samples, thereby encouraging the generator to produce realistic samples that fool the discriminator.

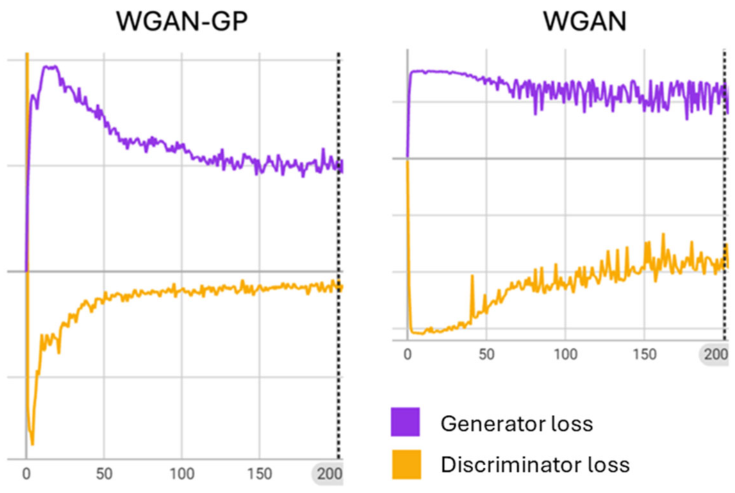

Figure 4. Losses during training of both unconditional models

For the WGAN model, the losses gradually decreased over time, whoever the stabilization seems lacking in comparison with the WGAN-GP model, where the convergence appears to be cleaner and stable. This, however, does not mean that the results obtained are better. Even if we consider that the WGAN-GP model had better stabilization, the discrepancy between the loss values indicates that the Generator performed poorly, when compared to the Discriminator. This means that, refinement is needed to reduce the discrepancy. Overall, both Generator and Discriminator have done their job, and the accuracy calculated (probability of the synthetic image being recognized as real) was on average 35% for the WGAN model, and 10% for the WGAN-GP model.

4.1.2 Visual Assessment

Based on the performed training, Figure 5 shows 64 images generated by the models, after training for 200 cycles. The first set of 32 images were generated by the WGAN model, while the other set were generated by the WGAN-GP model.

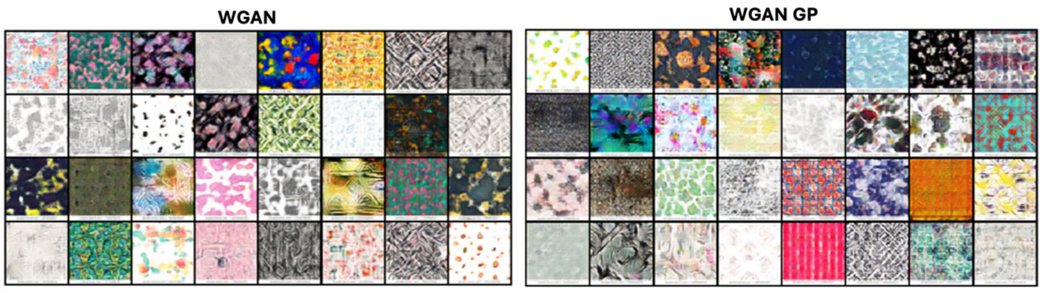

Figure 5. The 32 generated images from each model

From the beginning of this study, we expected that the lack of a higher resolution would make the visual comparison, between the models, more complicated and less viable. As such, taking into consideration that the images are 64x64, some generated examples, from both models, appear to resemble some characteristics of the dataset, even if mostly hard to perceive, as observed in Figure 6.

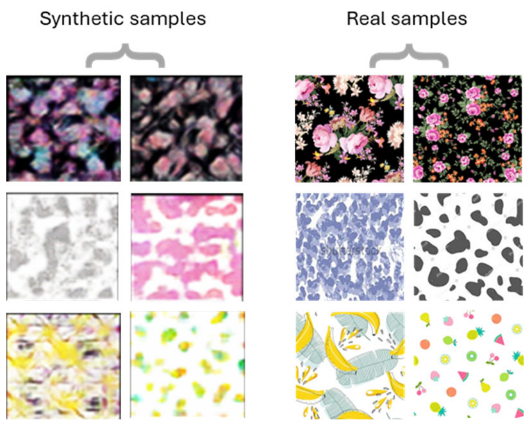

Figure 6. Synthetic and real samples using the WGAN model

4.2 Conditional Models

For the conditional models, where the models (WGAN and WGAN-GP) where trained, while embedding labels that represent the category/class that each image belongs to, resulting in the generation of images belonging to a specific category, we saved and assessed the convergence of the losses and the visual outputs of some of the categories.

4.2.1 Loss Stability Assessment

For the Conditional WGAN model, as represented by Figure 7, the losses gradually decreased over time, but the losses oscillations are not desirable. The accuracy calculated as an average of 40%. This indicates that the conditionalization made the accuracy go up by 5%, but still, the accuracy is considerably low, which shows that there is a lot of room for improvement.

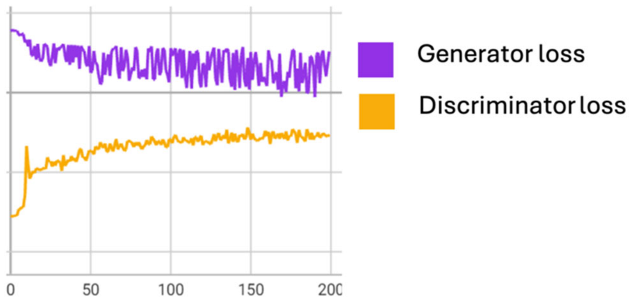

Figure 7. Losses during training of the CWGAN model

For the Conditional WGAN-GP model, represented in Figure 8, the losses presented an unexpected behavior, where the generator loss started to increase and diverge, from the 50 cycle forward. The main cause could be the fast decrease in discriminator loss, making the generator unable to maintain the discriminator. As a result, the discriminator will easily identify the synthetic images, as shown through the accuracy, which was also calculated with an output always around 0%.

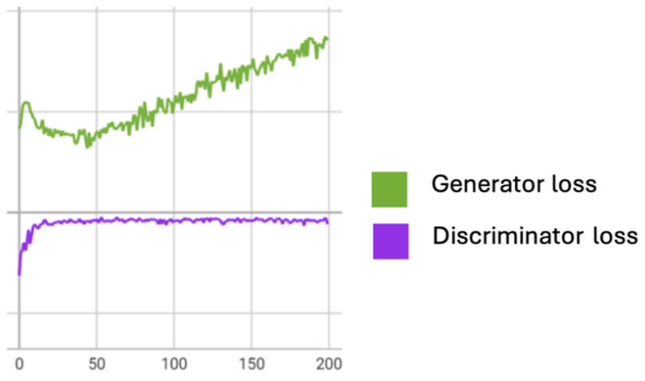

Figure 8. Losses during training of the CWAGAN-GP model

4.2.2 Visual Assessment

Figure 9 shows the generated images by the conditional models. The first set of 16 images were generated by the Conditional WGAN model, while the other set by the Conditional WGAN-GP model. The samples were generated by inputting the desired category, and as such, most of the samples show some characteristics of the class they belong to, even if hard to perceive as it lacks detail and resolution.

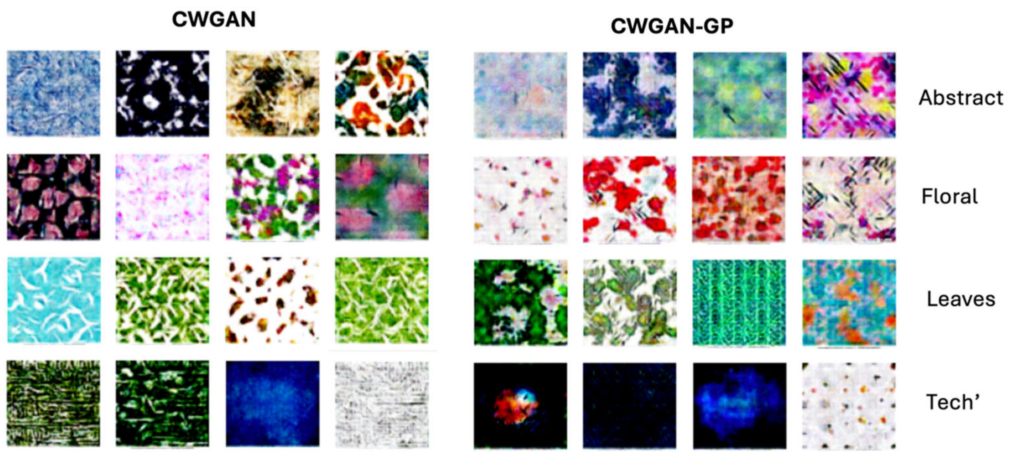

Figure 9. The 4 generated samples per selected class

The first set of samples were easy to pick, since the generator didn't deviate much in quality. As for the second set, the generator had a hard time generating consistent samples. This may be caused by the training of the WGAN-GP model, where it showed some unexpected behavior, as mentioned in the stability assessment. Figure 10 presents some synthetic and real images that shows some levels of similarity.

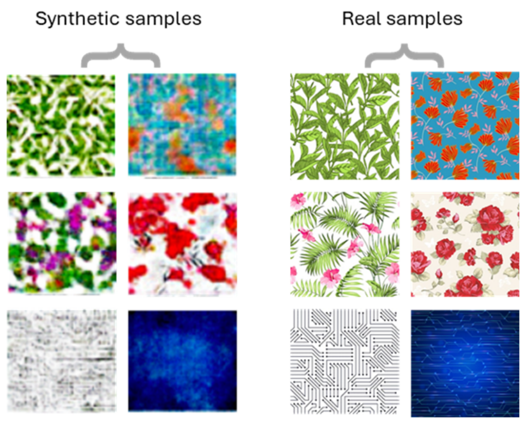

Figure 10. Synthetic and real samples using the WGAN-GP model

4.3 Conditional WGAN 256x256

For the Conditional WGAN model with 256x256 output with both lower and normal values, we followed the same training pattern as in the 64x64 model with some simple changes.

4.3.1 Loss Stability Assessment

As represented by Figure 11, the losses of the lower value model showed that the training did proceed somewhat since there is convergence, even if the oscillation is severe. The generation's results were poor, but we expected that, since we had lowered the values.

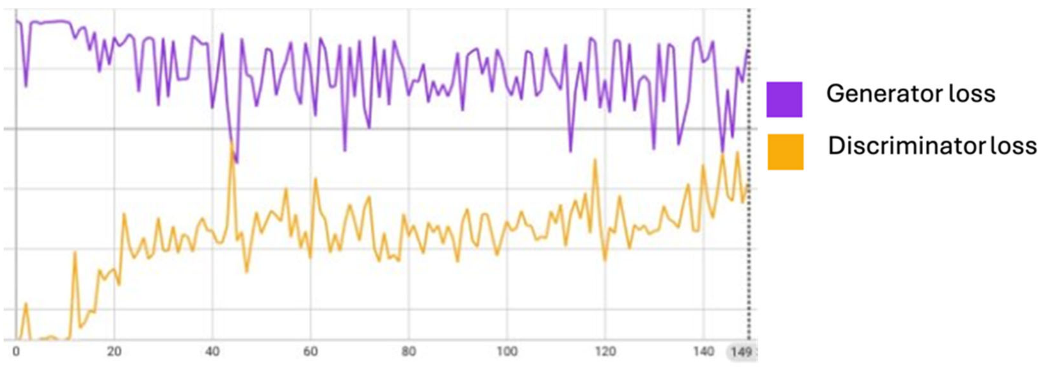

Figure 11. Losses during training of the CWGAN (256x256) model with lower values

For the normal values, represented in Figure 12, the losses presented an unexpected behavior, where both losses seemed stagnant. From these results we jumped into the generation of samples to verify if the outputs were at least acceptable.

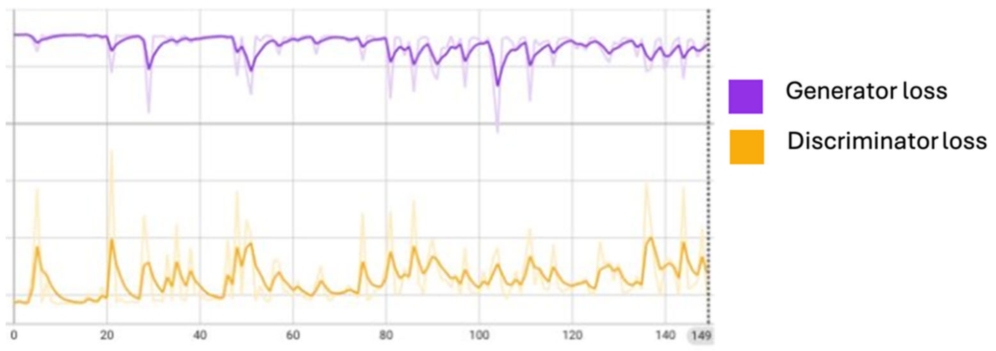

Figure 12. Losses during training of the CWGAN (256x256) model with normal values

4.3.2 Visual Assessment

Figure 13 shows the generated samples under the two sets of values i.e. The first set of 3 images was generated by the lower set, while the other set was generated by the normal set of values. With a higher resolution, while maintaining the same model structure, we expected the results not to change much, compared to the 64x64 output with the normal values, but, as with the loss assessment, the results were not acceptable.

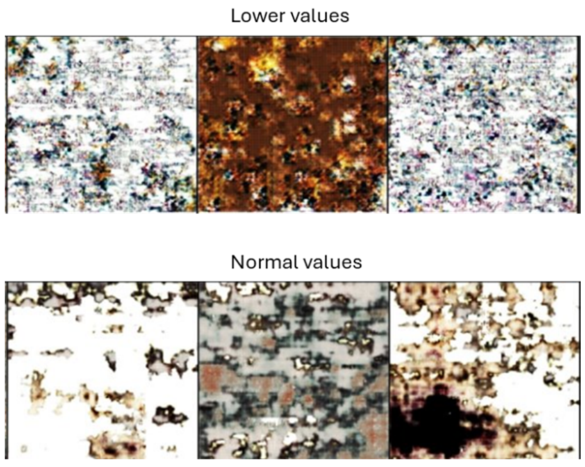

Figure 13. The 3 generated samples per value set

The first set of samples where, even when we expected it to be bad, was not acceptable, since we can actually try to recognize any features. As for the second set, the conclusion was the same because the results were poor. Even considering that the output resolution was correct, the overall experiment with 256x256 was a failure, because the generated samples did not present any similarity in characteristics to the real samples.

5. CONCLUSIONS

In this article, we implemented two models to solve the problem of unique textile pattern generation. As the results show, both models were functional, but there is a lot to take into consideration going forward. Using higher resolutions is quite necessary, if we want to do visual assessments properly, which in this case we could not analyze any type of refined details. Also, for both models, the losses absolute value gradually lowered, which is desired, but the WGAN-GP model needs extra refinement so the Generator can keep up with the Discriminator. Based on the accuracy calculated and the overall results, there is a long way to go until we reach the desired performance. Based on the accuracy results, knowing that the conditionalization of the WGAN model upgraded in average 5%, it indicated a positive value. Considering the experimentation failure using the 256x256 resolution, we intend to retest the training, after running some verifications on the architectures that seem to be wrongly built.

Since we lack computational resources for big experimentations, we will focus on a unique category of pattern and change our approach into the problem, taking in consideration various characteristics of the patterns such as composition, key elements, style, and padding, to verify if it can further upgrade the performance of the model. Also fine-tuning, in general, is needed for the aim of 90%+ accuracy and high-quality samples, for such we will completely change the architecture of the generator and discriminator to fit the needs of the problem. We also intend to get opinions from professionals in the field of textile design to better understand future results.

ACKNOWLEDGEMENT

This work has received financial support from integrated project TEXP@CT Mobilizing Pact – Innovation Pact for the Digitalization of Textiles and Clothing (TC-C12-i01, Sustainable Bioeconomy No. 02/C12-i01/202), promoted by the Recovery and Resilience Plan (RRP), Next Generation EU, for the period 2021 – 2026.

REFERENCES

Aittala, M., Aila, T. and Lehtinen, J., 2016. Reflectance modeling by neural texture synthesis. *ACM Transactions on Graphics*, Vol. 35, No. 4, 65.

Alexander, J. et al., 2017. Recognizing Clothing Colors and Visual Textures Using a Finger-Mounted Camera. *An Initial Investigation, Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility - ASSETS '17*, pp. 393-394.

Arjovsky, M., Chintala, S. and Bottou, L., 2017a. *Wasserstein GAN* [online]. http://arxiv.org/abs/1701.07875

Arjovsky, M. Chintala, S. and Bottou, L., 2017b. Wasserstein Generative Adversarial Networks. *Proceedings of the 34th International Conference on Machine Learning - Proceedings of Machine Learning Research*, Vol. 70, pp. 214-223. Available at: https://proceedings.mlr.press/v70/arjovsky17a.html

Cepa, B., Brito, C. and Sousa, A., 2023. Generative Adversarial Networks in Healthcare: A Case Study on MRI Image Generation. *2023 IEEE 7th Portuguese Meeting on Bioengineering (ENBENG)*, Porto, Portugal, pp. 48-51. doi: 10.1109/ENBENG58165.2023.10175330

Dwivedi, A. K., Tirkey, A., Ray, R. B. and Rath, S. K., 2016. Software design pattern recognition using machine learning techniques. *IEEE Region 10 Conference (TENCON)*, Singapore, pp. 222-227. doi: 10.1109/TENCON.2016.7847994

Fayyaz, R. A., Maqbool, M. and Hanif, M., 2020. Textile Design Generation Using GANs. *2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, London, ON, Canada, pp. 1-5. doi: 10.1109/CCECE47787.2020.9255674

Gao, X., Deng, F. and Yue, X., 2020. Data augmentation in fault diagnosis based on the Wasserstein generative adversarial network with gradient penalty. *Neurocomputing*, Vol. 396, pp. 487-494. https://doi.org/10.1016/j.neucom.2018.10.109

Goodfellow, I. J. et al., 2014. Generative Adversarial Nets. *Advances in Neural Information Processing Systems*, Vol. 3, pp. 2672-2680.

Gulrajani, I. et al., 2017. Improved Training of Wasserstein GANs. *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*. Available at: https://proceedings.neurips.cc/paper_files/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf

Huang, Z., 2018. Bird Generation with DCGAN. *Stanford CS230 Deep Learning*. Available at: https://cs230.stanford.edu/projects_spring_2018/reports/8289337.pdf

Karegowda, A. G., Pooja, R., Rani, A. L. and Devika, G., 2024. Detection of Stain Defects in Textile Industry using State-of-Art Transfer Learning Models. *International Conference on Smart Systems for applications in Electrical Sciences (ICSSES)*, Tumakuru, India, pp. 1-6, doi: 10.1109/ICSSES62373.2024.10561384

Li, M., 2023. Research on Intelligent Clustering of Textile Fabric Pattern Based on K-Nearest Neighbors Algorithm. *IEEE 3rd International Conference on Electronic Communications, Internet of Things and Big Data (ICEIB)*, Taichung, Taiwan, pp. 400-403, doi: 10.1109/ICEIB57887.2023.10170418

Li, M., 2024. Research on Textile Pattern Recognition Based on Artificial Intelligence. *IEEE 7th Eurasian Conference on Educational Innovation (ECEI)*, Bangkok, Thailand, pp. 335-338, doi: 10.1109/ECEI60433.2024.10510796.

Liu X., Huang, H. and Wu, H., 2020. Intelligent generation algorithm of ceramic decorative pattern. *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, Baltimore, MD, USA, pp. 122-126. doi: 10.1109/BigDataSecurity-HPSC-IDS49724.2020.00031.

Mishra, A. et al., 2024. GANs and Augmented Reality in Virtual Clothing Try-On. *International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, Bangalore, India, pp. 1-6. doi: 10.1109/IITCEE59897.2024.10467813

Ren, R., Gu, L., Fu, H. and Sun, C., 2017. Super-resolution algorithm based on sparse representation and wavelet preprocessing for remote sensing imagery. *Journal of Applied Remote Sensing*, Vol. 11, No. 2.

Samia, B., Soraya, Z. and Malika, M., 2022. Fashion Images Classification using Machine Learning, Deep Learning and Transfer Learning Models. *7th International Conference on Image and Signal Processing and their Applications (ISPA)*, Mostaganem, Algeria, pp. 1-5. doi: 10.1109/ISPA54004.2022.9786364

Soumith, M. and Denton, M., 2017. How to Train a GAN? Tips and tricks to make GANs work. *GitHub* [online]. Available at: https://github.com/soumith/ganhacks

Weng, L., 2019. *From GAN to WGAN*. doi: https://doi.org/10.48550/arXiv.1904.08994

Yar, G. et al., 2023. TexGAN: Textile Pattern Generation Using Deep Convolutional Generative Adversarial Network (DCGAN). *2023 IEEE International Conference on Emerging Trends in Engineering, Sciences and Technology (ICES&T)*, Bahawalpur, Pakistan, pp. 1-6. doi: 10.1109/ICEST56843.2023.10138848

Yi, M. et al., 2023. Research on Artificial Intelligence Art Image Synthesis Algorithm Based on Generation Model. *2nd International Conference on 3D Immersion, Interaction and Multi-sensory Experiences (ICDIIME)*, Madrid, Spain, pp. 102-106. doi: 10.1109/ICDIIME59043.2023.00026

Yoshimura, M. and Kasahara, S., 2016. Enhanced ray tracing algorithm to generate penumbra by point light. *Journal of Environmental Engineering (Transactions of AIJ)*, Vol. 81, No. 724, pp. 81-572.