Fabric Sketch Augmentation & Styling via Deep Learning & Image Synthesis

Omema Ahmed, Muhammad Salman Abid(✉), Aiman Junaid, and Syeda Saleha Raza

Habib University, Karachi, Pakistan
{oa04320,ma05045}@alumni.habib.edu.pk, aj05161@st.habib.edu.pk, saleha.raza@sse.habib.edu.pk

Abstract. This paper introduces a two-fold methodology of creating fabric designs and patterns, using both traditional object detection and Deep Learning methodologies. The proposed methodology first augments a given partial sketch, which is taken as an input from the user. This sketch augmentation is performed through a combination of object detection, canvas quilting, and seamless tiling, to achieve a repeatable block of a pattern. This augmented pattern is then carried forward as an input to our variation of the pix2pix GAN, which outputs a styled and colored pattern using the sketch as a baseline. This design pipeline is an overall overhaul of the creative process of a textile designer, and is intended to provide assistance in the design of modern textiles in the industry by reducing the time from going to a sketch to a pattern in under a minute.

Keywords: deep learning · GAN · image synthesis · quilting · textile pattern · fabric pattern · object detection

1 Introduction

The textile industry has been the rampart of Pakistan's economy. It contributes more than 60% to the total export earnings of the country, accounts for 46% of the total manufacturing and provides employment to 38% of the manufacturing labor force and 9% of GDP [1]. However, fashion design in Pakistan is more or less an entirely manual process, with little to no automation involved. It requires hours of manual labor to design a prototype pattern, and it goes through multiple iterations and feedback loops to get it finalized. The pattern itself also has to be designed from scratch, and its quality is entirely contingent upon the human designer.

This paper addresses both aspects of the problem in fashion designing in Pakistan; speeding up the design process and bringing these sketches to life. This is similar to how Microsoft developed an AI to reduce the time taken to design Indian jewelry designs [10], and this paper applies a similar approach on the domain of fashion design.

C. Johnson et al. (Eds.): EvoMUSART 2023, LNCS 13988, pp. 327–340, 2023.
https://doi.org/10.1007/978-3-031-29956-8_21

In this paper, image quilting techniques are applied to generate an augmented sketch from a given input sketch, in the form of a single flower or a collection of multiple flowers. For the second part of the problem, Conditional Generative Adversarial Networks (CGAN) are used to make these augmented sketches come to life by reproducing a photo-realistic rendition [3] in the form of an actual fabric pattern.

The rest of the paper is organized as follows: Sect. 2 explores the existing work done in this domain. Section 3 presents the preparation of the dataset used, and Sect. 4 details the methodologies employed in this paper. Section 5 gives a summary of our results, showcasing how the sketch has evolved from a given input. Finally, Sect. 6 concludes the paper, and Sect. 7 mentions the possible future work in this domain.

2 Related Work

The task of generating a realistic image from a sketch is a very challenging prospect, and has been the focus of research of various researchers in academia. In their paper [9], Gao et al. proposed a new neural architecture called EdgeGAN, which is used for automatic image generation from scene-level freehand sketches. A paper by Tian et al. [20] focuses on the colorization of images of logo sketches, to automate the process of manual logo and icon design. This paper builds upon the work done by Pix2Pix [12], with their contributions present in the architecture to adapt to the task of logo colorization. One similar project on Image-to-Image translation has also been done by Fan et al. [7] using Conditional GANs on single objects. Another paper by Hu, M., Guo, J [11] implements a convolutional generative network with encoder-decoder architecture and facial attributes classifier, that colorizes sketches of humans faces.

All the works currently done in the domain of sketch-to-image translation are restricted to either single objects and scenes, or facial features, both of which have a structural similarity in the objects to be colorized. This paper applies the Pix2Pix model [12] in the domain of fabric patterns.

According to the previous literature, a lot of work has been done on Image-to-Image translation using GANs. Different neural nets with conventional and hybrid models are used to generate photo-realistic Images. The implementations become challenging if user sketch input is taken into account to synthesize augmented textures. Some work on texture synthesis is presented below.

Alexei A. Efros and William T. Freeman [6] proposed in their paper 'Image Quilting for Texture Synthesis' a simple yet effective image-based method for generating novel textures. The quilting algorithm takes an input image and synthesizes a new, visually appealing, texture by stitching together small patches of an existing image. First, the image to be synthesized undergoes a raster scan to find a block that satisfies the overlap constraint. Once a block is found, the surface error between the new block and the old blocks is computed. The boundary of the new block is then determined by calculating the minimum cost path along the surface, and finally the chosen block is pasted onto the final texture.

These steps are repeated for every block until a high-quality image is obtained. In addition to this, an extended algorithm for texture transfer is also presented in the paper which involves rendering an image in the style of a different image.

Another similar paper on texture synthesis has been published by Alexei A. Efros and Thomas K. Leung [5], where they propose a non-parametric method. This method synthesizes a new texture from an initial seed, one pixel at a time. Their algorithm uses the Markov random field model to find similar neighborhoods in the input image to the pixels neighborhood. Then, it randomly selects one such neighborhood and takes its center to be the newly synthesized pixel. This method generates synthetic textures by preserving as much local structure as possible.

An interesting approach taken by Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum in their paper [15] is to use a real-time patching algorithm for synthesizing high-quality textures. The model first computes the local conditional probability density function (PDF) for each patch from the source image. The pixels are then synthesized incrementally on to the output texture. Lastly, feathering is used to blend the patch boundaries and synthesize a high quality texture.

The proposed pipeline for this paper is novel in its approach by being one of the first attempts to utilize a variety of objects for Image-to-Image translation, particularly in images dealing with sketches. Existing solutions are mostly focused on a specific category or object; a drawing of a piece of furniture, or a sketch of a fruit, whereas this work combines multiple categories of sketches into its supported function. Users may draw a sketch of a flower, a leaf, a butterfly, or all three objects combined, and it would still be able to generate a viable design.

For sketch augmentation, there are little to no existing works related to completing, or rather augmenting, user sketches. This part of the pipeline had to be built from scratch; the idea of using objects from a partial pattern, and then supplementing those with similar objects on a new canvas, was left unexplored in current literature. There have been some attempts at auto-completion of a given incomplete object [13,21], however very limited work is done to generate, or extend, existing images/objects in this domain. This paper proposes an algorithm that can leverage these image processing techniques, augmenting a partial pattern input by the user into a structure of a repeatable sketch.

3 Dataset

The dataset has been sourced from various open-source collections, and has been processed to convert it into a usable form for our CGAN model. Our dataset comprises approximately 1500 images, with the final dataset including $\sim$ 500 images from all three categories.

3.1 Data Collection

The dataset was scraped from Patternbank, The Sketchy Database and Kaggle Dataset for butterfly, floral and leaf categories. The collected dataset was then manually filtered to discard any images present which did not belong to the aforementioned categories. The filtered dataset then undergoes image resizing to be converted into images of dimensions 256 × 256, to be passed as training input to our CGAN model. A few examples of the collected dataset are shown in Fig. 1.

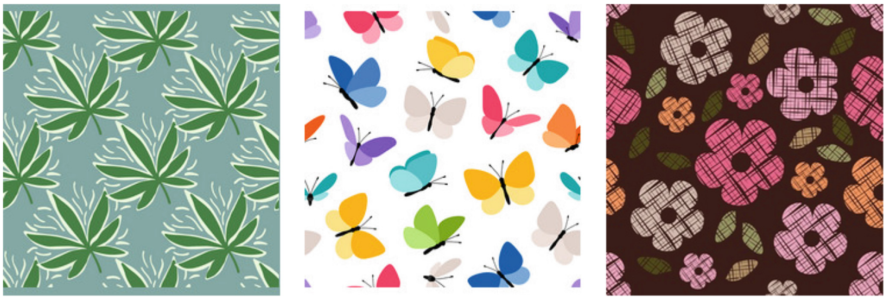

Fig. 1. Examples of the collected dataset, which includes images of three categories: leaf, butterfly and floral

3.2 Data Processing

In order to train the CGAN image-to-image translation model, images from the collected dataset are converted to their corresponding sketches through various image processing techniques.

For this purpose, Canny Edge Detection [2] is applied to all the images after they are converted to gray-scale. The resulting images are then combined together, including both their original pattern and the corresponding sketches. An example from the resultant combined dataset is shown in Fig. 2.

4 Methodology

This section provides an overview of the methodology to generate photo-realistic images from an input sketch. The process comprises the following two steps: user sketch to sketch augmentation, and augmented sketch to Photo-Realistic conversion.

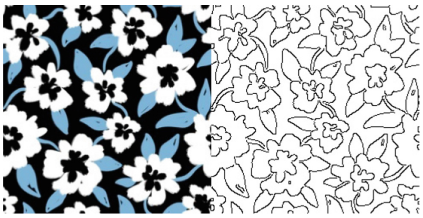

Fig. 2. Example of application of Edge Detection on source (left) with its result (right)

4.1 Sketch Augmentation

The first step in augmentation is further divided into the following steps:

- Object Detection
- Content-Based Image Retrieval
- Texture Synthesis

The following sub-sections will describe, in detail, what process each step follows.

Object Detection. This technique is used to extract objects from the input sketch and store them as separate images. The following steps are done to perform object detection:

- Load the source image and apply background removal to extract useful pixels from the image.
  Background removal removes any white/non-transparent pixel(s) from the sketch that are not part of the drawing.
- Convert image to gray-scale, and apply a filter for Gaussian blurring
- Perform Canny Edge Detection [2] and dilate [4] image to increase the size of the foreground object.
- Find contours and iterate through the loop to extract region of interest (ROI) i.e. objects in the source image.
- Save each extracted object in their respective file format. i.e. (.png or .jpg)

Content-Based Image Retrieval. CBIR is a technique which searches for similar images in our local dataset based on a given source image. The process begins with extracting features from both the input image and the dataset using a feature extraction algorithm. Then, it calculates the similarity distance between images present in the dataset and the query image, finally retrieving all similar images with the lowest distance from the dataset.

Our problem uses CBIR to retrieve similar objects from the dataset to the ones present in our input image. It uses a pre-trained, VGG-16 CNN model [19] to extract features from the image dataset and stores them in a feature vector. The previously separated objects are then imported as input images in the retrieval algorithm. After performing feature extraction, and comparing them with the previously saved feature vectors, we compute the distance between the query image and the images in the dataset using the L2 Norm [16]. The shorter the distance between images, the more similar they are to each other. Lastly, two images with the lowest distance are then retrieved from the dataset for further use in augmentation.

The image retrieval algorithm proceeds as follows:

- Apply feature extraction using VGG-16 on local dataset
- Apply feature extraction on input image
- Calculate the similarity between the preceeding two vectors using L2 Norm

– Search for the most similar results (lowest distance ranked highest) and return top two

Additionally, even if object retrieval is not used, the algorithm is designed to add a random shape object with random size from our fixed set of shapes – currently limited to stars and circles – to the design. This makes the generated design more stochastic, and results in more organic patterns. Some examples of randomly added shapes can been seen in Figs. 3 and 7. This is an optional feature, and can be toggled according to user preference.

Texture Synthesis. A final synthesized texture with 256×256 dimensions is augmented from the processed images using the following techniques:

Grid Texture Synthesis. Generate arbitrarily large textures by stitching together patches of the same size in a grid-like form. The process involves two user controlled parameters: source image and size, where source image is the user drawn sketch, and size is the total number of patches on the final image in each row/column. Once user inputs are defined, the next step is to calculate the block size by dividing the dimensions of the final image with the input size. Here, the block size B determines the size for each patch to be placed on the augmented image.

From the feature arrays obtained through feature extraction on both the local dataset and the input image, pick a random object and apply image resizing to obtain dimensions of $B \times B$. This ensures that every patch is of same size and a uniform grid can be obtained. The patches are placed side-by-side on the final image, forming a structured grid of objects as the final output.

To the final image, edge enhancement filters are applied to enhance contour fidelity and overall quality. To cater to the transparent background of each patch image while pasting, a bitmap is generated from the alpha channel of the image, where *true* indicates all non-transparent pixels and $false$ otherwise. Lastly, using the bitmap, all *true* pixels of the array are set to black and all others to white, outputting a black-and-white augmented sketch.

Random Texture Synthesis. Generate patterns by randomly stitching together different patches of varying sizes. Using the objects from Object Detection and Content-Based Image Retrieval, a random object is chosen to be placed on the canvas. Different transformation techniques are applied to the selected object, which include:

– Rotation: The patch undergoes any one of the four randomly selected rotation methods; horizontal flip, vertical flip, mirroring and counterclockwise rotation.
– Resize: The patch dimension is randomly resized in the range of $[200 - 600]$.

Additionally, the algorithm also checks for an existing patch while pasting a new object on the canvas, to ensure that there are no visually-jarring overlaps in the final sketch. To do this, once a random position on canvas is selected to paste the new object, the image pixels of the new location on canvas is checked for any existing black image pixels, and if there are none present, the new object is pasted. Otherwise, another location is randomly selected for the object. This process keeps repeating until an empty space is found. However, in case the locations are chosen more than 10 times for one patch, the process is halted, assuming that there are already enough objects present on the canvas. An example of the augmented pattern produced from given sketch is shown below.

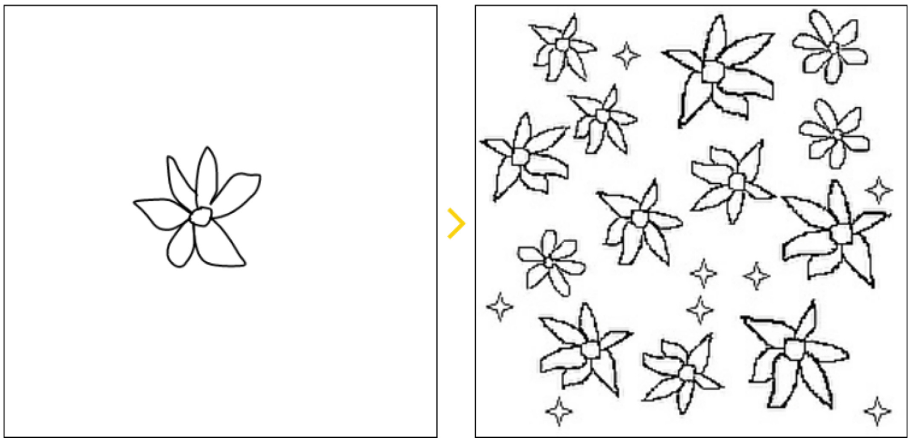

Fig. 3. Sketch Augmentation using Random Texture Synthesis on a sample sketch (left) with its result (right)

4.2 Sketch to Photo-realistic Image

This paper proposes a design generation technique based on Generative Adversarial Networks (GANs), which consist of two neural net models; a generator and a discriminator. It is a slight modification from the original pix2pix structure. The model architecture is detailed below:

Generator. The generator in the model is based on a modified U-Net architecture, adopted from Ronneberger et al. [18], comprising of an encoder and a decoder. Each block in the encoder includes a convolution layer, followed by batch normalization and activation function, Leaky ReLU. Each block in the decoder includes a transposed convolution layer, followed by batch normalization, dropout layer, and an activation function, ReLU. The loss function for the generator is binary cross-entropy loss and MAE (mean absolute error).

Discriminator. The discriminator in our CGAN is a convolutional PatchGAN classifier, where each block consists of a convolution layer, followed by batch normalization and an activation function, Leaky ReLU. The loss function for the discriminator is the binary cross-entropy loss.

We also make use of the Adam optimizer [14], with the learning rate set for both the generator and discriminator at 2e−4.

The images used for our training dataset go through several pre-processing techniques, as described in Sect. 3. These include random jittering and mirroring of the images to introduce stochastic behavior, and normalizing the images in the range of $[-1, 1]$ for them to be compatible with the neural-net layers.

For the purposes of training the GAN, we combine each pattern in the dataset with its sketch output, and concatenate the two images side-by-side. This represents, for each image, the domain mapping that has to be performed for the sample with the sketch and the ground-truth pattern representing their domains respectively.

The model has the following parameters for training:

– EPOCHS: 10,000
– BATCH SIZE: 4
– IMG WIDTH: 256px
– IMG HEIGHT: 256px

These parameters were tuned to get the best approximations of the ground-truth images as a result of prediction on user sketches. Post-training, the model was able to successfully generate a stylized image as a result of a user sketch input. An example of the generated styled image after sketch augmentation is shown in Fig. 4.

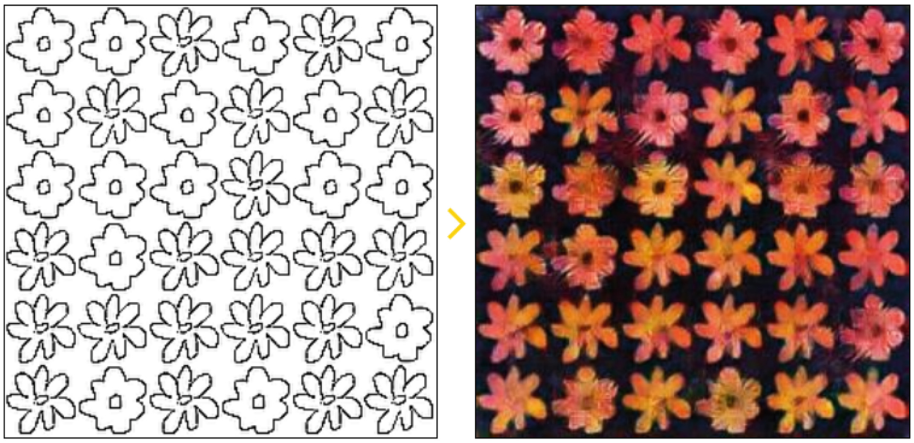

Fig. 4. Augmented image (left) converted into a photo-realistic image (right) via pix2pix

4.3 Tiling

The process of tiling makes use of an open-source library by Artëm G., named img2texture [8]. It creates a single tile-able block from a source image using image overlapping, to hide the seams when the image is tiled using multiple copies concatenated with each other to form a larger, repeating pattern. The parameters of the tiling process include the tile size (an N×N image tile), and the overlap percentage that controls the degree of overlap in the source image. Results from this tiling process are shown in Sect. 5.

Although similar to the augmentation process in Sect. 4, this process provides a birds-eye view on the final colourised image rather than the original sketch. Fabric patterns are better evaluated when they can be viewed on a larger-scale, hence the option to be able to visualize this on a dimension of the user's choosing.

5 Experiments and Results

This section contains experimentation results with different parameters, which includes sketches created using Grid Style (deterministic block placement) and Random (random block placement) Augmentation.

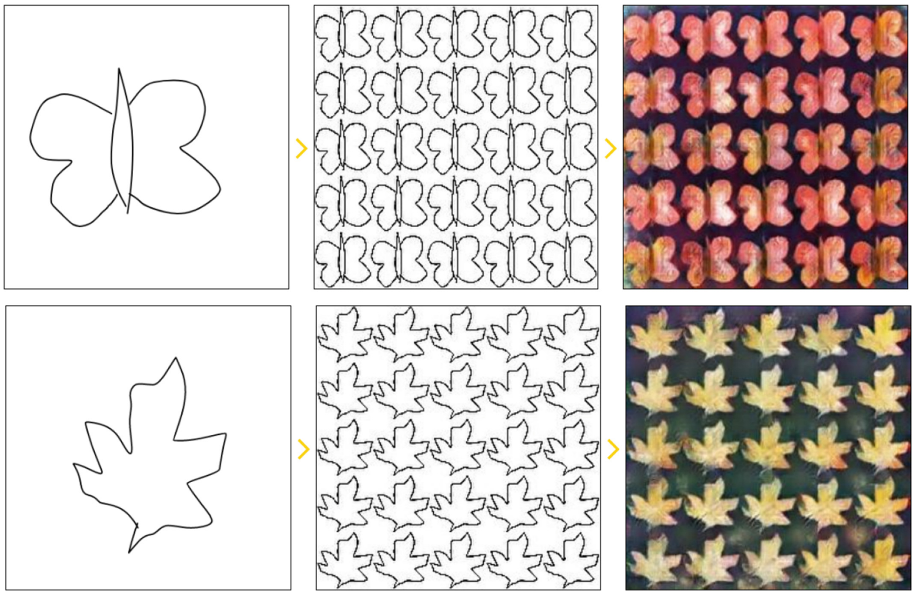

Fig. 5. End-to-end process via **Grid Style Augmentation** without object retrieval

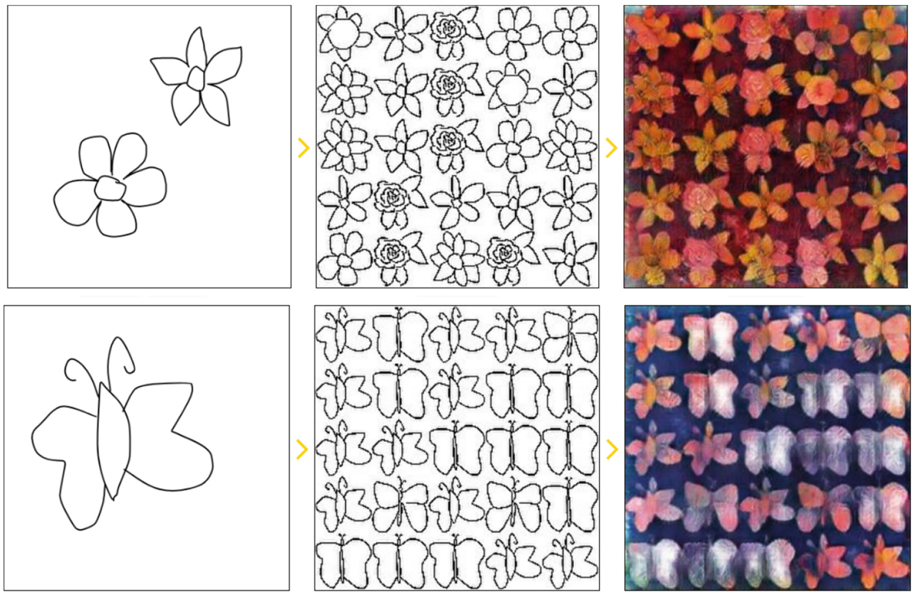

Fig. 6. End-to-end process via **Grid Style Augmentation** with object retrieval

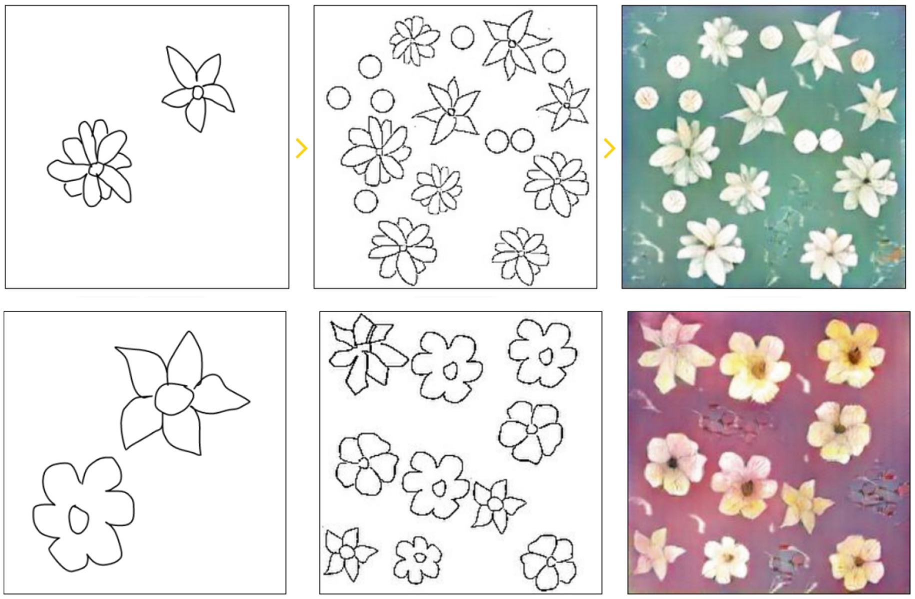

Fig. 7. End-to-end process via **Random Augmentation** without object retrieval

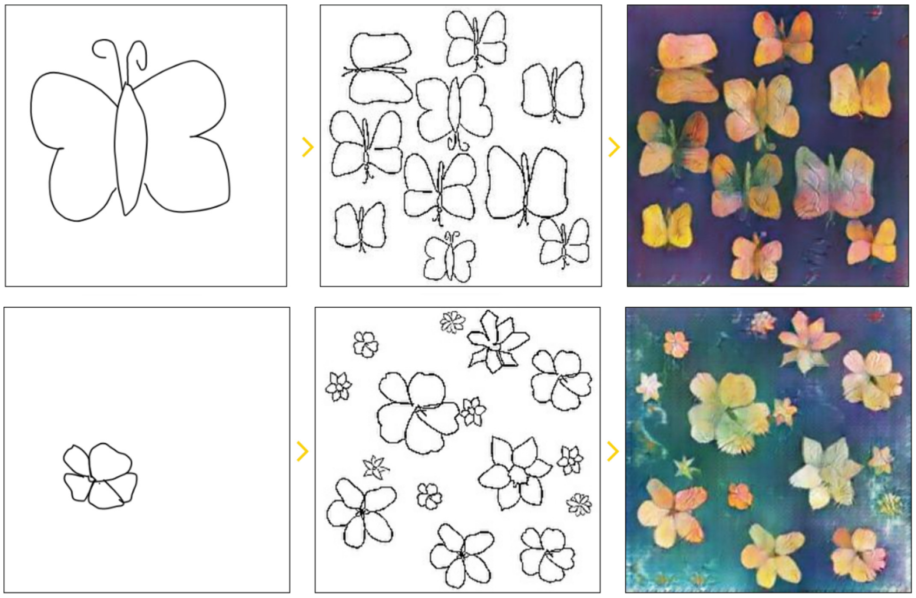

Fig. 8. End-to-end process via **Random Augmentation** with object retrieval

5.1 Discussion on Results

Figures 5 and 7 are examples of results obtained through both augmentation techniques, albeit without using Object Detection. Figures 6 and 8 are results that incorporate Object Detection in the process (see Sect. 4.1).

Given the empirical evidence of the performance of both augmentation techniques that are used to generate patterns, the Grid Texture Synthesis technique performs an order of magnitude faster than the Random Texture Synthesis. The average time for an end-to-end process for Grid-based augmentation was less than 10 s, whereas the latter averaged around 60–75 s based on the number of iterations for checking patch overlaps. The test runs were carried out on a compute-optimized Microsoft Azure F4s v2 instance (4 vCPUs, 8 GiB RAM). We identify areas of performance in the slower technique, and discuss these in Sect. 7, along with discussion on how the actual outputs may be evaluated for quality.

Given the potential application of this pipeline for designers, we intend this to act as a productivity tool that they can experiment their sketches on. A new design always goes through multiple iterations before it is perfected, and we intend this tool to provide the initial source of inspiration.

6 Conclusion

The majority of the textile design process is contingent upon the designer going through the various iterations of their designs, before choosing a viable candidate to move forward with. The proposed methodology of this paper accelerates this process by a significant degree, reducing the time to completion by preempting the design of the final pattern through the use of Deep Learning methodologies. By making use of this pipeline, designers in the industry can include an early visualization of their sketches in their decision-making, and leverage it to dictate the direction of their creative process.

7 Future Work

Although this pipeline caters well to floral patterns and designs, it is still a reduction in scope from its full potential. Both the pattern augmentation and design generation modules can have their datasets increased to contain multiple types of objects, perhaps even entire new categories, so that they are no longer restricted to floral designs. Doing so will enable the pipeline to become a more generalized tool, further enabling the designer in their tasks according to their needs.

Additionally, different design generation techniques may also be explored, such as the use of the Swapping Autoencoder for Deep Image Manipulation [17], to generate different styles of patterns along with a more fine-grained control on the texture of the image. This may prove to be a replacement for our CGAN, as the Autoencoder outputs boundaries and textures that are quite an improvement over those present in textures generated with the CGAN (albeit with a much lower degree of control what gets colored).

As for our performance concerns referred to in Sect. 5.1, we found our Random Texture Synthesis augmentation technique to be lacking in terms of compute efficiency. The technique iterates with the time complexity of $O(N^2)$ where N is the number of blocks in the augmented image, causing the time taken for a single block to be an order of magnitude higher than Grid Texture Synthesis. In future works, we can explore an alternative, more optimized algorithm that can perform the same with a reduced time complexity, as we have observed that the Random Texture Synthesis generates patterns which have more utility or scope of application – Grid textures may be a very niche application of this tool. Exploiting inherent parallelism in the algorithm is also an option, in case the algorithmic complexity does not change in the foreseeable future.

A notable part of the results is the subjectivity in human preference for art, due to which we are yet to propose a formalized evaluation metric. This is an avenue for further research, where user surveys can be conducted to learn what aspects of patterns are most important – turning those aspects into a potentially automated evaluation system.

References

1. Ahmad, Z.: Research report on Pakistan's textile industry analysis. SSRN Electron. J. (2009). https://doi.org/10.2139/ssrn.1651789
2. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-8**(6), 679–698 (1986). https://doi.org/10.1109/tpami.1986.4767851

3. Chen, W., Hays, J.: SketchyGAN: towards diverse and realistic sketch to image synthesis. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9416–9425. Salt Lake City, UT, USA (2018). https://doi.org/10.1109/CVPR.2018.00981
4. Dougherty, E.R.: An introduction to morphological image processing. SPIE-Int. Soc. Opt. Eng. (1992)
5. Efros, A., Leung, T.: Texture synthesis by non-parametric sampling. In: Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, pp. 1033–1038. IEEE, Kerkyra (1999). https://doi.org/10.1109/ICCV.1999.790383
6. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH 2001, pp. 341–346. Association for Computing Machinery, New York (2001). https://doi.org/10.1145/383259.383296
7. Fan, L., Krone, J., Woolf, S.: Sketch to image translation using GANs - LISA.FAN. https://lisa.fan/Resources/SketchGAN/sketch-image-translation.pdf. Accessed 25 Jan 2023
8. github-actions: rtmigo/img2texture: Cli for converting images to seamless tiles (2021). https://github.com/rtmigo/img2texture_py. Accessed 22 May 2022
9. Gao, C., Liu, Q., Xu, Q., Wang, L., Liu, J., Zou, C.: SketchyCOCO: image generation from freehand scene sketches. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5173–5182. IEEE, Seattle (2020). https://doi.org/10.1109/CVPR42600.2020.00522
10. Gupta, K., Damani, S., Narahari, K.N.: Using AI to design stone jewelry, November 2018. https://doi.org/10.48550/arXiv.1801.00723. Accessed 3 Feb 2023
11. Hu, M., Guo, J.: Facial attribute-controlled sketch-to-image translation with generative adversarial networks. EURASIP J. Image Video Process. **2020**(1), 1–13 (2020)
12. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5967–5976. IEEE, Honolulu (2017). https://doi.org/10.1109/CVPR.2017.632
13. Karimi, P., Davis, N., Grace, K., Maher, M.L.: Deep learning for identifying potential conceptual shifts for co-creative drawing (2018). https://doi.org/10.48550/ARXIV.1801.00723. Accessed 27 Jan 2023
14. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization (2014). https://doi.org/10.48550/ARXIV.1412.6980. Accessed 3 Feb 2023
15. Liang, L., Liu, C., Xu, Y.Q., Guo, B., Shum, H.Y.: Real-time texture synthesis by patch-based sampling. ACM Trans. Graph. **20**(3), 127–150 (2001). https://doi.org/10.1145/501786.501787
16. Moravec, J.: A comparative study: L1-norm vs. l2-norm; point-to-point vs. point-to-line metric; evolutionary computation vs. gradient search. Appl. Artif. Intell. **29**(2), 164–210 (2015). https://doi.org/10.1080/08839514.2015.993560
17. Park, T., Zhu, J.Y., Wang, O., Lu, J., Shechtman, E., Efros, A.A., Zhang, R.: Swapping autoencoder for deep image manipulation. In: Advances in Neural Information Processing Systems. Curran Associates Inc., Red Hook (2020). https://doi.org/10.48550/arXiv.2007.00653
18. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28

19. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations (ICLR 2015), pp. 1–14. Computational and Biological Learning Society, San Diego (2015). https://doi.org/10.3390/electronics10040497
20. Tian, N., Liu, Y., Wu, B., Li, X.: Colorization of logo sketch based on conditional generative adversarial networks. Electronics **10**(4) (2021). https://doi.org/10.3390/electronics10040497
21. Xing, J., Wei, L.Y., Shiratori, T., Yatani, K.: Autocomplete hand-drawn animations. ACM Trans. Graph. **34**(6) (2015). https://doi.org/10.1145/2816795.2818079