# Interpretable Font Generation with Texture Synthesis Networks

Xu Dong 200708160

School of Electronic Engineering and Computer Science
Queen Mary University of London, UK
x.dong@se20.qmul.ac.uk

**Abstract.** Automatic generation of glyph with texture image is a challenging task in computer vision area especially with controlled pattern output. In this paper, we propose a novel font generation system with two subnets, a interpretable glyph generation network based on Information Maximizing GANs (infoGAN) and a texture synthesised network based on Convolutional neural networks (CNNs). The system is able to generate different themes glyph image with controllable font properties (thickness, styles) and to apply synthesised texture on the glyph skeleton. We also provide a glyph dataset with 9 different font themes and more than 5000 different fonts. Experimental results are evaluated and demonstrated. How Machine Conditions express the machine existence of this system will be discussed as a higher-level computational creativity issue. Our work can be found at `https://github.com/dx199771/style-glyph-generator`.

## 1 Introduction

Font is a specific size, thickness and style of a typeface. There are around half a million fonts in the world. A delicate font style can always influence how readers perceive the text, product and visual content and help us to convey ideas that cannot directly express. However, creating a font style is not an easy feat. Typographers spend huge labour and invest a lot of time on creating a new font style. With the development of computational creativity, generative technologies become a trend for automatically creating fonts. In this work, we aim to propose an automatic font design system, which is able to automatically generate various fonts with texture effect and to adjust font properties such as thickness and rotation. The font design systems should consist two networks: A glyph generation network and a texture synthesis network.
Early research on font generation mainly focus on example-based method [23,22,2,21]. However these methods show limitations on generating different font style variations (different languages and styles) and poor font generation diversity. In recent decades, there are many deep learning-based methods have been developed to help font generation tasks [3,15,11,1,27,7,24]. These methods can robustly learned the correlations between different objects by using deep neural network (DNN). As for generative system, a recent trend is GANs based methods, many font generation works have been done with different varieties of GANs. However, most of above methods are not able to

automatically generate font images with controlled attributes. Our glyph generation network is based on infoGAN and provides an interpretable generative adversarial system that can control GANs' output.
Texture synthesis is a core area of computer vision and graphics. The goal of texture synthesis task is to obtain a new texture from sample textures. Existing texture synthesis approaches have two main categories: non-parametric resampling method [25,17,26,6] and parametric method [18,8,26,12]. In this work, we adopt a CNN texture synthesis method based on [8].
Furthermore, Like human condition, the expression of human existence and meaning of life. Machine has its own condition as a framework. This framework could be categorised by some rule and terms, according to Colton et al.[5]. *Machine Condition* as a framework that is able to assist with public understanding of computational creativity systems. In this paper, our font generation system's machine conditions are discussed, which express the machine existence of our system.

**Contributions.** In this work, we proposed **(i)** A parameter adjustable glyph generation network based on infoGAN is developed , which is able to disentangle and control the properties of generated font images (e.g. thickness, rotation). **(ii)** A texture synthesis network based on the optimised features spaces of convolutional neural networks, which is able to generate texture and apply them on generated font images. **(ii)** A scalable font dataset is proposed. It contains 5424 different fonts from Google Fonts and DaFont with 9 different categories(Sans serif, Old School, Curly, Pixel/Bitmap, etc.)

## 2 Background

**Font Generation.** Automatic font generation has attracted many research interests. Existing font synthesis methods can be classified into two main categories: A classical example-based font synthesis method [23,22,2,21] and a deep-learning based method [3,15,11,1,27,7,24].
Rapee and Takeo [23] developed a example-based font synthesis algorithm that takes outline and skeleton of characters and computes the blending weights from training dataset. The arbitrary font then can be synthesised by blending outlines and skeletons with weights. Comparably, Balashova et al. [2] proposed a stroke-based font generation model that can reparametrize arbitrary font with topological and fit them on a manifold based on geometric features. Xu et al. [21] introduced a novel prototype system that adopts a constraint-based analogues-reasoning system to generate aesthetically Chinese calligraphy from few existing training examples.
Apart from example-based methods, deep-learning based methods also gaining much popularity due to its applicability, accuracy and efficiency in many computer vision and image processing tasks. Baluja [3] learned a typographic style generation system based on a small set of letters by using deep neural networks. The system can both differentiate and generate new characters in the same style. Lyu et al. [15] investigated the Chinese calligraphy generation problem and proposed a synthesis system which uses a image-to-image translation method [13]. The image-to-image translation method focuses on the deformations between input images and output rather than only underlying

structure. Hayashi et al. [11] provided a GlyphGAN that uses Generative Adversarial Networks (GANs) to generate style-consistent font images. Our glyph synthesis network is based upon GlyphGAN, which employs WGAN-GP as basic network and add a output-control unit. While our glyph synthesis system utilizes infoGAN to adjust the properties of generated font images. Furthermore, Multi-content GAN [1] AGIS-Net[7] and JointFontGAN[27] utilize few-shot learning which only takes few sample images as input and generate a full set of letters.

**Generative Adversarial Networks** Since Goodfellow et al. [9] published Generative Adversarial Networks (GANs) in 2014, it has attracted the attention of many researchers in different areas of computer vision and deep learning. A GANs involves a Generator for generating images and a Discriminator that learns to discriminate images between real and fake. The whole process is to find a equilibrium to the below min-max problem:

$$\min_G \min_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$$

However, without auxiliary stabilization methods, vanilla GAN always shows poor convergence and requires intensive fine-tuning. Therefore, many recent studies has accordingly investigated the improvement of vanilla GANs. The deep convolutional GAN (DCGAN) [19] is a GANs structure based on Convolutional neural networks without fully connected layers or max pooling. It adopts convolutional stride, LeakyRelU and ReLu for discriminator and generator and transposed convolution for downscale and upscale, which contribute to its success.
Wasserstein GAN Gradient penalty (WGAN-GP) [10] provides a novel way to eliminate mode collapses and to boost convergence. It mainly defines an alternative Wasserstein distance loss with Gradient penalty function to ensure the training process going smoothly.
One of our work's aim is to control GANs' outputs. In vanilla GANs, it is difficult to control what type of data will be generated from a random latent space. Conditional GANs (cGAN ) [16] involves a conditional generation model that conditionally generate data by adding class label encoded as a one-hot vector. Additionally, Information Maximizing GANs [4] introduces a way to learn disentangled representations of input latent space. It provides a control variable that can be automatically learned by maximizing the mutual information and thus to generate interpretable and meaningful results. In our work, cGAN is adopted to control the class of generated character and infoGAN is employed to control the features of generated font images (thickness, rotation and style).

**Texture Synthesis** Texture synthesis has been an active research topic in computer vision and computer graphics areas. There are two main texture synthesis methodologies: non-parametric method and parametric model method. Previous non-parametric methods focus on resampling regular (with repeated texels) or stochastic (with non-repeated texels) textures to synthesis new texture without using parametric models. One of the first non-parametric model for texture synthesis is the work by Efros [6]. The basic idea of [6] is to firstly initialized a small texel region and then gradually enlarge the ini-


Fig. 1: Examples of various fonts with 7 different themes.

tial texel by assigning pixel via neighborhood search method. Several non-parametric methods are based on Efros's work [25,17,26].

However, most of non-parametric methods only process the frontal parallel textures which results in low synthesised texture quality. Also, it has issues in computational speed due to the exhaustive search methods. Parametric texture synthesis is an alternative approach to tackle the issues of non-parametric method and shows high synthesised quality [18,8,26,12]. These models can generate high quality texture though the textures are described via light and compact parameter models. Also the parameter sets provides texture editing mechanism (manipulate the texture parameter and combine different textures) [26].

Our texture synthesis network is based on Gatys et al. work [8]. It employs a Convolutional Neural Networks optimised for object recognition. The Gram matrix is proposed to describe texture features and the features of different layers of VGG19 are selected to calculate. Specifically, the system firstly generates an image that matches the Gram matrix of the ground truth image by using gradient descent from a noise image. The image is then be optimised by minimising the mean squared error of the Gram matrix. The formula as shown below:

**Machine Conditions**

## 3 Dataset

To the best of our knowledge, there is no existing datasets for font images generation with specific style. Therefore, a font images dataset is collected from online resources and proposed. The dataset contains 8 different classifications of typefaces: Sans serif, Old school, Curly, Pixel/Bitmap, Midevel, Western, Comic, Fancy as shown in the Fig.1. Each classification has more than 200 font styles with total 5254 fonts. In this work, only 26 capital letters are used, but the dataset can provide the entire collection of characters in a font.

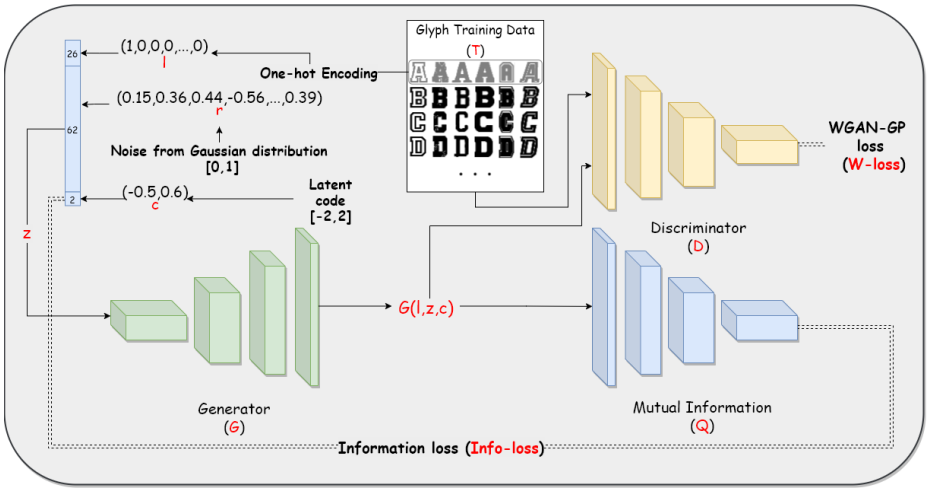


Fig. 2: Overall architecture of glyph generation network. The Generator $G$ generates take a concatenated vector which consists of a label vector $i$, a noise vector $r$ and a latent code $c$. Label vector is a one-hot encoded vector; Noise $r$ is a random Gaussian distribution vector from $[0-1]$; Latent code $c$ is a two-digits from $[-2, 2]$. The discriminator $D$ calculates the Wasserstein Gradient Penalty Loss $W-loss$. The mutual information $Q$ calculates the mutual information loss $Info-loss$ between latent code $c$ and the observations.

## 4 Techniques Implementation

Our model consists two subnets: A glyph synthesis network and a texture synthesis network. Numerous experiments with different network architectures were conducted. The final architecture of glyph generation network is shown in Fig.2. The font training data with 26 classes characters $T$ are fed into a one-hot encoder and generate label vector $I$. The noise vectors are sampled from a Gaussian distribution with range of 0 to 1. A two-digits random latent code are randomly sampled with range of -2 to 2. The generator $G$ then takes the input noise which generated by concatenating these three vectors and outputs a fake font image. Discriminator $D$

The final architecture of glyph generation network and texture synthesis network are shown in Fig.2 and Fig. respectively. In order to synthesis new texture on the basis of a ground truth texture image. The texture synthesis network firstly

## 5 Experimental Study

**Networks parameter settings** After fine-tuning the initial network. The structure of our final Glyph network is shown in Fig.2. As for generator G, we utilizes fractionally-strided convolution with 0.2 negative slope $LeakyReLU$ activation function at each layer except the last layer with $Tanh$ function. Batch normalization is used before each

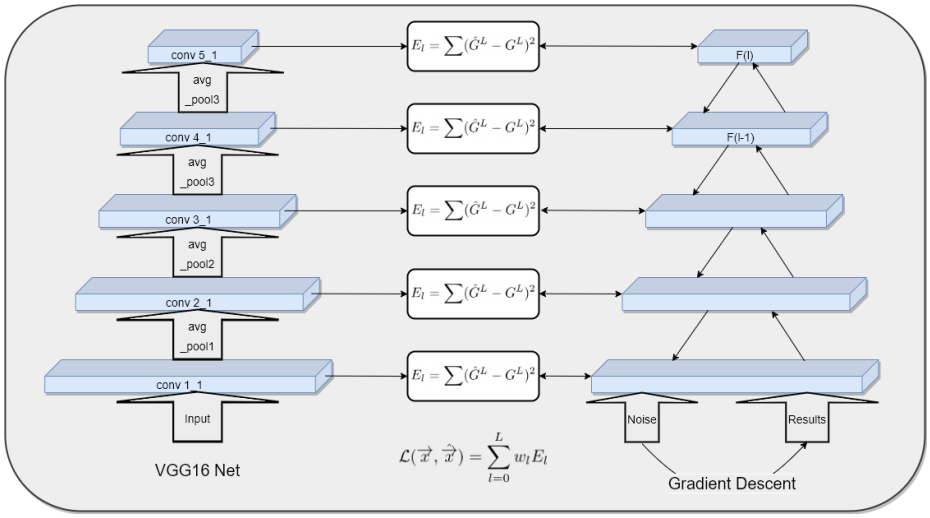


Fig. 3: Overall architecture of glyph generation network. The Generator $G$ generates take a concatenated vector which consists of a label vector $i$, a noise vector $r$ and a latent code $c$. Label vector is a one-hot encoded vector; Noise $r$ is a random Gaussian distribution vector from $[0 - 1]$; Latent code $c$ is a two-digits from $[-2, 2]$. The discriminator $D$ calculates the Wasserstein Gradient Penalty Loss $W - loss$. The mutual information $Q$ calculates the mutual information loss $Info - loss$ between latent code $c$ and the observations.

layer. As for discriminator D, 2D convolution with $0.15$ dropout functions is used for each layer to prevent the network from overfitting. For training process, we adopt Adam [14] as gradient-based optimization function with $0.0005$ learning rate. The batch size varies from dataset.

The structure of our texture synthesis network is shown in Fig.3. VGG-16 [20] is employs as our texture synthesis network. The network can be downloaded from our GitHub repository. The training process uses Adam [14] as gradient-based optimization functions with $0.001$ learning rate and $15000$ epochs. Average pooling layers are used instead of max pooling layer.

## 6 Results and Evaluation

**Generation Results** Fig.4 and Fig.8 show the generated result of glyph network using sans-serif dataset. In both figures of latent code manipulation, we uses the convention that one latent code varies from left to right while the other varies from top to bottom. In both figures, latent code from top to bottom represents the thickness of generated glyph images. In Fig.4, latent code from left to right represents the degree of rotation while in Fig.8, latent code represents the width of generated glyph images.

Fig.7 shows the 5 synthesised texture results of texture synthesis network. Three different layers in the model are tested ($avg\_poo2$, $avg\_poo3$, $avg\_poo4$). Each processing stage demonstrates different level of texture synthesis naturalness. The last column shows a non-texture image texture synthesis process, which shows a better understanding of how the entire model works.

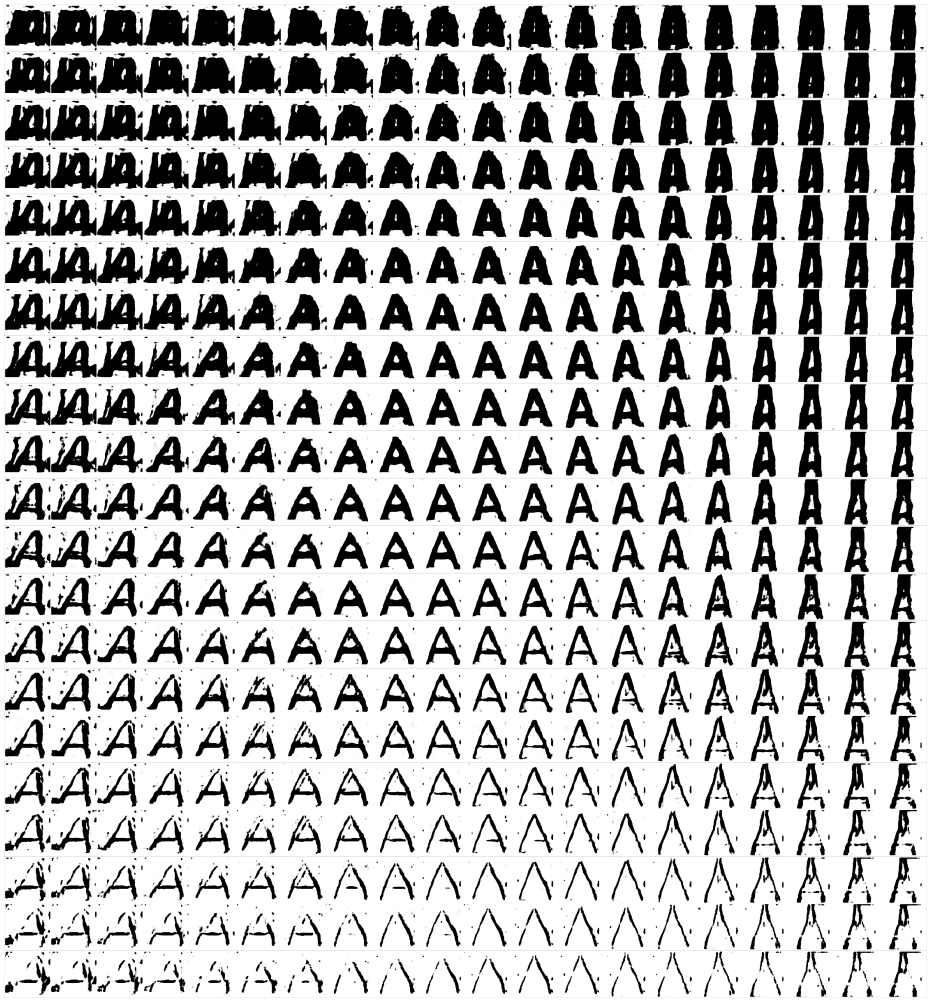

Fig. 4: Generated results of Sans-serif theme font character A.

Fig. shows the generation results of final font generation results which apply synthesised texture on generated glyph skeleton.

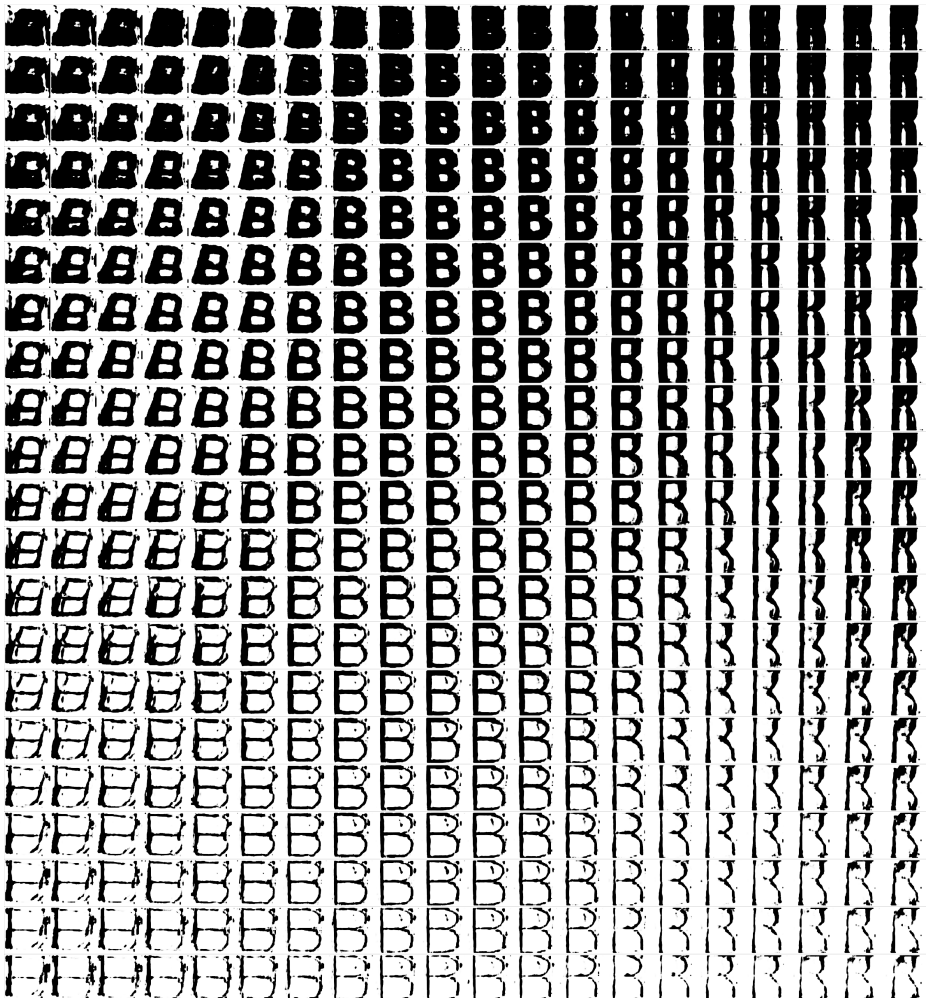

Fig. 5: Generated results of Sans-serif theme font character B.

**Evaluation** In this section, we show the quantitative and qualitative evaluation results of glyph generation network and texture synthesis network as well as final font generation results. Texture Similarity, Glyph Recognizability font Diversity

## 7 Discussion

From the computational creativity perspective.

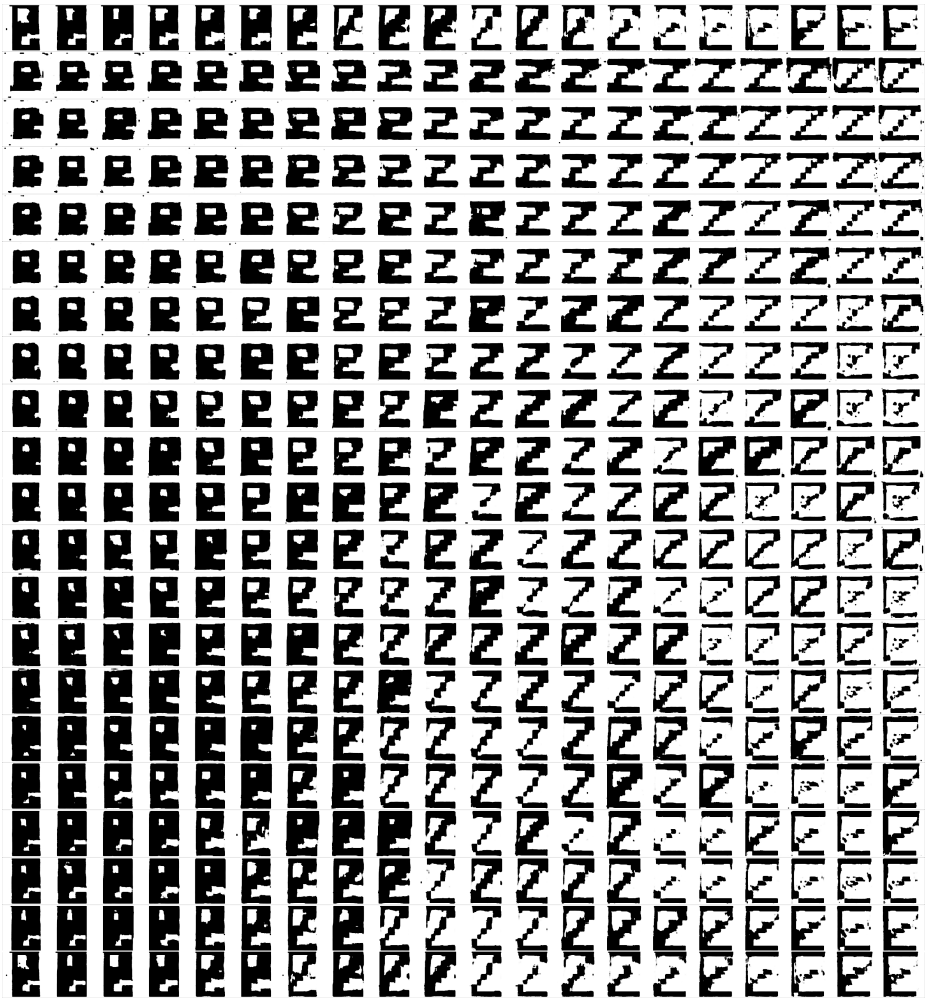

Fig. 6: Generated results of pixel/bit-map theme font character Z.

## 8 Conclusions and Future Work

In this paper, we present a interpretable font generation system consist of two subnets: A glyph generation network based on infoGAN and a texture syntehsis network based on CNN. In contrast to previous work, our system is able to generate controlled output font images with synthesised texture on glyph skeleton. *Machine Condition* of our work is discussed in terms of transience, learning, humanity, work and physicality aspects. However, as shown in the Fig.**??**, there are some failure generated examples, therefore, in further work, we will review all the font data and improve the quality of our font dataset. We will further investigate the internal of latent space and to learn more efficient

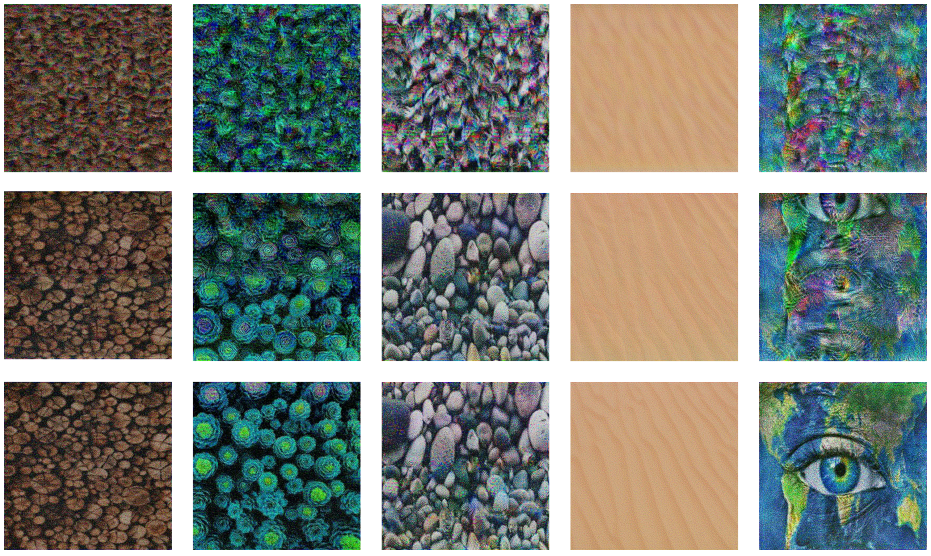

Fig. 7: Generated results of pixel/bit-map theme font character Z.

Fig. 8: Generated results of Sans-serif theme font character B.

disentangled representations and to improve the synthesised texture naturalness and quality. Some failure results.

## References


1. S. Azadi, M. Fisher, V. Kim, Z. Wang, E. Shechtman, and T. Darrell, "Multi-content gan for few-shot font style transfer," 2017.
2. E. Balashova, A. H. Bermano, V. G. Kim, S. DiVerdi, A. Hertzmann, and T. Funkhouser, "Learning a Stroke-Based representation for fonts," *Computer Graphics Forum*, vol. 38, no. 1, pp. 429–442, Feb. 2019.
3. S. Baluja, "Learning typographic style," *arXiv preprint arXiv:1603.04000*, 2016.
4. X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," 2016.
5. S. Colton, A. Pease, C. Guckelsberger, J. McCormack, and M. T. Llano, "On the machine condition and its creative expression," in *ICCC*, 2020.
6. A. Efros and T. Leung, "Texture synthesis by non-parametric sampling," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1033–1038 vol.2.
7. Y. Gao, Y. Guo, Z. Lian, Y. Tang, and J. Xiao, "Artistic glyph image synthesis via one-stage few-shot learning," *ACM Transactions on Graphics*, vol. 38, no. 6, p. 1–12, Nov 2019. [Online]. Available: http://dx.doi.org/10.1145/3355089.3356574
8. L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," 2015.
9. I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.
10. I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," 2017.
11. H. Hayashi, K. Abe, and S. Uchida, "Glyphgan: Style-consistent font generation based on generative adversarial networks," 2019.
12. D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis." New York, NY, USA: Association for Computing Machinery, 1995. [Online]. Available: https://doi.org/10.1145/218380.218446
13. P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 2018.
14. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.
15. P. Lyu, X. Bai, C. Yao, Z. Zhu, T. Huang, and W. Liu, "Auto-encoder guided gan for chinese calligraphy synthesis," 2017.
16. M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014.
17. A. Popat, "Conjoint probabilistic subband modeling," 04 2011.
18. J. Portilla and E. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *International Journal of Computer Vision*, vol. 40, 10 2000.
19. A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016.
20. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.
21. Songhua Xu, F. C. M. Lau, W. K. Cheung, and Yunhe Pan, "Automatic generation of artistic chinese calligraphy," *IEEE Intelligent Systems*, vol. 20, no. 3, pp. 32–39, 2005.
22. R. Suveeranont and T. Igarashi, "Feature-preserving morphable model for automatic font generation," in *ACM SIGGRAPH ASIA 2009 Sketches*, ser. SIGGRAPH ASIA '09. Association for Computing Machinery, 2009.
23. ——, "Example-based automatic font generation," in *Proceedings of the 10th International Conference on Smart Graphics*, ser. SG'10. Berlin, Heidelberg: Springer-Verlag, 2010, p. 127–138.

24. Y. Tian, “zi2zi: Master chinese calligraphy with conditional adversarial networks.” [Online]. Available: https://github.com/kaonashi-tyc/zi2zi
25. L. Wei and M. Levoy, “Fast texture synthesis using tree-structured vector quantization,” *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000.
26. L.-Y. Wie, S. Lefebvre, V. Kwatra, and G. Turk, “State of the Art in Example-based Texture Synthesis,” in *Eurographics 2009 - State of the Art Reports*, M. Pauly and G. Greiner, Eds. The Eurographics Association, 2009.
27. Y. Xi, G. Yan, J. Hua, and Z. Zhong, “Jointfontgan: Joint geometry-content gan for font generation via few-shot learning,” 10 2020, pp. 4309–4317.