Texture Generation (Synthesis) by Deep Learning

Xuzheng Lu
(G34363475)
Dept. of Computer Science
The George Washington University
Washington, DC
xuzheng_lu@gwu.edu

Zhiyuan Tao
(G45055489)
Dept. of Electrical and Computer Engineering
The George Washington University
Washington, DC
zhiyuan_tao0912@gwu.edu

Zetian Zheng
(G36558368)
Dept. of Computer Science
The George Washington University
Washington, DC
zhzt96@gwu.edu

***Abstract*—Texture generation is one of the computer vision and graphics problems, where the goal is to extract the texture from an image and then generate a new image with that texture or apply it to a content image. We implemented and evaluated three methods for texture extraction, generation, and transferring: Non-parametric Texture Synthesis (as the baseline model), Neural Style Transfer, and Cycle GAN. We designed a new metric, Texture Score, for model evaluation, and then performed experiments on the DTD dataset and Berkeley' monet2photo dataset. The results and outputs are evaluated and compared using the Texture Score.**

***Keywords*—Texture Generation, Style Transfer, Non-parametric Texture Synthesis, Neural Style Transfer, Cycle GAN.**

I. Introduction

Texture generation and Style transfer are popular research topics in the field of computer vision. Texture generation is mainly used in computer graphics and other fields to simulate geometric models' surface details and enhance the realism of models. Style transfer focuses on extracting a style from a specific painting and transferring it to another content image, for example, re-drawing a real photo in Monet or Picasso style.

Texture generation or synthesis has lots of application scenarios, and the industrial applications can bring commercial benefits. For internet products, many companies start using texture generation or synthesis technology to enhance the social communication experience.

II. Related work

1) *Non-parametric Texture Generation*

Non-parametric texture generation was proposed by Efros et al. in 1999. It is a simple algorithm to implement texture synthesis by growing a new image from an initial seed [1] to generate images with different textures. This algorithm has well performance but a large time cost. Except for that, Efros mentioned that this algorithm might grow garbage when some texture mistakenly "slip" or is locked onto someplace.

2) *Neural Style Transfer*

Neural Style Transfer (NST) is a Convolutional Neural Network-based (CNN-based) model [2]. This method extracts high-level semantic content like objects or background of an image and then applies it to a content image. The idea is to define two distance losses on a pre-trained CNN. Given the style image, the content image, and the input image, this model can apply the texture onto the input image by minimizing the losses with the content image and the style image.

3) *CycleGAN*

The CycleGAN [3] is a technique that automatically trains image-to-image translation models. This model's advantage is that it is an unsupervised training method using datasets that do not require the source and target samples to be paired. To implement a CycleGAN, two neural networks are required, the generative network and the discriminative network, and they will contest each other while be trained.

4) *Industrial Applications*

A mobile application named Prisma provides NST algorithm as a service, user uploads their pictures which are transformed into artistic style and achieve likes from their social network. Due to the accessibility and high quality of generated images, this app gains success among users from the whole world [4].

III. Approaches

A. *Non-parametric Texture Synthesis*

The non-parametric method is the baseline for texture generation. The synthesis process gets an initial seed and generates an image from inward to outward. The conditional distribution of a pixel is calculated by querying the sample image and finding all similar neighborhoods, which can make all neighbors to be synthesized [1].

The algorithm of Non-parametric Texture Synthesis is shown in Fig. 1 [7].

```
Choose a small square of pixels at random from the example image
Insert this square of values into the image to be synthesized
Until each location in the image to be synthesized has a value
    For each unsynthesized location on the boundary of the block of
    synthesized values
        Match the neighborhood of this location to the example image,
        ignoring unsynthesized locations in computing the matching score
        Choose a value for this location uniformly and at random from the
        set of values of the corresponding locations in the matching
        neighborhoods
    end
end
```

Fig. 1. Non-parametric Texture Synthesis algorithm

B. Neural Style Transfer

Neural Style Transfer (NST) is a Convolutional Neural Network-based (CNN-based) model proposed by Gatys et al. in 2015 [2]. It could extract high-level semantic information (objects and background scenery), while the non-parametric algorithm only utilizes low-level local features.

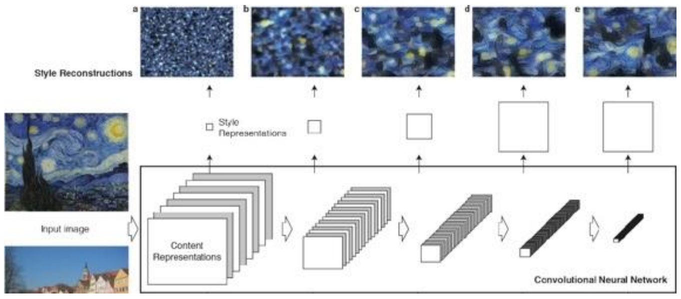


Fig. 2. Image reconstructed from feature maps

NST not only generates but also transfers the styles, which means to separate and represent the content of a content image and style of a style image. The word "style" is borrowed from the style of artworks, but it is not a brand-new concept for generation work. Transferring the style from one image onto another could be considered as a texture transfer. A CNN could be trained for both the texture generation and style transfer task because each convolutional layer could extract certain and different features from the input image.

1) Style Transfer

The architecture for the style transfer model is shown in Fig. 3 [8]. The feature maps nearer the input layer experience less convolution and pooling, and the filter nearer the output have larger receptive field sizes and feature complexity. Therefore, the lower layers contain more local textures, where the "style" of an image could be extracted. In comparison, the higher layers extract more general information, which could be considered as the "content" of the input image.

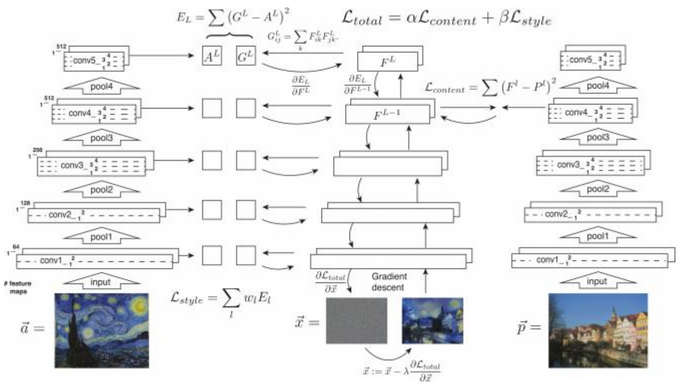

$E_L = \sum (G^L - A^L)^2$
$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$
$G^L_{ij} = \sum_k F^L_{ik} F^L_{jk}$
conv5_3
pool4
conv4_3
pool3
conv3_3
pool2
conv2_2
pool1
conv1_2
input
# feature maps
$\mathcal{L}_{content} = \sum (F^l - P^l)^2$
$\mathcal{L}_{style} = \sum_l w_l E_l$
Gradient descent
$\vec{a} =$
$\vec{x} =$
$\vec{p} =$
$\vec{x} := \vec{x} - \lambda \frac{\partial \mathcal{L}_{total}}{\partial \vec{x}}$

Fig. 3. Neural Style Transfer architecture

In Neural Style Transfer model, two losses are designed for evaluate the representation of the content and the style:

a) Content Loss

A mean square-error loss function $L_{content}$ of vectorized feature maps $F^l_{i,j}$ of the i^{th} filter at the position j in layer l of the content input $\vec{p}$ and $P^l_{i,j}$ of image been transferred $\vec{x}$ is defined for content representation:

$$L_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F^l_{i,j} - P^l_{i,j}\right)^2$$

Since the content information is represented in the high-level layers, the conv5 layer is chosen for calculation the content loss.

b) Style Loss

The Gram matrix is defined as follows to represent the style and calculate the differences between feature maps. The Gram matrix $G^l_{i,j}$ computes the correlations of feature maps $F^l_{i,k}$ and the transpose of it $F^l_{j,k}$. It captures similar features that co-occur at different parts of the image, and the displacement of position and global arrangement will not affect the detection.

$$G^l_{i,j} = \sum_k F^l_{i,k} F^l_{j,k}$$

Then, the style loss could be defined as the summation of the loss on each layer:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G^l_{ij} - A^l_{ij}\right)^2$$

$$L_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

The single layer loss E_l is normalized corresponding to that a layer has N_l feature maps each of size M_l. The weight w_l brings an allocation between styles from lower-scale to higher-scale. The total loss function is defined as a weighted summation of $L_{content}$ and L_{style}

$$L_{total} = \alpha L_{content} + \beta L_{style}$$

where the weight $\frac{\alpha}{\beta}$ is the key parameter to adjust the emphasis on style or on content.

Finally, the gradient descent technique is used to minimize the total loss and train a Gaussian White Noise image to be similar to both the style and content.

2) Texture Generation

When using the NST model for texture synthesis rather than style transfer, we could take only the left and middle part of the previous architecture, as shown in Fig. 4 [9].

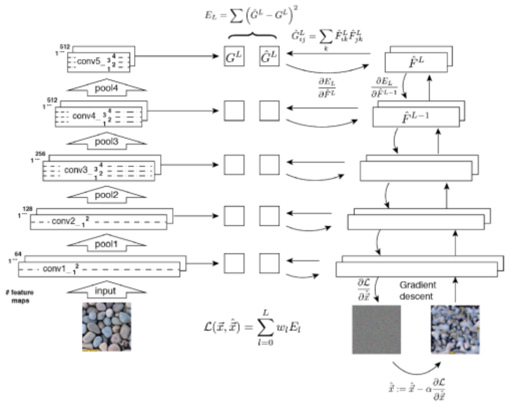

$E_L = \sum (\hat{G}^L - G^L)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik} \hat{F}^L_{jk}$
conv5_3
pool4
conv4_3
pool3
conv3_3
pool2
conv2_2
pool1
conv1_2
input
# feature maps
$\mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$
Gradient descent
$\hat{\vec{x}} := \hat{\vec{x}} - \alpha \frac{\partial \mathcal{L}}{\partial \hat{\vec{x}}}$

Fig. 4. Neural Texture Generation model

In this approach, the total loss is the same form as L_{style} in the style transfer section:

$$L(\vec{x},\hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$$

C. *CycleGAN*

A state-of-the-art method for texture (style) transfer is Generative Adversarial Networks (GAN) [10], proposed by Goodfellow in 2014. It is a very powerful Image-to-Image Translation model for generating new data with the same statistics as the training set. A GAN consists of two neural networks, the generative network and the discriminative network, and they are contesting each other in a game (in the sense of Game Theory). Typically, the GAN only works well on paired examples, which means there are similar patterns in the example x and y, as shown in Fig. 5. However, in practice, it would be challenging to build a paired dataset.

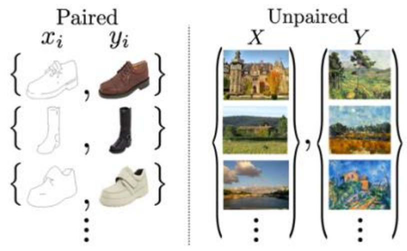


Fig. 5. Paired and unpaired datasets

In 2017, Zhu et al. proposed a Cycle-Consistent Adversarial Network (CycleGAN) [2], which could be trained on unpaired training examples, and the model can randomly choose the data samples from X and Y. CycleGAN is an optimized version of GAN, which has two generators and two discriminators, and they are in a recurrent loop while being trained, as shown in Fig. 6.

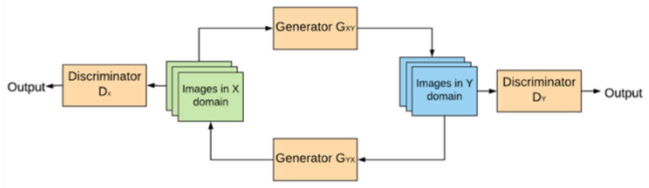


Fig. 6. CycleGAN architecture

The training process of CycleGAN consists of two steps, the forward generation G, and the backward generation F, as shown in Fig. 7. During the training, the cycle-consistency losses will be reduced to regularize the forward and backward mappings. The idea is that if we translate from one domain to the other and back again, we should arrive where we started.

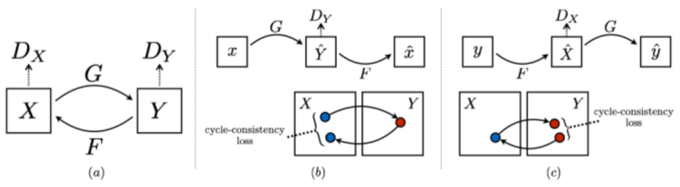


Fig. 7. Training process of CycleGAN

The objective of CycleGAN is to minimize the Total Loss

$$\begin{aligned} L(G,F,D_X,D_Y) = {} & L_{GAN}(G,D_Y,X,Y) \\ & + L_{GAN}(F,D_X,X,Y) \\ & + \lambda L_{cyc}(G,F) \end{aligned}$$

where the Adversarial Loss L_{GAN} is defined as

$$\begin{aligned} L_{GAN}(G,D_Y,X,Y) = {} & E_{y\sim p_{data}(y)}[log D_Y(y)] \\ & + E_{x\sim p_{data}(x)}\left[\log\left(1-D_Y(G(x))\right)\right] \end{aligned}$$

and the Cycle Consistency Loss L_{cyc} is defined as

$$\begin{aligned} L_{cyc} = {} & E_{x\sim p_{data}(x)}\left[\left\|F(G(x))-x\right\|_1\right] \\ & + E_{y\sim p_{data}(y)}\left[\left\|G(F(y))-y\right\|_1\right] \end{aligned}$$

IV. DATASETS

A. *Describable Textures Dataset*

We used the Describable Textures Dataset (DTD) [5] for training our models. It has different kinds of textures so that we could evaluate the algorithms on various data samples. DTD dataset consists of 5640 images, of which sizes range between 300×300 and 640×640, organized according to a list of 47 terms (categories) inspired by human perception. There are 120 images for each category, and images are both real-world and synthetic, collected from Google and Flickr by entering proposed attributes and related terms as search queries.

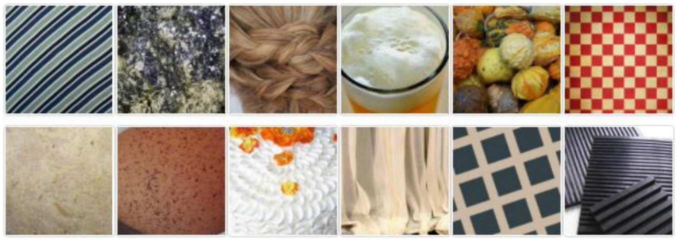
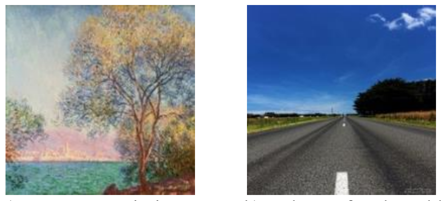


Fig. 8. Different kinds of textures in DTD dataset

B. *Berkeley's monet2photo Dataset*

We used Berkeley's monet2photo dataset for texture transfer and style transfer [6]. The dataset consists of over 1000 Monet's paintings and over 6000 real photos of different scenarios. The size of images is 256 × 256. Although the style samples and photo samples are unpaired, it is enough for a CycleGAN style transfer task.



a) a Monet's painting b) a photo of real world

Fig. 9. Images in Berkeley's monet2photo dataset

V. RESULTS

A. *Evaluation Metrics*

Two metrics are used to evaluate the results for models:

1) Texture (Style) Score

We designed a novel metric, Texture Score, to measure the texture consistency between the original and generated images. A lower score means that the model has a better performance.

The calculation process of Texture Score is similar to the Style Loss defined in the Neural Texture Generation model, as shown in Fig. 10 [8]. When computing the Texture Score, both left and right CNN blocks will be initialized by a pre-trained model, and no parameters will be updated.

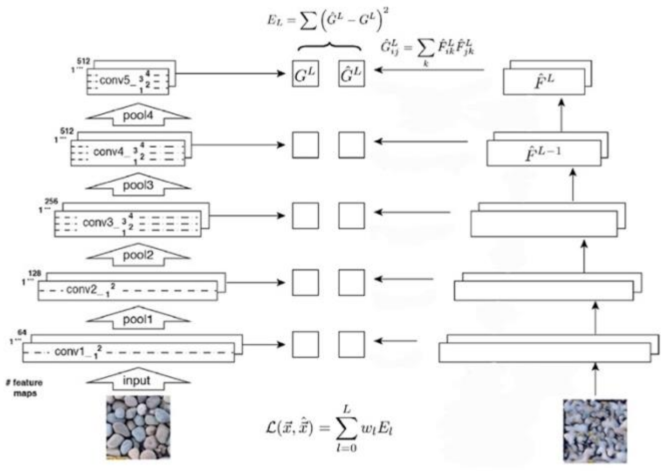

$E_L = \sum \left(\hat{G}^L - G^L\right)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik}\hat{F}^L_{jk}$
G^L
$\hat{G}^L$
$\hat{F}^L$
$\hat{F}^{L-1}$
conv5_1 2 3 4
pool4
conv4_1 2 3 4
pool3
conv3_1 2 3 4
pool2
conv2_1 2
pool1
conv1_1 2
input
# feature maps
512
512
256
128
64
$\mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$

Fig. 10. Calculation process of Texture Score

The ensembling technologies are also used to reduce the variance of the results.

2) Content Consistency

The content consistency measures the semantic similarity between the generated image and the content image. It could be manually observed by comparing the content image and the generated image.

B. Implementation Environments

We used Python 3.7 as the programming language and used TensorFlow, Keras, Pytorch, and scikit-learn packages for modeling and data processing.

The model is trained on different devices, including a GPU server in GWU Pegasus Cluster, which is equipped with 4 NVIDIA Tesla V100-16G graphic cards.

C. Non-parametric Texture Synthesis

The Non-parametric Texture Generation model was tested on the DTD data set. We split the textures with regular patterns, such as stripes, and stochastic patterns, such as Crystal. The basic idea is that the regular patterns have some simpler and more regular textures that are easier to describe, define, and learn from, while stochastic patterns are the opposite.

The size of input and output texture images is 100×100. The Gaussian weighting is applied for the generation process. The runtime is exceptionally long when we set the size of the input and output images to be large. To accelerate the execution process, we implemented a distributed algorithm to deploy the model onto a 40-CPU-Cores server in GWU Pegasus Cluster.

1) Results on different kinds of input textures

The results of texture generation on different kinds of textures are shown as the Fig. 11 and Fig. 12.

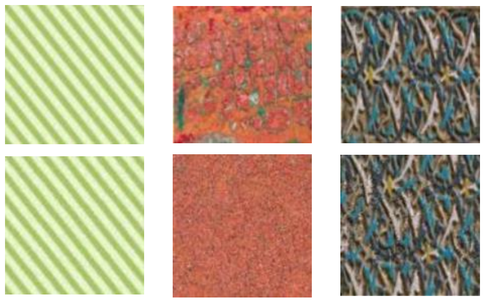


Fig. 11. Non-parametric texture synthesis with regular pattern

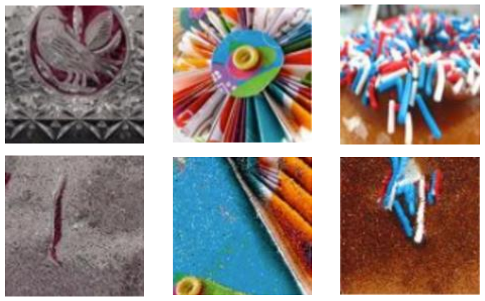


Fig. 12. Fig.11 Non-parametric Texture Synthesis with Stochastic Pattern

As shown above, the Non-parametric Texture Generation model works better on strong pattern images, and the results have lower Texture Scores.

2) Results for different window sizes

The results of Non-parametric model using different window sizes are shown in TABLE II.

To conclude, the quality of generated becomes better as the window size increases since a larger window could cover more texture features. However, the runtime also increases.

Whether a window size is good enough to get an acceptable image depends on the "thickness" and the "uniformity" of the input texture image. In the results, an 11×11 window is good enough for texture 1 and, but texture 2 did not perform well until we used a 15×15 window.

D. Neural Style Transfer

1) Texture Generation

The Neural Texture Generation model was trained and tested on the DTD dataset with 300 epochs. To reduce the noises and variance of the result, we selected three images for each texture, generated three images on the same input image, and then took the average over the same input results. In this way, we could reduce the impact of outliers and perform a robust evaluation of the model.

Texture Generation results on several different kind of textures are shown in the TABLE I.

TABLE I. TEXTURE GENERATION RESULTS OF NEURAL STYLE TRANSFER MODEL

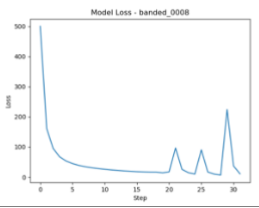
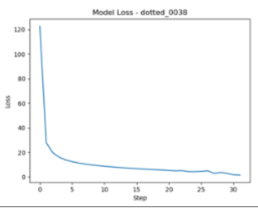
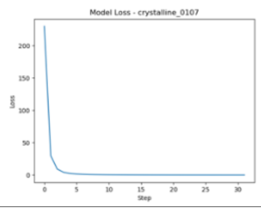
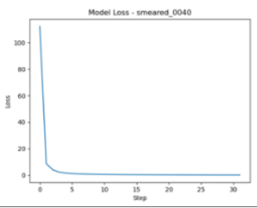
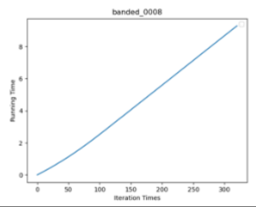
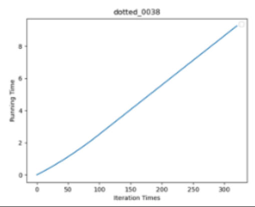
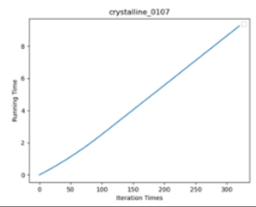
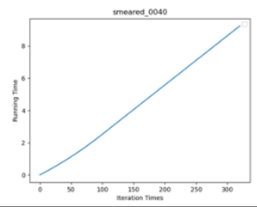
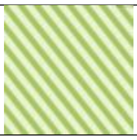
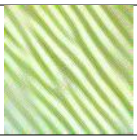
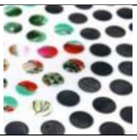
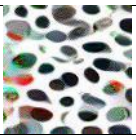
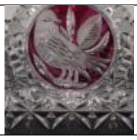
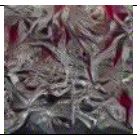
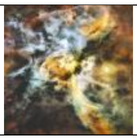
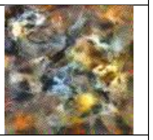
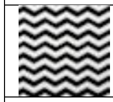
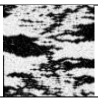
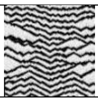
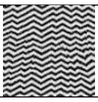
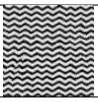
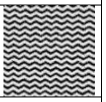
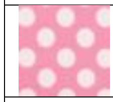
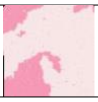
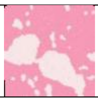
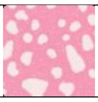
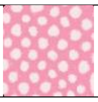
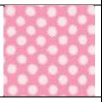
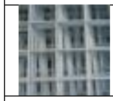
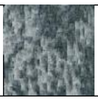
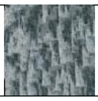
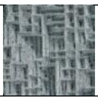
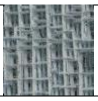
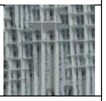
	Banded texture		**Dotted texture**		**Crystalline texture**		**Smeared texture**	
Loss								
Runtime								
Generated images	**original**	**generated**	**original**	**generated**	**original**	**generated**	**original**	**generated**
								
Texture Score	2313.51		2576.26		2919.33		2508.12	

TABLE II. IMAGES GENERATED USING DIFFERENT WINDOW SIZES

Original Image	**Window Size**				
	3x3	**5x5**	**11x11**	**15x15**	**21x21**
Texture 1					
					
Runtime	50.72s	40.94s	75.39s	104.21s	144.66s
Texture 2					
					
Runtime	50.26s	40.35s	77.24s	104.06s	148.24s
Texture 3					
					
Runtime	50.66s	40.98s	75.75s	104.05s	147.27s

The banded input texture has apparent regularity. The bands in the generated image have some branch patterns which are different from the original input. The loss curve shows some fluctuation but finally converges, which means the model is robust enough to jump out of the local minimum.

The dotted input is also plenty regular. The output dots are a little out of place, but they have similar patterns to the input shape, size, and color. The loss curve converged in about 100 epochs, and this verifies that the model is well trained.

The crystalline output disrupts the content graph from the input since the algorithm only learned the texture part. However, the generated image has some patterns of crystal texture.

It is hard to describe what the texture is inside the smeared input texture. It might be some brush stroke with a specific color map. However, the output learns it well enough to be considered as generated from the same artist. To be more specific, researchers cannot even describe mathematically what the model of input textures are. However, the magic is that the Neural Network-based method imitates and generates the texture from learning itself without a pre-defined mathematical or statistical model. This is something the traditional non-parametric way cannot do.

The loss curve of both crystalline and smeared texture decreased smoothly and converged in about 60 epochs, which tested and verified the robustness of NST model.

To sum up, the Neural Style Transfer method is capable of Texture Generation tasks.

2) Style Transfer

To obtain the result of the Neural Style Transfer algorithm, we used artworks “sunset” and “lotus” from Monet as the style images and used photos from Berkeley’s monet2photo datasets as content images.

The model was trained for 300 epochs with the parameter $\frac{\alpha}{\beta} = 1 \times 10^{-6}$, giving attention to both style and content.

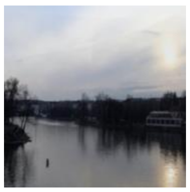
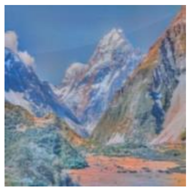
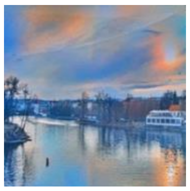
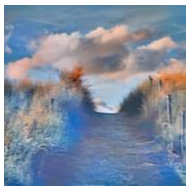
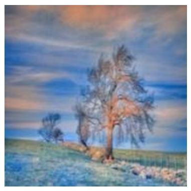
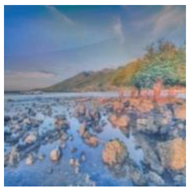
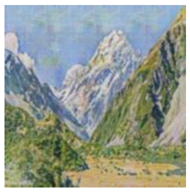
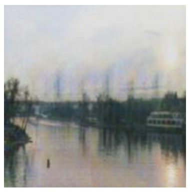
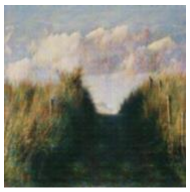
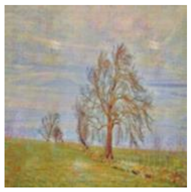
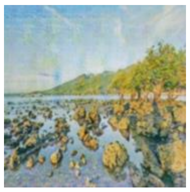
Original					
Generated (NST)					
Texture Score (NST)	209.24	158.41	144.64	145.98	148.60
Generated (CycleGAN)					
Texture Score (CycleGAN)	498.49	1168.21	778.90	721.65	425.18

Fig. 13. Texture synthesis results of Neural Style Transfer model and CycleGAN on different kinds of textures

The results of Neural Style Transfer model are shown in Fig. 13 and Fig. 14.

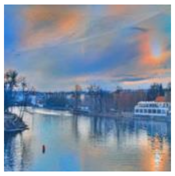
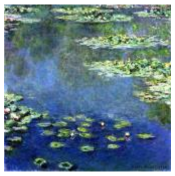
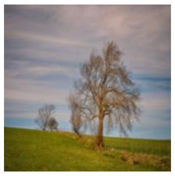
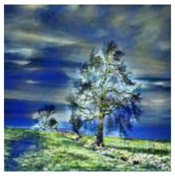
Style image	**Content image**	**Generated image**	**Texture Score**
			158.41
			578.56

Fig. 14. Resuls of Neural Style Transfer

The transferred images are very beautiful, and they learned the essential artistic style of Monet's style while keeping the content from content images. The texture scores are much lower compared with texture generation, which indicates that the texture is easier to be located and detected by the estimator when given the content information.

E. *CycleGAN*

We implemented the CycleGAN from scratch and used the Berkeley's monet2photo dataset for training and testing. The model was trained on over 6000 real photos and over 1000 Monet's paintings with 25 epochs on a NVIDIA Tesla V100-16G graphic card.

The losses during the training process are showing in Fig. 15, which indicates that the generators and discriminators are contesting each other.

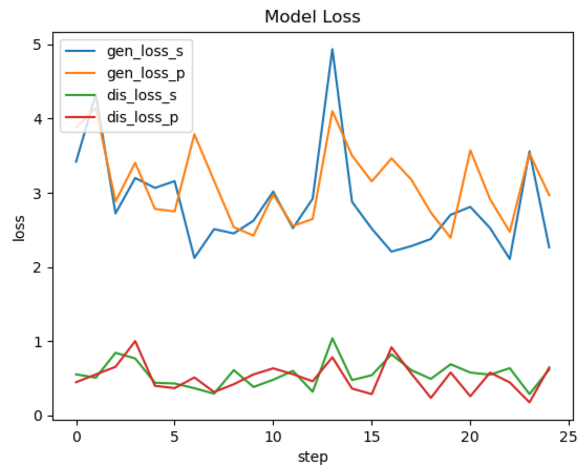


Fig. 15. Cycle GAN Training Losses vs. Epochs

The runtime of the training process was about 2 hours.

The images transferred from different real photos are shown in Fig. 13. The Texture Scores are low enough to verify the robustness of the model.

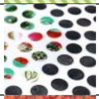
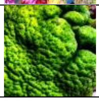
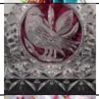
F. *Model Evaluation and Comparision*

1) *Texture Generation*

To compare the performance of the Non-parametric model and the Neural Style Transfer model on the texture generation task, we tested two models on different textures selected from the DTD dataset. The tests were performed on a bunch of images, and the average was taking for each kind of texture to reduce the variance of the results.

The results are shown in TABLE III.

TABLE III. TEXTURE GENERATION RESULTS OF NEURAL STYLE TRANSFER MODEL AND NON-PARAMETRIC MODEL

	Category	Original Image	NP	NST
Regular Patterns	banded		1322.87	5060.22
	dotted		4575.94	2576.26
	blotchy		1630.77	1581.62
	braided		2684.41	2030.48
	bumpy		14507.18	1253.75
	Average Score		4944.23	2500.47
Stochastic Patterns	Interlaced		1109.77	2788.23
	pleated		3975.54	2384.70
	crystalline		1247.58	1957.70
	sprinkled		9849.04	3944.74
	smeared		1232.80	2782.07
	Average Score		3482.95	2771.49
Overall Average Score			4213.59	2635.98

The overall average texture score indicates that the Neural Texture Generation model works better than Non-parametric model.

Both of the two models work well on textures with regular patterns. However, the Non-parametric model struggled with learning complex textures, such as interlaced and sprinkled texture, on which Neural Style Transfer performed impressively well.

2) *Texture (Style) Transfer*

To compare the performance of the Neural Style Transfer model and CycleGAN on the texture generation task, we tested two models on different photos selected from Berkeley's monet2photo Dataset. The results are shown in Fig. 13.

Although the CycleGAN got higher Texture Scores than the Neural Style Transfer model, it could generate various styles rather than just one. This is because of the nature of the CycleGAN – it uses an unpaired dataset for training. However, NST requires a style image as the input and only generates the same style patterns as the input image.

VI. CONCLUSION

In conclusion, we researched and implemented three, Non-parametric Texture Synthesis, Neural Style Transfer, and Cycle GAN, for texture generation and transfer. We collected data from DTD and Berkeley's monet2photo datasets and use them to test and evaluate the models we implemented. We designed a novel metric, Texture Score, for evaluating the performance of the texture generation and compared the score for the models in different scenarios and settings.

As shown in the results and the scores, all the methods are well trained and are robust on specific tasks. The results also verify that the hypothesis we made for this project – textures of an image could be extracted and generated by machine.

In the future, we will use more epochs with more data to train the models. Data augmentation and ensemble learning technologies could be used to reduce the overfitting, and an objective metric for content representation is needed for evaluation.

REFERENCES

[1] Efros, Alexei A., and Thomas K. Leung. "Texture synthesis by non-parametric sampling." *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. IEEE, 1999.

[2] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." arXiv preprint arXiv:1508.06576 (2015).

[3] Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." *Proceedings of the IEEE international conference on computer vision*. 2017.

[4] Jing, Yongcheng, et al. "Neural style transfer: A review." *IEEE transactions on visualization and computer graphics* (2019).

[5] Describable Textures Dataset (DTD), https://www.robots.ox.ac.uk/~vgg/data/dtd/

[6] Berkeley's monet2photo Datase, https://people.eecs.berkeley.edu/~taesung_park/CycleGAN/datasets/

[7] Forsyth, David A., and Jean Ponce. Computer vision: a modern approach. *Prentice Hall Professional Technical Reference*, 2002.

[8] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[9] Gatys, Leon, Alexander S. Ecker, and Matthias Bethge. "Texture synthesis using convolutional neural networks." *Advances in neural information processing systems*. 2015.

[10] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

[11] Li, Haochen. "A literature review of neural style transfer."

Appendix

A. *Codes*

The codes for this project are on the GitHub:

https://github.com/LeanderLXZ/texture-generation

B. *Files Descriptions*

```
texture-generation/
├── cycle-gan/                              CycleGAN model
│   ├── cycle_gan.py                        Base model
│   ├── env.yaml                            Anaconda environment
│   ├── image_helper_cycle_gan.py
│   ├── loss_100.png                        Loss of training on 100 epochs
│   ├── loss_25.png                         Loss of training on 25 epochs
│   ├── monet2photo/                        Data directory
│   ├── photo_to_monet.py                   Execution script for training
│   └── training_results_25/                Results
├── neural-style-transfer/                  NST model
│   ├── README.md
│   ├── README.pdf
│   ├── data/                               Data directory
│   ├── env.yaml                            Anaconda environment
│   └── src/
│       ├── cnn_executable.py               Execution script for training
│       ├── eva_executable.py               Execution script for evaluation
│       ├── evulation.ipynb                 Notebook for experiments
│       ├── model/                          Base models
│       │   ├── __init__.py
│       │   ├── cnn_image.py
│       │   ├── cnn_model.py
│       │   ├── evaluation.py
│       │   └── transfer.py
│       └── notebooks/                      Notebooks
└── non-parametric-texture-generation/      NP Texture Generation
    ├── data/                               Data directory
    ├── texture_dtd.py                      Execution script for generation
    └── texture_synthesis.py                Base model
```

C. *Instructions for Codes Execution*

1) *Non-parametric Texture Generation*

Step 1: Change the directory to non-parametric-texture-generation/.

Step 2: Run command:

```
python3 texture_dtd.py
```

2) *Neural Style Transfer*

Step 1: Change the directory to neural-style-transfer/src/. To set the environment, kindly use env.yaml file in this directory.

Step 2: Run command:

```
python3 cnn_executable.py
```

To evaluate generated images manually:

Step 1: Save the original image with name "image_name_0", and the rest images' name are following the pattern "image_name_x", where x start from 1. Then put them in to data directory.

Step 2: Import the files by changing the code in eva_excutable.py:

```
g='../data/image_name_directory'
```

Step 3: Change the directory to neural-style-transfer/src/. To set the environment, kindly use env.yaml file in this directory.

Step 4: Run command:

```
python3 eva_executable.py
```

3) *CycleGAN*

Step 1: Change the directory to cycle-gan/. To set the environment, kindly use env.yaml file in this directory.

Step 2: Run command:

```
python3 photo_to_monet.py
```

D. *Dataset Samples*

1) *Describable Textures Dataset*

2) *Berkeley's monet2photo Dataset*

a) a Monet's painting

b) a photo of real world

E. *Teamwork*

1) *Xuzheng Lu: Implementation and testing for Non-parametric Texture Generation model and CycleGAN, model evaluation*

2) *Zhiyuan Tao: Implementation and testing for Neural Style Transfer model, model evaluation*

3) *Zetian Zheng: Implementation and testing for Neural Style Transfer model, model evaluation*