EEC 289A Assignment 2 Report

Chenye Yang, Hanchu Zhou, Haodong Liang, Yibo Ma

1 Introduction

This project seeks to reproduce and expand upon the work by Efros & Leung (1999) on non-parametric texture synthesis. Originally, this method was groundbreaking for its ability to synthesize new texture fields that maintain the visual characteristics of a given sample texture. Our primary objective is to apply this technique to the textures assigned in the class, exploring both the effectiveness and the limitations of the method in replicating and scaling up these specific patterns.

In addition to replication, this project will extend the methodology to other images and contexts of interest. Particularly, we aim to test the feasibility of applying texture synthesis techniques to generate MNIST digits and a galaxy picture[1], which present a unique challenge in scaling the approach to handle more complex and structured data.

Moreover, we delve into some theoretical questions that arise from the practice of generative modeling. In summary, this report outlines our approaches, experiments, and tries to answer the following questions:

1. Scale up non-parametric texture synthesis?
2. Why are generative models assign higher probability to something it has not been trained on?
3. Can we use the state-of-the-art open-sourced large language models to re-estimate the upper bound of the conditional entropy of text, like Shannon has done in 1951?
4. Can you learn the words from text with dictionary learning?

2 Methods

2.1 Textures

We apply the non-parametric texture synthesis method to the following textures: the textures provided in the class, the MNIST digits, and the galaxy picture we chosen. The textures provided in the class have the size around 530×530, shown in Figure 1.

(a) Texture 2 (b) Texture 4 (c) Texture 9

Figure 1: Texture: provided in the class

[1]Galaxy starry night sky background, free public domain CC0 photo. `https://www.rawpixel.com/image/5924106/photo-image-public-domain-stars-galaxy`

For the MNIST digits, we load the MNIST dataset (28×28 handwritten digits)[2], including 60000 training images and 10000 testing images. We randomly select 20×20 images from the training set as the sample texture of size 560×560, shown in Figure 2.

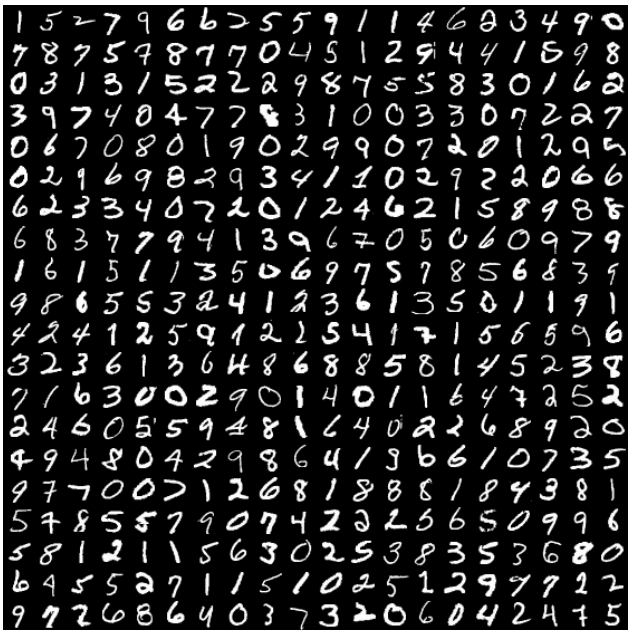


Figure 2: Texture: MNIST digits

For the galaxy picture, we load the galaxy picture (800×585), shown in Figure 3.

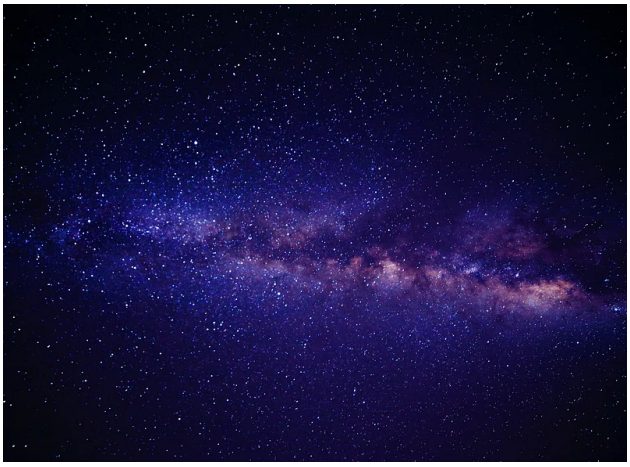


Figure 3: Texture: galaxy picture

2.2 Non-parametric Texture Synthesis

The algorithm proposed by Efros & Leung (1999), Algorithm 1, synthesizes texture by modeling the image as a Markov Random Field (MRF) where each pixel's value is dependent only on the values in its immediate neighborhood. The texture synthesis process grows the new image pixel by pixel from an initial seed based on this model. Here are some important features about the algorithm:

[2]Downloaded from `https://git-disl.github.io/GTDLBench/datasets/mnist_datasets/`

- Non-parametric approach: The algorithm does not build an explicit model but estimates pixel distributions dynamically from the sample image.
- Control over randomness: The size of the neighborhood window can be adjusted to control the randomness of the texture, affecting how structured or stochastic the synthesized texture appears.
- Efficiency: A variation of the k Nearest Neighbors technique is employed to find matching neighborhoods quickly.
- Local structure preservation: Emphasis on maintaining the local structure of the texture to ensure visual continuity and realism in the synthesized image.

Algorithm 1 Texture Synthesis by Non-parametric Sampling

Initialize with a small seed taken randomly from the sample image I_{smp}.
while there are unsynthesized pixels in the image I **do**
    Select an unsynthesized pixel p in I.
    Define the neighborhood $N(p)$ of p as a square window centered at p.
    Find all neighborhoods in I_{smp} that are similar to $N(p)$ using a perceptual distance metric d_{perc}.
    Construct a histogram of pixel values from these similar neighborhoods to approximate the conditional probability distribution $P(p|N(p))$.
    Sample from this distribution to set the value of p.
end while

2.3 Parallelization

To speed up the texture synthesis process, we parallelize the algorithm by dividing the image into blocks and synthesizing each block independently, given that the processing of each pixel (and its surrounding window) is somewhat independent. We use the Python `multiprocessing` library to create a pool of worker processes that can work on different blocks concurrently:

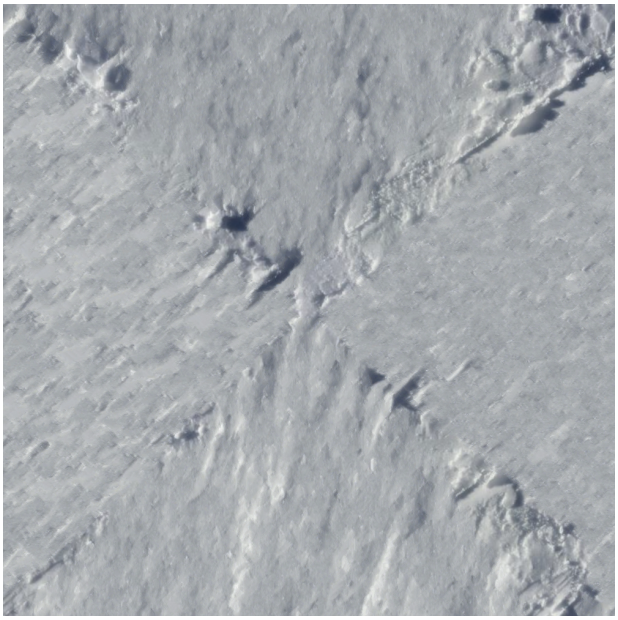
- Split the work: Distribute the processing of each pixel to different CPU cores. This can be done by creating a list of tasks where each task contains the information needed to process a pixel.
- Process pool: Use a multiprocessing pool to process these tasks concurrently.
- Result collection: Collect the results from each task and update the main data structures accordingly.

With this parallelization strategy, we can achieve a significant speedup (depending on hardware, around $8\times$) in the texture synthesis process.

3 Results

Here we present the results of applying the non-parametric texture synthesis method to the textures described in the previous section.

(a) Texture 2



(b) Synthesized Picture 2

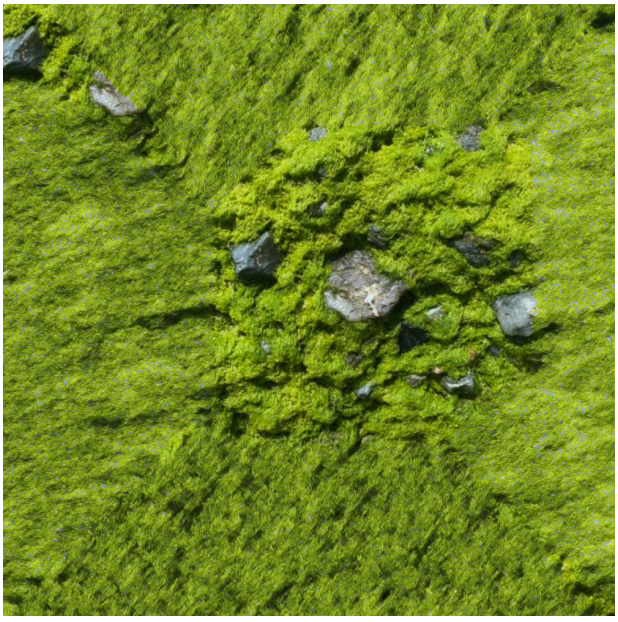
Figure 4: Synthesis of Texture 2

(a) Texture 4

(b) Synthesized Picture 4

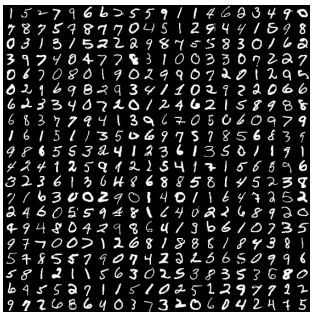
Figure 5: Synthesis of Texture 4

(a) Texture 9

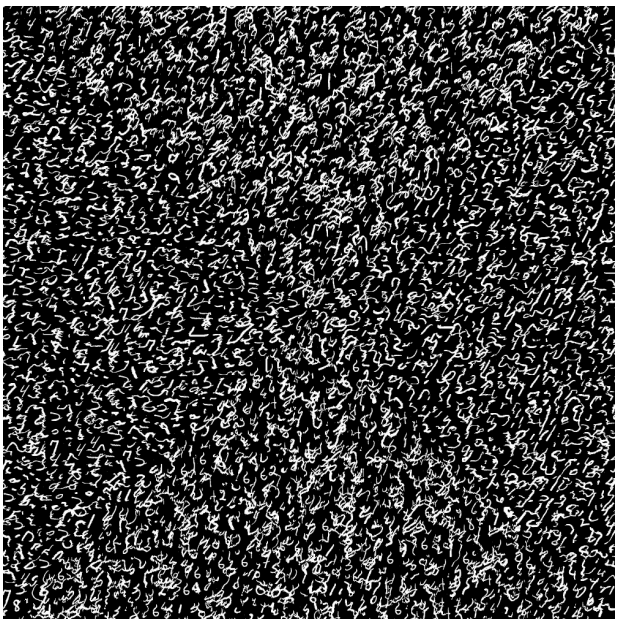


(b) Synthesized Picture 9

Figure 6: Synthesis of Texture 9

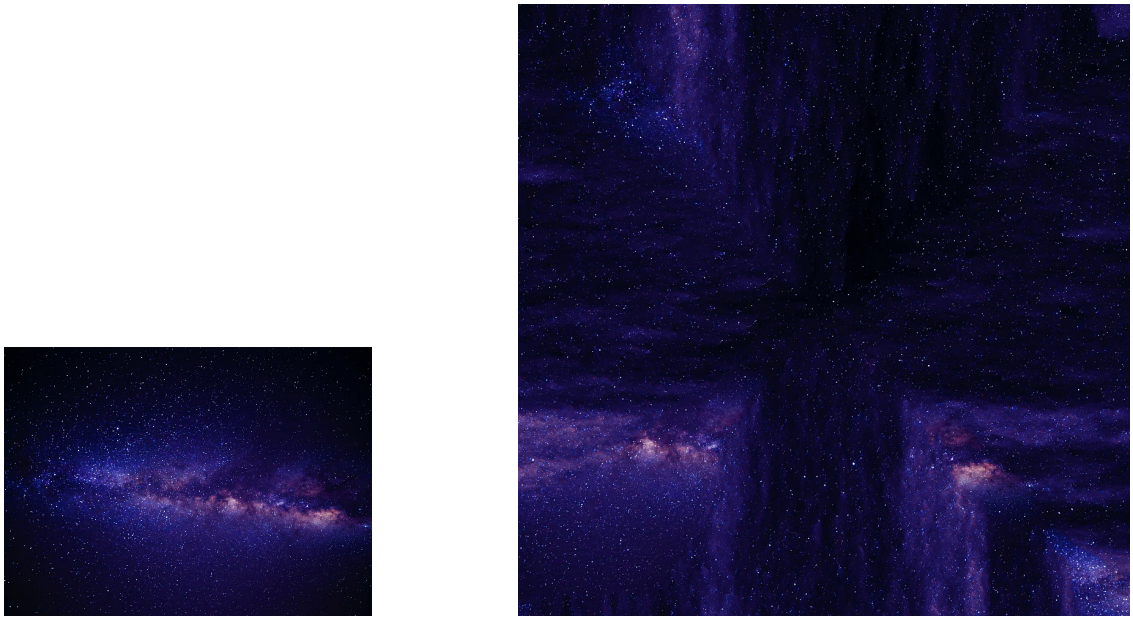


(a) MNIST Digits



(b) Synthesized MNIST Digits

Figure 7: Synthesis of MNIST Digits



(a) Galaxy Picture (b) Synthesized Galaxy Picture

Figure 8: Synthesis of Galaxy Picture