Technology
Arts Sciences
TH Köln

Cologne University of Applied Sciences

Faculty of Computer Science and Engineering Science

MASTERTHESIS

# Texture Asset generation through Transformer Models

Cologne University of Applied Sciences
Campus Gummersbach
Master Digital Sciences

written by:
Dennis Gossler
11140150

| | |
|---|---|
| **First examiner:** | Prof. Dr. Olaf Mersmann |
| **Second examiner:** | Prof. Dr. Boris Naujoks |

## Abstract

Blob

# Contents

# List of Figures

# List of Tables

# 1 Introduction

Transformers are predominantly utilized in generating text for Large Language Models (LLMs) like GPT (Radford et al., 2019) and LaMMA (Touvron et al., 2023). This thesis aims to explore the application of conventional Transformer-based architectures for the generation of texture assets, diverging from the traditional image generation methods primarily relying on stable diffusion techniques or Generative Adversarial Networks (GANs).

To achieve this, three distinct models are developed and trained, each incorporating unique modifications to harness the principal benefits of Transformer-based models for image texture generation. The dataset required for training these models is compiled from a variety of sources, necessitating thorough cleaning and analysis to ensure usability. The objective is for these trained models to be capable of generating basic floor textures suitable for use as assets in video games. These developed models are compared using a specific set of metrics. Furthermore, the development process will be briefly outlined, highlighting the differences from conventional programming workflows typically encountered.

## 1.1 Related work

The exploration of machine learning models for image generation has been a significant area of research lately, with notable advancements from Generative Adversarial Networks (GANs) to state-of-the-art diffusion models. This section reviews the seminal works and recent innovations in the field, particularly focusing on image/texture generation and the application of Transformer models in the context of images, laying the foundation for the current study's approach to image generating.

**Generative Adversarial Networks (GANs):** Since their introduction by (Goodfellow et al., 2014), GANs have been a cornerstone in the field of generative models, especially for image generation tasks. Works by (Radford et al., 2016), introducing the DCGAN architecture, demonstrated the potential of GANs in producing high-quality images. The adaptability of GANs has been explored in various contexts, including texture synthesis (Xian et al., 2018), showcasing their capability to generate seamless textures for different materials.

**Diffusion Models:** Diffusion models represent a cutting-edge development in the field of generative models, that demonstrate remarkable capabilities in image generation by iterative denoising a random signal to produce detailed images. The process, initially introduced by (Sohl-Dickstein et al., 2015), involves gradually adding noise to an image across several steps and then learning to reverse this process. Stable diffusion, a term often associated with these models, refers to the technique's ability to maintain stability throughout the noise addition and removal process, ensuring high-quality image synthesis. The paper "Diffusion Models Beat GANs on Image Synthesis" (Dhariwal & Nichol, 2021) further refined this concept with models like DDPM, showcasing exceptional fidelity in generated images. This approach contrasts traditional models by focusing on the controlled removal of noise, leading to the generation of coherent and visually impressive images.

**Transformers in Image classification:** The success of Transformer models in natural language processing, as seen with architectures like GPT (Radford et al., 2019) and LaMMA (Touvron et al., 2023), has inspired their application in image-related tasks. The Vision Transformer (ViT) by (Dosovitskiy et al., 2021) marked a significant leap, applying Transformers directly to sequences of image patches for classification tasks. This idea was extended to image generation through architectures like the VideoGPT by (Yan et al., 2021), which demonstrated that Transformer models could generate coherent and detailed videos.

**Texture Generation with Transformers:** TransGAN by (Jiang et al., 2021) revolutionizes image generation with a Transformer-based GAN architecture, moving beyond traditional CNN approaches. It features a memory-efficient generator and a nuanced, multiscale discriminator, both utilizing transformer blocks. With advanced training techniques to overcome common GAN challenges, TransGAN produces high-quality images, showcasing the potential of Transformers in this new domain.

## 1.2 Infrastructure for Model Development

To develop and train the models in this thesis, a powerful computing infrastructure is necessary to manage the extensive datasets and the substantial computational requirements for model training. Unlike conventional development environments where a standard laptop or desktop may suffice, most of the models in this thesis demand a more capable infrastructure. Therefore, a high-performance computing system situated in Berlin is used for the model training processes. This system contains an array of (NVIDIA Tesla A100 80 GB) GPUs, (INTEL Ice Lake 8360Y) CPUs and a significant quantity of RAM. Such a configuration, especially the substantial GPU memory, enables the training and execution of larger models that would be possible on a home workstation. The development of these models is carried out using Python and PyTorch, with the code being crafted in Visual Studio Code and managed through version control with Git. The model development and initial code testing are done on a local machine, reserving the high-performance system exclusively for the final training phases. This approach diverges from standard practices, where often both development and execution occur on the same development platform.

Ensuring the code is free of errors prior to giving the task of training the model to the high-performance computing system is crucial, as discovering bugs in the training process can be exceedingly time-consuming. For instance, to endure a training session that extends for 30 hours, only to realize it terminated prematurely due to script errors.

## 1.3 Data

This section describes the methods used for gathering, cleaning, and analyzing data in a research thesis on textures. Essential for training a machine learning model, the data is carefully collected from various sources, cleaned to maintain uniformity, and examined for patterns, with a focus on color distribution.

### 1.3.1 Data Retrieval

On the internet, a wide variety of textures can be found, but not all of them are suitable for this task. The textures should be seamless, devoid of shadows, and free from any objects. Textures of floors, such as carpets, tiles, wood, concrete, and more, were utilized. Two approaches were employed to acquire the data for this thesis.

- Web Data Collection

  The data for this project was obtained from various online sources. Numerous free texture providers, such as textures.com, texturehaven.com, and others, were utilized for data acquisition. Due to the limitation of downloading one texture at a time from most websites, a series of scripts were developed to compile a list of suitable textures and automate the downloading process. These scripts were created using UiPath and Python.

- Video Game Textures

  The second approach involved using textures from video games. The advantage of this approach is that these textures are already seamless and often of high quality and quantity. However, a drawback is that these textures can be very repetitive. To obtain these textures, downward-facing recordings of the game were made, and the textures were extracted from the video. The major challenge with this approach is the need to disable shadows and all UI elements (HUD elements) in the game, which is not always possible.

### 1.3.2 Data Cleaning

To ensure that the data is consistent and free from elements that could corrupt the model, various cleaning steps were applied. For example, all images containing 3D objects were removed, especially those gathered from video games. During the recording of the floor, unwanted debris or pieces of wood were often present, and all extracted frames were manually checked.

In the case of web-gathered textures, there were different folder structures, and it was necessary to standardize them across all data folders. Additionally, some of them had associated files that were irrelevant to this use case and needed to be discarded.

All the images were in high-definition (HD) quality, with a height of approximately 1024 pixels.

| Dataset | Size | Number of Images |
|---|---|---|
| Free PBR | 452.0 MB | 263 |
| Polyhaven | 298.0 MB | 439 |
| Poliigon | 70.4 MB | 49 |
| Minecraft-Textures | 636.0 MB | 493 |
| CsGoFloor-Textures | 18.3 GB | 44540 |
| Combined | 20.2 GB | 45784 |

Table 1: Datasets collected for this thesis

### 1.3.3 Patterns in the data

To examine whether the dataset encompasses a broad spectrum of colors, multiple plots are created. These plots illustrate the color distribution within the datasets, providing insights into the diversity of colors present. Prior to plotting, a comprehensive pixel count across all images is conducted. For instance, if an image features 10 pixels of the color $(255, 0, 0)$, this count is added to a dictionary. Should the subsequent image in the dataset contain 5 pixels of the same color, these are also incorporated into the dictionary, cumulating a total of 15 for that specific color. This process is repeated for each color encountered, aggregating the counts to yield the overall color frequency within the dataset.

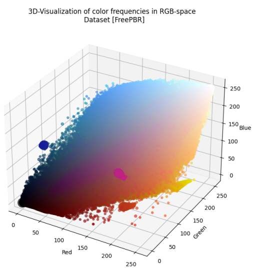
```python
color_counts = {}
for i, (data, _) in enumerate(dataset):
    # data is a tensor of shape (3, height, width)
    pixel_rgb_array = (data.view(3, -1).t() * 255).to(torch.int32)

    for pixel_color in map(tuple, pixel_rgb_array):
        if color in color_counts:
            color_counts[pixel_color] += 1
        else:
            color_counts[pixel_color] = 1
```

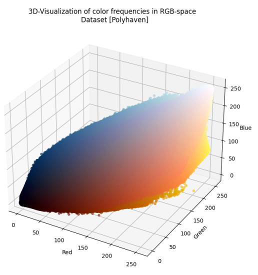
After analyzing the dataset through this method, visual representations of the color distributions were produced using Python and Matplotlib. These plots provide a three-dimensional view of the RGB color space, where the X, Y, and Z axes correspond to the Red, Green, and Blue color values, respectively, each ranging from 0 to 255.

$$\text{size} = \log(\text{count of color}) \times 20$$

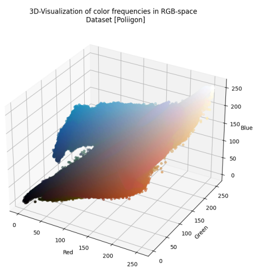
The size of each plotted point is calculated based on the logarithm of the color count, scaled by a factor of 20.



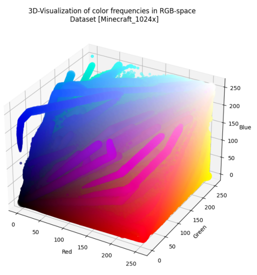
(a) FreePBR



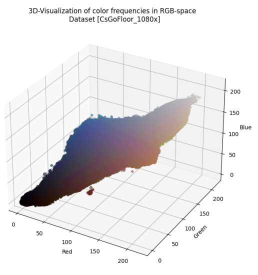
(b) Polyhaven



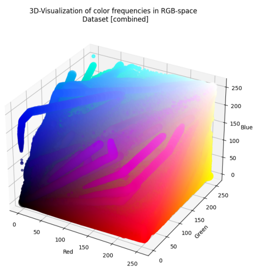
(c) Poliigon



(d) Minecraft-Textures



(e) CsGoFloor-Textures



(f) Combined

In the figure above, the color distributions of the individual datasets are shown. The first five subfigures represent the color distributions of the individual datasets, while the last subfigure (1f) shows the combined color distribution of all datasets. The color distributions of the individual datasets are quite similar, except for the Minecraft-Textures dataset, which is way more colorful than the others. The combined figure is a combination of all the individual datasets, and it is evident that the color distribution is quite diverse. This is a positive sign, as it indicates that the dataset is not focused on only a specific color spectrum.

### 1.3.4 Data Synchronization

In the thesis, a manual data synchronization routine is established to maintain data consistency between the supercomputer located in Berlin and the local workstation.

## 1.4 Models

(LLMs, basic idea, roll model, spiral model)

### 1.4.1 Large Language Models (LLMs)

Large Language Models (LLMs) are a class of machine learning models that have gained significant attention in recent years due to their ability to generate coherent and contextually relevant text. These models are trained on vast amounts of text data, enabling them to understand and generate human-like text. The GPT models by OpenAI (Radford et al., 2019) is a prominent example of an LLM, capable of generating human-like text and performing a wide range of language-related tasks. Like LLMs, the models developed in this thesis predict the next thing in a sequence. But instead of predicting the next word in a sentence, they predict the next pixel in an image. This is achieved by treating the image as a sequence of pixels and using the transformer architecture to predict the next pixel in the sequence.

### 1.4.2 Adapting Transformer Architecture for Image Prediction

The Transformer architecture, introduced by (Vaswani et al., 2023), has significantly impacted the field of natural language processing (NLP). Its widespread adoption across a variety of language tasks, such as machine translation and text generation, highlights its transformative influence. At the heart of the Transformer's success is the self-attention mechanism, which allows the model to weigh different portions of the input data dynamically. This critical feature enables the detection of long-range dependencies and a deeper understanding of the context within the input, making the architecture highly effective for complex NLP tasks.

Building upon this foundation, this thesis explores the extension of the Transformer architecture from its traditional role in NLP to the domain of image prediction. This adaptation employs the architecture's fundamental principles, especially the self-attention mechanism. By treating images as sequences of pixels, the Transformer architecture is applied to predict subsequent pixels in an image sequence, showcasing its potential versatility beyond text-based applications.

### 1.4.3 Column Image Transformer

In the context of this thesis, a model termed the Column Image Transformer (CIT) has been conceptualized and developed. This model embodies an adaptation of the conventional transformer architecture. Distinctively, the CIT model diverges from traditional image processing techniques by segmenting the image into vertical slices or columns of pixels. This segmentation allows for a method where each column is processed on its own, following the "B" batch dimension in the model's structure.

The adaptation of self-attention for image prediction involves sequentially processing the image, similar to text in natural language processing. However, instead of words or characters, the sequence consists of pixels. In the Column Image Transformer (CIT) model, the image is divided into columns, and the self-attention mechanism is applied to understand the relationships between pixels within each column. This should enable the model to predict the properties of subsequent pixels in a column by considering the context provided by preceding pixels.

### 1.4.4 Spiral Image Transformer

The second approach is represented by the Spiral Image Transformer (SIT). Unlike its predecessor, the Column Image Transformer (CIT), the SIT model employs a contextually spiral pattern. This architecture enables the generation of images starting from a central point and expanding outward (see ...). In the SIT model, the batch dimensions correspond to distinct images, whereas the H dimension represents the spiral context. Similar to the CIT model, the C dimension denotes the color channels.

One of the pivotal enhancements of the SIT model is its ability to analyze adjacent pixels on the horizontal axis, in contrast to the CIT model's limitation to columnar pixel analysis. This feature is particularly beneficial for interpreting textures with intricate patterns, such as diagonal ones, thereby offering an advantage over the Column Image Transformer. However, it is important to note that the SIT model operates within a constrained area of the image due to its 2D context. This limitation necessitates the use of only a portion of the image area, specifically a sector determined by the square root of the total area available to the Column Image Transformer (CIT), with an equal context length.

In the Spiral Image Transformer (SIT) model, the self-attention mechanism is adapted to analyze pixels in a spiral pattern. This approach allows the model to evaluate the context in a manner that incorporates both the immediate neighborhood and the broader context of the image, facilitating the prediction of pixel properties in a way that captures complex, two-directional patterns and textures. The self-attention mechanism's ability to dynamically focus on different parts of the spiral sequence should enable these models to generate coherent predictions for the next pixel, based on the learned importance of each pixel to the others.

# 2 Experiment

## 2.1 Transformer behind the models

(Illustration, Tensor board)

### 2.1.1 evaluate the results

## 2.2 Roll model

(explanation, generating new content, , , )

### 2.2.1 Classification or Regression

### 2.2.2

### 2.2.3 Discriminator

## 2.3 Spiral model

(explanation, Data to Spiral form, positional embedding, )

## 2.4 Problems

(layer norm(sigmoid vs clamp), color shift to gray (illustrations of average color), Text tokens vs imgs tokens)

# 3 Conclusion

## 3.1 Roll model

(strength, weaknesses)

## 3.2 Spiral model

(strength, weaknesses)

## 3.3 LLM Scaling Laws

## 3.4 Stable diffusion/ GANs with convolutional neural network

## 3.5 Further research

# A Appendix

## A.1 Unterabschnitt von Appendix

ABCDE

ABC

ABC

## B Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht.
Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

(Ort, Datum, Unterschrift)

## References

Dhariwal, P., & Nichol, A. (2021). Diffusion models beat gans on image synthesis.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks.

Jiang, Y., Chang, S., & Wang, Z. (2021). Transgan: Two pure transformers can make one strong gan, and that can scale up.

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., & Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., … Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). Attention is all you need.

Xian, W., Sangkloy, P., Agrawal, V., Raj, A., Lu, J., Fang, C., Yu, F., & Hays, J. (2018). Texturegan: Controlling deep image synthesis with texture patches.

Yan, W., Zhang, Y., Abbeel, P., & Srinivas, A. (2021). Videogpt: Video generation using vq-vae and transformers.