Brushstrokes of Identity: Urban Identity Reimagined with AI and Art

Ounadi Ikram

June 12, 2025

Introduction

This study investigates the complex link between my artistic expression and urban identity, using cutting-edge approaches like Cycle GANs [1]and Neural Style Transfer [2]. The project begins in two unique phases, each aiming at rewriting the visual tales of Klagenfurt and Udine through the eyes of renowned artists. Initially, I am immersed in the world of Rembrandt, as the Dutch master's strokes bring life to the streets of Klagenfurt, while Bellotto and Canaletto recover Udine with their individual creative styles. However, the research does not end there; in the project's second act, it transforms into a dynamic interchange of urban identities, beyond the boundaries of traditional artwork. Here, Klagenfurt and Udine participate in a reciprocal visual encounter, delivering a raw and unfiltered view of our shared history and transformation. The project follows a well planned framework, with the first section devoted to describing project specifics. Here, I discussed the two parts of my project. The second section digs into the complex relationship between AI and art, examining how these novel tools interact to reshape urban environments and my artistic expression. The third section is dedicated to methodological specifics, such as data collection and preparation, as well as the architectural complexities of the used models, which are thoroughly detailed to offer a thorough grasp of the project's framework. Finally, the long-awaited findings. without neglecting the SEL elements discussion. The inspiration for this project came from a revelatory experience I had while playing a puzzle game including Van Gogh's "Starry Night," which sparked my desire to investigate metropolitan settings through the views of other artists. The original idea for translating Udine to Klagenfurt and vice versa arose from a joint debate between my professor and me, motivated by our mutual desire to make a significant contribution in both the intellectual and artistic fields. As the project progresses, my goal is not only to reveal interesting insights into the intersection of art and AI, but also to pave the path for future advances in urban storytelling and creative expression. Through tireless pursuit and iterative improvement, I want to chart a road towards a greater comprehension of the symbiotic relationship between my creativity, technology, and urban identity.

1 Project Details

Project: AI and Art
Category: Artificial Intelligence

1.1 Artistic Identity Remix: A Fusion of Legends in Klagenfurt and Udine.

I go on a creative journey, reimagining Klagenfurt and Udine from the viewpoints of three renowned painters. Fueled by an overwhelming appreciation for both cities, my goal is to capture and transmit their distinct identities via the varied artistic techniques of these renowned persons. Using both Cycle GANs and Neural Style Transfer, this transformational trip aims to enrich the visual tales of Klagenfurt and Udine by providing a multifaceted depiction of their respective identities.

1.2 Urban Reverie: A Fantastical Merge of Klagenfurt and Udine's Architectural Spirits

At this point in the project, my study transcends reality as Klagenfurt and Udine engage in a fictitious architectural connection. Moving beyond geographical boundaries, this phase encourages a solo effort to imagine the connected identities of different locations. The goal is to demonstrate a symbiotic link that goes beyond traditional visualisation by combining contemporary architecture rendering techniques with the revolutionary power of Cycle GANs and Neural Cycle GANs. This novel technique seeks to give an illustration of interconnected history, highlighting the combination of Austrian aesthetics and Venetian elegance inscribed in every architectural detail.

2 AI & Art Integration

AI & Art: Exploring the intersection of AI and artistic expression through various techniques.

2.1 Artistic works by Bellotto, Rembrandt, and Canaletto

2.1.1 Bernardo Bellotto

Bernardo Bellotto, born in Venice in 1721, is most known for his vedute paintings, which vividly represent European towns in astonishing detail. Bellotto's art, which he learned from his uncle Canaletto, is the apex of the 18th-century veduta tradition. Critics used to reject impressionist painters like Bellotto as lacking skill, but his contributions to impressionist painting are now recognised. Bellotto's paintings include wide brushstrokes and lack precise lines, which are typical of the impressionist style. His outdoor pictures, which are rich in

architectural detail, represent both urban and rural landscapes. Bellotto's grasp of light and atmosphere, particularly when painting reflections on water, lends depth and vitality to his works. Bellotto's paintings elicit nostalgia and amazement due to his mastery of colour and arrangement.

2.1.2 Canaletto

Canaletto, an Italian artist from the 18th century, is well-known for his vedute paintings, which depict complex vistas of European towns. His meticulous representation of building detail and masterful use of light make his paintings renowned examples of impressionist art. Canaletto's paintings take viewers to Venice's bustling streets and canals, portraying the city's ambiance with extraordinary precision. His compositions, which frequently include huge cityscapes, are lively and brightly lit. Canaletto's paintings, with their excellent use of perspective and atmospheric effects, enable viewers to explore the dynamic world he so painstakingly depicts.
As shown in Figure 1 there is a sample from the data of Bernardo Bellotto and Canalleto .

Figure 1: Sample from the data of Bernardo Bellotto and Canaletto

2.1.3 Rembrandt Harmenszoon van Rijn

Rembrandt, born about 1606 in Leiden, is widely regarded as a pioneer of impressionist art. Critics previously misunderstood his method, but today his works are considered as genre masters. Rembrandt's paintings, distinguished by broad brushstrokes and the purposeful avoidance of harsh lines, depict a wide range of themes with depth and passion. His command of light and shadow, especially in outdoor landscapes, gives his paintings a feeling of authenticity and intensity. Rembrandt's paintings go beyond basic reproduction by expressively depicting the human experience.
As show in the Figure 2 there is a sample of the Rembrandt dataset.

Figure 2: Sample from the data of Rembrandt

2.2 Exploring the Frontiers of AI in Image Creation and Manipulation

In the realm of AI-powered picture production and manipulation, numerous prominent programmes have developed, each with its own set of skills and uses. Mid-journey stands out as a popular text-to-image generating tool, featuring a free tier for non-commercial usage and membership options for more advanced access. OpenAI's DALL-E 2 excels in image-to-text creation, but with different output quality depending on input quality, and is commercially viable. Notably, Adobe Illustrator is preparing to add GAN-based picture editing features, which might revolutionise the creative process. NVIDIA's StyleGAN has received notice for producing high-quality AI-generated graphics, and its new move into text-to-video models demonstrates their dedication to innovation. Furthermore, approaches like as CycleGAN, which uses cycle consistency loss to translate unpaired data, and Neural Style Transfer, which allows for seamless merging of content and style pictures, demonstrate the variety and sophistication of modern AI image editing technologies.

3 Methodology

3.1 Data Collection

The acquisition of datasets for the machine learning research required thorough sourcing and curation to verify the data's quality and integrity. The datasets covered a wide range of themes, styles, and settings, allowing for full training and assessment of the AI models.

3.1.1 Data collection of the artistic identity remix

Klagenfurt photos (108 photos): This collection is made up of 512x512 images of various scenes and sites in Klagenfurt. The photographs were sourced from Google Pictures [3] and iStock[4].

To get these photographs, I searched Google Pictures and iStock for the keyword "klagenfurt". To collect the needed photographs, I used the "Download All Images" extension [5] for Google Chrome. This extension enabled the quick collecting of numerous photos from online searches.

Rembrandt Paintings (264 Images): This collection of paintings by the famed Dutch artist Rembrandt was curated using the Kaggle dataset "Collections of Paintings from 50 Artists" by Paul Timothy Mooney [6].

Udine Photos (247): A wide collection of photographs of size 512x512 portraying the essence of Udine, gathered from Google Image [7] and Istock [8]. Some of the photos were taken by myself , but they were gathered the same way using the "Download All Images " extension [5].

Italian Artists' Paintings (148 images): This collection includes paintings by Bernardo Bellotto and Canaletto, sourced from reputable art repositories such as WikiArt [9, 10] and Google Arts and Culture [11, 12]. To guarantee a representative sample, each artwork was carefully assessed for its artistic value and technical excellence.

Challenges Encountered:

- Finding a means to change the size of the photographs without losing clarity and quality was difficult.
- The presence of text overlays, watermarks, and personal photographs added complexity to the curation process.
- Meticulous attention to detail was required to maintain dataset integrity amidst diverse image sources and quality variations.

3.1.2 Data collection for urban reverie

- The dataset for **Udine** comprises **734 images**, while **Klagenfurt** consists of **628 images**.
- Images were collected from Instagram accounts using the ToolMaster [13] application which is is a sophisticated software tool for getting and organising photographs from Instagram accounts. ToolMaster's powerful capabilities enable the seamless collecting of photographs from several Instagram accounts, allowing users to efficiently manage and organise their image collections.
- Instagram accounts used for data collection include **"my_udine"** [14] for Udine and **"klagenfurt"** [15] for Klagenfurt.

Challenges Encountered:

- Klagenfurt dataset included numerous lake images and festival scenes like the Krampuslauf, which were less relevant for the intended analysis.
- Udine's dataset predominantly featured architectural photographs, reflecting the city's main attractions, particularly its historic city center.
- Identifying suitable tools for data collection posed a challenge.
- Despite challenges, the data collection process provided insights into the unique vibes and characteristics of the two cities, as reflected in the content shared on their respective Instagram accounts.

Despite these challenges, the dataset collection phase served as a crucial foundation for subsequent stages of model development and experimentation. By prioritizing quality and diversity in dataset selection, the project aims to foster robust and generalizable machine learning models capable of addressing a wide range of tasks and applications. In the Figure 3 is a sample from Udine data as for Figure 4 a sample from klagenfurt data set . By the eyes we can somehow detect that Klagenfurt is a vivid colorful city as for Udine it is more historical with a dark vibe .

Figure 3: Sample from the data of Udine

Figure 4: Sample from the data of Klagenfurt

3.2 Data preprocessing

3.2.1 Data preprocessing for CycleGAN

In the preprocessing phase as shown in the Figure 6, essential steps are undertaken to enhance the dataset's quality and suitability for machine learning tasks. This section outlines the crucial preprocessing steps, including resizing, normalization, and random jittering, each contributing to the overall robustness and effectiveness of the machine learning model.

- **Resizing Images:** Uniformity in image dimensions is ensured through resizing, with images adjusted to 512x512 pixels using bicubic interpolation for consistency and computational efficiency.
- **Normalization:** Pixel intensity values are standardized within the range [-1, 1] to promote numerical stability and convergence during model training, enhancing the model's ability to learn meaningful patterns from the data.
- **Random Jittering:** Variability is introduced into horizontal flipping, augmenting the dataset's diversity and robustness.

By integrating these preprocessing techniques, the dataset is curated and optimized for subsequent model training, contributing to the overall success and performance of the machine learning project.

Challenges Encountered:

- Image Distortions: Cropping, zooming, and resizing pictures to 256x256 pixels resulted in noise and imperfections. Image quality was degraded as a result of the flipping and rotating processes.
- Impact on Model Learning: The introduction of noisy and distorted pictures reduced the model's learning and generalisation skills.
- Critical Reevaluation: These problems necessitated a thorough appraisal of the preprocessing pipeline to prevent negative impacts on dataset integrity and quality for future analysis and model building.

```python
#Preprocessing funcitons
#2 normalizing the images to [-1, 1]
def normalize(image):
  image = tf.cast(image, tf.float32)
  image = (image / 127.5) - 1
  return image


#3 random jitters
def random_jitter(image):
  # resizing to 360 x 360 x 3
  image = tf.image.resize(image, [512, 512],
                          method=tf.image.ResizeMethod.NEAREST_NEIGHBOR)

  image = tf.squeeze(image) # Remove the "None" dimension

  # random mirroring
  image = tf.image.random_flip_left_right(image)

  return image
```

Figure 5: Preprocessing for CycleGAN

3.2.2 Data preprocessing for Neural Style Transfer

- Read content and style images from file paths.
- Decode and convert images to `tf.float32` datatype.
- Resize images to a maximum dimension of 512 pixels while maintaining aspect ratio. The first step before the preprocessing is described in the Figure 6 as for the preprocessing it is shown in the Figure 7

```python
[ ] #Load Image
    def load_img(path_to_img):
        max_dim = 512
        img = tf.io.read_file(path_to_img)
        img = tf.image.decode_image(img, channels=3)
        img = tf.image.convert_image_dtype(img, tf.float32)

        shape = tf.cast(tf.shape(img)[:-1], tf.float32)
        long_dim = max(shape)
        scale = max_dim / long_dim

        new_shape = tf.cast(shape * scale, tf.int32)

        img = tf.image.resize(img, new_shape)
        img = img[tf.newaxis, :]  # Add batch dimension
        return img
```

Figure 6: first step before the preprocessing

- Preprocess input images using `tf.keras.applications.vgg19.preprocess_input` function [16] to meet VGG19 model requirements.

```
# Giving Content image as input to VGG19 for test run
x = tf.keras.applications.vgg19.preprocess_input(content_image*255)
x = tf.image.resize(x, (224, 224))

vgg = tf.keras.applications.VGG19(include_top=True, weights='imagenet')
prediction_probabilities = vgg(x)
prediction_probabilities.shape
```

```
TensorShape([1, 1000])
```

Figure 7: Preprocessing for Neural Style Transfer

3.3 Model Architecture

3.3.1 CycleGAN

The CycleGAN model used in this project is based on the TensorFlow tutorial on CycleGAN [17]. TensorFlow is an open-source machine learning framework developed by Google.

CycleGAN Overview

- **Purpose:** CycleGAN enables image-to-image translation of unpaired data without requiring a one-to-one mapping between the source and target domains.
- **Loss Function:** It utilizes a cycle consistency loss to enforce consistency between the original and translated images.
- **Applications:** CycleGAN can be used for various tasks such as photo enhancement, image colorization, and style transfer.

Model Architecture and Hyperparameters

- **Generators (G and F):**
  - G transforms images from domain X to domain Y (G: X -> Y).
  - F transforms images from domain Y to domain X (F: Y -> X).
- **Discriminators (D_X and D_Y):**
  - D_X discriminates between real images from domain X and generated images from domain X (F(Y)).
  - D_Y discriminates between real images from domain Y and generated images from domain Y (G(X)).

```
OUTPUT_CHANNELS = 3

generator_g = pix2pix.unet_generator(OUTPUT_CHANNELS, norm_type='instancenorm')
generator_f = pix2pix.unet_generator(OUTPUT_CHANNELS, norm_type='instancenorm')

discriminator_x = pix2pix.discriminator(norm_type='instancenorm', target=False)
discriminator_y = pix2pix.discriminator(norm_type='instancenorm', target=False)
```

Figure 8: Generators and Discriminators code

- **Learning Rate:** 2e-4 with Adam optimizer.
- **Cycle Consistency Loss Weight (LAMBDA):** 10.
- **Batch Size:** 1.
- **Epochs:** 20.

TensorFlow

- The Pix2Pix model from the `tensorflow_examples` [18] package is imported and used in this project.
- Generators and discriminators from Pix2Pix are adapted for the CycleGAN architecture as shown in 9, utilizing Leaky ReLU activation functions for better gradient flow.
- This facilitates the implementation of CycleGAN using established TensorFlow resources.

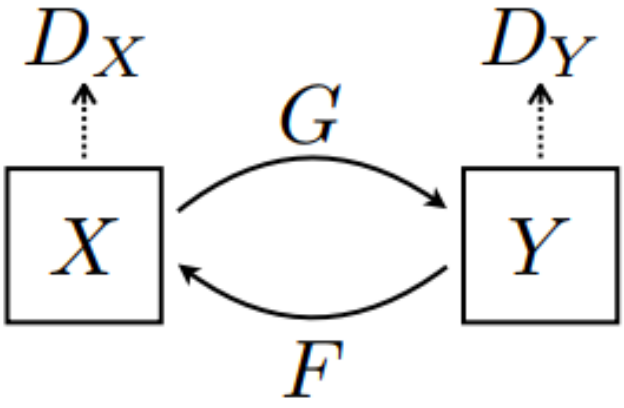

Figure 9: Cycle GANs [17]

Training Details

- The model is trained for 20 epochs using unpaired image datasets.
- During training in which I included screenshots from my code in Figures 10 and 11 , the following loss functions are utilized to update the model parameters:
  - **Adversarial Loss:**
    - **Loss function:** Binary Cross-Entropy.
    - Used to train both generators (G and F) and discriminators (D_X and D_Y).
  - **Cycle Consistency Loss:**
    - **Loss function:** Mean Absolute Error.
    - Ensures that the translated images maintain consistency with the original images after going through both generators (G and F).
  - **Identity Loss:**
    - **Loss function:** Mean Absolute Error.
    - Ensures that feeding an image from one domain into its corresponding generator yields an output close to the original input.

```python
def train_step(real_x, real_y):
    # persistent is set to True because the tape is used more than
    # once to calculate the gradients.
    with tf.GradientTape(persistent=True) as tape:
        # Generator G translates X -> Y
        # Generator F translates Y -> X.

        fake_y = generator_g(real_x, training=True)
        cycled_x = generator_f(fake_y, training=True)

        fake_x = generator_f(real_y, training=True)
        cycled_y = generator_g(fake_x, training=True)

        # same_x and same_y are used for identity loss.
        same_x = generator_f(real_x, training=True)
        same_y = generator_g(real_y, training=True)

        disc_real_x = discriminator_x(real_x, training=True)
        disc_real_y = discriminator_y(real_y, training=True)

        disc_fake_x = discriminator_x(fake_x, training=True)
        disc_fake_y = discriminator_y(fake_y, training=True)

        # calculate the loss
        gen_g_loss = generator_loss(disc_fake_y)
        gen_f_loss = generator_loss(disc_fake_x)

        total_cycle_loss = calc_cycle_loss(real_x, cycled_x) + calc_cycle_loss(real_y, cycled_y)

        # Total generator loss = adversarial loss + cycle loss
        total_gen_g_loss = gen_g_loss + total_cycle_loss + identity_loss(real_y, same_y)
        total_gen_f_loss = gen_f_loss + total_cycle_loss + identity_loss(real_x, same_x)

        disc_x_loss = discriminator_loss(disc_real_x, disc_fake_x)
        disc_y_loss = discriminator_loss(disc_real_y, disc_fake_y)
```

Figure 10: Train step

```
# Calculate the gradients for generator and discriminator
generator_g_gradients = tape.gradient(total_gen_g_loss,
                                      generator_g.trainable_variables)
generator_f_gradients = tape.gradient(total_gen_f_loss,
                                      generator_f.trainable_variables)

discriminator_x_gradients = tape.gradient(disc_x_loss,
                                          discriminator_x.trainable_variables)
discriminator_y_gradients = tape.gradient(disc_y_loss,
                                          discriminator_y.trainable_variables)

# Apply the gradients to the optimizer
generator_g_optimizer.apply_gradients(zip(generator_g_gradients,
                                          generator_g.trainable_variables))

generator_f_optimizer.apply_gradients(zip(generator_f_gradients,
                                          generator_f.trainable_variables))

discriminator_x_optimizer.apply_gradients(zip(discriminator_x_gradients,
                                              discriminator_x.trainable_variables))

discriminator_y_optimizer.apply_gradients(zip(discriminator_y_gradients,
                                              discriminator_y.trainable_variables))
```

Figure 11: Train step

3.3.2 Neural Style Transfer

Model Overview

The neural style transfer tutorial on .. utilizes the VGG19 convolutional neural network (CNN) architecture for feature extraction. VGG19 [19] is a deep CNN with 19 layers which is shown is figure 12, pretrained on the ImageNet dataset.

Components

- **Content Layers:** Intermediate layers, such as 'block5_conv2', capture content information.
- **Style Layers:** Multiple layers, including 'block1_conv1', 'block2_conv1', 'block3_conv1', 'block4_conv1', and 'block5_conv1', are chosen to capture style information.
- **Gram Matrix Calculation:** The Gram matrix, representing correlations between feature maps, is computed from style feature maps.

```
vgg = tf.keras.applications.VGG19(include_top=False, weights='imagenet')
# layers in VGG19
print()
for layer in vgg.layers:
  print(layer.name)
```

```
input_7
block1_conv1
block1_conv2
block1_pool
block2_conv1
block2_conv2
block2_pool
block3_conv1
block3_conv2
block3_conv3
block3_conv4
block3_pool
block4_conv1
block4_conv2
block4_conv3
block4_conv4
block4_pool
block5_conv1
block5_conv2
block5_conv3
block5_conv4
block5_pool
```

Figure 12: VGG19 layers

TensorFlow

- TensorFlow is used to load the pretrained VGG19 model and extract features.
- Custom TensorFlow functions compute the Gram matrix and define style Figure 13 and content losses.
- Automatic differentiation in TensorFlow is utilized for gradient descent optimization.

Training Details

- The model is optimized to minimize the total loss, including style and content losses, along with a total variation loss.
- Adam optimizer is used with specified hyperparameters.
- training step as shown in Figure 14 is performed for **50 epochs**, indicating that the model will iterate over the dataset 50 times, with each iteration updating the model's parameters to minimize the loss function.

```python
class StyleContentModel(tf.keras.models.Model):
  def __init__(self, style_layers, content_layers):
    super(StyleContentModel, self).__init__()
    self.vgg = vgg_layers(style_layers + content_layers)
    self.style_layers = style_layers
    self.content_layers = content_layers
    self.num_style_layers = len(style_layers)
    self.vgg.trainable = False

  def call(self, inputs):
    "Expects float input in [0,1]"
    inputs = inputs*255.0
    preprocessed_input = tf.keras.applications.vgg19.preprocess_input(inputs)
    outputs = self.vgg(preprocessed_input)
    style_outputs, content_outputs = (outputs[:self.num_style_layers],
                                      outputs[self.num_style_layers:])

    style_outputs = [gram_matrix(style_output)
                     for style_output in style_outputs]

    content_dict = {content_name: value
                    for content_name, value
                    in zip(self.content_layers, content_outputs)}

    style_dict = {style_name: value
                  for style_name, value
                  in zip(self.style_layers, style_outputs)}

    return {'content': content_dict, 'style': style_dict}
```

Figure 13: Style content Model

```python
@tf.function()
def train_step(image):
  with tf.GradientTape() as tape:
    outputs = extractor(image)
    loss = style_content_loss(outputs)

  grad = tape.gradient(loss, image)
  opt.apply_gradients([(grad, image)])
  image.assign(clip_0_1(image))
```

Figure 14: Train step

4 Results

4.1 Artistic Identity Remix

4.1.1 Rembrandt x Klagenfurt using CycleGAN

Following the model execution, the output demonstrated satisfactory results, notably resembling Rembrandt's style, particularly in terms of color palette rendition in Figure15

(a) Input (b) Rembrandt art (c) Output

Figure 15: CycleGAN on Rembrandt art

4.1.2 Rembrandt x Klagenfurt using Neural Style transfer

While the outcome of the neural style transfer wasn't as anticipated, it presented an unexpected twist. Despite appearing blurry, the transformed image took on a captivating vibe reminiscent of Mordor from "The Lord of the Rings," turning a beautiful castle into a scene with a unique and intriguing atmosphere .Presented in Figure 16

(a) Input (b) Rembrandt art (c) Output

Figure 16: Neural Style Transfer on Rembrandt art

4.1.3 Belloto and Canaletto x CycleGAN

The result shown in Figure 17 of the CycleGAN transformation on Udine wasn't quite what I had expected. Instead of a significant alteration, it merely infused an Italian vibe into the original image. However, despite not meeting expectations, I still appreciate the outcome because it managed to achieve a satisfactory result in the end.

(a) Input (b) italian art (c) Output

Figure 17: CycleGAN on Italian art

4.1.4 Belloto and Canaletto x Neural Style transfer

I was pleasantly surprised by the outcome presented in Figure 18 of the neural style transfer. Although the image appeared somewhat blurry, it evoked the distinctive style of Bellotto and Canaletto. Considering that most of their paintings depict scenes from Venice, and Udine is in close proximity to Venice, the resulting Italian ambiance was quite fitting. Overall, I had a positive impression of the outcome.

(a) Input (b) italian art (c) Output

Figure 18: Neural Style Transfer on Italian art

While working with AI to change art was certainly entertaining, it also made me consider the profound nature of artistic production. Art cannot be simply tinkered with utilising algorithms and programmes. Despite my efforts to polish and perfect the results via several iterations, I couldn't escape the impression that the spirit of the original artists' work was elusive and irreplaceable. It serves as a reminder that, while artificial intelligence might help with creative endeavours, the actual depth and spirit of art can never be recreated or entirely captured by robots.

4.2 Urban Reverie

4.2.1 Udine to Klagenfurt x CycleGAN

I am very disappointed with the outcomes of my latest experimentation shown in Figure 19. Despite running the model for 20 epochs, there was no increase in output. It appeared that the model had failed to learn or adapt well, with just slight changes in contrast. This conclusion not only used a significant amount of computing resources on Google Colab, but it also made little to no progress towards the desired outcome.

After some thought, I feel the problem is more than just my code or dataset. While 20 epochs may not be a long training time, the lack of meaningful changes reveals deeper concerns. Notably, I saw good outcomes in the early phases of my study when Using CycleGAN, I discovered that the problem may not be intrinsic to my technique or data quality.

(a) Udine (b) Klagenfurt

Figure 19: Cycle GAN Udine to Klagenfurt

4.2.2 Udine to Klagenfurt x Neural Style transfer

The effect of translating Udine to Klagenfurt shown in Figure 20 is blurry, but noticeable. Udine, famed for its rich historical legacy and sombre colour palette, contrasts with Klagenfurt, which combines history and modernity, with brilliant colours and energetic surroundings. Despite the inadequate translation, the NN was able to capture the spirit of Klagenfurt in Udine, providing a glimpse of its vibrant and dynamic ambiance against the city history.

(a) Udine (b) Klagenfurt (c) Output

Figure 20: Neural Style Transfer on Udine to Klagenfurt

4.2.3 Klagenfurt to Udine x Neural Style transfer

When translating from Klagenfurt to Udine shown in Figure 21, it felt as though life was being stripped away from the image. Klagenfurt, known for its vibrant atmosphere and modern charm, appeared muted and devoid of its characteristic liveliness after the transformation. The NN's output didn't fail to capture the essence and energy of Klagenfurt.

(a) Klagenfurt (b) Udine (c) Output

Figure 21: Neural Style Transfer on Klagenfurt to Udine

5 Social, Legal, and Ethical Reflections.

This project explores the interface of art, artificial intelligence (AI), and urban identity using novel approaches such as Cycle GANs and Neural Style Transfer. While the project promises revolutionary insights into urban narrative and artistic expression, it also presents a number of social, legal, and ethical questions.

From a **social** standpoint, the initiative encourages contemplation on the consequences of AI-generated artwork for artistic communities and cultural legacy. As AI technologies blur the distinction between human and machine innovation, issues emerge concerning artists' roles in the digital era and the preservation of creative traditions. Furthermore, the project's focus on reinterpreting urban identities via the perspective of iconic artists like Rembrandt and Canaletto demonstrates the capacity of AI to modify conceptions of cultural legacy and historical narratives, potentially impacting societal attitudes towards urban spaces and their significance.

On the **legal** sides, concerns of intellectual property rights and copyright protection are paramount. While the project recognises the sources of its datasets and artworks, the use of AI algorithms to change existing pictures and styles raises problems about copyright and credit. As AI-generated work becomes more common, ensuring compliance with copyright laws and ethical norms in artistic practice becomes critical for avoiding infringement and maintaining integrity within the creative community.

Ethically, the initiative encourages thought on the proper use of AI technology in creative endeavours, as well as the larger societal ramifications. As AI algorithms edit and produce visual material, bias, transparency, and accountability become increasingly important. Transparency in data gathering and model training, as well as an understanding of the possible social and cultural implications of AI-generated art, are required to negotiate the project's ethical complexity. Furthermore, discussions about the democratisation of art and access to AI tools emphasise the importance of equitable participation and representation in the digital arts landscape.

6 Conclusion

In conclusion, the trip via AI and artistic expression has been both illuminating and challenging. Using techniques like as CycleGANs and Neural Style Transfer, I dug into the depths of urban identity, envisioning places through the eyes of famous artists.

Although triumphs were acknowledged, serious challenges arose throughout the city's involvement in Cycle Gans. This difficulty spurred a shift towards investigating the possibilities of transformers for visual style transfer, inspired by recent advances described in "StyTr2: Image Style Transfer With Transformers" (Deng et al., CVPR 2022)[20].

Moving ahead, the project is still devoted to embracing the changing world of AI picture production and modification. It recognises the subtle dance between technology innovation and the fundamental essence of human creativity, recognising that, while AI may help with artistic processes, the genuine soul of artistry is found in the human touch.

In essence, this piece represents a never-ending path of inquiry, pushing the frontiers of AI integration in art while preserving the timeless attraction of human creativity.

References

[1] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[2] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[3] Google images: Klagenfurt, Accessed: February 2024.

[4] istock: Klagenfurt. `https://www.istockphoto.com/de/search/2/image?family=creative&phrase=klagenfurt`, Accessed: February 2024.

[5] Google Chrome. Download all images. `https://chromewebstore.google.com/detail/download-all-images/ifipmflagepipjokmbdecpmjbibjnakm`, Accessed: February 2024.

[6] Paul Timothy Mooney. Rembrandt collection. `https://www.kaggle.com/code/paultimothymooney/collections-of-paintings-from-50-artists/notebook`, Accessed: February 2024.

[7] Google images: Udine, Accessed: February 2024.

[8] istock: Udine. `https://www.istockphoto.com/de/search/2/image?family=creative&phrase=udine`, Accessed: February 2024.

[9] Wikiart: Canaletto. `https://www.wikiart.org/en/canaletto`, Accessed: February 2024.

[10] Wikiart: Bernardo bellotto. `https://www.wikiart.org/de/bernardo-bellotto`, Accessed: February 2024.

[11] Google arts & culture: Canaletto. `https://artsandculture.google.com/entity/canaletto/m01fbwn?hl=en`, Accessed: February 2024.

[12] Google arts & culture: Bernardo bellotto. `https://artsandculture.google.com/entity/bernardo-bellotto/m04_nhc9?hl=en`, Accessed: February 2024.

[13] Tool master. `https://chromewebstore.google.com/detail/toolmaster-t%C3%A9l%C3%A9chargez-la/bgbclojjlpkimdhhdhbmbgpkaenfmkoe`, Accessed: February 2024.

[14] Instagram: My udine. `https://www.instagram.com/my_udine/?next=%2Fnicelosso%2F&hl=de`, Accessed: February 2024.

[15] Instagram: Klagenfurt. `https://www.instagram.com/klagenfurt/?hl=de`, Accessed: February 2024.

[16] TensorFlow API Documentation: tf.keras.applications.vgg19.preprocess_input. https://www.tensorflow.org/api_docs/python/tf/keras/applications/vgg19/preprocess_input, Accessed: February 2024.

[17] TensorFlow Tutorial: CycleGAN. https://www.tensorflow.org/tutorials/generative/cyclegan, Accessed: February 2024.

[18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks (pix2pix). https://www.tensorflow.org/tutorials/generative/pix2pix, 2018. Accessed: February 25, 2024.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] et al. Deng. Stytr2: Image style transfer with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.