A Generative Style Transfer Model for Artworks

Boran Kılıç-22103444, Defne Yaz Kılıç-22102167, Furkan Gürdoğan-22102960
Ihsan Dogramaci Bilkent University, Ankara, Turkiye

***Abstract*—In this project, a generative style transfer model for unpaired image-to-image translation is implemented. Among different Generative Adversarial Networks (GANs), the Cycle-Consistent Adversarial Networks (CycleGAN) algorithm is used for transforming random real world photos to different art styles. Three styles are chosen among the available style datasets, these are Van Gogh and Ukiyo-e painting as well as a general engraving dataset. The images are generated by training the CycleGAN with the content image training dataset and the style datsets of each style. The generated images are then evaluated by different metrics; firstly by ArtFID followed by ArtDBF, our generated metric for this project.**

I. Introduction

In computer vision, deep learning architectures for image-to-image translation aim to map an input image to an output image over a training set of aligned image pairs. Due to the constraints created by the lack of paired datasets; Cycle GAN, a more flexible and general model was introduced by Jun-Yan Zhu et al. in 2017 [1]. Similar to the other translation models, CycleGAN captures the specific characteristics of one image set and translates them to another image collection like artworks. However, unlike other methods, such as Bayesian frameworks or CoGAN, for unpaired image-to-image translation, the formulation in [1] does not rely on any predefined similarity function linking the input (real image) and output (artwork-styled image).

II. Methodology

A. CycleGAN Overview

While style is inherently subjective, CycleGAN does not learn a specific style from a single artwork. Instead, it learns a distribution of features from a collection of artworks. For example, Van Gogh's paintings typically exhibit warm colors, brush strokes, and impressionist textures. A model trained on a dataset of Van Gogh paintings will learn these characteristics collectively, without being limited to any specific painting. It leverages an entire dataset to avoid overfitting rather than relying on individual artworks. This approach ensures that the model captures the broader sense of the style, making it adaptable and capable of generating diverse and consistent outputs, even when the style itself is subjective.

The core of CycleGAN consists of generators and discriminators. Two generator neural networks, namely G and F perform image translation. Simply put, they learn a mapping between two different visual domains. While maintaining the structural or semantic meaning of an image, they alter its visual characteristics. For the scope of this project, the generators G and F try to produce stylized images indistinguishable from paintings of different artists, while the discriminators D_X and D_Y attempt to classify generated images as fake.

- Forward Translation: $G : X \rightarrow Y$ maps images from domain X (real world images)to domain Y(Paintings).
- Backward Translation: $F : Y \rightarrow X$ maps images from domain Y back to domain X.

On the other side, the two discriminators, namely D_X and D_Y, are used to distinguish real images from generated images.

- D_Y evaluates whether an image belongs to domain Y or is generated by $G(X)$. It outputs a probability of how "real" the image looks.
- D_X differentiates between images from X and those generated by $F(Y)$ [1].

B. Loss Functions

To be able to preserve the meaningful content of the real-world images and also include the painter-specific style, several loss functions are used. They maintain the balance between creativity and realism.

1) Adversarial Loss: Adversarial loss enables the translation of images from one domain to another. It measures how well the generator G fools the discriminator D_Y. G aims to minimize eq 1 against an adversary D that tries to maximize it [2].

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)} [\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)} [\log (1 - D_Y(G(x)))] \quad (1)$$

- $\mathbb{E}_{y \sim p_{data}(y)} [\log D_Y(y)]$: Represents the expectation that real images y from domain Y will be classified correctly by the discriminator D_Y as real. The discriminator is rewarded when it assigns high probabilities to real images from domain Y.
- $\mathbb{E}_{x \sim p_{data}(x)} [\log (1 - D_Y(G(x)))]$: Represents the expectation that generated images $G(x)$ from domain X will be correctly classified as fake by the discriminator D_Y. The generator G tries to minimize this term by making $D_Y(G(x))$ close to 1, meaning the discriminator thinks generated images are real [2][1].

2) Cycle Consistency Loss: Adversarial loss alone ensures that the generated data $G(x)$ or $F(y)$ looks realistic in the target domain. However, it does not ensure that $G(x)$ preserves the content of x. For instance, a picture of a flower might be successfully transformed to resemble a real Van Gogh painting, but without proper constraints, it could no longer resemble a flower. The mapping can be arbitrary as long as it fools the discriminator. To overcome this problem, Jun-Yan Zhu et al. [1] introduced cycle consistency loss. It enforces that an image

translated to another domain and then back to the original domain should return to its original form. Cycle consistency loss has a bidirectional structure:

- *Forward Cycle Consistency:* Ensures $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$.

$$\mathcal{L}_{cyc}(G, F, X) = \mathbb{E}_{x \sim p_{data}(x)} \left[\|F(G(x)) - x\|_1 \right]$$

- *Backward Cycle Consistency:* Ensures $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$.

$$\mathcal{L}_{cyc}(G, F, Y) = \mathbb{E}_{y \sim p_{data}(y)} \left[\|G(F(y)) - y\|_1 \right]$$

Combining the two, lost function can be expressed as follows

$$\begin{aligned} \mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)} \left[\|F(G(x)) - x\|_1 \right] \\ + \mathbb{E}_{y \sim p_{data}(y)} \left[\|G(F(y)) - y\|_1 \right] \end{aligned} \quad (2)$$

- $\mathbb{E}_{x \sim p_{data}(x)} \left[\|F(G(x)) - x\|_1 \right]$: Forward cycle loss: After translating $x \rightarrow Y$ using G, the model should be able to translate back to X using F, producing an image close to the original x, the L1 norm enables to calculate the absolute difference between the original and reconstructed images.
- $\mathbb{E}_{y \sim p_{data}(y)} \left[\|G(F(y)) - y\|_1 \right]$: Backward cycle loss: After translating $y \rightarrow X$ using F, the model should be able to translate back to Y using G, producing an image close to the original y.

3) Identity Loss: As an additional control, identity loss is used to preserve color, structure, and other low-level details during the image translation process [1]. Without identity loss, the generator might apply unnecessary transformations that may change important features, even when an image already belongs to the target style domain. For instance, when converting a Van Gogh painting to another Van Gogh painting, the output should ideally be unchanged.

$$\begin{aligned} \mathcal{L}_{identity}(G, F) = \mathbb{E}_{y \sim p_{data}(y)} \left[\|G(y) - y\|_1 \right] \\ + \mathbb{E}_{x \sim p_{data}(x)} \left[\|F(x) - x\|_1 \right] \end{aligned} \quad (3)$$

- $\mathbb{E}_{y \sim p_{data}(y)} \left[\|G(y) - y\|_1 \right]$: Ensures that when a real image y from domain Y is passed through the generator G, it remains unchanged.
- $\mathbb{E}_{x \sim p_{data}(x)} \left[\|F(x) - x\|_1 \right]$: Ensures that when a real image x from domain X is passed through the generator F, it remains unchanged.

Combining all three loss components, the complete CycleGAN loss function becomes:

$$\begin{aligned} \mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \\ \mathcal{L}_{GAN}(F, D_X, Y, X) \quad + \lambda_{\text{cyc}} \mathcal{L}_{\text{cyc}}(G, F) + \\ \lambda_{\text{identity}} \mathcal{L}_{\text{identity}}(G, F) \end{aligned} \quad (4)$$

Where:

- λ_{cyc}: Weight for the cycle-consistency loss which is tuned as a hyperparameter.
- $\lambda_{\text{identity}}$: Weight for the identity loss which is set as $\lambda_{\text{cyc}}/2$

C. Evaluation Metrics

1) ArtFID: ArtFID is a metric tailored for style transfer evaluations. It produces a distance value by calculating the distance between the styles of the output and the style dataset, and the distance based on content preservation, and multiplying the obtained values to produce the ArtFID value.[3]

a) ArtFID: Content Matching: To determine how well the contents of an input image are preserved in the stylized output, the preferred similarity metric is Learned Perceptual Image Patch Similarity (LPIPS). The content similarity of an input-output image pair is calculated by extracting features from the input and output images and building feature maps using neural networks, then calculating the distance between their feature matrices that correspond to the same layer. The feature extraction can be done with different types of neural networks such as Visual Geometry Group (VGG) or AlexNet.

VGG The VGG network employs 3x3 filters for its convolutional layers, and pooling layers after every two or three convolutional layers to perform max-pooling. Max-pooling is a downsampling technique that aims to reduce the dimensionality of the feature maps and suppress noise. The main advantage of using the VGG network is that it provides a deeper structure to learn hierarchical features from images, since it includes 16-19 convolutional layers depending on the variant used [4].

AlexNet The AlexNet network has eight layers, five of which are convolutional layers (some followed by max-pooling layers) and the last three are fully connected layers. The network uses larger filters than VGG (11x11 in the first layer, for instance), and therefore can capture finer details. The increased filter size and decreased depth compared to VGG means fewer parameters and less computational cost [5].

After the features are extracted and put in feature matrices for the input and the output, the distances between feature matrices are calculated with the Mean Squared Error (MSE), which, given feature matrices A and B, has the formula:

$$\text{Content Distance} = \text{MSE}(A, B) \quad (5)$$

b) ArtFID: Style Matching: For style matching, the Fréchet Inception Distance (FID) metric is used. The FID metric works similarly to LPIPS by extracting feature matrices from the input (real) and output (generated) images using the Inception-v3 network and calculating the distance between the matrices with the Fréchet distance.

Inception-v3 is a convolutional neural network developed by Google for computer vision applications[6]. For feature extraction, the network is pretrained on a database. After training, the lower layers of the network capture the style information while the higher layers capture content information. Therefore, the earlier layers are taken from the network, forming a new model to extract style information from a given image. The real and generated images are then passed through this model, and feature matrices are formed with the extracted features. The similarity of the two matrices is then calculated with the Fréchet distance, giving the FID value as:

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r \Sigma_g\right)^{1/2}\right) \quad (6)$$

Where:

- μ_r and Σ_r are the mean and covariance of the real images, respectively.
- μ_g and Σ_g are the mean and covariance of the generated images, respectively.
- $\mathrm{Tr}(A)$ symbolizes the trace of a matrix A.

After obtaining the content and style similarities shown by MSE and FID as explained, the ArtFID value is calculated by:

$$\text{ArtFID} = (1 + \text{MSE}) \times (1 + \text{FID}) \tag{7}$$

The addition of 1 to both terms in the multiplication accounts for extreme cases where there is no difference between either the style or the content of the input and output datasets, which would give zero as the ArtFID result otherwise.

The ArtFID metric is widely used for style transfer applications and gives relatively accurate results, but it is most successful with large datasets. With smaller datasets, the style matching results may not be as expected based on human perception. Therefore, since the datasets used in this project were on the smaller side due to limited computational power, another metric, namely, the Kernel Inception Distance (KID), was employed for style matching.

2) Kernel Inception Distance (KID): The KID metric uses features extracted from images with the same procedure as FID. The difference of KID is that after feature extraction, the Maximum Mean Discrepancy (MMD) of the feature distributions is calculated. MMD is a statistical tool used to determine whether two given samples are taken from the same distribution. To calculate the MMD, a polynomial kernel function is applied to the extracted features. Then, the squared MMD is calculated by [7]:

$$\begin{aligned} \text{KID} &= \text{MMD}^2(X, Y) \\ &= \mathbb{E}[k(x, x')] - 2\mathbb{E}[k(x, y)] + \mathbb{E}[k(y, y')] \end{aligned} \tag{8}$$

Where:

- X and Y are the output and style feature matrices, respectively.
- The first term is the expected kernel value within the output images.
- The second term is the expected kernel value within the style images.
- The last term is the expected kernel value between the output and style images.

Then, a new error metric, Artistic Distribution-Based Fidelity (ArtDBF), is introduced, defined and calculated by:

$$\text{ArtDBF} = (1 + \text{MSE}) \times (1 + \text{KID}) \tag{9}$$

D. Datasets

In order to accomplish the style transfer task, a content dataset and three different style datasets are used.

1) Common Image Dataset: The content dataset contains various common objects in several different contexts. The images are taken from the COCO dataset [8]. Due to its large quantity, only half of the 2017 validation set containing 2500+ real-life images is used. The dataset is split into 3 sets which are train (70%), test (20%) and validation sets (10%).

Fig. 1. Coco Dataset

Fig. 2. Van Gogh style images

2) Van Gogh Paintings Dataset: The first style dataset used is the Van Gogh dataset taken from kaggle.com [9]. It contains 2000+ paintings drawn by Vincent Van Gogh with style and genre classification. Since the style dataset for style transfer models should contain images in the same color palette and texture, the dataset has been reduced by selecting specific images with oil painting texture and a warm color palette. This task was done manually by deleting the unwanted images. After the selection process, the number of images was reduced to 700.

3) Ukiyo-e Paintings Dataset: This dataset contains 1100+ Ukiyo-e art style images. Ukiyo-e is a style of Japanese art that thrived between the 17th and 19th centuries. Artists in this genre created woodblock prints and paintings depicting a variety of subjects, including beautiful women, kabuki performers, sumo wrestlers, historical and folkloric themes, travel scenes, landscapes, plants, and animals [10]. The texture and color palette of the dataset were usable with a few deletions.

4) Engraving: This dataset contains 750+ Engraving art style images. Engraving involves carving a design into a hard, typically flat surface using tools like a burin. This technique can create decorated items such as silver, gold, steel, or glass objects. Additionally, it produces intaglio printing plates, often made of copper or other metals, which are used to print images on paper. These printed images are also referred to as ”engravings” [11].

Fig. 3. Ukiyo-e style images

Fig. 4. Engraving style images

E. Training Methodology

The preprocessing step is done with VSCode on AMD Ryzen7 CPU. However, the computational capacity of this cpu was not sufficiently fast for this task. Therefore, all the other model trainings and image generations are done on Google Colab Pro using NVIDIA A100 Tensor Core GPU. In addition, all the models and generated images are saved to Google Drive in order to save storage space.

1) Preprocessing: Before training the CycleGAN model, the training images are processed to the required format for the training. The preprocessing involves several steps:

- Resize the Images: Adjust the dimensions of each image to 286x286 pixels using the nearest neighbor interpolation method. This method introduces variability, aiding the model's generalization.
- Random Crop: Extract a random crop of size 256x256 pixels from the resized image in order to standardize the input size and enhance robustness.
- Random Flip: Apply a random horizontal flip to introduce variability and augment the dataset.
- Cast Pixel Values: Convert the pixel values of the image to a float32 data type for compatibility with the training pipeline.
- Normalize the Pixel Range: Scale the pixel values from the range [0, 255] to the normalized range [-1, 1] by dividing each value by 127.5 and subtracting 1.

These steps ensure the dataset is appropriately preprocessed for the CycleGAN model training.

Test sets are also resized and normalized beforehand in order to be able to generate output images.

2) Model Building: The Style Transfer Model was developed using the TensorFlow CycleGAN tutorial. We imported the pix2pix model from the tensorflow_examples module, utilizing two U-Net generators and two discriminator models from that module. After defining the loss functions, we constructed the training unit cell based on the CycleGAN architecture, with the core training step code sourced from the TensorFlow CycleGAN tutorial [12].

3) Hyper-parameter tuning: While training this model, we have dealt with 2 hyper-parameters: the first one was the number of epochs, which stands for one complete pass of the entire training dataset through the CycleGAN, and the other was λ_{cyc}. Since the $\lambda_{\text{identity}}$ is fixed to be half of λ_{cyc}, we refer to λ_{cyc} simply as λ.

Initially, we fixed the lambda parameter at 10 and trained separate models for each style dataset across 10, 20, 30, 50, and 100 epochs. We then generated images using the validation set and evaluated them using our metric, ArtDBF, to determine the epoch count that yielded the lowest ArtDBF score for each style. Subsequently, keeping the selected epoch count constant, we assessed ArtDBF results across four different λ values: 1, 5, 10, and 15. Finally, we generated images using the validation set and evaluated the outputs with ArtDBF, selecting the optimal λ for each style set.

The following tables include the ArtFID results of each style set with different hyper-parameters.

TABLE I
RESULTS OF VAN GOGH WITH DIFFERENT HYPER-PARAMETERS

LAMBDA	EPOCH	ArtDBF	LPIPS	KID	ArtFID
10	10	176.15	1.01	86.44	3461.84
10	20	251.8	1.27	109.75	
10	30	232.62	1.22	103.7	
10	50	330.72	1.35	139.67	
10	100	355.66	1.6	135.63	
1	10	191.21	1.02	93.33	
5	10	222.1	1.08	105.6	
10	10	176.15	1.01	86.44	3461.84
15	10	202.1	0.95	102.64	

The chosen hyper-parameter values are highlighted in yellow on the tables.

4) Training and Evaluation: After the parameters have been chosen, the model has been trained with the specified values. Finally, using the trained models and test sets, final results have been generated and evaluated. The results of each model are discussed in the following section.

TABLE II
RESULTS OF UKIYO-E WITH DIFFERENT HYPER-PARAMETERS

LAMBDA	EPOCH	ArtDBF	LPIPS	KID	ArtFID
10	10	27.14	1.25	11.03	
10	20	27.69	1.46	10.24	
10	30	24.12	1.39	9.06	1655.65
10	50	33.4	1.52	12	
10	100	47.68	1.7	16.64	
1	30	31.66	1.82	10.22	
5	30	26.52	1.48	9.69	
10	30	24.12	1.39	9.06	1655.65
15	30	26.17	1.47	9.59	

TABLE III
RESULTS OF ENGRAVING WITH DIFFERENT HYPER-PARAMETERS

LAMBDA	EPOCH	ArtDBF	LPIPS	KID	ArtFID
10	10	734.18	2.89	187.36	
10	20	160.25	1.41	65.36	
10	30	139.71	1.5	54.76	2100.76
10	50	245.37	1.62	92.32	
10	100	298.84	1.7	109.52	
1	30	236.95	1.66	87.91	
5	30	236.48	1.48	94.2	
10	30	139.71	1.5	54.76	2100.76
15	30	255.63	1.41	105.07	

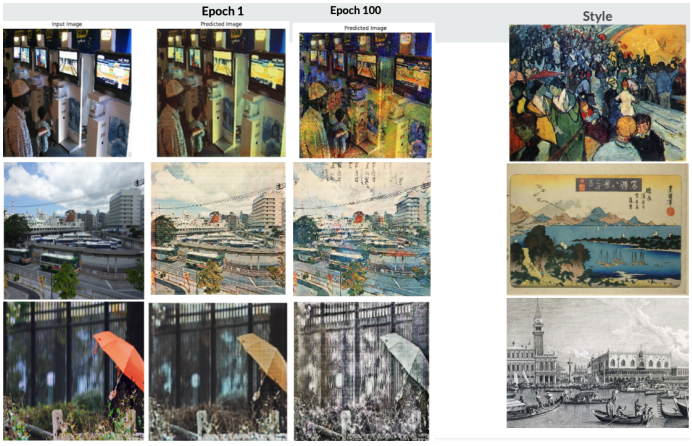

Fig. 5. Generated images with different number of epochs Van Gogh, Ukiyo-e, and Engraving style transfer.

III. RESULTS AND ANALYSIS

For each style, the results were obtained using the optimal values for λ and the number of epochs determined above.

Fig. 7 shows two samples each from the results of the Van Gogh, Ukiyo-e, and Engraving style transfer using their respective λ values for their respective number of epochs.

Fig. 8 shows the distances and similarity metrics.

The expected value from ArtFID for a well-done style transfer is an early 3-digit number or smaller. As can be seen from the results, the models could not reach such a score, even though the results seem accurate to human perception. The reason behind this, as stated in the Error Metrics section, is that the ArtFID metric assumes Gaussian distributions for the features of the style and output images. Since the sizes of these images are limited, the feature distributions cannot converge to Gaussians, and therefore the score is much higher than expected.

Fig. 6. Generated images with different λ Van Gogh, Ukiyo-e, and Engraving style transfer.

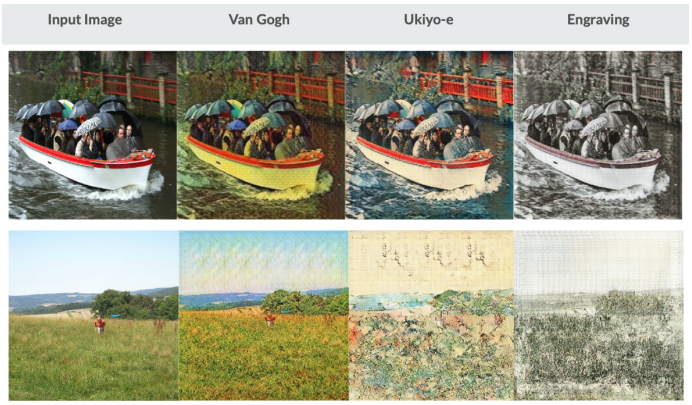

Fig. 7. Results of Van Gogh, Ukiyo-e, and Engraving style transfer.

```
Engraving Results with Tuned Parameters
Content distance: 1.5054219961166382
Style metric: KID=54.76430410879803
ArtDBF value: 139.71311950683594
ArtFID Value: 2100.760498046875

Van Gogh Results with Tuned Parameters
Content distance: 1.0145115852355957
Style metric: KID=86.44452426616874
ArtDBF value: 176.158047607422
ArtFID Value: 3461.84033203125

Ukiyo-e Results with Tuned Parameters
Content distance: 1.3946784764729362
Style metric: KID=9.006960382870381
ArtDBF value: 24.1296396789057
ArtFID Value: 1655.65625
```

Fig. 8. Distances and similarity metrics for the different styles.

It is seen that changing the error metric from ArtFID to ArtDBF significantly improved the obtained score, since the style distance metric KID used in ArtDBF does not have any assumptions about the feature distributions. Thus, the new and improved metric gives scores in the desired interval.

IV. Conclusion

In conclusion, the aim of this project was to implement a generative style transfer model for unpaired image-to-image translation from the COCO common object dataset to images in the styles of Van Gogh, Ukiyo-e, and engraving. For this, the CycleGAN algorithm was chosen and employed. The basic working principle of the algorithm includes a generator creating an image and a discriminator trying to determine whether its input is real or generated. The algorithm continues until the discriminator cannot perform its duty successfully. The loss functions of the algorithm are Adversarial Loss and Cycle Consistency Loss. The Adversarial Loss measures the success of the generator in fooling the discriminator, while the Cycle Consistency Loss measures the success of content preservation.

During the training of the model, two hyperparameters, namely, the number of epochs and the λ value, are tuned by repetitive trials for each style. Those tuned hyperparameters are used for the test and validation of the models from then on.

A frequently used error metric in style transfer applications is ArtFID. The ArtFID value is determined by multiplying the content and style distance values between the input and output images. Due to limited computational loss and dataset sizes, the assumption of ArtFID that the feature distributions are Gaussian is not satisfied, therefore the ArtFID metric is improved by changing the FID style distance metric by KID, which does not have such assumptions. We call this newly defined error metric as Artistic Distribution-Based Fidelity.

As a result, while the ArtFID metric gives results that are far from the intended interval (< 200), the new metric ArtDBF gives sufficiently good measurements. Overall, the model implementation and the results are a success. This project has provided with great insight into the world of computer vision and artificial intelligence, and will prove further useful in our careers.

References

[1] Jun-Yan Zhu et al. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2223–2232. DOI: 10.1109/ICCV.2017.238. URL: https://doi.org/10.1109/ICCV.2017.238.

[2] Ian J. Goodfellow et al. “Generative Adversarial Nets”. In: *arXiv preprint arXiv:1406.2661* (2014).

[3] Matthias Wright and Björn Ommer. “ArtFID: Quantitative Evaluation of Neural Style Transfer”. In: *arXiv preprint arXiv:2207.12280v1* (2022). URL: https://github.com/matthias-wright/art-fid.

[4] GeeksforGeeks contributors. *VGG Net Architecture Explained*. https://www.geeksforgeeks.org/vgg-net-architecture-explained/. [Accessed: Dec. 9, 2024].

[5] Wikipedia contributors. *AlexNet*. https://en.wikipedia.org/wiki/AlexNet. [Accessed: Dec. 9, 2024].

[6] C. Szegedy and et al. “Rethinking the Inception Architecture for Computer Vision”. In: *arXiv preprint arXiv:1512.00567* (Dec. 2015). DOI: 10.48550/arXiv.1512.00567. URL: https://doi.org/10.48550/arXiv.1512.00567.

[7] O. Tunali. *Maximum Mean Discrepancy in Machine Learning*. Onur Tunali Blog. [Accessed: Dec. 9, 2024]. Mar. 2019. URL: https://www.onurtunali.com/ml/2019/03/08/maximum-mean-discrepancy-in-machine-learning.html.

[8] T.-Y. Lin and et al. *Microsoft COCO: Common objects in context*. Accessed: Oct. 19, 2024. 2014. URL: https://cocodataset.org/.

[9] IpythonX. *Van Gogh Paintings*. Accessed: Dec. 8, 2024. 2024. URL: https://www.kaggle.com/datasets/ipythonx/van-gogh-paintings?resource=download.

[10] Wikipedia contributors. *Ukiyo-e*. https://en.wikipedia.org/wiki/Ukiyo-e. [Accessed: Dec. 9, 2024].

[11] Wikipedia contributors. *Engraving*. https://en.wikipedia.org/wiki/Engraving. [Accessed: Dec. 9, 2024].

[12] TensorFlow Developers. *CycleGAN Tutorial*. https://www.tensorflow.org/tutorials/generative/cyclegan. [Accessed: Dec. 9, 2024].