IMPLEMENTATION and ANALYSIS of CYCLE-CONSISTENT ADVERSARIAL NETWORKS for UNPAIRED IMAGE-TO-IMAGE TRANSLATION

CS512 Computer Vision Project Report

Nehal Dhanraj Patil
A20539654
Illinois Institute of Technology

Fall 2024

11/13/2024

MAIN PAPER

Zhu, J., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. IEEE International Conference on Computer Vision (ICCV), 2017.

PAPER LINK: arXiv:1703.10593v7 [cs.CV] 24 Aug 2020

WEBSITE: https://junyanz.github.io/CycleGAN/

The foundation for this project is based on the paper "CycleGAN & Pix2Pix" mentioned above, which implements CycleGAN for image-to-image translation tasks using unpaired datasets.

Objective:

The objective of this project is to implement the CycleGAN architecture to explore unsupervised image translation between two distinct image domains without relying on paired data. The project aims to replicate the results presented in the original paper's implementation & analyze the performance & limitations of CycleGANs.

Responsibilities:

Literature Research, Code Development, Documentation & Presentation, all is done by me.

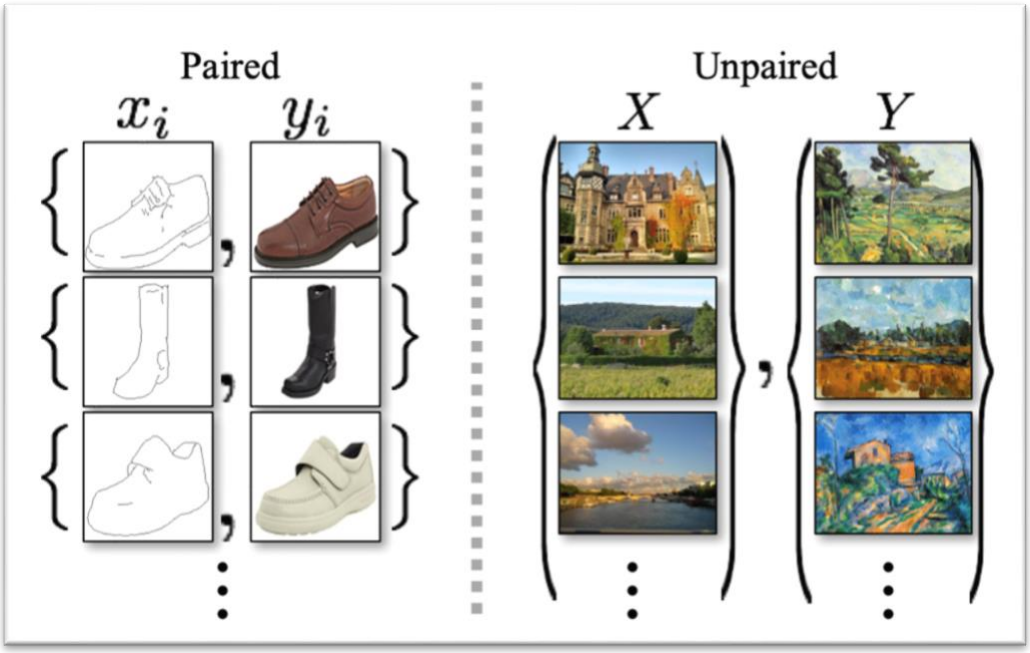


Fig 1: Unpaired & Paired Data Example

PROBLEM STATEMENT

Image-to-image translation refers to the task of converting an image from one domain to another while preserving the underlying content & structure of the original image. Traditional methods for image translation typically require paired datasets, where each image in one domain has a corresponding image in the other domain that depicts the same scene or object. However, collecting such paired datasets can be extremely challenging, time-consuming, & expensive. In many cases, it is even impossible to obtain paired data, such as transforming images between different artistic styles or converting medical images between modalities.

The Cycle-Consistent Generative Adversarial Network (CycleGAN) addresses this limitation by enabling unpaired image-to-image translation. This means that the model can learn to translate between two domains using only unpaired images from both domains. For example, CycleGAN can learn to translate between photos of horses & zebras without needing paired images of the same horse in both forms. The key innovation in CycleGAN is the introduction of a cycle-consistency loss, which ensures that if an image is translated from one domain to another & back again, it should return to its original form.

Importance:

Unpaired image-to-image translation opens a wide range of possibilities across various fields:

- Photography & Art: Artists can use CycleGAN to apply different styles to their work without needing paired examples. For instance, converting a photograph into a Van Gogh-style painting.
- Medical Imaging: In medical imaging, CycleGAN can be used to transform images between different modalities (e.g., MRI to CT scans) without requiring paired scans from the same patient.
- Data Augmentation: In object detection tasks, CycleGANs can generate synthetic data that mimics real world variations (e.g., daytime scene to nighttime scene), which can improve model robustness.

By eliminating the need for paired data, CycleGAN significantly expands the applicability of image translation tasks & reduces the reliance on costly or unavailable datasets.

Objective of this Implementation:

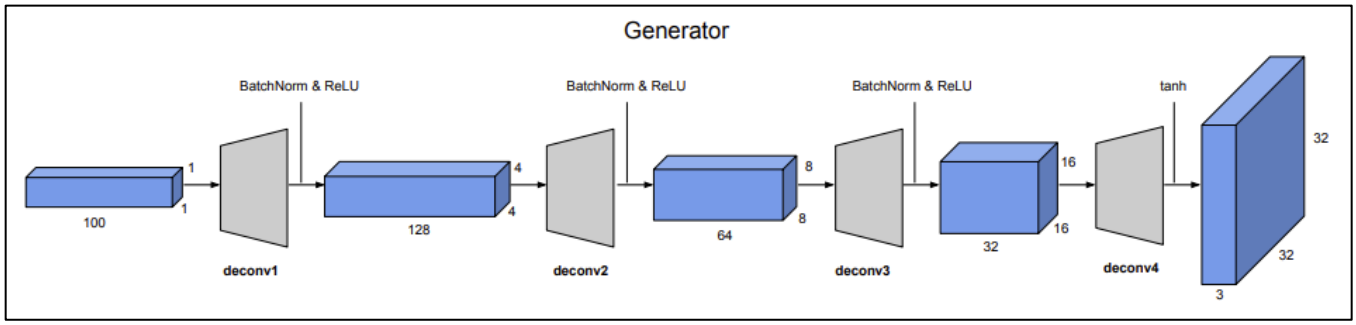
The objective of this project is to implement & validate CycleGAN based on the original paper by Zhu et al. The project focuses on translating images between two domains (**Monet** & **photo**) using unpaired datasets. The implementation will be evaluated both qualitatively (by visual inspection of generated images) & quantitatively (using metrics like Fréchet Inception Distance). The goal is to assess how well CycleGAN performs unpaired image translation & identify any strengths or limitations in its performance.

PROPOSED SOLUTION

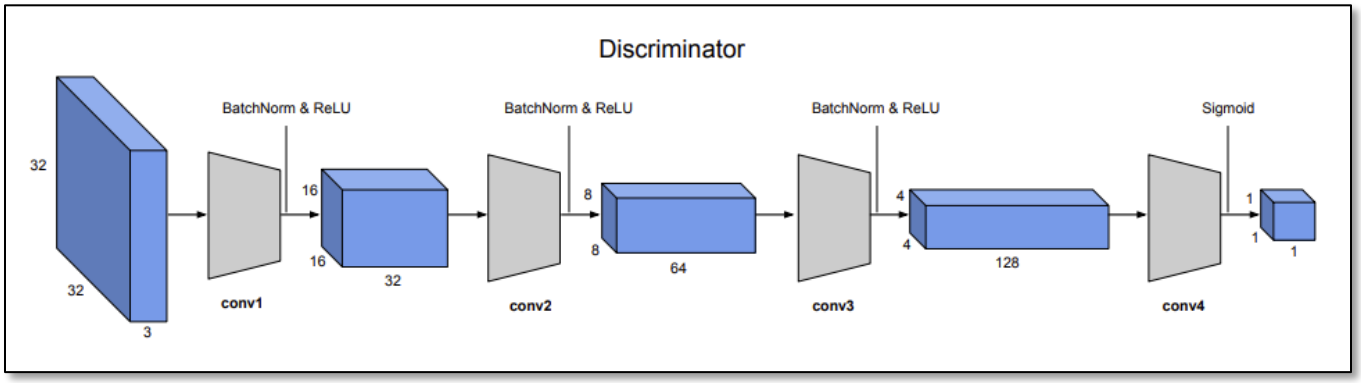
CycleGAN Architecture:

CycleGAN (**Cycle-Consistent Generative Adversarial Network**) is an unsupervised learning model used for **image-to-image translation** between two domains without requiring paired (exact image in two styles) training data. The goal of the model is to generate realistic images in both domains while ensuring that important content & structure are preserved during translation. The architecture consists of two components:

- **Generators**:
  - $\mathbf{G_{A \to B}}$: Translates images from domain A (photo) to domain B (painting).
  - $\mathbf{G_{B \to A}}$: Translates images from domain B to domain A.
  - The generators are based on a **ResNet** architecture with 6 or 9 residual blocks. Each generator includes down-sampling layers followed by residual blocks, which help retain important features during transformation. Finally, up-sampling layers are used to reconstruct the image in the target domain.[1]



- **Discriminators**:
  - $\mathbf{D_A}$: Distinguishes between real images in domain A & generated images that have been translated back to domain A.
  - $\mathbf{D_B}$: Distinguishes between real images in domain B & generated images that have been translated back to domain B.
  - The discriminators use a **PatchGAN** approach, which classifies overlapping patches of an image rather than the entire image at once. This helps capture fine details & improves stability during training.[1]



- **Implementation Details:**
  - **Generator Functions**: The code includes function build_generator() for constructing generators – one for $\mathbf{G}_{A\rightarrow B}$ & one for $\mathbf{G}_{B\rightarrow A}$. These generators perform down-sampling, residual block transformations, & up-sampling to produce translated images.[1]
  - **Patch Discriminator**: A patch-based discriminator is used for better detail recognition in distinguishing real vs. generated images. The function build_discriminator() handles this operation.[1]
  - **Optimizer**: The model uses the **Adam optimizer** with a learning rate of $2e^{-4}$ & momentum parameter ($\beta1=0.5$) to stabilize training.[5]
- **Hyperparameters:**
  - Batch size: 1
  - Number of training epochs: 10 per dataset
  - Generator:
    - Number of filters: 64, 128, & finally 256.
    - Generator kernel size:
      - First convolution: 7x7
      - Subsequent convolutions: 3x3
  - Discriminator:
    - Number of filters: 64, 128, 256 & finally to 512.
    - Discriminator kernel size: 4x4

- **Cycle Consistency Loss:**
  - One of the key innovations in CycleGAN is the cycle consistency loss, which ensures that if image is translated from one domain to another & back again, it should return to its original form. This constraint helps preserve content of the original image during translation. The cycle consistency loss is defined as:

    $L_{cyc} = E_{x\sim pdata(x)} [\|G_{B\rightarrow A}(G_{A\rightarrow B}(x)) - x\|_1] + E_{y\sim pdata(y)} [\|G_{A\rightarrow B}(G_{B\rightarrow A}(y)) - y\|_1]$

- **Adversarial Loss:**
  - CycleGAN also employs an adversarial loss, which trains each generator to produce images that are indistinguishable from real images in the target domain. The adversarial loss for generator $\mathbf{G}_{A\rightarrow B}$ & discriminator $\mathbf{D}_B$ is given by:

    $L_{GAN}(G_{B\rightarrow A}, D_B, A, B) = E_{y\sim pdata(y)} [\log D_B(y)] + E_{x\sim pdata(x)} [\log (1 - D_B(G_{A\rightarrow B}(x)))]$

The total objective combines both the adversarial losses & the cycle consistency loss, balancing between generating realistic images & preserving content.

Instructions for Building & Running the Code:

1. Prerequisites:
   - Install required libraries such as TensorFlow, NumPy, PIL, etc.
   - Download the dataset from CycleGAN Datasets & place it in the designated folder.
2. Running the Code:
   - To install dependencies, run:

     ```
     pip3 install -r requirements.txt
     ```

   - Prepare the datasets, as a directory. Refer below section - *Dataset*.
   - To train run the training cell in the notebook.

Troubleshooting & Challenges:

1. Mode Collapse: During training, generators may start producing repetitive outputs (mode collapse). Adjusting learning rates or applying different weight initializations can help mitigate this issue.
2. Cycle Consistency Loss Balance: Tuning the weight of cycle consistency loss relative to adversarial loss is important for improving image quality without overfitting.

Visual Representation of CycleGAN Architecture:

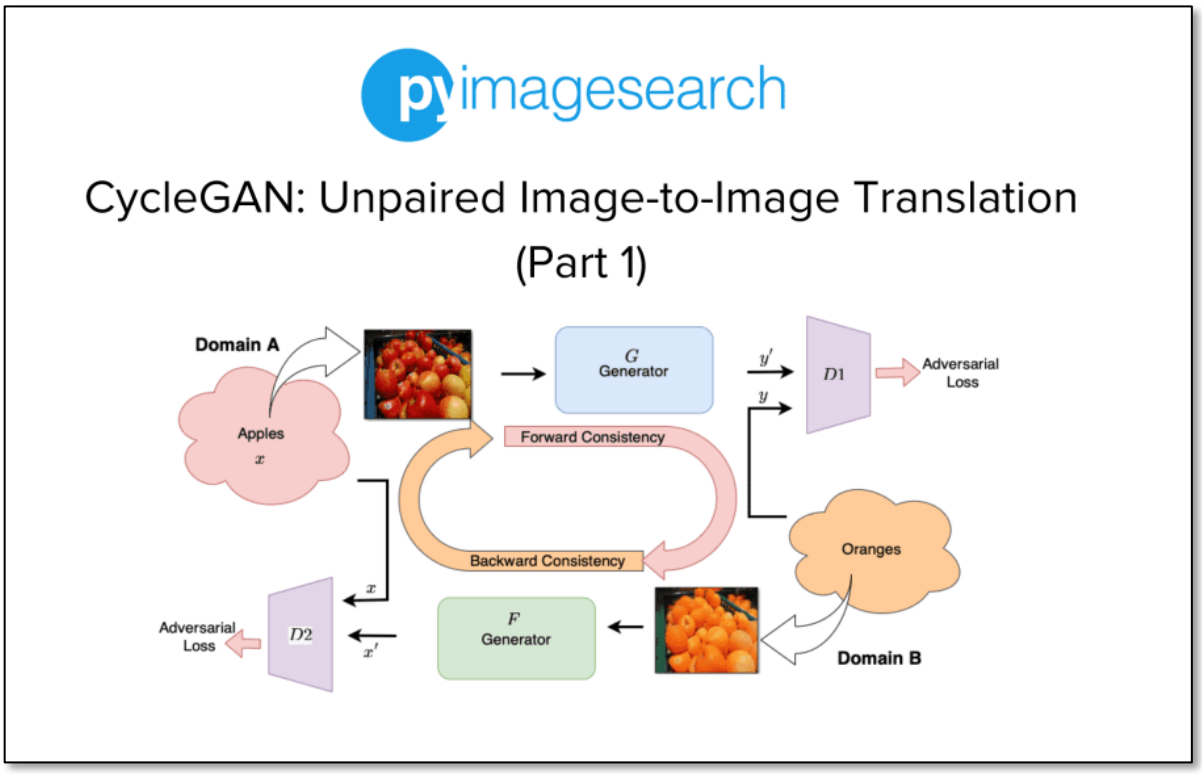


Fig 4: CycleGAN Architecture Representation (reference)

DATASET

Overview:

The CycleGAN model is trained & tested on unpaired datasets, where images from two distinct domains are available, but without corresponding pairs. The datasets used include the following subsets. It is available here.

- horse2zebra
- summer2winter_yosemite
- monet2photo
- vangogh2photo
- facades
- apple2orange

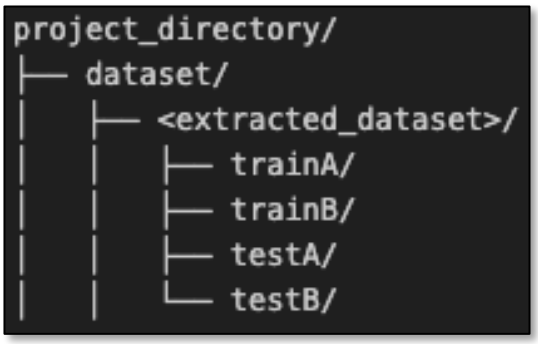


Fig 5: Directory structure for dataset.

Preprocessing:

1. **Image Resizing & Normalization**:
   - Each image is resized to [256 x 256] pixels to standardize input sizes for the network. This resolution allows the model to capture fine details without being computationally prohibitive.
   - The pixel values are normalized to the range [-1, 1], which is a preprocessing step necessary in GAN training. This normalization helps improve training stability & convergence.[4]
2. **Augmentation Techniques**:
   - **Random Cropping**: During training, images are resized to [286 x 286] pixels & then randomly cropped back to [256 x 256] pixels with all 3 channels. This slight resizing & cropping provide a form of data augmentation, helping the model generalize better by preventing overfitting to specific pixel arrangements.[4]
   - **Horizontal Flipping**: Random horizontal flips are applied to each image with a 50% probability. This augmentation further diversifies the training data, allowing the model to handle mirrored versions of content effectively.
3. **Data Pipeline**:
   - The datasets are loaded & processed using TensorFlow's data API (`tf.data.Dataset`), with the images fed into the model in batches of size 1 (consistent with the original CycleGAN setup). The preprocessed images are stored in a pipeline with prefetching & caching enabled to improve loading efficiency.

Challenges Posed by the Dataset:

- **Domain Shift**: Each dataset involves a significant domain shift, such as transforming natural images into artistic styles or changing seasons. The model must learn style changes while preserving structural content (e.g., shape of objects).
- **Style Variability**: Artistic datasets like monet2photo & vangogh2photo include varying brushstroke styles & color schemes, making it challenging for the model to capture fine details & replicate the artist's distinct style.

Dataset Distribution and Unpaired Nature:

Since CycleGAN does not rely on paired images, it learns to map general features and textures from one domain to another rather than learning one-to-one mappings. This unpaired setup poses a challenge because the model must learn a consistent transformation without direct pixel-wise supervision.

For example, when transforming horses into zebras using the horse2zebra dataset, CycleGAN must learn how to apply zebra-like textures while preserving the overall structure of the horse without ever seeing a paired horse-zebra image.

Visual Representation:

Below is an example of how CycleGAN learns unpaired image translation between summer and winter styles:

Fig 6: Dataset example – summer / winter

RESULTS & EVALUATION

After training the CycleGAN model on various datasets (e.g. summer2winter_yosemite, monet2photo, and vangogh2photo), we evaluated the model's performance in translating images to & from domains. The results, however, indicate some challenges in applying the expected transformations, especially for more complex styles.

Qualitative Results:

1. **summer2winter_yosemite**:
    - **Input (Summer Scene)**: Landscape images with vibrant summer colors.
    - **Output (Translated to Winter)**: Some images displayed winter-like color schemes with cooler tones, but other images showed minimal change, and transformations lacked consistent seasonal elements, such as snow or leafless trees. The results indicate a challenge in learning seasonal transformations across varied landscape images.
2. **monet2photo**:
    - **Input (Photo)**: Real-world photos.
    - **Output (Monet Style)**: While a few images displayed characteristics of Monet's impressionistic style, such as muted colors and softer textures, most of the translated images did not convincingly replicate the Monet aesthetic. The model occasionally failed to fully capture the brushstroke and color details unique to Monet's work.

Parameter Settings for Results:

For all experiments, the following parameters were used:

Parameter	Value
Batch size	1
Generator filters	64, 128, 256
Discriminator filters	64, 128, 256, 512
Number of epochs	10 * 7
Cycle consistency loss weight	5
Identity loss weight	2.5
Learning rate	calculated dynamically by linear decay
Optimizer	Adam ($\beta1 = 0.5$)

Quantitative Results:

To further evaluate CycleGANs performance, we calculated the following metrics:

1. Cycle Consistency **(L1 Reconstruction Loss):** The L1 reconstruction loss measures the pixel-wise difference between the original and cycle-reconstructed images, providing insight into how well the model preserves content after a full cycle. A lower L1 loss indicates better cycle consistency.
2. As of today, the L1 loss achieved is approx. 0.176. Which indicates the generators are trained well & can generate back the input image with minimal losses.
3. We also utilize FID score to evaluate the results. But, as of now it is not integrated in the training step of the model, and the training time being huge, we will evaluate it later.

Sample Image Set: We generated image sequences for each dataset, showing **Original Images**, **Translated Images**, and **Cycle-Reconstructed Images**. While the cycle consistency loss preserved the overall structure in cycle-reconstructed images, the translated images often lacked the intended stylistic changes, particularly for complex styles like Monet and Van Gogh.

CONCLUSION

Summary of Findings:

In this project, we implemented and evaluated the **CycleGAN** model for **unpaired image-to-image translation** across various domains. The results demonstrated that CycleGAN is effective in performing image translation tasks without requiring paired data, making it a versatile tool for applications where paired datasets are either unavailable or difficult to obtain. The model performed well on natural image transformations, such as **monet2photo** & **summer2winter_yosemite**, successfully preserving structural integrity while applying stylistic changes. However, the model faced challenges when translating more complex artistic styles, such as in the **vangogh2photo** and datasets. While the model was able to capture some artistic features, it struggled with maintaining consistency in colors & overlapping styles, which are critical for artistic style transfer.

Strengths of the Model:

1. **Unpaired Learning**: One of the most significant strengths of CycleGAN is its ability to perform image-to-image translation without needing paired datasets. This opens a wide range of applications where paired data is scarce or unavailable.
2. **Cycle Consistency**: By enforcing cycle consistency, the model ensures that content & structure are preserved across domains. This is particularly valuable for tasks like seasonal changes in landscapes or object texture alterations, where maintaining the original shape of objects is crucial.

Limitations and Future Directions:

While CycleGAN has proven to be a powerful framework for unpaired image translation, several limitations were observed during this project:

1. **Style Fidelity**: The model struggled to fully capture high-level artistic features in abstract domains like Van Gogh's paintings. To address this, future work could explore the introduction of **perceptual loss functions** (implementations available) or style-specific layers that focus on capturing intricate details such as brushstrokes and color schemes.
2. **Handling Abstract Domains**: Artistic datasets like **vangogh2photo** require a more nuanced approach due to the abstract nature of Van Gogh's style. Incorporating **attention mechanisms** like the ones in LLMs or specialized architectures could improve performance in these highly variable domains.
3. **Dynamic Hyperparameter Tuning**: The current implementation used fixed hyperparameters throughout training. Future research could explore adaptive learning rates or weighted loss adjustments that dynamically balance content preservation with stylistic transformations, especially for more complex datasets.

REFERENCES

1. J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks,” *IEEE International Conference on Computer Vision (ICCV)*, pp. 2223-2232, 2017.
2. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative Adversarial Nets,” *Advances in Neural Information Processing Systems (NIPS)*, vol. 27, 2014.
3. M.-Y. Liu and O. Tuzel, “Coupled Generative Adversarial Networks,” *Advances in Neural Information Processing Systems (NIPS)*, pp. 469-477, 2016.
4. X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, “Multimodal Unsupervised Image-to-Image Translation,” *European Conference on Computer Vision (ECCV)*, pp. 172-189, 2018.
5. P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-Image Translation with Conditional Adversarial Networks,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1125-1134, 2017.
6. T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, “Learning to Discover Cross-Domain Relations with Generative Adversarial Networks,” *International Conference on Machine Learning (ICML)*, pp. 1857-1865, 2017.
7. A. Brock, J. Donahue, and K. Simonyan, “Large Scale GAN Training for High Fidelity Natural Image Synthesis,” *International Conference on Learning Representations (ICLR)*, 2019.
8. Scikit Learn (2024), “train_test_split,” Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.
9. T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, “High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8798-8807, 2018.
10. M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein GAN,” *arXiv preprint arXiv:1701.07875*, 2017.
11. H. Bansal and A. Rathore, “Understanding and Implementing CycleGAN in TensorFlow,” *Towards Data Science*, 2019. Available: https://towardsdatascience.com/understanding-and-implementing-cyclegan-in-tensorflow-7d7f7f87c83a.
12. Y. Taigman, A. Polyak, and L. Wolf, “Unsupervised Cross-Domain Image Generation,” *International Conference on Learning Representations (ICLR)*, 2017.