Advanced Image Generation and Translation Techniques: A Comparative Study

A

report submitted in fulfillment for the award of the degree of

Bachelors

in

Computer Science

By

Nalam Anunay(2019-BCS-034)

Under the Supervision of

Dr. W Wilfred Godfrey

Department of Computer Science

ABV-INDIAN INSTITUTE OF INFORMATION TECHNOLOGY AND MANAGEMENT GWALIOR

GWALIOR, INDIA - 474015

2023

DECLARATION

I hereby certify that the work, which is being presented in the report, entitled **Advanced Image Generation and Translation Techniques: A Comparative Study**, in fulfillment of the requirement for the award of the Degree of **Bachelor of Technology** and submitted to the institution is an authentic record of our own work carried out during the period *January 2023* to *May 2023* under the supervision of **Dr. W Wilfred Godfrey** . We also cited the reference about the text(s)/figure(s)/table(s) from where they have been taken.

Dated: **Signature of the Candidate**

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dated: **Signature of the Supervisor**

Acknowledgements

I am highly indebted to **Dr. W Wilfred Godfrey** , and am obliged for giving me the autonomy of functioning and experimenting with ideas. I would like to take this opportunity to express my profound gratitude to her not only for her academic guidance but also for her personal interest in my project and constant support coupled with confidence boosting and motivating sessions which proved very fruitful and were instrumental in infusing self-assurance and trust within me. The nurturing and blossoming of the present work is mainly due to her valuable guidance, suggestions, astute judgment, constructive criticism and an eye for perfection. My mentor always answered myriad of my doubts with smiling graciousness and prodigious patience, never letting me feel that I am a novice by always lending an ear to my views, appreciating and improving them and by giving me a free hand in my project. It's only because of her overwhelming interest and helpful attitude, the present work has attained the stage it has. Finally, I am grateful to our Institution and colleagues whose constant encouragement served to renew our spirit, refocus our attention and energy and helped me in carrying out this work.

Nalam Anunay

Abstract

Domain adaptation is a crucial component of transfer learning, which involves transferring knowledge from a source domain to a target domain. The process of domain adaptation involves adjusting the model parameters of the source domain to improve the performance of the model on the target domain. This adaptation is necessary because the distributions of the data in the source and target domains are usually different, which can lead to poor model performance on the target domain.

We will explore various methods, including domain adaptation using domain-specific features. We also highlight the importance of selecting appropriate evaluation metrics and datasets to accurately assess the effectiveness of domain adaptation techniques. Finally, we discuss future directions in the field of domain adaptation in transfer learning

Keywords: *Domain Adaptation, transfer learning, Generator, Discriminator, CycleGan, VQGan, Clip architecture*

Contents

List of Figures

List of Tables

List of Acronyms

DA	Domain Adaption
SD	Source Domain
ID	Intermidiate domain
TD	Targen Domain
G and F	Generators
Dx and Dy	Discriminators
F(G(x))	Forward cycle consistency loss
G(F(y))	Backward cycle consistency loss
F(x)-x and G(y)-y	Identity loss
c7s1-k	Convolution-InstanceNormReLU layer with k filters and stride 1
dk	Convolution-InstanceNorm-ReLU layer with k filters and stride 2
Rk	Residual block that contains two 3×3 convolutional layers
uk	Fractional-strided-ConvolutionInstanceNorm-ReLU layer with k filters

1

Introduction & Literature Review

This chapter includes the details of Domain adaptation, Transfer Learning, Problem Statement, and my objectives for this project.

1.1 INTRODUCTION

Machine learning models often face a common challenge in various real-world applications: the distribution of data encountered during testing or deployment may differ significantly from the distribution of data used for training. This discrepancy, known as domain shift or distribution shift, can lead to a drop in model performance and limit the generalization ability of the learned models. Domain adaptation, a subfield of machine learning, aims to address this issue by enabling models to adapt and generalize well to new, unseen domains.

This paper explained Domain Adaptation through two experiments: image generation using text prompts with annotated images as an intermediate domain and unpaired image-to-image translation using Cycle GAN, where the source and target domains were distinct.

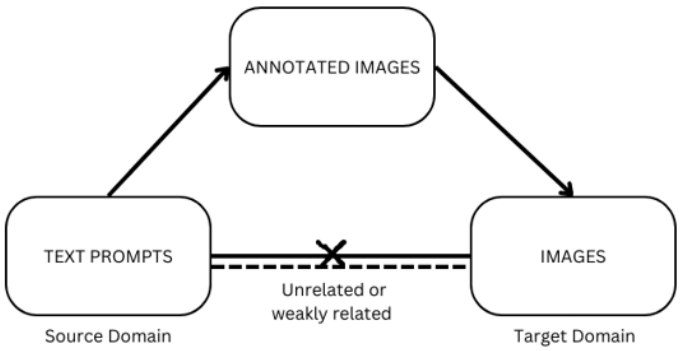

Figure 1.1: Domain Adaptation from text to image

Above figure explains how domain adaptation is done from text prompts to images using annotated images as intermediate domain it was done using clip architecture and Vqgan architecture

Domain adaptation is the process of adapting a machine learning model trained on one domain to another domain where the training data may be limited or unavailable. In the

context of text-to-image synthesis, domain adaptation is crucial when the target domain lacks sufficient annotated image data for training. In such scenarios, it is challenging to learn a generative model that can produce high-quality images that match the style and content of the target domain. To overcome this challenge, one approach is to use an intermediate domain that acts as a bridge between the source and target domains. Annotated images from the intermediate domain can be used to fine-tune the generative model and improve its ability to generate images that resemble the target domain. In this paper, we explore the use of annotated images as an intermediate domain for domain adaptation in text-to-image synthesis and evaluate its effectiveness using state-of-the-art models such as CLIP and VQ-GAN.

In the second experiment, unpaired image-to-image translation using Cycle GAN, the objective was to transform images from the summer domain into winter images and vice versa. By leveraging the power of Cycle GAN, the model learned the mapping between the summer and winter domains without relying on paired examples. The purpose was to generate realistic and visually convincing winter images from summer images and similarly, transform winter images to resemble summer scenes. This approach allowed for domain adaptation by bridging the gap between the two distinct domains, enabling the model to effectively translate images and adapt to the characteristics and style of each domain.

Unpaired image-to-image translation is the task of translating an image from one domain to another without a paired dataset that contains corresponding images. For instance, converting a photo of a sunny beach into a painting of a winter landscape. In recent years, Generative Adversarial Networks (GANs) have shown impressive results in image-to-image translation tasks. However, training GANs requires a large amount of paired data, which is often difficult to obtain.

CycleGAN is a popular framework for unpaired image-to-image translation that uses a cycle consistency loss to enforce the translated image to retain its original content. The key idea is to learn two mappings: one from domain A to domain B and the other

from domain B to domain A, without any paired training data. CycleGAN leverages the idea that if we translate an image from domain A to B and then back to A, it should retain its original characteristics. By enforcing cycle consistency, CycleGAN encourages the translated images to be more realistic and consistent with their original content.

The CycleGAN model consists of two GANs, each with a generator and a discriminator. The generator in the first GAN maps images from domain A to domain B, and the discriminator learns to distinguish between the generated images and real images from domain B. Similarly, the generator in the second GAN maps images from domain B to domain A, and the discriminator distinguishes between generated images and real images from domain A. The cycle consistency loss is used to ensure that the output of the first generator, when translated back to domain A, is similar to the input image, and vice versa for the second generator.

CycleGAN has shown impressive results in various image-to-image translation tasks, including style transfer, object transfiguration, and even domain adaptation. Its ability to perform unpaired image-to-image translation without requiring paired data makes it a promising approach for many practical applications in computer vision and image processing.

If source and target domains have a large distribution gap, then transferring knowledge between them may cause a substantial performance loss in the target domain. Domain Adaptation helps by finding intermediate domains and transferring knowledge from source to target domains using those intermediate domains. Like in text-to-image, where both source and target domains are totally different. Both are completely heterogeneous domains, but knowledge was transferred using annotated images as intermediate domain

Furthermore, obtaining labeled data in the target domain was not always possible. It may be costly to obtain, or even infeasible, which makes it impractical to rely solely on labeled target data for training. Therefore, domain adaptation techniques must leverage the available labeled data from the source domain and adapt it to the target domain,

where labeled data may be scarce or unavailable. Like in the unpaired image-to-image translation experiment where images are not labeled or paired.

There were many papers about image-to-image translation, but most of them used paired image data. Their models are trained on the original image and the corresponding acquired image after translation. Still, it is hard to create such datasets, and existing datasets are usually too small.

1.2 Motivation/Problem

When there is a large distribution gap between the source and target domains, transferring knowledge between them may cause a significant performance loss in the target domain.Domain adaptation finds intermediate domains to transfer knowledge between source and target domains

There are many research papers based on image-to-image translation, but most of them used paired image data.However, creating such datasets can be difficult, and existing datasets are usually small.

1.3 Literature Review

[1] The paper proposes a new method for training deep neural networks with limited labeled data using self-supervised learning. The proposed method, called Clip, is based on a contrastive learning framework that learns representations by maximizing the agreement between different views of the same data. The authors show that clip outperforms existing self-supervised and supervised learning methods on several image classification benchmarks, even when the amount of labeled data is small.

[2] The paper proposes a novel method for text-to-image generation using a combination of neural style transfer and attention-based generative adversarial networks (GANs).

The proposed method uses a pre-trained style transfer model to extract style features from an input text description, and then uses these features to condition a GAN-based image generator. The authors show that their approach outperforms existing methods for text-to-image generation on several metrics and also conduct user studies to demonstrate the quality of the generated images.

[3] The paper introduces a framework for transitive transfer learning and proposes various approaches to decrease the domain distance between two unrelated domains. It also explains different approaches to select an efficient intermediate domain.

[4]The paper presents a new approach to image-to-image translation called CycleGAN, which uses a cycle-consistent adversarial network to learn mappings between two image domains without requiring paired training data. The proposed method involves training two generative adversarial networks (GANs) in opposite directions, with each GAN learning to map images from one domain to the other.

[5] This paper introduces a new strategy used to stabilize the CycleGAN Training called Replay buffer.it is used to train the discriminator. Generated images are added to the replay buffer and sampled from it. The replay buffer returns the newly added image with a probability of 0.5. Otherwise, it sends an older generated image and replaces the older one with the newly generated one. This is done to reduce model oscillation.

[6] The paper "Transfer Learning with Dynamic Adversarial Adaptation Network" proposes a novel approach to transfer learning that utilizes a dynamic adversarial adaptation network (DAAN) to adapt to the target domain by aligning the source and target feature distributions in a joint feature space. The proposed approach aims to improve the performance of transfer learning across different domains where the source and target domains have different data distributions.

[7]The paper "Cycle-consistent Conditional Adversarial Transfer Networks" proposes a novel approach to transfer learning that utilizes a cycle-consistent conditional adversarial transfer network (C2ATN) to learn mappings between source and target domains while preserving the underlying semantic information.

[8]The paper "Cycle-consistent Conditional Adversarial Transfer Networks" proposes a novel approach to transfer learning that utilizes a cycle-consistent conditional adversarial transfer network (C2ATN) to learn mappings between source and target domains while preserving the underlying semantic information.

[9]The paper "Generating Images from Caption and Vice Versa via CLIP-Guided Generative Latent Space Search" proposes a novel approach to generating images from captions and vice versa by utilizing a CLIP-guided generative latent space search.

The approach consists of two modules: a generative model and a CLIP (Contrastive Language-Image Pre-Training) model. The generative model is used to generate images from captions and vice versa by performing a search in the latent space. The CLIP model is used to guide the search by providing a similarity score between the generated image and the input caption or vice versa.

[10]The IEEE paper "A Survey on Transfer Learning" provides a comprehensive overview of transfer learning, which is a technique that enables knowledge transfer across different domains or tasks. The paper begins by introducing the basic concepts of transfer learning and its applications in various fields such as computer vision, natural language processing, and speech recognition.

1.4 Objectives

The primary objectives of this project are:

- The primary objective of domain adaptation is to develop effective techniques that can adapt pre-trained models to new domains with different characteristics. These techniques can handle the differences between the source and target domains and improve the performance of the models on the target domain.

- Objective of the first experiment is that the encoding of the text and the encoding of the images should get as close as possible, and their similarities are as high as possible.

- Objective of the unpaired image-to-image translation experiment is to learn a mapping G: X $\rightarrow$ Y, such that the distribution of images from G(X) is indistinguishable from the distribution Y using an adversarial loss. Because this mapping is highly under-constrained, we couple it with an inverse mapping F: Y $\rightarrow$ X and introduce a cycle consistency loss to push F(G(X)) is similar to X.

2

Design Details and Implementation

2.1 Generating image using text

2.1.1 Architecture Details:

This experiment is a combination of two models, clip and vq gan:

2.1.1.1 Clip architecture

Clip (Contrastive Language-Image Pre-Training) architecture is a generative architecture developed by OpenAI created by training a lot of images and a lot of text prompts, it has the capability of encoding text and it is able to predict what text corresponds to an image and vice versa. It creates the capability to encode and quote, both text and image elements into a sort of common space where the network learns to connect both kinds of encoding

The CLIP architecture consists of a transformer encoder that processes the input text and a convolutional neural network (CNN) that processes the input image. The transformer encoder is responsible for encoding the input text into a fixed-length vector representation, while the CNN extracts feature from the input image. The output of the transformer encoder and CNN are then fed through a multilayer perceptron (MLP) that maps the joint representation of the input text and image to a common embedding space.

The training process of CLIP involves using a contrastive loss function, which encourages the model to learn a joint representation space that maps semantically similar text and images to nearby points in the embedding space. Specifically, during training, CLIP is fed pairs of text and images, and the model is trained to maximize the similarity between the embeddings of the corresponding text and image pairs while minimizing the similarity between the embeddings of the non-corresponding pairs. This encourages the model to learn a joint representation space that captures the underlying semantic relationships between the input text and images.

CLIP has achieved state-of-the-art performance on several benchmark image recognition and natural language understanding tasks, including image classification on the

ImageNet dataset and visual question answering on the VQA dataset. The success of CLIP has demonstrated the effectiveness of using a contrastive learning approach for pre-training neural networks and has opened up new avenues for research in the field of multimodal learning.

2.1.1.2 VQGAN architecture:

VQGAN is a variant of GAN architecture that uses vector quantization to improve the quality and diversity of generated images.VQGAN uses a codebook to represent the input data as discrete codes, which are then used to generate new images. and it works with patches. One of the major issues with transformer networks applied to images is images have a very large number of pixels, and this complicates a lot of the creation of long sequences in transformers but we can solve this by working in patches,so we reduce the dimensionality problem in this way. Instead of working with Pixels directly,they train the network to basically learn a sort of matrix, which again learns kinds of representations of parts of the image that are stored in the codebook. It has different parts convolution, decoder, etc

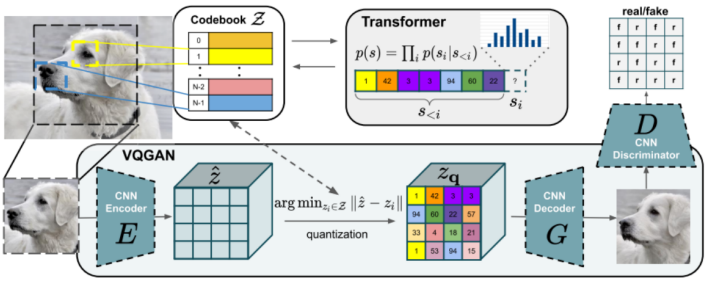

Figure 2.1: convolutional VQGAN[5]

2.1.2 Implementation:

In this experiment, I have implemented a multimodal architecture which is a combination of both clip architecture and vqgan architecture

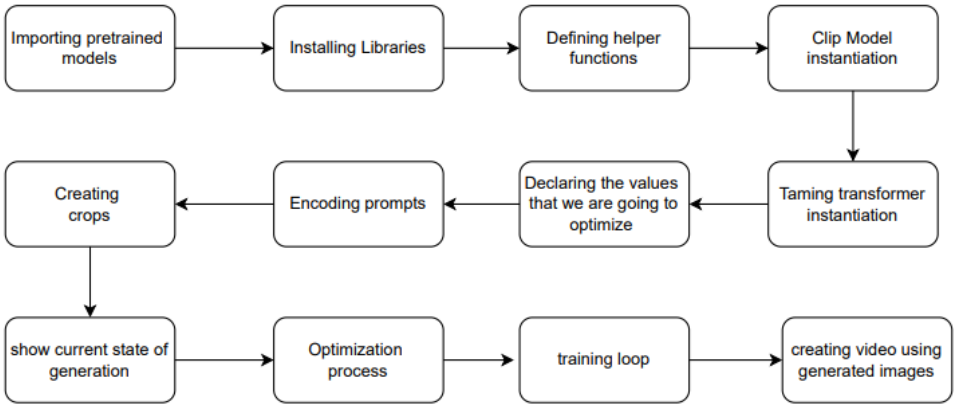

Figure 2.2: work flow

2.1.2.1 Libraries Imported:

- numpy: to efficiently perform numerical computations and manipulate multi-dimensional arrays of data.
- torch: to provide efficient tensor operations and a high-level framework for training
- imageio:to read, write, and manipulate image data, facilitating the preprocessing and analysis of images in various tasks.
- math:to perform mathematical operations and calculations required for various algorithms and computations.
- torchvision: providing convenient access to popular datasets, pre-trained models, and image transformations.
- PIL: image processing and manipulation tasks, such as resizing, cropping, and converting image formats.

- omegaconf: to provide a flexible and hierarchical configuration management system for managing experiment settings and hyperparameters.
- yaml: for convenient configuration and storage of model parameters and settings in a human-readable format.

2.1.2.2 Helper Functions:

showFromTensor:As we train our system, the values are stored in a tensor format, which represents a multi-dimensional array. Since tensors contain information in the range of 0 to 1, we multiply them by 255. The purpose of this multiplication is to scale the tensor values to a range of 0 to 255, which is suitable for displaying the tensor as an image. Therefore, the function is designed to take a tensor as input and display it as an image. showme: This function is used to display the generated images at the end of the training phase. Here, we pass the parameters through the generator, normalize the images, and then display them.

2.1.2.3 Hyper Parameters:

learning_rate = 0.5

batch_size = 1

wd = 0.1

noise_factor = 0.22

total_iter = 100

im_shape = [450, 450, 3] # height, width, channel

size1, size2, channels = im_shape

2.1.2.4 Encoding Text Prompt:

To encode the text prompt, we first normalize it by inserting the values of mean and variance. Then, we create a function called 'encodetext' that receives the text and tokenizes

it. It takes each part of the text prompt and represents it with tokens. We create encodings for all the included, excluded, and extra prompts. When generating an image, we create crops of the image and send a set of these crops, as the more variations we send, the better the model will understand. We also perform transformations and augmentations on the image, such as sequential transformations

2.1.2.5 Creating crops:

To create the crops, we add some extra padding around the image to accommodate for any transformations or rotations. This allows us to offset the position and take a crop of the image.

2.1.2.6 Optimization Process:

In the optimization process, we gradually change the value of the parameters so that the image gradually moves in the direction of matching the content of the text prompt. We design a loss function called 'optimizeResult,' which sets the values of the optimizer using backpropagation. With the alpha and beta variables, we can regulate how important the exclude encodings are in comparison with the include encodings. Then, we create image encodings by calling the CLIP model. In this process, w1 and w2 are the weights of the include encodings vs. the extras, which we declare while performing the experiment. We calculate the main loss using the cosine similarity of the text encoding with the image encoding, and we calculate the penalized loss for the exclude encodings. The total loss is calculated using alpha and beta.

2.1.2.7 Model training:

Here, we create two lists, one for resolved images and another for resolved latent space, which are going to accumulate. We save them for different images that we generate and also for different sets of parameters as we optimize them, so that we can use them later for interpolation.

2.1.2.8 Creating a video from the generated images:

To create a video we take the input list of latent parameters and duration list. we have to create a data structure holding the generated images taking list of latent values and distributions altogether and enumerating them to loop each of latent parameters and each of durations this will make the images interpolate from first to second, second to third and third to first

2.1.2.9 Summary :

First, a fixed text input, such as "a man on a horse," is chosen, and an image is initialized with random noise. An optimization process is then performed over 100 iterations. In each iteration, the text prompt is encoded using the CLIP architecture to obtain an understanding of the text, and the image is transformed using augmentation and rotation techniques. Crops of the image are then created to help the CLIP architecture understand the image better. Thirty different versions of the image are sent to CLIP, which encodes them and returns their encodings. Finally, the cosine similarity function is used to compare the text and image encodings, which calculates the loss value or performance of the network.

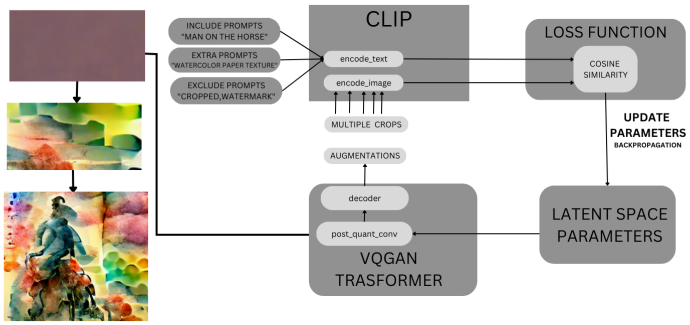

Figure 2.3: Multimodal architecture

2.2 Unpaired Image-to-Image Translation

2.2.1 Architecture Details:

2.2.1.1 CycleGAN architecture:

CycleGAN is a variant of GAN architecture that is used for image-to-image translation tasks. Unlike traditional GANs, which require paired training data (i.e., input and output pairs), CycleGAN can learn to translate between two domains without any paired data. This is achieved by introducing a cycle consistency loss, which ensures that the translated images are consistent with the input images.

In this CycleGAN experiment, the aim is to learn a mapping between two datasets using two generators, G and F, along with two discriminators, Dx and Dy. The G generator attempts to transform images from dataset X into images that resemble dataset Y, while the F generator aims to do the opposite. The Dx discriminator distinguishes between real images from dataset X and fake images created by the F generator, while the Dy discriminator distinguishes between real images from dataset Y and fake images created by the G generator. By training the generators and discriminators together, the CycleGAN model can learn to effectively translate images between the two datasets.

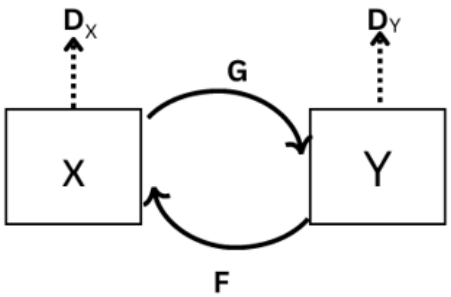

Figure 2.4: CycleGAN

2.2.2 The generator:

The generator consists encoder and decoder. It downsamples or encode the input image, then interprets the encoding with Residual Blocks having skip connections. After that with a series of layers, it upsamples or decodes the representation to the size of the fake image.

Reflection padding "reflects" the row into the padding. It is used mostly for reducing artifacts.

Batch norm normalizes across the mini-batch of definite size.On the other hand, Instance normalization normalizes across each channel in each data instead of normalizing across input features in a data. Instance Norm normalizes each batch independently and across spatial locations only.

Use of instance normalization layers, the normalization process allows to remove of instance-specific contrast information from the image content, which simplifies image generation. This results in vastly improved images.

As you can see below, the representation size shrinks in the encoder phase, stays constant in the transformer phase, and expands again in the decoder phase.

The generator with 9 residual blocks consists of: c7s1-64,d128,d256,R256,R256,R256,

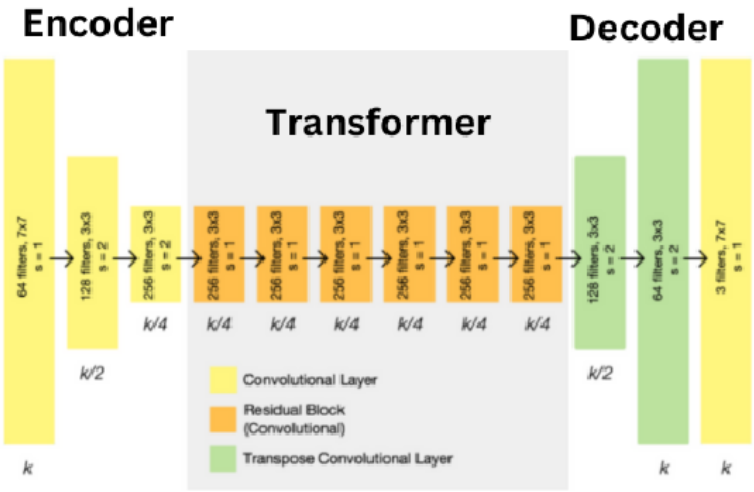

Figure 2.5: Generator architecture[2]

R256,R256,R256,R256,R256,R256, u128, u64,c7s1-3.Here c7s1-k denotes a 7×7 Convolution-InstanceNormReLU layer with k filters and stride 1. dk denotes a 3 × 3 Convolution-InstanceNorm-ReLU layer with k filters and stride 2. Reflection padding was used to reduce artifacts. Rk denotes a residual block that contains two 3 × 3 convolutional layers with the same number of filters on both layers. uk denotes a 3 × 3 fractional-strided-ConvolutionInstanceNorm-ReLU layer with k filters.

2.2.2.1 Discriminator:

Discriminator network is responsible for distinguishing between real and fake samples. The architecture of the discriminator was C64-C128-C256-C512 shown below where Ck denotes a 4 × 4 Convolution-InstanceNorm-LeakyReLU layer with k filters and stride 2. After the last layer, we apply convolution to produce a 1-dimensional output. We do not use InstanceNorm for the first C64 layer. We use leaky ReLUs with a slope of 0.2.
the model also has a final hidden layer C512 with a 1×1 stride. Model is mostly used with 256×256 sized images as input, the size of the output feature map of activations is 16×16. If 128×128 images were used as input, then the size of the output feature map of activations would be 8×8.

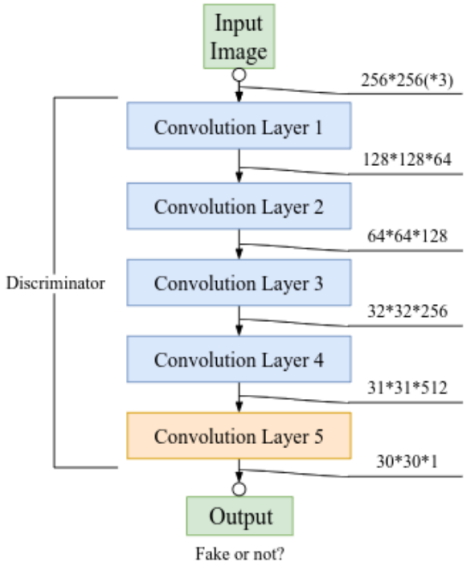

Figure 2.6: Simplified Discriminator architecture

2.2.2.2 Patch Gan

PatchGAN is a variant of Generative Adversarial Networks (GANs) that is commonly used in image-to-image translation tasks, such as image segmentation and image style transfer. Unlike traditional GANs, which generate an entire image at once, PatchGAN generates an image by generating smaller patches of the image and then stitching them together. This approach allows PatchGAN to capture local image details and produce higher-quality images. The main difference between a PatchGAN and a regular GAN discriminator is that - the regular GAN maps an input image to a single scalar output in the range of [0,1], indicating the probability of the image being real or fake, while PatchGAN provides Matrix as the output with each entry signifying whether its corresponding patch is real or fake.

In PatchGAN, the discriminator network is trained to classify whether each patch of the generated image is real or fake, instead of classifying the entire image at once. The discriminator network is typically a convolutional neural network (CNN) that takes in a patch of the image and outputs a probability indicating whether the patch is real or fake. The generator network generates the image by generating patches of the image and combining them to form the final image. The loss function used in PatchGAN is similar to that of traditional GANs, where the generator tries to generate images that can fool the discriminator and the discriminator tries to distinguish between real and fake patches.

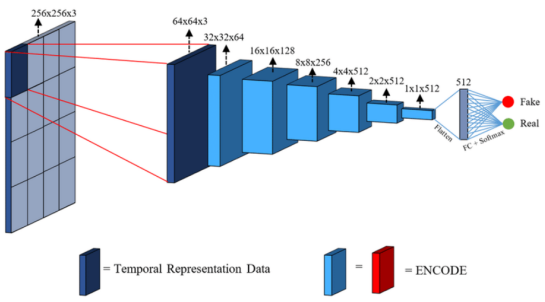

Figure 2.7: PatchGAN structure in the discriminator architecture[6]

2.2.3 GAN Loss:

GAN Loss or Generative Adversarial Network loss is the loss function used in GAN models to train the generator and discriminator networks. In GAN, the generator network generates fake data while the discriminator network distinguishes between the fake and real data. The GAN loss is calculated by minimizing the cross-entropy loss between the real and fake data distribution.

2.2.4 Cycle Consistency Loss:

Cycle consistency loss is a loss function used in CycleGAN models that helps to enforce the consistency between the input and output images. The cycle consistency loss ensures that the reconstructed image after going through the generator network and then the inverse generator network remains similar to the input image. The loss is calculated by comparing the input image with the reconstructed image using a distance metric such as mean squared error.

Cycle loss is added to ensure an image a generator outputs can be mapped back into the original image by the other generator.$x \rightarrow G(x) \rightarrow F(G(x)) \approx x$ represents forward cycle consistency loss and $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$ represents backward cycle consistency loss

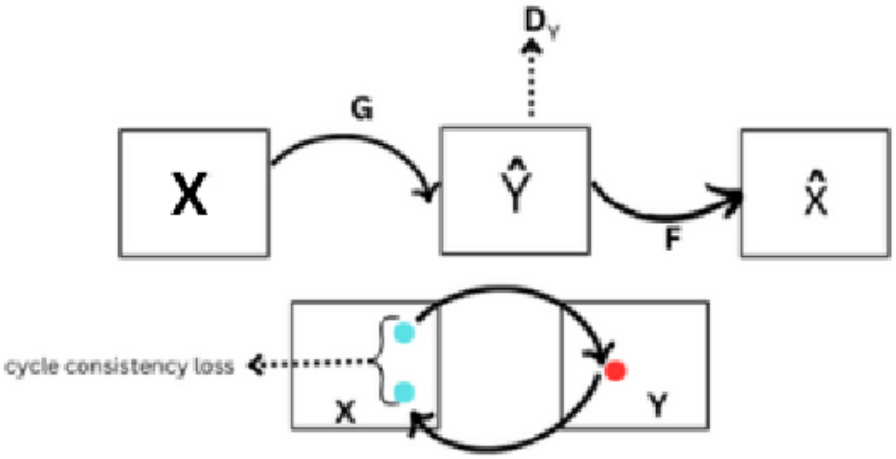

Figure 2.8: forward cycle consistency loss

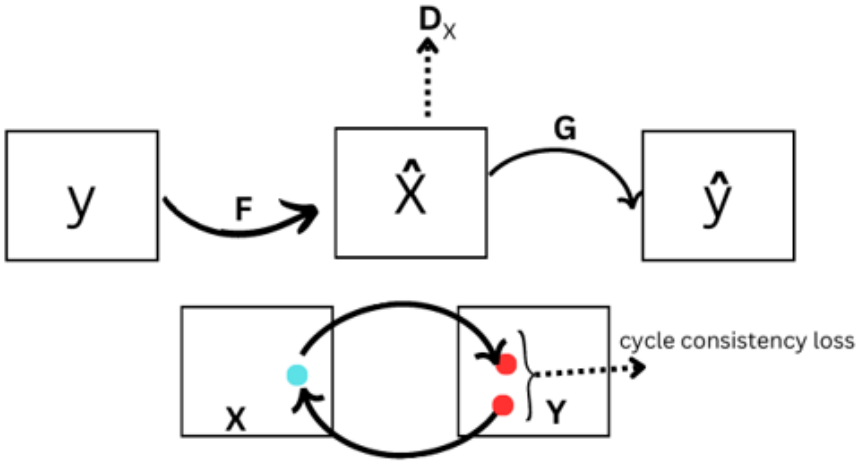

Figure 2.9: Backword cycle consistency loss

2.2.5 Identity Loss:

Identity loss is a loss function used in image-to-image translation models to maintain the content of the input image in the output image. The identity loss encourages the generator network to maintain the input image's content and only change the style or other specific attributes of the image. The identity loss is calculated by minimizing the mean squared error between the input and generated images.

Identity loss is added to help preserve tint. It states that when given an image of the target class a generator should return the same image i.e F(x)-x and G(y)-y

2.2.6 Adversarial Loss:

Adversarial loss is a loss function used in GAN models to train the generator network. The adversarial loss measures how well the generated data fools the discriminator network. The generator network is trained to minimize the adversarial loss, which means generating data that is more similar to the real data distribution. The adversarial loss is calculated by maximizing the log probability of the generated data being classified as real by the discriminator network.

2.2.7 Implementation:

Below figure explains work flow of experiment2:

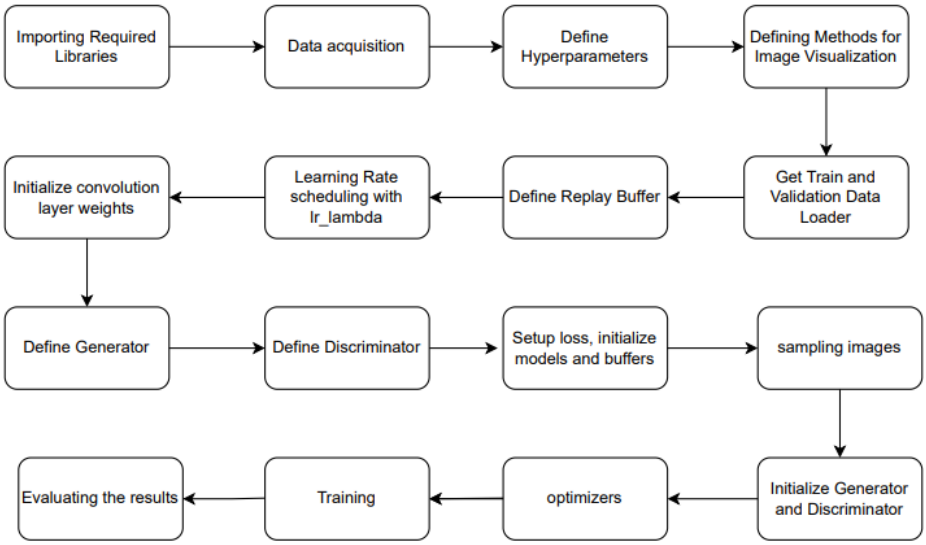

Figure 2.10: work flow

2.2.7.1 DataSet:

In this experiment, I have used the Summer2Winter Yosemite dataset. It consists of 1540 Summer Photos and 1200 Winter Photos, each split into train and test subsets.
At first I loaded the dataset into four folders test a, test b, train a, and train b. Here we are just replicating like in train mode there are 2 folders train a and train b.In test mode, it will be test A and test b. Then if they are not in RGB they will convert to RGB and get stored.

2.2.7.2 Libraries imported:

- numpy: To efficiently perform numerical computations and manipulate multi-dimensional arrays of data.
- torch: to provide efficient tensor operations and a high-level framework for training
- scipy: It is used in machine learning for its extensive collection of mathematical,

statistical, and signal processing functions and algorithms, which are essential for building and training machine learning models..

- math: To perform mathematical operations and calculations required for various algorithms and computations.
- torch-vision: providing convenient access to popular datasets, pre-trained models, and image transformations.
- PIL: image processing and manipulation tasks, such as resizing, cropping, and converting image formats.
- sys: To access system-specific parameters and functions, such as command-line arguments, system path variables, and memory allocation.
- datetime: To handle date and time data

2.2.7.3 Hyper parameters Used:

epoch=0

n_epochs=200

dataset_train_mode="train"

dataset_test_mode="test"

batch_size=4

lr=0.0002

decay_start_epoch=100

b1=0.5

b2=0.999

n_cpu=8

img_size=128

channels=3

n_critic=5

sample_interval=100

num_residual_blocks=19

lambda_cyc=10.0

lambda_id=5.0

2.2.8 Helper Functions:

2.2.8.1 Replay Buffer:

A replay buffer is often used to train the discriminator network. During the training process, the generator network generates images that are then evaluated by the discriminator network. It was used to reduce model oscillation; we updated the discriminator using a history of generated images rather than the ones produced by the latest generators. We keep an image buffer that stores the 50 previously created images.
This approach is used to reduce the oscillation of the discriminator network during training. Without the replay buffer, the discriminator may overfit to the most recently generated images, leading to oscillations in the generator network. By random sampling from the replay buffer, the discriminator is exposed to a wider range of generated images, which helps to reduce the oscillations and improve the overall stability of the GAN training process.

2.2.8.2 Learning Rate scheduling:

In Learning rate scheduling, there will be a function called lr lambda which computes a multiplicative factor given an integer parameter epoch, or a list of such functions, one for each group. The implementation in this experiment goes like this: In order to linearly decay the learning rate after 100 epochs, the lambda function checks whether the current epoch has exceeded the decay start epoch(which is 100). If the current epoch is less than decay start epoch(which is 100), it returns 1. So the initial lr remain the same for the first 100 epochs. If the current epoch has exceeded the decay start epoch(which is 100), the initial lr will be decreased throughout the rest of the epochs among total epochs(rest

100 epochs out of total 200 epochs of training).If we equally divide the lr for the last 100 epochs and keep subtracting from "Base-LR" or "Initial LR", it will reach to 0 by the end of the last 100 epochs. As lambda lr multiplies initial lr with given function, the epoch beyond the decay start epoch will sum up the consistent decrease in lr value from starting of decay epoch(which is 100) to the current epoch(for example 110).

2.2.9 Sampling images:

It is just creating a grid for sampling images.Like saving the image samples at a particular part and the part will be defined.then arrange the images along the x axis and y axis.

2.2.10 Training:

After defining the three loss functions, namely the normal GAN loss, cycle consistency loss, and identity loss, we initialize two generator models and two discriminator models. The generator models come from the ResNet class, so we pass the input shape and the number of residual blocks as parameters. For the discriminator models, we only need to pass the input shape as a parameter. We then initialize the weights and invoke the replay buffer created by the generator, which will hold 50 images and pass them to the discriminator through this buffer. Next, we define the optimizers for both generators, which are both Adam optimizers. The learning rate for the Adam optimizer is defined in the hyperparameters. We also define the learning rate update schedulers using the LambdaLR class, which provides the multiplicative factor that decays the learning rate. Finally, we set the model inputs to 1 for all real images and 0 for all fake images.

First, we get the identity loss by passing real A images to the generator, which will generate A-domain images. We repeat this step for real B images to generate B-domain images. The total identity loss is the average of both identity losses.

Next, we compute the GAN loss.

Then, we compute the cycle consistency loss by comparing the reconstructed A images with the real A images of domain A. The lambda for the cycle loss is set to 10.0 to penalize

and force the network to learn the translation. We repeat this step for the reconstructed B images and real B images of domain B.

Finally, we compute the total generator loss and perform backpropagation. We add up all the generator losses and cyclic losses to get the total generator loss. Then, we perform backpropagation to update the weights of the generator and discriminator models

2.2.10.1 Summary :

A CycleGAN attempts to learn mapping from one dataset X to another dataset Y. For example in this experiment from winter and summer images.It does this with two generators, G and F and two discriminators, Dx and Dy:Dx attempts to differentiate real images sampled from X and images produced by generator F. F is updated accordingly to get better at fooling Dx.Dy attempts to differentiate real images sampled from Y and images produced by G, G is updated accordingly to get better at fooling Dy.Cycle loss is added to ensure an image a generator output be mapped back into the original image by the other generator, i.e., F(G(X)) x and G(F(y)) y.Identity loss is added to help preserve tint. It states that when given an image of a target class, a generator should return the same image.The generator uses a series of convolutional layer to map one image to another. discriminator uses a Patch GAN architecture to classify the images

3

Results and Discussion

3.1 Results

3.2 Generating Image using Text

Model used: CLIP Architecture and VQGAN

Input:

include=['A boy at the top of a mountain ',

'A forest with red trees',

'computer on a table',

'one hundred people with green jackets']

exclude='watermark, cropped, confusing, incoherent, cut, blurry'

extras = ""

w1=1

w2=1

noise_factor= 0.22

total_iter=100

- The include parameter is a list of textual descriptions of images that the program should try to generate. For example, one image could depict "A boy at the top of a mountain", another could show "A forest with red trees", and so on.
- The exclude parameter is a list of words or phrases that should not appear in the generated images. These could be things like watermarks or images that are blurry or confusing.
- The extras variable is likely a placeholder for any additional parameters or settings that the program might need.
- The w1 and w2 variables are weighting factors that control how important different aspects of the generated images are.

- The noise_factor variable is a parameter that controls how much random noise is added to the generated images. This noise can help create more natural-looking textures and details.

- The total_iter variable is likely the number of iterations or steps that the program will use to generate each image. These iterations could involve things like adjusting the color balance, sharpening the image, or adding texture. A higher number of iterations typically means a more detailed and accurate image, but can also take longer to generate.

Generated images:

Figure 3.1: A boy at the top of a mountain

Figure 3.2: Forest with red trees

Figure 3.3: One hundred people with green jackets

Figure 3.4: Computer on the table

3.3 Unpaired Image-to-Image Translation

Result:

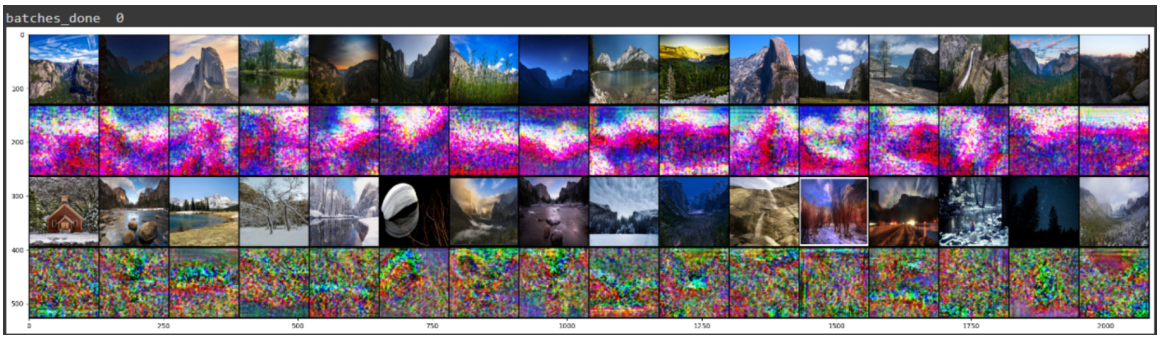

Figure 3.5: After 0 epochs(initial stage)

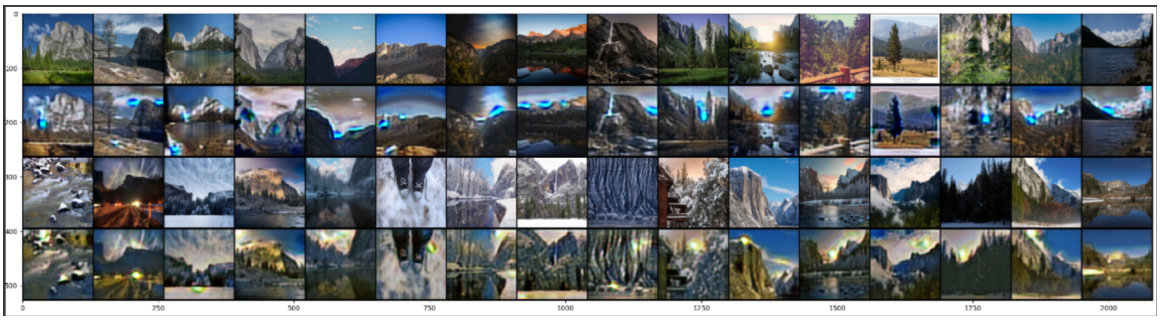

Figure 3.6: After 1 epoch

Figure 3.7: After 20 epochs

Figure 3.8: After 42 epochs

Table 3.1: Evalution Metrics

No of epochs	G loss	adv loss	cycle loss	identity loss
0	12.557497	2.723003	0.664508	0.637882
1	5.048811	0.529835	0.308508	0.286779
5	3.252326	0.265248	0.200170	0.197025
20	2.937852	0.446763	0.143718	0.157017
42	2.797258	0.649425	0.115019	0.128308

In the above table, G loss represents GAN loss,adv loss represents adversarial loss, cycle loss represents cycle consistency loss and other is identity loss

4

Conclusion

This domain adoption report explored the effectiveness of two different approaches, utilizing different architectures, for adapting a source domain to a target domain. Employed the CLIP architecture in combination with VQ-GAN, while this experiment utilized Cycle GAN.

The integration of CLIP architecture and VQ-GAN demonstrated promising results in the domain adoption process. The combination of these models facilitated the transformation of the source domain to closely resemble the target domain. By leveraging the power of CLIP's vision-language understanding and VQ-GAN's generative capabilities, the model successfully captured and reproduced the visual features, textures, and semantics of the target domain. The generated samples exhibited a high level of fidelity and realism, demonstrating the effectiveness of this approach.

Introduced Cycle GAN as the primary framework for domain adaptation. By employing cycle consistency loss and adversarial training, the model learned to translate images from the source domain to the target domain and vice versa. The results showed that Cycle GAN was capable of capturing the domain-specific characteristics and producing visually convincing outputs. Although there might be some limitations regarding fine-grained details and texture preservation, the overall effectiveness of the approach in adapting the domains was evident.

It is important to note that the choice of architecture and approach depends on the specific requirements and constraints of the task at hand. Further research and experimentation can explore variations of these models and architectures to enhance their performance and address their respective limitations. With continued advancements in deep learning and generative models, domain adoption techniques are poised to have a significant impact on a wide range of applications in various domains. [src code:https://github.com/ANUNAY-NALAM/BTP-Code]

Bibliography

[1] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021.

[2] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," 2021.

[3] B. Tan, Y. Song, E. Zhong, and Q. Yang, "Transitive transfer learning," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1155–1164. [Online]. Available: https://doi.org/10.1145/2783258.2783295

[4] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2020.

[5] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," 2017.

[6] C. Yu, J. Wang, Y. Chen, and M. Huang, "Transfer learning with dynamic adversarial adaptation network," 2019.

[7] J. Li, E. Chen, Z. Ding, L. Zhu, K. Lu, and Z. Huang, "Cycle-consistent conditional adversarial transfer networks," in *Proceedings of the 27th ACM International Conference on Multimedia*, ser. MM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 747–755. [Online]. Available: https://doi.org/10.1145/3343031.3350902

[8] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "CyCADA: Cycle-consistent adversarial domain adaptation," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 1989–1998. [Online]. Available: https://proceedings.mlr.press/v80/hoffman18a.html

[9] F. Galatolo, M. G. C. Cimino, and G. Vaglini, "Generating images from caption and vice versa via clip-guided generative latent space search," 01 2021, pp. 166–174.

[10] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.