UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA DELL'AUTOMAZIONE

Using GANs to create training images for Object pose estimation

Relatore
Prof. Alberto Pretto

Laureando
Roberto Del Ben

ANNO ACCADEMICO 2022-2023

Data di laurea 18/10/2023

Contents

List of Figures

Abstract

Object pose estimation is a significant challenge in computer vision, involving determining the position and orientation of an object within an image. This task requires a large training dataset representing various poses of the object. However, collecting such data can be costly and labor-intensive. This thesis examines the architecture of Generative Adversarial Networks (GANs), which can generate realistic images from random noise, and aims to study various GAN architectures and apply them to generate images that can be used in object pose estimation.

Throughout the experiments, pre-rendered images of the Cat's model from LINEMOD [1] will serve as input to the considered GAN's models, aiming to reproduce realistic images containing the reference Cat in the same predefined pose.

First, a Domain Adaptation (DA) approach has been used to generate underwater images based on an unpaired dataset. This experiment served as a proof-of-concept to illustrate the process of generating images in environments where data collection is notably challenging. Next, the focus has been shifted to generating new samples from the LINEMOD's dataset. Here, the objective is to capture features and spatial relationships between objects from the real images and use them to generate new pictures coherent with the input Cat's model. Finally, the best GAN models have been combined with Pixel-wise Voting Network (PVNet [2]) to compare their performance against the PVNet model trained using the conventional image Superimposing Method (PVNet-SM).

The main conclusion of this work is that GANs can achieve similar performance as PVNet-SM while eliminating the need for manual work in background selection and dataset preparation.

Chapter 1

Object pose estimation

Accurately estimating the poses of objects represents a fundamental challenge with profound implications for a wide range of applications, from autonomous navigation and augmented reality to industrial automation and medical imaging. Object pose estimation involves determining the position and orientation of an object relative to a reference frame. It serves as a critical building block for enabling machines to perceive and interact with their surroundings in a manner similar to human perception.

For example, a robot must precisely assess the pose of an object to manipulate it effectively. In augmented reality, a seamless integration of virtual objects into the real world relies on accurate pose estimation. Moreover, applications such as quality control in manufacturing, warehouse automation, and even remote surgery heavily depend on this capability.

1.1 Objective of Object pose estimation

Object pose estimation is a computer vision problem that involves determining the 3D position and orientation (pose) of an object in a 3D space, usually based on a 2D image or a set of images. In other words Object pose estimation aims to find the 6-degree-of-freedom (6-DoF) pose of an object in a 3D space, expressed by:

- **The position of the object**, represented as a 3D vector $\mathbf{t} = [t_x, t_y, t_z]$, where t_x, t_y, and t_z are the coordinates of the object's center in a global coordinate system.
- **The orientation of the object**, represented by a 3x3 rotation matrix $\mathbf{R}$ that encodes the rotation from the object's local coordinate system to the global coordinate system.

The complete 6-DoF pose of the object, denoted as $\mathbf{T}$, can be represented as a 4x4 transformation matrix:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$$

1.2 Challenges

While object pose estimation is a critical problem with numerous applications, it has to deal with many challenges. This section introduces some of them.

1.2.1 Ambiguity

Ambiguity arises when multiple valid pose hypotheses exist for an object based on the available data. This challenge can be characterized by several key aspects:

- **Symmetry and Repetitive Patterns:** Objects exhibit symmetrical or repetitive characteristics, making it difficult to distinguish one pose from another.
- **Occlusions and Partial Observations:** Object is partially occluded or observed only in part.
- **Low-Dimensional Data:** Challenges in distinguishing poses when data is low-dimensional, like 2D images.

1.2.2 Real-Time Requirements

Real-time object pose estimation is a critical challenge that demands the development of efficient algorithms capable of rapid data processing and timely pose estimation. This challenge is particularly important in applications where immediate decision-making and responsiveness are essential.

1.2.3 Challenging Environments

Sometimes Object Pose Detection must be done in challenging environments that do not offer the possibility to acquire high quality data. Underwater cameras, for instance, contend with refraction, distortion and low visibility. Addressing these limitations necessitates specialized techniques, such as sensor calibration and underwater-specific computer vision algorithms.

1.3 PVNet

PVNet [3] is a model that aims to achieve 6DoF pose estimation from a single RGB image, even under severe occlusion or truncation. It takes inspiration from other Perspective-n-Point (PnP) methods, that are characterized by the following two-stage pipeline:

- detect 2D object keypoints using CNNs
- compute 6D pose parameters using the PnP algorithm

PVNet novelty consists in using a Pixel-wise Voting Network (PVNet) to detect 2D keypoints.

1.3.1 Voting-based keypoint localization

Given an RGB image, PVNet predicts pixelwise object labels and unit vectors that represent the direction from every pixel to every keypoint. Given the directions to a certain object keypoint from all pixels belonging to that object, the algorithm generates hypotheses of 2D locations for that keypoint as well as the confidence scores through RANSAC-based voting. Based on these hypotheses, we estimate the mean and covariance of the spatial probability distribution for each keypoint

1.3.2 Uncertanty-driven PnP

Traditional methods use off-the-shelf solvers to solve the Perspective-n-Point (PnP) problem and find the object's pose. However, these methods often overlook the fact that keypoints may have varying confidences and uncertainties. Conversely, PVNet initializes the 6D pose (**R** and **t**) of the object using the standard EPnP algorithm, but then refines this values by minimizing the Mahalanobis distance, which takes accounts of the mean and covariance of each keypoint estimation calculated before.

Chapter 2

LINEMOD dataset

The LINEMOD dataset is a valuable resource in the field of computer vision. It was created to enable the development and evaluation of object detection algorithms, especially in scenarios where objects exhibit distinctive and well-defined features. Introduced by [1], LINEMOD has become a benchmark dataset for researchers working on object detection and localization.

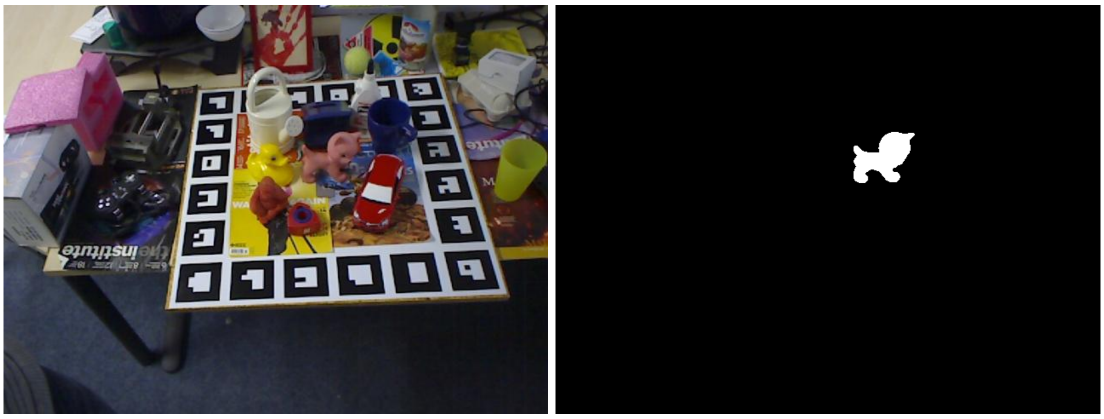
2.1 Dataset Composition

LINEMOD is composed of several object instances, each meticulously curated to provide a challenging yet diverse set of test cases. The key components of the dataset include:

- **Object Models:** For each object instance, LINEMOD provides detailed 3D models that accurately represent the object's shape and appearance. These models serve as ground truth for the object detection task.

- **Annotated Images:** The dataset includes a collection of annotated images captured from various viewpoints. These images contain instances of the target object(s) against complex backgrounds, simulating real-world scenarios.

- **2D Keypoints and Masks:** To aid in the development and evaluation of object detection algorithms, LINEMOD images are annotated with 2D keypoints that indicate specific object landmarks. Additionally, object masks are provided to indicate the pixel-wise location of the object within the image.



(a) RGB image (b) Mask associated to the image

Figure 2.1: LINEMOD dataset example

2.2 Dataset variety

In order to reflect real-world scenarios, LINEMOD images are intentionally made challenging for Computer Vision tasks. Images include:

- **Variations in Lighting:** Images in the dataset exhibit variations in lighting conditions, including shadows and highlights, which require algorithms to be robust to changes in illumination.
- **Occlusions:** LINEMOD images often feature occlusions of varying degrees, emphasizing the need for object detectors to handle partial object visibility.
- **Cluttered Backgrounds:** The dataset includes images with cluttered and complex backgrounds, making it necessary for detectors to discriminate objects from their surroundings effectively.

2.3 Usage and Evaluation

Researchers and developers in the field of computer vision use LINEMOD to benchmark their object detection algorithms, since it provides a standard evaluation framework for assessing the performance of these models. Common evaluation metrics used with LINEMOD include precision-recall curves, average precision (AP), and intersection over union (IoU) scores.

Chapter 3

Generative Adversarial Networks (GANs)

3.1 Introduction

Generative Adversarial Networks (GANs) have emerged as a revolutionary approach in the field of machine learning and computer vision since their introduction by Ian Goodfellow and his colleagues in their 2014 paper titled "Generative Adversarial Nets" [4]. GANs have made significant strides in generating realistic data, creating artwork, super-resolution of images, and even aiding in drug discovery.

3.1.1 Background

Prior to the inception of GANs, generative models primarily relied on probabilistic approaches like Variational Autoencoders (VAEs) and Markov Chain Monte Carlo (MCMC) methods. While these methods were effective, they often struggled with generating high-quality, coherent samples from complex datasets. GANs introduced a new paradigm for generative modeling by employing adversarial training.

3.2 The Original GAN Paper

The original paper [4] introduced the concept of GANs as a two-player minimax game between a generator (G) and a discriminator (D). The central idea is to train the generator to create data that is indistinguishable from real data, while the discriminator aims to distinguish between real and generated data. This adversarial process drives the improvement of both the generator and discriminator over time.

3.2.1 Adversarial Framework

The adversarial framework in the original GAN paper can be summarized as follows:

- **Generator (G)**: The generator takes random noise as input and produces synthetic data samples. Its objective is to create data that is visually and statistically similar to real data.
- **Discriminator (D)**: The discriminator assesses whether a given input is real (from the dataset) or fake (generated by the generator). Its goal is to become proficient at distinguishing real data from generated data.

The generator and discriminator are trained iteratively. The generator tries to minimize the discriminator's ability to distinguish real from fake data, while the discriminator aims to maximize its accuracy in discrimination. This results in a Nash equilibrium where the generator creates increasingly realistic data, and the discriminator's ability to distinguish real and fake data approaches chance.

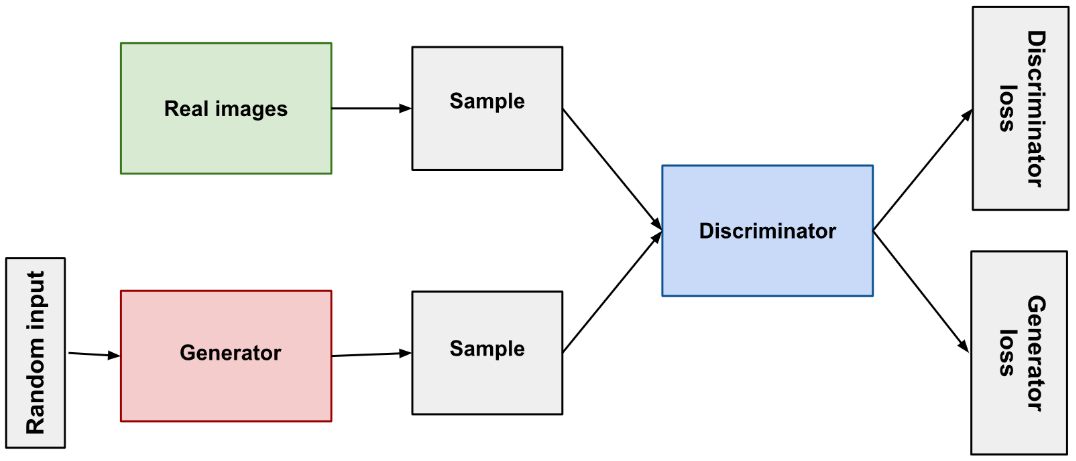


Figure 3.1: Sketch of a GAN network

3.2.2 Loss Function

The original GAN paper introduced the following minimax loss function:

$$V(G,D) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

Where:

$V(G,D)$ is the value function.

$\mathbb{E}_{x\sim p_{\text{data}}(x)}$ represents the expectation over real data.

$\mathbb{E}_{z\sim p_z(z)}$ represents the expectation over the noise vector.

$D(x)$ is the discriminator's output for real data.

$D(G(z))$ is the discriminator's output for generated data.

3.3 GAN Architectures and Variants

Since the original GAN paper, numerous architectures and variants have been proposed to address various challenges and improve GAN performance. Some notable variants include:

- **Conditional GANs (cGANs)**: Extending GANs to generate data conditioned on additional information such as class labels, enabling controlled data generation.
- **Deep Convolutional GANs (DCGANs)**: Leveraging convolutional neural networks to generate high-resolution images with improved stability.
- **Wasserstein GANs (WGANs)**: Introducing the Wasserstein distance as a new metric for measuring the discrepancy between real and generated data distributions, leading to more stable training.

3.3.1 Generator Architectures in GANs

The generator is responsible for generating data that resembles the real data it is trained on. Over the years, various generator architectures have been proposed, each with its own strengths and applications.

Fully Connected Networks (FCNs)

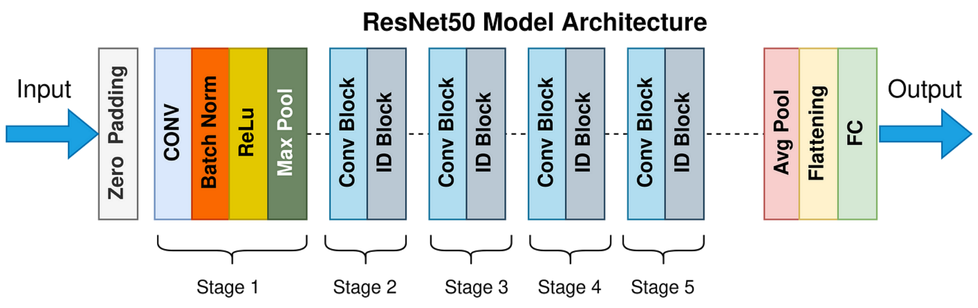
Fully Connected Networks (FCNs), also known as feedforward neural networks, were among the earliest generator architectures used in GANs. These networks consist of multiple layers of densely connected neurons. While FCNs are simple and can be used for various tasks, they may struggle to capture complex spatial dependencies in data such as images.

Convolutional Neural Networks (CNNs)

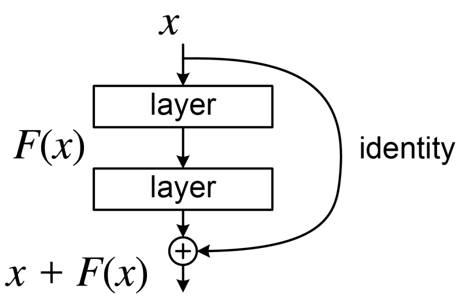
Convolutional Neural Networks (CNNs) have become the backbone of many modern generator architectures in GANs. CNNs are well-suited for image generation tasks because they can capture local patterns and spatial hierarchies effectively. These networks typically comprise convolutional layers, followed by normalization and activation layers.

Residual Networks (ResNets)

Residual Networks (ResNets) introduced residual connections that enable the training of very deep networks. In GANs, ResNets have been employed in generator architectures to facilitate the generation of high-quality images with improved training stability. These architectures are particularly useful when dealing with high-resolution image generation.



(a) Resnet



(b) Resnet Block

Figure 3.2: Sketch of Resnet and Resnet Block structures

Autoencoders and Variational Autoencoders (VAEs)

Autoencoders and Variational Autoencoders (VAEs) can also serve as the basis for generator architectures in GANs. Autoencoders compress input data into a lower-dimensional representation and then decode it to reconstruct the original data. VAEs add a probabilistic component, making them suitable for generating diverse data samples with controlled variability.

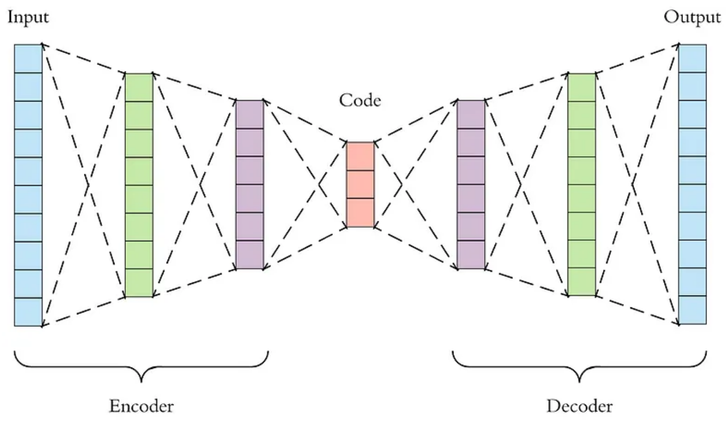


Figure 3.3: Sketch of Autoencoder structure

U-Net Architecture

The U-Net architecture is a specialized network design primarily used in tasks involving image segmentation, medical imaging, and image-to-image translation. It consists of a contracting path, which captures context, and an expansive path, which generates the output. U-Net's unique structure incorporates skip connections that help preserve spatial information and alleviate the vanishing gradient problem.

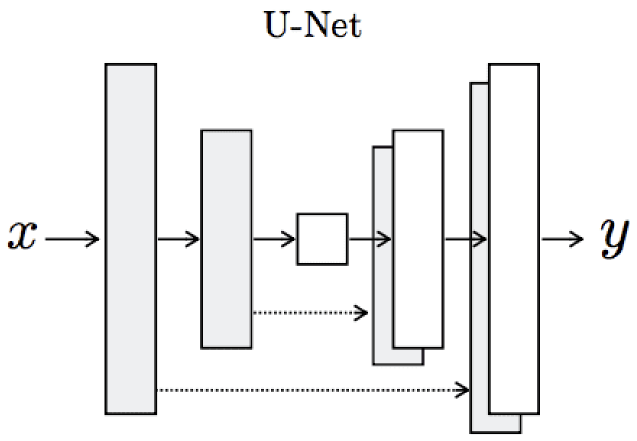


Figure 3.4: Sketch of U-Net structure

Self-attention generators

Self-attention generators in GANs utilize self-attention layers to calculate attention scores between different parts of the generated content, allowing the model to focus on relevant regions and capture long-range dependencies.

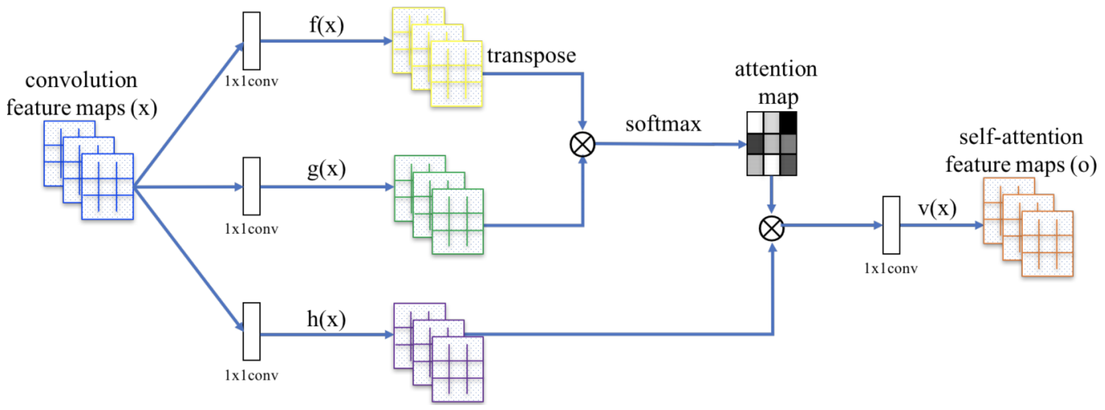


Figure 3.5: Sketch of Self-attention layer structure

3.3.2 Discriminator Architectures in GANs

CNN Discriminator

Similarly to the Generator counterpart, CNN-based discriminators are effective at capturing spatial hierarchies in the input data.

PatchGAN Discriminator

The PatchGAN discriminator evaluates small patches of the input image rather than the entire image. This approach allows for fine-grained discrimination and can be particularly useful for tasks like image-to-image translation.

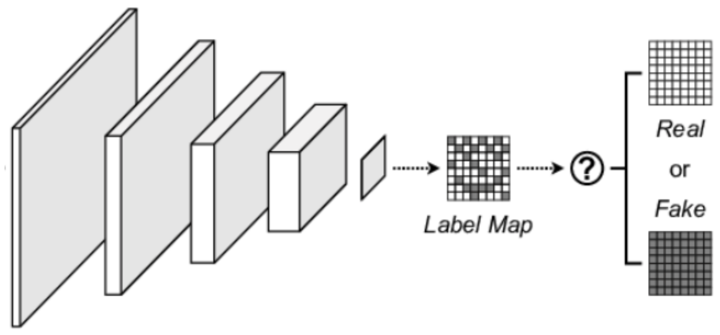


Figure 3.6: Sketch of Patch Discriminator structure

Multi-Scale Discriminator

In some GAN variants, multiple discriminators are employed at different scales of the image. These discriminators help capture both local and global features, enhancing the model’s ability to generate high-quality images.

3.4 Applications of Generative Adversarial Networks (GANs)

Image Generation and Style Transfer

GANs are renowned for their ability to generate high-quality, photorealistic images. They have been used to create art, simulate natural scenes, and generate images from textual descriptions. Style transfer, which involves transferring the artistic style of one image onto another, is another popular application of GANs, enabling the creation of visually stunning images and videos.

Super-Resolution and Image Enhancement

In the realm of image processing, GANs have been employed to enhance image quality and achieve super-resolution. They can upscale low-resolution images, restoring fine details and improving visual clarity. This is invaluable in medical imaging, surveillance, and enhancing the quality of old photographs.

Data Augmentation

GANs can be used to augment datasets by generating synthetic data that is similar to the real data. This is particularly useful when working with limited data or addressing class imbalance in machine learning tasks.

Text-to-Image Synthesis

Text-to-image synthesis is the process of generating images from textual descriptions. GANs have shown promise in this domain, allowing for the creation of visual content from natural language input.

Generative Models for Audio and Music

Beyond visual data, GANs have been adapted for audio generation and music composition. They can produce realistic audio samples and even compose music.

Chapter 4

Image-to-Image Translation

Image-to-image translation is a popular and challenging computer vision task that involves converting an input image from one domain to another while preserving relevant content and structure.

4.1 Image-to-Image Translation applications

- **Semantic Segmentation to Real Image**: Converting a semantic segmentation map, which assigns a class label to each pixel, into a realistic image that visually represents the segmented objects.
- **Style Transfer**: Changing the artistic style of an image, such as converting a photograph into the style of a famous painting.
- **Colorization**: Adding color to grayscale images, preserving the semantic information and object boundaries.
- **Super-Resolution**: Increasing the resolution of a low-resolution image while preserving its content.
- **Domain Adaptation**: Adapt a model trained on data from one domain to perform well on data from a different but related domain, acting only on the relevant aspects of the domain shift.

4.2 Pix2Pix: cGAN for Image-to-Image Translation

One of the pioneering approaches to image-to-image translation using GANs is the Pix2Pix framework [5]. Pix2Pix employs a conditional GAN architecture, where both the generator and discriminator are conditioned on the input image. Generator is a U-Net that takes an input image and generates a corresponding output image. The discriminator is a PatchGAN Discriminator and it is trained on paired input and output images.

The training objective is to minimize the adversarial loss, which encourages the generator to produce realistic outputs, and a pixel-wise loss, such as L1 loss, which enforces pixel-level similarity between the generated and target images.

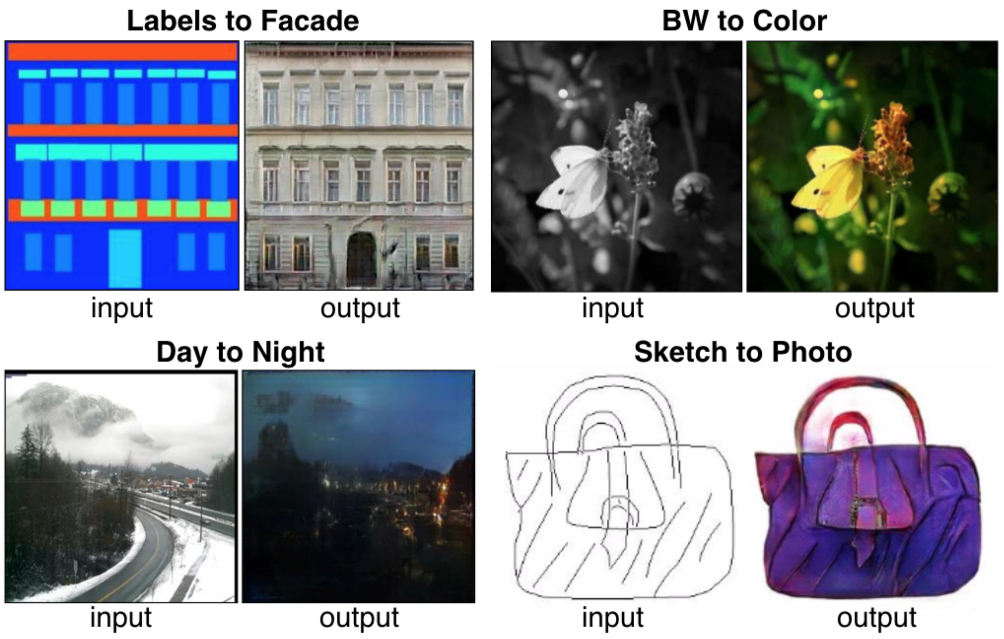


Figure 4.1: Pix2Pix examples

4.3 CycleGAN: Unpaired Image-to-Image Translation

While Pix2Pix requires paired training data (input-output image pairs), CycleGAN introduced a breakthrough in unpaired image-to-image translation. It uses a cycle-consistency loss to enable the translation between two domains without direct correspondences.

CycleGAN consists of two generators and two discriminators, one for each domain. The generators aim to translate images from one domain to the other and back, while the discriminators evaluate the authenticity of the generated images in the respective domain. The cycle-consistency loss enforces that translating an image from domain A to domain B and then back to domain A should result in an image close to the original input.

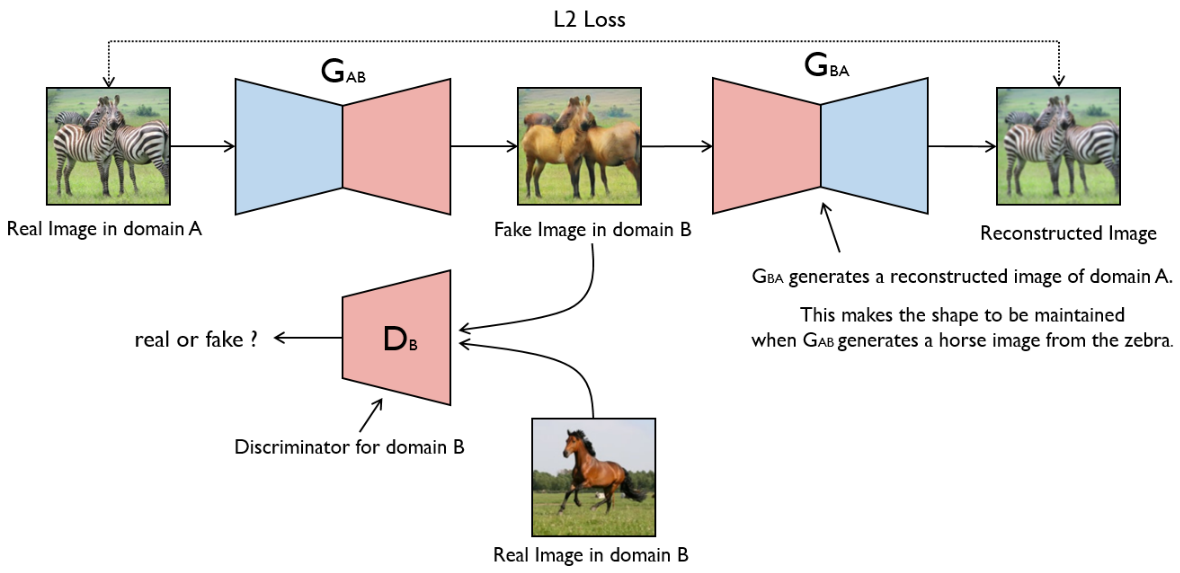

L2 Loss
G_AB
G_BA
Real Image in domain A
Fake Image in domain B
Reconstructed Image
G_BA generates a reconstructed image of domain A.
This makes the shape to be maintained when G_AB generates a horse image from the zebra.
real or fake ?
D_B
Discriminator for domain B
Real Image in domain B

Figure 4.2: Sketch of CycleGAN structure

4.4 Contrastive Unpaired Translation (CUT)

CUT is another model designed for unpaired image translation tasks that aims to be an evolution over CycleGAN. One of its key advantages is faster training as it doesn't require training a second module to return to the original domain. CUT instead employs a loss function called PatchNCELoss (Patch-wise Noise-Contrastive Estimation Loss), which is a variant of the standard cross-entropy loss. The main idea behind this loss is that corresponding patches should have mutual information.

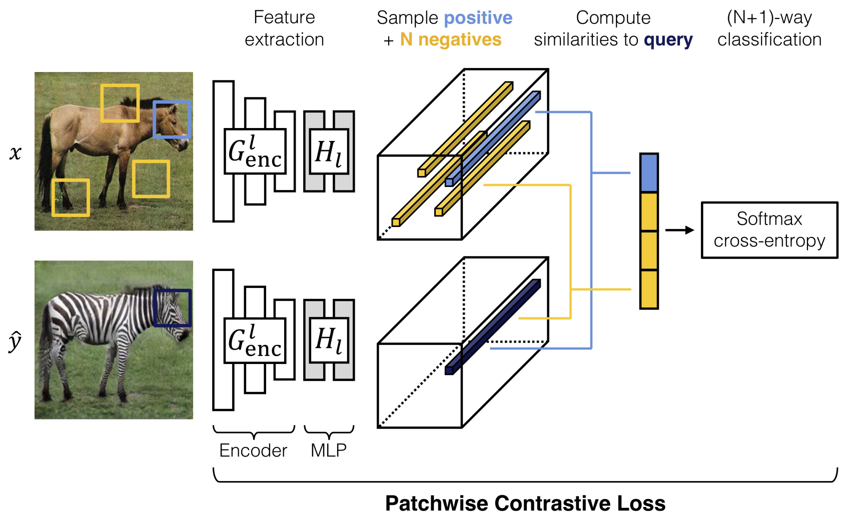


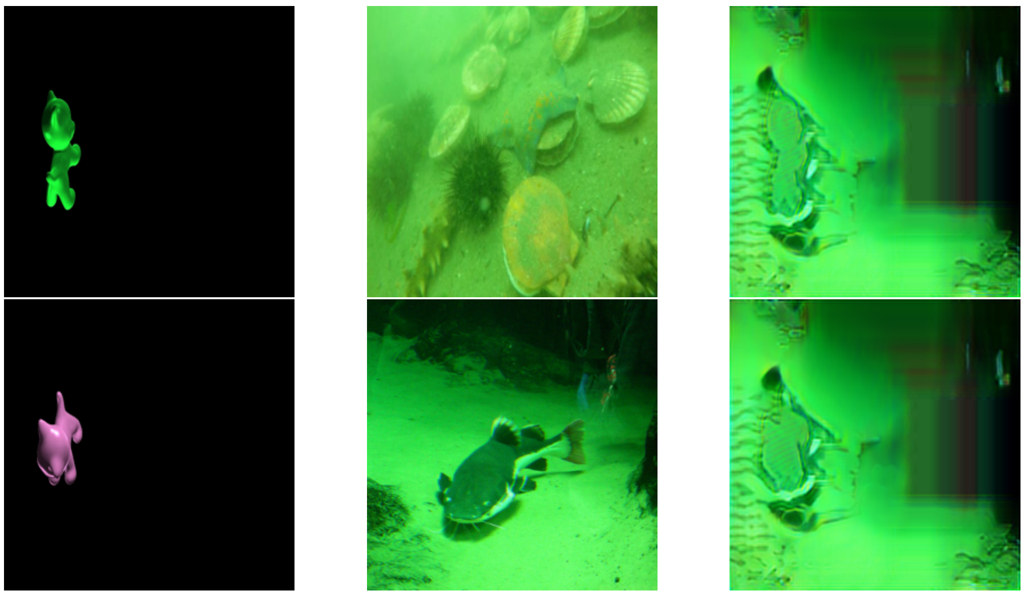
Figure 4.3: CUT's PatchNCE Loss

4.4.1 Using CUT to generate underwater images

As said before, gathering high quality underwater images is a very challenging task. To address this challenge a key idea is to use CUT to generate synthesized underwater images, an subsequently use them as dataset in other applications. In this section CUT has been trained to adapt images from a Domain A, containing images with the Cat's model from the LINEMOD, to a Domain B containing underwater images.

Black background

In the first experiment, the Cat's model was superimposed on a plain black background, revealing some difficulties for CUT in handling it. It appears that a uniform color is not well adapted in the generated image, resulting in visible texture inconsistencies. One possible reason for this occurrence could be that the PatchNCE loss, which aims to maintain high correlation between patches of the input and output images, struggles when there is a lack of variety within the input picture, making the network training less effective.



(a) Domain A (b) Domain B (c) Fake Domain B

Figure 4.4: Results of CUT training with black background

Water-environment background

To address the previous issue, the next step involved using a random water-environment background for the Cat's model. As expected, this change led to significantly more consistent images with finer texture details. When examining the Cat's model, a small halo can be observed around it, and it exhibits some transparency, as if it were made of glass.

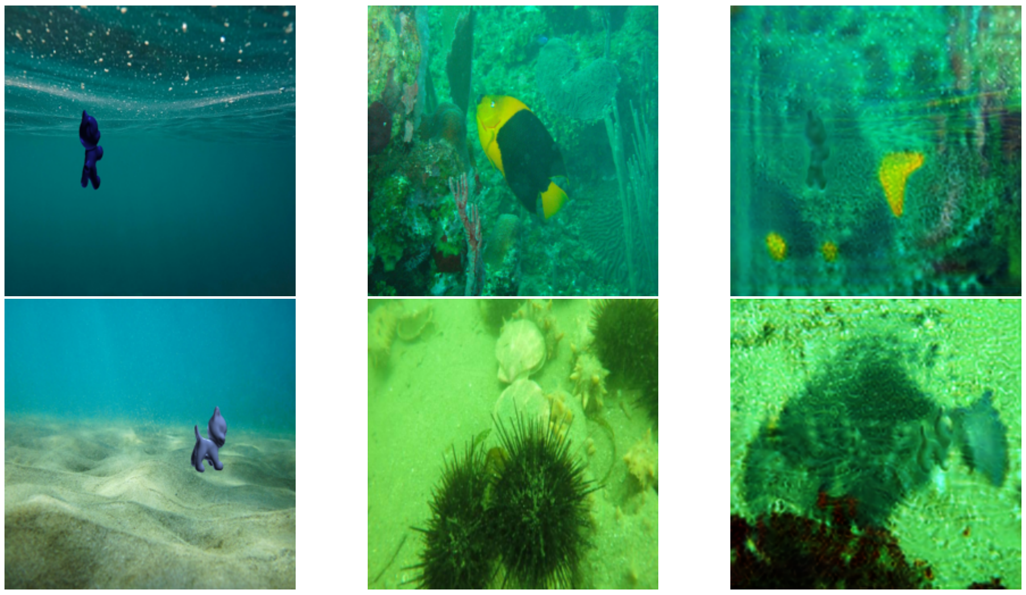
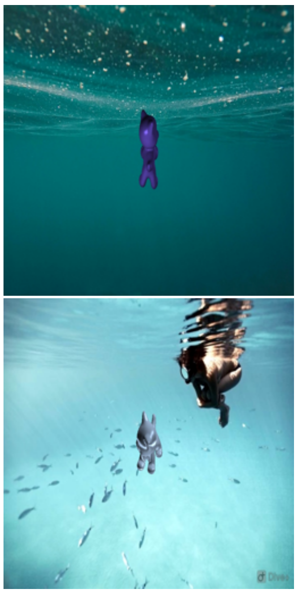
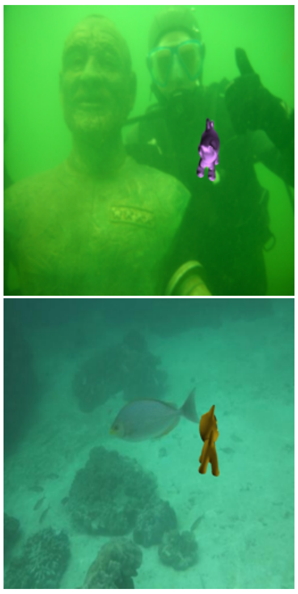

(a) Domain A (b) Domain B (c) Fake Domain B

Figure 4.5: Results of CUT training with a random background

Water-environment background and Cat superimposed in domain B

From an experimentation point of view it might be interesting to see how CUT behaves if the Cat's model is superimposed also in domain B. Unfortunately this change led to a worse image generation, with the appearance of distortion around the model, which is a deal-breaker for object pose estimation. The likely reason is that the Discriminator should be trained to understand the textures of the target domain rather than the Cat's model itself, allowing the Generator to ensure that the output image's structure remains similar to the input.


(a) Domain A


(b) Domain A

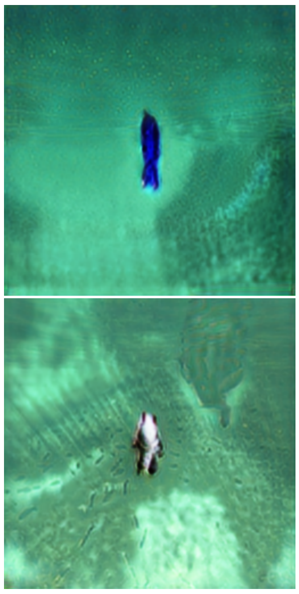

(c) Domain A

Figure 4.6: Results of CUT training with a random background and Cat superimposed in both domains

Chapter 5

Image Synthesis

5.1 Introduction

The primary goal of this chapter is to show solutions for generating new data samples from a dataset source using GANs.

In the following experiments, the objective is to create new sample data from the LINEMOD dataset. Beginning with an image of the Cat's model positioned in a predefined location as a prior, the aim is to synthesize an image that closely resembles the LINEMOD dataset while retaining the Cat's model in its original position.

The key components of the experimental roadmap include:

1. **Architectural Innovations**: Modifications to the neural network architecture, such as the incorporation of attention mechanisms and the utilization of patch-wise discriminators. This techniques are explored with the aim of not only improving the model's capability to produce high-quality images, but also effectively managing prior knowledge given by input images.

2. **Loss Function Refinements**: Provide additional losses to enhance particular aspects of the synthetic image. For example the incorporation of L1 loss on masked sections of the image, and the usage of grayscale loss to emphasize structural details.

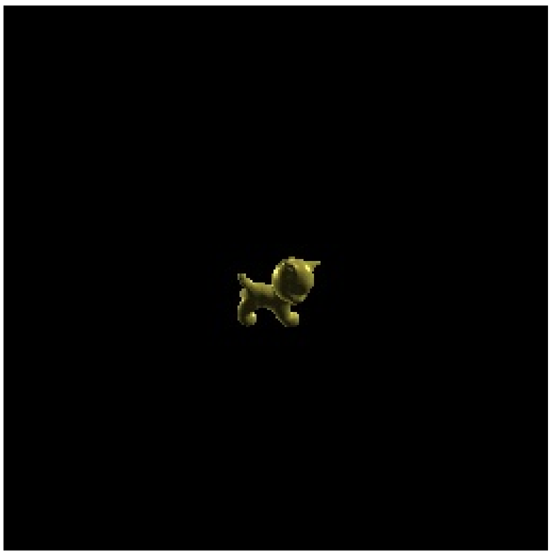
3. **Data Augmentation**: Given the context of data scarcity, data augmentation assumes a pivotal role in enhancing model robustness and generalization.

5.2 Dataset preparation

The dataset for all of these experiments is sourced from two primary channels:

- **Input:** Images rendered with PVNet utility
- **Ground truth:** LINEMOD real images

The entire LINEMOD dataset contains in total 1179 real images, but for the scope of the upcoming chapter, most of the subsequent experiments will show results derived from a subset of 177 images, which corresponds to the PVNet default subset used for training and not for validation or testing.


(a)

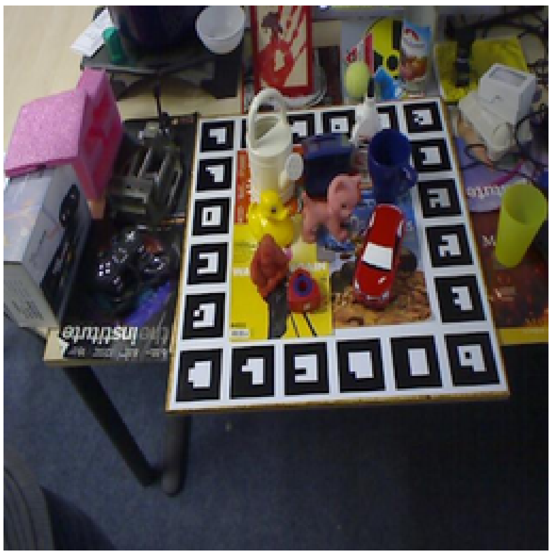

(b)

Figure 5.1: Sample Images from Input Dataset and Target Domain

5.3 Building a model from scratch

5.3.1 DCGAN

Beginning with DCGAN establishes a strong foundation for the exploration into image synthesis. It is the simplest model used in image synthesis and it provide an immediate insight about what are the challenges to overcome.

In the results displayed in Figure 5.2, it is immediate to notice that the DCGAN model's performance doesn't meet the expectations. Specifically, there are the following issues:

- **Cat Doesn't Appear:** DCGAN model is supposed to generate images containing the Cat's model, but it's failing to do so.
- **Low Image Quality:** The generated images don't look very good; they're blurry and lack detail.
- **Mode Collapse:** This is a common problem with GANs. It means the generator is getting stuck and producing very similar images instead of a variety.

The reasons behind these problems are quite simple, and next experiments aim to fix or reduce them:

- **Poor Model Design:** DCGAN might not be suitable for this task. It might lack the necessary complexity or features to capture the required image details properly.
- **Data Scarcity:** There is a limited amount of training data, and this made it challenging for the model to learn and generate high-quality images. The Discriminator overfit early in the training process, which stops the Generator from effectively learning.

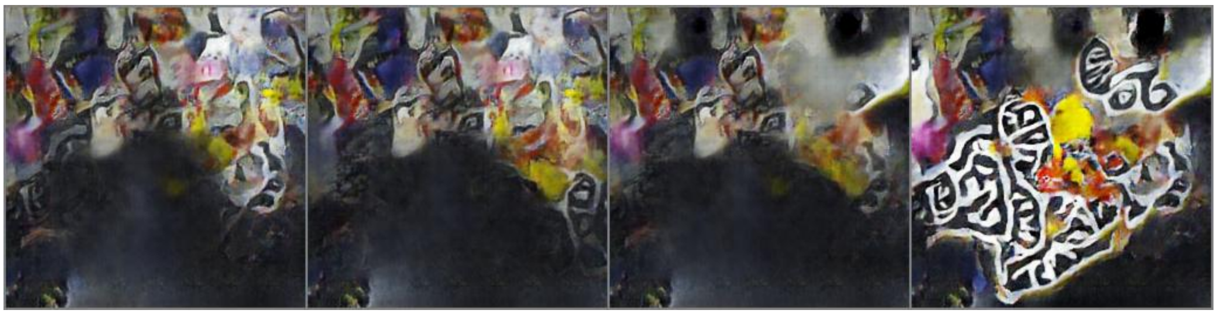


Figure 5.2: Result of training DCGAN with LINEMOD dataset

5.3.2 U-Net generator and masked loss

One simple solution to address the issue of Cat not appearing is to use the pre-rendered Cat image as input to the generator and apply a loss function to ensure that rendering occur in the output as well. As discussed in Section 3.3.1, the U-Net generator helps in preserving spatial information between the input and the generated image, making it well-suited for this particular task. The loss can be calculated exclusively on the pixels related to Cat's model.

```
#CODE FOR THE GRAYSCALE MASKED LOSS

X_MASKED = torch.mean(X * X_mask, (1)).view(
    b_size, 1, params["imsize"], params["imsize"]
)

FAKE_Y_MASKED = torch.mean(fake_Y * X_mask, (1)).view(
    b_size, 1, params["imsize"], params["imsize"]
)

non_zero_pixels = torch.nonzero(X_mask).size()[0]

masked_loss = mask_criterion(X_MASKED, FAKE_Y_MASKED) / non_zero_pixels
```

Figure 5.3 shows in fact that after this changes, the Cat's model becomes visible in the generated image, however the overall image quality still falls below the desired standard.

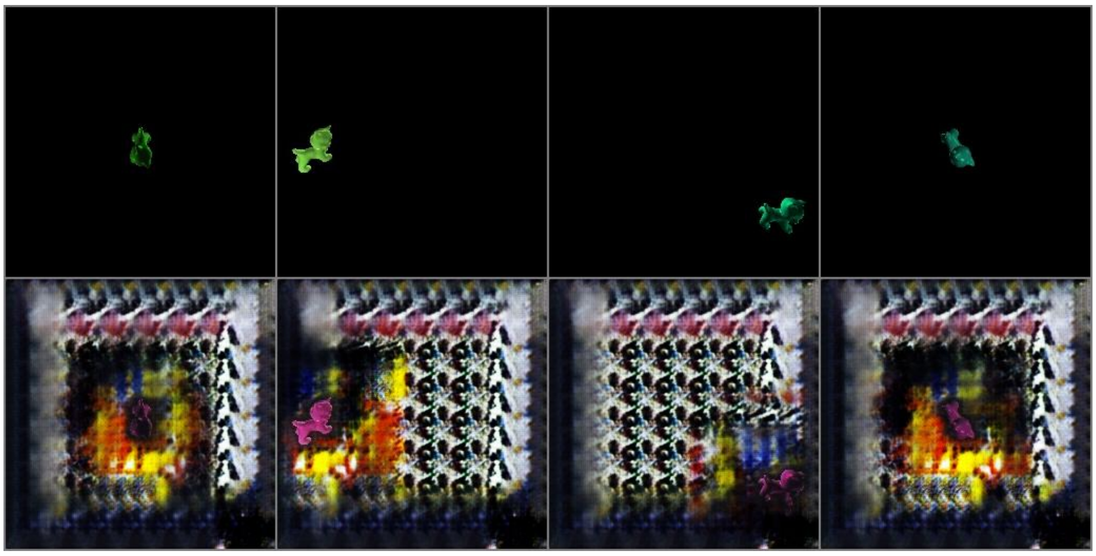
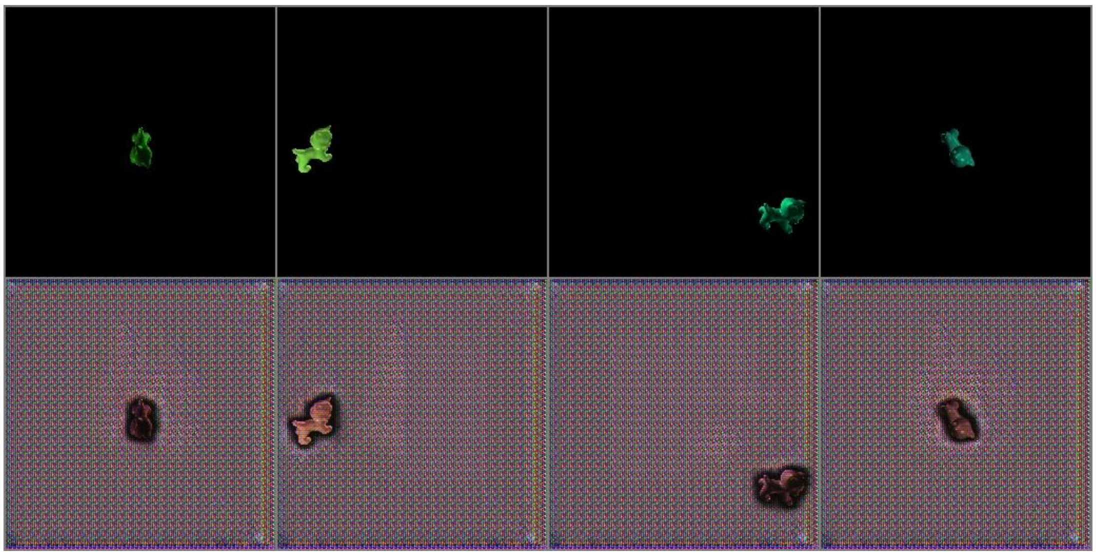


Figure 5.3: Result of training DCGAN with LINEMOD dataset with U-NET and masked loss

5.3.3 Introducing conditional information

The idea is to use the input as a conditioning factor of the discriminator training, in order establish a connection between the Cat's model and the background. The LINEMOD dataset conveniently supplies masks for the photographed Cat, allowing for straightforward extraction of only the model from the dataset.

Unfortunately this final step resulted in non-convergence of the model, both on the training dataset and the complete dataset, leaving only one viable option: the need to update the model again.


(a) Training LINEMOD dataset

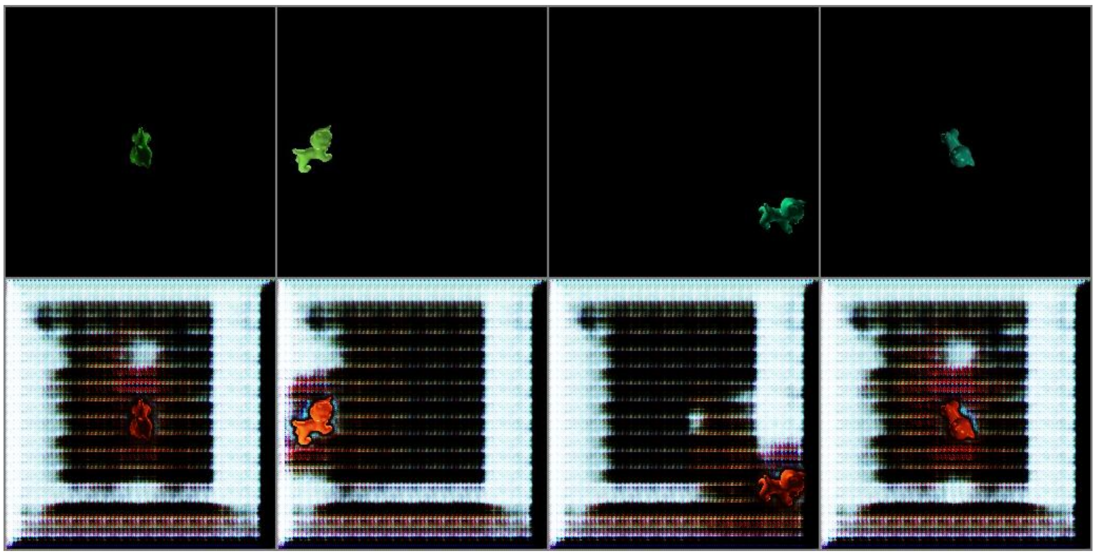

(b) Complete LINEMOD dataset

Figure 5.4: Results of DCGAN with conditional information

5.3.4 Patch Discriminator

Since the Generator has already been updated, it's time to focus on the Discriminator. Taking cues from the Pix2Pix [5] approach, one viable option is implementing a Patch discriminator (Section 3.3.2).

Figure 5.5 and Figure 5.6 shows that Patch discriminator successfully restored convergence to the model and, in addition, introduced certain spatial information between Cat and the background, even though this one lacks geometric coherence with the ground truth.

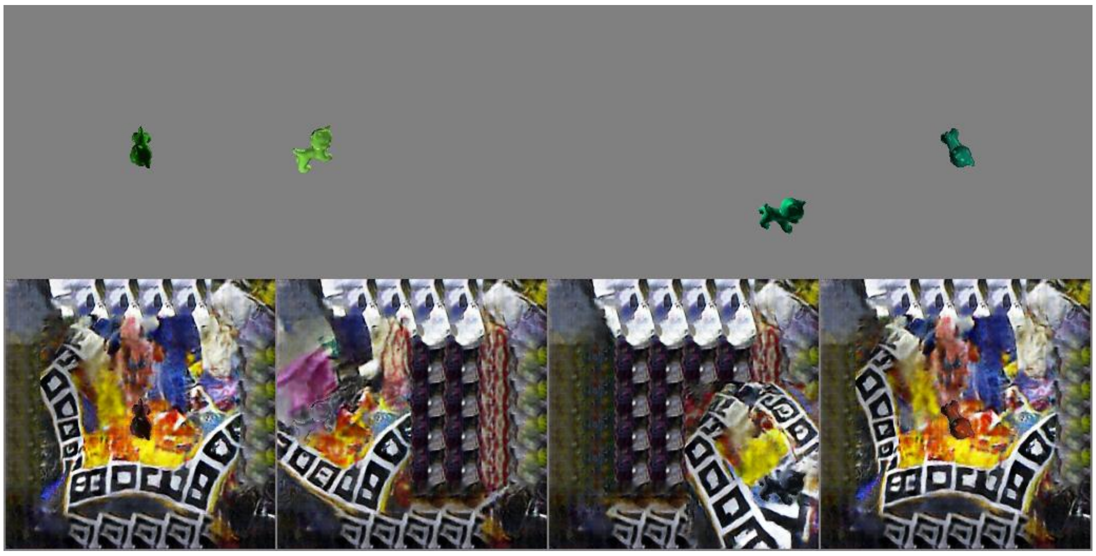


Figure 5.5: Patch Discriminator + L1 masked loss

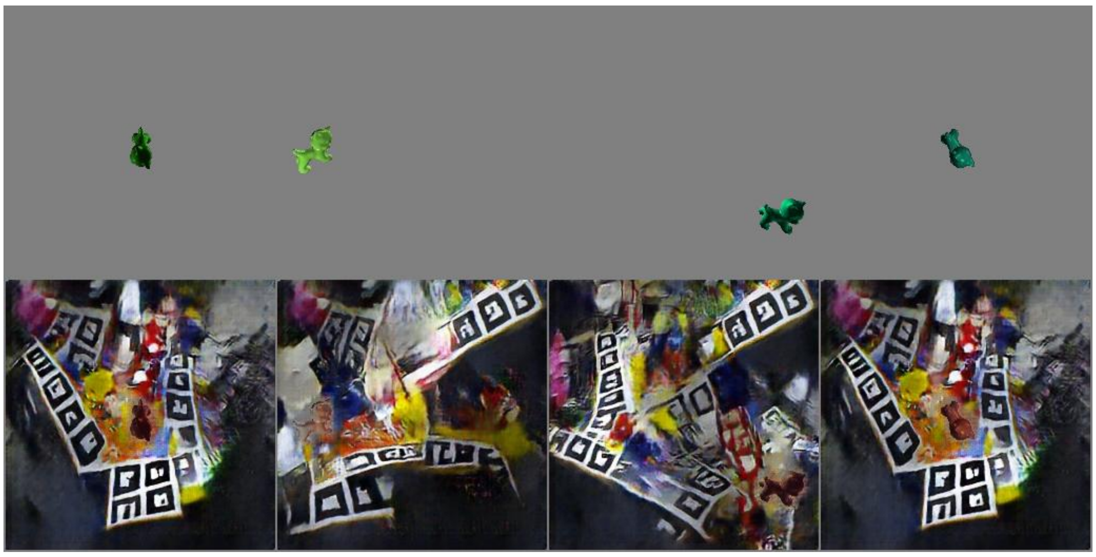


Figure 5.6: Patch Discriminator + MSE masked loss

5.3.5 Self-attention U-Net

Incorporating self-attention into the generator is a common practice employed to strengthen spatial relationships between objects in the generated images. As depicted in Figure 5.7, the board now consistently maintains a coherent position relative to the Cat's model. Additionally, the board resembles more to the ground truth than before, confirming that the inclusion of self-attention was indeed the appropriate choice.

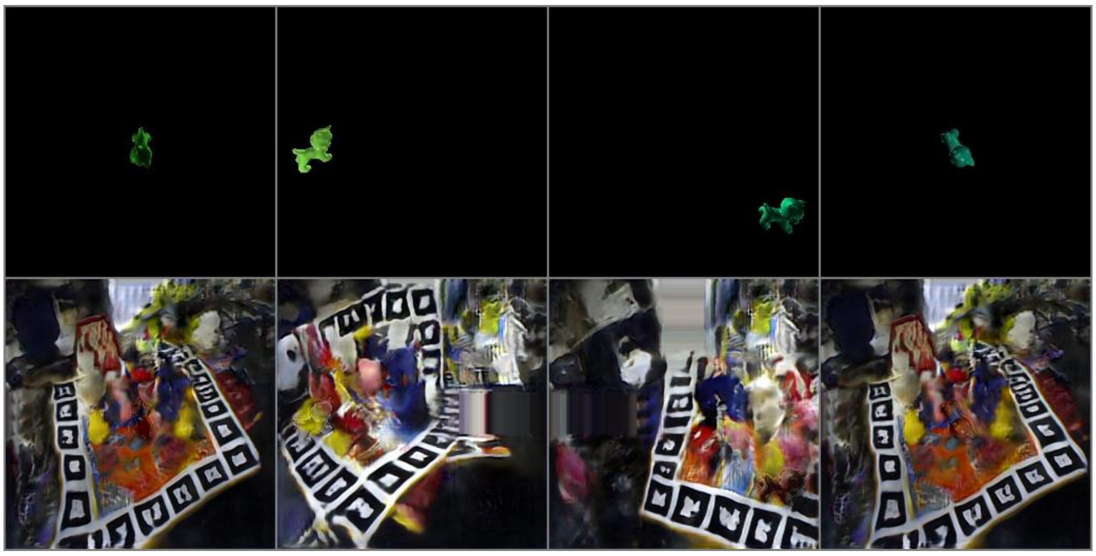
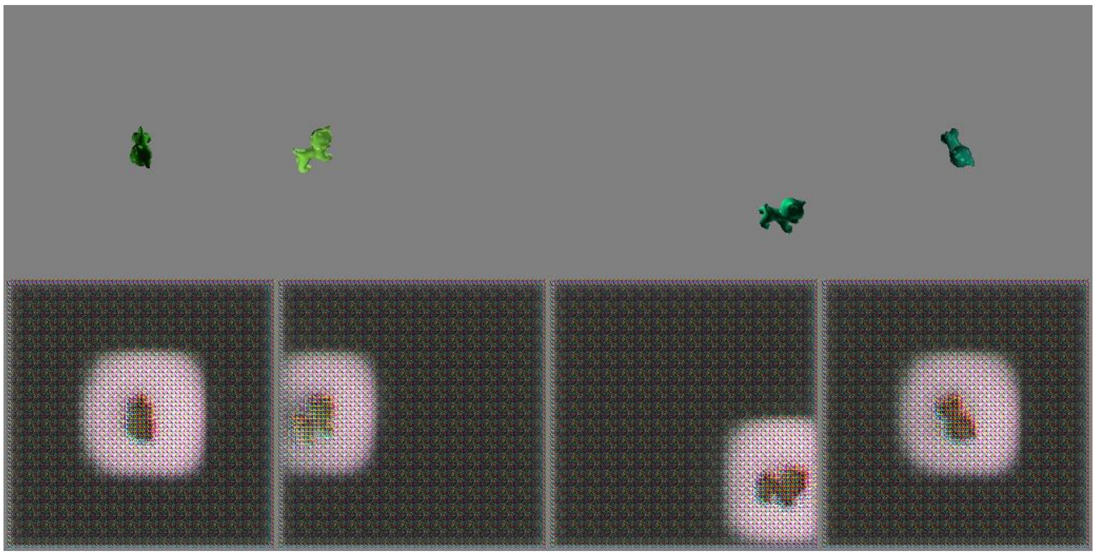


Figure 5.7: Result of training with Self Attention U-Net

Even in the early stages of training, the impact of self-attention becomes particularly pronounced. As shown in Figure 5.8, it's observable that when self-attention is employed, the generated samples already exhibit a distinct pattern around the Cat's model. This suggests that, to some extent, the generator immediately manages to capture information about the board's structure from the ground truth images.


(a) Without SA

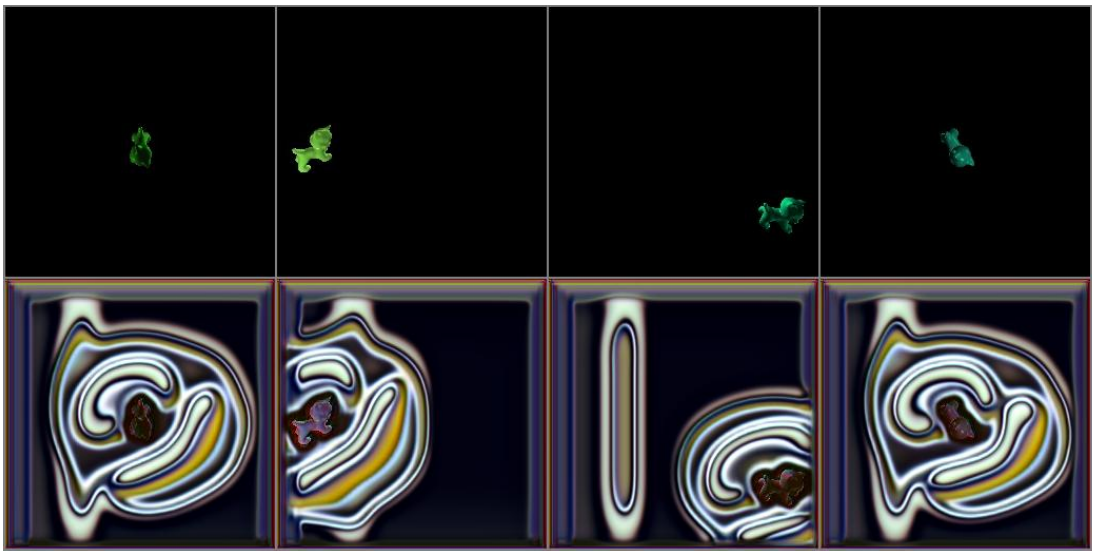

(b) With SA

Figure 5.8: Early stages of training w/wo Self Attention mechanisms

5.4 FastGAN

FastGAN, as introduced in [6], is a ready to use model specifically designed to perform well on low-dimensional datasets. Its primary objectives are minimizing mode collapse and avoid discriminator overfitting.

5.4.1 Main Features

Generator

To generate high-resolution images the Generator inevitably needs to become deeper. This usually leads to longer training time and the risk of suffering from the gradient vanishing problem. To avoid all of this, FastGAN's creators introduced a module called Skip-Layer Excitation. This module takes inspiration from the ResBlock, but it manage to connect layers at different resolutions within the upsampling process. By doing so, it effectively addresses the vanishing gradient problem while avoiding the additional computational overhead associated with the traditional ResBlock.

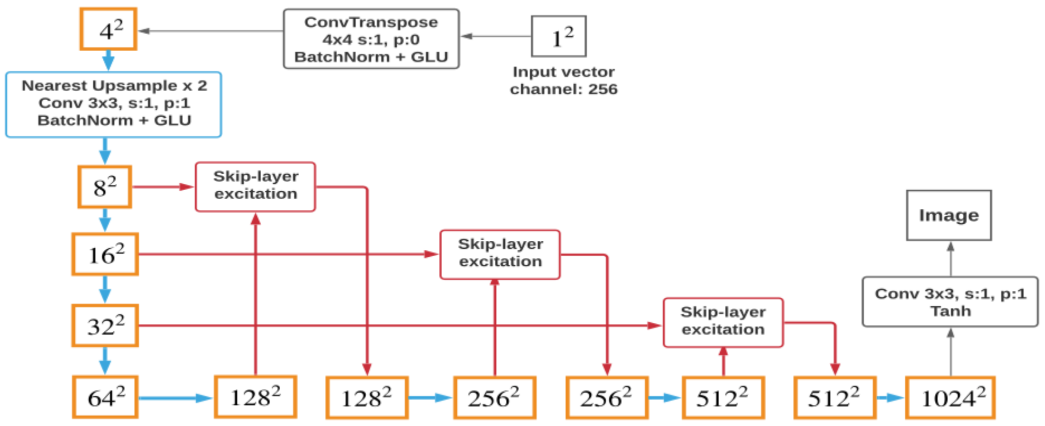


Figure 5.9: Sketch of FastGAN Generator

Discriminator

The Discriminator is treated as an encoder and trained with small decoders. This auto-encoding training forces the Discriminator to extract image features that the decoders can effectively reconstruct. The decoders are optimized using a basic reconstruction loss, exclusively trained on real samples.

5.4.2 Training with LINEMOD

After training FastGAN with LINEMOD images, it becomes clear that the model can indeed generate high quality images from a small dataset. However, as shown in Figure 5.10, they tend to lack in finer details. To address this limitation and have images that can train an Object pose estimation models, it becomes necessary to introduce additional information of the Cat's model into the image generation process.

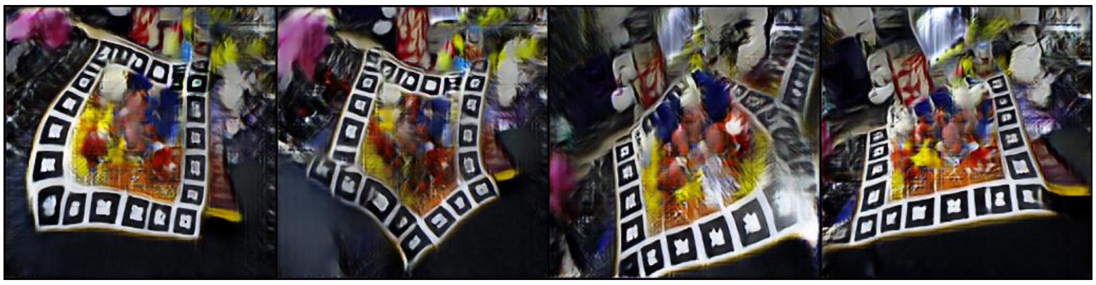


Figure 5.10: Result of training FastGAN on LINEMOD dataset

Introducing Cat in FastGAN

Since it's crucial to maintain Cat's structure properties as close as possible to the input, the initial attempt involves superimposing Cat over the generated image. Unfortunately, this approach made the pipeline unstable, as evidenced in Figure 5.11.

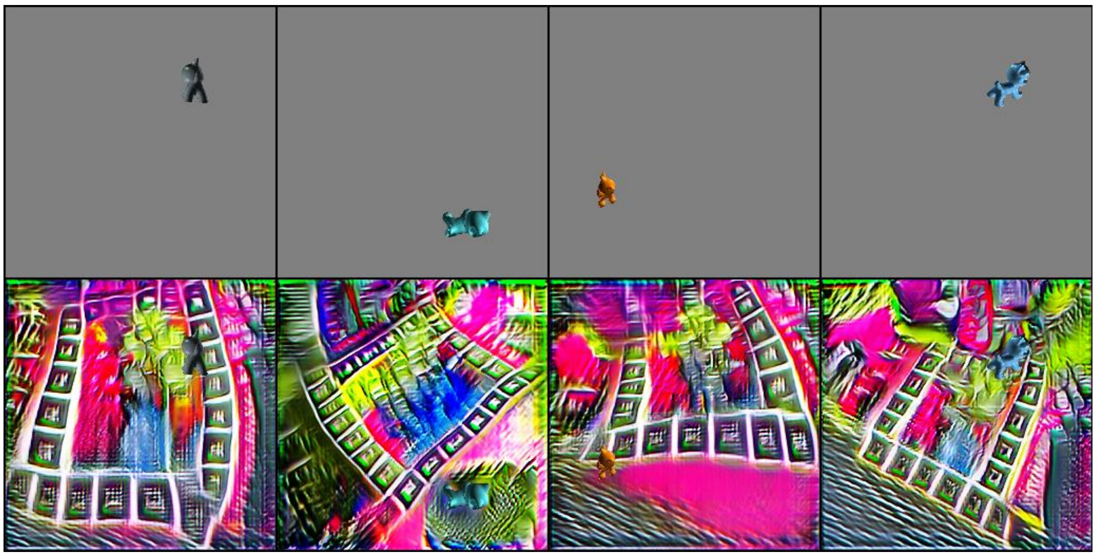
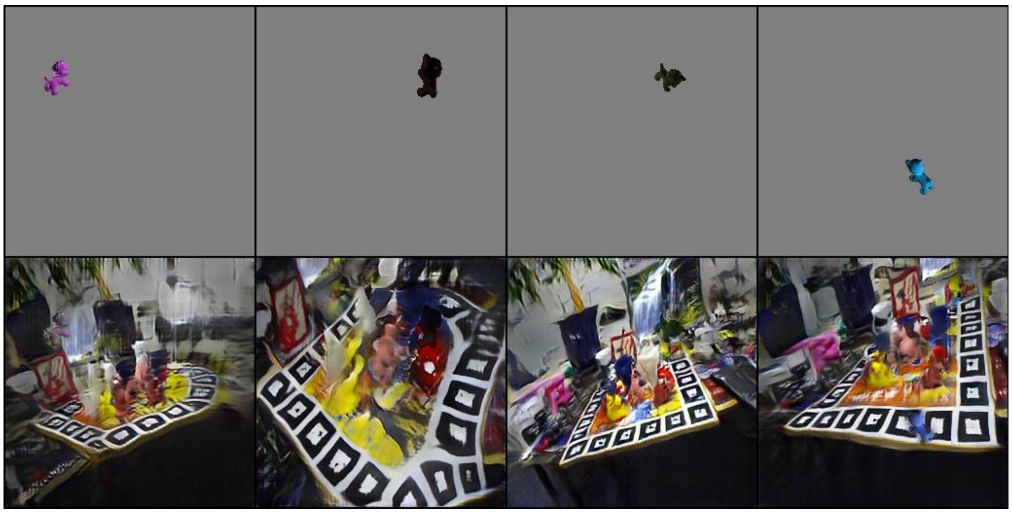


Figure 5.11: Results of superimposing Cat over the generated image

The next approach involve making slight modifications to the FastGAN model. These consist in introducing an encoded version of Cat's image during the upsampling process, and also directly adding (not superimposing) the input image to the output, similarly to the U-Net structure. The objective is to enable the training process to learn how to effectively blend Cat's model in the background. In order to stabilize the training a masked loss has been added. Results in Figure 5.12.

These results demonstrate that FastGAN is highly effective for generating samples from low-dimensional datasets. However, harmonizing an input image with the generated instance is not its intended purpose. Nonetheless, it can serve as an alternative approach for general image superimposition.


(a) Grayscale loss

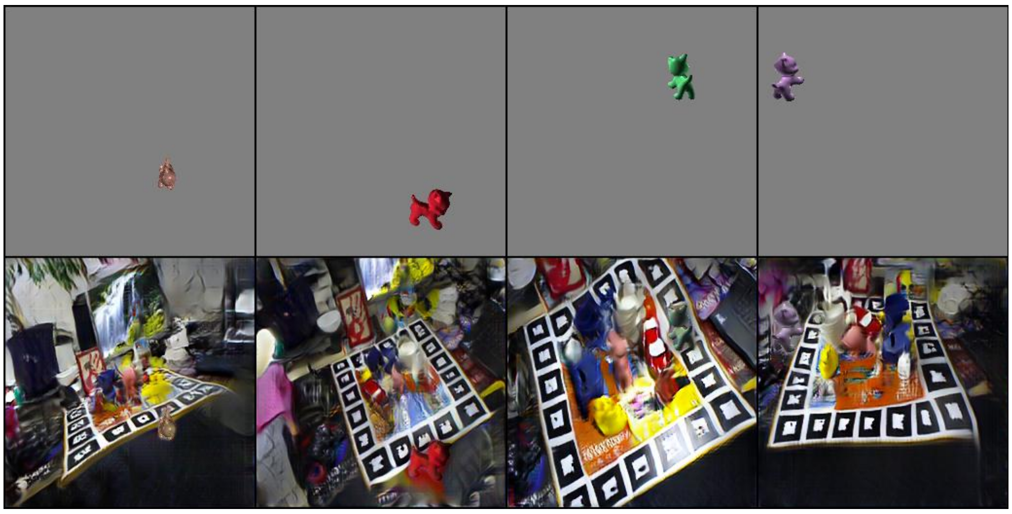

(b) RGB loss

Figure 5.12: Results of the modified FastGAN model

Chapter 6

End-to-End models

Finally, this chapter will assess the performance of Generative Adversarial Networks (GANs) in the context of object pose detection, comparing them to a simple superimposing technique. This evaluation aims to show the potential of GANs to enhance the accuracy and effectiveness of object pose detection models when combined with them.

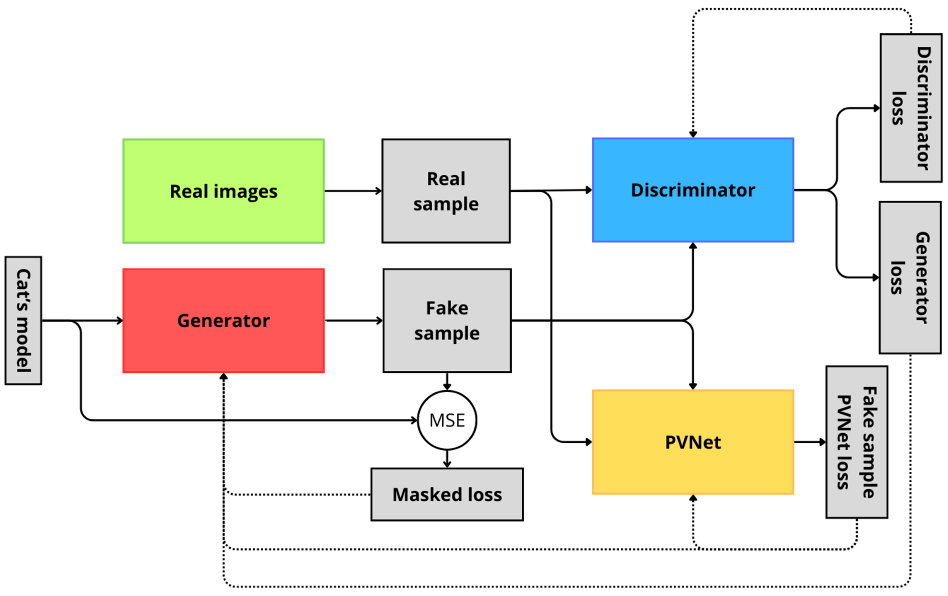
6.1 Training pipeline

Best of PatchGAN and FastGAN models have been incorporated into the PVNet model and trained through the following pipeline:

1. Train the Discriminator using both real images and images generated by the Generator.
2. Train the Generator using the output from the Discriminator, a masked loss, and the loss returned by PVNet.
3. Train PVNet using the dataset of real images along with a set of images generated by the Generator.

The ratio of real images to fake images is adjusted to ensure an equal number of total iterations in both the training on superimposed images and the training with the Generative models. It is worth noting that the pipeline is not only training an Object Pose Estimation model, but also a Generative model at the same time.

```
# End-to-End pipeline

# Retrieve inputs
real_image, real_mask, real_pose = real_sample
CAT_image, CAT_mask, CAT_pose = CAT_sample

# PVNet training
fake_image = Generator(CAT_image)
pvnet_real_loss = PVNet(real_image, real_mask, real_pose)
pvnet_fake_loss = PVNet(fake_image, CAT_mask, CAT_pose)

# Discriminator training
discriminator_real_loss = Discriminator(real_image, fake=False)
discriminator_fake_loss = Discriminator(fake_image.detach(), fake=True)

discriminator_loss = discriminator_real_loss + discriminator_fake_loss

# Generator training
generator_disc_loss = Discriminator(fake_image, fake=True)
masked_loss = grayscale_masked_loss(fake_image, CAT_image, CAT_mask)
generator_pvnet_loss = pvnet_fake_loss

generator_loss = generator_disc_loss + masked_loss + generator_pvnet_loss
```



(a) Sketch of the End-to-End network

6.2 Evaluation metrics

The numeric evaluation of the End-to-End network is performed using the following standard Object Pose Estimation metrics:

- **Average Distance of Displacements**: The ADD measures the mean distance between transformed model points based on the estimated and ground truth poses. A pose is considered correct when the computed distance is below 10% of the model's diameter.
- **Average Precision**: AP computes the area under the precision-recall curve.
- **Correctness with a 5-Pixel Threshold**: CMD5 is a measure indicating the correctness of pose estimation when the computed distance between 2D projections of model points for estimated and ground truth poses is less than 5 pixels.
- **2D Projection**: PROJ2D measures the mean distance between the 2D projections of model points for estimated and ground truth poses, considering a correct pose if the distance is less than 5 pixels.

6.3 Comparing to the conventional approach

When training Object Pose Estimation models, a common practice involves to create a diverse dataset superimposing 3D models of objects in various positions, rotations, and lighting conditions onto random backgrounds. It's important to note that achieving perfect blending of these 3D models with background images can be challenging, particularly in terms of ensuring consistent lighting and perspective alignment. The PVNet module of the End-to-End network will be compared to PVNet trained with the same Cat's models, but superimposed into random backgrounds.

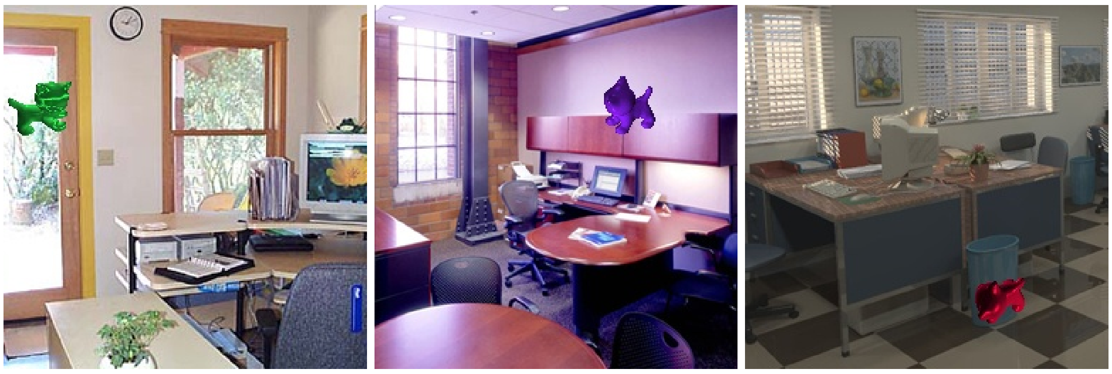


Figure 6.2: Images from the superimposition dataset

6.4 Results

Looking at Figure 6.4, the first observation is that the training converges, with PatchGAN still performing better in blending the Cat model into the background. Furthermore, introducing the PVNet loss to the Generator made the Cat's model much more distinguishable from the background.

Moreover, also from the standpoint of Object pose estimation results are at least acceptable. As illustrated in Figure 6.3, metrics show comparable performance among the different models, achieving the same top-notch results in both PROJ2D and AP. However, it's worth noting that both GAN models performed less optimally in ADD and CMD5 metrics. This can be attributed to the early training phases of the Generative Models, where PVNet encounters suboptimal images due to the limited capabilities of the GAN network at that stage. Consequently, PVNet undergoes to fewer effective training iterations than the model relying on superimposed images.

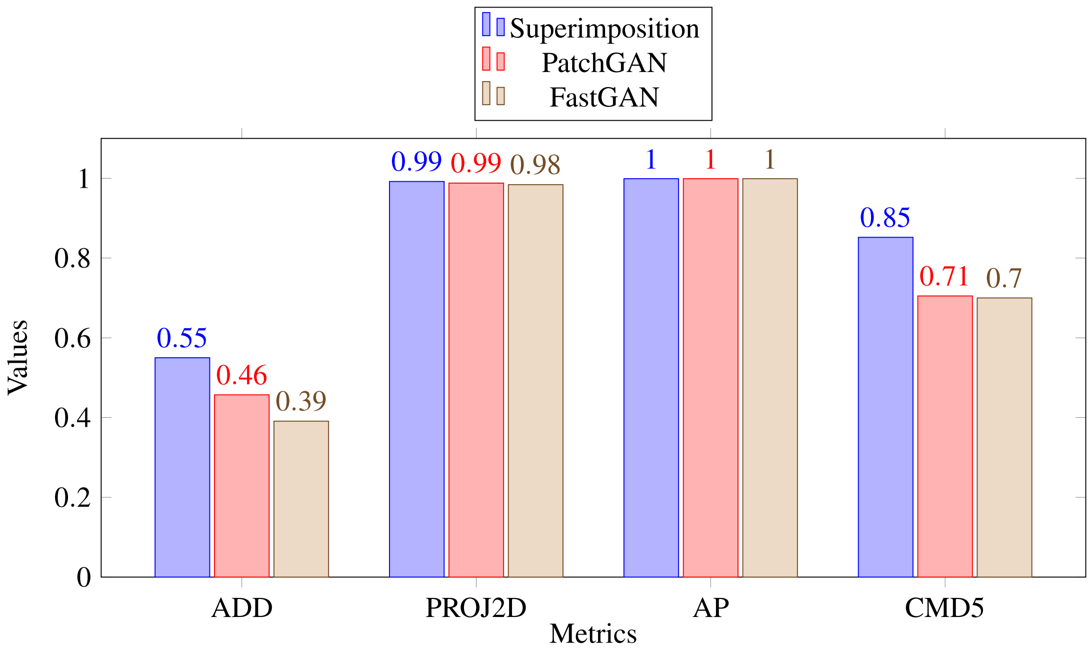
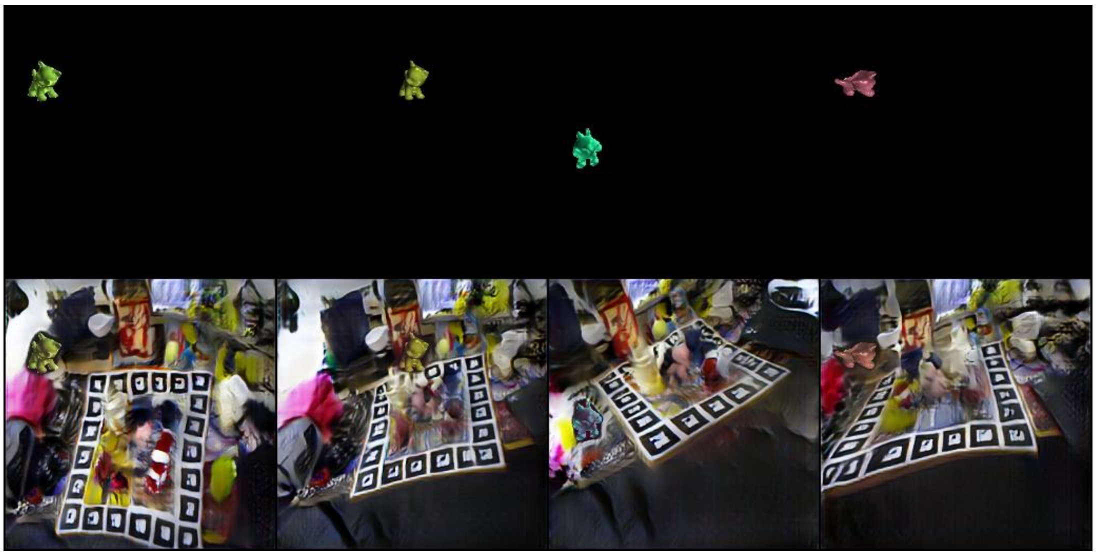


Figure 6.3: Metrics evaluation on end-to-end models


(a) FastGAN

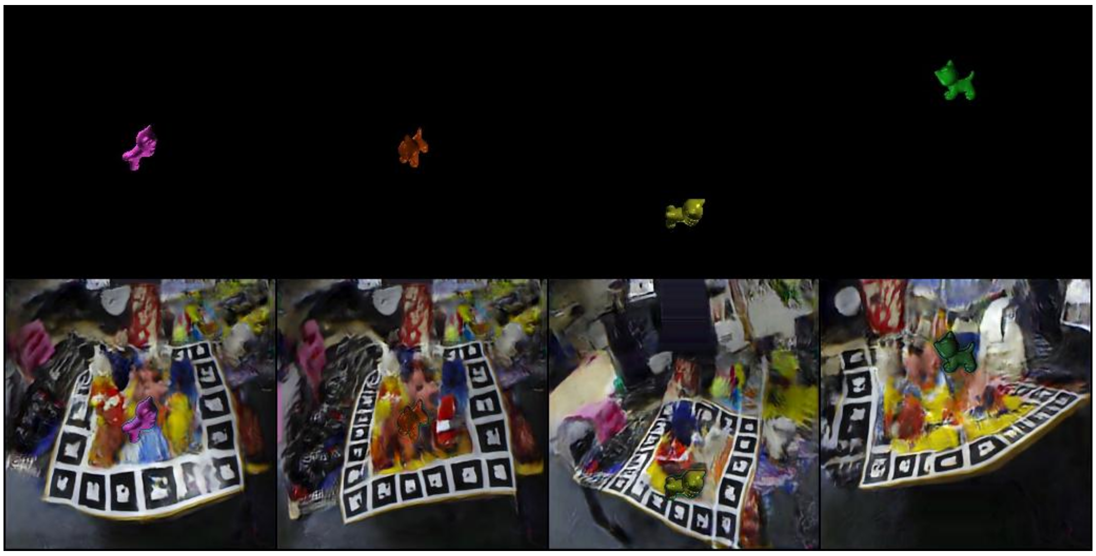

(b) PatchGAN

Figure 6.4: Samples form Generative Models trained End-to-End with PVNet

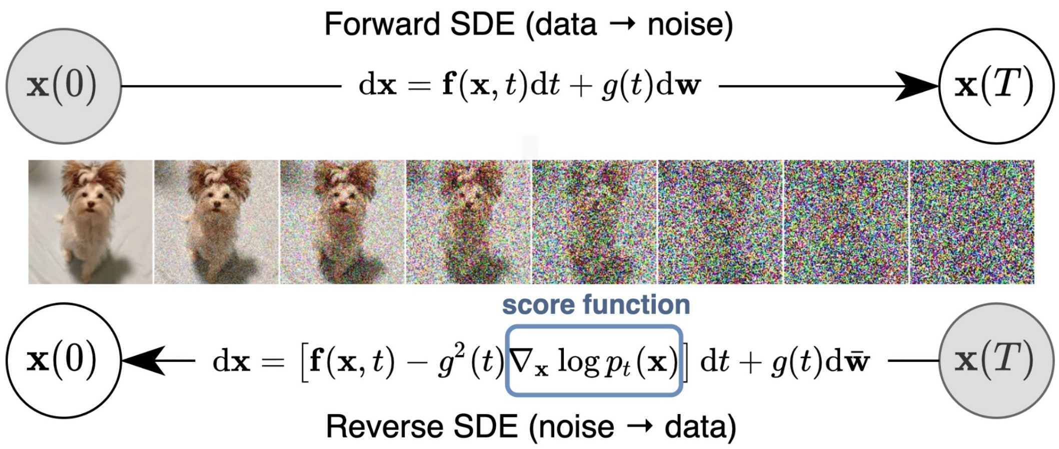
Chapter 7

Diffusion Models

Towards the conclusion of this thesis, it is noteworthy to introduce a novel category of generative models known as Diffusion Models.

7.1 How diffusion models work

Fundamentally, they operate by gradually introducing Gaussian noise to the training data, destroying them, and then learning how to reverse the process recovering the data from the noise. Once the training is complete, the Diffusion Model can be used to produce new data from random noise.



Forward SDE (data → noise)
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$
$\mathbf{x}(T)$
score function
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}\right]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$
$\mathbf{x}(T)$
Reverse SDE (noise → data)

Figure 7.1: Stable diffusion process

Diffusion models have been proven [7] to be at least as powerful as GANs in generating images, with the clear advantage on mode coverage and diversity (Figure 7.2), which is indeed a valuable property in the field of Object pose estimation.

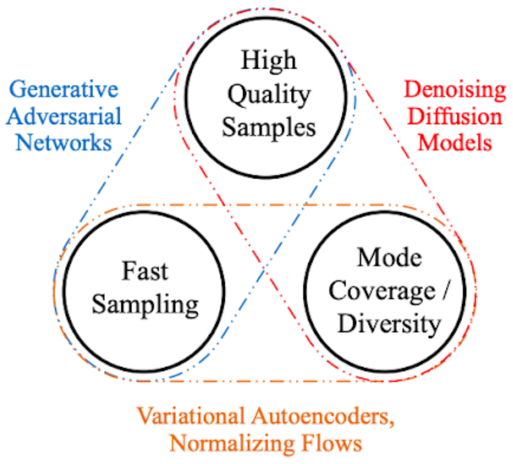


Figure 7.2: Strong and weak points of Generative models

In this chapter will be presented some key ideas that could inspire further researches on the topic of combining object pose estimation with generative models.

7.2 Repaint

RePaint is an inpainting method based on Denoising Diffusion Probabilistic Models (DDPM). RePaint excels at handling arbitrary binary masks, providing high-quality and diverse image inpainting. It outperforms existing approaches, including Autoregressive and GAN methods, in various inpainting scenarios, making it a robust solution for free-form inpainting tasks.

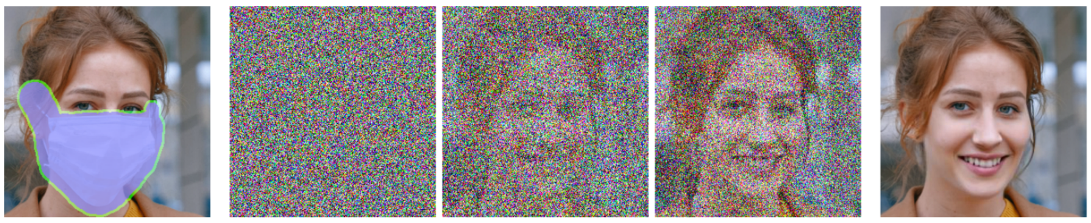


Figure 7.3: Example of how RePaint works

7.2.1 Application to LINEMOD

This section acts as a proof of concept, demonstrating how a model like RePaint could potentially be leveraged to generate new samples for the LINEMOD dataset. It's important to note that it has been employed a pretrained model on ImageNet [8], and the results obtained here may not be directly applicable to the specific objectives of this thesis. Nevertheless, they offer valuable insights into the potential utility of such a model in this context.

Figure 7.4 displays the results of applying RePaint to images featuring a random background, a circular mask of random size, and Cat within it. It is possible to observe how RePaint generates an image that attempts to seamlessly blend Cat with the background, exhibiting good results in terms of variety and background reconstruction. However, as previously mentioned, the pretrained model lacks prior knowledge of Cat, leading to less-than-ideal outcomes, such as a black halo around the Cat's model or the occasional hallucination of a dog. This last issue is a known problem attributed to a bias in the ImageNet dataset.

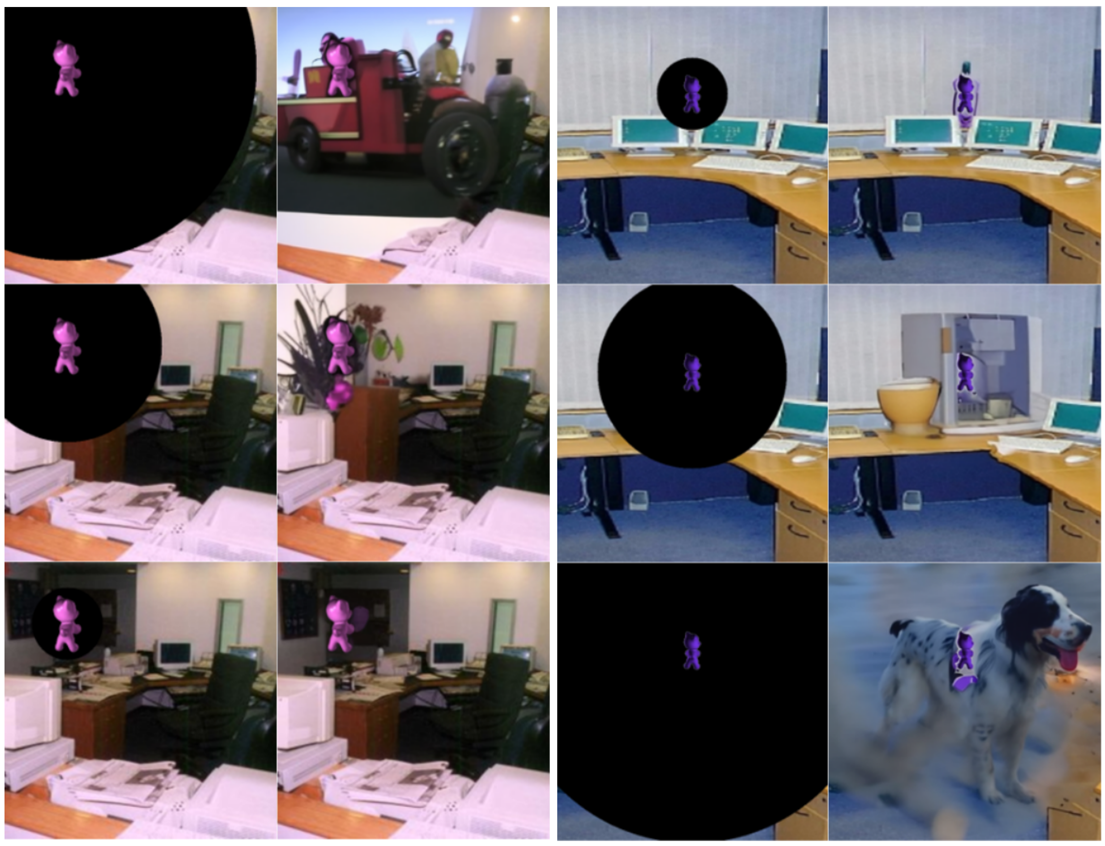


Figure 7.4: Utilizing RePaint with Cat and random backgrounds

Chapter 8

Conclusions

In this thesis, various GAN models were employed to generate datasets for Object pose estimation. Specifically, the goal was to create images containing pre-rendered models, serving as ground truth for training models such as PoseCNN or, in this case, PVNet. The input model used for all experiments was the Cat's model from LINEMOD.

8.1 Image-to-Image Translation

Chapter 4 introduced essential Image-to-Image Translation models suitable for domain adaptation, where images from Domain A are transformed to resemble those from Domain B.

This functionality ca be leveraged to generate datasets of images in environments where data acquisition is particularly challenging. In this thesis, the CUT model was utilized to generate underwater images by providing it with three types of input:

- Cat with a black background from Domain A and underwater images from Domain B.
- Cat with a random image background and underwater images from Domain B.
- Cat with a random image background and underwater images from Domain B, also containing the Cat's model.

The results demonstrated that achieving optimal performance requires the use of a background from Domain A; otherwise, CUT may attempt to maintain a uniform background in the adapted image. Furthermore, superimposing the Cat's model in Domain B results in undesirable distortions of the model.

8.2 Image Synthesis

The experiments in Chapter 5 aimed to explore ways to expand an existing dataset, using different generative architectures.

8.2.1 Building a Model from Scratch

In Section 5.3 experiments began with DCGAN, in order to gain a preliminary understanding of the main challenges to overcome. The generated images exhibited blurriness, Cat was not clearly visible, and mode collapse was evident.

Model changes and relative improvement included:

- U-Net generator coupled with L1 loss to help to preserve the details of the input image in the output.
- Patch discriminators to improve the model's stability and to introduce some spatial information between objects and the background.
- Self-Attention layers to further enhance the spatial relations between objects, noticeable even in the early stages of training.

All of these mechanisms helped the model generate satisfactory images in the end.

8.2.2 FastGAN

In Section 5.4 another approach was adopted: an already performing model was trained on LINEMOD images. FastGAN was selected for this task because it was specifically designed to perform well with small datasets, which aligns with the focus of this thesis.

The results showed a generally high quality of the samples, but Cat didn't appear clearly in the output, so two further experiments were conducted to introduce it into the final image. Firstly, Cat was superimposed directly in the output, but this approach made the pipeline unstable. The changes were made to the FastGAN model in order to provide information of the Cat's model in input. This involved using an encoder linked to the FastGAN generator at the early stages of the upsampling, and then summing the input Cat with the output of the network. Results showed that this changes ensured the presence of the Cat in the final image, although the network struggled to capture the spatial relationship between Cat and the background.

8.2.3 End-to-End Approach

The best-performing models from previous experiments have been integrated into an end-to-end network that concurrently trains:

- A GAN network that generates samples resembling the LINEMOD dataset starting from a rendered Cat.
- PVNet, a 6DoF Object pose estimation model.

The pipeline has demonstrated to be stable, with images showing a more distinguishable Cat from the background. What's particularly noteworthy is that in all cases PVNet performed similarly to the model trained with the more traditional technique of superimposing the object into random backgrounds. Some metrics fell slightly below expectations, but this may be attributed to the poor quality of images provided to PVNet during the early stages of GAN training. This means that this pipeline can eliminate the need for manual labor in collecting backgrounds and creating customized datasets.

8.2.4 Diffusion Models

Diffusion Models are an innovative framework that has proven to be very promising in terms of generating high-quality samples and achieving mode coverage. The main idea is to progressively degrade an image with random noise and then train a network to reverse this process.

An example of how these networks can be applied in Object Pose Estimation is by starting with an image of the rendered object and solving an inpainting problem with RePaint. This network can be employed to fill in missing or damaged portions of an image, so, by considering the mask of the object as the part of the image to retain and allowing RePaint to handle the rest, it is possible to generate new samples containing the original pose of the object.

The main achievement of this thesis work has been to demonstrate that GANs, and, more in general, Generative Models, represent a powerful tool for dataset generation, and can boost the performance of Object Pose Estimation models to train on larger dataset and consequently increase precision and accuracy.

Bibliography

[1] S. Hinterstoisser, V. Lepetit, S. Ilic, *et al.*, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Computer Vision ACCV 2012*, K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, Eds., vol. 7724, Springer Berlin Heidelberg, 2012, pp. 548–562.

[2] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[3] S. Peng, Y. Liu, Q. Huang, H. Bao, and X. Zhou, *Pvnet: Pixel-wise voting network for 6dof pose estimation*, 2018. arXiv: `1812.11788 [cs.CV]`.

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, *Generative adversarial networks*, 2014. arXiv: `1406.2661 [stat.ML]`.

[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, *Image-to-image translation with conditional adversarial networks*, 2018. arXiv: `1611.07004 [cs.CV]`.

[6] B. Liu, Y. Zhu, K. Song, and A. Elgammal, *Towards faster and stabilized gan training for high-fidelity few-shot image synthesis*, 2021. arXiv: `2101.04775 [cs.CV]`.

[7] P. Dhariwal and A. Nichol, *Diffusion models beat gans on image synthesis*, 2021. arXiv: `2105.05233 [cs.LG]`.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. DOI: `10.1109/CVPR.2009.5206848`.

[9] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, *Contrastive learning for unpaired image-to-image translation*, 2020. arXiv: `2007.15651 [cs.CV]`.

[10] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, *Unpaired image-to-image translation using cycle-consistent adversarial networks*, 2020. arXiv: `1703.10593 [cs.CV]`.

[11] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, *Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes*, 2018. arXiv: 1711.00199 [cs.CV].

[12] J. Ho, A. Jain, and P. Abbeel, *Denoising diffusion probabilistic models*, 2020. arXiv: 2006.11239 [cs.LG].

[13] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. V. Gool, *Repaint: Inpainting using denoising diffusion probabilistic models*, 2022. arXiv: 2201.09865 [cs.CV].