Generation of Realistic Images for Learning in Simulation using FeatureGAN

Nicolás Cruz
Dept. of Electrical Engineering
Universidad de Chile, Chile
nicolascruz2187@gmail.com

Javier Ruiz-del-Solar
Dept. of Electrical Engineering and the Advanced Mining Technology Center
Universidad de Chile, Chile
jruizd@ing.uchile.cl

Abstract: This paper presents FeatureGan, a methodology to train image translators (generators) using an unpaired image training set. FeatureGan is based on the use of Generative Adversarial Networks (GAN) and has three main novel components: (i) the use of a feature loss to ensure alignment between the input and the generated image, (ii) the use of a feature pyramid discriminator, which uses a tensor composed of features at different levels of abstraction generated by a pre-trained network, and (iii) the introduction of a per class loss to improve the results in the simulation-to-reality task. The main advantage of the proposed methodology when compared to classical approaches is a more stable training process, which includes a higher resilience to common GAN problems such as mode collapse, as well as better and more consistent results. FeatureGan is also fast to train, easy to replicate, and especially suited to be used in simulation-to-reality applications where the generated realistic images allow to close the visual simulation-to-reality gap. As a proof of concept, we show the application of the proposed methodology in soccer robotics, where realistic images are generated in a soccer robotics simulator, and robot and ball detectors are trained using these images and then tested in reality. The same methodology is used to generate realistic images from images rendered in a video game. The realistic images are then used to train a semantic segmentation network.

Keywords: Sim-to-Real, Learning in Simulation, Image-to-image translation, GAN, Unaligned

1 Introduction

Simulators are fundamental tools to develop robotic applications since they provide an environment to create and test algorithms without the need of using a real robot. However, developing solutions in simulation is problematic, since there is a mismatch between the samples collected in simulation and those collected in a real environment. This mismatch is commonly referred to as the simulation-reality gap and usually results in sub-optimal operation for algorithms developed in simulation when deployed in real-world conditions. Furthermore, if an algorithm is properly tuned to work in a real environment, for example, by training it with real data, then its behavior cannot be accurately tested in the simulator, because the performance of the algorithm would be vastly different in a simulated environment compared to a real environment, given the almost noise-free nature of simulators.

Recently, several methodologies have been proposed to bridge the gap between reality and simulation [1][2], most notably in the image processing area. Among these approaches, the ones based on generative models [3][4] usually require large training sets of aligned real-simulated samples. Given an image *A* in the Dom_A domain and another image *B* in the Dom_B domain, we call the pair of images (A, B) aligned if both images share a common scene layout. Achieving large datasets of aligned images is very time consuming and requires a lot of manual labeling. To account for this, methodologies using active learning [5] have been proposed in order to try to automate the process of data labeling. These methodologies are usually convoluted, still require some manual labeling and tend to achieve sub-optimal results. Architectures based on cycle consistency, such as Zhu et al. [6], allow to train a generator using a dataset of unaligned images to infer an image $\hat{A}$ from an image B such that the pair ($\hat{A}$, B) is aligned. However, we found that architectures based on cycle consistency do not perform well in the simulation-to-reality image translation task when confronted with simple environments.

As an alternative, we propose FeatureGan, which is trained using unaligned images in an unsupervised manner and achieves consistently good results in the simulation-to-reality image translation task. During training, FeatureGan uses a feature loss function to ensure that the generator produces an output image that is aligned to the input image. A classic GAN loss function is also used to steer the generator into producing images that are part of the target domain. To further improve the quality of the generated images, a class-based feature loss function is developed, which takes into account the ground truth information provided by the simulator.

As a proof of concept, we test our methodology in the soccer robotics domain, specifically in the RoboCup Standard Platform League (SPL) [7]. In the SPL, a large part of the development of the required vision, self-localization and strategy modules are done in simulation, since the robots are fragile and therefore are used only when strictly necessary. The simulators used in the SPL (e.g., SimRobot [8]) suffers from the simulation-to-reality gap. We show that by using the proposed methodology, we significantly reduce this simulation-reality gap, allowing to generate realistic images in simulation, and then to train and test object detectors directly in simulation. We compared FeatureGan to the standard CycleGan implementation [6] and found that FeatureGAN produces less artifacts in the generated images and better textures, which is very relevant for closing the visual simulation-to-reality gap. Additionally, we test the proposed methodology in the GTA to Cityscapes problem to prove that the proposed approach can be used in a wide range of applications.

The main contributions of this paper are the following: First, a new approach to ensure the alignment between the input image and the corresponding generated image based on a feature loss. Second, the introduction of a feature pyramid discriminator which uses as input a feature tensor composed of several concatenated feature maps of different abstraction levels obtained using a pre-trained network. Third, the introduction of a per class loss for the simulation-to-reality task. Videos, images and the code can be found at https://nicolascruzw21.github.io/FeatureGan/.

2 Related work

The use of simulators to develop algorithms is a standard practice in the robotics community. Recently, simulators have also been used to train and test machine learning algorithms [9][1][2][10][11][12][13], which are then deployed to the robot to operate in real world conditions. One of the main challenges of this approach is the simulation-reality gap: Given the mismatch between samples obtained from simulation and samples obtained in real conditions, algorithms which were developed in simulation tend to underperform in real environments. Furthermore, this also means that simulation is not a representative testing environment for algorithms developed using real data.

Several approaches have been proposed to reduce the gap between the development environment (simulation) and the operational environment (reality). Such techniques usually involve domain transfer to generate realistic samples in simulation [14][1] or projecting both simulated and real samples to an intermediate domain [2][15]. In the case of image samples, this domain transfer is known as the image to image translation task, which has been a constant subject of research in the computational vision community. Isola et al. [4] proposed an unsupervised architecture based on GAN networks [16] that is able to translate an image from an input domain Dom_B to a target domain Dom_A. Chen and Koltun [3] reported that a supervised approach based on a feature loss was able to achieve state of the art results in the image translation task. Both approaches require aligned image pairs to train the generator network. Since usually large databases are required to train a good generator, and given the complexity of the sample collection by real robots, methodologies such as Cruz and Ruiz-del-Solar [5] have been proposed to simplify this process. Zhu et al. [6] proposed in

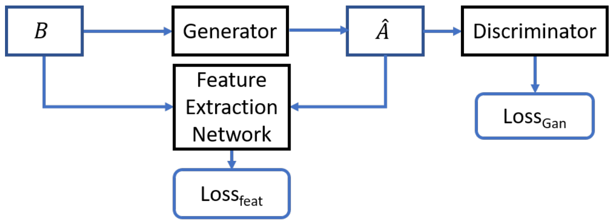


Figure 1: Pipeline used to train the Generator. The generator takes an image *B* in the domain *Dom*$_B$ as input and outputs a generated image $\hat{A}$ in the Dom$_A$ domain.

CycleGan, training a generator using unaligned images and used a cycle consistency loss to preserve the layout of the input image across the different domains in the cycle. However, when applied in a soccer robotics application we found that the artifacts produced by CycleGan resulted in a large number of false positives for the robot's detection algorithms, which makes it difficult to use to generate realistic simulated environments.

3 Methodology

We propose FeatureGan, a methodology to train a generator using unaligned images belonging to two different domains Dom$_B$ and Dom$_A$. Once the generator is trained, it can be used to perform aligned image to image translation from Dom$_B$ to Dom$_A$.

During training, FeatureGan makes use of three neural networks (see Fig.1). The generator network *G* is used to perform the domain transfer by taking an input image *B* in the *Dom*$_B$ domain and outputting a generated image $\hat{A}$ in the Dom$_A$ domain. The loss function *Loss*$_{GAN}$ scores the quality of the generated image by checking how close it is to the Dom$_A$ domain via a discriminator network *D*.

To preserve the layout of the input image *B* at the output image $\hat{A}$ we use a third network, the *Feature Extraction Network*, to calculate a feature loss Loss$_{feat}$ between B and $\hat{A}$. If the Dom$_B$ and Dom$_A$ domains are similar enough, and given that image classification CNNs are invariant to illumination changes as well as to small high frequency features changes in the input image, then, the Loss$_{feat}$ should be minimized when B and $\hat{A}$ are aligned. We combine both the Loss$_{GAN}$ and the Loss$_{feat}$ into a single loss function, and use it to train the generator G to produce realistic images in the Dom$_A$ domain, while preserving the geometric layout of the input image B. To further improve the quality of the results, the ground truth information from the simulator is used to introduce labels into the training process, by adding them as extra channels to the generator during training.

3.1 The Generator

The generator is used for domain transfer: from an input image B in the simulated domain it generates an output image $\hat{A}$ in the real domain. In this work we tested several base architectures for the generator including Resnet Generator [17], Cascade refinement network [3] and U-net [18]. We found that U-net achieved the best overall results in the sim-to-real task. We further improve the performance of the generator by introducing class labels to the input as one hot-encoded vectors. To achieve this, we use the ground truth information of the simulator to separate each of the classes in the input image B, into different additional channels. This allows the generator to have simultaneous access to the class and texture information in B. The generator is trained using Loss$_{\text{gen}}$, which is built using the feature loss and the GAN loss described by equations 4 and 5:

$$\text{Loss}_{\text{gen}} = \sum_{j=1}^{J} (\text{Loss}_{\text{feat-j}}) + \sum_{i=1}^{I} (\lambda_i \times \text{Loss}_{\text{GAN-i}}) \tag{1}$$

with J the number of classes and I the number of resolutions/discriminators.

3.2 Feature Extraction Network

The Loss$_{feat}$ is used measure the similarity of the layout between the input image and the output image. Similar approaches have been used in the style transfer task [19] and conditional image synthesis task [3]. Given a *Feature extraction network*, which was pre-trained on some database, in this case ImageNet [20], we first feed the image B to the feature extraction network, and record the resulting activation maps $Map_l(B)$ at different layers of the network. We then feed the corresponding generated image $\hat{A} = G(B)$ to the feature extraction network and record the activation maps $Map_l(\hat{A})$, at the same layers. Supposing that the D_A and D_B domains are similar, then if $\hat{A}$ and B are aligned, the corresponding activation maps $Map_l(B)$ and $Map_l(\hat{A})$ should be similar. On the other hand, if $\hat{A}$ and B are not aligned the difference between the corresponding activation maps should increase. Since classification CNN networks are trained to be invariant to changes in illumination, then it can be assumed that the illumination information is lost as the data flows through the network. It follows that the activation maps of two aligned images with vastly different illuminations should be approximately equal for the deep layers of the network. Our network of choice for feature extraction, a modified VGG-19 [21], includes average pooling as well as strided convolutions. These operations result in a loss of high-resolution features of the input image which translates on a loss of details. The relative position of the objects in the scene is preserved by using coarse coding [22]. Since the high-resolution features of the input images are lost in the deeper layers of the network, the generator G will be able to add small details to the generated image without any mayor increase in Loss$_{feat}$.

Finally, we introduce a per class feature loss to have an extra degree of control over the results of the generator. Since the ground truth for each B image is provided by the simulator, then this information can be used to separate both B and $\hat{A}$ into different images B_j and $\hat{A}_j$, with $j = 1...J$, and J the number of classes in B:

$$\begin{cases} \text{pix}(B_j) = \text{pix}(B) & if\ \text{class}(pix(B)) = j \\ \text{pix}(B_j) = 0 & \text{otherwise} \end{cases} \tag{2}$$

$$\begin{cases} \text{pix}(\hat{A}_j) = \text{pix}(\hat{A}) & if\ \text{class}(pix(B)) = j \\ \text{pix}(\hat{A}_j) = 0 & \text{otherwise} \end{cases} \tag{3}$$

Then, the feature loss for class j is defined as:

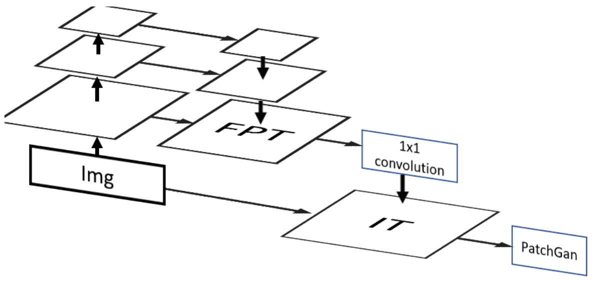
$$\text{Loss}_{\text{feat-j}} = \alpha_j \sum_{l=1}^{L} (\text{MSE}(\text{Map}_l(B_j), \text{Map}_{l,j}(\hat{A}_j)) \tag{4}$$

Where α_j is a user tunable parameter which measures how similar an object of class j in $\hat{A}_j$ must be to the corresponding object in B, and L represents the number of activation maps. By relaxing this parameter for a particular class, the generator will be able to introduce more changes to the class, but the preservation of the geometric layout constraint will also be relaxed. We found that results further improve when applying layer normalization [23] to $\hat{A}$ and B before feeding them to the *Feature Extraction network*.

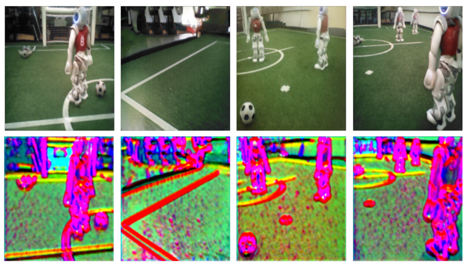
3.3 A Feature Pyramid Based Discriminator Approach

We use an approach consisting of multiple PatchGan discriminators [24] operating at different scales over a concatenation of a feature pyramid tensor and the input image. A traditional PatchGan discriminator, as proposed by Isola et al. [24], is composed of a fully convolutional network with a narrow receptive field (usually 70 $\times$ 70), which extracts local features from an input image (Img) to form a vector of predictions. The small receptive field improves generalization which results in a more stable training, since regions of the image are classified independently, meaning that the generator is still rewarded for small local improvements in image quality.

In traditional neural network design, higher abstraction features are provided by the deeper layers in the neural network, while high resolution features come from the layers closer to the input. We



a)



b)

Figure 2: a) Proposed FPN discriminator. b) First row: Input-Image. Second row: compressed FPT tensor with N = 3.

propose an improved version of PatchGan based on the idea behind Feature Pyramid Networks (FPN), which was introduced by Lin et al. [25] to improve the performance of image detectors and semantic segmentation networks. The FPN approach consists of upsampling the deeper feature maps generated by a CNN. The upsampled features are then combined with feature maps generated at layers closer to the input of the network.

For our approach, shown in Fig. 2 a), we chose to fed Img to a VGG-19 network which was pre-trained in ImageNet [20], and to extract the feature maps at different depths in the network. These feature maps are then concatenated into a single Feature Pyramid Tensor (FPT), which is then compressed into N channels using a 1x1 convolution, and upsampled to match the height and width dimensions of Img. As an example, the resulting compressed FPT from the 8th and 12th layers for the sim-to-real image translation task is shown in Fig. 2 b). From these images, it is apparent that the compressed features give the discriminator important information on the shape and position of the relevant objects in the scene, which have a high contrast with the background information. Finally, the compressed FPT is concatenated with Img to form the final input tensor IT to the PatchGan discriminator. Using IT as input to the discriminator offers several advantages. First, the features of the IT come from layers of different depth, with different receptive fields, meaning that the discriminator has access to spatially local and global features simultaneously. Second, the discriminator still works by classifying small regions and has access to the extracted feature maps as well as to the original image, therefore, its ability to generate very realistic textures is preserved. Third, in addition to texture information, each region classified by the discriminator has also access to high level features. By using both simultaneously, the discriminator can achieve better performance at classifying each individual region. Most importantly, the features extracted by the pre-trained network are stable through the training process and are independent of the quality of the images inferred by the generator. This might help to avoid overfitting of the discriminator to the current state of the generator, and to stabilize the training process. Furthermore, since the VGG-19 network has no tunable parameters, the relative learning capacity between generator and discriminator is maintained.

Isola et al. [24] found that small patch sizes lead to an increase level of artifacts in the image, which can be attributed to the lack of spatial awareness of a small patch size discriminator. We also found that discriminators with larger patch sizes lead to spatial consistency in terms of illumination and object placement. Following the guidelines of Wang et al. [26], we construct several discriminators to evaluate the input tensor at different resolutions. Each discriminator is run over the IT with a different level of downsampling. The combination of the loss Loss$_{GAN-i}$ provided by each discriminator i results in the total Loss$_{GAN}$ shown in eq. 5:

$$\text{Loss}_{\text{GAN}} = \sum_{i=1}^{I} \lambda_i \times \text{Loss}_{\text{GAN-i}} \tag{5}$$

where λ_i is a parameter which indicates the relative importance of the i-th discriminator, and I is the number of resolutions.

Heusel et al. [27] showed that GANs trained by using different learning rates for the generator and the discriminator converge to a local Nash equilibrium. Following this guideline, we selected a learning rate for each discriminator equal to two times the learning rate of the generator. We also use one-sided label-smoothing to achieve smoother probabilities. Finally, to prevent model oscillation we keep an historic record pool of generated images as suggested by [28] and [6]. We randomly sample from this pool to train the discriminator.

4 Results

4.1 Sim-to-Real Image Translation in Soccer Robotics

As a proof of concept, we apply FeatureGAN for sim-to-real image translation in soccer robotics. FeatureGan is used to increase the realism of the images rendered by the SimRobot simulator [8]. In this task, FeatureGAN is compared to a Cascade Refinement Network (CRN) trained using the supervised approach presented by Cruz and Ruiz-del-Solar [5] and CycleGan [6]. The training procedures of the three models are specified in the appendix material.

Fig. 3 shows an image of the SimRobot simulator and the corresponding realistic images $\hat{A}$ generated by FeatureGan, CycleGan and CRN. Additional images can be found in the additional material section. Most of the artifacts generated by CycleGan are the result of a bias in the network, meaning that they remain mainly static across different input samples and indicate a mode collapse of the GAN. After further examination, we concluded that CycleGan artifacts were the results of a mode collapse originating from a problem with the network's architecture for the specific case of simulation-to-reality rather than of bad training parameters.

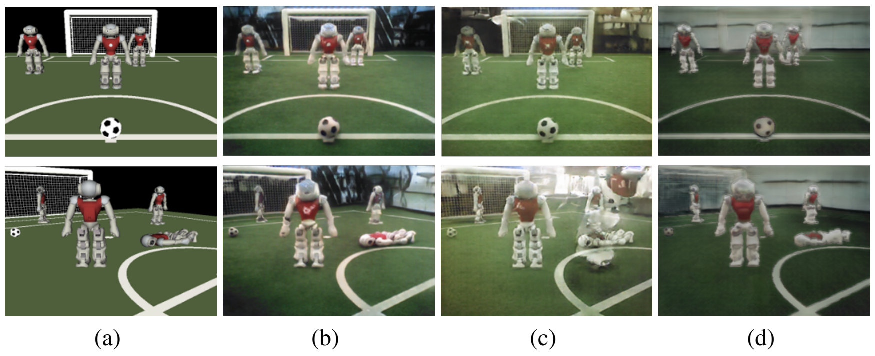


Figure 3: a) rendered images from simulation, b) aligned images generated with FeatureGan, c) aligned images generated with CycleGan, d) aligned images generated with CRN.

During our participation in the RoboCup SPL our team developed some of the first CNN based robot and ball detectors [29] [30]. The architectures for the CNNs can be found in [30] and the design principles are further expanded on [29]. Training these models involves creating a manually annotated database which is time consuming.

To validate the quality of the generated images, we trained a robot detector and a ball detector using five different sets of images: (1) real images collected using a real robot in a real field, (2) simulated images rendered directly by SimRobot, (3) images generated using FeautureGAN, taking as input the SimRobot images, (4) images generated using the standard implementation of CycleGan [6], taking as input the SimRobot images and (5) images generated using a Cascade Refinement Networks (CRN) trained using the methodology presented by Cruz and Ruiz-del-Solar [5], taking as input the SimRobot images. All trained detectors (models) were tested using a test database composed purely of real samples collected using the real robot: 760 images for the ball detector and 960 images for the robot detector.

Table 1: Robot and ball models trained in different datasets and evaluated using real data.

Testing data type	Robot detector accuracy (%)	Ball detector accuracy (%)
Real images	90.7	98.2
Images generated using FeatureGan	89.3	95.4
Images generated using CycleGan	87.4	95.1
Images generated using CRN	83.7	95.3
Rendered images from the simulator	66.4	51.4

Table 1 shows the obtained results. Models trained using real samples achieve very high accuracy. Models trained using images generated by FeatureGan result in metrics that are similar to those achieved by the models trained with real data. CRN also shows a good performance while detecting the ball, but it decreases for the robot detection case. CycleGan has a slightly lower performance than our method, even while producing numerous artifacts in our tests. This is simply explained by the nature of the artifacts, which are a result of a bias in the generator network and by consequence have very little diversity. In practice, these samples are almost uni-modal and thus the classifier network can learn to classify the artifacts as outliers. In contrast, models trained using samples collected from the classical SimRobot simulator and tested with real data achieve close to random results for the ball classifier model while the robot classifier model achieves 24.3% less accuracy than the corresponding model trained using real samples. From these results, it is clear that FeatureGAN, CRN and CycleGan are bridging the gap between reality and simulation, and that best results are obtained by FeatureGAN. The use of these methods allows to fully train algorithms in simulation, which can then be deployed to a real environment with only a marginal loss in performance.

4.1.1 Real-to-sim Task

Robot simulators can serve as tools to evaluate the performance of robotics algorithms by offering a testing environment in which they can run under realistic conditions. To validate this idea, robot and ball detectors were trained using images of the real-world, and then evaluated in five different environments: (1) real images collected using a real robot in a real field, (2) simulated images rendered directly by SimRobot, (3) images generated using FeautureGAN, (4) images generated using the CycleGan, and (5) images generated using CRN. Table 2 show the obtained results. It can be observed that simulated environments generated using FeatureGan and CRN allow to close the visual simulation-to-reality gap. The main advantage of FeatureGAN over CRN is the use of unpaired, unlabeled images to train the network. In the case of CycleGan, the robot detector tested using the generated images presented a notable decrease of 14.2% in accuracy, when compared to its accuracy in a real environment. We found this to be a direct consequence of the artifacts produced by the generator, which translate in a lot of false positive for the Robot classifier. Finally, testing in the classic rendered SimRobot environment resulted in sub-optimal performance. Surprisingly, the ball model achieved good performance in all testing environments. We attribute this to the simplicity of the problem.

Table 2: Robot and ball models trained with real data and evaluated in different datasets.

Testing data type	Robot detector accuracy (%)	Ball detector accuracy (%)
Real images	90.7	98.2
images generated using FeatureGan	95.2	98.7
Images generated using CycleGan	76.5	94.3
Images generated using CRN	95.7	98.8
Rendered images from the simulator	66.2	97.0

4.2 Sim-to-Real Image Translation in CityScapes

This section examines if using FeatureGan can improve the graphics quality of videogames. In addition to the standard implementation of FeatureGan we also train versions without the feature pyramid network and using a regular feature loss rather than the proposed class-based feature loss. The training parameters can be found in the annexed material. We use images from Grand Theft

Auto 5 (GTA) [31] as the input domain Dom$_B$ and images from CityScapes Cordts et al. [32] as the target domain Dom$_A$. An image of the Dom$_B$ domain alongside the corresponding realistic image generated by FeatureGan is presented in Fig. 4. A realistic image generated by the FeatureGan methodology without using the class-based feature loss is also shown. Additional images can be found in the supplementary material. The results of FeatureGan without using a feature pyramid network are not included since the training process consistently diverged. We believe that convergence is easier to achieve with a feature pyramid discriminator since similar objects from different domains will be closer in the feature space provided by the deeper layers of the network.

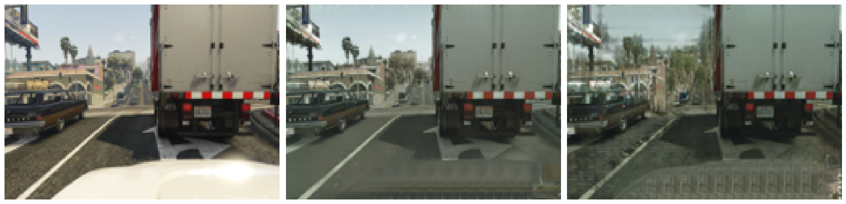

(a) (b) (c)

Figure 4: a) GTA image, b) aligned image generated with FeatureGan, c) aligned image generated with FeatureGan without using a class based feature loss.

We train a semantic segmentation network using these images to quantify the level of the reality gap of each of these databases. In particular we train a DeepLabv3+ with a mobileNet [33] backbone using four different sets of images: real images of a city from the CityScapes database, realistic images generated by the whole FeatureGan model and by FeatureGan with no class feature loss, and rendered images from the GTA database. The models were then evaluated on the CityScapes test dataset, composed of real samples.

Table 3: Semantic segmentation models trained in different datasets and tested using real data.

Training data type	Pixel accuracy (%)
CityScapes images	91.4
FeatureGan images	84.3
FeatureGan no class feature loss	80.6
GTA images	69.5

Table 3 show the results of this test. DeepLabv3+ achieves excellent results at the semantic segmentation task when trained and tested using the CityScapes dataset. However, the same architecture trained using synthetic data from the GTA dataset shows a significant reduction in performance when tested with real samples. Training with samples generated by FeatureGan results in a massive improvement which shows that FeatureGan is able to significantly reduce the gap between simulation and reality. Furthermore, we found that the training process of FeatureGan diverges when using a traditional PatchGan generator, while replacing the class based feature loss results in artifacts in the generated image (see Fig. 4).

5 Conclusion

In this work we presented FeatureGan, a methodology to train a generator to perform image-to-image translation that can be trained without the need of aligned images or manual labeling. We proved the validity of our approach in bridging the simulation-reality-gap of SimRobot, a simulator for the RoboCup SPL and of GTA, a popular video game. Compared to alternative approaches our method achieves high quality images, with minimal artifacts and is easy to train without requiring any form of manual labeling. FeatureGan introduces several improvements in terms of architecture when compared to other approaches, such as a feature pyramid discriminator, a loss function which integrates the ground truth of the simulator and an image alignment based on a feature loss. All this contributes to state of the art results in the simulation-to-reality image translation task.

Acknowledgments

This work was supported by the NVIDIA GPU program, and was partially funded by FONDECYT project 1201170 and CONICYT/PIA Project AFB180004.

References

[1] F. Leiva, K. Lobos-Tsunekawa, and J. Ruiz-del-Solar. Collision avoidance for indoor service robots through multimodal deep reinforcement learning. *RoboCup Symposium*, 2019.

[2] H. Bharadhwaj, Z. Wang, Y. Bengio, and L. Paull. A data-efficient framework for training and sim-to-real transfer of navigation policies. *CoRR*, abs/1810.04871, 2018. URL http://arxiv.org/abs/1810.04871.

[3] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. *CoRR*, abs/1707.09405, 2017. URL http://arxiv.org/abs/1707.09405.

[4] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016. URL http://arxiv.org/abs/1611.07004.

[5] N. Cruz and J. Ruiz-del-Solar. Closing the simulation-to-reality gap using generative neural networks: Training object detectors for soccer robotics in simulation as a case study. *IJCNN*, 2020.

[6] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017. URL http://arxiv.org/abs/1703.10593.

[7] RoboCup. Robocup standard platform league official website. https://spl.robocup.org/, 2020.

[8] T. Laue, K. Spiess, and T. Röfer. Simrobot – a general physical robot simulator and its application in robocup. volume 4020, pages 173–183, 07 2005. doi:10.1007/11780519_16.

[9] K. Lobos-Tsunekawa, F. Leiva, and J. Ruiz-del-Solar. Visual navigation for biped humanoid robots using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 3(4):3247–3254, Oct 2018. ISSN 2377-3766. doi:10.1109/LRA.2018.2851148.

[10] F. Zhang, J. Leitner, M. Milford, and P. Corke. Sim-to-real transfer of visuo-motor policies for reaching in clutter: Domain randomization and adaptation with modular networks. *CoRR*, abs/1709.05746, 2017. URL http://arxiv.org/abs/1709.05746.

[11] A. A. Rusu, M. Vecerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. Sim-to-real robot learning from pixels with progressive nets. *CoRR*, abs/1610.04286, 2016. URL http://arxiv.org/abs/1610.04286.

[12] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. *CoRR*, abs/1612.07828, 2016. URL http://arxiv.org/abs/1612.07828.

[13] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. *CoRR*, abs/1702.05464, 2017. URL http://arxiv.org/abs/1702.05464.

[14] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *CoRR*, abs/1709.07857, 2017. URL http://arxiv.org/abs/1709.07857.

[15] T. Inoue, S. Chaudhury, G. D. Magistris, and S. Dasgupta. Transfer learning from synthetic to real images using variational autoencoders for robotic applications. *CoRR*, abs/1709.06762, 2017. URL http://arxiv.org/abs/1709.06762.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[17] J. Johnson, A. Alahi, and F. Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016. URL http://arxiv.org/abs/1603.08155.

[18] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

[19] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015. URL http://arxiv.org/abs/1508.06576.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[22] G. E. Hinton, J. L. Mcclelland, and D. E. Rumelhart. Distributed representations. In D. E. Rumelhart and J. L. Mcclelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pages 77–109. MIT Press, Cambridge, MA, 1986.

[23] J. Ba, J. Kiros, and G. Hinton. Layer normalization. 07 2016.

[24] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016. URL http://arxiv.org/abs/1611.07004.

[25] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. URL http://arxiv.org/abs/1612.03144.

[26] T. Wang, M. Liu, J. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *CoRR*, abs/1711.11585, 2017. URL http://arxiv.org/abs/1711.11585.

[27] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017. URL http://arxiv.org/abs/1706.08500.

[28] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. *CoRR*, abs/1612.07828, 2016. URL http://arxiv.org/abs/1612.07828.

[29] N. Cruz, K. Lobos-Tsunekawa, and J. Ruiz-del-Solar. Using convolutional neural networks in robots with limited computational resources: Detecting NAO robots while playing soccer. *Lecture Notes in Computer Science*, 11175, (RoboCup Symposium), 2017, pp. 19-30.

[30] F. Leiva, N. Cruz, I. Bugueño, and J. Ruiz-del-Solar. Playing soccer without colors in the SPL: A convolutional neural network approach. *Lecture Notes in Computer Science*, 11374 (RoboCup Symposium), pp. 122-134., 2018.

[31] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016.

[32] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016. URL http://arxiv.org/abs/1604.01685.

[33] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL http://arxiv.org/abs/1704.04861.

6 Additional Material

6.1 Sim-to-Real Image Translation in Soccer Robotics

To train FeatureGan, an unpaired training set was used. The dataset is composed by 3,579 images collected using the robot in a real environment, and 4,027 images collected from simulation. The learning rate for the generator was set to 0.00005 and the learning rate for the discriminator was set to 0.0001. A class based $Feature - Loss$ was used, with the selected classes being robot, goal post, field line, shirt, ball and background. α_j was set to 6 for the field class, to 4 for the robot and goal post classes, 10 for the shirt class, 8 for the line class and 0 for the background class, meaning that the network is free to produce unaligned backgrounds since there is no background information in the simulator (the background is defined by a single constant value). Three feature pyramid discriminators with receptive fields over the input image of 70 $\times$ 70 (local), 140 $\times$ 140 (medium) and 280 $\times$ 280 (global) were used to evaluate the quality of the generated images at different scales. λ_i was set to $\frac{1}{4}$ for the local discriminator, $\frac{2}{4}$ for the medium discriminator and $\frac{1}{4}$ for the global discriminator. The network was trained for 80 epochs with a constant learning rate, and then for 150 epochs with linear decay at a resolution of 512 $\times$ 512 for three days in a Nvidia RTX 2080 ti.

The standard version of CycleGan, as presented by Zhu et al. [6] was trained on the same dataset as FeatureGan at a resolution of 512 $\times$ 512. A discriminator with a receptive field of 140 $\times$ 140 was used. The same generator and learning parameters were also employed to ensure a fair comparison. The identity loss was set to zero to diminish the blur of the generated images.

Finally, the Cascade Refinement Network (CRN) based on a supervised approach was trained by following the methodology proposed in [5]. The database consists of 2,000 pairs of aligned $(real, segmented)$ images of size 512$\times$512, which were obtained after performing domain randomization using 125 images of robots and 56 images of balls. The learning rate was set to a constant value of 0.0001.

Randomly selected pairs of simulated images B from the test dataset and the corresponding realistic images $\hat{A}$ generated by FeatureGan, the Cascade Refinement Network (CRN) and CycleGan [6] are shown in Fig. 5.

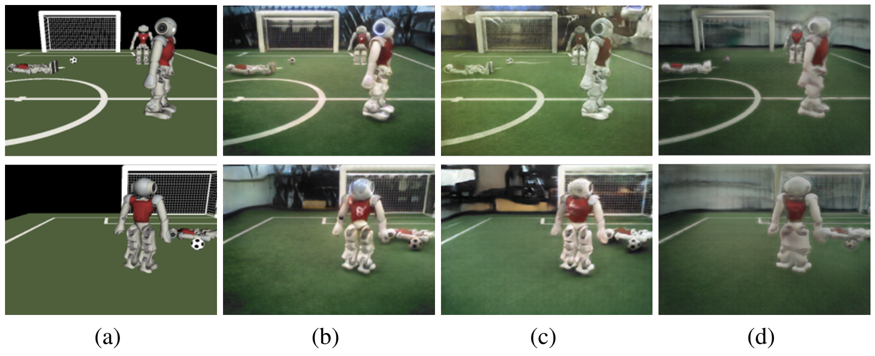


Figure 5: a) rendered images from simulation, b) aligned images generated with FeatureGan, c) aligned images generated with CycleGan, d) aligned images generated with CRN.

6.2 Visual Quality Analysis of CycleGan

CycleGan achieves some very crisp images with detailed textures in all the objects, a realistic background and realistic lighting conditions. However, artifacts were quite common both during training and inference over the test dataset. Images of the input rendered images B, generated realistic images $\hat{A}$ which present major artifacts and the reconstructed images $\hat{B}$ are presented in Fig. 6.

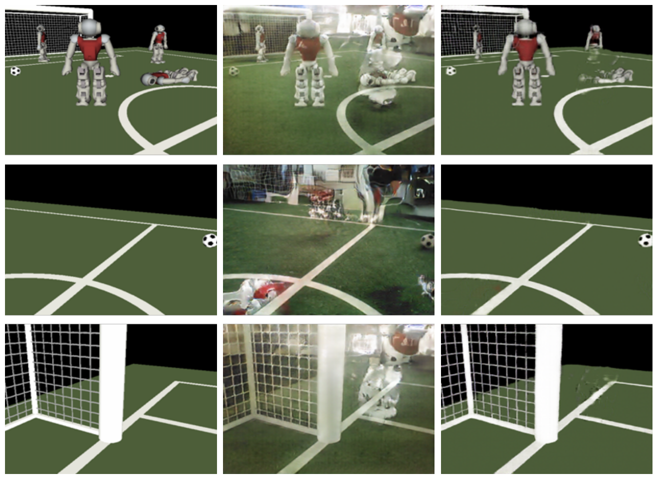


Figure 6: a) rendered image from simulation B, b) aligned images $\hat{A}$ generated by G_B, c) aligned reconstructed images $\hat{B}$ generated by G_A.

The artifacts appear to be a problem that derives from the CycleGan architecture when applied to the specific problem of the simulation-to-reality image translation task. CycleGan operates with two cycles, the first cycle works by generating a realistic image $\hat{A}$ from an input image B such that $\hat{A} = G_B(B)$. It then generates a second image $\hat{B}$ such that $\hat{B} = G_A(\hat{A})$. The loss for a single cycle is defined as the sum of the generator loss estimated by a discriminator over $\hat{A}$ and the $\text{Loss}_{\text{cycle}}$, which corresponds to the L1 loss between the reconstructed image $\hat{B}$ and the input image B, and is used to maintain the alignment between B, $\hat{A}$ and $\hat{B}$. The second cycle works in a similar fashion but from A to $\hat{A}$ instead of from B to $\hat{B}$.

The CycleGan methodology works very well for high variance, complex environments, with several successful test cases shown in the original paper. However, this methodology does not translate well for simpler environments. Indeed, the artifacts found in Fig. 6 are the result of training CycleGan over a dataset of samples collected from a low variance, highly structured environments such as the SimRobot simulated environment. If the generator G_A has learned a very accurate representation of the simulated scene, which is easy given the domain's low complexity and low variance, then the generator G_A can learn to generate an image $\hat{B}$ without artifacts in the simulated domain from an input realistic image $\hat{A}$ with artifacts. This is possible if the generator G_A is able to accurately detect the artifacts in $\hat{A}$ and then eliminate them in the output recreated image $\hat{B}$ by interpolating from known scene features. It follows that the generator G_A is not penalized by the $\text{Loss}_{\text{cycle}}$ for generating artifacts in the $\hat{A}$ image, since the generator G_B learns to delete such artifacts in $\hat{B}$. Then, there is nothing preventing CycleGan from falling into a mode collapse. This translates into the generator G_A outputting images that are most plausible to the discriminator, minimizing the generator loss. Then. the optimal solution for CycleGan is to generate an image $\hat{A}$ that minimizes the generator loss estimated by the discriminator (for example by adding objects to the scene that are not present in the input image), and then the generator G_B can detect these artifacts and delete them from the recreated image $\hat{B}$ to replicate the input image B almost perfectly. By doing so, both the generator loss estimated by the discriminator as well as the $\text{Loss}_{\text{cycle}}$ are minimized and by doing so the methodology encourages artifacts. This phenomena is shown in the images presented in Fig. 6, which are real results obtained from images in the test dataset and models resulting from later iterations of our training process. This artifacts translate in false positives for most heuristic based detectors and render the simulator basically unusable.

6.3 GTA-to-Cityscapes

The traditional implementation of FeatureGan was trained using 24,807 images from the Grand Theft Auto (GTA) 5 database as well as 23,417 samples of the Cityscapes dataset. Each class was assigned to a one-hot encoded vector, which was then concatenated to the input image B. A class-based feature loss was used to ensure alignment between B and $\hat{A}$. The classes terrain, sky, pole, traffic light and traffic sign which were more prone to artifacts were assigned an α_j of 2, while the rest of the classes were assigned an α_j of 6 to encourage domain transfer. The Loss_{GAN} was estimated using two pyramid discriminators with receptive fields over the input tensor of 70×70 (medium) and 140×140 (global). The input image was not used to generate IT. This prevents the training from collapsing. λ_i was set to $\frac{2}{3}$ for the medium discriminator and to $\frac{1}{3}$ for the global discriminator. The U-net generator was trained for 10 epochs with an initial learning rate of 0.0001 and with linear decay at a resolution of 256×256. The learning rate for the discriminator was set to $\times$ 1.5 the learning rate of the generator during the complete training process. A version of FeatureGan using two traditional PatchGan discriminator as well as another version using a traditional feature loss were also trained using the same parameters and at the same scales than the traditional implementation. Randomly selected results are shown in Fig. 7.

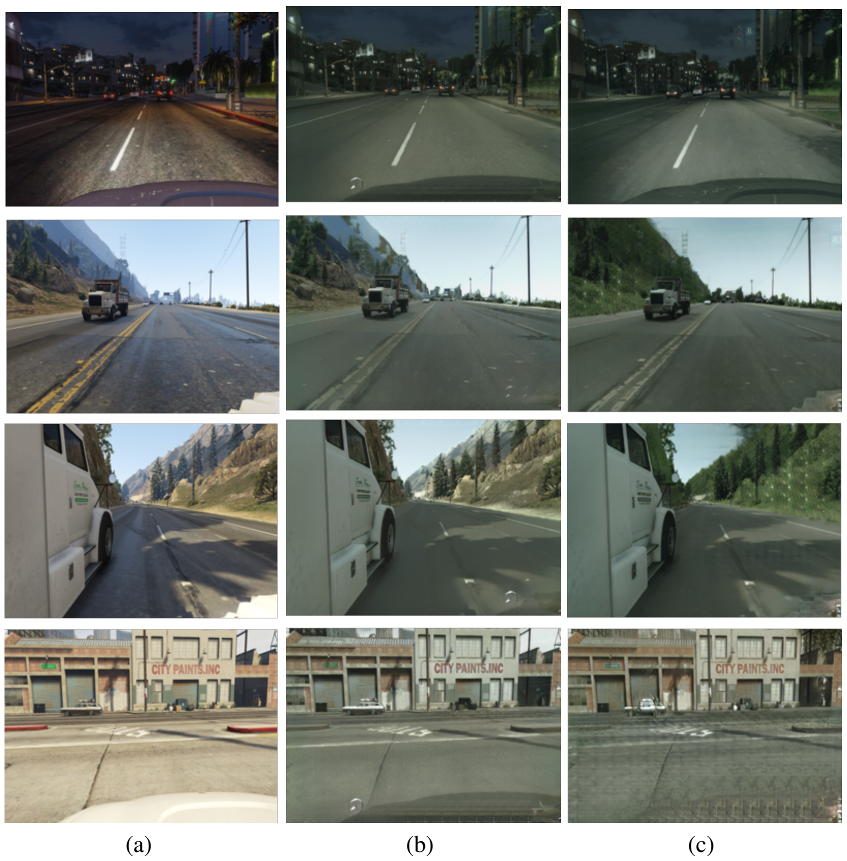


Figure 7: a) rendered image from simulation B, b) aligned image $\hat{A}$ generated with FeatureGan, c) aligned image $\hat{A}$ generated with FeatureGan without using a class based feature loss.

Fig. 8 shows the results of segmenting real samples of CityScapes with models trained with GTA samples and FeatureGan samples. It is apparent that the images segmented using a network rained with images generated by FeatureGan achieve results closer to the target segmentation than those produced by a network trained using samples from the GTA database.

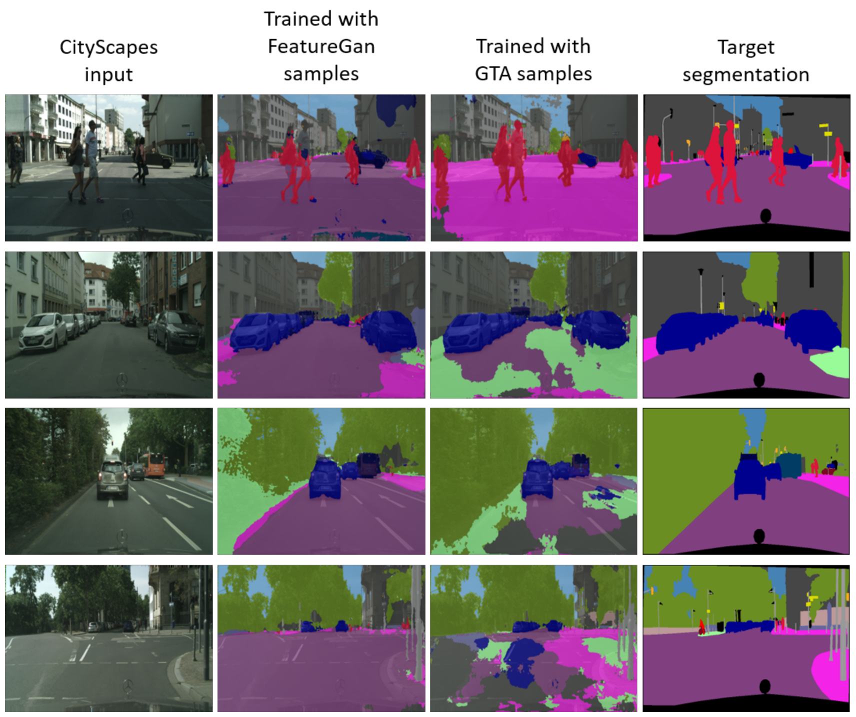
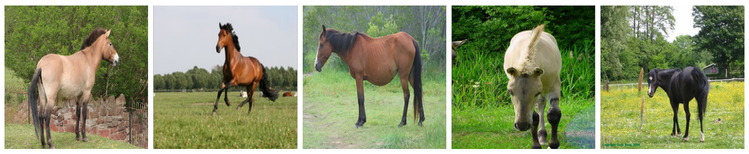


Figure 8: Segmentation results of DeepLabv3+ trained with different simulated datasets and applied to real samples.

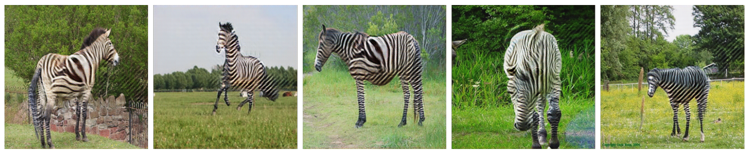
6.4 Horse-to-zebra Image Translation Task

FeatureGAN is used to successfully train a generator to solve the horse-to-zebra transfiguration task solved in CycleGan [6]. Since no class information is available, a traditional feature loss was used rather than the class-based implementation. The training database was composed of 1,067 images of horses as the input domain Dom$_B$, and 1,334 images of zebras as the target domain Dom$_A$. The Resnet generator proposed in Johnson et al. [17] was used to transform the images from Dom$_B$ to Dom$_A$. CycleGan was also trained using the same database and using the same generator architecture. Both networks were trained for 80 epochs with a constant learning rate of 0.00005 and for 80 additional epochs with decreasing learning rate.

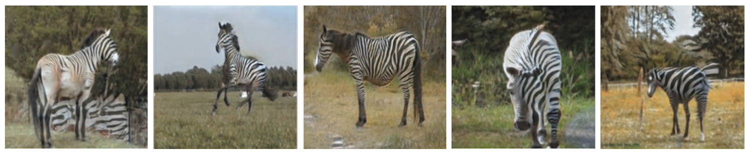
Results obtained using FeatureGan and CycleGan are presented in Fig. 9, and samples of the features from the compressed Feature Pyramid Tensor are shown in Fig. 10.



(a)

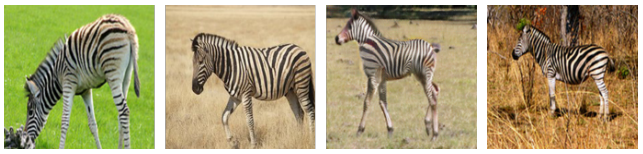


(b)

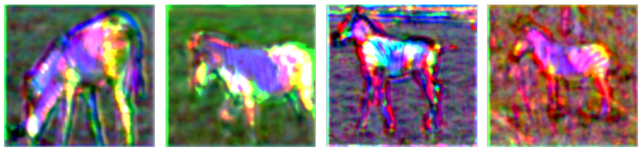


(c)

Figure 9: (a) Input images, (b) aligned images generated with FeatureGan. (c) aligned images generated with CycleGan.



(a)



(b)

Figure 10: (a) input images, (b) compressed FPT tensor with N = 3.

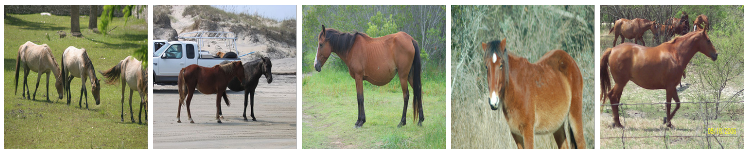
Both methods achieve results of similar visual quality which indicates that FeatureGan is not limited to the simulation-to-reality image translation task but can also be used to solve classic problems.

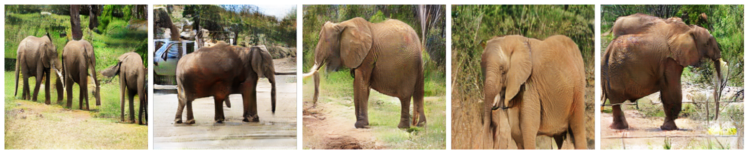
6.5 Horse-to-elephant Image Translation Task

A different problem is also addressed: the horse-to-elephant image translation task. This problem is more complex than horse-to-zebra since the system needs to learn to change the shape of the relevant objects in addition to the textures. Indeed, while horses and zebras are geometrically very similar, there is no clear way of mapping the geometry of a horse to an elephant. Both FeatureGan and CycleGan are trained using the same database consisting of 2,000 images of horses and 3,634 images of elephants for 80 epochs, with a constant learning rate of 0.00005, and 80 epochs with decreasing learning rate. A Resnet generator was chosen to perform the image translation.

Results obtained with FeatureGan and CycleGan are presented in Fig. 11 b) and c) respectively. Samples of the features from the compressed Feature Pyramid Tensor are shown in Fig. 12.

From Fig. 11 b) it is clear that FeatureGan is capable of solving the horse-to-elephant task, given that the method produces visually realistic images. This is a problem that CycleGan cannot solve since according to the original paper, one of the main constraints of CycleGan [6] is that the objects involved in the transfiguration task must share similar geometries. This is reinforced by the results achieved with CycleGan, which are shown in Fig. 11 c).


(a)


(b)

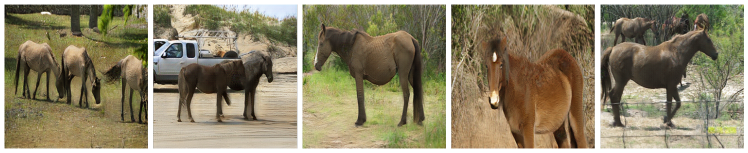
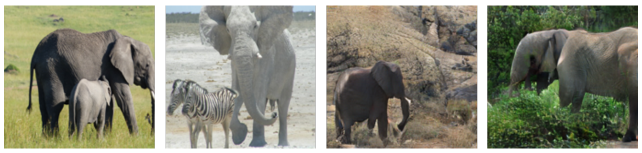

(c)

Figure 11: (a) input images, (b) aligned images generated with FeatureGan. (c) aligned images generated with CycleGan.


(a)

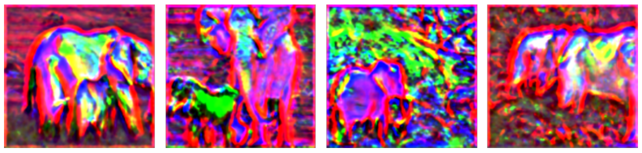

(b)

Figure 12: (a) input images, (b) compressed FPT tensor with N = 3.