Simulated Learning and Domain Transfer for Indoor Robot Navigation

Pierre Marza Hector Missiaen Grégoire Gentil

INSA Lyon
Villeurbanne, FRANCE

Abstract

The robot navigation task in a known or unknown environment is a trending subject in the Computer Vision field. Various studies showed promising results with Reinforcement Learning, often coupled with Deep Learning methods. However, these approaches usually require extensive training to be done with the robot in the anticipated environment. The field experimented with Simulated Environment Learning as it allows for very fast bootstrapping and learning. This specific task requires then to use the learned model in a real environment. An efficient way to close the gap between simulated and real data distributions is through Domain Transfer, i.e. methods trying to build models that are robust to a change of characteristics inside the considered data. In this paper, we provide a comparison between several state-of-the-art approaches to reduce Domain Shift in the case of our Robot Navigation task. We focus on adversarial methods, leveraging the generative capabilities of the Variational AutoEncoder and CycleGAN architectures to learn general latent representations. Unsupervised Domain Adaptation for robot pose estimation is also experimented. The experiments are to be done on CITI's provided robot equipped with a Microsoft Kinect video and depth sensor. We thus introduce CITI-Sim2Real, a new dataset adapted to our problem.

1. Introduction

Certain tasks requiring learning can suffer from a lack of available labelled data. Among them, we can cite 3D pose estimation as in [5] or robot navigation problem we are focusing on in this paper. The final aim is to provide navigation orders to a trained agent which will have to move to the right place in the space. To this end, the agent is given an image of its environment and has to pick actions in order to move. The robot policy is trained using a Deep Reinforcement Learning approach [2], considered a fixed component of the method that will thus not be discussed in this paper, based on features extracted from the input image.

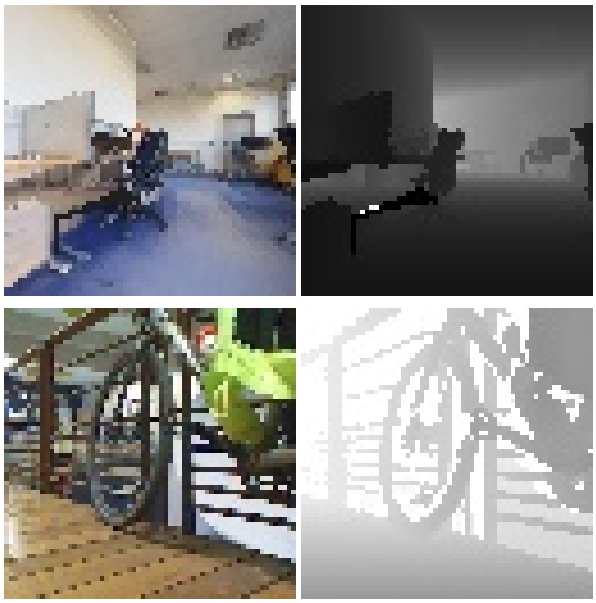


Figure 1: Examples of samples from CITI-Sim2Real dataset. **Top:** Simulated pair **Bottom:** Real-world pair **Left:** RGB images **Right:** Depth Maps

Mnih et al. [12] presents a Deep Reinforcement Learning framework to play Atari Games using game images. The power of Neural Networks feature representation allows the authors to predict very efficient agent policies, this method being now seen as a breakthrough in the field.

Feature extraction is an important Computer Vision task that plays a central role in any system dealing with complex input data. In the case of images, the power of Deep Convolutional Neural Networks comes from the hierarchical feature extraction they are performing while the information is passed through layers. In this paper, we are focusing on the Feature Extraction Block of the final Deep Reinforcement Learning Architecture.

As the training of the agent involves taking actions and, thus, navigating, not only we can lack labelled data but training a robot to navigate in a real-world environment is not possible. Therefore, simulated environments provide a straightforward generation of controlled and easily labelled information. Another interest of such synthetic framework is to allow the robot to evolve and learn in safe conditions. However, one drawback of systems trained on synthetic data is their possible lack of generalization on other data distributions, e.g. real-world images. A task of Domain Transfer has then to be performed. The core aim is here to extract features from images that will be as pertinent on a particular distribution of samples, e.g. real-world data, as on another one, e.g. simulated data.

Instead of tackling directly the robot navigation problem, we focus in this paper on an easier task that is robot pose estimation. This choice was made to simplify the evaluation process and promising results could easily be extended to the initial navigation goal. Robot pose estimation is indeed already an interesting measure of how knowledge gained on synthetic data can be generalized to real images.

Simulated Learning being really convenient to deal with particular tasks, several synthetic environments were designed. An example of this is Habitat [13] that provides a realistic 3D space any agent can evolve into. In the context of our work, we were given a similar simulation of one specific floor of the CITI Lab building. We also sampled real-world photos of this same space using a Microsoft Kinect video and depth sensor that is quite similar than the camera that will be used by the robot when navigating. CITI-Sim2Real, our new dataset, is then made of 2 parts - simulated and real-world data - that both contain (RGB image, depth map) pairs. It is important to notice that the dataset is not aligned. In fact, the simulated and real images are not paired, i.e. the photos are not taken from the same points of view. Figure 1 presents samples taken from CITI-Sim2Real dataset.

The contributions of this work are :

- Creation of CITI-Sim2Real, a new Robot Navigation Dataset made of real-world and simulated pairs of RGB images and depth maps from various viewpoints of the CITI Lab environment. The synthetic and real samples are unpaired.
- A comparison of state-of-the-art methods to perform robust Feature Extraction on RGB and depth inputs
- The use of a Domain Transfer approach to predict robot pose given real-world images.

2. Related Work

Deep learning Explained by LeCun [11] in 2015, Deep Learning was introduced as the outcome of several decades of research on text, speech and images recognition tasks. Deep learning is defined as the combination of independently trainable modules that decompose sequentially an input in its significant features. The structure hence constructed is then trained against labelled data which tune the connection between each modules using a backpropagation method.

Deep Learning neural networks can perform nicely when trained on large-scale datasets to perform a precise task. In [14], K. Simonyan and A. Zisserman introduce their well-known VGG16 architecture to perform classification task on the Imagenet large-scale dataset [3]. They show they can achieve drastically better results when increasing the depth of their Convolutional Neural Network, beating prior state-of-the-art architectures. Such models can indeed encapsulate a high level of complexity and adding more layers is a way to extract more fine-grained features. However, the features being dependent on the task and data used, these algorithms often tend to be biased towards the information they were trained on. Generalizing a Neural Network to a new task or data can be done using fine-tuning techniques. The most famous is Transfer Learning. Tan et al. [15] provide a complete overview of current Deep Transfer Learning methods. The core goal of these is to use the hierarchical nature of Convolutional Networks. First layers will indeed extract low-level features such as edges in the images while last layers encapsulate high-level semantic information. The idea behind Transfer Learning is thus to adapt a pre-trained model to a new task or dataset by only re-training the last layers to adapt them to the new considered semantic information. However, such approach generally involves access to a huge amount of data and can be computationally expensive.

Domain Transfer Domain Transfer methods try to close the gap between different data domains, usually during training.

An approach can be to build models using data that is less sensitive to domain change. By extending their Estimation Pose model to video, Doersch et al. [5] allow their architecture to be more robust to differences between synthetic and real-world data. They indeed leverage motion-related information such as an optical flow that captures motion between frames and not task or dataset-specific details. Indeed, vectors describing motion will be more robust to domain shift than low-level pixel-related data.

Based on a similar distribution of both domains samples,

the interface between domains can be modelled as a common representation (e.g. the latent space of a variational autoencoder) or can be done by transferring the input directly from source to target domain (e.g. a target sample is considered as a source sample).

Manifold Alignment, as in [1], can also be used to perform Domain Transfer. However, in this paper, authors tackle the problem of transferring knowledge between different actions taken by an agent, quite different from closing the gap between simulated and real environments in our case.

Then, lots of recent works have been focusing on adversarial approaches to reduce domain shifts by transferring feature representations. The first of its kind [6] introduced the adversarial learning method based on domain adaptation task. The concept of their method resides in the fact that a model trained both on labelled data from the source domain and unlabelled data from the target domain tends to factor in features that discriminate on the main learning task but indiscriminate according to the domain shift. Later with [16], Tzeng et al. build a general framework describing adversarial methods applied to Domain Transfer. This effort of formalization allows them to highlight interests and drawbacks of current state-of-the-art methods and therefore to propose their own approach called Adversarial Discriminative Domain Adaptation (ADDA). They tackle the task of classification while having a set of source images X_s, their corresponding labels Y_s and some target images X_t. They, therefore, want to learn a mapping function M_t encapsulating a representation of the target input and a classifier C_t using the latter. As supervised learning is not possible due to the lack of target labels, they build a general definition of the transfer method aiming to learn M_s and C_s (equivalent to M_t and C_t but on source domain) with M_s close enough to M_t so that C_s can be used on target inputs. They define features of different possible adversarial Domain Transfer approaches: weight sharing between source and target feature extractors, symmetric or asymmetric methods and finally the chosen adversarial loss. After experimenting on this, they show previous generative methods such as CycleGAN [18] (see below) for instance often allow interesting visual results but are sub-optimal when finally used on discriminative tasks. They can also struggle to close important domain gaps. On the other hand, discriminative methods in the literature can deal with larger domain differences but didn't take advantage of powerful adversarial losses. They finally use what they learned from their experiments to design ADDA. It is a discriminative two-step approach with unshared weights. The source feature extractor is first trained along with the classifier in a supervised fashion before transferring knowledge to the target feature extraction network through a domain-adversarial loss. Having two separate discriminative representation networks allows performing more domain-specific feature extraction.

The model of Zhu et al. [17] is trained using the previously mentioned ADDA framework for a task similar to ours, indoor robot navigation. The goal (RoomGoal in their use case) is defined as a feature vector and embedded in an LSTM network. They paired the method with a policy distillation process and obtained a 21.73% improvement over their estimated baseline.

Unlike [16], others try to leverage the advantages of generative approaches. Gupta et al. [9] use indeed a CycleGAN to transfer an image used by a robotic arm from one environment to another. As described in [18], CycleGAN consists of two GAN [8] networks, each one learning an opposite mapping: the first is trained to transfer the input image from environment A to B, and the other GAN to transfer from B to A. Both GANs are thus trained using a consistency loss. This paper is interesting in the way that it provides an alternative learning method to train a model to transfer one image from a domain to another one. They take a policy trained in a given environment, then try to learn the adversarial model that allows transferring input (images taken by the robot camera) from the new domain to the domain in which the policy has been trained. Thus, they introduce the idea that we can consider transferring the real environment image taken during the navigation to the simulated domain before forwarding it to the policy network, instead of trying to make the simulated and real environment features closer in terms of distribution during training. One of the most interesting points of this approach is that there is no need to re-train the policy network in a new environment, but only the transfer model.

Variational AutoEncoder In this work, we try to take advantage of the interesting characteristics of the Variational AutoEncoder (VAE). It is a generative model framework built upon basic functions like neural networks. It allows for a closely coupled generation of data from a dataset of unknown distribution. Based on a Bayesian estimator (Kingma and Welling) [10], the generated data then fits the distribution with slight variation. Doersch [4] provides a very detailed description of VAEs and the theories they are built on top of. Like all autoencoders, the initial goal is to learn an informative enough latent representation of the input in order to be able to reconstruct it properly. However, unlike other encoder-decoder approaches, VAEs learn to retrieve the distribution of the bottleneck latent variables. As a result, it makes it possible to sample directly latent values, thus generating new examples belonging to the training distribution. To this end, aside from reconstructing the input correctly, the network has to be regularized to ensure the latent space is continuous enough. Indeed, autoencoders tend

to build discrete latent representations to minimize reconstruction error. The idea behind VAEs is then to enforce latent distributions to be close to standard normal distributions.

Exisiting models Another approach for unsupervised learning with domain adaptation was explored by Ganin and Lempitsky [6] [7] which showed better than average results on several datasets which, however, were quite less complex than our task. We chose to adapt their framework for the pose estimation domain transfer task and the architecture will be detailed much more in the Methods section. Basically, it extracts features from the input and store it in a feature vector, then during training we backpropagate two losses, a label related loss and a domain discriminator classifier loss.

3. CITI-Sim2Real Dataset

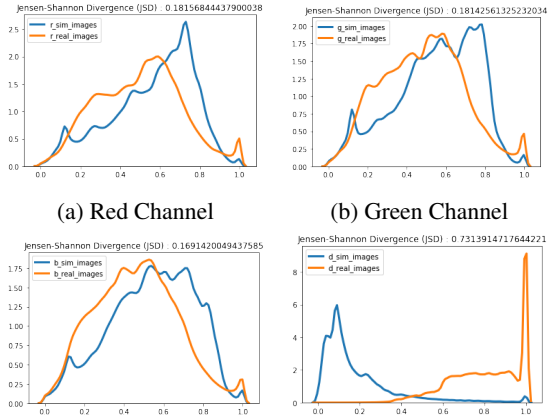
Our approach on the task of robot navigation was data-driven by the CITI-Sim2Real dataset. This dataset is made of both real and simulated samples. The images from the two domains are not aligned, i.e. there are not pairs of simulated and real images corresponding to the same viewpoint in the real scene. Figure 1 shows few examples taken from the dataset.

The latter is a large image repository split into training and validation subsets. It is split up with a 70-30 strategy. It contains 100,000 samples directly shot from the simulation environment, with separate files for depth and RGB information. The simulation is based on a Matterport scan of the CITI lab which is a large floor with 8 offices, corridors and large spaces. The Matterport simulation models the space from a wide angle visible-light camera and a depth sensor. Then, the dataset is built from a script that walks through the simulation. These images are labelled with their position and angle according to the referential of the space.

The real environment dataset was built for the purpose of these experiments. It contains a small number of pictures manually shot from the robot purposed to the simulation. The robot is a Turtle Bot, which is basically a motorized platform with a Kinect camera mounted on it and a "brain" computer to handle the policy choices. Our coverage strategy was to shot most of the discoverable space with the robot in every angle to cover the possible field of view of the robot. We ended up with a set of 870 independent poses with both visible and infrared depth information in separate files as well. Note that these images are not labelled in any way.

Our biggest concern was to guarantee that the pixel-level distributions of the two datasets were similar to facilitate domain transfer. An issue for us was to ensure consistent lighting conditions in the space.

Figure 2 presents pixel-level distributions of the dataset



(a) Red Channel (b) Green Channel

(c) Blue Channel (d) Depth Channel

Figure 2: Pixel-level distributions of real and simulated images inside the CITI-Sim2Real Dataset for each of their four channels (RGBD). Blue and orange curves correspond to simulated and real distributions respectively.

images. We computed the Jensen-Shannon divergence between R, G, B and Depth value distributions of data from both domains. Real and simulated R, G, B channel value distributions are really close. However, the huge divergence in Depth value distributions highlights the poor quality of depth measures from the Kinect sensor (as can be seen in Figure 1) compared with the ideal results coming from the simulation. To deal with this problem, we tried to apply some modifications on the depth images so their histograms are more similar to the simulated ones. First, we applied a scale over each pixel values to shift histogram in the direction of the black colours. In fact, as we can see, the real depth images are significantly clearer than the simulated ones. In a second step, we tried to correct some errors of the depth sensor that resulted in unexpected white pixels on the images. We applied an algorithm that replaces white pixels by the adjacent darkest colour.

The output data is based on a representation of the CITI lab and has to be normalized accordingly especially in terms of coordinates.

4. Methods

4.1. Image feature extraction from simulated environments

Building a robust feature extraction is an important goal when dealing with domain transfer. We thus applied two well-known approaches on the CITI-Sim2Real image samples, the Variational AutoEncoder and the CycleGAN.

4.1.1 Variational AutoEncoder

Our first task was to see how a Variational AutoEncoder (VAE) would allow for unsupervised feature extraction and the construction of general representations. Figure 3 is an abstracted representation of the learning framework we could use to perform Domain Transfer on the latent distributions learnt by a VAE. In our case, we focused only on training the encoder and decoder parts (including the extraction of a latent space representation) on simulated data.

The RGBD input is first encoded. To this end, it is sent through a series of fully convolutional layers with chosen kernel size and stride values to downsample the image sequentially. The dimension of the feature maps increases during the encoding forward pass. Linear layers are used to infer the parameters, i.e. mean and variance, of the Gaussian distributions from where the bottleneck latent representation is sampled. The latter is then resized using a linear operation before to go through the decoder. A series of deconvolutions allows to upsample the feature map dimensions until being equal to the initial input image. In a symmetric fashion, the number of filters decreases during the decoding forward pass. Figure 6 presents the final VAE architecture in more details, as explained in the Results and Experiments section.

The learning process involves the minimization of a loss function composed of two terms. The first one is the reconstruction term, i.e. how much the output image is far from the input. This allows to verify the quality of the learnt latent representation. Indeed, in the case it doesn't contain enough information, the reconstruction from the decoder will lack quality. The reconstruction error is measured using pixel-wise binary cross-entropy between the reconstructed input and the input. The second loss term is built by computing the Kullback–Leibler divergence between the latent Gaussian distributions and Standard Gaussian distributions $\mathcal{N}(0, I)$. This enforces the learnt latent space to be continuous, thus providing generative capabilities.

4.1.2 CycleGAN

We also tried to apply the CycleGAN solution to our problem. As described before, it allows to transfer images from the real domain to the simulated domain, and vice versa, using two GANs. Each GAN consists of a discriminator that can make the difference between images from real and simulated domains, and a generator that is trained to generate images of the real domain from images of the simulated domain (symmetrically, from real to simulated). The first loss that is optimized measures the ability of the discriminators to determine if a given image was either generated or came from the dataset. The parameters of the discriminators must be modified to reduce this loss, while those of the generators must, on the other hand, be optimized so that the discriminators be less confident, meaning the generated images are close to the ones taken from the dataset distribution. The second loss used to train the generators is the cycle consistency loss. It measures the difference between an original image from one domain, and the generated image passed through the two generators, first transferred to another domain by a generator, then transferred back to its original domain by the second generator. This allows to train the GANs using an unpaired dataset, which is the case of our CITI-Sim2Real dataset.

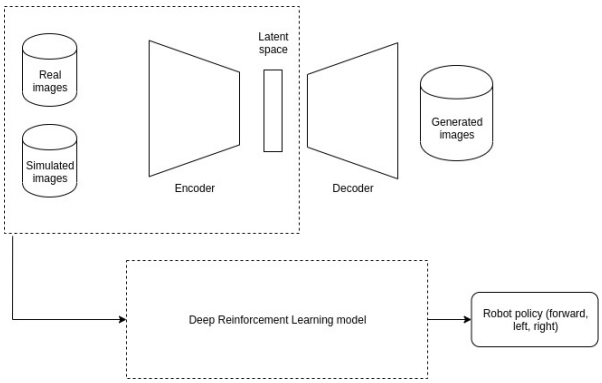


Figure 3: Potential domain transfer learning framework using a Variational AutoEncoder as feature extractor

If we want to use this trained CycleGAN to perform robot navigation in the real world, there are two possibilities. The first one is to use a specific part of the generators as a feature extractor and use it in the workflow presented in Figure 3 in the same way as the VAE. The second solution is to use generators directly as domain transfer tools, to convert our real images to images that are close to the simulated data distribution. Then, we can use these transferred images directly in the Deep Reinforcement Learning model trained in the simulated environment. This last approach could be hard to use in real time as a first forward pass would have to be processed to generate a simulated-like representation of a real image before to infer the best action to take.

4.2. Domain transfer for pose estimation

Each one of the architectures mentioned before provides us with a good framework to approach new problems, especially for our domain transfer task. An intermediate task on our road to indoor robot navigation was to prove that the domain adaptation could be done properly. As previously mentioned, the simulated images are labelled with their pose, noted $X = (pos_x, pos_y, pos_z, angle)$. However, considering the flatness of the CITI lab, as the pos_y coordinate represents height, it's irrelevant and thus will be ignored during data processing.

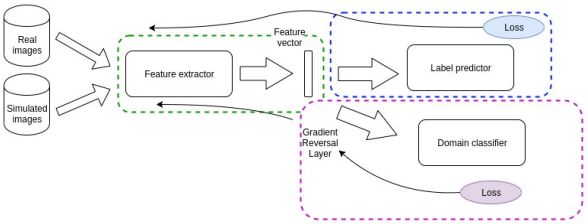


Figure 4: Model of our network from Ganin and Lempitsky's work [6]

Our approach was to define a network using the previously studied methods of feature extraction to estimate those 3 pose values. The model engineered was inspired by Ganin and Lempitsky [6] Figure 4 [1] which consists of a feature extractor, a label predictor, and a domain classifier, also called discriminator. The latter is trained through a gradient reversal layer and aims at classifying the extracted features as coming from a real or simulated image in input. In our case, the label predictor is a pose regression network made of linear layers that aims to predict the previously mentioned 3 values describing the current state of the robot. Except for the regressor, all of these structures are made of deep convolutions and errors from the label predictor constitute a second loss during training. In the end, we should obtain a robust system that estimates as closely as possible the position of the robot solely from its RGB input and recognition of its surroundings.

This approach is really interesting because, unlike the VAE and the CycleGAN that try to learn the joint distribution of input and output variables in order to encapsulate generative properties, we are here only extracting discriminative features. As explained in [16], this could lead to better performance in terms of pose regression.

5. Results and Experiments

To perform experiments, we have been given access to the LIRIS-Imagine computing cluster which consists of 4 nodes, with each one of them accessing 4 high-end workstation grade GPUs. The maximum training time was two days, that we sometimes used entirely or not, depending on the convergence time of the chosen methods.

5.1. Variational Auto Encoding

The first step was to train the VAE on the simulated images to show that the learnt latent space was indeed a representation of the dataset. We performed an important hyperparameters search to find the most appropriate learning rate, training batch size, latent representation size and more generally the network capacity (number of layers in the encoder and decoder, but also convolutional parameters such as numbers of filters, strides and kernel sizes). Because of the huge size of this search space, we performed a random, thus not complete, search but also guided a lot by the best state-of-the-art architectures.

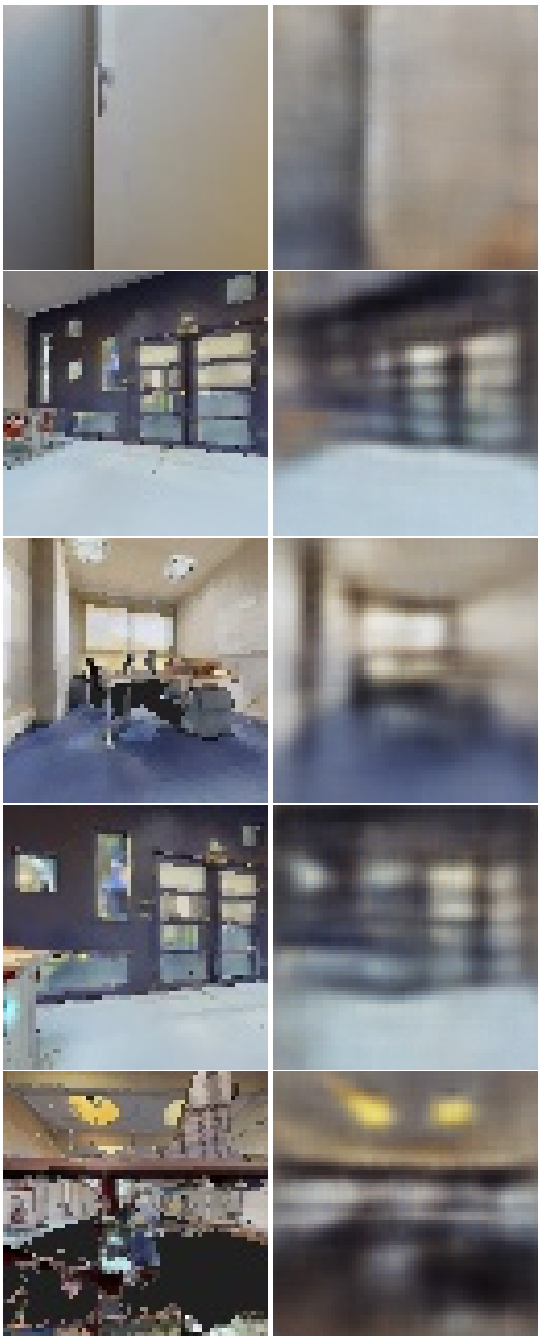


Figure 5: Reconstruction results from the Variational AutoEncoder on validation RGB images. **Left :** Input images **Right :** Reconstructions

[1] https://github.com/machine-perception-robotics-group/DANN

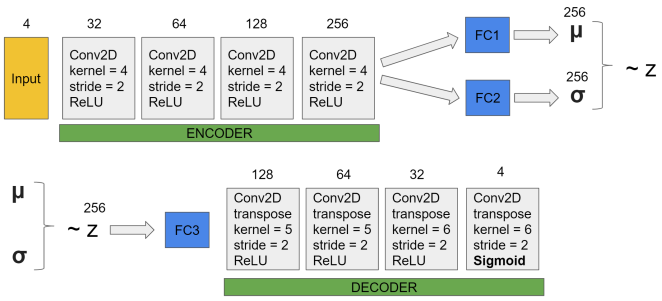


Figure 6: VAE Architecture

Figure 6 presents the final chosen architecture. As a result, the encoder is made of four convolutional layers with a kernel size of 4 and a stride of 2 each. The number of filters goes from 32 to 256, doubled between layers. We apply batch normalization on the outputs of the convolutional layers to normalize these between 0 and 1 and use the well-known Rectified Linear Unit (ReLU) as activation function, useful to avoid vanishing gradient. Inside the decoder, we also have batch normalization and ReLU activations. It is made of 4 deconvolutions, symmetric to the encoder in terms of numbers of filters. The kernel sizes are 5 for the first two and 6 for the next ones. The last activation function is a Sigmoid in order to get output pixel values between 0 and 1. Finally, the latent representation is made of 256 values.

With the setup of our Variational AutoEncoder, we defined an arbitrary quality evaluation based on comparing the input picture and the output of the decoder. We then describe the quality of the reconstruction through its sharpness and actual feature reproduction. Figure 5 shows couples of validation inputs and outputs. The aim of training a VAE is of course not to get a reconstruction as detailed as the input because we are also focusing on getting a continuous latent space. However, it looks like the model encapsulates most of the semantic as the main elements such as walls, windows, lights seem to be retrieved. More fine-tuning could have been interesting to get sharper outputs though. Another interesting finding is that the depth information doesn't seem to improve the reconstruction quality significantly while increasing the complexity of the model as it adds one dimension to the input. Figure 5 was thus obtained with RGB inputs only.

A part of our experimentation has also been to walk inside the latent space of the VAE to verify the semantic meaning of the learnt condensed representation. To this end, we chose random images from the dataset and computed their latent representation using the encoding path of the network. We then changed the values of the different variables to see which impact they had on the reconstructed image, thus indicating their semantic meaning. Figure 7 shows two examples of the impact of modifying the value of a chosen latent variable, keeping all the others fixed. The first variable (left column) seems to be responsible for the brightness of the output. Indeed, as we increase its value (going from the top to the bottom of the figure), the obtained output becomes much brighter. The second latent variable is correlated with the presence or not of a blue object in the background of the image. When increasing its value, the given object starts appearing. It also changes slightly the color of the desk on the left. The results where generated using a trained VAE with a latent representation of size only 32 as it allows for easier visualization of the latent space. As a result, outputs are less sharp then when using a VAE with a bigger representation size, e.g. 256 in Figure 5.

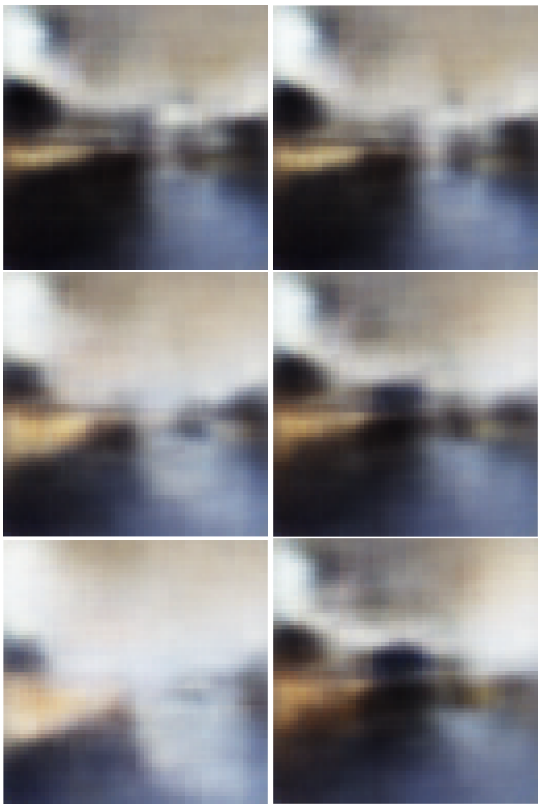


Figure 7: Outputs of the VAE decoder when modifying the value of 2 latent variables individually, keeping the others fixed. Each column corresponds to a particular variable. First row is a small value of the variable, second an intermediate value and third row a huge value.

5.2. CycleGAN

To experiment with the CycleGAN, we used an implementation proposed on GitHub [2] by the author of the origi-

[2] https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

Figure 8: Examples of result generated by the CycleGAN. **Left :** Original Real. Generated Sim. Reconstructed Real. **Right :** Original Sim. Generated Real. Reconstructed Sim.

nal paper [9] that introduced the CycleGAN. We trained the models using our dataset. You can find some transfer examples in Figure 8. On each column, the first image is an original image taken by the Kinect sensor or captured in the simulated environment. The second one is the result of the transfer to the other environment. The last one is the reconstruction of the original image after using both GANs. We can observe some color and contrast changes. It is not evident to analyse if the results seem good or not. Indeed, we have not found how to clearly differentiate real and simulated distributions of colours and characteristics, so we cannot act if the transfer is good or not. Unfortunately, we did not test the trained model on the navigation task to evaluate its performances, for time constraints.

5.3. Domain transfer with pose estimation

The pose estimation network was built from the feature extractor and label decoder of one of our fellow colleagues Theo Jaunet and then integrated to the Ganin-like [6] framework. It is known that this network used for pose estimation solely on simulated images is working properly with a satisfying 83% of estimated poses close enough to test ground truth (difference smaller than a chosen margin). The proper

Figure 9: Examples of couples validation real input and nearest simulated image based on the predicted pose. **Left :** Real images **Right :** Simulated nearest inputs

task was then to train once again this network in the framework with both real and simulated images.

For results evaluation, as the pose of real images were unknown, we computed the distance between predicted pose and all the target poses known for the simulated images in the dataset. The domain transfer efficiency is evaluated by displaying the image of the simulated domain that is the closest to the real image prediction pose in terms of both position and angle.

With this configuration and proper parameters for the framework, we obtained pairs of images representing real images and their predicted representations in the simulated space that didn't match as well as we could hope. However, this is difficult to really evaluate results as visual comparison can be quite subjective. Some issues in training could also be caused by the lack of real images. We tried to bootstrap the network from pre-trained versions on simulated images of the feature extractor and pose regressor components (warm start through initialization using a pre-trained model) which showed promising results. Thus, during training, we progressively tend to lose a bit of precision on pose estimation for simulated images (compared with initially pre-trained architecture), which can be explained as we are also now focusing on closing the gap between real and simulated domains by extracting a more general and robust representation. However, everything was not lost, we still got some promising results as shown in Figure 9. We can't say definitely that these good results are the consequence of proper training or the remain of the pre-training.

We also tried to compare a set of real images to the pre-trained feature extractor and pose regressor to prove that the domain transfer was indeed necessary. This showed us that simulated images would get very good positioning results when it would perform poorly on our set of real images.

Originally, the feature extractor architecture was designed with the depth dimension, however, we discovered that when training on the framework, the results were poorer using the depth on real images. This is probably due to misalignment of the IR emitter and receiver on the Kinect which lead to poor image resolution and artifacts on the captured depth images.

6. Conclusion

We've iterated on many works from the literature to experiment with domain transfer on a robot navigation task. Our work gave marginal results on the simulation to reality transfer. We've targeted an intermediate results set with the position estimation which proved domain transfer competence. We have also focused strongly on understanding and experimenting with unsupervised feature extraction, mainly using the VAE model on our data. We've contributed to build a starter dataset for the task of robot navigation in the CITI laboratory. The dataset contains 870 pictures from various positions and angles on the floor, the data contains both RGB and depth information extracted from the Kinect sensors.

This approach of simulation training and domain transfer looks very promising as it will require solely a 3D scan of the desired walkable space for the robot to adapt. The simulation training speed is then supposed to be very fast and the RoomGoal is easily solved.

References

[1] H. B. Ammar, E. Eaton, P. Ruvolo, and M. E. Taylor. Unsupervised Cross-Domain Transfer in Policy Gradient Reinforcement Learning via Manifold Alignment. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, Feb. 2015.

[2] E. Beeching, C. Wolf, J. Dibangoye, and O. Simonin. Deep reinforcement learning on a budget: 3d control and reasoning without a supercomputer. *CoRR*, abs/1904.01806, 2019.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2009.

[4] C. Doersch. Tutorial on Variational Autoencoders. *arXiv preprint, arXiv:1606.05908*, June 2016.

[5] C. Doersch and A. Zisserman. Sim2real transfer learning for 3d human pose estimation: motion to the rescue. In *Advances in Neural Information Processing Systems 32*, pages 12929–12941. 2019.

[6] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1180–1189. JMLR.org, 2015.

[7] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, Jan. 2016.

[8] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[9] A. Gupta and J. Booher. CycleGAN for sim2real Domain Adaptation. In *Stanford Technical Publications*, page 8, 2019.

[10] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, Dec. 2013.

[11] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. In *Nature*, volume 521, pages 436–444, May 2015.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv preprint, arXiv:1312.5602*, Dec. 2013.

[13] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, page 9, 2019.

[14] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2015.

[15] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A Survey on Deep Transfer Learning. In *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 270–279, Springer International Publishing, 2018.

[16] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[17] F. Zhu, L. Zhu, and Y. Yang. Sim-real joint reinforcement transfer for 3d indoor navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11388–11397, 2019.

[18] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks. In *IEEE International Conference on Computer Vision*, pages 2242–2251, 2017.