

Object Detection using Domain Randomization and Generative Adversarial Refinement of Synthetic Images

Fernando Camaro Nogues Andrew Huie Sakyasingha Dasgupta
 Ascent Robotics, Inc. Japan
 {fernando, andrew, sakra}@ascent.ai

Abstract

In this work, we present an application of domain randomization and generative adversarial networks (GAN) to train a near real-time object detector for industrial electric parts, entirely in a simulated environment. Large scale availability of labelled real world data is typically rare and difficult to obtain in many industrial settings. As such here, only a few hundred of unlabelled real images are used to train a Cyclic-GAN network, in combination with various degree of domain randomization procedures. We demonstrate that this enables robust translation of synthetic images to the real world domain. We show that a combination of the original synthetic (simulation) and GAN translated images, when used for training a Mask-RCNN object detection network achieves greater than 0.95 mean average precision in detecting and classifying a collection of industrial electric parts. We evaluate the performance across different combinations of training data.

1. Introduction

The successful examples of deep learning require a large number of manually annotated data, which can be prohibitive for most applications, even if they start from a pre-trained model in another domain and only require a fine-tuning phase in the target domain.

An effective way to eliminate the cost of the expensive annotation is to train the model within a simulated environment where the annotations can be also automatically generated. However, the problem with this approach is that the generated samples (in our case images) may not follow the same distribution as the real domain, resulting in what is known as the reality-gap. Several approaches exist that try to reduce this apparent gap. One such method is domain randomization ([1], [2]). In this, several rendering parameters of the scene can be randomized, like the color of objects, textures, lights, etc, thus effectively enabling the model to see a very wide distribution during training, and

seeing the real distribution as one variation in it.

Another approach that directly tries to minimize this reality-gap is to refine the synthetic images so that they look more realistic. One possible way to build such a refiner is by using a generative adversarial training framework [3]. An alternative and more indirect approach to reduce the negative effect of this reality-gap is to use again the GAN framework, but in this case, directly on the features of some of the last layers of the network being trained for the specific target task [4].

In this work we present an experimental use case of an object detector in a real industrial application setting, which is trained with different combinations of synthetic images and refined synthetic images (synthetic images refined to look more realistic). We evaluate our method robustly across various combinations of training data.

2. Synthetic Image Generation with Domain Randomization

The architecture to produce the synthetic images for our experiments is composed of two main parts. First, the physics simulation engine, Bullet¹ is used to place the objects in a physically consistent configuration after letting them fall from a random position. Second, the ray tracing rendering library POV-Ray² is used to render an image based on this configuration. In POV-Ray we introduce domain randomization, by randomizing several parameters, namely, the number of lights and their color, the color and texture of each part of the target objects and the scene floor plane, as well as the camera position. The camera position is drawn from a uniform distribution in a rectangular prism that is 10cm above the floor plane, with a squared base of side 20cm and 10cm height. Although the location of the camera was uniform, the camera was always pointing to the global coordinates origin with no rolling angle. This variation of the camera position was intended to achieve robustness against different positions of the camera in the real

¹<https://pybullet.org/wordpress/>

²<http://www.povray.org/>

world.

3. Refinement of synthetic images by adversarial training

An alternative way we consider to reduce the reality-gap is to use the GAN framework to refine the synthetic images to look more realistic. Here, we selected the Cyclic-GAN [6] architecture since it only requires two sets of unpaired examples, one for each domain, the synthetic and the real one. The original synthetic images of size 1024x768 were too large for the training of our Cyclic-GAN model, as such, instead of resizing the image, we opted for training on random crops of size 256x256. This way we can train in the original pixel density and exploit the fact that our generators are fully convolutional networks, such that during the inference phase we can still input the original full-size image.

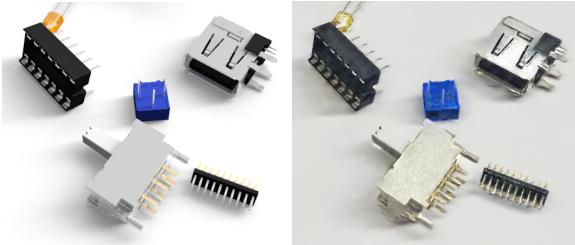


Figure 1: Left: example of synthetic image. Right: corresponding synthetic image after translation to real domain. The USB socket has gained a more realistic reflection, and the switch has gained a realistic surface texture and color.

We notice that after training, one particular target object lost its color and turned gray, while the remaining objects were refined in a realistic manner without loosing their original color. We think that this was mainly due to the particular architecture of the discriminators. The discriminator model final layer consisted of a spatial grid of discriminator neurons whose receptive field with respect to the input image was too small to capture that object. In order to solve this we added more convolutional layers to the discriminator models. This effectively increased the receptive field size. Furthermore, instead of substituting one grid of discriminators by another, we preferred to maintain both, one with small receptive field intended to discriminate details of the objects, and another with large receptive field, that can understand the objects as a whole (Fig. 3 in Appendix). The final loss was computed as the mean of all individual discriminator units for both of these two layers. This small modification enabled us to maintain the color of all the objects. The Cyclic-GAN model was trained using 10K synthetic images and 256 real images. Fig.1 shows an example

of the resulting image with our model that translates from synthetic domain to the real domain; see Fig. 4 in Appendix for more examples.

4. Experiments

In this section we compare different combinations of training data and its impact on the mAP for object detection with a Mask-RCNN model [5]. As a test dataset we have used 100 real images.

The different types of datasets used for training were: S_{fix} : synthetic images with fixed object colors without texture, and white background. $S_{fix \rightarrow real}$: translated images from S_{fix} to the real domain. $S_{rand-tex}$: synthetic images with objects and background with randomized colors but without texture. $S_{rand+tex}$: synthetic images with objects and background with randomized colors and texture. See Fig. 2 in the appendix for a general overview of the training architecture and Fig. 5 for some examples of different types of images employed.

The target objects to be detected, consisted of 12 tiny electronic parts for which accurate 3D CAD models were available (Fig. 6). In all the experiments we used 10K training samples, the same number of training iterations and the same hyperparameters.

The object detection performance for the different combinations of datasets used in the experiments are presented in Table 1 . Using a training set made purely of one type of data resulted in a mAP below 0.9 in most cases, with the exception of the case with $S_{rand+tex}$. Overall, the best detection results were obtained when the refined synthetic images set ($S_{fix \rightarrow real}$) was combined with high variation randomized data ($S_{rand+tex}$). The results indicate that neither domain randomization or GAN based refinement is enough on its own to get sufficient performance. In combination, they reduce the reality-gap effectively, resulting in a significant boost in performance (see the real-time object detection video at <https://youtu.be/Q-WeXSSnZOU>). Refer to Fig. 7 for the training curves associated with the different experiments, and to Fig. 8 for some detection result images.

Training data	mAP (0.5 IoU)
100% S_{fix}	0.812
100% $S_{fix \rightarrow real}$	0.874
100% $S_{rand-tex}$	0.867
100% $S_{rand+tex}$	0.911
20% S_{fix} and 80% $S_{rand+tex}$	0.914
20% $S_{fix \rightarrow real}$ and 80% $S_{rand+tex}$	0.955
50% $S_{fix \rightarrow real}$ and 50% $S_{rand+tex}$	0.950

Table 1: Performance of the Mask-RCNN network for the different training datasets.

References

- [1] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, Pieter Abbeel. *Domain Randomization for Transferring Deep Neural Networks from Simulation*. In the proceedings of the 30th IEEE/RSJ International Conference on Intelligent RObots and Systems (IROS), Vancouver, Canada, October 2017
- [2] Sadeghi, Fereshteh and Levine, Sergey. *CAD2RL: Real Single-Image Flight without a Single Real Image*, Robotics: Science and Systems(RSS), 2017.
- [3] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang and Russell Webb. *Learning from Simulated and Unsupervised Images through Adversarial Training*. 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017
- [4] Ganin, Yaroslav et al. *Domain-adversarial Training of Neural Networks*. The Journal of Machine Learning Research, 2016.
- [5] Kaiming He, Georgia Gkioxari, Piotr Dollár and Ross B. Girshick. *Mask R-CNN*. 2017 IEEE International Conference on Computer Vision (ICCV), 2017.
- [6] Jun-Yan Zhu, Taesung Park, Phillip Isola and Alexei A. Efros. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. arXiv preprint arXiv:1703.10593, 2017 2017 IEEE International Conference on Computer Vision (ICCV), 2017.

Appendix

In Fig. 2 we provide a schematic overview of the object detection training data generation pipeline.

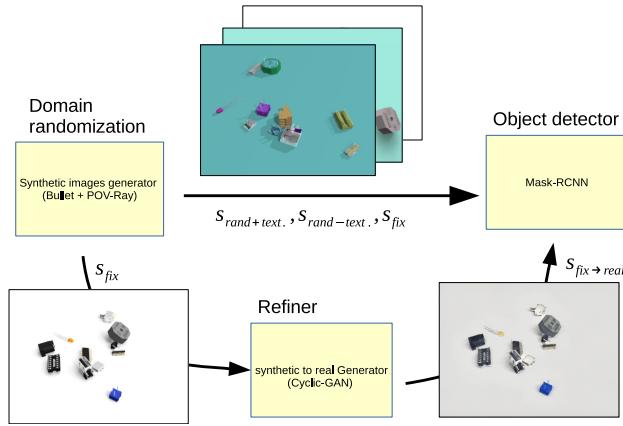


Figure 2: General architecture for training the object detector.

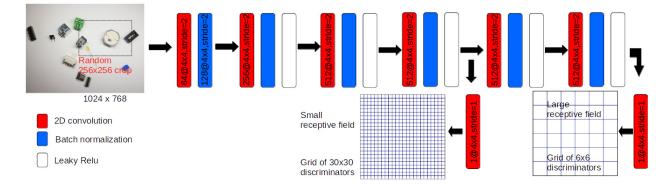


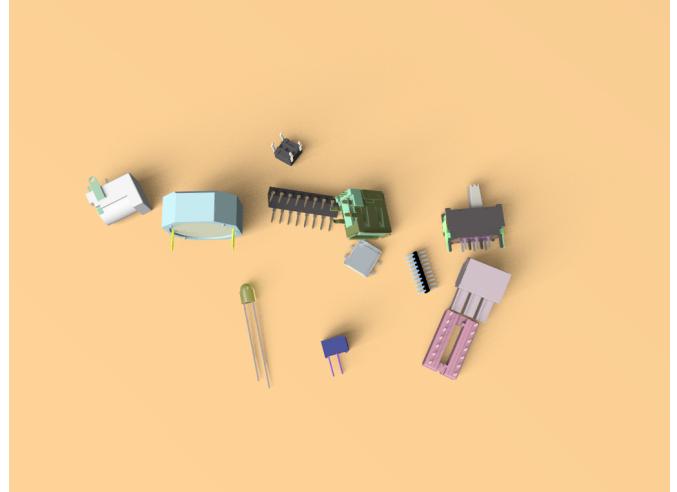
Figure 3: Discriminator network with two grid layers of discriminator cells, one with small receptive field and the other with bigger receptive field.



Figure 4: Left column: images from S_{fix} . Right column: corresponding refined images ($S_{fix \rightarrow real}$).



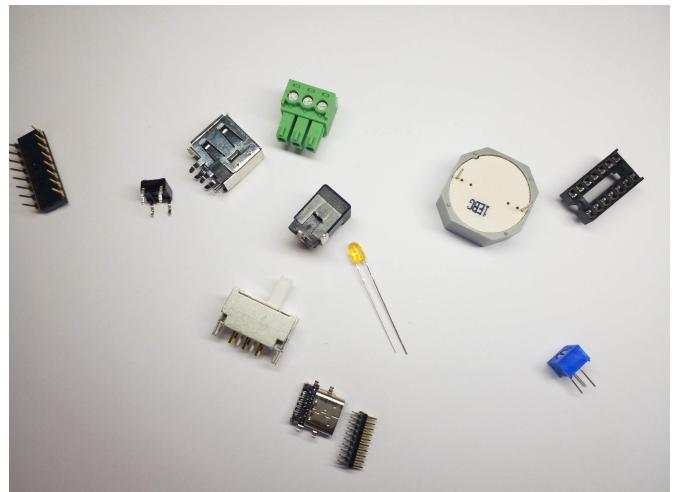
(a) Example of S_{fix} image



(b) Example of $S_{rand-tex}$ image



(c) Example of $S_{rand+tex}$ image



(d) Example of a real image used to train the cyclic-GAN

Figure 5: Examples of different types of images employed in the experiments.



Figure 6: Electronic parts used in the experiments.

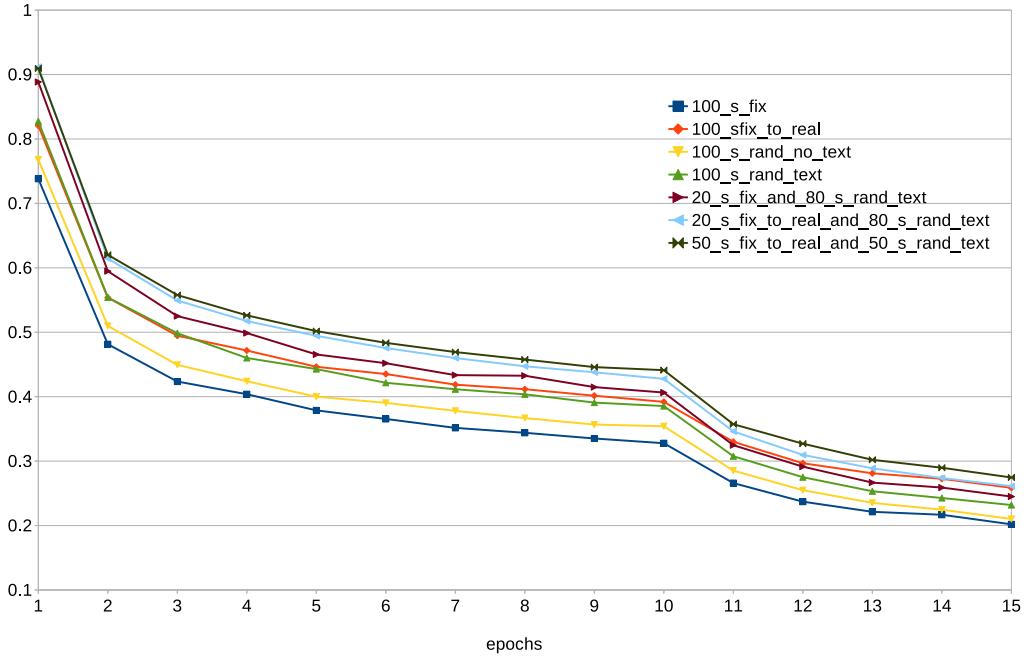


Figure 7: Mask-RCNN training loss. The model was trained by fine-tuning a Mask-RCNN model pre-trained on the COCO dataset. First by training only the mask-rcnn heads (without training the region proposal network or the backbone model) for 10 epochs with a learning rate of 0.002, and then the whole network for another 5 epochs with a learning rate of 0.0002. We used a SGD optimizer with a momentum of 0.9. The configurations that achieved the better performances, "20% $S_{fix \rightarrow real}$ and 80% $S_{rand+tex}$ " and "50% $S_{fix \rightarrow real}$ and 50% $S_{rand+tex}$ ", are the ones that had worse loss values during training. We think that this is because these datasets were more difficult, but at the end prepared the model better for the also difficult real test dataset.

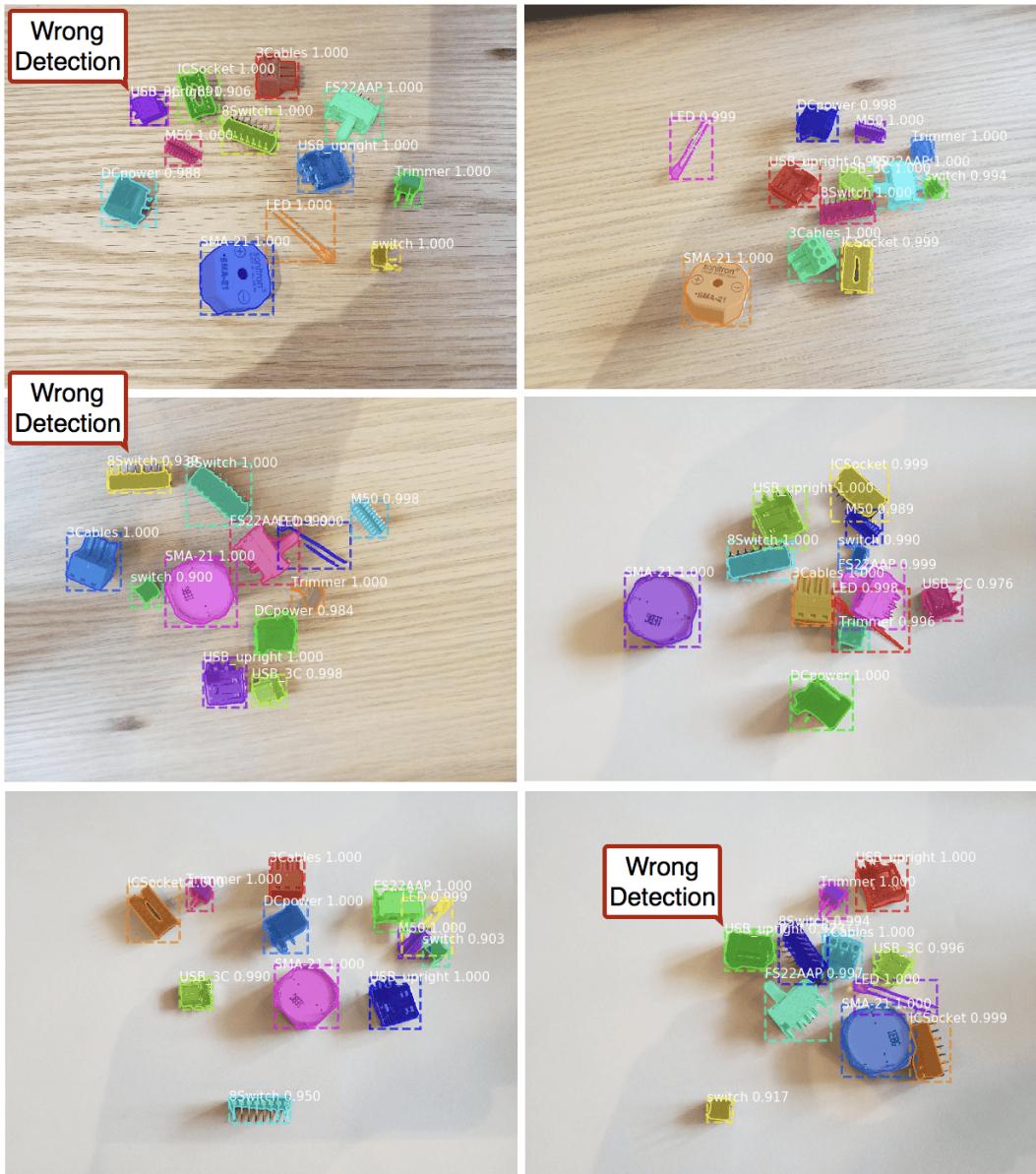


Figure 8: Example of detection results for 20% $S_{fix \rightarrow real}$ and 80% $S_{rand+tex}$.