ME 5493 Robotics
December 15, Fall 2017
San Antonio, TX, USA

DOMAIN RANDOMIZATION FOR CLASSIFYING IMAGES

Ezra Ameperosa
Robotics and Motion Laboratory,
Dept. of Mechanical Engineering
San Antonio, TX, USA 78249
ezra.ameperosa@gmail.com

ABSTRACT

When creating simulations for robots we often find that when implementing on the actual system, it doesn't always work as intended as we cannot model such things as wear and tear of robot parts and so on. We refer to this as the "reality gap". In this paper we investigate Domain Randomization for computer vision, a technique in deep neural networks in which a network is trained on generated images with random image features. We use GoogleNet CNN with pretrained weights and train it with our synthetic dataset. We train the neural network three separate times with varying learning rate and observe how well each generalizes to the real world. Our results show that our synthetic data may need more complexity in its randomization and we discuss possible improvements.

1. INTRODUCTION

There lies an issue in transferring simulation into hardware: the reality gap. While we can make exquisite simulations that represent the real world accurately, there is the underlying fact that nothing is made perfect. Moreover, trying to model all the discrepancies between a simulated and real environment is a challenging task.

With the advancements in machine learning and deep neural networks, we see work in closing the reality gap using synthetic data. Using synthetic data is attractive as neural networks require a lot of data and creating synthetic data is quick with the advances in computation power. Furthermore, collecting and labeling data is time consuming, however we can quickly and easily label our data if we create it synthetically. In [1] Bousmalis et al. uses domain adaptation in which they train their network with synthetic data and transfer policies to real-world model and continue with training their network to better adapt to the real world. Peng et al. [2] use inaccurate 3D CAD models to create synthetic images that improves performance on the PASCAL data set. In [3] the authors use pure synthetic data to have robotic manipulator locate and pick up objects. Here they claim to be the first to successfully transfer a deep neural network trained only on fabricated data.

In this paper we explore the use of domain randomization in computer vision and create a binary classification neural network trained only on synthetic data to identify a toy resembling the school's mascot, Rowdy.

2. METHODS.

We wish to perform binary classification of whether Rowdy is in a picture or not with the use of domain randomization. In the proceeding sections we describe our approach to creating synthetic images and the deep neural network architecture we use to train the classifier.

2.1 Domain Randomization

In creating the synthetic data, we randomize various aspects to train the classifier to be invariant to these randomizations when it is expose to real images. For creating and rendering the images, we use Blender. We randomize the following elements:

- Three cameras varying in position
- Number of lighting sources and their placement
- Number and type of distraction shapes
- Size, position and orientation of each object
- Color of the background and objects.

Each generated scene contains three cameras set at a fixed radius from the center of the scene. The cameras are oriented spherically about the center of the scene randomizing ϕ and θ angle. The lights are randomized in a similar manner as the cameras. Between 1 and 5 light sources are added in each scene are set normal to the background.

There are 0 to 5 distraction shapes added to each scene that can either be a sphere or a cube. Each distraction shape is given a random color, and is positioned randomly in each scene.

We generate 3,350 randomized scenes taking images of each scene with the three cameras with half of the scenes containing Rowdy. We then separate 90% of the images into a training data set and the rest as the validation data set. There was no preprocessing of the generated images.

2.2 Neural Network

For our neural network we used the GoogleNet architecture with pretrained weights from ILSVRC2014 data set. We choose GoogleNet because of its speed and accuracy compared to other architectures [4]. We also use transfer learning rather than learning from scratch as [3] shows learning is faster than random weights, which is typical for transfer learning. We modify GoogleNet to output binary classification and change the last three layers into a two neuron fully connected layer, a softmax layer and a classification layer.

We use 5 epochs to train the network with a minibatch size of 32. We perform stochastic gradient descent with momentum experimenting with different learning rates (1e-4, 1e-5, and 1e-6). We keep the momentum constant to default at 0.9 and add L2 regularization with the constant equal to 0.1.

3. RESULTS

Initially, we trained the classifier at a learning rate of 1e-6 giving us a 95.82% accuracy on the validation images. We test the trained classifier on actual images with some distractions and find that this first classifier is overfitting the training data; the classifier falsely identifies Rowdy in the real images. We retrain the classifier two additional times with different learning rates (Table 1). We test each classifier, examining how well the classifiers generalize to real images. Alike Figure 2, we exhibit each classifier to the same images to determine how well each perform. We find that the classifier with 85% validation accuracy performs the best on classifying real images.

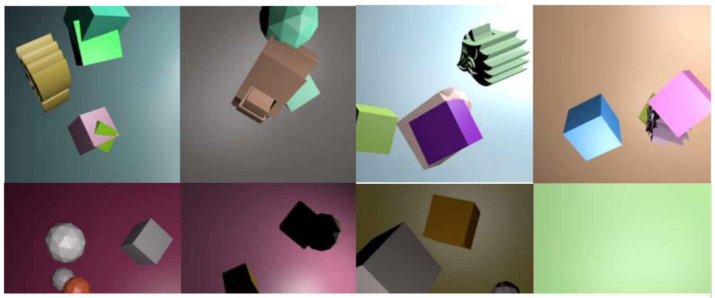
Figure 1: Sample Synthetic data. The top row has Rowdy randomly oriented in the pictures. The bottom row does not have Rowdy.

Table 1: Accuracy of the classifier is on synthetic data using different learning rates. We test how well each generalize to real images.

Learning Rate	Validation Accuracy
1e-4	95.82
1e-5	91.84
1e-6	85.57

We perform further tests on the classifier with 85% accuracy and exhibit to different images changing features such as the Rowdy figure, the position and orientation (Figure 3). We also test on images with Rowdy partially occluded with the classifier showing it can identify correctly. The robustness of the classifier is tested, changing the background and introducing different distraction objects.

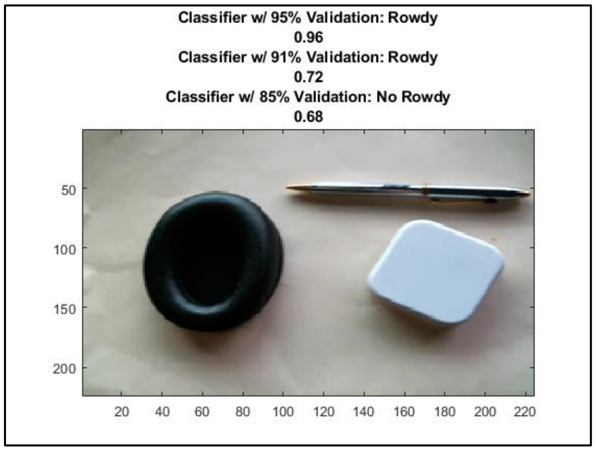
Classifier w/ 95% Validation: Rowdy
0.96
Classifier w/ 91% Validation: Rowdy
0.72
Classifier w/ 85% Validation: No Rowdy
0.68
50
100
150
200
20 40 60 80 100 120 140 160 180 200 220

Figure 2: Certainty of the classifiers of different validation accuracies. Each classifier is subjected to the same image, labeling the image of either having Rowdy in the picture or not. The number under each label state each classifiers certainty.

4. DISCUSSION

The classifier is robust in identifying images that are closely related to the synthetic data, however it has limitations on classifying with textured backgrounds as we trained on a single-color background. Adding more variability in the background, texturing and patterns, would make the classifier adapt better to backgrounds. There are also difficulties in misclassifying real images with distraction objects whose shapes are exceptionally more complex than spheres and cubes (Figure 4). Because we used two simple shapes with no textures for distraction,

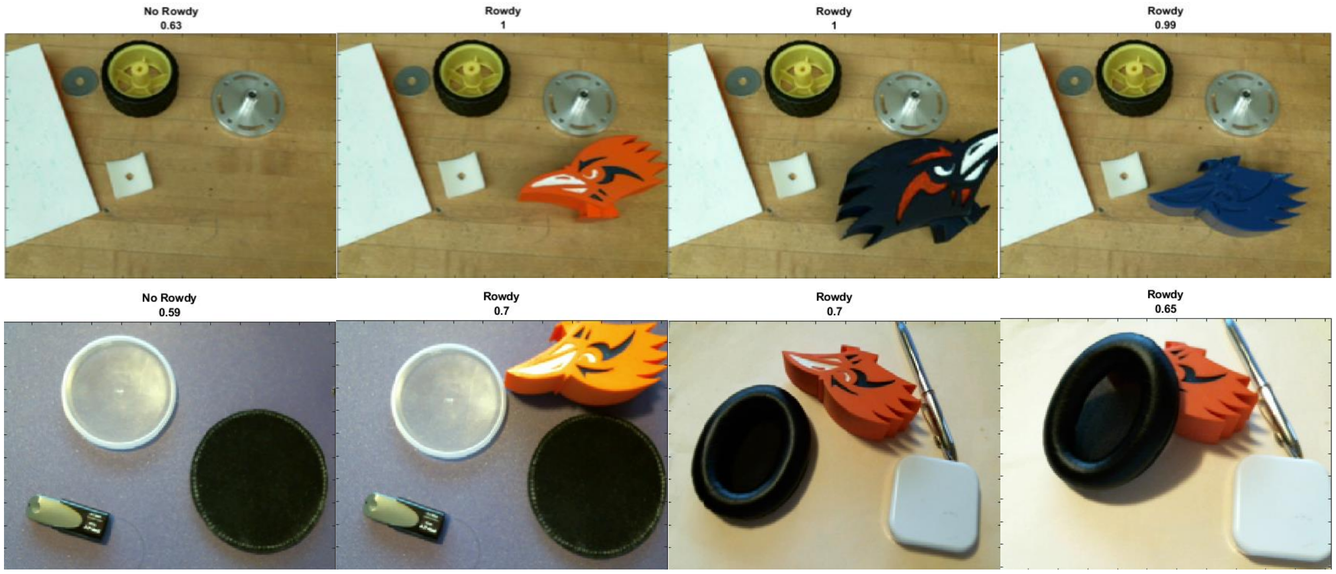

Figure 3: Classifier tested on real images

perhaps our we need to add new shapes and add textures as well to help with the classifier from overfitting.

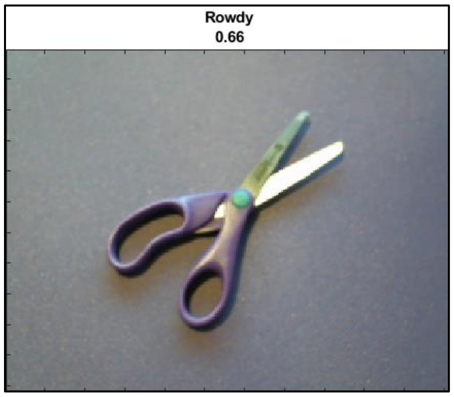

Figure 4: False Postive detection of Rowdy

Evaluating the classifiers of different validation accuracies, we showed that the classifier with the lowest accuracy to be the best at classifying real images—even then the classifier is not perfect. We speculate this behavior to be that there is not enough randomness in generating the images. We understand that if the training data is too simple, the neural network will tend to become bias.

5. CONCLUSION AND FUTURE WORK

This paper demonstrated that training a neural network on strictly synthetic images to classify real images is possible. We see in making randomly generated images a certain threshold of complexity is needed to properly use domain randomization.

For future works we will add more elements to randomize in each scene to increase the complexity. In exploring how complex a scene would need be to train a neural network would give a threshold or a standard to using domain randomization.

Using domain randomization for closing the reality gap between simulation and the real world is a promising tool for neural networks. Moving forward we hope to capitalize on the potential domain randomization has and improve general robotics.

REFERENCES

[1] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. arXiv preprint arXiv:1709.07857, 2017.

[2] Xingchao Peng, Baochen Sun, Karim Ali, and Kate Saenko. Learning deep object detectors from 3d models. In Proceedings of the IEEE International Conference on Computer Vision, pages 1278–1286, 2015.

[3] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," arXiv preprint arXiv:1703.06907, 2017

[4] MathWorks,"Pretrained Convolutional Neural Networks" www.mathworks.com/help/nnet/ug/pretrained-convolutional-neural-networks.html