Robotic Grasp Quality Prediction using Machine Learning

Carnegie Mellon University
Mechanical Engineering Department
Ardalan Tajbakhsh
Lakshmi Maddirala
Rachit Mishra
Sumedh Vaidya

Abstract

In recent years of rapid machine learning growth, there have been numerous robotics applications that leverage machine learning algorithms as an alternative approach to control of robots. Specifically, Robot manipulation is one sub area where machine learning could be applied for a robust grasp of random objects. Improving the quality of robotic grasp tasks will be essential in high volume factory automation. The Quality of Grasp is especially important in the efficiency and speed of pick and place tasks. The goal of this project was to design a machine learning algorithm that accurately predicts the quality of a robotic arm grasp. The challenge of learning this dataset was excessive number of features, large dataset size, and explainability of grasp performance in terms of robot parameters.

1. Introduction

Dataset of UR10 Robotic Arm states (Joint positions, velocities, and torques) was produced from ROS Simulation in Gazebo (Smart Grasping Sandbox). Once the object was grasped, the arm performed series of random movements, and the quality of grasp was measured as the variation of distance between palm and object after grasp was completed. The data was then trained on MLP, Random Forest, Light GBM, and SVM algorithms with parameter optimization to identify the optimal classifier. Furthermore, various feature engineering methods were implemented to boost the test accuracy of high performing algorithms.

2. Related Work

Most of the recent work in robot grasping techniques and grasp quality has been data driven and are generally divided into objects that are known, familiar, or unknown [2]. Learning is generally handled using Convolutional Neural Networks [3, 4]. In some cases, two separate CNN's, one used for pushing a dense cluster of objects to separate them, and the other to grasp and predict object quality are used.[1].

In reinforcement learning approach, robots can collect training data for grasping in a self-supervised manner by attempting thousands of random grasps and observing whether or not they result in successful grasps [5, 6, 8]. The success and failure can also be judged by shaking the object and measuring the change in position of the object relative to the palm of the end effector [1].

With the rapid development and increasing complexity of simulation software, it is significantly easier, faster, and less costly to collect grasping data from simulations instead of using actual robots [9]. Using simulators, it is possible to obtain data for different working environments that could be difficult to get otherwise. On the down side, learning on simulation data often suffers from the simulation-to-reality (sim2real) gap, where the visual appearance and object dynamics of simulated data deviate from their real world counterparts. Methods for overcoming the sim2real gap include domain randomization [9] and domain adaptation [10]. For grasping, the sim2real gap can be reduced by using only depth images for the visual input. Depth-only grasping methods used in [3] do not require further fine-tuning or domain adaptation to perform well in the real world.

In Zeng et al [1] have demonstrated the effectiveness of model-free reinforcement learning to discover complex synergies between non-prehensile (e.g. pushing) and prehensile (e.g.grasping) actions: pushing can help rearrange cluttered objects to make space for arms and fingers; similarly, grasping can help displace objects to make pushing movements more precise and collision-free.

In this project, grasping dataset available on Kaggle was used to determine the robustness of robotic grasp using shadow grasper. The separating factor between this project and the above-mentioned counterparts is the focus on gaining valuable insight from a grasp example

3. Data

The data was collected from ROS Simulation in Gazebo for a UR10 Robotic Arm. The end effector of the UR10 Robotic Arm has three fingers. Each finger is composed of three joints. The dataset contains 27 features and 900,000 data points that summarize position, velocity, and effort (torque) for each joint. The Distance Variation of palm and object after the grasp has been stabilized is measured as Robustness.

3.1. Simulation

The simulation was given in ROS with Gazebo environment. The platform was a common UR10 robot arm with shadow 3-finger hand as the manipulator. In each cycle, the object is placed at a random location within the workspace. An RGB camera finds the centroid of the object and sends the coordinates to the arm controller. The arm then executes the pick action through ROS motion planning interface. Once the object is grasped, the arm starts a random sequence of movements in the air and measures the variation between palm and object. Based on prior empirical relationships, this variation can be mapped to a value representing "Grasp Quality". Furthermore, robot state information is collected after each pick place action is performed.

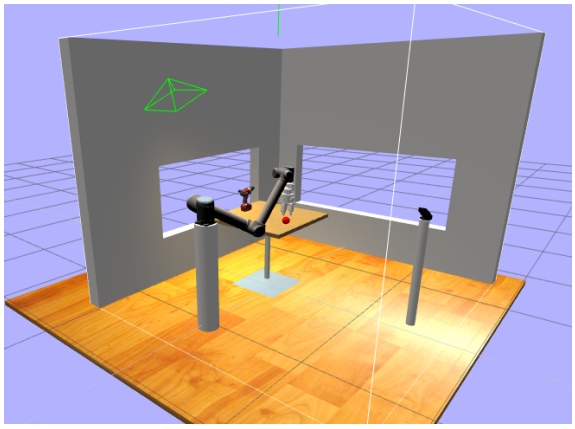

Figure 1. Simulation of UR10 in Gazebo

3.2. Data Pre-processing

This dataset was collected based on a ROS grasping simulation for a particular object. To further understand the labels, robustness vs measurement of each experiment has been plotted as shown in fig2. The Robustness number ≥ 100 is considered as good grasp and Robustness number $\lesssim 100$ is considered as bad grasp. This is essentially converting our problem to binary classification.

In order to generalize the data to random objects and various shapes, the position features for all fingers were omitted. Thus, the number of features were reduced to 18. Due to the massive size of dataset, a random subset of 10,000 data points with equal distribution of "good" and "bad" grasps was extracted. This enabled faster iteration time during training while maintaining a representative number of samples. Furthermore, due to variable range of values in different features, the dataset was normalized with mean of zero. This prevented features with larger magnitudes to dominate the learning process.

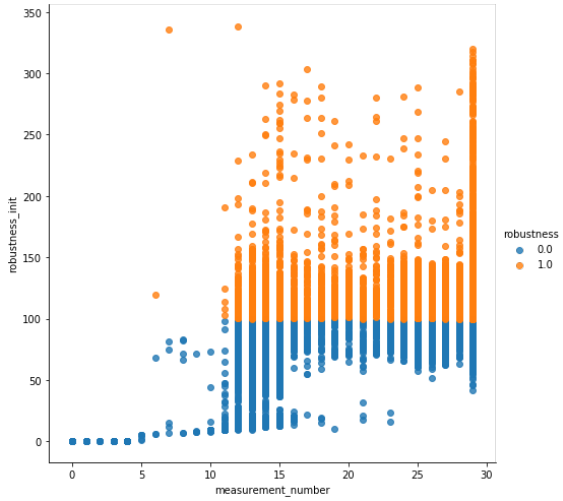

Figure 2. Robustness vs Measurement number

Furthermore, feature engineering was explored by adding two additional features to enrich the dataset. These were the sum of squared values of the velocity and effort of joint number three (the closest joints to the object) for all the three fingers. These features were selected intuitively from domain-specific literature.

On both the full and reduced dataset, Principal Component Analysis (PCA) algorithm was applied to identify the features that contribute the most to the grasp quality. The results suggested that more than 90% of the data can be represented by the features two fingers in a three finger grasp. It was also observed that the joints closest to the grasped object contributed the most to the grasp quality. This made intuitive sense as the third finger is "almost redundant" in the presented grasp configuration. The following figure demonstrates the variance explained by each feature in the dataset.

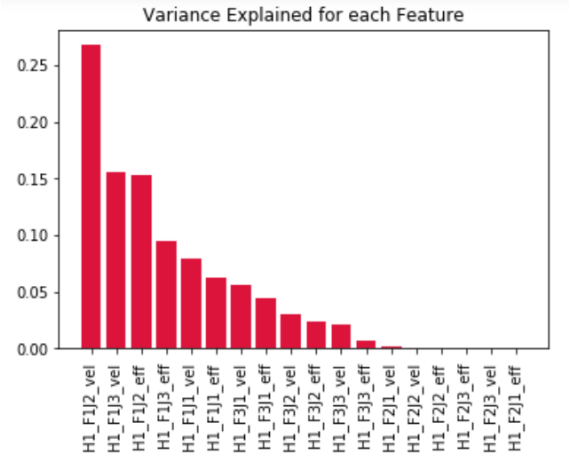

Figure 3. Variance of data using PCA

4. Methods/Algorithms

In order to gain insights from the presented dataset, various supervised learning algorithms were explored. This includes, Random Forest, Light GBM, Support Vector Machine (SVM), Deep neural network, Logistic Regression, and NN + Light GBM. In applicable cases, hyperparameter optimization was performed to further boost the accuracy.

4.1. Random Forest

Random forest algorithm consists of a large number of individual decision trees that operate on random subsets of data. Each individual tree in the random forest outputs a class prediction and the class with the most votes becomes the model's prediction. Effect of number of estimators on accuracy was explored, and the following plot summarizes the results.

As shown in the figure below, the best accuracy achieved was 93.84% for 90 estimators.

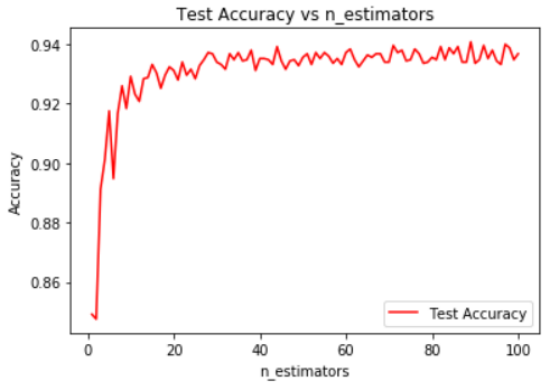

Figure 4. Test accuracy vs n-estimators for random forest

4.2. Light GBM

Light GBM is a gradient boosting framework that builds on tree-based learning algorithm. Unlike Random Forest which grows trees horizontally, Light GBM grows trees vertically i.e it grows trees leaf-wise. The leaf in the direction of the maximum delta loss will be grown. When growing the same leaf, leaf-wise algorithm reduces more loss than a level-wise algorithm. As compared to Random Forest, Light GBM runs faster on larger datasets and consumes less memory. This made it an ideal candidate for this massive dataset.

Figure 5. Leaf-wise tree growth diagram in LightGBM

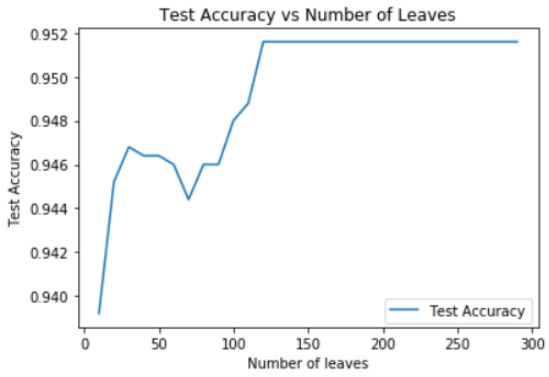

Figure 6. Test accuracy vs number of leaves for Light GBM

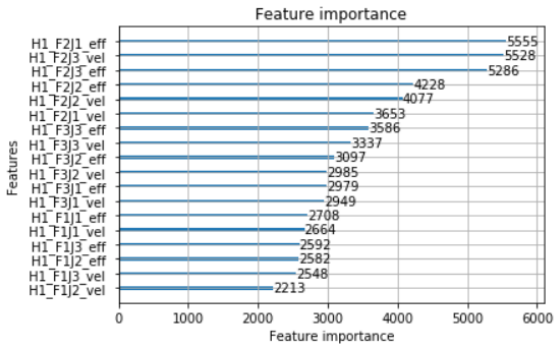

Figure 7. Importance of each feature in LightGBM

The hyper-parameters in Light GBM are the number of iterations and the number of leaves. For this mode, it was determined that the optimal values of these

hyper-parameters were 1500 and 150 respectively, while the highest test accuracy achieved was 95.16%

4.3. Neural networks

Neural networks are a set of algorithms, modeled loosely after the human brain consisting of multiple layers of neurons that are designed to identify patterns. They interpret and transform sensory data until they can classify it as an output. Each neuron multiplies an initial value by some weight, sums results with other values coming into the same neuron, adjusts the resulting number by the neuron's bias, and then normalizes the output with an activation function. A neural network generally consists of an input layer, multiple hidden layers and an output layer. This allows neural networks to fit all the nooks and crannies of our data including the nonlinear parts. The ability of the back propagation process to efficiently and optimally set the weights and biases of each model lets the neural network to robustly "learn" from data in ways that many other algorithms cannot.

4.3.1 Hyperparameters

The parameters that can be tuned in a neural network are optimizer, loss function, number of neurons per layer, number of hidden layers, and the activation functions at each layer.

The hyper parameters that resulted in optimal test accuracy were:

1. Activation: Relu, Softmax (output layer)
2. Optimizer: Rmsprop
3. Loss Function: Sparse Categorical Cross-entropy
4. Number of hidden layers: 4

4.3.2 Accuracy

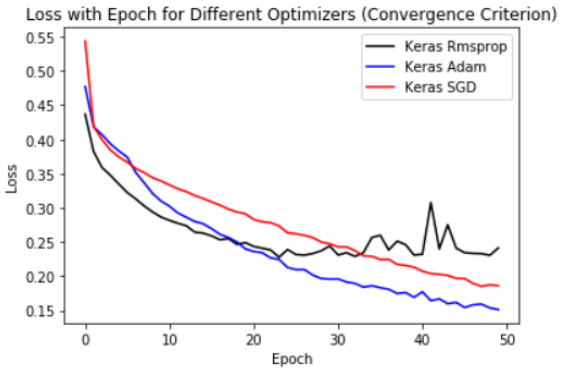

Figure 8. Loss vs Epoch in Keras for different optimizers

Highest accuracy achieved using neural network was 91.8%. This result was obtained when the dataset was complemented with additional features, including squared sum of the velocity and effort of joint 3 of each finger.

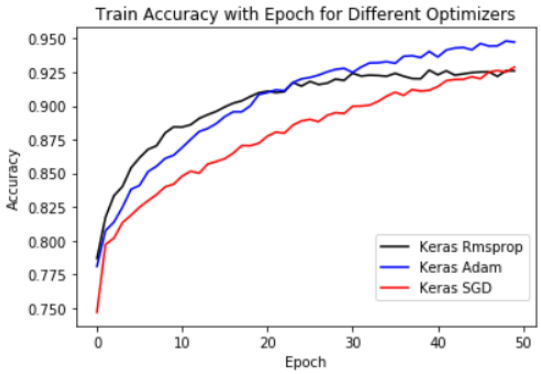

Figure 9. Accuracy vs epoch in Keras

4.4. SVM

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm finds an optimal hyperplane that separates the data into different classes. In two dimensional space, this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

4.4.1 SVM Hyperparameter Discussion

This section explains the effect of tuning and optimizaing hyper parameters in for support vector machine (SVM).

kernel:

Given the nature of this dataset, it was very unlikely for the data to be linearly separable. As a result, various kernels were applied to separate the non-linear data. Essentially, the kernels map the data to a higher dimensional space where it can be separated with a hyper-plane. In this case, linear, sigmoid, polynomial of degree 3, and RBF kernels were implemented. RBF kernel performed best at about 85% accuracy.

Regularization (C Parameter):

The Regularization parameter (often termed as C parameter) measures the importance of misclassifying data points. For large values of C, the optimization will choose a smaller-margin hyperplane providing that it does a better job at minimizing number of misclassified points. Conversely, a relatively small value of C will cause the optimizer to find the widest gutters at the cost of misclassifying more points.

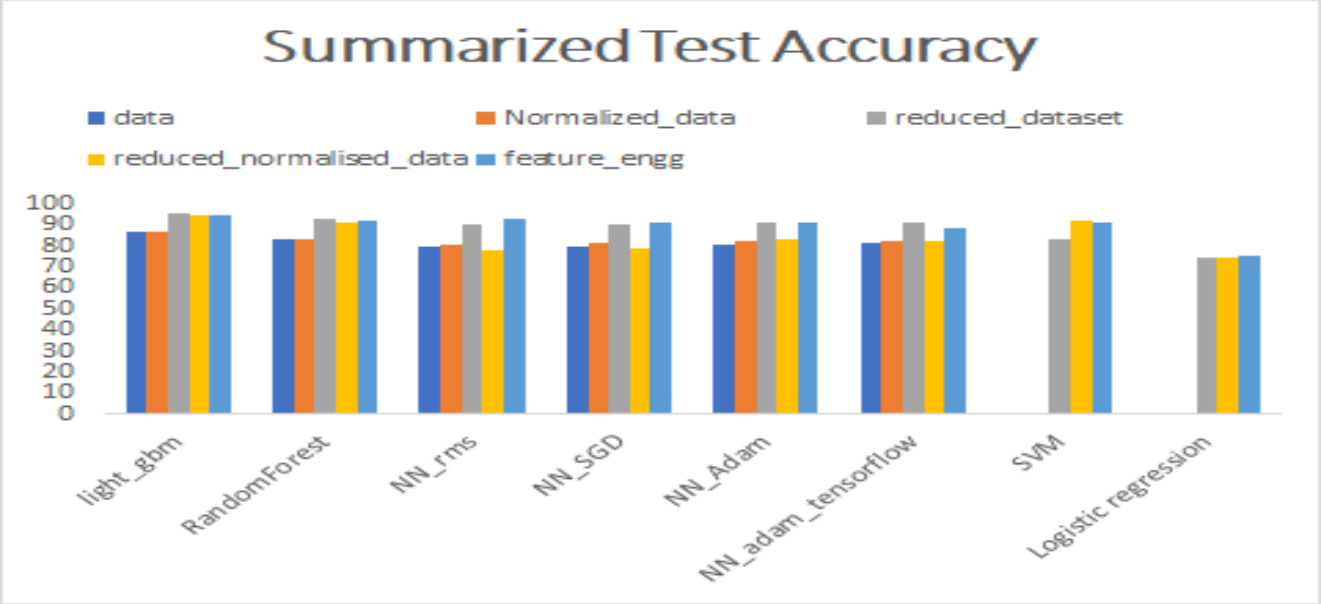

Figure 10. Summarized test accuracy plots for all experimented models

Decision Boundary Stiffness (Gamma):

The gamma parameter determines the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line and vice versa.

Parameter Optimization:

Ultimately, the goal was to find the optimal set of parameters that maximizes test accuracy without overfitting the training data. To accomplish this, a grid search optimization algorithm was performed on the following discrete combination of parameters for the RBF kernel. Gamma = [0.001, 0.01, 0.1, 1, 10], C = [0.1, 1, 10, 100].

Using the RBF kernel, gamma = 1 and C = 10 were the optimal parameters with test accuracy of 92%

From the above graphs we can also infer that in neural networks, sometimes the accuracy of the train data may increase even when the loss is increasing. This is because of mini-batch gradient descent. Also, since this loss eventually goes down, we can conclude that the model has not overfit the train data and regularization is not required.

The highest accuracy of neural network was attained with Stochastic Gradient Descent (SGD) as the optimizer at (90.56%). SGD was also superior to other optimizers in terms of training speed, which made it an ideal candidate for this massive dataset.

4.5. Neural Network + Light GBM

As an experiment to further improve the test accuracy, ensemble learning was applied. This was done by feeding the output of the neural network as input to Light GBM. During this iterative process, the first, third, and fifth layer of NN were used as the inputs. In all cases, this resulted in a significant decrease in the accuracy. The test accuracy with ensemble learning varied between 55% and 71%.

Throughout this process, it was understood that the combination of strong performing classifiers does not necessarily lead to better performance. In this specific, it hurt the test accuracy by a significant margin.

4.6. Logistic Regression

Logistic model is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. Logistic regression is used to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

The highest accuracy from logistic regression was 74.44%. This was achieved with reduced normalized dataset. Logistic regression normally fails when the data is highly non-linear and features are strongly correlated, which seems to be case here.

4.7. Conclusion

1. The accuracy of the Light GBM model was superior to other classifiers. The next best classifier was Support

Vector Machine (SVM), followed by Neural Network with SGD optimizer.

2. Normalizing and reducing the size of the dataset generally increased the test accuracy of the models. This was mainly due to eliminating the effect of variable ranges in dominating the learning process.

3. Feature engineering (adding sum of velocities and efforts of the last fingers squared as additional features) increased the accuracy, however the accuracy of the models did not improve further for subsequent feature additions.

4. Accuracy on full dataset was generally lower than the accuracy on the reduced dataset, however the reduced dataset was still representative with 10k samples.

5. All proposed models were validated using 10-fold cross validation, and the uncertainty was bounded to less than 5%.

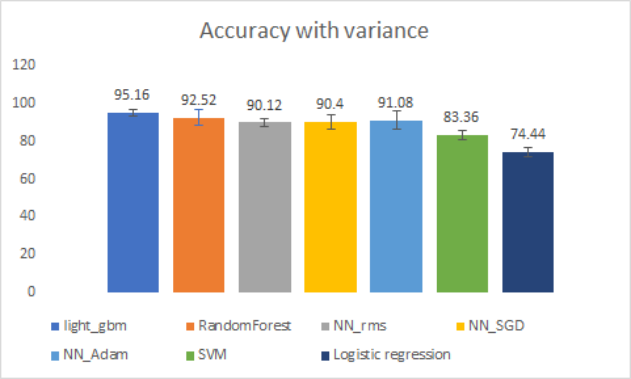

Figure 11. Accuracy with variance using 10-fold cross validation

Acknowledgement

We would like to thank professor Amir Barati Farimani, Mechanical Engineering, CMU for his guidance and support with this project. We would also like to thank Malhar Bhoite, TA,24-787, Mechanical AI Lab for his suggestions that resulted in improvements throughout this project.

References

[1] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, Thomas Funkhouser *Learning Synergies between Pushing and Grasping with Self-supervised Deep Reinforcement Learning*. IEEE International Conference on Intelligent Robots and Systems (IROS), 2018. 1

[2] Jeannette Bohg, Antonio Morales, Tamim Asfour, Danica Kragic *Data-Driven Grasp Synthesis - A Survey*. IEEE Transactions on Robotics 30(2):289–309 1

[3] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, Ken Goldberg *Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics*. Robotics: Science and Systems (RSS), 2017 1

[4] Andy Zeng, Shuran Song, Kuan-Ting Yu, Elliott Donlon, Francois R. Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald Romo, Nima Fazeli, Ferran Alet, Nikhil Chavan Dafle, Rachel Holladay, Isabella Morona, Prem Qu Nair, Druck Green, Ian Taylor, Weber Liu, Thomas Funkhouser, Alberto Rodriguez *Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching*. IEEE International Conference on Robotics and Automation (ICRA), 2018 1

[5] Kalashnikov D, Irpan A, Pastor P, Ibarz J, Herzog A, Jang E, Quillen D, Holly E, Kalakrishnan M, Vanhoucke V, et al *Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation* Conference on Robot Learning (CoRL), 2018 1

[6] Levine S, Pastor P, Krizhevsky A, Ibarz J, Quillen D *Learning handeye coordination for robotic grasping with deep learning and large-scale data collection* The International Journal of Robotics Research 37(4-5):421–436, 2018 1

[7] Saxena A, Driemeyer J, Ng AY *Robotic grasping of novel objects using vision* The International Journal of Robotics Research 27(2):157–173, 2008

[8] Pinto L, Gupta A *Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours* IEEE international conference on robotics and automation (ICRA), pp 3406–3413, 2016 1

[9] Tobin J, Biewald L, Duan R, Andrychowicz M, Handa A, Kumar V, McGrew B, Ray A, Schneider J, Welinder P, et al *Domain randomization and generative models for robotic grasping* IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp 3482–3489, 2018 1

[10] Bousmalis K, Silberman N, Dohan D, Erhan D, Krishnan D *Unsupervised pixel-level domain adaptation with generative adversarial networks* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017 1