ÉCOLE CENTRALE DE NANTES

MASTER CORO-IMARO
"CONTROL AND ROBOTICS"

2018 / 2019

Thesis bibliography report

Presented by

Ashok Muralidharan

21/01/2019

Training Data Generation for Deep Learning based Robotic Vision Systems

Jury

President:	Olivier KERMORGANT	Assistant Professor (ECN, LS2N)
Evaluators:	Olivier KERMORGANT	Assistant Professor (ECN, LS2N)
	Ina TARALOVA	Assistant Professor (ECN, LS2N)
Supervisor(s):	The Duy NGUYEN	Lead Developer - AI (Gestalt Robotics)
	Olivier KERMORGANT	Assistant Professor (ECN, LS2N)

Abstract

Neural Networks based techniques are currently becoming the state of art methods for performing several computer vision tasks like image classification, localisation, segmentation, etc. But to train these neural networks, we require a large amount of labelled training dataset. This acts as a hindrance for the use of these techniques in the fields like robotics since they deal with a very particular domain and labelled training datasets are simply not for that specific domain. This leads to the investigation and use of artificial image generation techniques. Direct use of synthetic images from simulators are proven to perform not so well since they are not from the same domain as like the real world images. The domain shift can be thought as a distribution change that is caused due to various factors like illumination, pose, etc. Techniques like Domain Adversarial Training[5] have proven to learn the invariant features of both the simulation and real-world domains and uses them for computer vision tasks. Latest techniques like Generative Adversarial Networks(GAN)[7] can be used to produce photo realistic images from the given image dataset distribution. Also, there are some recent advances that make use of 3D model information of the object to generate images along with the generative model approaches. Thus the goal of this thesis work is to study the state of art image generation techniques, implement and test them on real-world data for robotic vision based tasks like classification and localisation.

Acknowledgements

I would like to express my gratitude to my supervisors Prof. Olivier Kermorgant and Dr.-Ing. The Duy Nguyen for their continuous support and guidance in helping me prepare this bibliography report. I also would like to thank Prof. Ina Taralova for her guidance in helping me with formal construction of the report and for sharing her knowledge about effective research methodologies.

Abbreviations

ANN Artificial Neural Networks

CNN Convolutional Neural Networks

GAN Generative Adversarial Networks

PixelDA Pixel-Level Domain Adaptation

VAE Variational Autoencoder

List of Figures

Contents

Chapter 1

Introduction

Artificial Neural Networks(ANN) based techniques have become the state of art approach for a number of computer vision tasks. Convolutional Neural Networks(CNN) are such neural network based architecture which has proven to work well with images in computer vision tasks[10]. Cameras are the popular vision sensors that are used for perception in robotics. Thus the robotic vision seems a direct application candidate for using neural network based techniques. But the main problem with neural network based approaches are that they are data hungry and they require large amounts of labelled samples for training, in our case large amount of labelled images. Generally the images that a robot sees are not available in abundance to us(i.e) it is tightly coupled to its domain. It is very hard to find a labelled training image data set that is similar to what the robot sees in its application.

One solution to this problem would be to manually take pictures in its domain(as like how the robot sees) and label them. But this process is very time consuming, requires a lot of human effort and money. Since the images taken during this process are actual images that a robot might face during its task, these images remain in the similar domain as in during the operation of the robot during the task. Hence the images generated during this process are perfect for training the network. But labelling involves human annotation and it can be prone to errors. Also in tasks like image segmentation, it is hard to label every pixel in the image manually and this solution does not scale well in the long run.

Another solution to this problem can be to use the data/images from the simulators for training the networks. This way we can generate huge collection of training dataset with minimal effort. Also the annotation of data is automatic and is perfect. But this approach is proven to not perform well [13] with the real world image since the data originates from a different domain(i.e) the data comes from simulation which misses many information from the real world image.

Thus we need to find a way to efficiently use this simulated image/model to train the neural networks. So the problem in hand is to generate synthetic labelled training images/datasets that can be used for training the neural networks for several computer vision tasks for robotics based application especially in classification and localisation tasks.

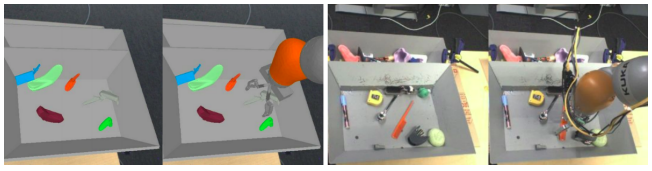

Figure 1.1: Image from Simulation and Real World [3]

Chapter 2

State of the art

2.1 Potential Techniques

Now we investigate the different state of art approaches that are used for synthetic data generation. There are also techniques that learn domain invariant features and uses these features for computer vision task. We start by looking at the generative models, especially Generative Adversarial Networks(GAN). We then briefly discuss about the domain adversarial training of neural networks [5], which are used to make the models resilient to domain invariance. We finally discuss about the recent advances in the GAN based approaches for synthetic data generation for robotic tasks.

2.2 Generative Models

A generative model defines how the data is generated, based on the probabilistic model. If the observed data is X and the corresponding label is Y , the generative model estimates the joint probability distribution $P(X, Y)$ between between them. This distribution can be used to generate likely (X, Y) pairs. The generative models that we investigate are Bayes Classifier, Variational Autoencoder(VAE), Generative Adversarial Networks(GAN).

2.2.1 Bayes Classifier as Generative Model

First we start of by understanding and looking at Bayes classifier as a generative model. We know the Bayes theorem,

$$P(Y|X) = \frac{P(Y)P(X|Y)}{P(X)} \tag{2.1}$$

As stated earlier, the generative model tries to capture the $P(X, Y)$ where X is the data and Y is the label. For the case of Bayes classifier as a generative model, we are interested in $P(X|Y)$(i.e) the probability of X given Y. For example, if X represents the image and Y represents the corresponding label, then we need all $P(X|Y = y_i)$ where $i = 1..N$(total number of images). Thus for an image we try to model the probability of every pixel value given the label. Let us see an example of image generated from MNIST dataset[11]. The figure shows the generated image from Bayes classifier used as Generative model. Here, we consider that all the data follows gaussian distribution for simplification purpose. We find the mean and covariance of all the pixel values of the images of given label(i.e) we collect all the mean and covariance of all the labels. We can then sample image from $P(X|Y = y_i)$.

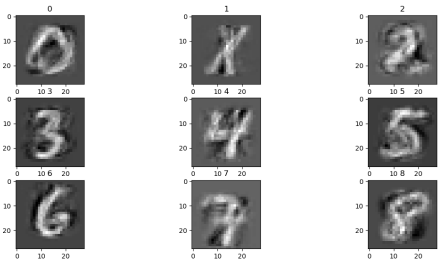

Figure 2.1: Generated Images by Bayesian Classifier

As we can see from the figure 2.1, the generated synthetic images are actually not bad for a simple bayes classifier. Also we made a assumption that all the data follows gaussian distribution. Advanced techniques with neural network used to learn the distribution have proven to yield better results.

2.2.2 Variational Autoencoder(VAE)

To get a clear understanding of VAE we first look at autoencoders. An autoencoder is a type of Artificial Neural Network used to learn efficient data codings or embeddings in an unsupervised manner. The autoencoder learns the representation of data typically in a reduced dimension. This reduced dimension data can be seen as the representation of original data and this part is called the encoder. This reduced dimension data can be expanded or can be used to retrive the original compressed data. This part of the network is called the decoder. Thus the autoencoder has an encoder and a decoder network working together to learn the reduced dimension representation or the latent representation. The figure below 2.2 should give a clear picture of the process. The auto encoder can be used for image compression and extraction processes.

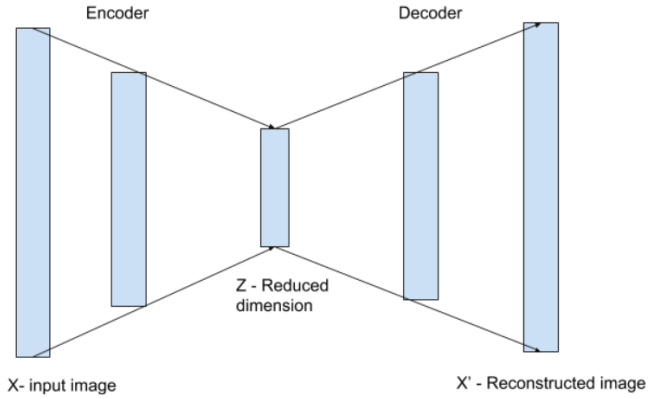

Figure 2.2: Auto Encoder - Block Diagram

The VAE [9] and [12] draws ideas from the auto encoder in its process. It is actually very similar to the autoencoder but is a generative model. To make the auto encoder a generative model, we add an additional constraint to the encoder part of the network to produce the latent vectors to follow a Gaussian distribution. The VAE can now generate images similar to the one that were trained from the latent vectors. We need to make the network vector variable close to the unit gaussian distribution. The diagram below 2.3 should given a idea about how this works.

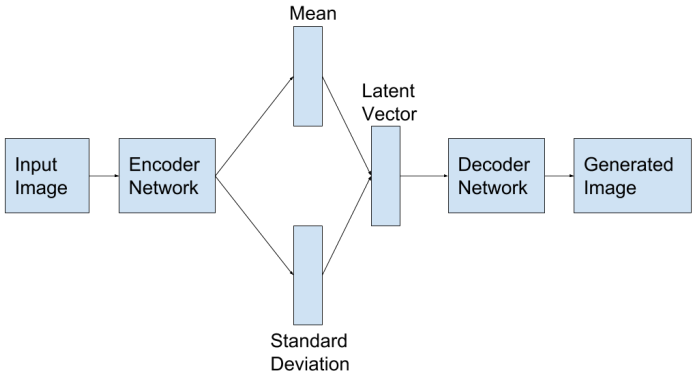

Figure 2.3: Variational Auto Encoder(VAE) - Block Diagram

Let x represent the input image, z represent the latent vector. The encoder part 2.4 is a neural network that learns a approximation $q(z|x)$ for the actual $p(z|x)$. In practice, the $p(z|x)$ is intractable with large dimensions of z. Thus in VAE we consider the $p(z|x)$ to be a gaussian distribution with mean 0 and variance of 1.

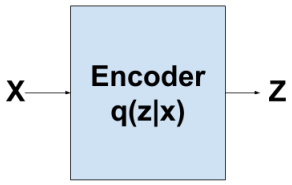

Figure 2.4: VAE - Encoder

The decoder part as seen in figure 2.5 is used to generate the probability distribution of the data x given the latent variable z(i.e) the decoder is represented as $p(x|z)$. $p(x|z)$ can be intutively thought as the reconstruction of the image dependant on the latent vector.

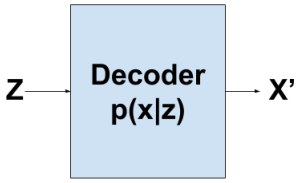

Figure 2.5: VAE - Decoder

The loss function that is used in VAE is given by

$$L = -E_{q(z|x)}[\log p(x|z)] + KL(q(z|x)||p(z)) \tag{2.2}$$

where $KL(q(z|x)||p(z))$ is the Kullback-Leibler divergence between the two distributions. This can be intutively thought as the measure of difference between the actual $p(z)$ and the encoder output $q(z|x)$. This is measure of how close is q to p. The first term $-E_{q(z|x)}[\log p(x|z)]$ is nothing but the expectation of $\log p(x|z)$. This term is the expected negative log likelihood.

This can also be seen as the reconstruction loss. The figure below 2.6 shows some images that were generated by VAE.

Figure 2.6: Variational Auto Encoder(VAE) - Generated Images [6]

2.2.3 Generative Adversarial Networks(GAN)

Generative Adversarial Networks(GANs)[7] are the current state of the art techniques for synthetic image generation. GANs are capable of producing photo realistic images from the given images distribution. The block diagram for GAN can be seen in the figure 2.7

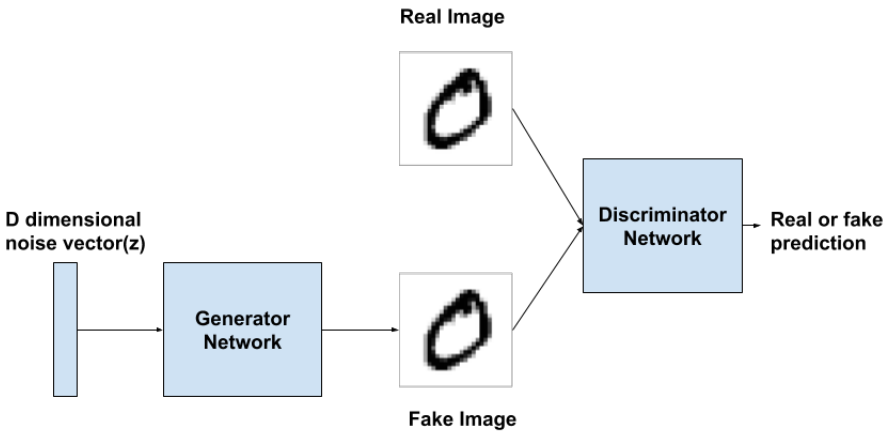

Figure 2.7: Generative Adversarial Networks - GAN

The GAN consists of a generator network and a discriminator network. As we can see from the figure 2.7, the generator network which is basically a neural network based architecture, takes in a latent variable z, which is a d dimensional noise vector. Then the generator through a number of hidden layers tries to generate the fake image. The real image distribution is available to us(say MNIST dataset). The job of the discriminator network, which again is a neural network, is that it has to detect the real image from the fake images. Since it is based on the neural network based architecture we can compute a *differential loss function* and back-propagate the gradients to tune the parameters of the networks.

In GAN the aim of the generator is to produce images that are similar to the images from the dataset, while the objective of the discriminator is to identify the fake image from the real image. As we can see, both the networks are trying to optimize a opposing objective and they play a zero-sum game. Let us define the loss function for the GAN formally,

The generator is defined as

$$x = g(z; \theta_g) \tag{2.3}$$

where z is the input noise vector and θ_g is the training parameters of the generator model

The discriminator is defined as

$$d = d(x; \theta_d) \tag{2.4}$$

where x is the real or the fake image and θ_d is the training parameters of the discriminator model.
The training can be done by using a mini-max objective function whose solution is defined by Nash equilibrium in game theory for non-cooperative games. Thus for our case, we are looking for optimal parameters (θ_g, θ_d) that would lead to this equilibrium at which the discriminator will not be able to distinguish the real and fake image. We can imagine the discriminator should output with equal probabilities(0.5) in its prediction for both real and fake classification. Thus we can define the mini-max objective function as

$$v^* = arg \min_g \max_d v(g, d) \tag{2.5}$$

For GAN, the function is defined as cross-entropy loss function given below,

$$v(\theta_g, \theta_d) = E_{x \sim p_x}[\log d(x)] + E_{z \sim p_z}[\log(1 - d(g(z)))] \tag{2.6}$$

Thus given the generator g, we try to maximize the above function 2.6 and given the discriminator d, we try to minimize the same function. Thus GAN plays a mini-max game with generator and discriminator as its players. Finally when the sufficient training is done, we can remove the discriminator part and use the generator part to generate new images from similar to the dataset on which it was trained on. The figure below 2.8 shows some sample images generated by the GAN.

Figure 2.8: GAN- Generated Images [6]

As we can see that the images produced by GAN are much more realistic and sharp than images generated by VAE.

2.3 Domain Adaptive Training

Domain adaptation is not a generative model or a image generation method. But the idea of domain adaptation is very crucial to our study because our aim is to synthesize artificial image or features that resembles the original images with its distribution. Domain adaptation is a technique that lets us learn this domain shift and correct them. The idea of domain adaptation is quite popular in machine learning. The key idea of domain adaptation is to use the network trained in one domain with some source data distribution for a different domain with different target data distribution. The transfer learning used in the training neural networks is closely related to the domain adaptation. For instance, in transfer learning, a neural network trained on highways dataset(source domain) can be used for to train a neural network with city dataset(target domain). Domain Adaptation shares similar ideas like transfer learning.

In Domain-Adversarial Training of Neural Networks paper [5], the key idea is to extract the domain invariant features from both the source and the target distribution and use them for given task(say classification). In this technique, the neural network architecture is trained both on source domain with labels and target domain without labels. As the training progresses, the domain invariant features emerges. A simple block diagram is shown below,

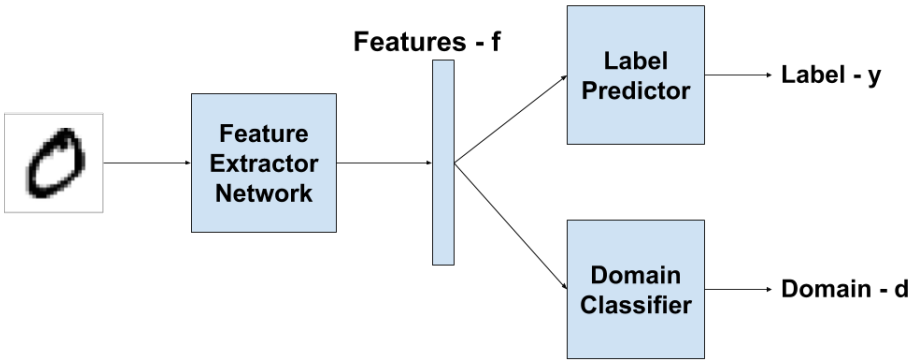

Figure 2.9: Domain Adversarial Training of Neural Networks - Block Diagram

As we can see from the block diagram, the input to the feature extractor is the image - either the simulated/synthetic images with labels(source domain) or the real images without labels(target domain). The feature extractor extracts the features of the supplied image to a vector. This vector passes through a domain classifier that classifies whether the given image is from the synthetic domain or the real domain. Also there is a label predictor network that classifies the label y of the supplied image. Using these two predictions, a loss function can be synthesized and backpropagation of errors can be used to train all three networks.

Now let us try to formalize the method and express its objective function. Let the $G_f(x, \theta_f)$ be the generator with x the input image and θ_f its tunable parameters. Let $G_y(f, \theta_y)$ be the label predictor network where f is the feature vector and θ_y is its tunable parameters. Similarly let $G_d(f, \theta_d)$ be the domain classifier network where f is the feature vector and θ_d is its tunable parameters. The loss function can be seen as

$$L_y(\theta_f, \theta_y) = L_y(G_y(G_f(x, \theta_f), \theta_y), y) \tag{2.7}$$

$$L_d(\theta_f, \theta_d) = L_y(G_d(G_f(x, \theta_f), \theta_d), d) \tag{2.8}$$

The method involves in optimizing this function

$$E(\theta_f, \theta_d, \theta_y) = L_y(\theta_f, \theta_y) - \lambda(L_d(\theta_f, \theta_d)) \tag{2.9}$$

So we are effectively trying to find a saddle point $(\hat{\theta_f}, \hat{\theta_y}, \hat{\theta_d})$ such that,

$$(\hat{\theta_f}, \hat{\theta_y}) = arg \min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta_d}) \tag{2.10}$$

$$\hat{\theta_d} = arg \max_{d} E(\hat{\theta_f}, \hat{\theta_y}, \theta_d) \tag{2.11}$$

This way we can tune the network and finally use the feature extractor and the classifier part for our application. We can also utilize the feature extractor part seperately which would give us the domain invariant features.

2.4 Latest Advances using GANs

A number of research papers have emerged that uses GANs as a model for generating images for training neural networks. We focus on few of these papers that showed promises in the field of robotic vision. Particularly the papers that uses image generation techniques adapted with domain adversarial training.

2.4.1 PixelDA [2]

This paper[2] from Google is titled "Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks". This paper aims to generate source images as if they were drawn from the target image samples. The source domain, say for example is a simulated images of the objects for which we have the labels and target domain may be the images of objects from our real world domain. This method proposes a unsupervised approach(i.e) we donot train using corresponding pairs of images from both the domains. The key advantage of this approach is that it leverages the use of domain adaptation [5] at the pixel level of the image and thus it is not tied with the task at hand(i.e) in simple terms it is just a way to learn how to make the simulated images similar to the target images provided. Also since this method uses GAN, thus we can have virtually have unlimited samples that are similar to the target domain. A simple block diagram of this method is show below,

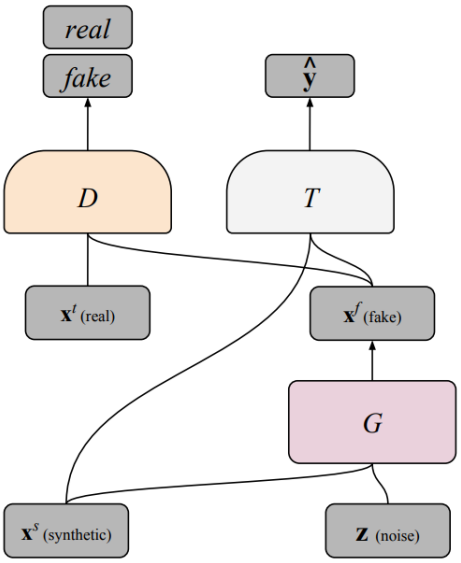

Figure 2.10: PixelDA[2]

As we can see from figure 2.10 the generator(G) which is actually a GAN, is conditioned not only noise vector(z) as in GAN paper but also on the synthetic image(x^s) to generate the fake image(x^f) similar to the target image (x^t). The fake and the real images are passed through a discriminator network which predicts whether the generated image is real or fake. Also there is another task specific network (T) that predicts the task specific labels($\hat{y}$) that is specific to a given problem.

Thus we can imagine that we will have two loss functions that we would together like to optimize on(i.e) the loss function from the discriminator network and loss from the task specific network. Thus our goal here would be to optimize,

$$\min_{\theta_G,\theta_T} \max_{\theta_D} \alpha L_d(D,G) + \beta L_t(G,T) \tag{2.12}$$

where α and β are the weighing parameter from the individual loss functions to control the interaction loss.

The paper also talks about a content similarity loss fuction along with the above two loss functions. This loss is basically added when we have prior knowledge regarding the low level image adaptation process. To avoid redundancy and have brevity of information, this report tries to explain the key idea behind the papers. Also an interesting part in paper is where they use the network with LineMod dataset [8], where they used the CAD models of 11 different objects in variety of poses rendered on a black background as a synthetic image information. Thus the new task based loss function has a pose information along with classification task(i.e)

$$T(x;\theta_t) -> (\hat{y},\hat{q}) \tag{2.13}$$

Where $\hat{y}$ is the classification prediction and $\hat{q}$ is the 3D pose estimation in the form of quaternion vector.

Thus the task specific loss will also contain the correction for quaternion vector along with the classification task.

$$E_{x^s,y^s,z}[-y^{s^T}\log\hat{y}^s - y^{s^T}\log\hat{y}^f + \epsilon\log(1-|q^{s^T}\hat{q^s}|) + \epsilon\log(1-|q^{s^T}\hat{q^f}|)] \tag{2.14}$$

As we can see, the first two terms are the for the classification loss. The third and the forth term are for the quaternion loss, where q^s represents the ground truth 3D pose sample, $\hat{q^s}$ represents the predicted 3D pose for the synthetic image directly from the CAD model and $\hat{q^f}$ represents the predicted 3D pose for the generated synthetic image. This this paper is a very interesting one for current thesis as it involves use of 3D information along with the image and also as an example usage for localisation task.

2.4.2 GraspGAN

This is yet another interesting paper from Google [3] where simulated environments and domain adaptation methods were used to train a grasping system to grasp objects from monocular RGB images. This paper was an extension of the previous method PixelDA. This paper is interesting to us because this paper evaluates the synthetic image generation model with non trivial task such as localisation and grasping in a robotic application. A good understanding of how the implementation served the purpose can shed some light during designing of image generation pipeline for training deep neural networks for robotic applications.

The main process in the paper had two parts - one a grasp prediction convolutional neural networks(CNN) and the other a manually designed servoing function that acts according to the grasp probabilities obtained from the first part. We focus on the grasp prediction CNN part. The grasp prediction CNN takes in (x_i, v_i) where v_i is the motion command and $x_i = x_{i_0}, x_{i_c}$, x_{i_0} is image when the robot starts the grasping attempt and x_{i_c} is the image at the current time step. This paper is a application of PixelDA which we discussed earlier in detail. We can already see where the image generation pipeline is required. We used image pairs in this network for training the deep convolutional neural network based grasp predictor. The images are used for this training are generated using a generator network. A figure from the original paper [3] is shown below

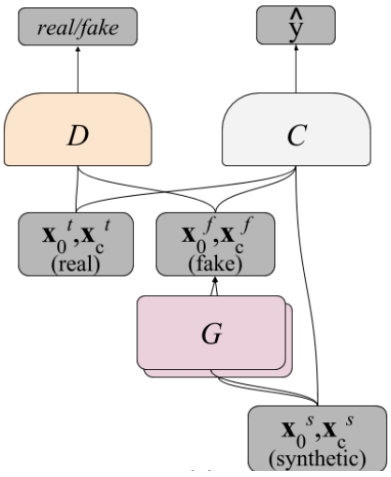

Figure 2.11: GraspGAN[3]

The generator network G and discriminator D used are convolutional neural networks(CNN). Here the key difference is the discriminator D, which is fully convolutional on three different scales of the two input images x_0^s nad x_c^s which are stacked into 6 channel producing domain estimates. They termed this discriminator a *multi-scale patch-based discriminator*. There is also a task network C which predicts the success of grasping. Thus we have the following loss function on which we optimize on with respect to parameters of each network.

$$\min_{\theta_G} \lambda_g L_{gen}(G, D) + \lambda_{tg} L_{task}(G, C) + \lambda_C L_{content}(G) \tag{2.15}$$

$$\min_{\theta_D,\theta_C} \lambda_d L_{discr}(G, D) + \lambda_{td} L_{task}(G, C) \tag{2.16}$$

where L_{gen} and L_{discr} are the GAN generator and discriminator losses respectively. L_{task} is the task loss and $L_{content}$ is the content similarity loss and all the λ's are weights to particular losses. The content similarity loss used here composes of multiple loss functions. During the time when generator tries to generate images similar to the real world images, the simulated image must not lose its semantics. The $L_{content}$ is such term that is used to anchor the generated image to simulated one on semantic level.

More details about the architecture of the networks and the results are detailed in the paper [3].

2.4.3 RenderGAN

This paper [14] uses a modified version of GAN that uses a 3D model of the data and also adds constraints to the image augmentation function. This paper used this idea for generating images of a bee marker tag. Although the application is different, the technique used for image generation is still of interest to us.

The approach uses a modified version of GAN which takes noise vector z as a input parameter. Unlike the traditional GAN which directly outputs the synthetic image, here the generator network generates parameters for the 3D model in hand. Then the image generated by 3D model with those parameters is augmented by a function ϕ_i, which is again parameterized by G_i from the generator. The image below should give an idea about the architecture,

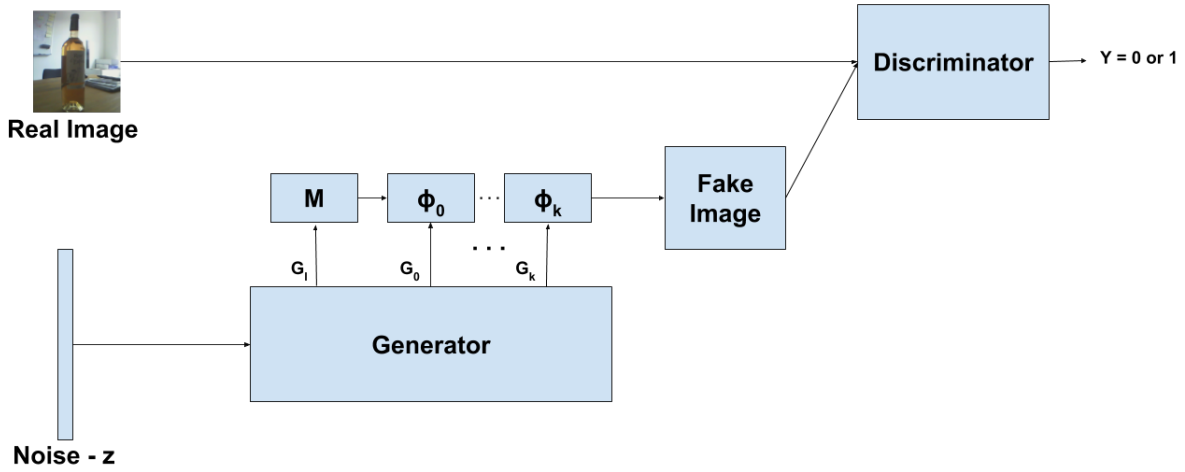

Figure 2.12: RenderGAN

As we can see from the figure 2.12, the input to the generator network is just the noise vector z. The 3D model is represented as M in the figure. The generator generates parameters G_0, G_1, .. G_k. The generator outputs the parameter G_l for the 3D model M and this output is fed to series of augmentation function ϕ_0 which has the parameter G_1 from the generator. There are number of augmentation function ϕ_k connected in series to generate the fake image as shown in the figure 2.12. The real image and this fake image are used by the discrimintor network to predict whether the supplied image is real or fake. We can write the generator function as,

$$g(z) = \phi_k(\phi_{k-1}(...\phi_0(M(G_l(z), G_0(z)))..., G_{k-1}(z)), G_k(z)) \tag{2.17}$$

In their paper, they have detailed four augmentation function (ϕ) namely bluriness, lighting, background and details. But here for brevity, we are not going in details about the augmentation functions. We can look to design and use other augmentation functions according to our application like colors, affine transformations, etc.

2.4.4 Geometric Image Synthesis

This paper[1] proposes a method that leverages the image generation pipeline by being geometry aware (i.e) by using geometry and segmentation information for generating photorealistic images that matches the desired scene structure. This method generates geometrically consisitent images from limited input information like segmentation, normal and depth, while the remaining aspects of the image like lighting conditions, etc. are learned by the network. Traditionally, the 3D models are rendered into natural images using appropriate blending. But it is very hard to model all the physical processes like lighting conditions, etc. and blend them. It remains time consuming and a non-trivial process. Thus this method proposes this blending process which maps the intermediate representation of the scene to the desired output. The network representation diagram from the original paper [1] is shown below,

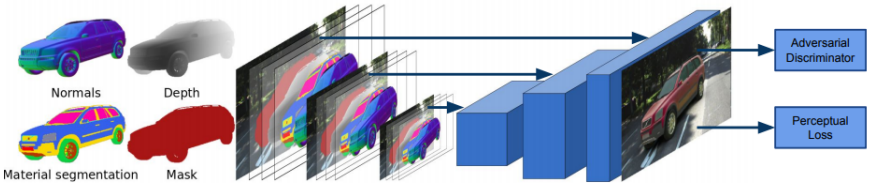

Figure 2.13: Geometry Image Synthesis(GIS)[1]

The image generation process by this method can be defined as a mapping from $(G, S) -> I$, where G is the given geometry and S is the material segmentation. We donot have the appearance information and the environment parameters, which are actually learned by the model. More formally, the mapping can be represented as $(N, D, S, M, I_{bg}) -> I$, where N, D, S are the normal map, depth map and the segmentation map of the input object. M is the material label denoted at each pixel and I_bg is the background image that is provided and I is the generated image.

The training process here involves to produce synthetic images I_s similar to the target images I_t which are obtained by "cycles" renderer and at the same time being close to real appearance in order to fool the adversarial discriminator. The paper describes about a perpetual loss(feature matching) function which is the loss between I_s and I_t. There are two more loss functions for the interaction with background image L_g and the adversarial discriminator loss function L_A which are detailed in the paper. The network architecture and the specifics are detailed in the original paper. Here we try to understand and grasp the key features that the paper brings on to our interest.

2.5 Summary of various approaches

As we have seen, each paper had its own approach to the problem at hand. We summarize the important approaches that we saw till now for easier reference in the future in the table below.

Paper	Core idea	Comments
Variational Autoencoders(VAEs)	Maximize a lower bound on the log likelihood of the data. Explicit modelling of latent space.	Easier to train and roboust hyperparameter settings. Easier to evaluate the quality of the model. Images produced are blurry due to the addition of noise.
Generative Adversarial Networks(GANs)	Two neural nets(generator and discriminator) contesting in zero sum game in which generator tries to generate fake image and discriminator detects the fake image.	Produces photo realistic(visually appealing) images. Sensitive to hyperparameter settings. Difficult to train. Stability issues(mode collapse)
Domain-Adversarial Training of Neural Networks	Learn invariant features between the source domain(simulated image) and target domain(real image).	Minimal representation of the feature information for tasks. Difficult to understand and infer information from feature space.
PixelDA	GAN based network which is conditioned on a synthetic image data and a noise vector which leverages the use of domain adaptation.	Domain adaptation at pixel level. Easy to visually understand the representation since it is an image. Better training stability with a task specific loss function and pixel similarity regularizations.
Geometric Image Synthesis	GAN based network that uses geomertic information like segmentation, normal and depth map to generate novel poses.	Uses geometry information of the model to genenrate image. Produces geometrically consistent structure in the images produced. Requires 3D model with normal, depth, segmentation information as source domain information.

Chapter 3

Proposed Work

So far, we have discussed our motivation for generation of synthetic images for training neural networks and many such state of the art approaches. The first task going forward would be to select an appropriate approach that would be suitable for the image generation. Clearly, GAN based approaches shows lots of promises and would form the core of our generation pipeline. We have also seen domain adaptive approaches yielding very good results, so using them along with anyother approach is also a possibility. In GAN based approaches, the PixelDA paper from Google is an interesting one for implementation and usage. Thus it would be wise to try and implement the state of art approaches and see how they work in the test scenario for our required task.

Based on this bibliography search, going furthur, we can imagine a pipeline for image generation something similar to the PixelDA like the figure shown below,

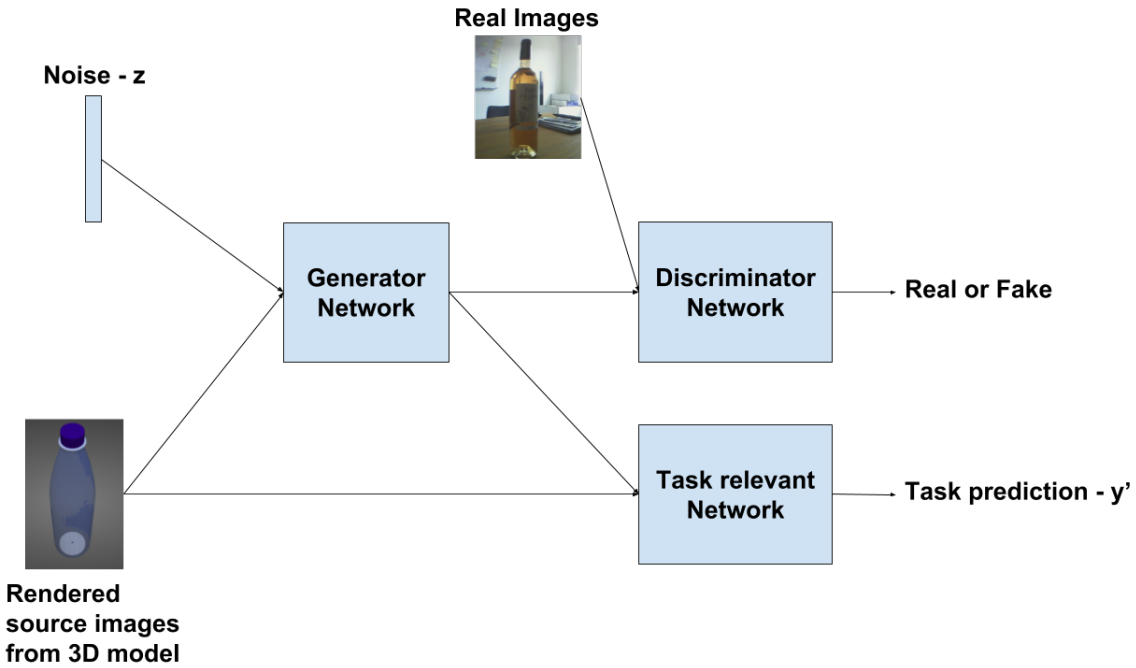

Figure 3.1: Likely architecture for image generation

We also aim to evaluate a process which contains the following steps,

- *Background image generation* - We need to generate the background images for the task. If we can find and collect enough relevant real world background images for the task, then it would be wise to collect and use them. Or this process can be imagined as a

GAN based approach that is used to generate such images. We need to test and see its performance.

- *Foreground image generation* - The foreground image can be images from a 3D model from a software. But this foreground image has to be refined so that it adapts to real world images.

- *Image refinement* - The final step would be image refinement. The image refinement can be done with the help of procedures like domain adversarial training or domain adaptation procedures[15]. Again here we need some real world image dataset for domain adaptation.

So currently we can use the ShapeNet database [4] which contains number of 3D models suitable for computer vision and robotics tasks. The datasets are well annotated with number of images of the model from different perspectives. Also the LineMOD dataset [8] which was used in PixelDA paper can be tested tried and tested on. It would also be interesting to use "cycles" renderer from softwares like blender as stated in "Geometric Image Synthesis" paper [1] so it can be used as a source for generator network and evaluate their performance.

Thus the primary work for this thesis would be to evaluate and implement the state of art image generation pipeline for robotic vision based applications. In the process, the thesis work will also try and study the advantages of various approaches, pick appropriate techniques and will strive to produce state of art results in the image generation task.

The following gantt chart plan(3.2) gives a snapshot of the plan going ahead with the thesis work. The idea is to simultaneously work on atleast two tasks at a given time frame, so as to not get stuck on any one. Also except for the month of June, tasks are almost equally spread across other months.

TASK	February	March	April	May	June	July	August
Studying and implementing the state of art image generation techniques							
Preparing 3D models and Images for training GANs and other networks							
Testing proposed architectures and finalizing the best of them							
Implementation of the proposed architectures in the company's pipeline							
Evaluating and benchmarking the obtained results							
Writing the final thesis report							
Final presentation							

Figure 3.2: Proposed timeline for the thesis

Conclusion

Thus this bibliography report presents an overview of the current state of art approaches for image generation for training neural networks based robotic vision problems. We started from the classical Bayes classifier as a generative model to the latest advances in generative models - Generative Adversarial Networks(GAN) based models. This bibliography work also collected some important papers and articles regarding the topic, studied and documented them.

This report presents the future roadmap for the thesis work that is to be continued in the topic. Also this report will serve as a point of reference during the thesis work as it contains the important information regarding the current state of art techniques.

Bibliography

[1] Hassan Abu Alhaija et al. "Geometric Image Synthesis". In: *CoRR* abs/1809.04696 (2018).

[2] Konstantinos Bousmalis et al. "Unsupervised Pixel-level Domain Adaptation with Generative Adversarial Networks". In: 2017. URL: https://arxiv.org/abs/1612.05424.

[3] Konstantinos Bousmalis et al. "Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping". In: *CoRR* abs/1709.07857 (2017).

[4] Angel X. Chang et al. "ShapeNet: An Information-Rich 3D Model Repository." In: *CoRR* abs/1512.03012 (2015). URL: http://dblp.uni-trier.de/db/journals/corr/corr1512.html#ChangFGHHLSSSSX15.

[5] Yaroslav Ganin et al. "Domain-adversarial Training of Neural Networks". In: *J. Mach. Learn. Res.* 17.1 (Jan. 2016), pp. 2096–2030. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2946645.2946704.

[6] *Generative Models*. URL: https://blog.openai.com/generative-models/.

[7] Ian J. Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. 2014, pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.

[8] Stefan Hinterstoisser et al. "Model Based Training, Detection and Pose Estimation of Texture-less 3D Objects in Heavily Cluttered Scenes". In: *Proceedings of the 11th Asian Conference on Computer Vision - Volume Part I*. ACCV'12. Daejeon, Korea: Springer-Verlag, 2013, pp. 548–562. ISBN: 978-3-642-37330-5. DOI: 10.1007/978-3-642-37331-2_42. URL: http://dx.doi.org/10.1007/978-3-642-37331-2_42.

[9] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *CoRR* abs/1312.6114 (2013).

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[11] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[12] Yunchen Pu et al. "Variational Autoencoder for Deep Learning of Images, Labels and Captions". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 2352–2360. URL: http://papers.nips.cc/paper/6528-variational-autoencoder-for-deep-learning-of-images-labels-and-captions.pdf.

[13] K. Shmelkov, C. Schmid, and K. Alahari. "How good is my GAN?" In: *Proceedings of European Conference on Computer Vision*. 2018.

[14] Leon Sixt, Benjamin Wild, and Tim Landgraf. "RenderGAN: Generating Realistic Labeled Data". In: *Frontiers in Robotics and AI* 5 (2018), p. 66. ISSN: 2296-9144. DOI: 10.3389/frobt.2018.00066. URL: https://www.frontiersin.org/article/10.3389/frobt.2018.00066.

[15] Mei Wang and Weihong Deng. "Deep Visual Domain Adaptation: A Survey". In: *Neurocomputing* 312 (2018), pp. 135–153.