Technische Universität Berlin

Faculty IV: Electrical Engineering and Computer Science
Institute of Software Engineering and Theoretical Computer Science

Neural Information Processing Group
Marchstraße 23
10587 Berlin

Master Thesis

Learning Representations of RGBD Data using Adversarial Training

Duy Pham

Matriculation Number: 0387589
10.01.2018

Supervised by
Prof. Dr. rer. nat. Klaus Obermayer

Assistant Supervisor
Youssef Kashef

Acknowledgments

This dissertation originated in a research project in the Neural Information Processing Group (NI) in TU Berlin.

First of all I would like to thank Prof. Dr. Klaus Obermayer, head of the NI Group, for giving me the opportunity to participate in a state of the art research in Deep Learning, and for the knowledge that he gave in the courses Machine Intelligence 1 and 2, which provided me the necessary background for the thesis.

Special thanks to Mr. Youssef Kashef for being my personal and direct supervisor. The time working with him was full of pleasure. Mr. Kashef was very helpful in redirecting the project when the results were not adequate and in raising good critical questions when the outcomes got better. He also spent effort on proofreading my thesis and gave valuable comments.

Many thanks to my group mates, Rodrigo Parra and Zitong Lian, for sharing the working spirit, both technically and mentally. Although our topics are somewhat different, it is still much easier for me to have partners going along in the same direction.

Furthermore I would like to thank all the people in the NI Group that I had a chance to meet and discuss Machine Learning topics in the common student room. Some of those conversations were very useful for my thesis.

Last but not least, this work was supported by the Deutsche Forschungsgemeinschaft (GRK1589/2). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X (Pascal) GPU used for this research.

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Berlin, 10.01.2018

. . . . . .

Duy Pham

Abstract

Generative Adversarial Networks (GAN) have shown potentials for being a powerful implicit model. They have achieved impressive results in image super-resolution, image generation and image-to-image translation. In this thesis, we train two GAN models to learn 3D information from different household objects and then use them to draw samples to augment training datasets of an Object Classifier, as they are potential for learning data representations. One of the GANs synthesizes a 3D depth map from an object image, while the other generates a rotated RGB version of a given object. On the Washington RGB-D dataset, our experiments show that the first GAN significantly improves the performance of an Object Classifier, especially when its synthesized data is added in large proportions, such as 75% of the total training data, to the classifier's training set. The improvement is verified by a T-Test. The second GAN is not yet able to get the same achievement, but its synthesized images still look acceptable. Besides, we also do some noise injection experiments to evaluate the contribution of Depth in a dual-channel Object Classifier and show that Depth contribution is still limited to some extents, comparing to RGB.

Zusammenfassung

Generative Adversarial Netzwerke (GAN)s erziehlen bemerkenswerte Leistungen bei der Lösung von Super-Auflösungs-, Bildergenerierungs-, Bild-zu-Bild-Übersetzungsproblemen. In dieser Arbeit werden GAN Modelle in zwei verschiedenen Variationen eingesetzt, um Representationen von alltäglichen Gegenständen im Washington RGB-D Datensatz anhand von 3D Information zu lernen. Daraufhin wird das generative Modell verwendet um die Bilderkennungsdatenbank zu erweitern. In der ersten Variation hat das GAN die Aufgabe das RGB Bild mit tiefen Information zu komplimentieren. In der zweiten Variation schätz das GAN anhand eines RGB Bildes ein Zweites aus einer anderen Perspektive zu schätzen und damit ein Stereo Bild eines Objektes zusamenzusetellen. Ergebnisse wiesen darauf, dass die erste GAN Variation zu deutlich höheren Bilerkennungsleistungen führt, sogar, wenn die Anzahl der wahren Datenpunkte nur 25% des trainigssatzes ausmacht und der Rest durch den Generator künstlich erweitert wurde. Die Verbesserung ist an einem Signifikanztests bestätigt. Die zweite Variation führt zu weniger deutlichen Verbesserungen, obowhl die generierten Datenpunkte qualitativ akzeptable erscheinen. Zusätzlich erfolgt eine Bemessung des Beitrags der zwei Eingangs-Modalitäten, RGB und Tiefe, und deren Einfluss auf die Entscheidung des Klassifikators. Es lässt sich erkennen, das die Klassifikation viel stärker vom RGB Bild hängt, als die Tiefen Eingabe.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

1.1.1 Context

In the recent years, Machine Learning (ML) has become a powerful tool for computer scientists, business analysts and other professionals to improve their work because it can make use of a large amount of data using probabilistic models. It is able to capture properties of different objects and events by making observations, which is difficult to do using human labour or traditional rule-based methods which require many manual engineering works. ML became popular thanks to the boost of data generation in the Internet era.

There are many different ML methods, in which Deep Learning (DL) is currently the most exposure ML one because it could already beat traditional methods in a lot of different fields from Computer Vision to Natural Language Processing or Sound Recognition without the need of thorough domain knowledge. It is basically a class of functions which consists of different layers of computing, each of which is supposedly responsible for learning a specific level of understanding about the objective problem. In most of the cases, Deep Learning is implemented in different forms of Multi-layer Neural Networks.

1.1.2 The Need of Good Data Representations in ML

A common point for all of the ML methods is that we have to find a good vector representation of our data, otherwise the learning algorithms will either not work or not produce the most reliable outcomes. The task is called "Feature Engineering" as it extracts "Features" from raw data. For instance, in text data, document features can be vectors of tf.idf statistics of each word in the document; in image data, image features can be vectors of corners, edges and some other special properties of the image content. In many cases, Feature Engineering takes most of the time of a learning job.

Because of the importance of having good features, Representation Learning has become an important research topic in ML. In traditional learning methods, this is crucial because the set of models only understand the input in vector space. In Deep Learning, this is less heavy because of techniques such as Convolutional Neural Networks, which take directly the raw data and learn different levels of representations in the hidden layers. However, having a good representation helps not only simplify the learning task dramatically but also increase the robustness of the learning engine.

1.1.3 Generative Adversarial Networks (GAN)

Generative Adversarial Networks [5] are Unsupervised/Semi-Supervised Deep Neural Networks where the learning engine consists of two competing components: The Generator and the Discriminator. As its name indicates, the generator tries to generate data from a noise distribution, could be conditioned on some input or have no conditions; while the discriminator tries to classify between a real data item and the synthesized one from the generator, under the same condition. While using supervised losses as components, GAN is usually referred as an Unsupervised ML task because it does not try to learn a mapping between the data and a set of labels. Instead, GAN tries to learn the generalized distribution of a particular dataset and to be able to draw samples from that distribution. The distribution that GAN, or particularly the Generator, learns can be unconditional or conditional. The original idea of GAN is unconditional, whereas many following-up papers use a conditional version to better train it for specific tasks. The advantage of GAN comparing to some other alternatives such as Variational Auto Encoder (VAE) [7] is that GAN tends to produce more realistic images with less blurry boundaries, which is a shortcoming of the L2 loss [5]. However, because GAN does not minimize a standard loss function between the outputs and the targets, evaluating it is still a challenge. In many cases, human evaluation is used.

The nature of probabilistic distribution estimation of GANs motivates researchers to utilize it for extracting data representation. There are different ideas on using GAN to improve an ML task. For instance, the Generator, which is supposed to capture a probabilistic distribution, can draw new samples from its knowledge of the data, which could enrich the training and evaluation sets of a ML task. This is potential if the GAN is trained well and the Generator can capture the data distribution.

1.2 Objective & Scope

The main focus of this Master Thesis is training and evaluating of Generative Adversarial Networks (GAN) on RGB-D data, particularly the Washington RGB-D Dataset [9]. We would like to find out if GAN is able to generate good object images, particularly useful images as training items for an Object Recognition task. In the effort to evaluate if GAN is able to help improving classification performance or not, the thesis demonstrates different ways of training a GAN to learn from RGB and Depth data, and testing the GAN results on another Deep Network for Object Recognition, which we call the "Baseline Classifier".

The thesis focuses on evaluating the performance of a trained GAN's outputs on training a standard ML task. Particularly, we aim to train GANs to learn distributions from the Washington Object RGB-D Dataset [9], then evaluate the effect that the GAN data have in an Object Classifier. We choose the work from Eitel et al. [3]. It is basically a Two-Channel Deep Object Classifier taking an RGB image and a Colorized Depth-map of every object as inputs of two copies of AlexNet [8], then doing classification in two deeper fully-connected layers. In all of the experiments, a conditional version of GAN is

used. The original unconditional GAN is not in the scope of this thesis.

1.2.1 Using a GAN to learn representations from Washington RGB-D Dataset

In this thesis, two different settings of GAN are experimented.

In the first model, the objective of the Generator is to generate a good depth map, conditioned on an RGB frame of an object. Basically, it is the transformation of an object from a 2D space to a 3D space. The task of the Discriminator, as usual, is to classify a real depth image from a depth image produced by the generator.

In the second model, the objective of the Generator is to generate an RGB frame in another pose of an item, conditioned on both the RGB and the Depth map. Briefly speaking, this Generator tries to "rotate" the object by an angle. The Discriminator, as usual, tries to classify the real RGB-D pair from the "fake" pair.

1.2.2 Training a deep neural network to do object classification

In this thesis, we use the work from Eitel et al. [3] because of several reasons. First, it is relevant to our scope; the authors train an Object Recognition Network using Transfer Learning from CaffeNet, one of the first reference implementation of AlexNet [8]. Second, its evaluation is based on same dataset (Washington RGB-D) and in the manner that fits us (learning from both RGB and Depth data). Finally, it achieves high accuracies (more than 91% when using both RGB and Depth data).

1.2.3 Comparing classification performances on original data and data from GANs in different scales

We perform the evaluation of a GAN in multiple steps.

- First, we construct our "Baseline Classifier" by reproducing the Eitel et al. [3].
- Second, we substitute parts of the training data by the generated data from the corresponding GAN.
- Third, we re-train the Baseline Classifier accordingly and compare the accuracies.
- Finally, we perform some noise injection experiments to the Depth and RGB components respectively to better understand how much those components contribute to the classification results.

1.3 Outline

This section gives a brief outline of the content of the following chapters in this thesis.

Chapter 2 describes the relevant researches and the foundation works which are used

in this thesis. It also gives an introduction to the background needed to follow the next chapter.

Chapter 3 discusses how we train the models and some relevant theoretical reasoning. The evaluation strategy and the train-test split are also thoroughly discussed.

Chapter 4 describes the specs and architectures of the networks we use together with some relevant hyper-parameters.

Chapter 5 explains how we set up the experiments and thoroughly analyzes the evaluation results of our models.

Chapter 6 summarizes the thesis, describes the problems that occurred and gives an outlook about possible future works.

2 Fundamentals and Related Work

2.1 Image Classification using Deep Learning

Deep Learning (DL) has a lot of advantages compared to the other ML methods. First, in many cases, it does not need a thorough and careful Feature Extraction, the techniques such as CNN help the network learn the features by itself during the training. Second, the multiple-layer nature enable it to learn multiple levels of features from the data. Because of that, Image Classification has been a very common domain application for Deep Learning since the beginning, and is one of the domains where it achieved the most so far.

In Computer Vision, different works are usually evaluated on some standard "de facto" labeled datasets. The accuracies on those datasets are considered as the performance metric of a network; and those having higher scores on the datasets are considered the better ones. ImageNet [16] is currently the biggest and most popular Object Image Database available, consisting of millions of images and over 20000 object categories.

At the moment, there are many different approaches of Deep Learning to solve the ImageNet Image Classification challenge. A common point of all of them is the use of the pattern Convolutional, Max-Pooling and ReLU layers. In this section we briefly summarize the three most popular networks which are AlexNet [8], VGG [18] and Inception [21, 22, 20].

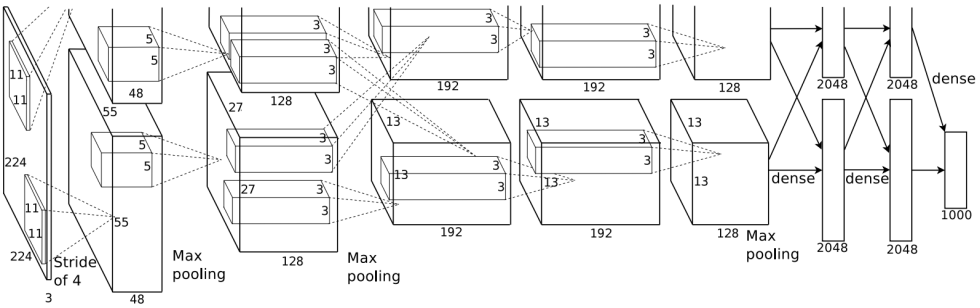


Figure 2.1: AlexNet Architecture [8]

On ImageNet, Krizhevsky et al. [8] trained an 8-layer Convolutional Neural Networks which achieved 37.5% Top-1 error rate and 17.0% Top-5 error rate. The network, which is often referred as AlexNet later, was one of the first Deep Neural Networks to improve the classification score on ImageNet significantly, and started an era that Deep Learning leaves most of the traditional Computer Vision methods far behind.

AlexNet consists of 8 layers as shown in Figure 2.1. There are 5 convolutional layers and 3 fully-connected layers. The last layer is connected to a 1000-way Softmax layer because the ImageNet dataset [16] used to train the network contains 1000 classes. There are different implementations on the network in different platforms, and there are also many researches inferring the architecture, which create small varieties of AlexNet architectures. Those differences are usually about the size of the filters and the number of convolutional layers.

After AlexNet, there were a lot of efforts from different research groups on further improving the classification results, resulting in many different architectures. A famous example is VGG Net [18]. It was introduced at the Visual Geometry Group in Oxford and was able to achieve a Top-1 error rate of 23.7% and Top-5 error rate of 6.8% on ImageNet in the best setting. The difference between VGG and AlexNet is in the filter size and the depth of the network, which can be seen in figure 2.2. Unlike AlexNet, the authors of VGG propose a stack of three fixed filter size of 3×3 and introduce much deeper architectures. Depending on the particular configuration, the network can grow up to 19 layers. Despite the highly achieved ImageNet accuracies, VGG also has big advantages which are the training time and the storage size of the network, because of the huge depth.

Another popular State-Of-The-Art image classification architecture is Inception [21]. It has quite a different design than the other two. Particularly, the authors define a module called "Inception" and apply it multiple times in the network. As being shown in Figure 2.3, it uses different convolution filters at different scales and a Pooling layer, all followed by a concatenation operation. The 1×1 convolution acts as a dimensionality reduction layer. In the overall network, a few Inception modules are used together with other common layers such as Convolution-MaxPool and Softmax, resulting in a network with totally 22 layers. With some different improvements in the follow-up papers [22, 20], the Inception-v4 version could reach a Top-1 Error of 17.7% and a Top-5 Error of 3.8% on ImageNet.

A common point of the three designs is that every network has a variety of different configurations, creating different versions of the same design. People usually pick one setting depending on the problems and the hardware power.

2.1.1 Transfer Learning

If we take a closer look at a Deep Classification Network such as AlexNet [8], we can see that the last layer is often a Softmax layer, which converts the output to the range $[0, 1]$ so that we have a prediction probability of each class. The real classification work is done in the previous layer (fc8), where the probability that the object image belongs to each class is calculated by a Linear Regression. In AlexNet, as we have 1000 classes, we have 1000 neurons in fc8 corresponding to 1000 Linear Regressions. The intuition here is that, the input of that decision layer is a good representation of the data if the network is performing well. In principle, we can reuse the output of the layer before fc8, which is fc7, in other ML tasks. That forms the basic idea of Transfer Learning, which is the essential method of a number of State-Of-The-Art Object Classifiers.

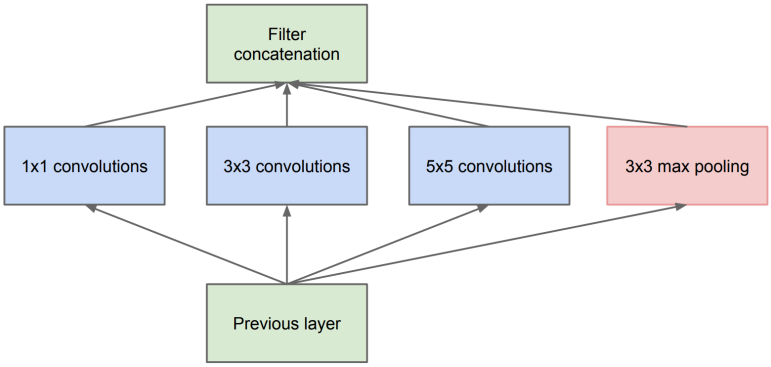
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 2.2: VGG architecture and specs from the original paper [18]

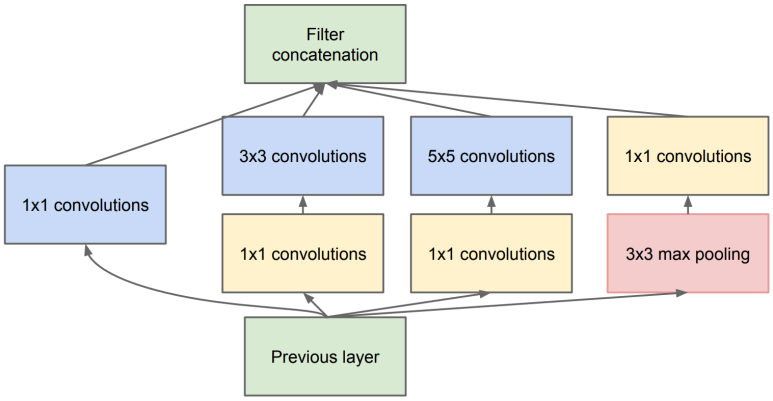
2.1.2 Washington RGB-D Dataset

Beside ImageNet, the Washington RGB-D Object dataset is another one which is often used as a "de facto" of benchmarking an Image classifier. It is a big Image dataset introduced by Lai ei al. [9] at the University of Washington in Seattle. The advantage of this dataset is that it provides depth information alongside with RGB images. Furthermore, the RGB-D dataset is very structurally organized. Particularly:

- It contains 300 common indoor objects in 51 different categories, 45 of which are shown in Figure 2.5.
- Each category is then divided into different instances. For instance, in the "Apple" category there are 5 different apples shown in Figure 2.4.
- Each instance is recorded in 3 360°sequences by 3 different camera angles: 30°, 45°, and 60°.
- Each frame has:



(a) Inception module, naive version



(b) Inception module with dimensionality reduction

Figure 2.3: Inception Module from the original paper [21]

- an RGB image of the object
- a Depth map
- a Mask indicating the object position for extracting the object without background (which we do not use in this project)
- For every 5th frame, there is an estimated pose.

2.1.3 Object Recognition on Washington RGB-D dataset

As we have discussed in section 2.1.2, the Washington RGB-D Dataset has some advantages over the other ones as a "de facto" benchmarking dataset. The biggest one is the depth maps provided alongside with the RGB frame of each item which enables researchers to explore the potential of depth information in different ML tasks.

The earliest work on the RGB-D Object Dataset came from the authors of the dataset themselves [10]. They apply Histogram of Gradients (HOG) to both the RGB frame and the Depth map and propose a distance metric among different instances to do some

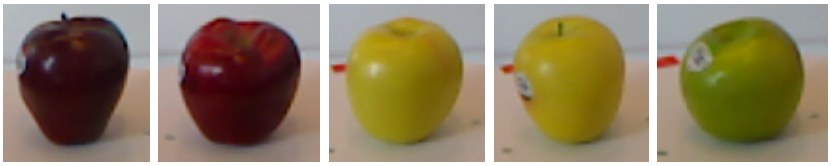


Figure 2.4: 5 different instances of the "Apple" category in the Washington RGB-D object dataset

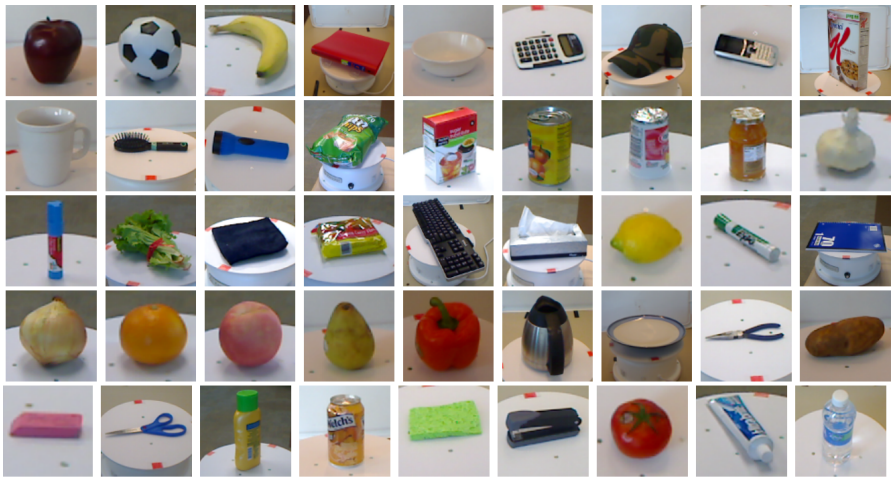


Figure 2.5: Sample RGB Images from the dataset [9]. Each object belongs to a different category. Figure from the original paper.

classifications. They achieved an accuracy of 85.4% on the category classification task, which was the State-Of-The-Art before the CNN era.

Soon after CNN was introduced, it surpassed all other traditional Computer Vision methods. Together with the Transfer Learning techniques, the researchers quickly got very high results on the RGB-D Object Dataset and demonstrate that depth data does indeed reduce the classification errors [3, 1]. The differences between different approaches are usually about which pre-trained networks the authors use and how they pre-process the image channels, especially the depth channel. In this thesis, we build our baseline classifier based on the main components from Eitel et al. [3] as it achieved high accuracies on the same dataset and the architecture is simple enough to not take too much time to train, which is very important because we also have to train other bigger networks (GANs).

Transfer Learning is the main component of Eitel et al. [3]. In the paper, the authors use an implementation of AlexNet in Caffe (CaffeNet) to get the representations of the objects in the Washington RGB-D dataset [9]. Particularly, there are two channels of AlexNet, one is for the RGB and the other one is for the depth-map. The Depth-map is colorized using the jet color-map before feeding into AlexNet. The outcome representations of both channels are then fed into a fully-connected layers, where the

final results are connected to a 51-way Softmax layer, to classify 51 different classes in the Washington Object Dataset [9]. Figure 2.6 describes the network.

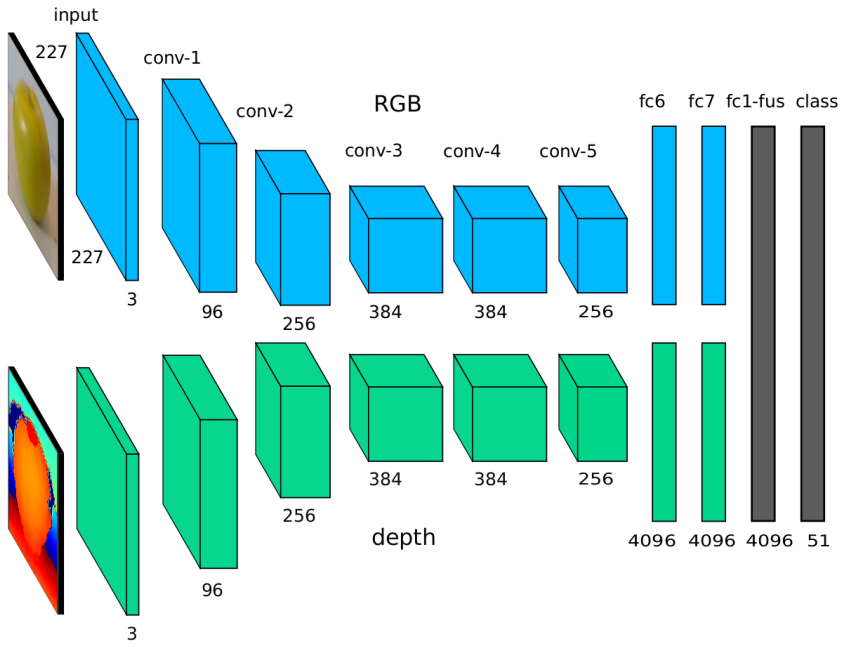


Figure 2.6: Eitel et al. architecture for Object Classification. Figure from the original paper [3]

2.2 Generative Adversarial Networks (GAN)

As being already mentioned in section 1.1.3, GAN consists of two differentiable and adversarial functions: the Generator and the Discriminator, usually in different forms of Deep Neural Networks.

There can be different designs for the two networks depending on the domains that GANs are applied. In Computer Vision applications, it is common that the Generator is a Convolutional Neural Networks (CNN) which behaves in a similar way as how an Auto Encoder [2] works. The first half is an Encoder where an image is reduced to a representation of 1D array, then the second half tries to decode that representation to the original image. The difference here is that this Generator does not minimize the direct loss between the output and the input. Instead, it tries to maximize the loss of the Discriminator. The Discriminator, as already been explained in section 1.1.3, is a binary classifier who tries to distinguish the original image and the output image from the Generator.

In the most basic setting, the Discriminator always receives a pair of inputs:

- Target: The ground-truth from the dataset, also called "Real Item", label 1

- Output: The product of the Generator, also called "Fake Item", label 0

Let G denote the generating function and D denote the classification function, and given an arbitrary image x, then $G(x)$ is also an image and $D(x)$ is a scalar label where $D(x) \in 0, 1$.

Obviously, the goal of the Discriminator is to assign label 1 for all the real images and label 0 for all the fake images. Thus the Discriminator Loss is a Cross-Entropy between the output of the Discriminator and the set of labels $0, 1$. In other words, the Discriminator maximizes the probability of assigning the correct labels to its inputs by maximizing the Value Function $V(D) = \log(D(x)) + \log(1 - D(G(z)))$.

At the same time, the goal of the Generator is to "create troubles" for the Discriminator. Particularly, it tries to make the Discriminator think that its output is "real" (label 1), meaning it minimizes the second component of the Discriminator Value Function $\log(1 - D(G(z)))$.

Based on the setting above, we end up in a two-player minimax game with the following value function:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

In practice, for implementation purpose, people usually set the Discriminator Loss to be $L(D) = -V(D, G)$ so that we can use Gradient Descent to minimize it and utilize the existing Deep Learning frameworks. The Generator is also set to maximize $\log(D(G(z)))$, equivalent to minimizing $-\log(D(G(z)))$, instead of minimizing $\log(1-D(G(z)))$ because the latter has problems of vanishing gradients, shown in [5].

We can see that the Discriminator job is as simple as any other binary classifier, it distinguishes the real data distribution p_{data} from the Generator outcome distribution p_g. In order to "fool" it, the Generator has to learn the data distribution p_{data}. In every step, the optimal Discriminator is $D^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$, proved in [5]. When the system reaches an equilibrium, p_g should converge to p_{data}, thus the optimal Discriminator Outcome should be $D^*(x) = 0.5$, equivalent to a Value Function with absolute value $|\log(0.5) + \log(0.5)| = 1.38629436112$. This is an important property that we can use to determine if the GAN training has converged or not.

As an example, if we want to learn an apple image distribution, the Generator will draw synthesized apple images from an arbitrary input z, and the Discriminator will try to tell that the output apple image is a "fake" one and an arbitrary image from the data is a "real" one. When the two networks converge, we should have an implicit model representing an apple image distribution.

2.3 Conditional GAN and Pix2pix

The original non-conditional GAN is sometimes referred as the "Vanilla GAN". It sets up a good foundation for the field, but is sometimes unstable and hard to train. In

some applications, people are not interested in the general distribution but rather pay attention to a directed data generation, that is, a conditional distribution. For instance, generating a random apple may not be as interesting to somebody as creating an image of "Pink Lady" apple. Just shortly after the introduction of GAN, Mirza and Osindero [13] proposed Conditional GANs, an improved version of GAN. The idea is simple, instead of generate from an arbitrary noise input z, the Generator also takes another vector y as the condition. In the example of apple image generation, we can provide a green apple image, a string "Green Apple", or a binary label vector as y. The Discriminator also takes y as its condition. By being trained in that way, GAN now learns a conditional model $p_g(x|y)$ and tries to make it as close to the conditional distribution of the data $p_{data}(x|y)$ as much as possible. Figure 2.7 demonstrates how conditional GAN works.

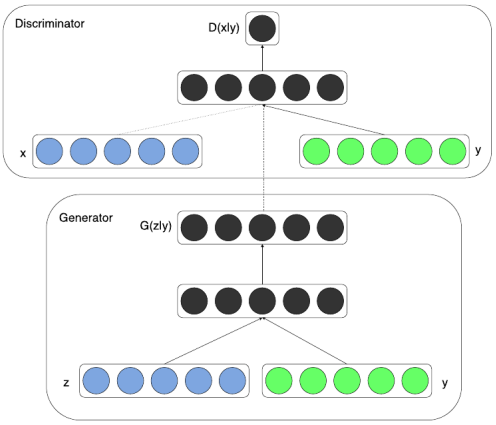


Figure 2.7: Conditional GANs. Figure from original paper [13]

Conditional GANs have many applications, one of them is image super-resolution. The idea is to sharpen low-resolution image by converting it to a higher-resolution one. In SRGAN [11], shown in Figure 2.8, the Generator takes a downsampled image as the input condition and produces a higher-resolution image through a convolutional network with some skip connections and upscaling layers; and the Discriminator distinguishes the original input image from the output of the Generator while being conditioned on the same downsampled image. The results were more realistic than the other State-Of-The-Art methods, as being shown in Figure 2.8.

Image-to-Image Translation with Conditional Adversarial Networks (Pix2Pix) [6] is another conditional GAN. It maps an image to another image. The authors demonstrate that Pix2Pix is able to generate satellite images from map views and vice versa, detailed buildings with texture from skeleton sketches, or day to night transformation. Those impressive results come from the idea of conditioning a GAN on an input image. In figure 2.10, we can see some outputs of Pix2Pix from the authors. The general workflow is demonstrated in Figure 2.9.

As this is a conditional GAN, the loss function is also a little bit different from the

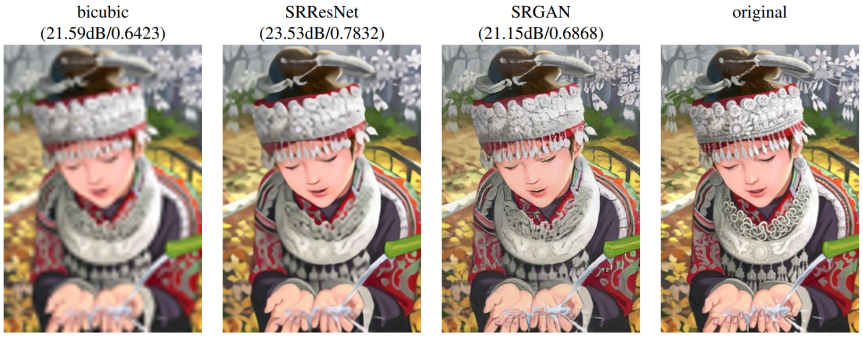


Figure 2.8: Examples from SRGAN [11]

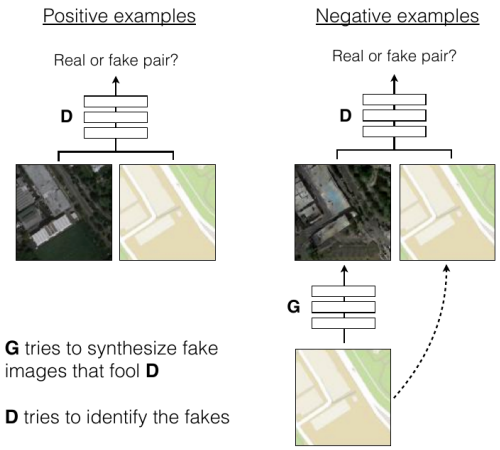


Figure 2.9: How Pix2Pix works [6].

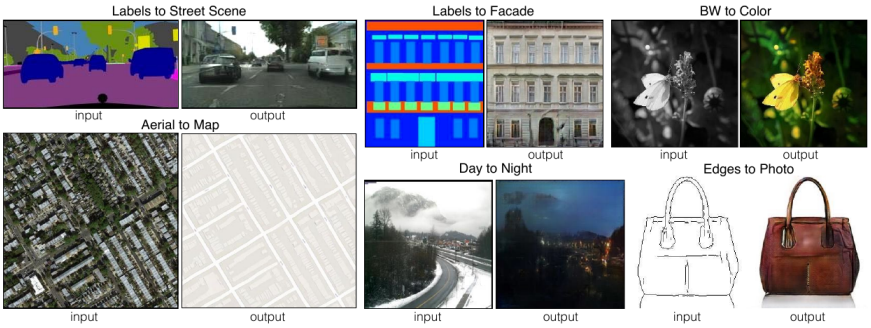


Figure 2.10: Examples of generated images from Pix2Pix [6]

Vanilla GAN in the sense that all the components now are conditional on an image y. Besides, the authors also suggest using an L1 loss to boost up the Generator training as

it helps the Generator learn faster and L1 enforces less blurry images than L2.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x,y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(y,z),y))] + \lambda L1(G)$$

3 Methodology

In this chapter, we describe how we design and implement the experiments with GANs to achieve the objectives of the thesis.

3.1 Our GANs

The main contribution of the thesis is to use the concept of GAN to adversarially train a generative model which can produce a reasonable depth-map of a given 2D image of a particular object. The model is expected to be able to understand the 3D properties of images of single objects. Because of its tasks, we call this one the "Depth-GAN".

In addition, we also train another GAN to learn the 3D knowledge in another dimension, the pose estimator. This GAN's job is to generate a 2D image of the same given object in another pose. It is expected to be able to rotate, in 3D space, the object by a particular angle while being conditioned on the object 2D image and its corresponding depth-map. We call this one the "Pose-GAN"

3.1.1 Network Architecture

We inherit the architecture from Pix2Pix with only some occasional modifications in the input and output layers because our experiments, especially the first one which generates depth-maps from an RGB image, is very close to their settings. The detailed settings are shown in chapter 4.

The Generator is basically an U-Net [15], an encoder-decoder with skip connections between every decoder layer and the corresponding encoder layer. The intuition behind this is that, there are relevant information between the input and the output. Especially the mirrors in the encoder part and the decoder part share the same resolution and same level of encoding. Therefore, letting them connect directly is a good idea. Figure 3.1 visualizes the architectures of both a standard Encoder-Decoder and U-Net.

Similarly, the Discriminator is also a Convolution-Batch-Norm-ReLu structure. It implements an idea which the authors call "PatchGAN", where an image are divided to some $N \times N$ patches and the network only judges the real-fake properties of each patch. The final decision is averaged over the single-patch decisions. The rest of its job are exactly the same as a normal binary classifier.

3.1.2 Training GANs

In training GANs, the Discriminator Loss is usually much more important than the Generator Loss because it can show if the adversarial training has converged or not. As

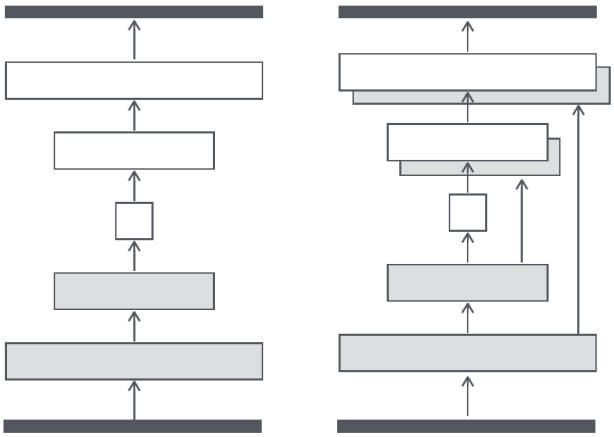


Figure 3.1: Standard Encoder-Decoder and U-Net architectures

being already explained in Chapter 2, when the (absolute) Discriminator Loss is stable at 1.38629436112, we know that they two have reached an equilibrium.

GANs are notorious for being very hard to train. As it does not minimize a simple scalar loss function, fluctuation in training is very popular and the network does not converge eventually. This is a very hard problem and researchers are still working on it. Fortunately, it does not always happens. Another common issue is the Vanishing Gradient problem caused by a very quickly-converged Discriminator. In most of the cases, the job of the Discriminator is much simpler than that of the Generator. When its loss is too small, the gradient quickly vanishes in the Back-Propagation steps and prevent the Generator to improve. The consequence is that the Discriminator Loss can never reach the equilibrium of 1.38629436112.

A common method to solve this problem is to regularize the Discriminator. In this thesis, we occasionally apply the following tricks:

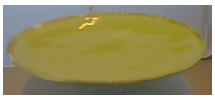
- Slowing down the Discriminator by reducing its learning rate or making it wait for two or even more update steps of the Generator
- Adding instance noise to the inputs of the Discriminator. [19]
- Doing One-sided Label Smoothing: replacing the label of the real image by a random number in a small range around 1, [0.7, 1.2] for example, instead of 1 [17]

While the first two methods are quite trivial as they aim to make the Discriminator job harder to balance the work with the Generator, the third method has an interesting explanation in Salimans et al. [17]. The authors reuse an old concept from several decades ago, Label Smoothing, and state that it helps reduce the sensitivity of the network with respect to adversarial examples. The reason why we smooth only the

real samples (label 1) is that, when we apply the trick we now have the new optimal Discriminator $D(x) = \frac{\alpha p_{data}(x)+\beta p_{model}(x)}{p_{data}(x)+p_{model}(x)}$. When $p_{data}(x)$ is very low and $p_{model}(x)$ is very large, p_{model} has no incentive to move towards p_{data}. Setting $\beta = 0$, as in usual GANs, completely removes p_{model} from the equation, thus prevents the problem.

Besides, in training our Pose-GAN, we also perform another trick: removing all of the round-shape objects, such as "Apple", "Bowl" and "Food Can". We believe that those items can confuse the GAN instead of adding important values. This idea is inferred from Elhoseiny et al. [4], where the authors also select only 34 out of 51 categories but do not declare explicitly which categories are chosen. In our Pose-GAN, we came up with the same number of to-be-removed categories, which are listed in Table 3.1.

Table 3.1: The round-shape objects removed from Pose-GAN training set

Category	**Example**	**Category**	**Example**	**Category**	**Example**
Apple		Garlic		Food Jar	
Peach		Tomato		Food Cup	
Food Can		Lemon		Glue Stick	
Orange		Lime		Water Bottle	
Onion		Mushroom		Soda Can	
Bowl		Plate			

3.1.3 Data Manipulation

As being mentioned above, in this thesis we focus on 2 tasks on GAN: learning to generate depth information and learning to generate another pose of an object. Just to clarify the terminologies, as we are doing image-to-image translation tasks, we call the conditional image the "input", the to-be-generated image the "output", and the ground-truth the "target". In that way, the Generator is supposed to make the "output" as close to the "target" as possible, while the Discriminator still tries its best to tell them apart.

In the first GAN, we line the Washington RGB-D dataset up by putting every RGB image along with the corresponding depth-map. Here the RGB is the input, the generated depth-map is the output, and the depth-map provided by the RGB-D dataset is the target. As the RGB image is an 8-bit one and the depth-map is 16-bit, we have to manipulate the data loading process to make sure that we have the correct forms. A wrong manipulation can lead to a great loss of information.

In the additional GAN where we want to learn to generate a different pose of the same object, we use both the RGB frame and the corresponding (real) depth-map as the input condition to GAN, to both the Generator and Discriminator. The depth-map is pre-processed and attached to the RGB frame as the 4th dimension, along with the three color channels. This GAN will then use this information to produce a normal 3-channel RGB image of the same object, but in another pose. The Discriminator's task is to classify this rotated image with the real image in that pose. Here the input is a 4-dimensional image, the output is an RGB, and the target is a real RGB image of that pose.

For every frame, we choose the target pose of the 10th frame in the video sequence, which is equivalent to a rotation of roughly 20°. We do that for several reasons. First, there are subtle differences and errors among different camera scans, which makes choosing an exact pose, such as 15°or 20°, unrealistic. Second, as we also have a complex stratified Train-Test split which will be explained in details in section 3.3, we would like to make sure that the target pose of every input frame should be in the same part of the split. A combination of an input from the training split and a target from the test split can lead to dangerous bias in evaluation, for both the GAN and the baseline classifier. Because we pick every 5th frame from every instance to be in the test set, choosing the 10th frame as the target makes sure that the rotated pose is in the same train-test split. Finally, a target rotation of roughly 20°is a reasonable choice as it is big enough to observe a clear rotation but still contains many similar details as the input, which makes the task more feasible.

For all types of GANs that we are working with, the Generator produces the Output, and the Discriminator reads both the Output and the Target and try its best to classify the two, while both of them are conditioned on the Input. To simplify the language, from this point we will sometimes refer both of them (i.e. the Generator and the Discriminator) to GAN only. For instance, the statement "The GAN reads image A and generates image B" is equivalent to "The Generator generates image B and the Discriminator tries to distinguish image B and a target B', while both of them are conditioned on image A"

3.1.4 Handling depth missing values

Due to technical limitations, there are corners that the Kinect sensor cannot capture the depth information. Those values are represented as 0's in the depth-maps. If we just keep the depth-maps as they are and feed them to the Discriminator, the GAN will learn the missing value patterns and will probably somehow generate a representation of missing values according to its understanding. This is not really what we want; thus, interpolating the missing values can be a good pre-processing step.

In this thesis, we use the interpolation method from a toolbox from the authors of NYU Dataset [14], which applies the colorization method from Levin et al. [12]. The NYU Dataset is another one with RGB-D Images recorded with Kinect. Unlike the Washington Object Dataset, the NYU contains mostly large indoor scenes instead of single objects. The colorization algorithm uses the RGB information to infer the depth values of the missing points. The details of the optimization method can be found in the paper [12].

3.2 Baseline task

For the baseline task, we keep most of the architecture of the two-channel network proposed by Eitel et al. Our implementation makes some changes on the manipulation of the depth missing values and the way we train the network.

First, regarding the depth missing values, in the paper, the authors propose a noise injection method to the depth-maps in the training process. The idea is too mimic the missing values patterns (the 0 values in the maps) so that the classifier becomes robust towards them. In other words, missing values are considered as noise in data. In our implementation, because we also remove those missing values from the GAN training process for some reasons that we have explained in section 3.1.4, we also do the same to the classification. Basically, we would like the data we use to train our GAN and the classifier to have identical formats.

Second, regarding the training strategy, as our approach of handling the missing values in 3.1.4 already gives better accuracies on the original data, we do not perform fine-tuning the AlexNet channels as the authors did. A benefit of no fine-tuning is that we can utilize the AlexNet representations for different training rounds without re-doing back-propagation throughout the entire 9 layers of the two-channel network. In fact, we only train the 2 last layers for most of the experiments.

For evaluating our Pose-GAN, we keep the same architecture, but instead of using the colorized depth-map for the second channel, we feed the target pose of every frame. In this way, for every item the network will receive a pair of RGB images in 2 different pose, which we call "Stereo RGB Input". Our objective is to find out if training with stereo images is better than with single RGB image or not. Depth maps are not involve in this experiment.

3.3 Evaluation Method

As we are working on 2 Deep Learning tasks and the dataset contains many details in different layers, a naive random Train-Test split is not the optimal approach. For this purpose, we design a more stratified split, which is demonstrated in Figure 3.2.

For the baseline classifier, we split the data using the same strategy as in Eitel et al [3], the work we refer to. In each instance, every fifth frame is sampled and in every object, one instance is randomly and completely taken out. This gives us about 138,000

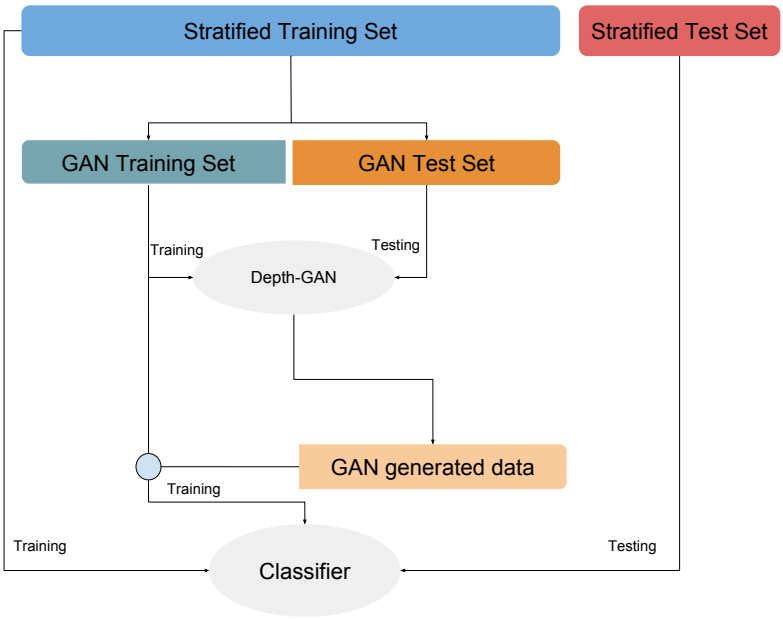
items for training and about 69,000 items for testing. We also perform the 10-cross-validation splits introduced in Lai et al [9]. That is, each of the lifting angle sequence is divided in to 3 sub-sequences (let us recall that each instance in the Washington RGB-D dataset is recorded in 3 different lifting angles: 30, 45 and 60 degrees). Among those 9 sub-sequences, two are randomly added to the evaluation set. In total, each experiment is done by using about 128,000 items for training and about 78,000 items for testing.

For the Depth-GAN, as we aim to use the GAN results to substitute the real data in training the baseline classifier, we also have to design an additional train-test split. The evaluation set generated from the previous strategy is left alone without any further modification. Instead, we randomly split the classifier's training set in different scales: 50-50, 25-75 and 10-90. The left portions are used for training the GAN and the right portions are used to evaluate it, of which the results will substitute the corresponding real items in the training set of the classifier. In this way, we make sure that:

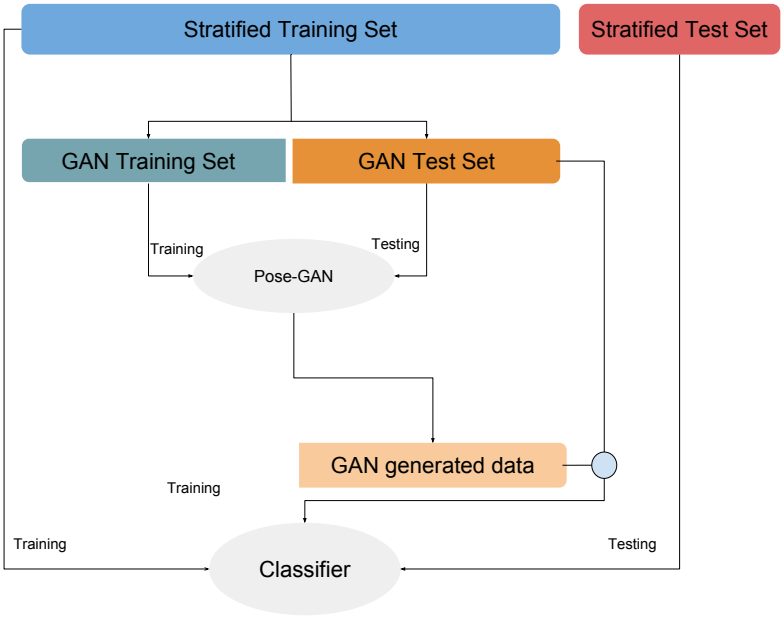
- GAN has not seen the items it is generating samples from. In other words, GAN is in evaluation mode when being used for training the baseline classifier. This is important because we cannot directly use the training output from GAN to train the baseline classifier, which is bias and encourages over-fitting.
- We can objectively compare the classification results trained from similar amount of data but with different portions of real-fake components. In this way, we can see how GAN contributes to the classification accuracies in different scales.

However, this is worth knowing that this approach also has a drawback. The comparisons of different GAN training setups are not entirely fair because the amounts of data used in training the GAN are different. For instance, if we would like to have 90% of GAN data to train the classifier, we can use only the remaining 10% of the data to train the GAN to make sure that the generated data has not been seen by GAN before. However, if we keep only 50% of GAN data in the classifier's training set, we can use up to 50% of the remaining data to train it, which potentially results in a stronger GAN.

For the Pose-GAN, the train-test split is mostly similar but has an important difference from how we do with the Depth-GAN. After training the GAN, we combine the "GAN Generated Data" with "GAN Test Set" instead of "GAN Training Set", demonstrated clearly in Figure 3.2b. The reason is that our generated data now contains all the poses supposedly in the GAN Training Set because we are doing pose rotation. In other words, we are replacing the "GAN Training Set" by the "GAN Generated Data". Besides, because of time limitation, we do not perform the experiments in different proportions like what we do with the Depth-GAN; only the 50-50 proportion is experimented.



(a) Evaluating Depth-GAN



(b) Evaluating Pose-GAN. Note the difference here: GAN generated data is combined with GAN Test Set instead of GAN Training Set.

Figure 3.2: Our stratified Train-Test split for evaluating GANs. The Stratified-Sampled Training set is split into a GAN Training Set and GAN Test Set with different proportions.

4 Model

This short chapter is a detailed and layer-wise description of the networks we used in this thesis.

4.1 GAN Model

Both of our GANs use the architecture inherited from Pix2Pix [6]. Table 4.1 describes the layer-wise details of our Discriminator.

Table 4.1: Discriminator architecture

Layer	**Configuration**	**Filters**	**Strides**	**Receptive Field**	**Comments**
Input	batch size x 256 x 256 x total input channels				The Discriminator input (GAN's output or Target) is concatenated with the condition
1	Convolution-ReLU	64	2	4x4	Output dimension: 128 x 128
2	Convolution-BatchNorm-ReLU	128	2	4x4	Output dimension: 64 x 64
3	Convolution-BatchNorm-ReLU	256	2	4x4	Output dimension: 32 x 32
4	Convolution-BatchNorm-ReLU	512	1	4x4	Output dimension: 31 x 31
5	Convolution-Sigmoid	1	1	4x4	Output dimension: 30 x 30

The Discriminator is a normal binary image classifier, so the output layer (layer 5) is a Sigmoid producing the label (1: Real or 0: Fake) of the input image. All Leaky ReLU functions use slope 0.2. The only variation we make is the total number of channels of the input layers. In our Depth-GAN, we have 3 channels of the RGB frame and 1 channel of the depth map, thus the input layer is $[batch\_size \times 256 \times 256 \times 4]$. In our Pose-GAN, we have 4 channels of the condition (RGB and D) and 3 channels (RGB) of the rotated frame, thus the input layer is $[batch\_size \times 256 \times 256 \times 7]$. Due to the configuration nature of our experimental device, we usually choose $batch\_size$ to be from 25 to 32.

The Generator is basically a U-Net Encoder-Decoder with skip connections between the mirrors in the Encoder part and Decoder. The detailed specs of each layer is described in Table 4.2. There are 8 layers of Encoding and 8 layers of Decoding. A Dropout of 0.5 is applied in the first 3 layers of the Decoder to create a form of randomization in both Training and Testing. This Dropout replaces the role of the z vector in Vanilla GAN.

4.2 Object Classification Model

As we have mentioned before, we use the model from Eitel et al. [3]. The architecture has been described in figure 2.6. We extract the representations of both RGB and Depth at the last layer of the corresponding AlexNet tunnel, then concatenate them into a $[1 \times 8192]$ vector as the input to the fusion network, which consists of 2 fully-connected layers. The specs of the network is described in Table 4.3. We train this network using the AlexNet representations of RGB-Depth pairs from the Washington RGB-D dataset.

For Depth-GAN data, we always train with all categories, so the output dimension is always $[1 \times 51]$ as we have 51 classes. For Pose-GAN data, we perform another experiment where we remove all the round objects. In that case the output dimension is $[1 \times 34]$.

Table 4.2: Generator Architecture

Layer	**Configuration**	**Filters**	**Strides**	**Receptive Field**	**Comments**
Input	batch_size x 256 x 256 x input_channels				The input is the condition.
Encoder_1	Convolution-ReLU	64	2	4x4	Output dimension: 128 x 128
Encoder_2	Convolution-BatchNorm-ReLU	128	2	4x4	Output dimension: 64 x 64
Encoder_3	Convolution-BatchNorm-ReLU	256	2	4x4	Output dimension: 32 x 32
Encoder_4	Convolution-BatchNorm-ReLU	512	2	4x4	Output dimension: 16 x 16
Encoder_5	Convolution-BatchNorm-ReLU	512	2	4x4	Output dimension: 8 x 8
Encoder_6	Convolution-BatchNorm-ReLU	512	2	4x4	Output dimension: 4 x 4
Encoder_7	Convolution-BatchNorm-ReLU	512	2	4x4	Output dimension: 2 x 2
Encoder_8	Convolution-BatchNorm-ReLU	512	2	4x4	Output dimension: 1 x 1
Decoder_8	ReLU-Deconvolution-BatchNorm	1024	2	4x4	Dropout = 0.5 Output dimension: 2 x 2
Decoder_7	ReLU-Deconvolution-BatchNorm	1024	2	4x4	Dropout = 0.5 Output dimension: 4 x 4
Decoder_6	ReLU-Deconvolution-BatchNorm	1024	2	4x4	Dropout = 0.5 Output dimension: 8 x 8
Decoder_5	ReLU-Deconvolution-BatchNorm	1024	2	4x4	Output dimension: 16 x 16
Decoder_4	ReLU-Deconvolution-BatchNorm	512	2	4x4	Output dimension: 32 x 32
Decoder_3	ReLU-Deconvolution-BatchNorm	256	2	4x4	Output dimension: 64 x 64
Decoder_2	ReLU-Deconvolution-BatchNorm	128	2	4x4	Output dimension: 128 x 128
Decoder_1	ReLU-Deconvolution-Tanh	Output Channels	2	4x4	Output dimension: 256 x 256

Table 4.3: Fusing Network for Eitel et al. [3]

Layer	**Configuration**	**Output**	**Comments**
Input	batch_size x 8192		AlexNet representations of RGB and Depth
Fusion Layer	Fully Connected Layer - ReLU	1 x 4096	"Fusing" RGB and Depth Inputs
Classification Layer	Fully Connected Layer - (Softmax)	1 x 51	51 classes or 34 classes

5 Evaluation

As we have discussed in the previous chapters, we feed the GAN generated data to the two-channel Object Classification network in different proportions with respect to the amount of original data. On one hand, we expect that GAN, especially the Generator can be a good data generation engine which can help training the classification task better. On the other hand, we hope that a well-trained classifier could be a good tool for quantitatively evaluating GAN data.

In this chapter we describe how we evaluate the data generated from GAN using such a baseline classifier. The basic idea is to implement the two-channel network from Eitel et al. [3] and compare the task's performances between using real data and using data generated from GAN. Different levels of noise injection are also used in order to evaluate how the baseline task uses depth and RGB information to make decisions.

We train the deep classifier in the following settings:

- Using the original data with both Depth and RGB information
- Using only a part of the original data (in different proportions) with both Depth and RGB information
- Using a part of the data (in different scales) with both Depth and RGB information, the remaining is substituted by GAN generated data
- Using the original data but with only RGB information in different proportions (10%, 25%, and 50%), to determine the contribution that Depth data adds to the learning procedure

5.1 Learning Curves

GANs are notoriously hard to train, mainly because it does not minimize a particular scalar function. Fluctuation and Gradient Vanishing of the Discriminator are common problems, as being described in section 3.1.2. In this project, fortunately we only face training problems with the Pose-GAN, which is acceptable because this GAN actually do a much more complicated work than Depth-GAN.

The Depth-GAN's Discriminator-Loss can converge quite early to around 1.4 and becomes a constant since then, which is a good sign because we have shown in chapter 2 that the Optimal Discriminator is at the Absolute Loss of 1.38629436112. It is demonstrated in Figure 5.1. As it already converges, we do not need to perform any regularization here.

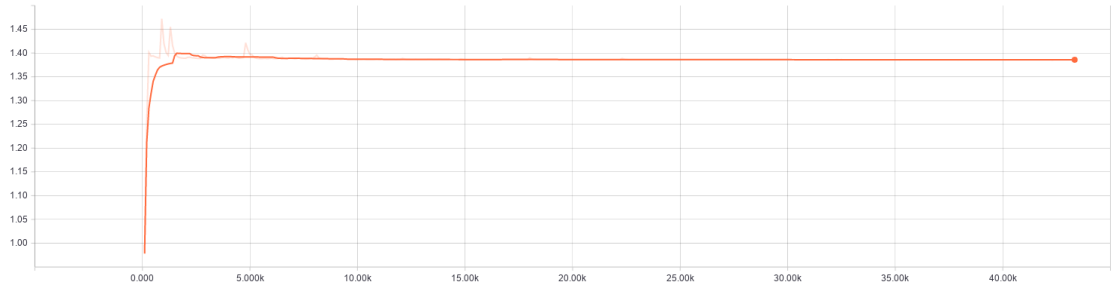
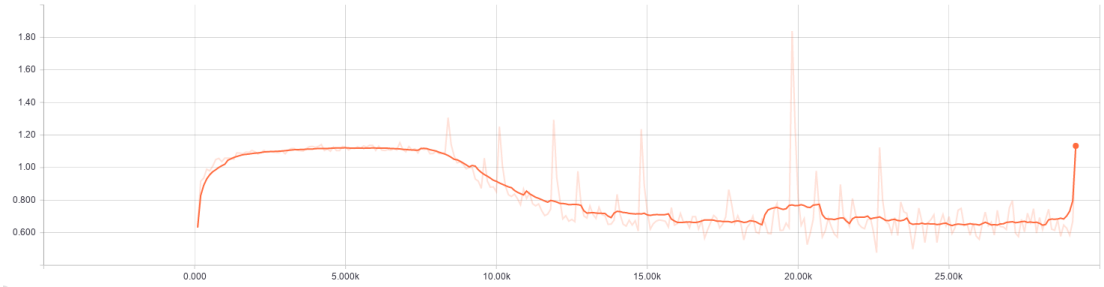
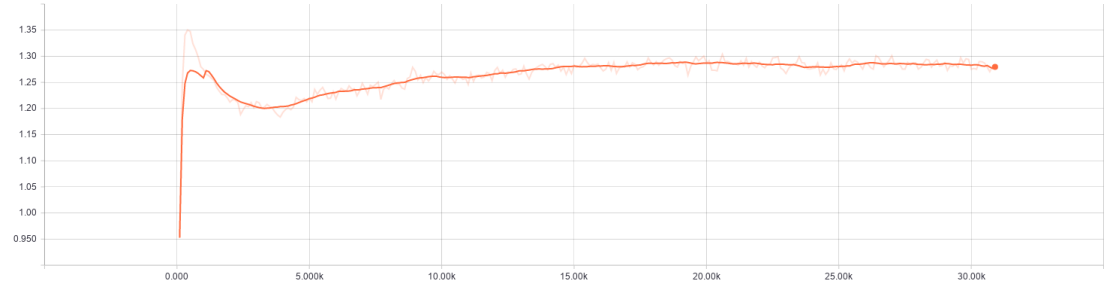


Figure 5.1: The discriminator learning curve of Depth-GAN

Things are different for the Pose-GAN, as the Generator has to do a more challenging work so it is easy to see that the Discriminator can become very good after a short time, causing Gradient Vanishing. It is very clear in Figure 5.2. In the same figure, we can see that the training is much more stabilized after we perform some regularizations. Particularly, one-sided label smoothing and instance noise injection are very helpful. Those regularizations help the discriminator loss reaches almost 1.3, which is much better than previously and is closer to our theoretical Optimal Discriminator.



(a) The discriminator learning curve of Pose-GAN, without any regularization



(b) The discriminator learning curve of Pose-GAN, with slower learning rate, input instance noise, and label smoothing

Figure 5.2: Pose-GAN's Discriminator Learning Curves

5.2 Results

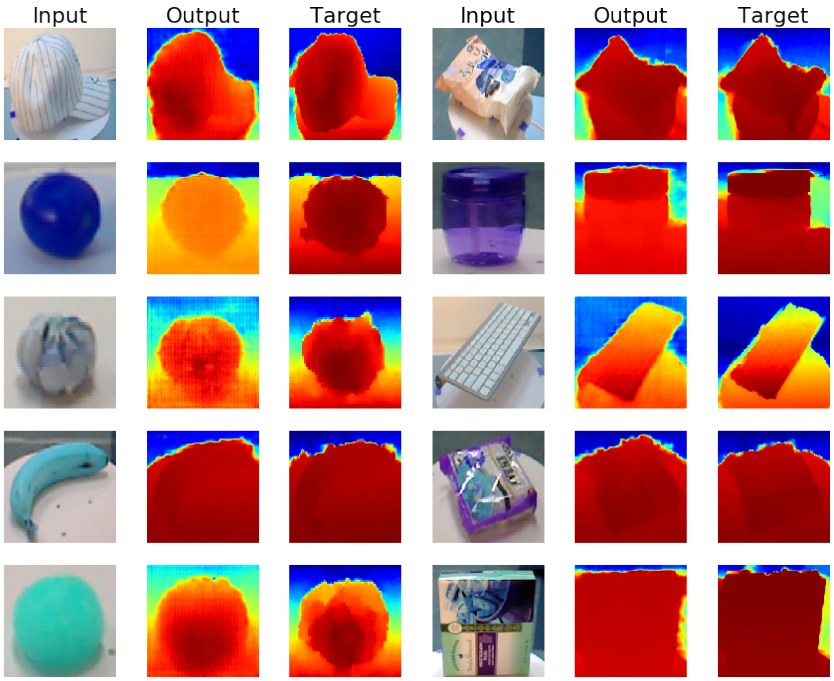
Before going to quantitative measurements, it is worthy to take a look as what our GANs produce.

5.2.1 Depth-GAN

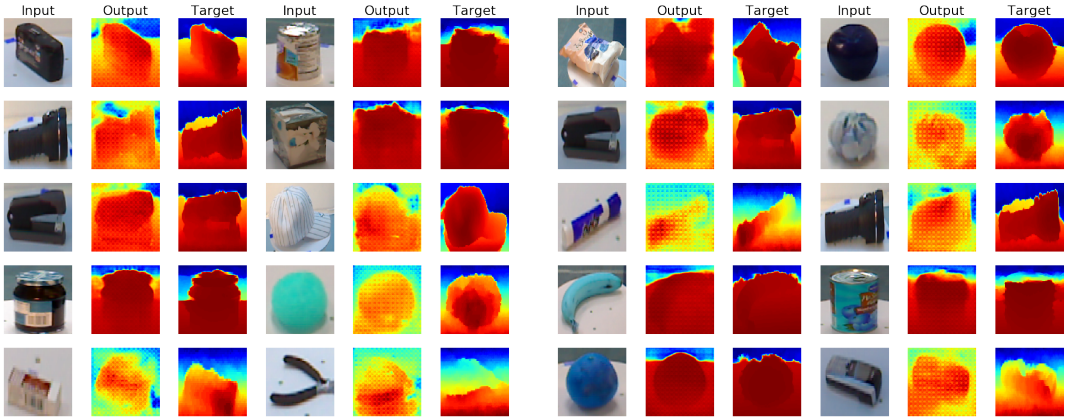
In Figure 5.3, 5 random samples from 5 random categories in the dataset are the outputs from our Depth-GAN in Evaluation Mode, which means the corresponding GAN did not see those samples in the training process. We show the results from all the three versions that we trained: Using 50%, 25% and 10% of training data. The depth maps are all colorized using the jet color-map for visualization purpose. It is easy to see that the GAN produces very reasonable depth maps with a smooth range of values. It is also important to see that when we reduce the amount of training data from 50% to 25% and 10%, the quality of the synthesized depth maps also noticeably reduces. The latter Depth-GANs produce images with more noise; but they still have decent details. This is understandable as we have explained in section 3.3.

Taking a look at those maps, we can see that the GAN does learn some trivial knowledge. For instance, the bottom pixels tend to be closer to the sensor than the pixels at the top of the image; and when there is a sharp edge or a sudden change of color patterns, there is usually an object placed on top of another. We can clearly see the second property in the sample of the fruits and the keyboard in Figure 5.3.

In these sample outputs from our Depth-GAN, we also see another popular advantage of GAN which is already discussed in some other works [5, 13, 6]. That is, the edges in the synthesized images are very sharp. If we instead minimized a squared loss function for example, we could expect a much more blurry outcome. It is also one of the reasons that using common scalar error functions such as MSE is not a good way to evaluate GANs' results.



(a) Depth-GAN trained with 50% data



(b) Depth-GAN trained with 25% data

(c) Depth-GAN trained with 10% data

Figure 5.3: Some generated samples from Depth-GAN. The Output and the Target are colorized using the jet color-map.

5.2.2 Pose-GAN

Similarly to the section above, Figure 5.4 shows some examples generated from our Pose-GAN. Visually, the Outputs do not look too bad, some of them, such as the cell phone, the flashlight and the boxes, are clearly rotated. However, the details are clearly not

optimal, and it works better with round-shape objects because they look the same in different points of view. The overall feeling is that, the synthesized objects are acceptable in terms of human eyes, but there is a certain level of noise in those images, which means the Generator is still somehow confused with the given condition/input.

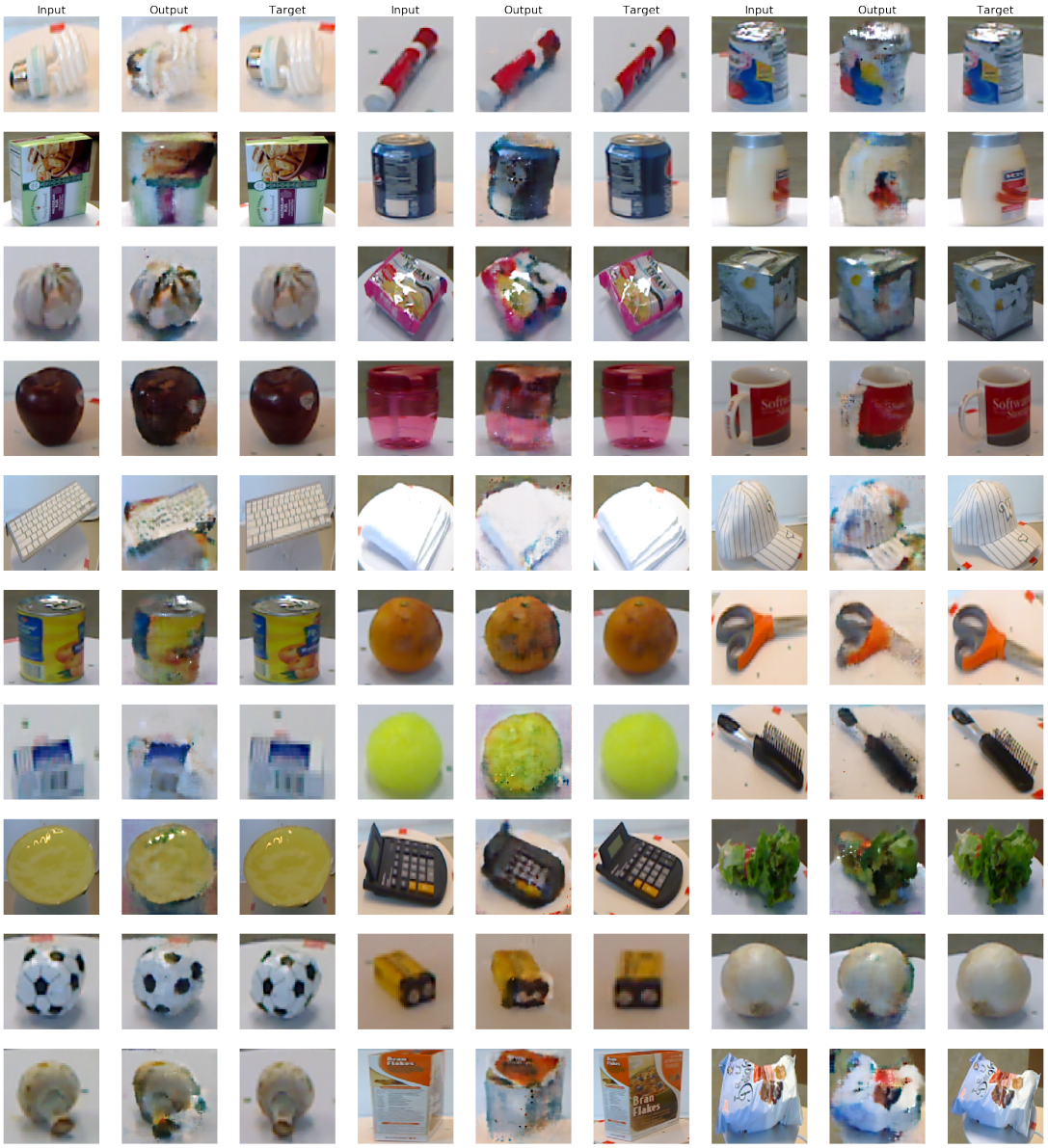


Figure 5.4: Some generated samples from Pose-GAN trained with 50% data. Note: The actual input is a 4-D image with the 4th dimension being the Depth-map, here we show only the RGB frame.

5.3 Performance of Depth-GAN Data

In the first experiment, we try to reproduce Eitel et al. [3]. We can see the plotted results in Figure 5.5.

In general, we can see that GAN has good contribution in the classification performance. It is worth mentioning that the transfer learning technique works very well, so we still have the accuracies more than 80% even when we use only 10% of the data. It is also expected to see the accuracies drop when we use only RGB data for training the network, but the difference is only a few percentages, which is an indication that the contribution of depth information is low in this training procedure. It is not a surprise because the RGB channels themselves have already achieved a very high accuracy (approximately 90%). However, as other works also have shown [3, 1], depth information does help improving classification on Convolutional Neural Networks (CNN) when combining with RGB, our experiment also follows that trend where all the accuracies using depth data are higher in average than the corresponding results without depth.

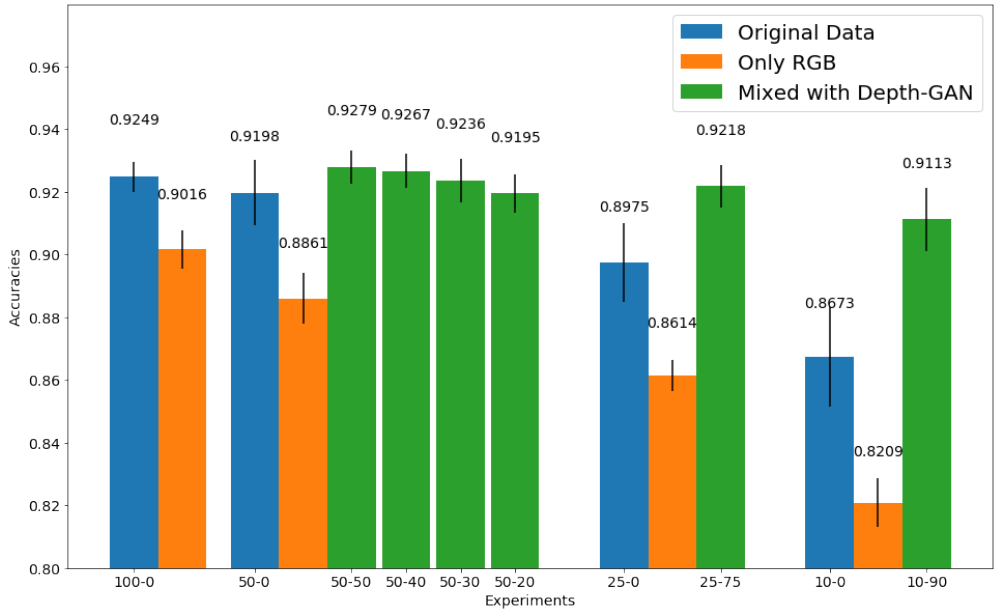
Table 5.1: Dependent T-Test (rounded) results between different training experiments. For each block, Depth-GAN data is added to the training set (the right-hand-side numbers) and there is one experiment with only RGB data. All accuracies are compared with the result batch trained by the corresponding amount of original data. All values lower than 0.05 are highlighted.

Experiments	p-values	Compared To
50 with only RGB	**0.0001**	50-0
50-50	**0.0156**	
50-40	**0.0220**	
50-30	0.2173	
50-20	0.9369	
25 with only RGB	**4.7233e-05**	25-0
25-75	**2.4228e-05**	
10 with only RGB	**1.2940e-05**	10-0
10-90	**2.7698e-06**	
50-50	0.1249	100-0
25-75	0.1803	
10-90	**0.0006**	

It can be seen from Figure 5.5 that the absolute mean accuracies of the classification noticeably increases when we add the GAN synthesized data to the training set. In order to verify that, we run a Dependent T-test for the relevant experiments to see if the accuracies significantly change or not. The reason why we choose the Dependent Test is that our statistics are the results of the same experiment (Evaluating a Deep Neural Network) under different settings. So using the Test for Paired Samples is reasonable.

In Figure 5.5, there are 14 different experiments. We definitely do not test every pair of experiments there, but instead only compare the results involving GAN (the green

Figure 5.5: Classification accuracies on Washington Dataset in different settings, trained and tested in 10 stratified random splits. The left number in a pair indicates the proportion of real data, and the right number indicates the proportion of synthesized data added.



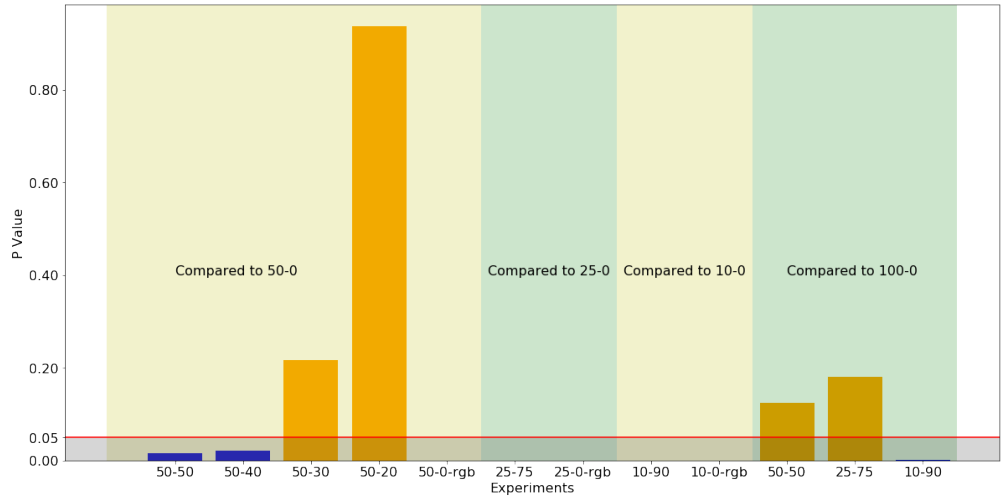


Figure 5.6: Visualization of table 5.1. The blue bars are lower and the orange bars are higher than 0.05. Note: Some values are too small that they do not show clearly in this plot (see table 5.1), but the important point is they are much smaller than 0.05 (the red line)

bars) with the results with original data (the blue bars). In this way, we would like to verify if training with GAN data is significantly better than training without it. A significant level of 0.05 is used in all the tests. The null hypothesis is "the 2 set of results are equal". In this statistical test, we combine both the T-test and the mean accuracies plot in 5.5 to reach a conclusion. If the test says that we can reject the null hypothesis, we conclude that the one having higher mean is significantly better than the one having lower mean.

We can see in Table 5.1, or even more clearly in the visualized version in Figure 5.6, the T-test p-values for the experiments plotted in Figure 5.5. Based on those plots, we can conclude which strategies significantly improve the classification training procedure.

In the case of 50% of the original data, following the significant level of 0.05, we can reject the null hypothesis for the case of 50-50, 50-40, and 50 with only RGB, meaning they are significantly different from the results of 50-0. As the plot in Figure 5.5 shows higher mean values for 50-50 and 50-40 and lower value for 50 with only RGB, it is reasonable to conclude that adding those proportions of GAN data improves training the classifier significantly, and adding depth data to the training indeed adds important values to the network. In this case we add incremental amounts of GAN data (20%, 30% and 40%) to see the trend of GAN's contribution. The accuracy plot 5.5, together with the T-test results 5.1, shows a clear incremental contribution with respect to the amount of GAN data.

For the cases of 25% and 10% of original data, we only experiment with the full amount of GAN data. The p-values are even more significantly lower than 0.05, indicating that the results we have by adding 75% and 10%, respectively, of the training data from our GAN are strongly better than the accuracies without them. The p-values of the training with only RGB still shows the same trend with the 50% case, saying that it is meaningful to use Depth data in training our Object Classifier.

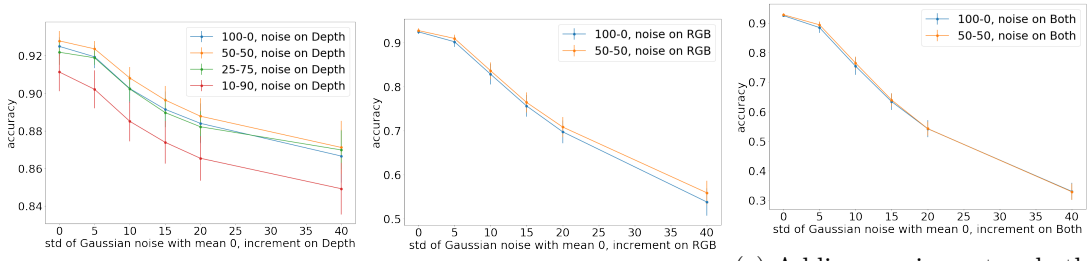
In addition, we also compare the results from full-combination results (i.e. 50-50, 25-75 and 10-90) with the accuracies of the classifier trained from 100% of the original data from the Washington RGB-D dataset. The p-values are in the last 3 rows of Table 5.1. We can only reject the null hypothesis in the case of 10-90. Statistically, we cannot conclude that our accuracies with 50-50 and 25-75 are equal to that of 100-0, but we have a strong base for saying that our GAN data can train a comparable classifier with the optimal case. Only the 10-90 data shows a significantly lower accuracies than 100-0, which is understandable as this Depth-GAN is trained with only 10% of the data which is probably not enough for it to capture the whole data space. It is also one of the limitations of our experiment strategies, which is already discussed in section 3.3.

5.4 Contribution of Depth Data in the Baseline Classifier

In the previous section, we have already seen that the depth data from GAN helps improving the classification performance in an RGB-Depth two-channel Deep Neural Network. However, it was still not clear how much Depth Data contributes to the task. This is important because if the network only learns from RGB data, our previous results

do not make too much sense and do not prove that our Depth-GAN is learning good Depth Representations. A good news is, the previous experiments also revealed that the accuracies achieved using only the RGB channels are significantly lower than with Depth involved.

In this section, we would like to clarify how Depth contributes to the learning process by doing a noise injection to the evaluation procedure of the Object Classification network. We add Gaussian noises with mean 0 and standard deviations of 5, 10, 15, 20, and 40 respectively to the RGB, Depth, and both RGB and Depth channels of the Test Set and observe the behaviors of the Deep Network during the Evaluation process.



(a) Adding noises to depth (b) Adding noises to RGB (c) Adding noises to both channels

Figure 5.7: Noise Injection experiments. When adding noise to Depth, we experiment with the original training set (100-0) and all the combinations involving synthesized depth. In the other two noise experiments, we perform them on only the original training set and the strongest combination of synthesized depth (50-50) as we expect similar behaviors from the combinations of synthesized depth. Note: The scales of 3 figures are different.

Figure 5.7 summarizes the behaviors of different experiments. There are clearly different trends between adding noises to Depth and adding noises to RGB. When adding noises to the RGB channels, the decrement of accuracies is more linear, whereas when adding noises to the Depth channel, it is only linear up to a standard deviation of 20. Between 20 and 40, the decrement clearly vanishes, indicating that we are reaching a limit of Depth contribution. In the same scale of noises, the behavior of the network when we add noises to the RGB channels is different, it is almost a straight line from 5 to 40. Figure 5.7c shows a very similar pattern (to RGB) when we add noises to both channels, which means RGB are the dominant channels here. In addition, Figure 5.8 compares the rate of decrement when putting the noise injection behaviors of the same dataset (50-50) in the same scale. The injection to RGB is much steeper than the injection to Depth.

5.5 Performance of Pose-GAN Data

For Pose-GAN, we train the baseline classifier with only the 50-50 proportion. Although the resulting images look reasonable, they are still not perfect. There are many blurry

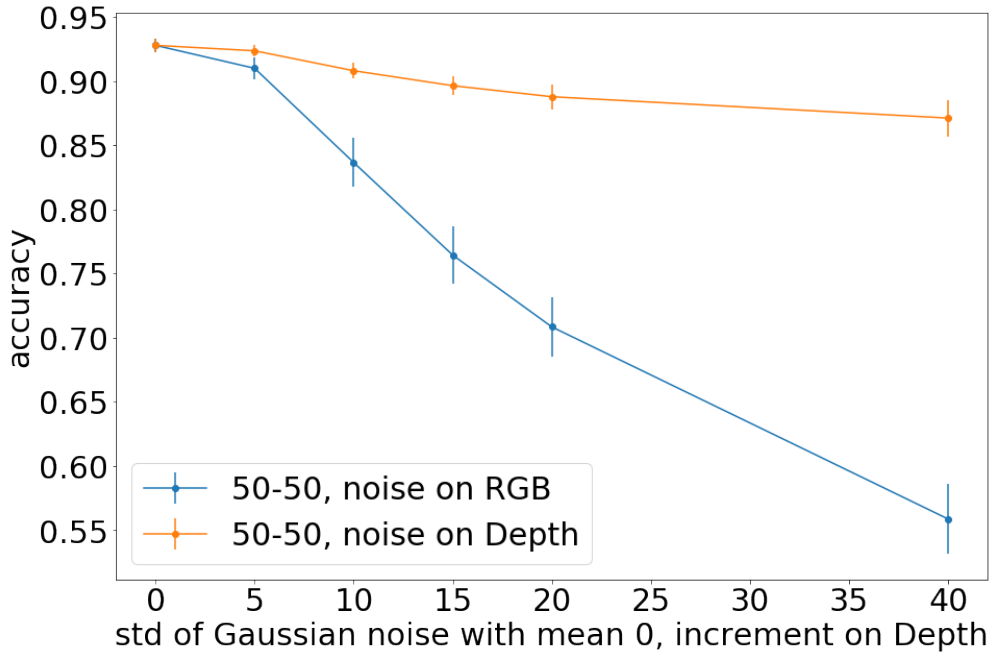


Figure 5.8: Noise Injection on RGB and Depth separately, performing on the same synthesized depth combination and visualized in the same scale.

details, especially in complex objects such as the cereal box with textures and the comb, or in objects with very similar color with the background such as the towel and the rubber eraser. Therefore, it is not a surprise that we achieve only an accuracy of 66% when training the object classifier with the GAN data. As this is a very bad overall accuracy and because of time limitation, we do not go further to perform the same experiments as what we do with Depth-GAN. However, it is probably a good idea to look at the category-wise accuracies to see which categories are easy for the GAN to learn and which are not.

In Figure 5.9, only a few categories are as-well-or-better classified using the data from the Pose-GAN, which means this GAN is still not good enough to capture the entire 3D space representation, although most of the synthesized RGB images are still distinguishable by human eyes (see Figure 5.4).

We demonstrate some categories that the Pose-GAN performs quite well in Figure 5.10. Most of the synthesized objects still have problems with complex details, but some of them are very fine. The calculator, for instance, is well-rotated and can be clearly detected by human eyes; and the details such as the keypad are still adequate. The Bell Pepper is almost perfect, but it is a round object so it is easier for the Generator to learn. The Coffee Mug is successfully rotated, but loses all the sharp textures.

On the contrary, Figure 5.11 shows some examples of the objects behaving much worse in the classifier than the original data. Among the failed ones, only The Light Bulb and the Glue Stick are indeed bad. They do not have a clear shape and are not clearly

distinguishable from the background. The rest of them still look quite nice, such as the Instant Noodles, the Stapler and the Rubber Eraser. The Shampoo and the Food Can are not as good as the two above, but are still better, or at least comparable to the Cap and the Food Box in Figure 5.10. An observed phenomenon is that regardless of how we train the Pose-GAN, the classification results of those categories are still equally bad, although the visual outcomes do improve noticeably as we apply more regularization tricks. This raises an interesting question on why the classifier fails to correctly recognize the synthesized images while they still look very good to human eyes; and at the same time performs greatly with the real counterparts. More work might need to be done in order to find the reason why they behave differently while sharing similar appearance qualities.

Because we remove 17 categories from the Pose-GAN's training set as being mentioned in section 3.1.2, we also try to make it a fairer game by also removing the same categories from the training and test sets of the classifier. However, the results of the remaining categories show no differences from what we have discussed above.

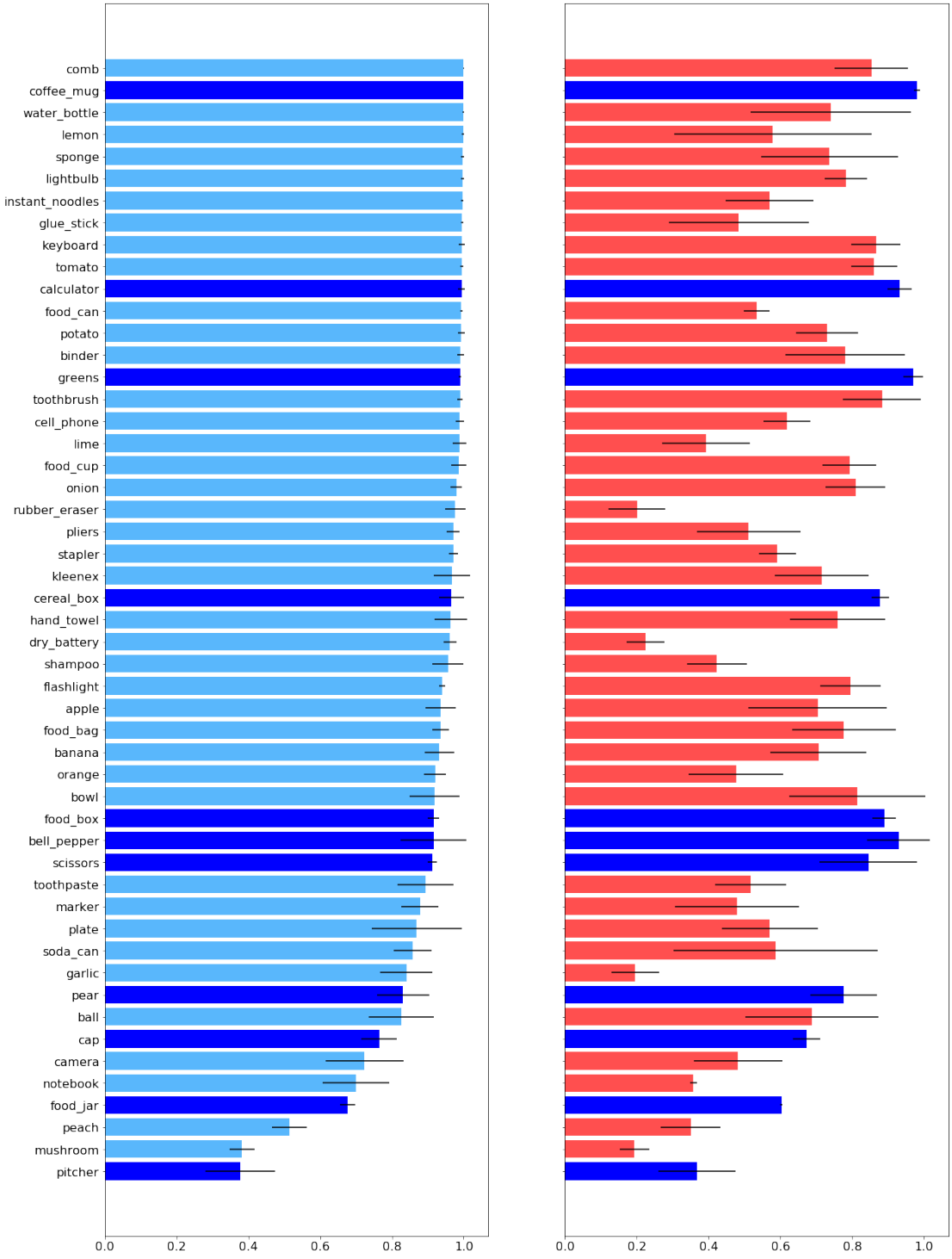
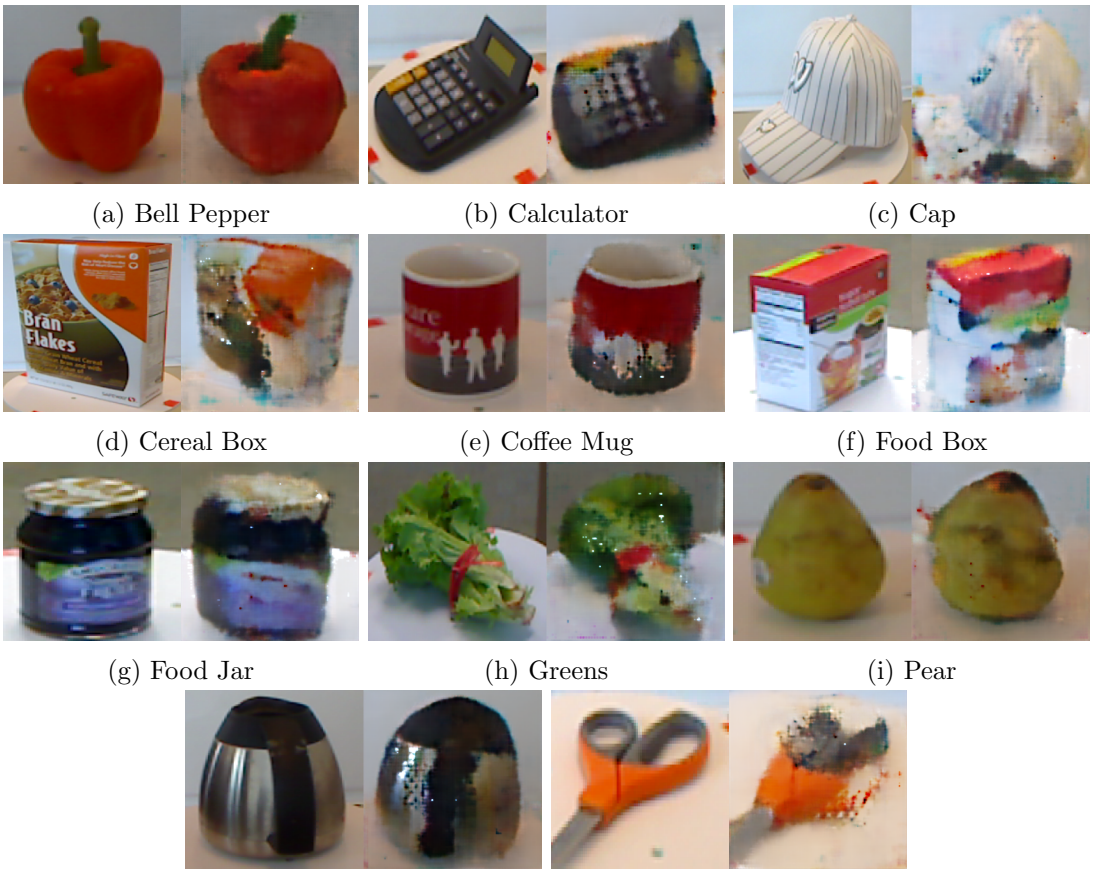


Figure 5.9: Category-wise accuracies of the original data (Left) and Pose-GAN Data in 50-50 proportion (Right). All the categories that the Pose-GAN performance is acceptable are highlighted in blue. By "acceptable" we mean that the accuracies is within a 0.1 error bound compared with the original data.



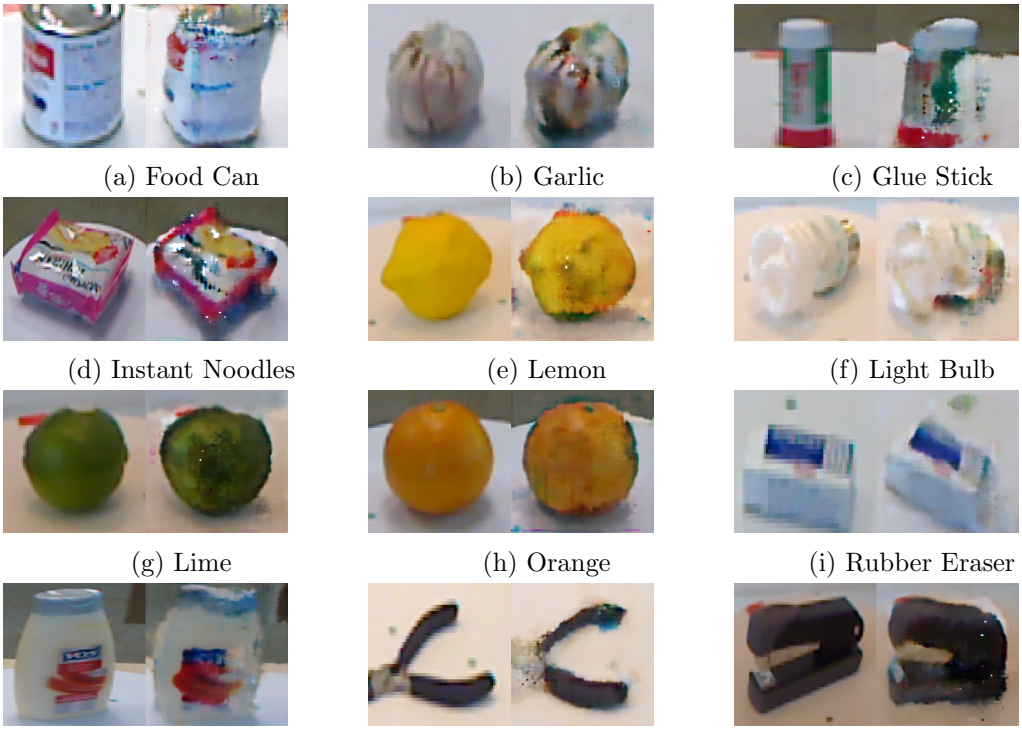
(a) Bell Pepper (b) Calculator (c) Cap

(d) Cereal Box (e) Coffee Mug (f) Food Box

(g) Food Jar (h) Greens (i) Pear

(j) Pitcher (k) Scissors

Figure 5.10: Samples from the categories in which the synthesized objects produce acceptable accuracies, in figure 5.9. On the left are the original objects, on the right are the rotated pose produced by Pose-GAN.



(a) Food Can (b) Garlic (c) Glue Stick

(d) Instant Noodles (e) Lemon (f) Light Bulb

(g) Lime (h) Orange (i) Rubber Eraser

(j) Shampoo (k) Pliers (l) Stapler

Figure 5.11: Samples from the categories in which the synthesized objects does not produce good results in figure 5.9.

6 Conclusion and Future Work

Our experiments show that Depth Data is contributive and adding depth data from our Depth-GAN is beneficial to training an Object Classifier to different extents. It is worth noting that, because of a drawback of our train-test split explained in section 3.3, our GANs creating different proportions are not equal. This is clearly demonstrated in section 5.2.1. However, despite the disadvantages it has, the GAN trained by 10% of the data still performs greatly by improving the classification results significantly, comparing to the classifier trained on only 10% of the original data. The same is true for the GAN trained with 25% of the original data.

Therefore, the Depth-GAN is particularly useful when our "real" training data is not complete and the synthesized data is added in a large proportion, such as the 10-90 case. It is a good idea to synthesize the depth-maps when we have a large training set but only a small portion of images have depth-maps included or there is even no depth-maps at all.

In the way around, if we consider the baseline classifier as a quantitative tool to evaluate GAN data, then our Depth-GAN is showing its potentials. The results are good in both ways: to human eyes, as demonstrated in Figure 5.3, and in quantitative metrics, shown in the accuracies plot 5.5 and T-Test results 5.1.

The Pose-GAN, in another way, only does a good job on some particular objects. It is understandable as its task is much more complicated than the Depth-GAN and other Pix2Pix applications. It is not a simple Image-to-Image translation task, but requires a good representation of the 3D space. Although we have provided the depth map (as the 4th dimension), the Pose-GAN still does not utilize this information well and cannot capture the full 3D space distribution of the objects. A reason could be that this setting (4-dimensional image with 4 channels R,G,B, and D) is still not optimal. A reasonable future work is to try another depth representation when training the Pose-GAN.

Beside some very poorly produced objects, there are some categories which the Pose-GAN still does a decent job, as we have discussed in section 5.5. There is still questions left on why there is such a big difference on classification performance between synthesized and real images although they look quite similar. One possible approach is to inspect the color distribution of the synthesized images and verify if there is any major deviation from that of the real images. A further and more costly step could be tracing the layers of the classifier and/or the Discriminator to spot the points where things go wrong. Because GANs do not minimize a single scalar loss function which the classifier uses, there could be some level of mismatching between the two.

While training the baseline classifier with both the original and the synthesized data, we observe a behavior that the network learns much more from the RGB channels than the Depth one. We tried to regularize the RGB channels in different ways, such as adding

Gaussian Noise to the RGB frame and Dropouts, but could not change the Network's learning behavior. Similar to the point above, using another representation of Depth instead of colorizing and treating it as a second RGB frame, could be the answer and could improve the classification results even further.

Furthermore, as the Discriminator of GANs is trained to perform good classification between real data and the synthesized data from the Generator, it can also contain layers with useful information about the dataset. Another feasible future work is to take the weight vectors of one or a few layers before the Softmax as the representations for other tasks, such as our baseline classification.

List of Acronyms

GAN	Generative Adversarial Networks
ML	Machine Learning
VAE	Variational Auto Encoder
DL	Deep Learning
Pix2Pix	Image-to-Image Translation with Conditional Adversarial Networks
AI	Artificial Intelligence
MSE	Mean Squared Error
CNN	Convolutional Neural Networks
HOG	Histogram of Gradients

Bibliography

[1] Luís A. Alexandre. *3D Object Recognition Using Convolutional Neural Networks with Transfer Learning Between Input Channels*, pages 889–898. Springer International Publishing, Cham, 2016.

[2] Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[3] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin A. Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust RGB-D object recognition. *CoRR*, abs/1507.06821, 2015.

[4] Mohamed Elhoseiny, Tarek El-Gaaly, Amr Bakry, and Ahmed M. Elgammal. Convolutional models for joint object categorization and pose estimation. *CoRR*, abs/1511.05175, 2015.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[7] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[9] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *2011 IEEE International Conference on Robotics and Automation*, pages 1817–1824, May 2011.

[10] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Sparse distance learning for object recognition combining rgb and depth information. In *IEEE International Conference on on Robotics and Automation*, 2011.

[11] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016.

[12] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 689–694, New York, NY, USA, 2004. ACM.

[13] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[14] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[16] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[17] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[19] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised MAP inference for image super-resolution. *CoRR*, abs/1610.04490, 2016.

[20] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[21] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[22] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.