Prediction of Depth Maps using Semi-Supervised Learning

Thesis · January 2020

CITATIONS
0

READS
14

2 authors, including:

Simranjeet Singh
Technische Universität Chemnitz
1 PUBLICATION **0** CITATIONS

SEE PROFILE

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Prediction of Depth Maps using Semi-Supervised Learning

Master Thesis

Submitted in Fulfilment of the
Requirements for the Academic Degree
M.Sc in Information and Communication Systems

Dept. of Computer Science
Chair of Computer Engineering
January 2020

Submitted by:
Simranjeet Singh
Student ID:
391126

Supervising tutor:
Prof. Dr. Dr. h. c. Wolfram Hardt
Supervisor:
M.Sc Shadi Saleh

Abstract

Depth maps prediction is the process of assigning the value of depth to each pixel. The depth maps using the monocular camera predict depth maps till now is an ill-posed problem. In this master thesis, we try to propose a method using semi-supervised learning to predict depth maps from a single image without having any prior knowledge of the surrounding. The classic computer vision is missing cues like the dense depth and in the real-time application is not possible. The development of deep neural network depth prediction from a single image is now the state of the art. There are many existing approaches that use supervised and unsupervised learning cues for depth estimation. In supervised learning is a regression problem which requires a huge amount of the ground truth training data. Whereas in the unsupervised learning which uses a stereo pair of images for training doesn't provide the complete depth information for each pixel in the particular scene. In this master thesis we focus to use the semi-supervised learning which uses both the knowledge from the supervised and unsupervised learning. When training the convolutional network only with ground truth data taken from the lidar the depth information is typically sparse and when training the only with the stereo pair of images the resulted in poor depth information and some the depth information is lost especially in the horizontal horizon. We try to predict depth maps from a single image using the semi-supervised learning approach. This approach is been achieved by using the depth annotated data and stereo pair of RGB images provided by the KITTI dataset. The model is trained with lidar data which provide the sparse depth information and stereo pair of images to achieves high information of depth during the inference. In addition to the training we try to explore the left-right consistency in a stereo reconstruction through a loss function. The evaluation of our model is being tested on the popular Kitti dataset which tests images and corresponding to the test images the depth maps are predicted.

Acknowledgement

It gives me great pleasure to acknowledge completion of my Master Thesis, the satisfaction that accompanies for completion would be incomplete if I do not mention the people who made it possible and whose constant guidance and encouragement crowned my efforts.

Firstly, I would like to thank Prof. Dr. Dr. h. c. Wolfram Hardt for providing me with an opportunity to carry out Master Thesis under the chair of Professorship of Technische Informatik at the Chemnitz University of Technology. In this work, I try to support My supervisor research Mr.Shadi Saleh which focus on the estimating depth information from monocular camera pose. Therefore, this information can be used to evaluate his approach in estimating the distance and depth information.

I would like to express my deepest gratitude to Mr.Shadi Saleh for offering me topic of my interest, for her continuous guidance throughout my thesis and supervising along the way with ideas and solutions whenever I encountered a problem.

I also would like to thank my friends who constantly filled me with confidence and help me relax at times whenever I needed the most. Finally, I would like to extend my appreciation to my parents because of whom I am pursuing my Master's in Germany.

Contents

List of Figures

List of Abbreviations

ARD Absolute Relative Difference

GAN Generative adversarial network

GPU Graphics processing unit

LIDAR Light Detection and Ranging

MAV Micro air vehicle

RAM Random-access memory

RGB Red Green Blue

RMSE Root Mean Square Error

SRD Squared Relative Difference

1 Introduction

This chapter introduces the motivation and outlines the background of depth maps prediction using the semi-supervised learning approach. It briefly reviews the aim of the thesis and gives a small overview of all the chapters in this report. In this section we mainly talk about an understanding of the depth information from a single RGB image. Application of the depth maps and how the depth information is useful for real-world scenarios.

1.1 Motivation and Objectives

Depth estimation from a single input RGB image or estimating absolute or even relative depth without having a second input RGB image seems to an ill-posed problem [11]. In-depth estimation previously lot research has been done which was totally based on computer vision approaches without using the deep learning networks and which depends on the structure from motion, binocular and multi-view stereo setup. In the modern-day with the evolution of the deep neural network which one can hard code any network to perform any task, so the depth estimation using the deep neural network became the course of study. Depth estimation using the deep neural networks helps in understanding the shape of the particular scene from a single image, not depending on its appearance, which is a fundamental problem in machine perception [7].

Depth estimation have much application like synthetic computer graphics [39], synthetic depth of field in computational photography [40], grabbing in robotics [41], using depth as a cue in human body pose estimation [42], modern-day surgery [43], and automatic 2D to 3D conversion in film [44]. Depth estimation also has shown advancement in the MAV system for obstacle avoidance using deep learning technique[9]. Traffic sign estimation [45]. Avoiding colliding risks[46]. Vehicle safety[47]

In the last 10 years, a couple of tremendous research has been taking place in the sector of depth estimation from a single image. They can be majorly categorised in two main sectors based on the approach of computer vision and deep learning. The computer vision approach [48] has its own shortcoming when the model is applied to real-world the results are typical spares. Most of the computer vision approaches are based on the indoor environment that is the reason these approach does not work for the real world scenarios. To compensate for the disadvantages by the computer vision approaches the deep learning sector has evolved and brought new light to the sector of depth estimation. In this master thesis we will discuss the depth estimation only using the monocular camera in brief.

1.2 Problem Statement

The main objective of this Master Thesis is the ensure the best approach for the depth estimation using the deep learning approach of semi-supervised learning. To ensure that the results are evaluated and depth information in the predicted results are justified by the comparison to the state of the art. The prediction accuracy of the depth would also be judged by two parameters one would be the visualisation and others would using the evaluation metrics like RMSE, RMSE(log) and ARD and simultaneously to see what new researches or approaches can develop throughout the process.

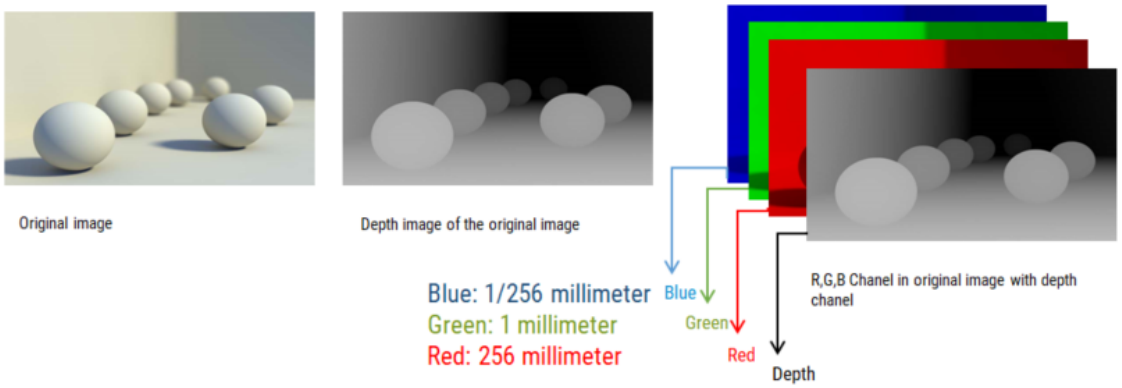

Figure 1.1: *Depth estimation from single image.[1]*

1.3 Thesis Structure

- Chapter 1: This chapter is regarding the basic introduction of the depth estimation and discusses the aim of the master thesis. This chapter will give a close understanding of the depths. This chapter deals with the motivation and objective to find the depth from the single image using the deep learning approaches.

- Chapter 2: This chapter will discuss the state of the art for depth estimation. This chapter guides through the approaches for depth estimation which involves computer vision and various deep learning platform (Supervised learning Unsupervised learning Semi-Supervised learning Self-Supervised learning).

- Chapter 3: This chapter is the benchmark for deep learning basics. This chapter will give an idea of the various process involved in deep learning and will give an idea of underhood maths going between the convolutional layers. This chapter deals with deep nets like ResNet and VGG. This chapter answers to the question which nets should we use for depth estimation and why.

- Chapter 4: This is an important chapter it gives the overview of the term which we will use often in this master thesis. The main understanding disparity to depth relation. This chapter focus on various depths like inverse depth, sparse depth and

dense depth.

- Chapter 5: This chapter answer to question why semisupervised learning is the sate of the art. How semisupervised learning can be used for estimating depths.

- Chapter 6: This chapter is about training dataset which we have used for depth estimation. This chapter give an idea of why the KITTI dataset is important for training purpose. We will also talk about the data set preparation like different splits through which we have trained our network.

- Chapter 7: This chapter is about the network and the algorithm which have used. The chapter deals with every detail like optimization, regularization how to avoid gradient problems. This chapter also deals with the losses which we have used to make the model more accurate. Finally in this chapter will also discuss about the hyperparameter and the environment setup for the training and inference purposes.

- Chapter 8: This chapter is very important as it deals with the result which we have obtained through the training. We will discuss here the results for each different splits like training loss and learning rate. The evaluation of our results with the state of the arts by using different evaluation metrics.

- Chapter 9: This chapter describe the conclusion of our approach and in this chapter we will discuss about the future scope of this master thesis research.

2 Literature Review

In this section we will make some discussion about the advancement which are related to the depth estimation state of art in the previous years. The previous researches will be sorted year wise and will discuss in detail. In this chapter we will talk about the main question which arises when understanding or estimating the depth maps from a single camera. The method to solve the problem of depth estimation using CNN would be discussed and which model or the state of the art performed the best would be described below in the main chapter.

2.1 State of the Art

Depth estimation is an important component to understand the geometry of a given scene and help to compute ways to navigate in free space. Depth estimation provides helpful information in recognition task like obstacle avoidance. In the recent year, a lot of research in sector of depth estimation has been taken place.

One of the important research was done by the A. Wedel et al [49]. The major work was related to find the static obstacle in a traffic scene with the help of depth information. The motion of image sequence which was obtained by the monocular camera was taken as input to provide the depth. The finding was really good for that period of time but they were using the Hypothesis [49] method which totally depended on the Computer vision technique for depth estimation. In 2008 A. Saxena [50] with his team at the Stanford university brought significant research in which they use the multi scale network in combination with Markov Random Field (MRF) and try to predict the depth maps. They used supervised learning approach for training and introduced a new dataset called the MAKE 3D. The approach was first as they estimate the 3D information and orientation of the input images which were divided into patches. They used linear model to predict the plane parameters. The main problem with this approach was that they lack the accuracy of the predicted depth maps at the global scale.

The research in the sector of depth estimation can be categorised into two main approaches. One approach would be by using computer vision techniques. Other would-be using modern-day technique of deep learning combining with computer vision methods to produce an effective model for depth estimation. Figure 2.1 gives an overview of the approaches which are used in depth estimation.

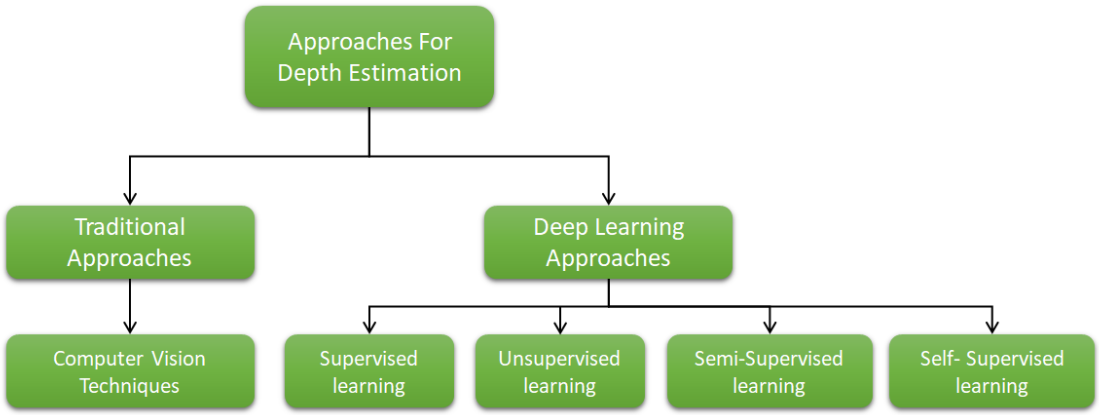

Figure 2.1: *Approaches For Depth Estimation*

2.2 Computer Vision

According to the traditional based approach for depth estimation. The major idea behind depth estimation relates to distance between two corresponding point in left and right image of stereo image pair. This method produces a disparity image in which the image would contain the information of distance to every pixel corresponding to other stereo images. With figure 2.2 we would be able understand briefly about computer vision approach.

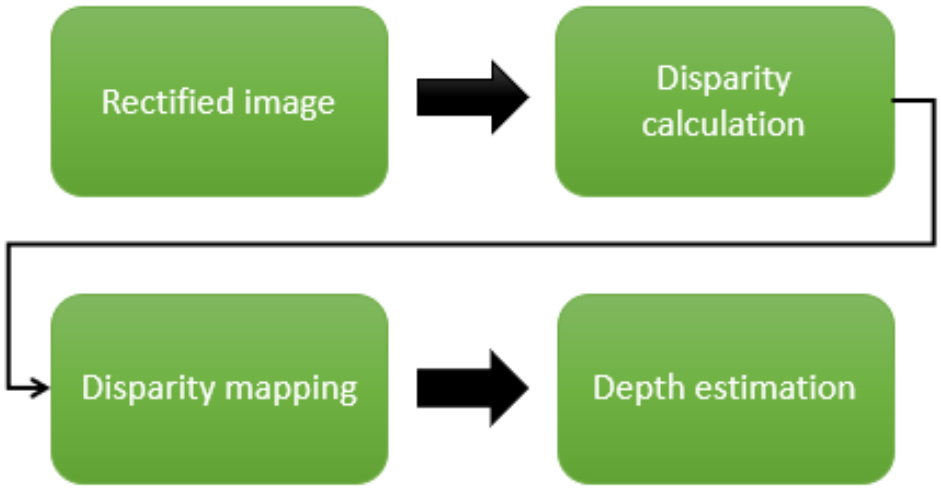

Figure 2.2: *Computer-vision Defocus cue technique for depth estimation[2]*

According to the computer vision approach the depth is estimated using the concept of disparity. One researcher[3] shows that having the stereo pair of images see figure 2.3 and finding the matching points by using some image rectification technique disparity is calculated. With the calculated disparity, the depth information can be generated see figure 2.3.

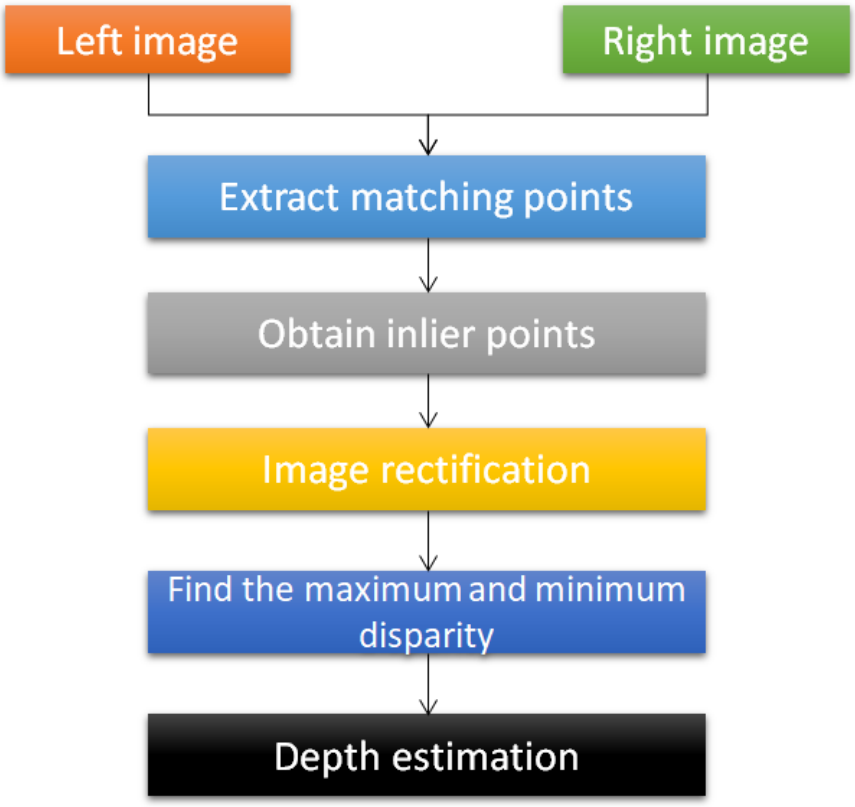

Figure 2.3: *Computer vision approach for depth estimation[3]*

But the main problem with computer vision approaches is that they require lots pre-processing techniques like data augmentation to get rectified pair of images for depth estimation. To counter this problem the modern-day technique of using deep learning for depth shows extraordinary results which will be discussed in this chapter below in section 2.3.

2.3 Deep learning

Deep Learning approaches can be divided into 4 major categories. As we can see that in figure 2.4 each approach brought significant results. The brief detail about each approach would be discussed in this section.

Approach	Associated Person	Organization	Year	Remarks
Supervised learning	Saxena, Ashutosh	Stanford University	2008	Was one of the first to set the benchmark, Multi scale handcrafted features, System relies on horizontal alignment of images
	Eigen, David	New York University	2014	Coarse global prediction based on image, Refine the prediction loaclly
	Laina, Iro	Technische Universität München	2016	No post-processing techniques, such as CRFs or refinement steps ResNet based Encoder- Decoder architecture
Un-Supervised learning	Garg, Ravi	The University of Adelaide, Australia	2016	Requiring no manual annotation, nor calibration of depth sensor to camera The loss is linearized using first-order Taylor approximation and hence requires coarse-to-fine training
	Godard, Clément	University College London	2017	left-right consistency of the predicted disparities in the stereo pair
Semi - Supervised learning	Kuznietsov, Yevhen	RWTH Aachen University	2017	Image alignment as a geometric cue Does not require manual annotations
	Smolyanskiy, Nikolai	NVIDIA	2018	Run at 20 fps in real time system
	Amiri, Ali Jahani	University of Alberta	2019	Uses depth annotated data for tarining
Self-Supervised Learning	Goldman, Matan	Tel Aviv University	2018	Train the network by flipping the images around the vertical axis
	Godard, Clément	UCL, Caltech, Niantic	2019	Predict depth for object in motion

Figure 2.4: *Deep learning approach for depth estimation*

2.3.1 Supervised learning

Supervised learning can be defined as using the fully labelled data for training an algorithm. The labelled data in the training dataset have the information or the answer what the algorithm will come up. For example the labelled dataset of dog images have the images of different dog breeds. The two major areas where the supervised learning are classification and regression problem where the problem is continuous.

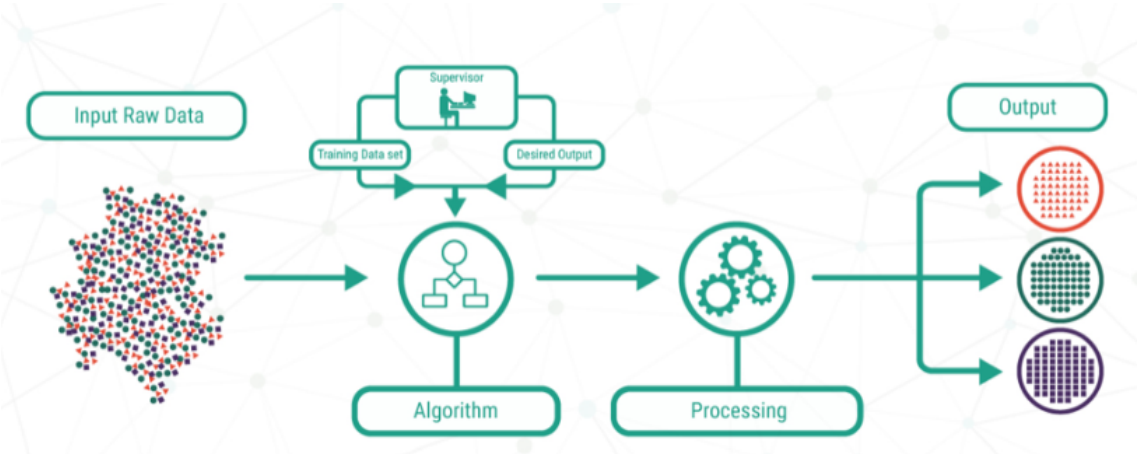

Figure 2.5: *Basic understanding of Supervised learning[4]*

Saxena, Ashutosh [50]

According to the research done by [50] using the Markov Random Field (MRF) they set a certain set of parameters. Using the all the important set of parameters, they are able to get the 3 D orientation and 3D location of the patch in the image. They trained the Markov Random Field with supervised learning approach, through which they are able to predict the depth in the image and relation to the different part of the image.

Eigen, David[51]

Eigen et al[51] at New York University proposed a method that uses the multi-scale deep network to predict depth from a single image. Presented method uses two stacks of deep networks one stack of the network predicts depth at the global scale and the global scale network is connected to the fine-scale network which predicts the fine depth. Eigen et al[51] set the benchmark(discussed in Chapter 5 section 5.3) for training and evaluating the model by splitting the data for training, evaluating and testing the method through various evaluation techniques like RMSE, RMSE(log).

Li, Bo [52]

According to the research [52] proposed a framework that works at two levels. One level at super-pixel and other at pixel level. Li et al proposed to extract the feature using DCNN on an image patch using the CRFs as processing technique at super-pixel segmentation of the image. The relative depth obtained at super-pixel level is further refined to the pixel level by using the hierarchical CRFs technique at every pixel in the image. The research uses lot of processing technique for depth estimation and for real time system it hard to predict depth.

Liu, Fayao [53]

In 2015 year researcher Liu, Fayao [53]proposed a method to overcome the shortcoming of research done by [52]. In his method developed a method that uses the deep continuous CRF model to learn unary and pairwise potentials in the image.

Laina, Iro [5]

Laina, Iro [5] brought major breakdown in the research of depth estimation from single image without using the post processing technique like CRFs as used in previous researches [52] [53]. [5] trained the model end to end to predict depth. To provide better depths they used the encoder decoder architecture for up-scaling the information in the network. As in normal architecture one would have fully connected layer at end, so they replaced fully connected layer with small block of up-projection see in the figure 2.6. For optimization they used reverse Huber loss.

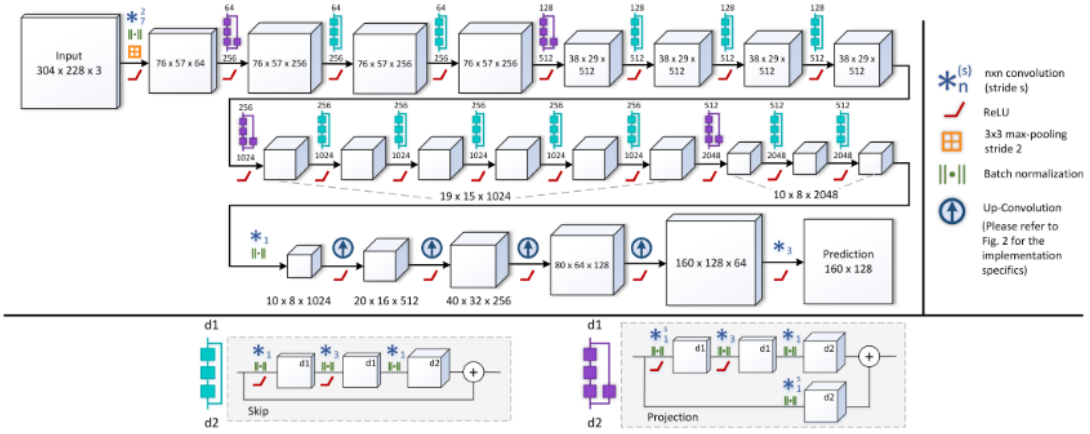

Figure 2.6: *ResNet-50 architecture (encoder-decoder) proposed by[5]*

2.3.2 Unsupervised learning

The model using unsupervised learning are trained with the data which is unlabelled which means no ground truth information is available. The model has to judge itself on how to handle the unlabelled data without having knowledge about the data. The neural network needs to extract the feature itself and understand its structure. The unsupervised learning model is used in clustering, Anomaly detection.

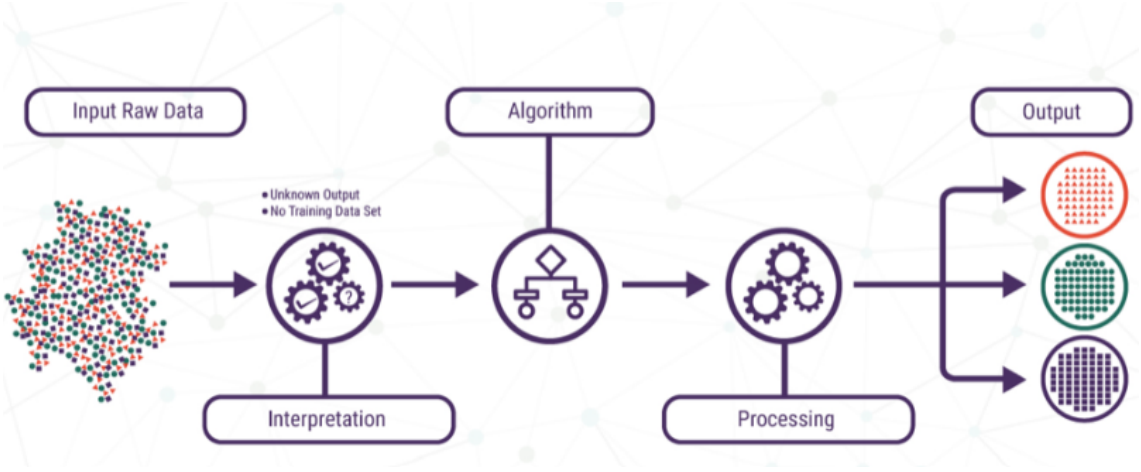

Figure 2.7: *Basic understanding of Unsupervised learning[4]*

Garg, Ravi [6]

Garg et al [6] uses the image alignment loss to predict depth from a single RGB image. The network is divide in into encoder and decoder structure which have skip connection. The method uses unsupervised learning which only require the stereo pair of images for depth estimation. Network uses the photo consistency loss for training, due to the input image is wrapped to the associated stereo image through using the predicted depth. The proposed model is not fully differential[7]. See bilinear sampler in section 7.3.2 for understanding what does fully differential means. For optimization they used reverse Taylor approximation which is a challenging task.

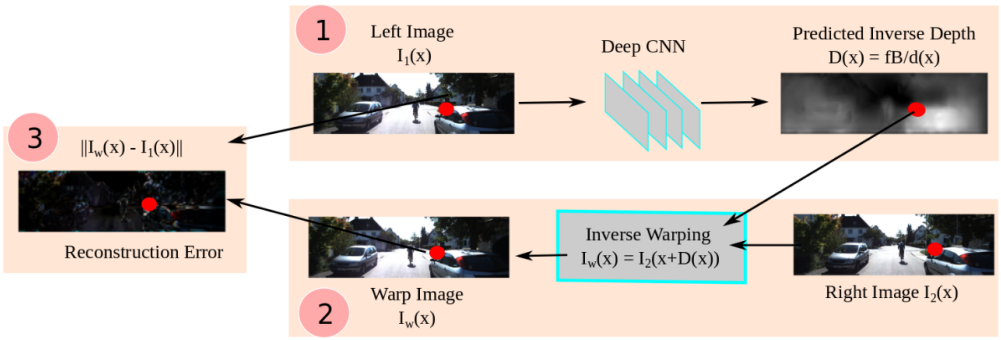

Figure 2.8: *Garg, Ravi [6] method of depth estimation*

Godard, Clément [7]

Godard et al [7] proposed a method that uses the image construction loss to train the encoder-decoder architecture. The training loss incorporates the photo consistency between the disparities predicted which were relative to the left and right image.

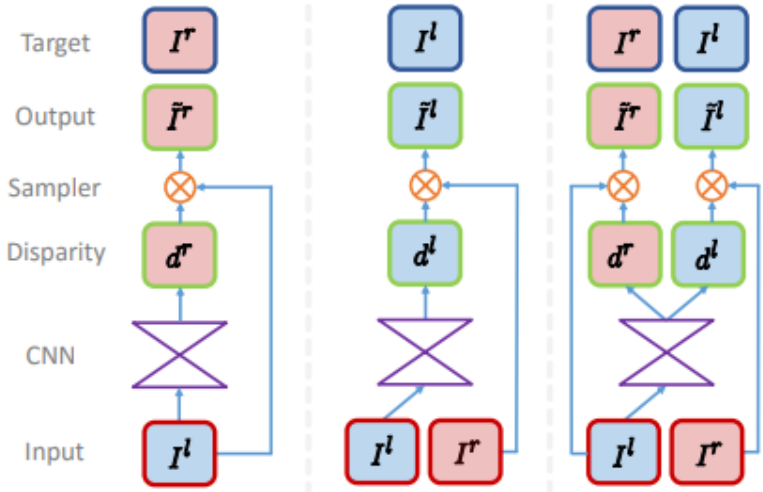

Figure 2.9: *Godard et al[7] method of depth estimation*

2.3.3 Semi-Supervised learning

In Semi-Supervised learning the training dataset contains both labelled and unlabelled data. The method is useful when the feature extraction and labelling the data is difficult. In some aspect when the ground truth data is not easily available or in some sense when the data is very less the semi-supervised learning can bring a more reasonable approach for training and achieving the required results

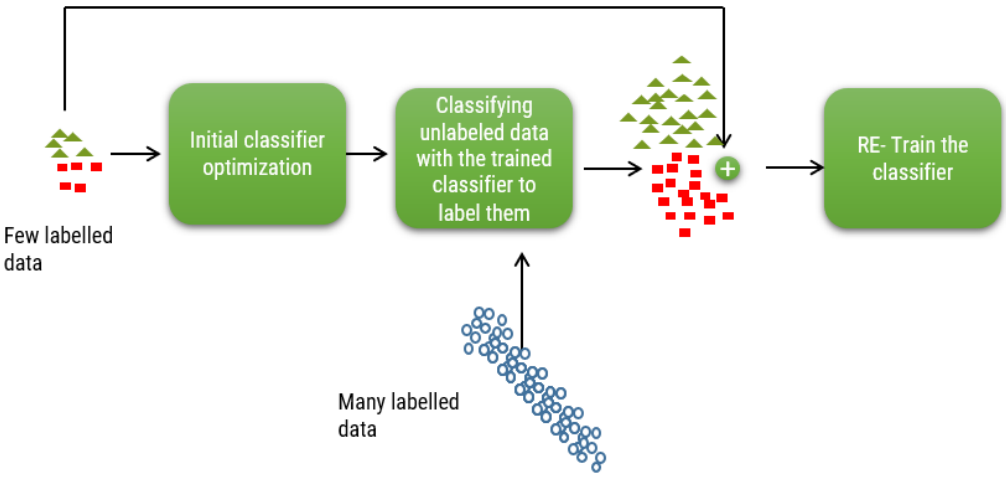

Figure 2.10: *Basic principle of Semi-Supervised learning [8]*

Kuznietsov, Yevhen [54]

In Recent year lot of researches have been taken place by using the deep learning approaches one of the major research done by Yevhen Kuznietsov et al[54] at University of RWTH Aachen University combined both the methods using supervised and unsupervised learning approach for depth estimation. Yevhen Kuznietsov et al[54] used the sparse ground truth and stereo setup of images for training purpose using popular kitti dataset. The results were outstanding by comparing to the state of the art. The supervised loss in Yevhen Kuznietsov et al[54] research was directly defined on the depth values not on the inverse depth[10]. The loss on depth values makes the training unstable at early stage due to its high gradients[10]. Proposed [54] to gradual fade in supervised loss to overcome the problem of the high gradient.

Smolyanskiy, Nikolai [9]

The research done by Smolyanskiy et al [9] estimated depth information which was applied to the a task for obstacle avoidance in Autonomous Driving. The method uses Stereo DNN models for estimate depth from the stereo camera in a real-time system shown in figure 2.11 and figure 2.12.

Figure 2.11: Image from camera[9].

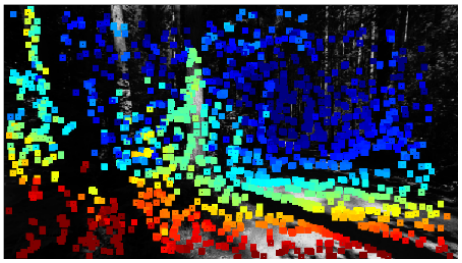

Figure 2.12: Sparse Depth map [9].

Amiri, Ali Jahani [10]

In 2017 Amiri et al [10] proposed a method that incorporates the work done by[7] using the approach of semi-supervised learning by introducing the depth annotated data for training see figure 2.13. The model also used the left and right photo consistency loss for training as proposed in [7].

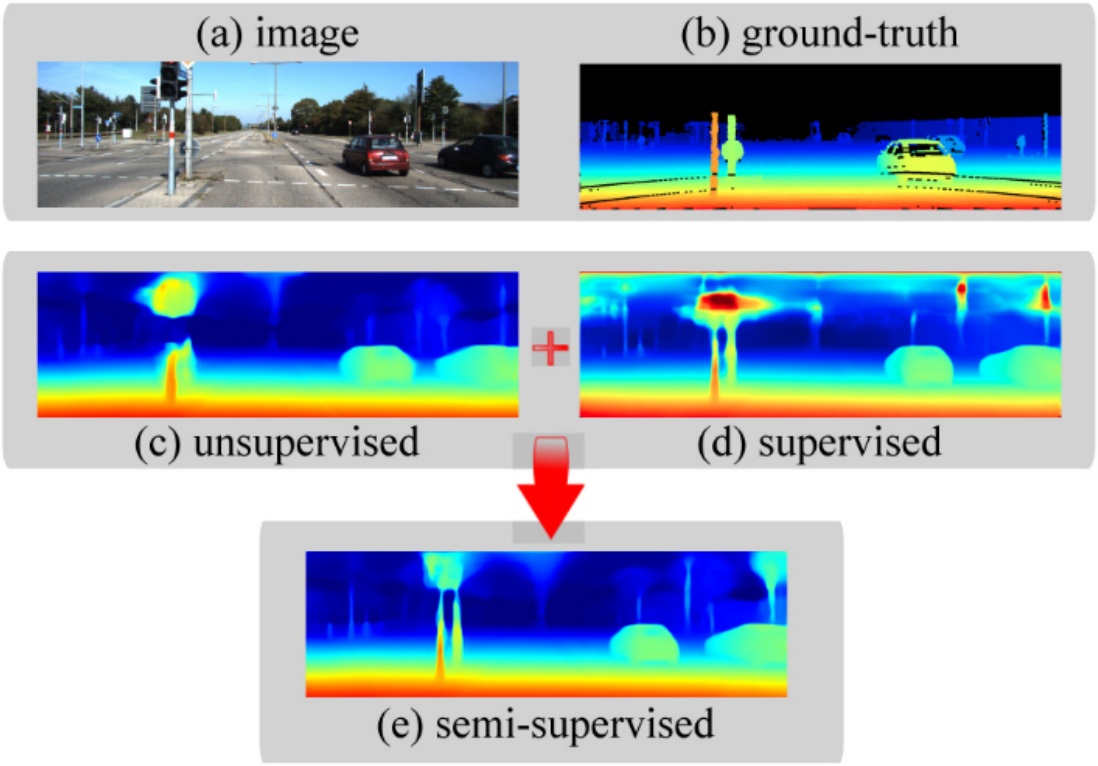

Figure 2.13: *Amiri et al[10] semi supervised approach for depth estimation*

2.3.4 Self-Supervised learning

Self-Supervised learning is the process which uses unsupervised learning cues expect the data in which the data provides the supervision. To describe self-supervised learning the information regarding the data is suppressed and give the task to the model for predicting it[55].

Godard, Clément [11]

Self-supervised approach shows versatile model that predicts depths from a monocular vision approach. As self described the training doesn't include ground truth data. so in this research Godard et al[11] proposed to use either the stereo or monocular images for training and delousing the depth information and projecting to the nearby views to train the model using the image reconstruction loss . Godard et al[11] trained the network to predict the appearance of the image from the viewpoint of another image[11] see figure 2.14.

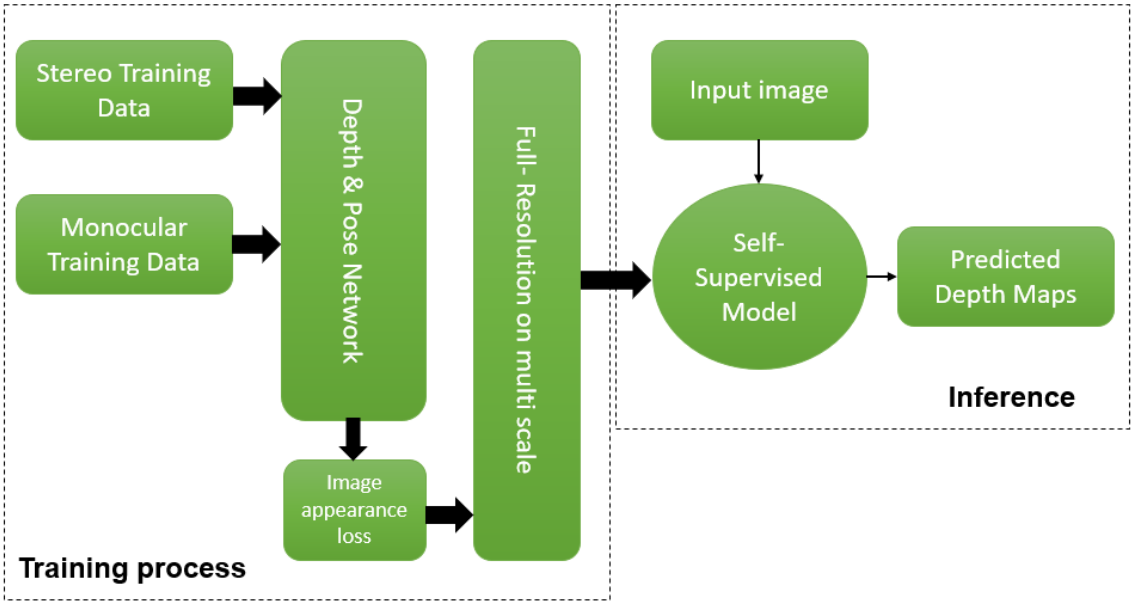

Figure 2.14: *Godard et al [11] approach for depth estimation using self supervised learning*

Goldman, Matan [12]

Goldman et al [12] Proposed a method to train the network by flipping the images around the vertical axis. The training of the model gives a left disparity map and they obtained the right disparity by just flipping the right image. This lead to disparity map between left and right image.

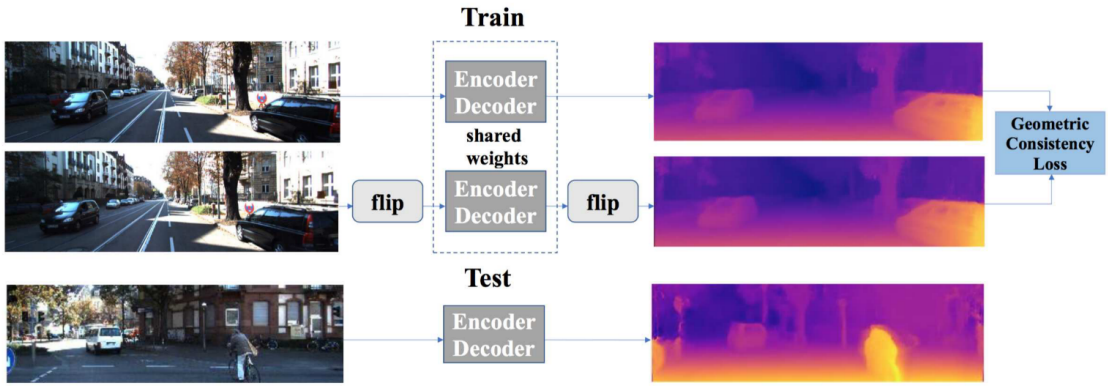

Figure 2.15: *Goldman et al [12]approach uses stereo data during training*

After discussing all the past researches we can conclude the depth estimation from its earliest days to the present day all the researchers are trying to use deep learning and traditional computer vision approaches to estimates the accurate depth maps. We conclude the depth from the supervised learning does not give a robust model as lack lots of depth information to get the 3d geometry of the scene. Supervised learning is totally based on the depth information taken from an expensive sensor like LIDAR for training. The main problem arises the to get the ground truth data (depth images taken from

lidar) and that is a pretty expensive and time-consuming task. The information taken from lidar is also sparse which does not get the good sense of the environment like a shadow, trees, occlusion of objects and etc. This concludes that the depth from supervised learning misses the information and the model after training is not robust. Next, we come to unsupervised training which is totally based on the stereo pair of RGB images. The beauty of unsupervised training is that we do not have to worry about the training data. In unsupervised training, the training data is easily available which in our case is the stereo pair of images. Unsupervised learning is based on computer vision techniques like disparity calculation and using the calculated disparity to find the depth. The only main disadvantages of the unsupervised training are the lack of supervision because the model sometimes is not able to judge some regions of the scene which leads to noisy depth. Now we come to semi-supervised learning which use the stereo pair(unsupervised learning) of images to tackle the poor depth resulting from the stereo pair of images and depth images(supervised learning) to tackle the from the gap in the field of view of cameras and LiDARs. Hence the semisupervised learning is becoming the state of the art which provide robust model, therefore, leading to accurate depths.

3 Deep Learning Basics

This chapter will talk about the fundamentals of deep learning. Deep leaning can be described as a sub-part of machine learning. To bring more clarity to deep learning we will talk about the neural network, how the problem of vanishing gradient leads to development of the advanced convolutional neural network. In this chapter, we will also see the under-hood mathematics of the Convolutional neural network and which network we choose in this master thesis of depth estimation would also be discussed.

3.1 Neural Network

Deep learning is based on neural networks. The neural network consist of nodes and edges. The nodes are called neuron which is interconnected with the edges. The figure 3.1 gives an idea what a neural network looks like.

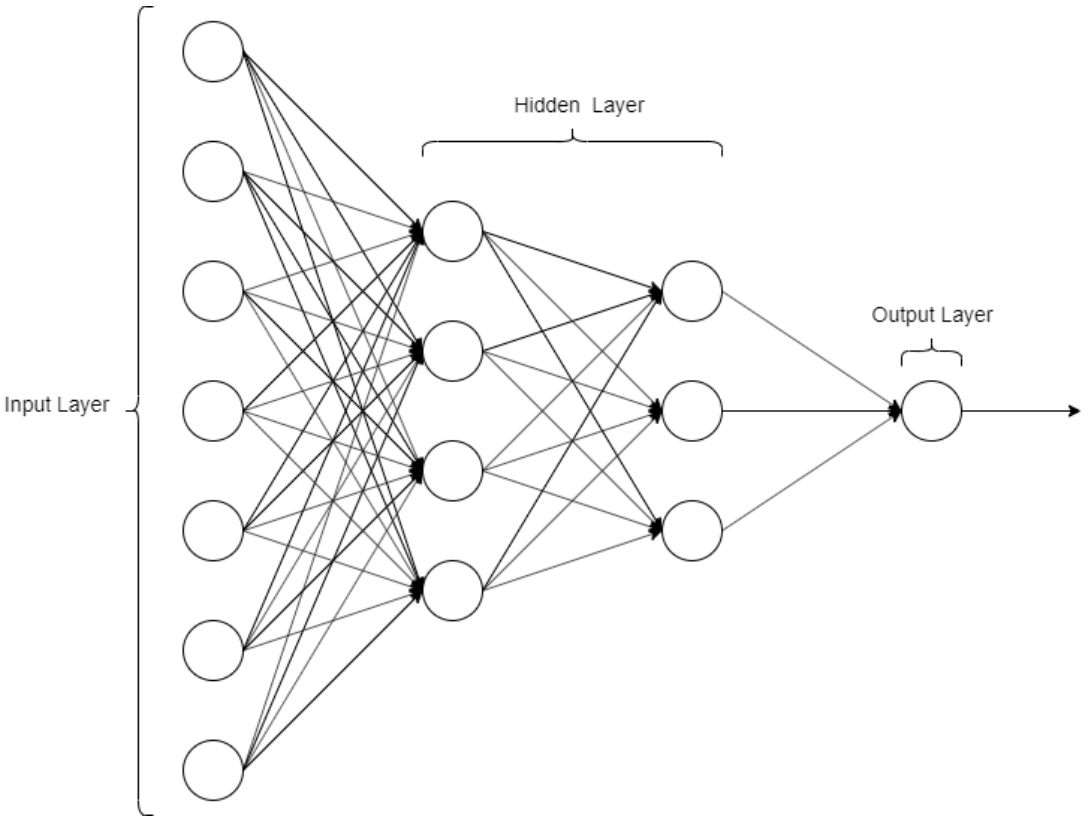

Figure 3.1: *Depicting the layer inside a neural network[13]*

To understand the neural network think of your brain. Our brain consists of 100 billion neurons[56] which help us to overcome a lot of complex tasks. On the one hand the com-

puter is a systematic process which requires an algorithm to complete a task. The main objective is to use the human brain as connecting line which leads to the development of the neural network, without an algorithm to complete a task. As we try intimidate neural network as human brain which is capable of leaning itself, allowing parallelism. The computer doesn't have the capability to learn itself to do a task rather it works on a set of commands which are human-annotated. Figure 3.2 shows the different parameters of human brain vs a computer. We can see that the human brain has more processing human and computer have fewer units. On the same hand the neural network learning like a human brain, is capable of leaning and judging at a much faster rate. The switching time of the neural network to predict in comparison to a computer which uses an algorithm to predict the same thing is faster.

	Brain	**Computer**
No. of processing units	$\approx 10^{11}$	$\approx 10^{9}$
Type of processing units	Neurons	Transistors
Type of calculation	massively parallel	usually serial
Data storage	associative	address-based
Switching time	$\approx 10^{-3}$s	$\approx 10^{-9}$s
Possible switching operations	$\approx 10^{13}\frac{1}{s}$	$\approx 10^{18}\frac{1}{s}$
Actual switching operations	$\approx 10^{12}\frac{1}{s}$	$\approx 10^{10}\frac{1}{s}$

Figure 3.2: *Comparison between human brain vs computer [13]*

To understand a neural network, we should first understand the working components of the neural network. A neural network consists of many processing units and neuron which are connected through weighted connections. In mathematics term, we can define a neural network like (N, V,w). In this N, V are two sets and w are the weight function. The N sets contain neuron and V sets contain the connection from one neuron to another neuron. The figure 3.3 shows the basic idea of neural network working. Sets of a neuron are given as inputs to the propagation function as we talk about neuron there would be multiple of neuron which would be leading to the same connection. The propagation function helps transfer a neuron to its output location and the result of the propagation function is called network input. A neuron can be described as active or excited based on the nature of model. Neuron from the propagation function reaches an activation state which describes the extent of the neuron activation and is often referred to activation. Then activated neuron reaches the output function which calculates the value which is transferred to other neurons from the root neuron.

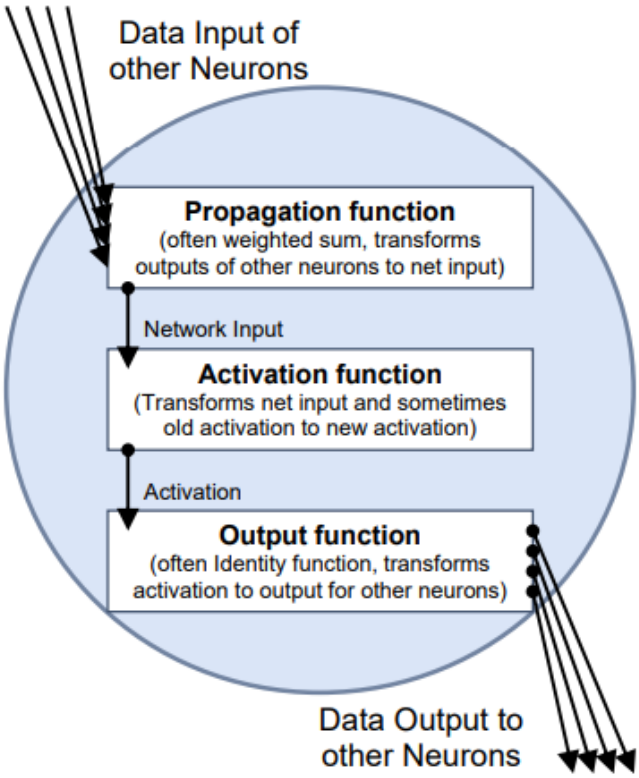

Figure 3.3: *Comparison between human brain vs computer [13]*

3.2 Biological Neural Network

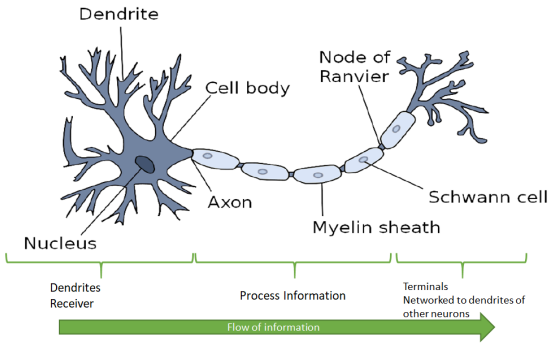

Figure 3.4: Biological Neuron.

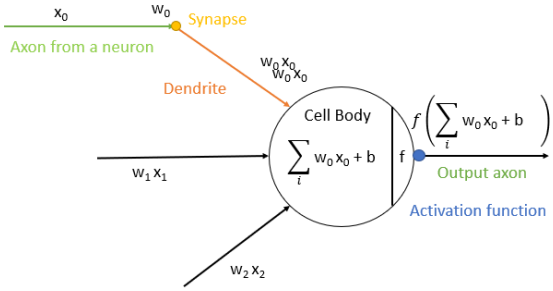

Figure 3.5: Mathematical Model.

To understand more deeply about the neural we have to understand more about biological neural network. Our brain has a huge system which has interconnected neuron through which one information is transferred from one point to another point. The biological neural Figure 3.4 and 3.5 consist of four major components called Dendrites, Nucleus, Axon and finally Axon terminal. The flow of information from A point to point B. The starting point(A point) is called dendrites which are also called the receiver. The dendrites get the information from the outside world. The information gathering from

the outside world is processed with in the cell body. The processed information reach to the final point (B point) which is known as terminal. The terminal is a junction where the refined information is transferred to other neurons.

3.3 Artificial Neural Network

An artificial neural network have many processing units which are known as neurons. The main motive of an artificial neural network is to replicate a biological neural network. As described in the biological neural network above the artificial neural network also consist of inputs, processing units with activation function and finally the output. In figure 3.6 $X_1....X_n$ is the inputs of the neuron. To the inputs neuron some bias value is added which generally initialized as to 1. $W_0....W_n$ is the weights. The weights are defined as the connection to the signal. The multiplied results of the input and weight give the strength of the signal. At the final the neuron receives inputs from many sources and at the end it has only one output.

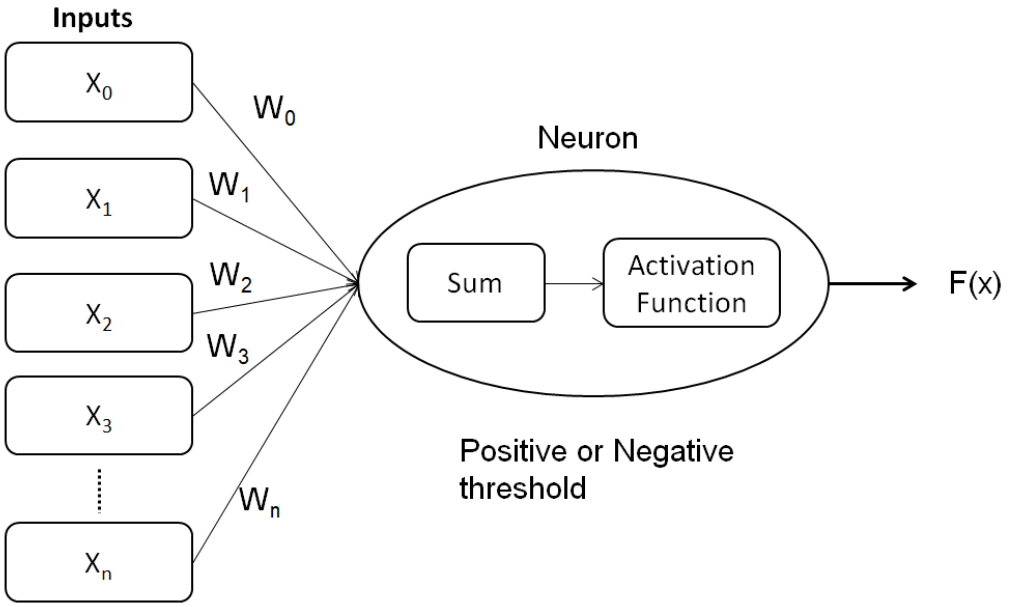

Figure 3.6: *Model of an Artificial Neuron[14]*

3.4 Convolutional Neural Network

The convolutional neural network can be described as a feed-forward network which is used in much application like image analysis and classification task. The convolutional neural network is made up of stacking layers like pooling, convolutional and fully-connected layers[57]. The convolutional neural network has neurons that have learnable weight and bias. They take an image as input and extract the feature from the input image therefore the extracted features are encoded in the architecture. To describe a neuron has 3 dimensions i.e. height, width and depth.

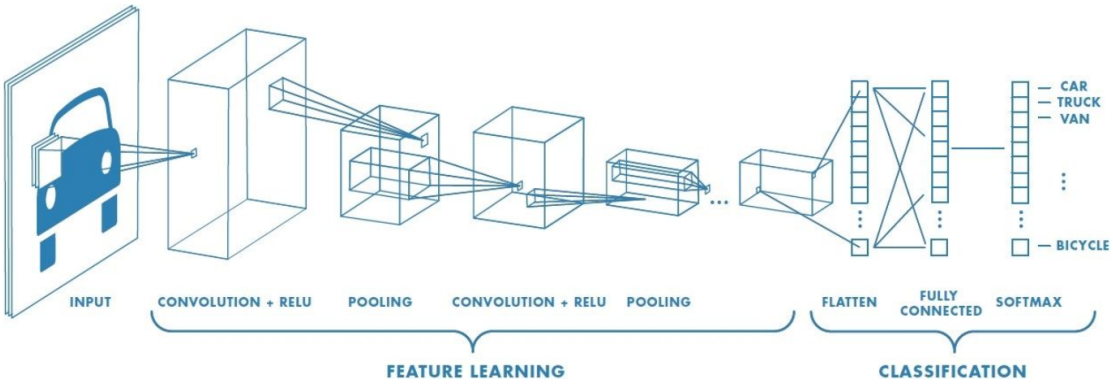

Figure 3.7: *A basic architecture of CNN*

3.4.1 Image

To understand the convolution we have to understand an image. An image can be defined as array or a matrix of vales which are related to the each pixel value. For better understanding see the figure 3.8 on the left-hand side an image of height and width of $17 * 17$ corresponding to it on right-hand figure 3.9 shows its pixel value for each pixel in the image.

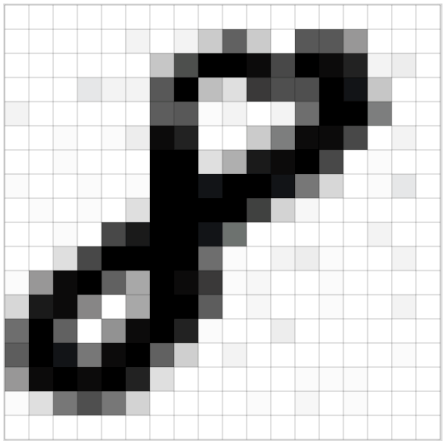

Figure 3.8: What wee see[15]

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	12	0	11	39	137	37	0	152	147	84	0	0	0
0	0	1	0	0	0	41	160	250	255	235	162	255	238	206	11	13	0
0	0	0	16	9	9	150	251	45	21	184	159	154	255	233	40	0	0
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11	0
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0
0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9	0
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0	0
6	23	112	157	114	32	0	0	0	0	2	0	8	0	7	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3.9: What computer see

A correct representation of image is defined by 3 components (height, width, channel). In general image from a digital camera have three channels – red, green and blue. One can think three 2d matrices stacked over each other see figure 3.11.

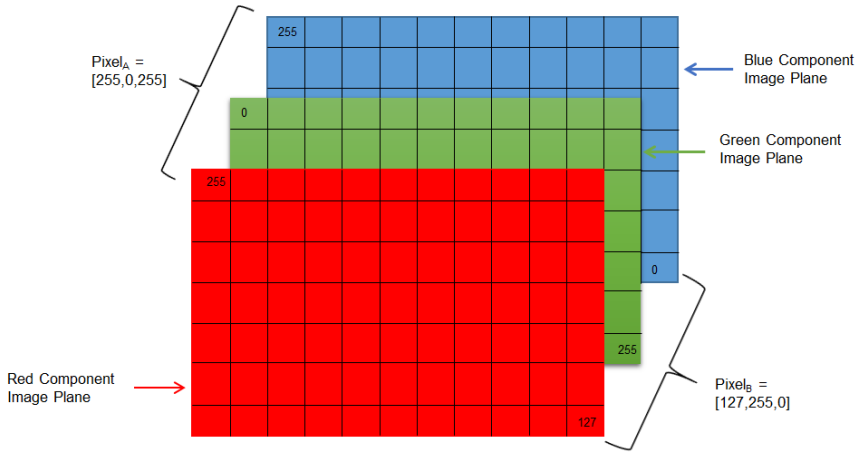

Figure 3.10: *A Pixel of RGB image.*

we can see in the figure 3.10 above the pixel(A) has a value of (255,0,255). The pixel(B) has a value of (127, 255, 0). Both the pixel A and B have a combination of intensities of red color, green color and blue color.

3.4.2 Convolution

Convolution can be defined as a mathematical operation which is performed are over the two functions. After the convolution a new function is generated which expressed as how the shape of 1st function modified by the 2nd function. The main purpose of the convolution layer is to extract the features from the input image. The feature extraction is done by some sets of filters and is generally knows as kernels or filter. The kernels are of n*n size and they slide over the input image see figure 3.11 and performs the convolution over each segment of the image. The obtained matrix received after the convolution of input image to kernel is called feature map.

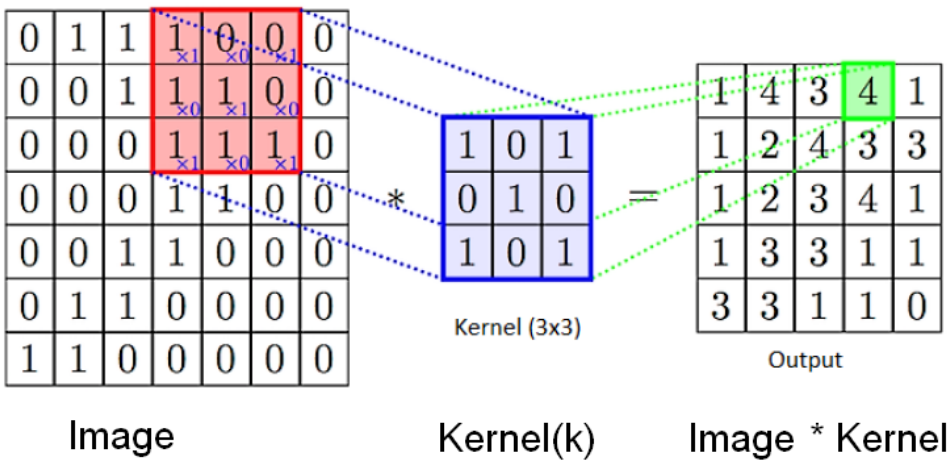

Figure 3.11: *Convolution performed on an image of 7x7 with a kernel of size 3x3 [16]*

The obtained feature map depends on the three main parameters known as depth, stride and padding. The depth parameter defines the number of filter or kernels which has been used for the convolution function. In the figure 3.12 below the convolution has been performed with three different filters and thus produces three distinct feature map. In this we have three different filters which produce three different feature hence the depth is three.

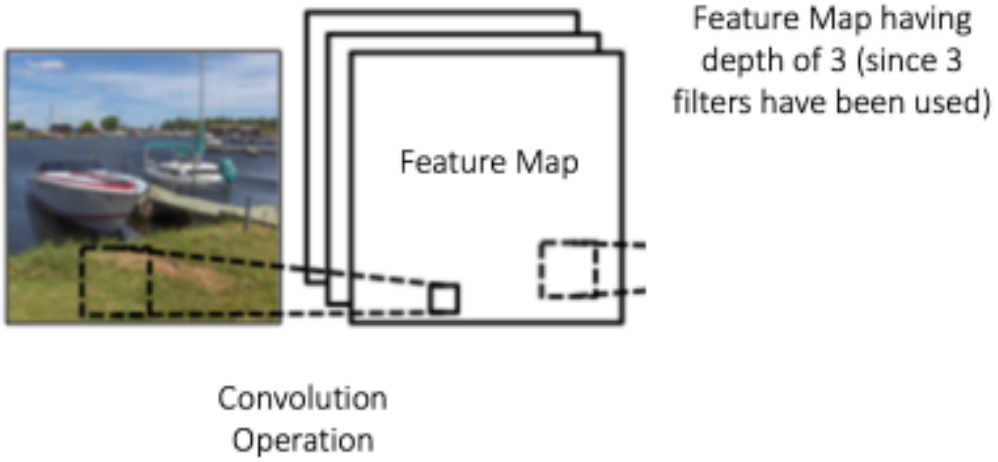

Figure 3.12: *Feature map having depth of three [17]*

The stride parameter can be described as siding window which slides over the input image matrix. For example when we set the stride as one the sliding window moves one pixel at a time. Padding is generally adding the border to the input matrix it generally of zero padding means adding zero to the borders of the given matrix. Adding padding help to control the size of the feature maps.

3.4.3 Pooling

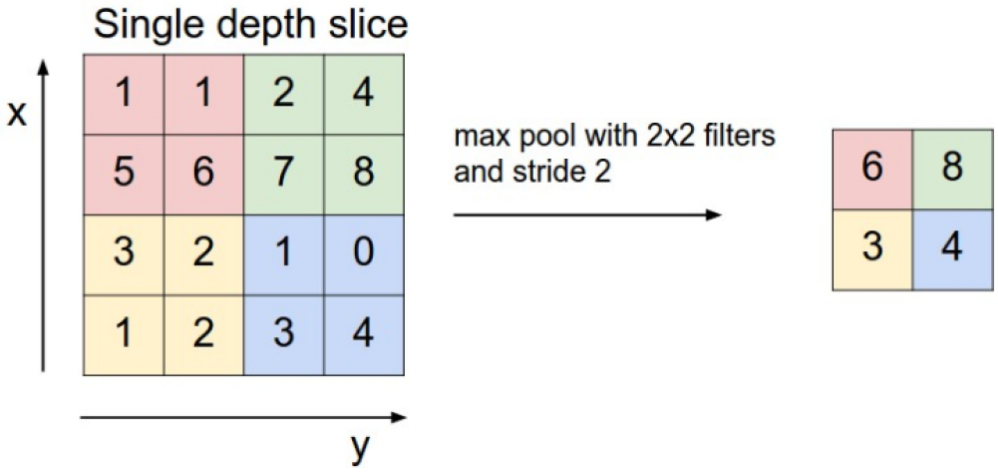

Figure 3.13: *Pooling [18]*

The pooling layer is added after the convolution layer. The basic idea behind the pooling layer is to sub sample the inputs. The general polling layer which is used is called max pooling layer. The max pooling layer help in reducing the spatial size of the outcome which comes from the computation in the convolution layer. This help further in reduce the computation time . The max-pooling layer help in reducing the number of parameters. This reduction of parameters help in increasing the speed up time for computation. In figure 3.13 uses the max pool layer to reduce the number of parameters to compute. The figure 3.13 uses the max pool layer of size 2*2 with stride of 2 max pool layer takes the maximum number out of each colour box in above figure and we obtained a 2*2 matrix from a 4*4 matrix.

3.4.4 Activation function

Activation functions are math equations that specify the outcome of a neural network. This function is associated with each neuron in the network and specifies whether it should be activated (”fired”) or not, depending on whether the input of each neuron is relevant to the prediction of the mode. The activation function $F(x)$ defines a relation between the states of the neighbouring neurons and the state of the neuron under study[58] To activate a neuron there are many activation functions but the most common activation function are Relu[59] and Sigmoid function[59].

Sigmoid function

In mathematical term, the activation function can by equation 3.1[59] and 3.2[59] .

$$F(x) = \frac{1}{1 + e^{-sum}} \tag{3.1}$$

$$sum = \sum_{i=0}^{n} X_i W_i \tag{3.2}$$

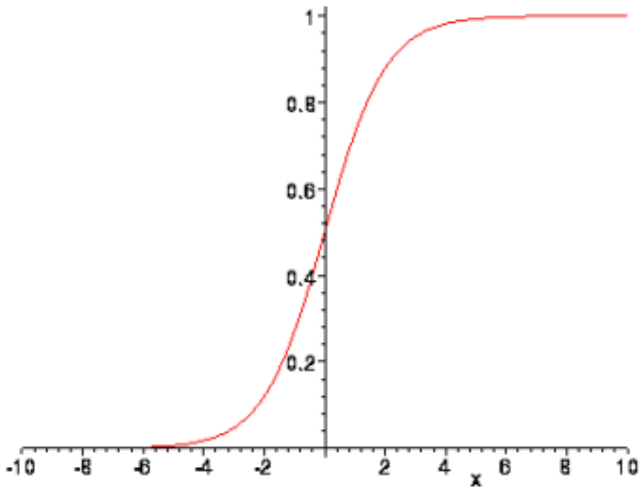

Figure 3.14: *Activation function Sigmoid [19].*

The activation function is a combination of summing the given inputs with the some bias added to it see equation 3.2. The activation function looks at the best results and then activate the weighted sum. The range of the sigmoid function is from 0 to 1. The sigmoid function is an approximation of the step function continuously and differentially.

ReLu

Rectified linear unit function (figure3.15) in short know as ReLu is widely used activation function for many deep learning applications. Relu is an element-wise operation and in mathematical term ReLu in equation 3.3[59].

$$F(x) = max(0, x) = \begin{cases} x_i & if x_i \geq 0 \\ 0 & if x_i < 0 \end{cases} \tag{3.3}$$

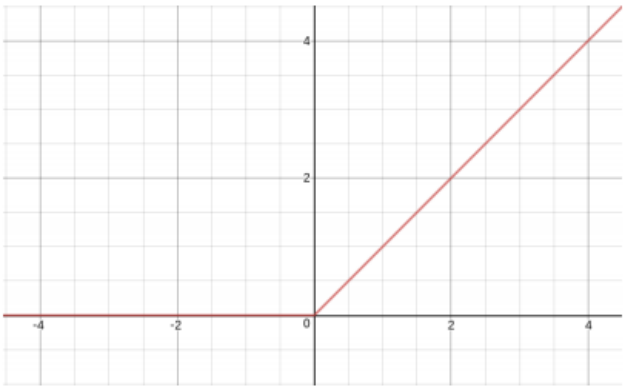

Figure 3.15: *ReLu activation function produces zero when x is less than 0 and produces linear function when x greater than zero.[20]*

The most interesting thing Relu does it replaces all the negative values in the feature maps with zero. There are many advantages of using ReLu as activation function as it uses rectified linear units in its computation, therefore, leads to the faster computation rather than computing exponential and division which leads to increased computation time. However Relu has its own disadvantages like easily over-fit and it's sometimes fragile as in training it leads dying gradient[59].

3.4.5 Fully connected layer

In fully connected layer all the neuron have connections to all activation's in the previous layers and their activation are generated by through matrix multiplication followed by bias offset [18]. In fully connected layer each layer is connected to another layer which transfer important information between these layers. In general, the previous layer is attached to all the neurons in the next layer. The connection is an important model where the high-level information is being shared. The high-level information is then retrieved by the fully connected layer which is the main layer to decide which class

the object belongs to. In the figure below 3.16 a car was given as input for training after extracting the important features from the image the convolution and pooling layer store the important information of the features and fully connected layer classifies the image as car based on the training data.

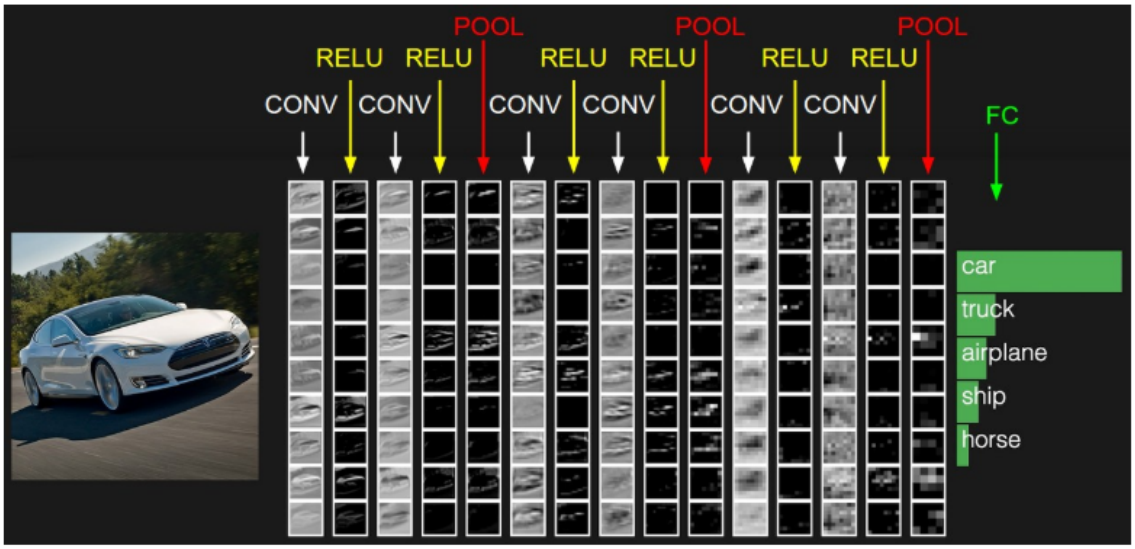

Figure 3.16: *Fully connected layer [18].*

3.4.6 Back-propagation

Backpropagation is a key aspect of the neural network as the help in finding the gradient and loss function in respects to weights of the network.In more simple word the back-propagation helps in getting correct predictions. After staring the selecting the weight for the network randomly the backpropagation is a necessary step which help to find the necessary correction. The backpropagation algorithm can be divided into 4 main steps[60].

- Feed-forward computation
- Back propagation to the output layer
- Back propagation to the hidden layer
- Weight updates

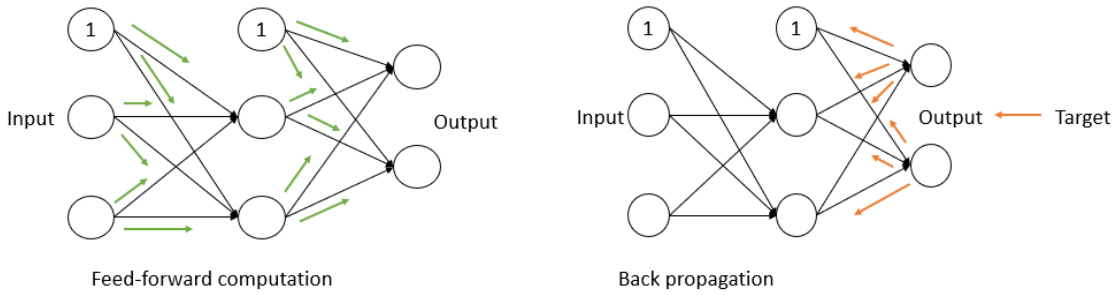

Figure 3.17: *Back-propagation*

The forward pass computes the network output and in the backward pass we simply back-propagate the error which means to find the cost by comparing the calculated output and the known ground-truth output[21]. Using this cost function to update the weights and biases to decrease the loss function.

3.4.7 Vanishing Gradient

As the number of layer increase, the problem of vanishing gradient occurs. At every time during the training of neural nets, the cost functions are calculated which is nothing but the difference between predicted output by the network to the actual output. The process of training is two-way process which means forwards pass and backpass which generally known as backpropagation. During the backpropagation weights are adjusted so as achieve the lower values for the cost function. During training an important properties accounts which are known as gradient which measures at what rate the cost function changes with respect to changes in weight and bias value. During the process of backpropagation the gradient starts decaying since the gradient values lie between o and 1 the backpropagation leads to vanishing gradient problem.To avoid vanishing gradient many techniques are used to optimize the training like using the small learning rate with gradient descent.

3.4.8 Overfitting and Regularization

Overfitting can be defined as a modelling error or a phenomenon where the neural network memories the input data rather than learning and understanding the data. Due to overfitting, it memorizes the data and works fine on that dataset, when a new dataset is given it doesn't show promising results what are expected. This leads to overfitting to that data and the network performs well on the training data but poorly on the unseen test data [61]. Regularization is can be defined as alteration in the learning algorithm. This alteration in the Regularization term help in reducing the general error[62]. In general L_1 and L_2 regularization and dropout are commonly used. In L_1 regularization a penalty $(\lambda|\omega|)$ is added to the cost function which leads vector to become sparse. L_2 regularization is commonly known as weight decay. L_2 regularization penalizes the squared weights $1/2\lambda|\omega|^2$ in the cost function and does not make the feature vectors sparse rather only reduces their contribution by reducing their magnitude.

3.4.9 Hyperparameters

Hyperparameters are the manual parameters which are defined manually by the programmer before it can start the training process. These parameters are not learned during the training. The programmer has to test and check which hyperparameters suit best for training. Hyperparameters are the epoch, the number of nodes in a layer, learning rate which is used for determining how much the weights must be adjusted with respect to the loss gradient and regularization penalty. These parameters play an important role in defining the speed and quality of the learning process so its important to select the right hyperparameters to get better training. For more understanding of the hyperparameters please refer to chapter 7 section 7.6 this section have better discussion over the hyperparameters.

3.5 Math under Convolutional Neural Network

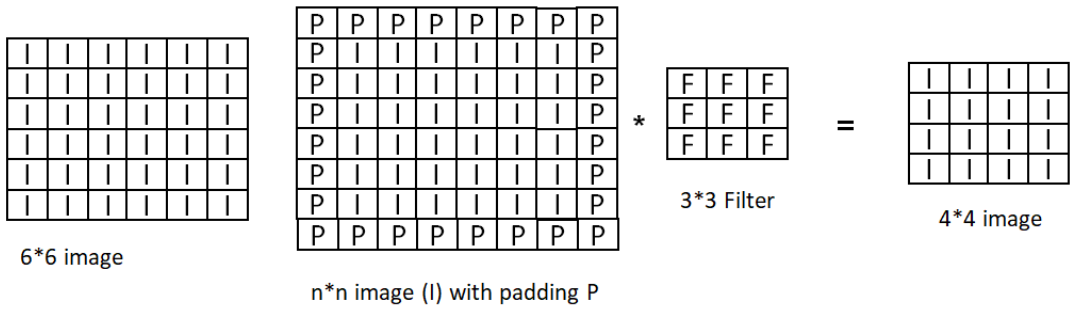

Figure 3.18: *Fully connected layer [21]*

2D Convolution

To understand the math under the convolution layer we have understood the math of the 2D convolution. In the figure 3.18 we have an image of the 6*6 size similar to it, we represent any image by n*n size. When we multiply n*n with each other we get the total number of pixel which is nothing but the resolution of an image. Some time due to some information loss when the filter slides over the image for convolution some padding is added. In general case, some sort of zero padding's is added to preserve the features. In mathematics term we represent any convolution process, see equation 3.4, 3.5 and 3.6. With no Padding the convolution can be represented as:

$$[n * n] * [f * f] = [n - f + 1] * [n - f + 1] \tag{3.4}$$

With Padding the convolution can be represented as:

$$[n * n] * [f * f] = [n + 2p - f + 1] * [n + 2p - f + 1] \tag{3.5}$$

With Padding and stride the convolution can be represented as:

$$[n * n] * [f * f] = [\frac{n + 2p - f}{s} + 1] * [\frac{n + 2p - f}{s} + 1] \tag{3.6}$$

$n * n$: represents the image size(height and width).
$f * f$: represents the filter or kernel size.
s : represents the stride
p : represents the padding

3D Convolution

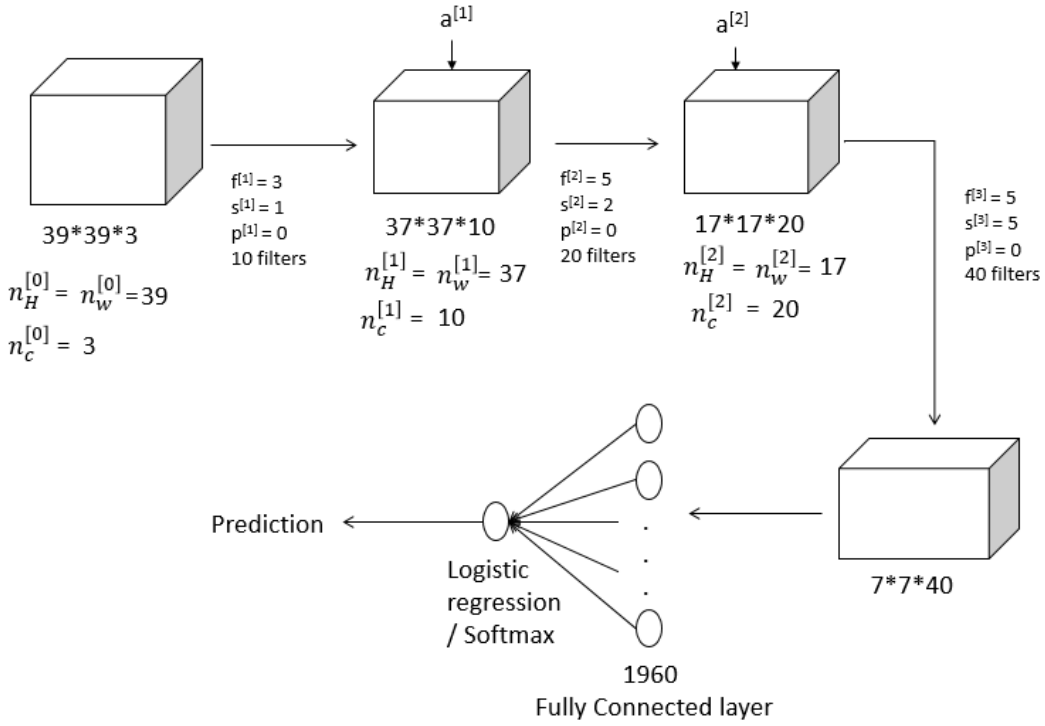

Figure 3.19: *Fully connected layer [21]*

After the understanding the 2D convolution we can move to understand the working of the 3D convolution. In 3D convolution we have the more channel in the image which is represented by the (height, width and channel). The size of the filter also increases. In the figure 3.19 we have an image of size (39,39,3) which means the height and width of 39 and the number of channels is 3. Suppose we have kernel size of 3 with zero padding stride of 1. The convolution produces an image of size (37,37,10) after extracting some building feature for convolution. In the second step of convolution, produces an image of size (17*17*20) with the kernel size of 5 and stride 2. After all the process we have final convolution output of (7,7,40) which in total produces 1960 outcomes see figure 3.19. The outcome having the highest probability is stated as the prediction.

$f^{[i]}$: filter size
$p^{[i]}$: padding size
$s^{[i]}$: filter size
$n_c^{[i]}$: number of filter
$f_h^{[i]} * f_w^{[i]} * n_c^{[i-1]}$: Each filter

$a^{[i]}$: represents the activation function.

Input to the convolution is a:

$$n_h^{[i-1]} * n_w^{[i-1]} * n_c^{[i-1]} \tag{3.7}$$

Output to the convolution is a:

$$n_h^{[i]} * n_w^{[i]} * n_c^{[i]} \tag{3.8}$$

$$n_h^{[i]} = \frac{n_h^{[i-1]} + 2p_h^{[i]} - f^{[i]} + 1}{s^{[i]}} \tag{3.9}$$

$$n_w^{[i]} = \frac{n_w^{[i-1]} + 2p_w^{[i]} - f^{[i]} + 1}{s^{[i]}} \tag{3.10}$$

3.6 Deep-Nets

Deep nets are great when solving the computer vision and image-related task. Since 1989 when the earliest neural network was developed many till now many developed has been made like optimization[22], regularisation[22] and moreover, the layer has better structure. In the last decade, much new architecture has been developed which focuses over the depth and spatial exploitation. According to the research done [22], the CNN depending on the structure can be categorized into seven categories spatial exploitation[22], depth[22], multi-path[22], width[22], feature map exploitation, channel boosting[22], and attention-based[22] CNN described in the figure below 3.20.

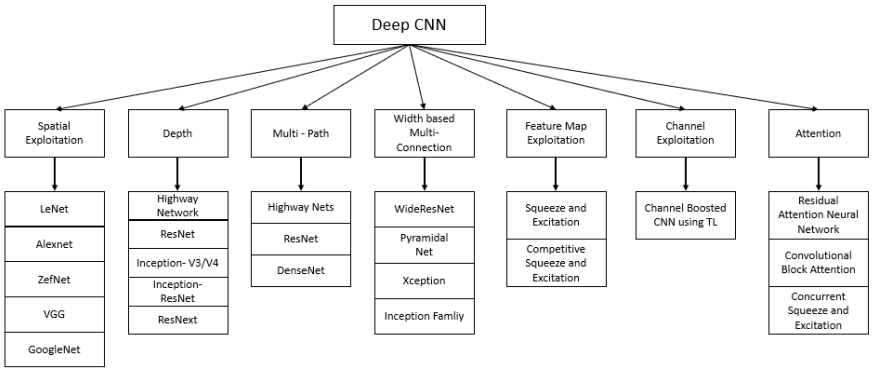

Figure 3.20: *Various Deep CNN architectures[22]*

3.6.1 VGG

VGG nets are typically large scale deep nets which are helpful in an image recognition task. The network was proposed at University of Oxford by K. Simonyan[24] and A. Zisserman[24]. In comparisons to the previous deep nets like AlexNets and ZefNet which had only 9 layers the VGG nets have 19 layers. The VGG nets are much denser. In an ImageNet task which computes over 14 million images which have over 1000 classes

of images the VGG nets received an accuracy of 92.7% [23][24] when they performed over 5 different tasks.

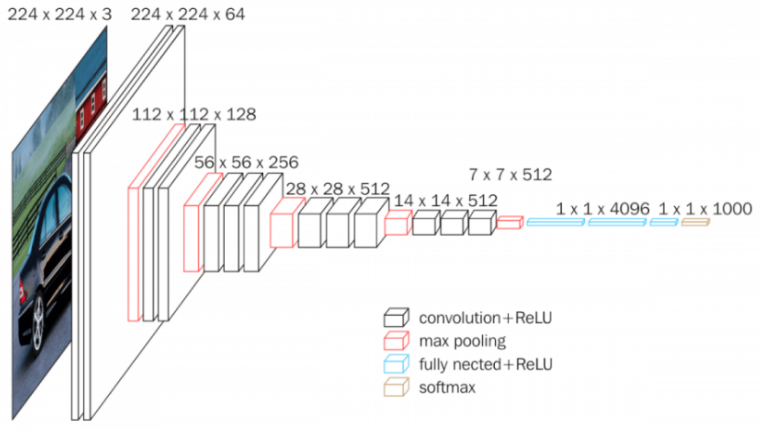

Figure 3.21: *VGG16 convolutional neural network[23]*

An RGB image of fixed size 224×224 is given as input to the Vgg nets. Vgg nets have a stack of layer which are known as convolution layers. The image is passed through these different convolutional layers. The filter size is 3 to 3, helping to capture each picture segment from left to right, top to bottom and middle. The figures 3.21 shows above an idea how the layers are stacked in Vgg nets.

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**
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**
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**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 3.22: *ConvNets Configuration [24]*

In the research done by Lei hu[25] proposed an encoder-decoder architecture in which the uses skip connection for depth estimation see figure 3.23. The encoder part is based on VGG network normal they are initialized with pre-trained weights and the second part was Upconvolution, this part majorly consists of global transfer layer[25] and up-sampling architecture[25] which help to bring a high-resolution depth maps from a piece of global information. The skips connections are represented by T-nets[25] which help in getting the information from one block to another distribution for monocular depth. the T-Nets is key aspect in this architecture which help to bring the information from earlier layer to end layer for better reconstruction of depth information. The last part of the architecture has fully connected layers for embedding the focal length information[25].

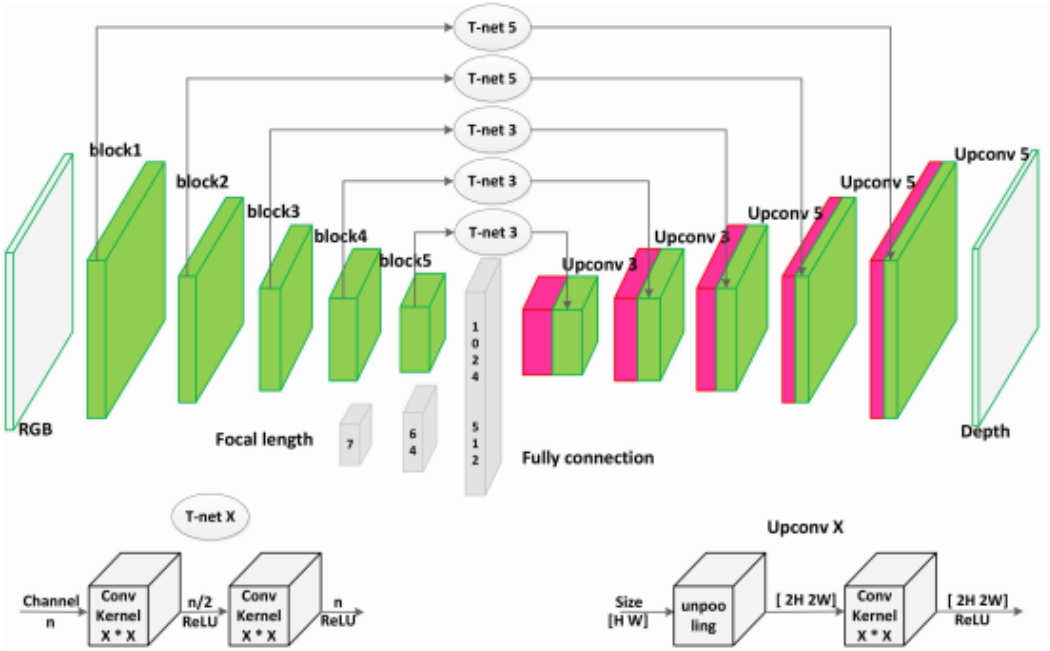

Figure 3.23: *Vgg in a encoder decoder Architecture proposed by et al Lei Hu[25] for Depth Estimation using Single Image*

Another research did by the et al Jack Zhu[26] at the University of Standford proposed a method for real-time depth estimation from 2D images[26]. In the depth estimation, the researcher uses the VGG architecture with the pre-trained weights which were originally trained on Imagenet[63] on the object detection task and a fully connected layer which helps to find the depth estimation from the given 2d input image. The green layer in figure 3.24 is the area where the backpropagation is performed and the red part is the frozen layer where the weight is frozen. For activation at each layer they used the ReLu function instead of the sigmoid function. In final section of the architecture they have the single up scaling layer with convolution layer[26] instead of Fully-Connected layers as the final layer predict the correct depth information for the given input image.

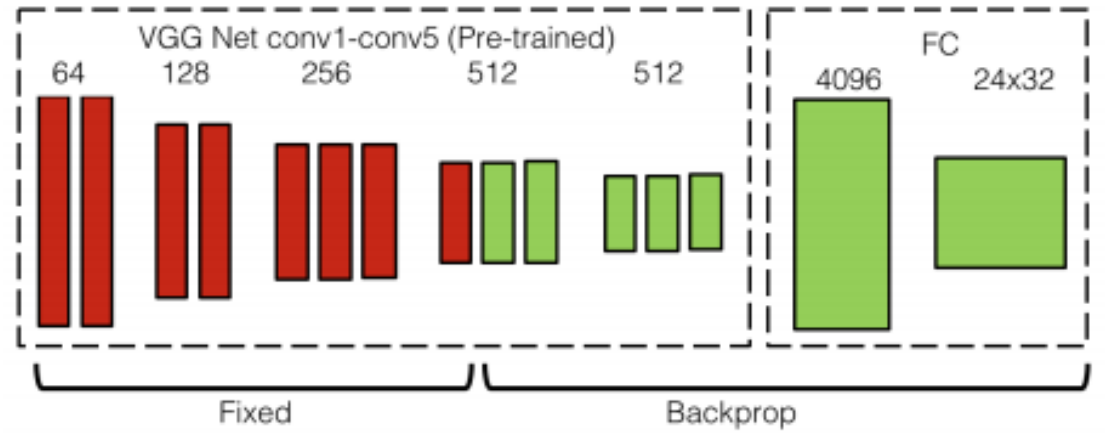

Figure 3.24: *VGG architecture proposed by Jack Zhu for Real-Time Depth Estimation from 2D Images[26]*

3.6.2 ResNet

ResNet is defined as Residual Networks and till now they are the state of the art and are used in many challenging computer vision tasks. In a residual neural network have short cut or skip connection which enables a better flow of communication between the layers which help in the optimization[28].

The Residual network are those network which have skip connection contains Relu. The main motivation to use the skip connection is to avoid the major problem of vanishing gradient, by using the activation's form the last layers until the new layer learns its weight.

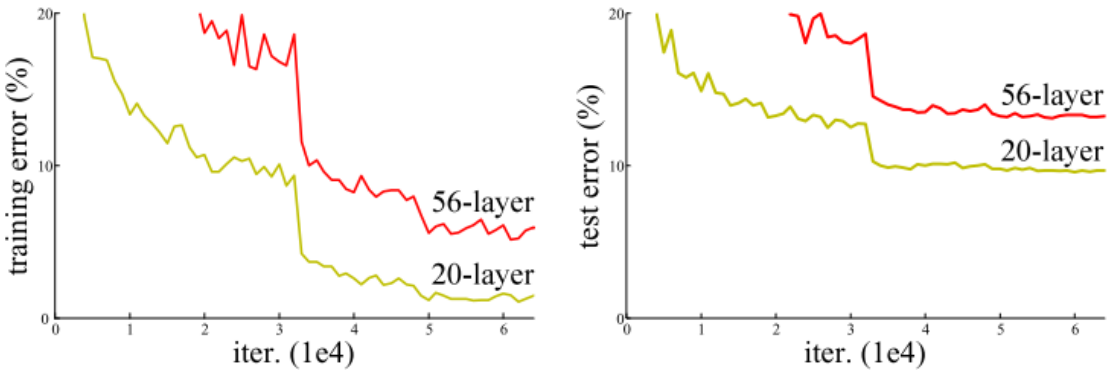

Figure 3.25: *Plain Net Training and Test error loss [27]*

Talking about the Plain neural network as the number of layer increase training and test error also increase. As we can see from the above figure 3.25 shows degradation of training accuracy on left and test accuracy on right when conducted on popular data-set of CIFAR–10 for image classification task. The data-set of CIFAR–10 contain the collection

images which are used of training and testing the complex computer vision and image processing algorithm.

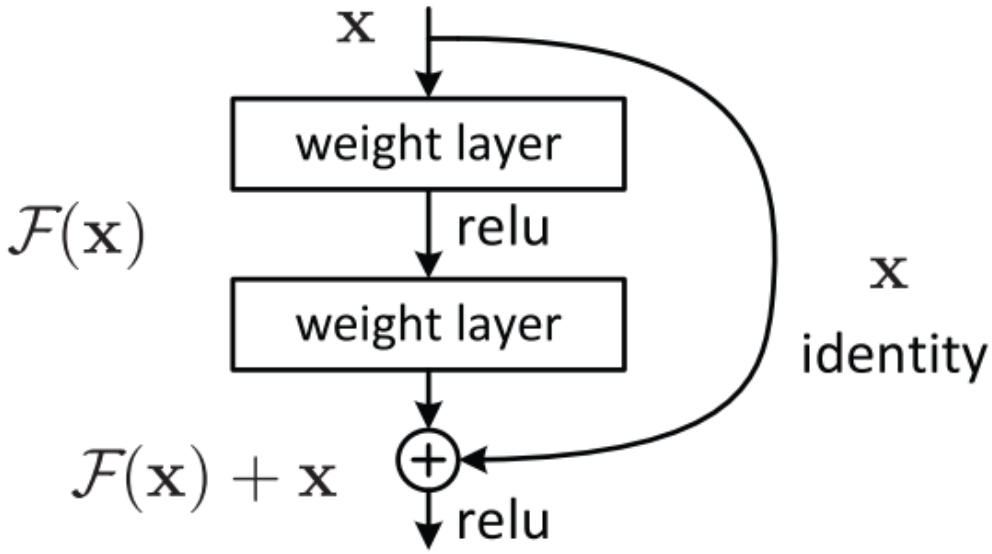

Figure 3.26: *Residual learning: a building block[27].*

To define a building block we see the figure 3.26 and the residual block can be derived from the equation.

$$y = F(x, \{W_i\}) + W_i x \tag{3.11}$$

In the above equation, x and y are the input and output vector of the layer. The function $F(x, \{W_i\})$ defines the residual mapping which has to be learned. When the number of layers is 2 the residual mapping function which needs to be learned is defined by $F = W_2\sigma(W_1 x)$ in which σ denotes the Relu function.

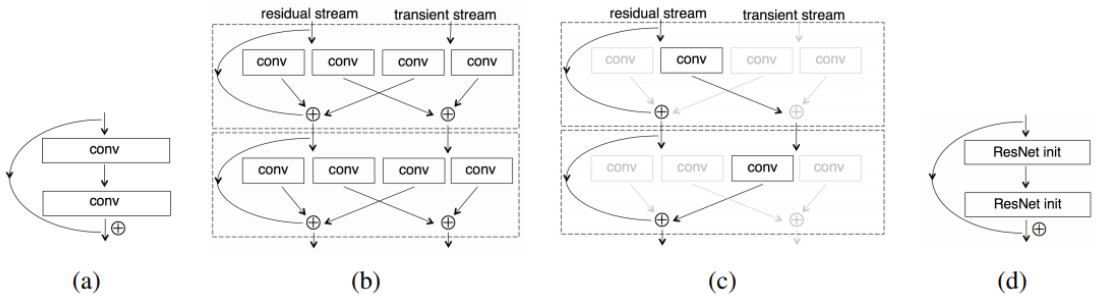

Figure 3.27: *Different type of Skip Connection in ResNet Block[28].*

In the figure 3.27 above (a) represent a Convolutional ResNet block with skip connections. Figure (b) is a combination of residual network and standard network which are

working in parallel this type of residual network is called ResNet Init. In the above Figure (c) we can see that some connection is grayed out which means they tends to become zero makes it network of 2-layers ResNet block, therefore makes the efficient working between the ResNet and standard ConvNet. This type of network is represented by name RiR (ResNet in ResNet). Figure(d) is basically the stack of 2 layers RiR block.

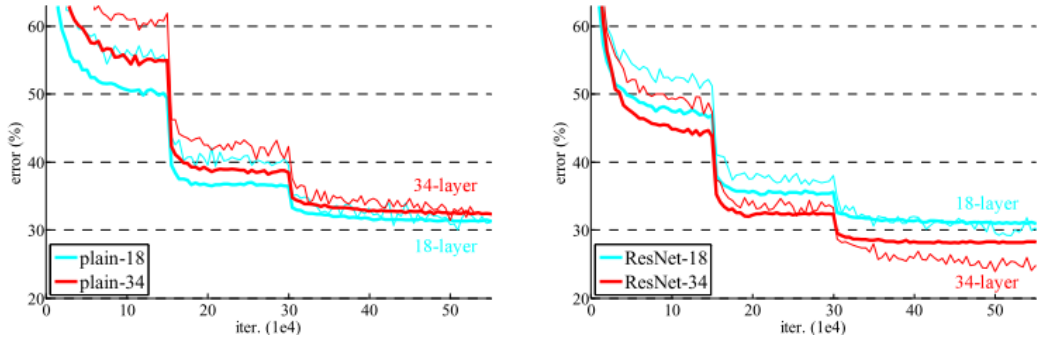

Figure 3.28: *Comparison of Plain Net Vs Residual Net [27].*

In comparison to the different architecture, the ResNet performed better in comparison to the other architecture. In figure 3.29 the resent producing a really good accuracy when performed over the ImageNet for object classification.

Convolutional Neural Networks Architectures			
Architecture	Top-1 Accuracy	Top-5 Accuracy	Year
Alexnet	57.1	80.2	2012
Inception-V1	69.8	89.3	2013
VGG	70.5	91.2	2013
Resnet-50	75.2	93	2015
Inception-V3	78.8	94.4	2016

Figure 3.29: *Different architecture with their accuracy[29]*

The ResNet-50 Architecture is powerfully deep nets which are used for many tasks like object detection and depth estimation. Figure 3.29 showing the ResNet-50 outperformed various ImageNet task for object detection. In ResNet-50 50 defines the number of layers. The ResNet is being the state of the art. The main advantages of the ResNet are that they have skip connections which help to get the information from the starting layer to last layers. This leads to better-amplified information at the end of the full fully connected layer.

ResNet-50 Architecture for Depth Estimation

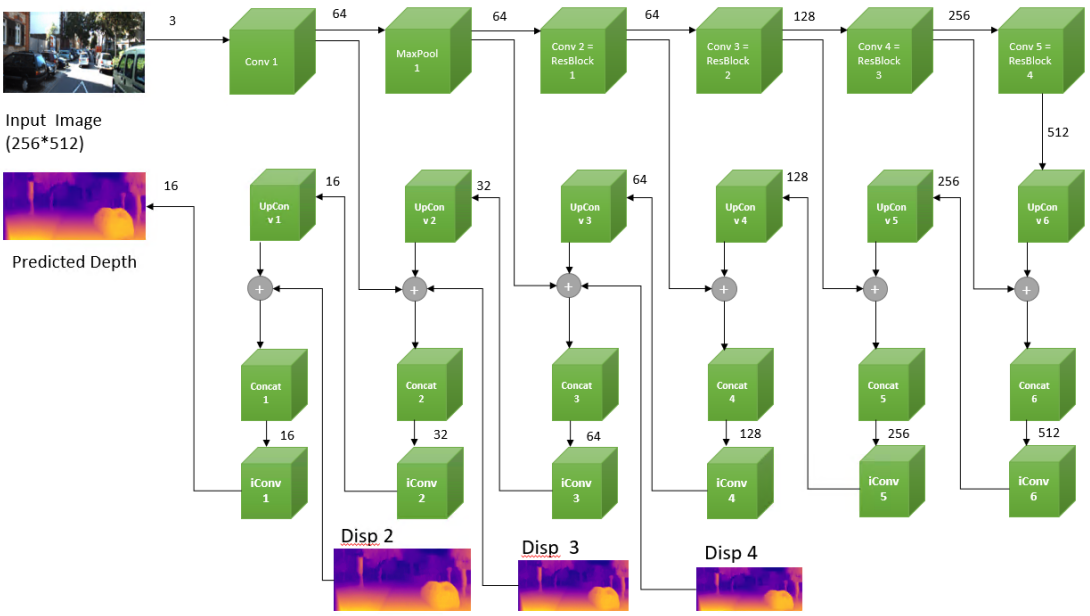

Figure 3.30: *ResNet-50 Architecture[5] [7].*

The ResNet architecture brings skip connection which helps to overpass two or more convolutional blocks [5]. The ResNet for depth estimation is being used by many researcher [5] [7] [10]. In figure 3.30 we have an input image of size(Height, width, Channel) $256 * 512 * 3$ given to the first encoder part which contains the first convolutional layer defined in figure 3.30. The input of the convolution layer(Conv1) has an output size of 64. The output uses the skip connection which takes the output from the conv1 to the decoder part(UpConv2). The skips connection are showed in figure 3.30. As we move forward we follow the same principle of using the skips connection and the output of the one convolution block or max pool layer is given to the next convolution block. At the architecture is based on [5] which have two parts one is encoder and other is a decoder which uses skip connection in between. The output of the networks is a 16-bit depth image which has depth information about every pixel. For a better understanding of the architecture refers to figure 3.31 and 3.32 which will give you direction to the various connection between the layer and their outputs.

Figure 3.31 gives knowledge about the encoder part of the architecture. The figure gives the knowledge about each layer which layer is connected to which layer. The "k" in figure 3.31 and 3.32 represents the kernel size. The "S" represents the stride. The "channels" describe the exact number of input and output channels for each layer. The "in" and "output" are defined as the scaling factor which is being used to upscale or down the given input image. The "input" is defined as the input which is given to each layer. The final main operation $+$. The $+$ defines the interconnection between the layer or concatenation shown in figure 3.30.

Encoder						
Layer	K	S	Channels	In	Out	Input
Conv 1	7	2	(3/64)	1	2	Image
MaxPool	3	2	(64/64)	2	4	Conv 1
Conv 2	3	3	(64/64)	4	8	MaxPool
Conv 3	3	4	(64/128)	8	16	Conv 2
Conv 4	3	6	(128/256)	16	32	Conv 3
Conv 5	3	3	(256/512)	32	64	Conv 4

Figure 3.31: *ResNet-50 Architecture[5] [7].*

Decoder					
layer	K	S	Channels	In	Out
Upconv 6	3	2	(512/512)	64	32
iconv 6	3	1	(1024/512)	32	32
Upconv 5	3	2	(512/256)	32	16
iconv 5	3	1	(512/256)	16	16
Upconv 4	3	2	(256/128)	16	8
iconv 4	3	1	(128/128)	8	8
Disp 4	3	1	(128/2)	8	8
Upconv 3	3	2	(128/64)	8	4
iconv 3	3	1	(130/64)	4	4
Disp 3	3	1	(64/2)	4	4
Upconv 2	3	2	(64/32)	4	2
iconv 2	3	1	(66/32)	2	2
Disp 2	3	1	(32/2)	2	2
Upconv 1	3	2	(32/16)	2	1
iconv 1	3	1	(18/16)	1	1
Disp 1	3	1	(16/2)	1	1

Figure 3.32: *ResNet-50 Architecture[5] [7].*

4 Depth Estimation Technique

In this, we will talk about different types of approaches in-depth maps. How the depth from these approaches have advantages and disadvantages. At first, we will see how the depth can be generated from the disparity method. We will also understand what inverse depths are and how the sparse depth can help to generate dense depth.

4.1 Depth-Disparity relation

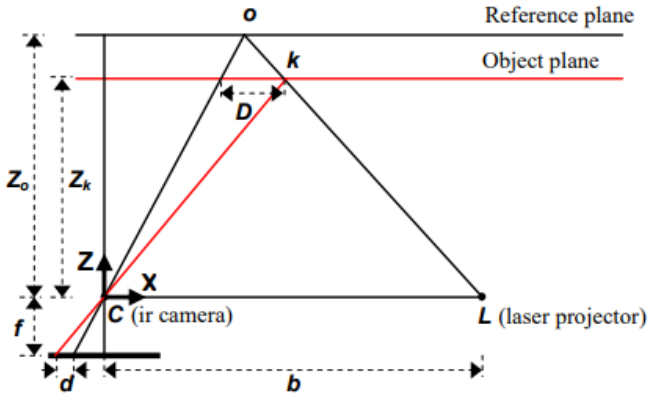

Figure 4.1: *Schematic representation of depth-disparity relation[30]*

The figure 4.1 shows a connection between the calculated disparity and the distance of object point at k to the sensor[30]. As we can find from the figure we have two similar triangle and using the similarity property of triangle we get two equation 4.1 and 4.2.

$$\frac{D}{b} = \frac{Z_o - Z_k}{Z_o}[30] \tag{4.1}$$

$$\frac{d}{f} = \frac{d}{Z_K}[30] \tag{4.2}$$

Z_o : Defines that the object is on a reference plane at a distance[30]
Z_k : Distance (depth) of the point k in object space.
d : Disparity in the image space.
D : Displacement of the point k in object space
b : Base length
f : Focal length

From the equation 4.2 we are able find the disparity see equation 4.3. We just need focal length and the known depth with displacement distance to the object space.

$$d = \frac{D}{Z_k} * f[30] \tag{4.3}$$

To find the depth of the using the concept of disparity Substituting D from the equation 4.2 into equation 4.1.

$$Z_k = \frac{Z_o}{1 + \frac{Z_o}{f*b}d}[30] \tag{4.4}$$

4.2 Inverse Depth

A monocular camera is a type of projective sensor [31] which captures the important features of the surrounding. The information of the depth is captured by the camera through repeatedly observing the scene, as it captures the depth information each time a ray of light is captured which passes through the optic centre. The angle between the optic centre of the camera to the feature parallax (when view an object from two different viewpoint the displacement between the two view angle is known as parallax) provides the information of the depth. Some information is too far, and this information is known as point at infinity can be represented as homogeneous coordinate.These homogenous coordinates provide a good motion estimation from image sequences [31]. In a camera motion the point at infinity are hard to capture. There is multiple point which are distributed far way from each other. To parametrize these points so that each point at infinity which carries the information of depth can be captured[31]. With the help of the parametrization (The parametrization of inverse depth means, in general, to find a smooth correlation between one frame to another frame when the camera is moving. This help in finding the point which is in the previous frame far way or infinity using theses point to build a better correlation to the new frame.) of the inverse depth and using the Gaussian distribution which allows the cover the points which are nearby and the points which are at infinity and help to estimate the finite depth of feature which were at infinite point for a long period of time see figure 4.2.

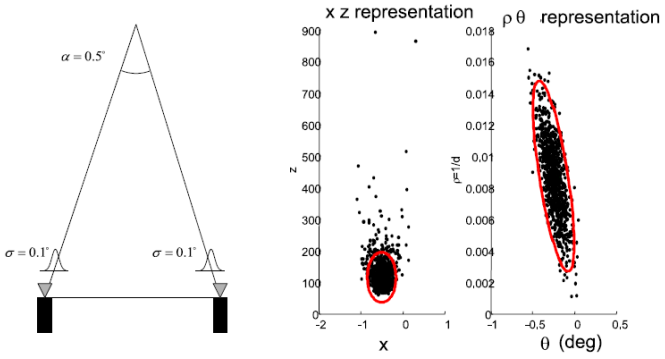

Figure 4.2: *point reconstruction from two low parallax observations[31]*

As seen in the figure 4.2 the most left side shows us when we have one viewing angle

at left and one viewing angle at right which relates to two observation point. In the Cartesian coordinates(in the middle of figure 4.2) the depth information shown by the dots are scattered which means some the information is not captured. When we use Gaussian to encode the error(ρ, θ)bring better values for depths which means the depth information which is at infinity is also being captured.

As in classic approach inverse depth has been very useful in estimating the relation between the image disparity to the point depth in stereovision [31], motion filed inducted by scene points with camera velocity [31] and structure motion error analysis [31].

The figure 4.3 shows the state vector of the camera in a free space which is expressed by (x_v). The following parameters which are related to calculating the feature parametrization are defined below.

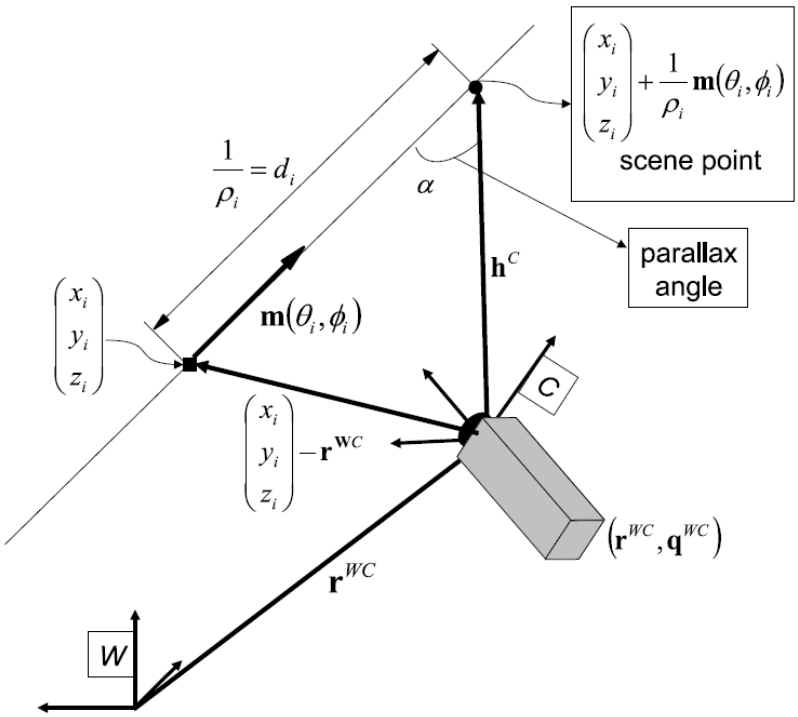

Figure 4.3: *Feature parametrization [32]*

x_v : Camera state
r^{WC} : Camera optical centre position
q^{WC} : Quaternion defining orientation
v^W : Angular velocity relative to world frame W
ω^c : Angular velocity relative to camera frame C
a^W : Linear acceleration
α^C Angular acceleration
V^W : Linear velocity
$V^W = a^W \Delta t$
Δt : Camera movement at each step

Ω^C : Angular velocity
$\Omega^C = \alpha^C \Delta t$

Camera motion can be defined as initial state which in our case is represented by x_v and the next step case which in this case is represented f_v

$$x_v = \begin{Bmatrix} r^{WC} \\ q^{WC} \\ v^W \\ \omega^c \end{Bmatrix} \quad f_v = \begin{Bmatrix} r^{WC}_{k+1} \\ q^{WC}_{k+1} \\ v^W_{k+1} \\ \omega^c_{k+1} \end{Bmatrix} \tag{4.5}$$

$$f_v = \begin{Bmatrix} r^{WC}_{k+1} \\ q^{WC}_{k+1} \\ v^W_{k+1} \\ \omega^c_{k+1} \end{Bmatrix} = \begin{Bmatrix} r^{WC}_k + (v^W_k + V^W_k)\Delta t \\ q^{WC}_k \times q((\omega^W_k + \Omega^W)\Delta t) \\ v^W_k + V^W \\ \omega^W_k + \Omega^W \end{Bmatrix} \tag{4.6}$$

Euclidean (X,Y,Z) point

$$x_i = (X_i, Y_i, Z_i)^T \tag{4.7}$$

Inverse depth point parameterization The 3D point I can be defined by 6 DOF or 6-degree state of vector

$$y_i = (x_i, y_i, Z_i, \theta_i, \phi_i, \rho_i)^T \tag{4.8}$$

$$x_v = \begin{Bmatrix} X_i \\ Y_i \\ Z_i \end{Bmatrix} = \begin{Bmatrix} x_i \\ y_i \\ z_i \end{Bmatrix} + \frac{1}{\rho_i} m(\theta_i, \phi_i) \tag{4.9}$$

$$m = (\cos\phi_i \sin\theta_i, -\sin\phi_i, \cos\phi_i \cos\theta_i) \tag{4.10}$$

y_i : initial camera state
θ_i : azimuth angle
ϕ_i : elevation angle
d_i : point depth along the ray
ρ_i : inverse depth
$\rho_i = \frac{1}{d_i}$
m : unidirectional vector

To understand the inverse depth we have understood the concept of points which are at infinity. As from the equation 4.11, we can say that depth is inversely proportional to the distance. Suppose we have any object which is at infinity see

$$Disparity \propto \frac{1}{Depth} \tag{4.11}$$

$$Disparity = \frac{f.b}{Depth} \tag{4.12}$$

To visualize the depth at infinite we put the value depth as ∞ we obtain $\frac{1}{\infty}$ which is nothing but zero information of depth. Inverse depth points help to find information which are infinity, for example inverse help to find parallel lines which are infinity which general case the normal depth information would not capture the depth information.

4.3 Sparse Depth

Sparse is the depth which doesn't have complete information about the depth of the environment. They provide only partial information regarding the scene geometry. The figure 4.4 shows the sparse depth which does not provide complete information about depth to every pixel in an image. However sparse depth bring helpful information to construct a dense depth. Now we know what does sparse depth means. The incomplete information of depth is very much useful when introduced with the deep learning approaches to find the dense depths. The sparse depth, when provided to the model at a time of training, brings a more robust knowledge of depth. To construct a 3d scene without having any knowledge about the environment the sparse is the very crucial part which helps tp represents the depth maps.

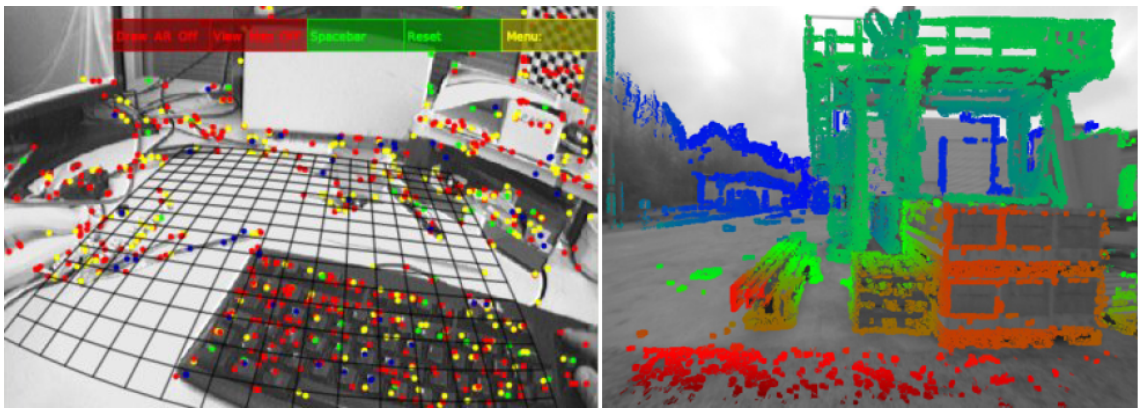

Figure 4.4: *Sparse Depth information [33]*

Figure 4.4 shows the sparse which are shown by the dotted point. As the image comprises of (height, width, channels). When multiplying the height with width we get the number of pixels (x). In the figure 4.4 for x number of the pixel we very less amount of depth pixels which means the depth information not complete which therefore conclude to sparse depth images.

4.4 Dense Depth

Dense depth can be described as depth image which has complete information about the depth in the image. As the image comprises of (height, width, channels). When multiplying the height with width we get the number of pixels (x). In the figure 4.5 for x number of the pixel, we have x number of depth pixels which means the depth information is complete which therefore conclude to a dense depth image. In the recent development of the deep learning sector involving the computer vision approaches. The dense depth has become a very crucial part in the sector automobiles and robotics.

Fangchang Ma et al[33]uses a self-supervised learning approach to train the model using the sparse depth maps and an image to predict the dense depth maps. In figure 4.5 we can see that the a single RGB image with the sparse depth provide a dense depth information.

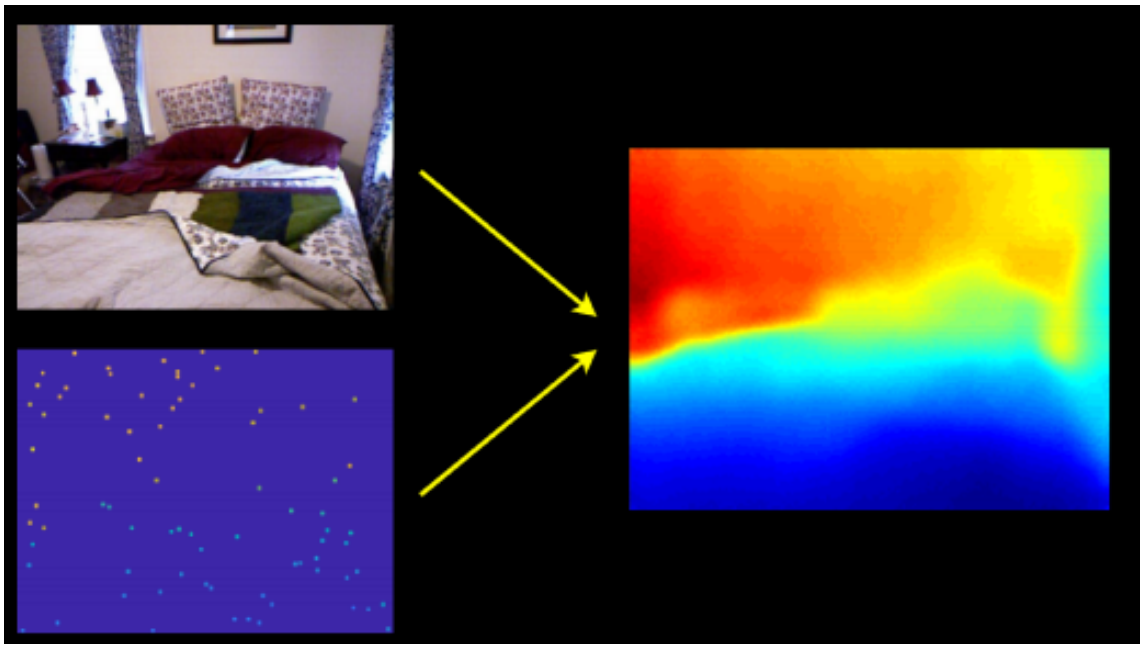

Figure 4.5: *RGB image and Sparse Depth image build a Dense Depth image[33]*

In research done by the Yanchao Yang try to generate the dense depth maps by exploiting the sparse range mesaurments[34] like lidar. The lidar measurement provides a depth value of a certain amount of the pixel in that scene. They proposed a Depth Completion Network (DCN)[34] which uses the Sparse depth and an image to find each value of the dense depth for every pixel in an image. They proposed an encoder-decoder architecture. In figure 4.6 (A) represents the conditional prior network that predicts depth maps using the single image. In (B) the approach [34]uses the sparse depth maps and normal RGB image for predicting the dense depth maps.

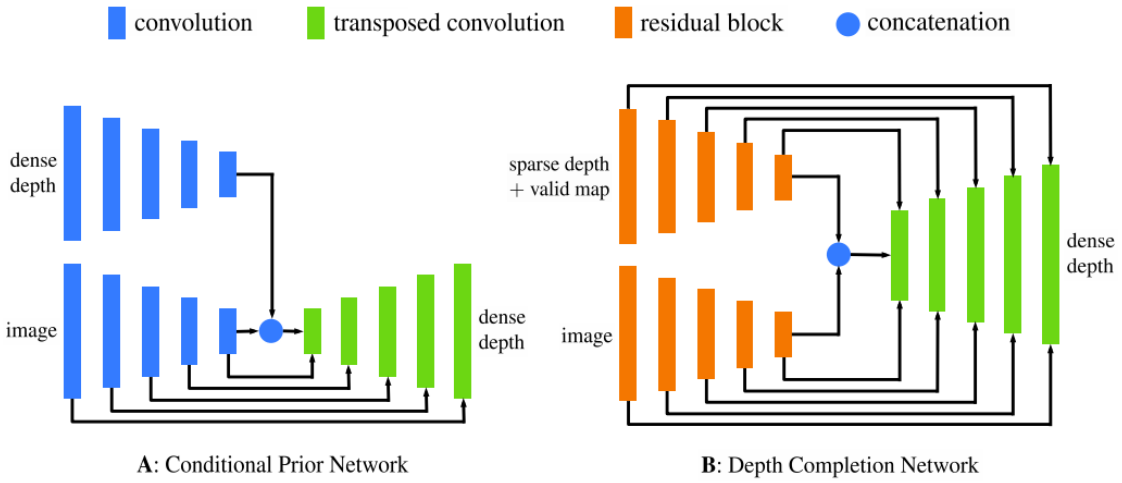

Figure 4.6: *Depth Completion Network (DCN) Architecture [34]*

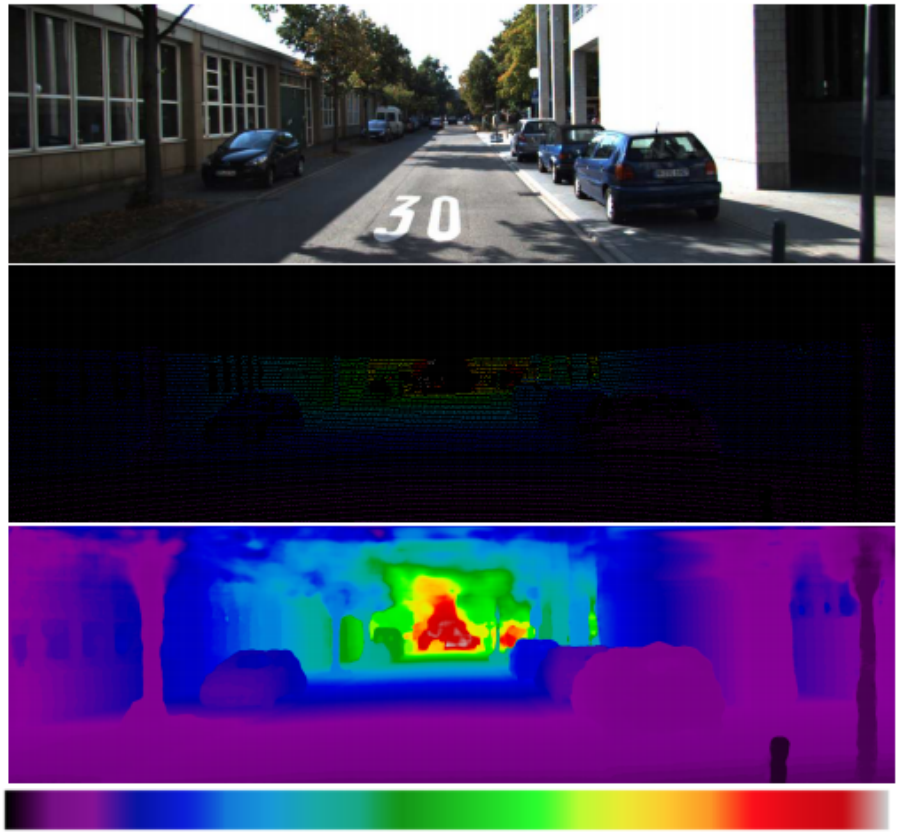

Figure 4.7: *Dense depth maps[34]*

In figure 4.7 shows us an RGB image at the top which defines the surroundings and the geometry of the scene. The middle parts show us the sparse depth in form of point cloud taken from lidar and which doesn't have complete information depth in the scene. The last one is the predicted dense depth maps which combine the single image and lidar point cloud at training. The colour bar shows the extreme left the zero which means close and the extreme right infinity.

4.5 Depth

The depth is a channel in an image which help to define the relationship of distance to the object in a scene from the viewpoint. The depth is of different bits. The most general way to store the depth information are 8-bit or 16-bit depth. Each bit of depth provides the information about the surroundings. In a colour image(.png format) the 8-bit depth describes the total 1,67,77,216 colour information. This information is combined through individual Red, Blue, Green colour. Each R, G, B colour has the value 256. We can see that the 16-bit colour depth gives the topmost colour resolution shown in the figure 4.8.

Color depth	Red		Green		Blue		Total colors
1 bit	2	x	2	x	2	=	8
2 bit	4	x	4	x	4	=	64
3 bit	8	x	8	x	8	=	512
4 bit	16	x	16	x	16	=	4.096
5 bit	32	x	32	x	32	=	32.768
6 bit	64	x	64	x	64	=	262.144
7 bit	128	x	128	x	128	=	2.097.152
8 bit	256	x	256	x	256	=	16.777.216
9 bit	512	x	512	x	512	=	134.217.728
10 bit	1.024	x	1.024	x	1.024	=	1.073.741.824
11 bit	2.048	x	2.048	x	2.048	=	8.589.934.592
12 bit	4.096	x	4.096	x	4.096	=	68.719.476.736
13 bit	8.192	x	8.192	x	8.192	=	549.755.813.888
14 bit	16.384	x	16.384	x	16.384	=	4.398.046.511.104
15 bit	32.768	x	32.768	x	32.768	=	35.184.372.088.832
16 bit	65.536	x	65.536	x	65.536	=	281.474.976.710.656

Figure 4.8: *The total color values of different bits of depth [35]*

In the figure 4.9, the example of the same image with different values of bits is shown. The 1-bit depth in the top left corner is the sparse depth the information about some pixel is lost. The 4-bit depth gives an approximation about the scene in the image. The 8-bit gives a better resolution of colour in the image. The final 16-bit depth gives, even more, better colour resolution each pixel can be differentiated from the other pixel. In 16-bit depth the colour is more precise the depth information is more easily available.

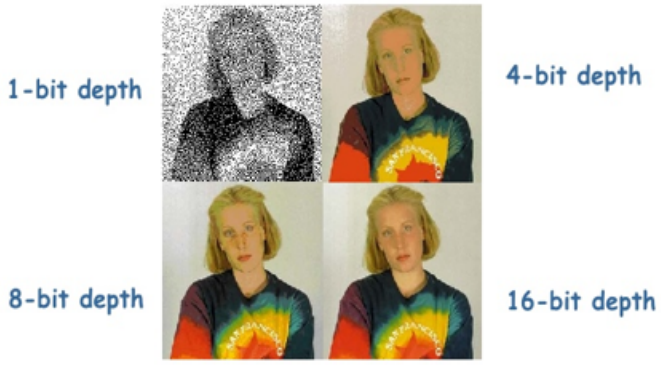

Figure 4.9: *How different bits of depth look like[35]*

5 Semi-Supervised Learning for Depth Estimation

Semi-supervised learning is a unique approach which combines both cues from supervised and unsupervised learning. In some aspect when the ground truth data is not easily available or in some sense when the data is very less the semi-supervised learning can bring a more reasonable approach for training and achieving the required results. From the previous chapters we would have now a small overview what the semi-supervised learning means but the big question arise how this approach can be used in prediction of depth estimation. Depth estimation using the deep learning approaches needs three main steps 1st the model or your algorithm that means how would approach the problem of depth estimation. 2nd is your training data which in our case is the depth annotated data which acts as supervised training as it provides the ground truth for training and raw RGB images which acts as the unsupervised training data. 3rd is the coding of how one programmer would embed 1st and 2nd step in programming.

5.1 Understanding Semi-Supervised Learning

Semi-Supervised learning is the technique in which the problem of the labelled data can be answered in a sense such that during the training the shortage of the labelled data can be overcome by combing the unlabeled data for training purpose. As in general to have a labelled data for training is a rare and expensive problem. The closest to the semi-supervised training can be related to the self-training[64], self-labelling[64] or decision directed learning[64]. This can be described as a wrapper function or algorithm that uses the supervised learning which uses only labelled data for training at staring. Initiating the training with supervised training and then retraining the model with its own prediction brings a more efficient way of getting the labelled. The only main problem with this approach is that the algorithm or wrapper uses a supervised method[64]. The first problem in 1970 that uses semi-supervised learning was an estimation of the fisher linear discriminant rule[65]. The model used labelled and unlabeled data which uses an expectation-maximization algorithm. The main question on how can semi-supervised learning can work with today's deep learning the framework. In mathematical term, the learned knowledge on $p(x)$ that one gains through the unlabeled data has to carry information and this information is used for inference $p(l_x|x)$. The semi-supervised is approaches which use both the cues from supervised and unsupervised learning.

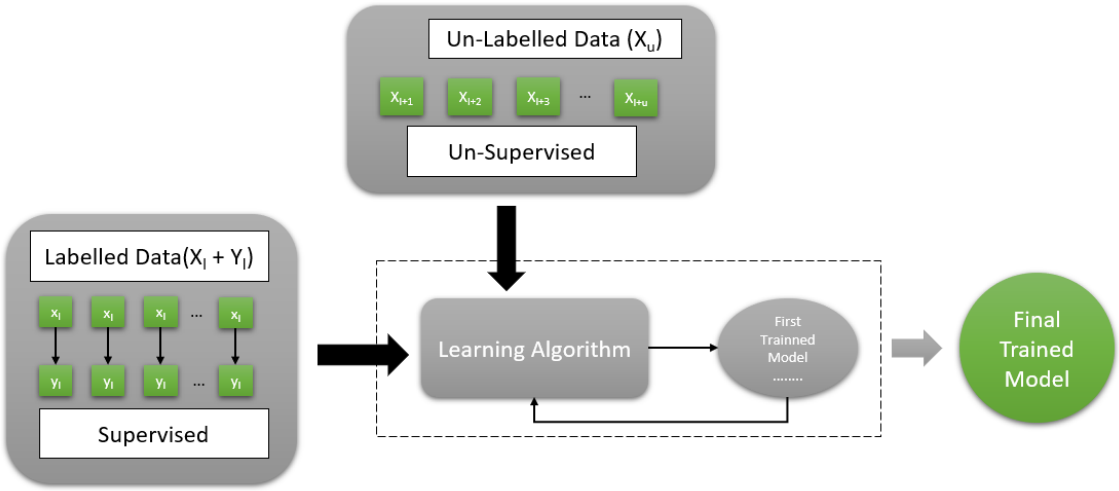

Figure 5.1: *Semi-Supervised Training in General*

Semi-supervised is the approach of combining the unlabelled data, labelled data with the learning algorithm under some kind of supervision information[64] see figure 5.1. To understand semi-supervised learning in a better way, we think of a data set $X = (x_i)_i \in [n]$ which comprises of two parts one part contains particular know as supervised parts which contains the points$X_l := (x_1, ..., x_l)$, for which the have corresponded labels $Y_l := (y_1, ..., y_l)$ are available and the second part which is known as unsupervised learning which contains the $X_u := (x_{l+1}, ..., x_{l+u})$, for which the labels are not available or the information is complete.

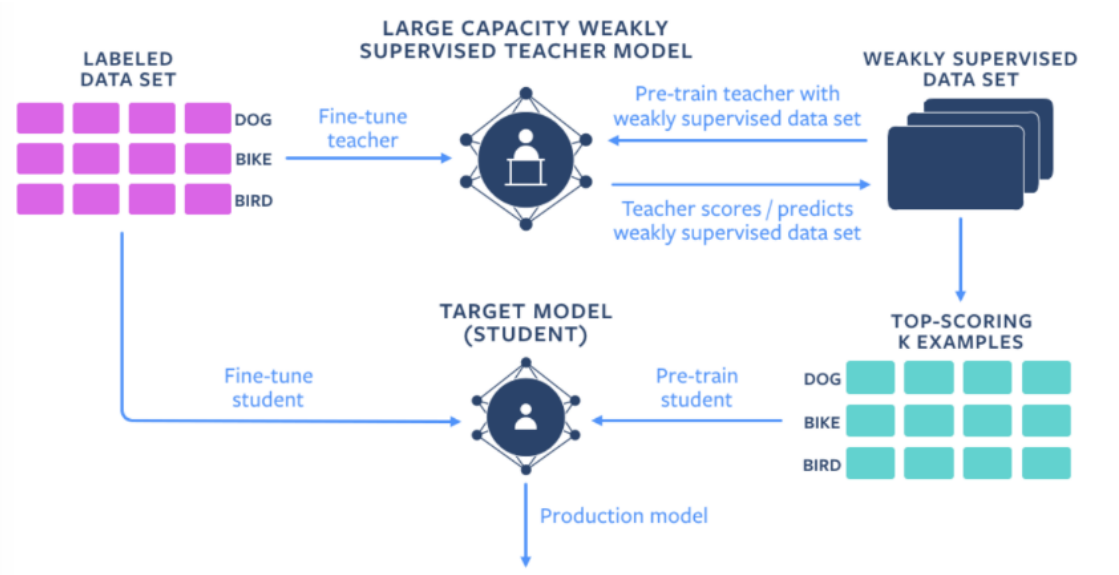

Figure 5.2: *Semi-Supervised Training framework used by the Facebook [36]*

In figure 5.2 we can see that at the starting of the training the model or the initial model(teacher)[36] is trained with the huge amount and accurately labelled data. The teacher model which they get after the training is used for getting the approximated scores for unlabelled data and to predict the labels. The scores are ranked according

to each class. The top-scoring result is used for pre-training the small and light model named student for classification. To get a fine tuned model the student model is trained with labelled data to get the target model. This gives a complete utilization of the labelled and unlabeled data for training at the pre-training stage. For evaluation of framework they use the image net benchmark for photo classification. The obtained accuracy was 81.2 per cent which top-1 accuracy with the semi-supervised framework for resent-50 benchmark model.

5.2 Semi-Supervised Training for Depth estimation

Figure 5.3 describes how the framework for semi-supervised learning works. In figure (a) represents a normal RGB image (b) represents a sparse depth image taken form a LIDAR sensor. The (c) represents the unsupervised training where only a stereo pair of images are being used at the time of training which leads to depth image where the horizontal depth information is not possible retrieve. The depth information using the unsupervised part is not accurate. The (d) is the supervised part where the only ground truth information is being used for training. The only problem is that the availability of ground truth data is a challenging task. The (e) part is where our approach comes into the light which uses both information of stereo images and the depth for finding the complete depth information from a single image. As seen from figure 5.3 the (e) has the most accurate depth in comparison to the (c) and (d).

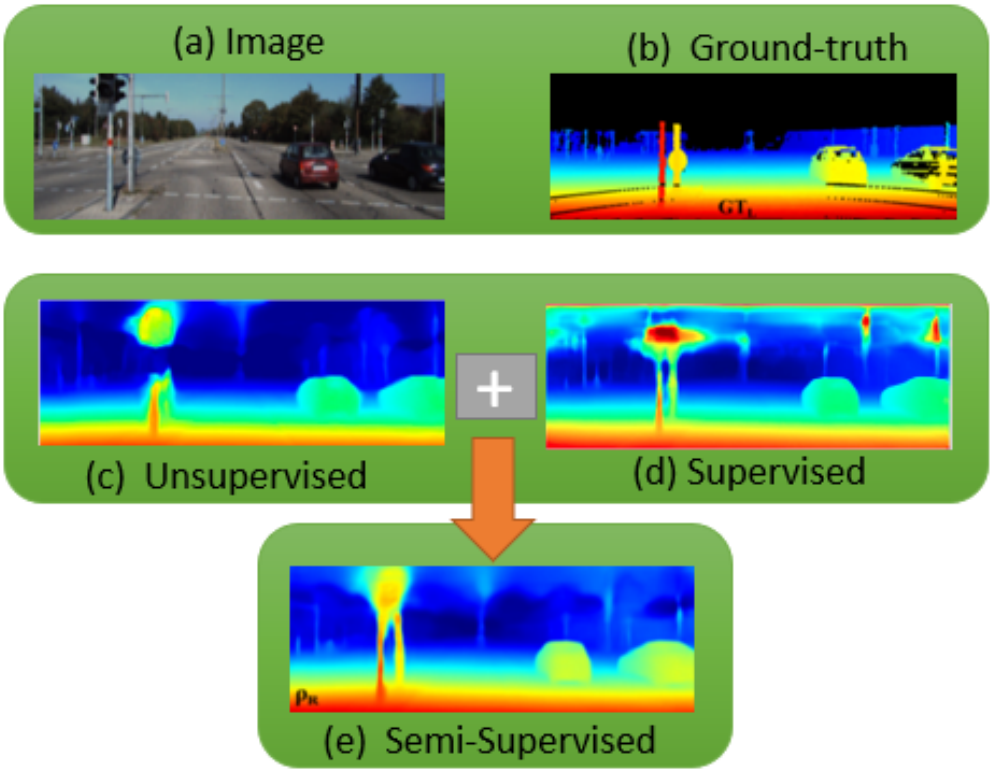

Figure 5.3: *Semi-Supervised Training framework for Depth Estimation Training*

As we have discussed in the above section 5.2 the semi-supervised learning is an approach that uses both the cues of supervised and unsupervised learning data. The depth estimation is a technique used for finding the information of the distance from a viewpoint which means each pixel in the image will have the information of depth or distance. In our model, we try to explore the problem of finding the depth information from a single using the semi-supervised learning approach which incorporates Left-Right consistency at the time of training. The training data is consisting of the raw RGB stereo images which act as an unsupervised part for training and for the supervised part we use the sparse depth information which is in the form of images. Through combing both the novel approaches brings the depth information which munch denser and accurate from a single image at the time of inference.

6 Dataset KITTI

This chapter focuses mainly on the KITTI dataset. In this section, we will see why we chose KITTI dataset. See how Diverse is the dataset in the sense of image collection. We will see how the KITTI dataset is used for training purposes. In this chapter, we will also discuss we have chosen the KITTI dataset and its important.

6.1 History Of KITTI Dataset

The KITTI dataset is which have a collection of image scenario different scenarios figure 6.1 like City, Residential, Road,Campus and Person[37]. The dataset includes images which are helpful for solving complex computer vision task. The images are categorized into stereo, depth, optical flow, visual odometry, 3D object detection and 3D tracking[37]. As in this master thesis we focus on depth estimation, so we have used the Raw depth images and depth annotated images for the training purpose. The raw data set has colour and greyscale stereo sequences of images which are stored in .png format.The depth annotated data have more than 93k training and 1k images for evaluation and 500 test images.The raw images which we obtain from the KITTI dataset conatins nearly about 42,382[7] rectified stereo pairs from 61 scenes with image resolution of $1242 * 375$.

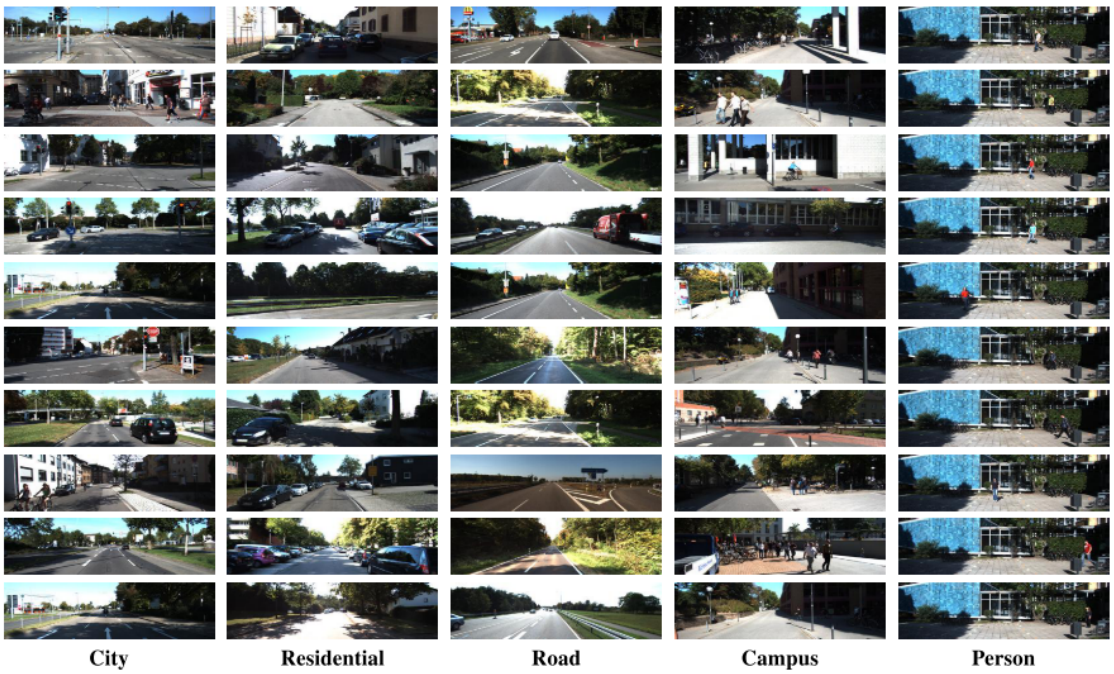

Figure 6.1: *Diversity of Kitti Dataset [37]*

Figure 6.2 shows the part of the images which we have used for training. The left side of figure 6.2 shows the stereo images which are in form left RGB image and right RGB image. The stereo pair of images are unlabeled part of the training which is unsupervised learning aspect. The middle part of figure 6.2 shows the depth images taken from the LIDAR which provide the label data fro training hence its called the supervised learning. The data set is a huge variety of images in a different scenario like sun, shadow various environment aspect so that model gets different images to learn from.

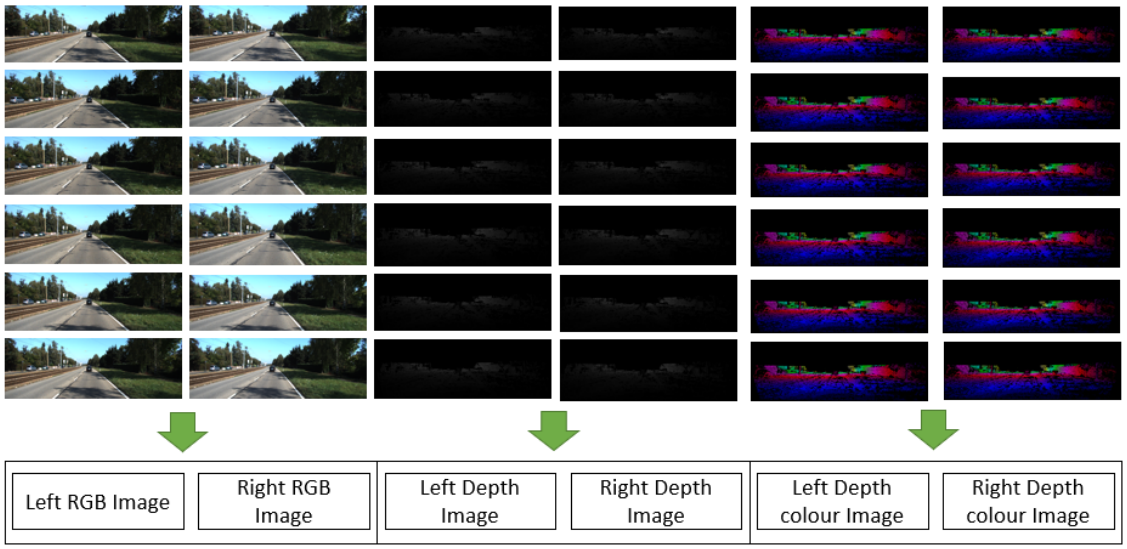

Figure 6.2: *The left side shows the RGB images(stereo pair), the middle is corresponding depth images for stereo pair and the right corner depth images in colour form[37]*

6.2 KITTI Dataset Download

To download KITTI dataset their are two ways. Make sure one have enough space around 175 GB and 200 GB to unzip the files in total 400 GB approx.

- To download the KITTI dataset go to the kitti website `http://www.cvlibs.net/datasets/kitti/raw_data.php`
- Download the raw dataset download script which of size 1Mb `http://www.cvlibs.net/download.php?file=raw_data_downloader.zip`
- Place the downloaded file to the location where you want to save the KITTI dataset
- Open the Ubuntu command prompt line
- Then simply run ./raw_data_downloader.sh
- Dataset will start downloading and will take good amount of time to download depend on the speed of internet also

6.3 Preparing Training Data

The downloaded dataset is divided into two main part. One part contains the raw stereo images in form of left_image, right_image. The other part contains the depth images which are in form of left_annotated_depth, right_annotated_depth.

To read all the files inside the folder and extract the image names in put those images inside a text file was achieved with the successfully library of python called OS.

The code snippet below is shown below.

```
import os
a = open("output.txt", "w")
for path, subdirs, files in os.walk(r'_________'):
    for filename in files:
        a.write("_____________________")
```

The dataset comprises of the various sorts of data which can be used for different purpose like object detection, optical flow, 3D object detection etc. For the depth estimation, we take the data set from KITTI raw data(stereo RGB images) and depth data (stereo depth images). Figure 6.3 gives a general idea about the split of data which we have used in a semi-supervised learning way. On the left-hand side, we have raw data which comprises of the stereo image(unsupervised part of training) shown in the form of the left image and right image. On the Right-hand side, we have the depth data(supervised part of training) which comprises of the left depth image and right depth image. we combine both approaches of supervised and unsupervised to form the semi supervise training which over the shortcoming of the supervised and unsupervised training.

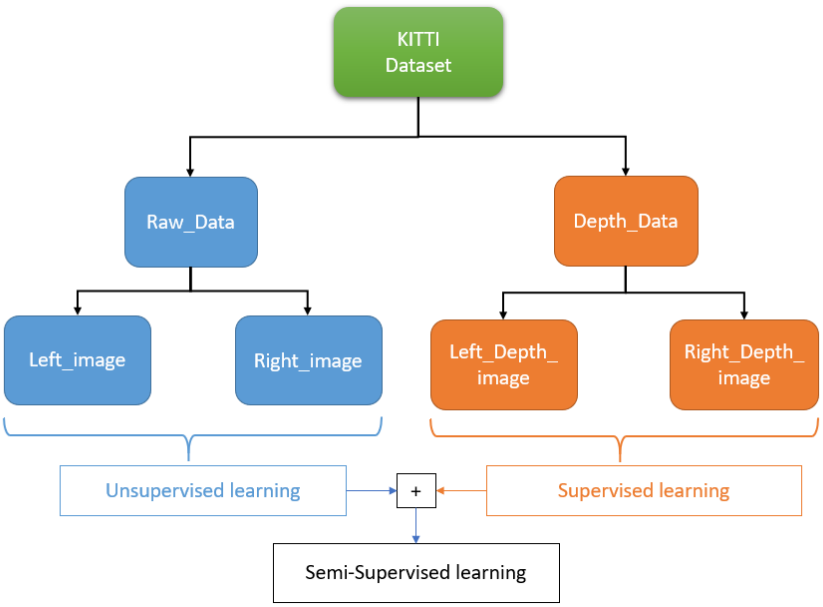

Figure 6.3: *General Split of KITTI Dataset for Training*

For training we split the data in two form

- KITTI Split with Depth_annotated Data.
  - For training 57378 raw images which cover about 33 scenes from KITTI dataset.
  - For training 57378 depth annotated images with 1,386 images for validation from KITTI dataset.
  - left_image, right_image, left_annotated_depth, right_annotated_depth

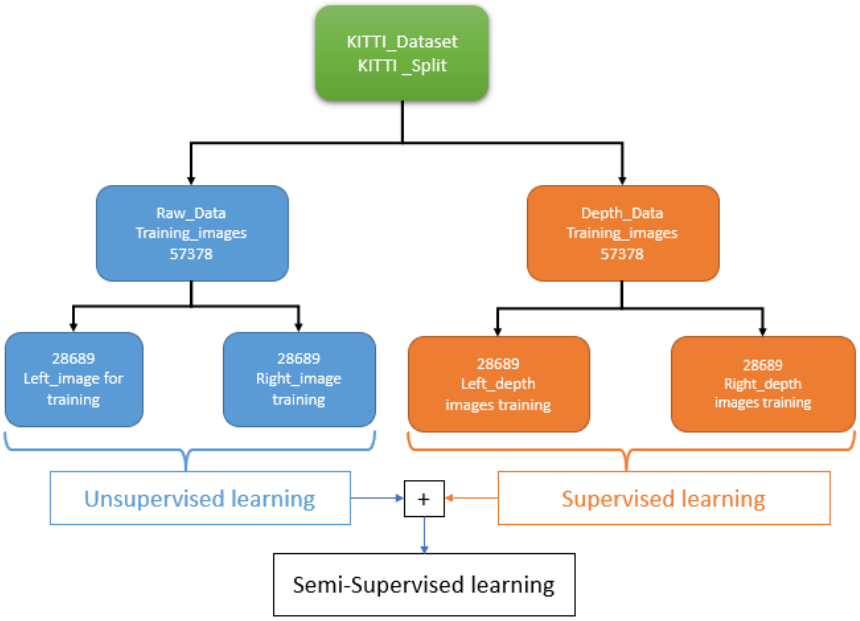

Figure 6.4: *KITTI Split with Depth_annotated Data[10]*

KITTI split is comprised of 57378 stereo images which can be further divided into 28689 left images and 28689 right images. The KITTI split has more than the 33 scenes which have around 30,000 stereo images from which the supervised training we have used from 30,000 stereos images near about 28689 stereo images. For the evaluation, we have used the 697 images proposed by[51]. The depth information which can be extracted through these images is about 80 meter. For unsupervised learning, we have used the 57378 depth images which are shown in figure left depth image and right depth image. Combining both data for training provide a more robust model.

- Eigen Split with Depth_annotated Data
  - For training 44,400 raw images which cover about 33 scenes from KITTI dataset.
  - Test split of 697 images which covers a total of 29 scenes proposed by[6].
  - 21,591 depth annotated images for training with 694 depth images for validation and test 315 images.
  - left_image, right_image, left_annotated_depth, right_annotated_depth

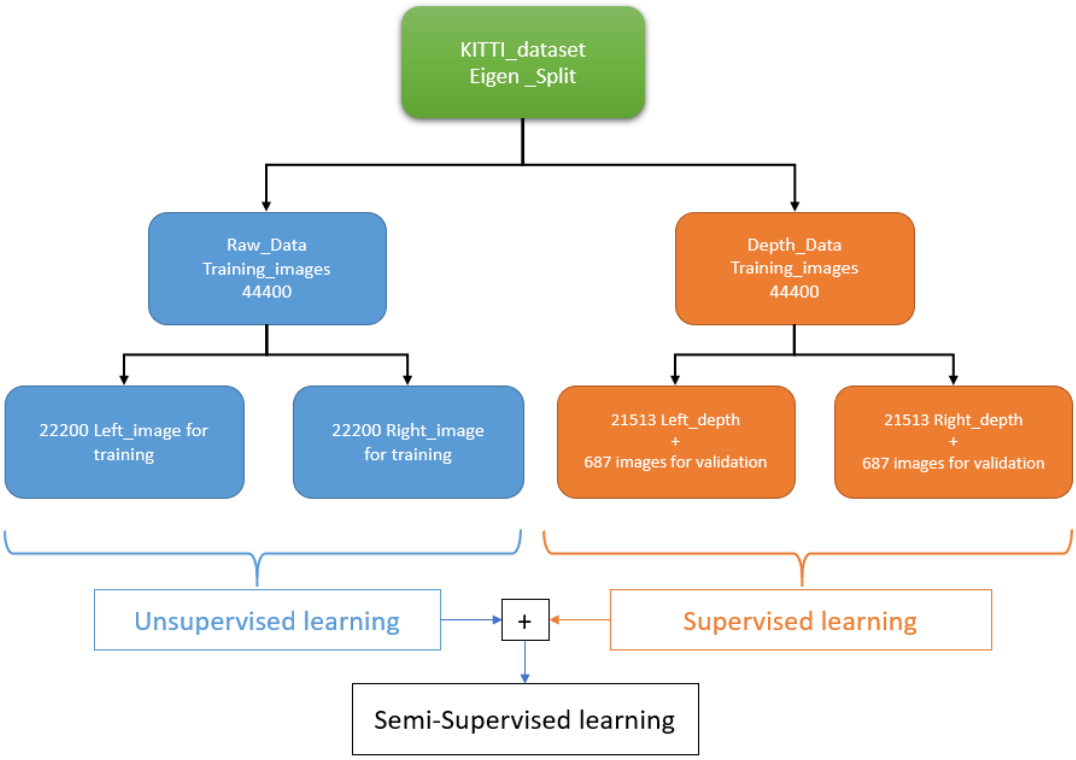

Figure 6.5: *Eigen Split with Depth_annotated Data[10]*

Eigen split is comprised of 44400 stereo pair of images for training which can be further divided into the 22200 images left image and 22200 right images this using the stereo pair of images is the unsupervised learning part. For the supervised learning as seen in figure 6.5, we have used about 22200 left depth images and 22200 right depth images.

7 Methodology

This is one of the main chapters for the thesis report. In this chapter, we will discuss the concept of disparity and inverse, therefore using these concepts for defining a better 3D shape of the scene. The chapter will also discuss how the left image can be generated from the right image using a dense corresponding field or some wrapping function. The chapter will also deal with the depth estimation network which covers the steps for final depth estimation model and inference. This chapter will deals also with the training loss that we encountered for making a stable model and at the end of the chapter will focus on implementation deals and environment setup for training and inference.

7.1 Depth Estimation as Image Reconstruction

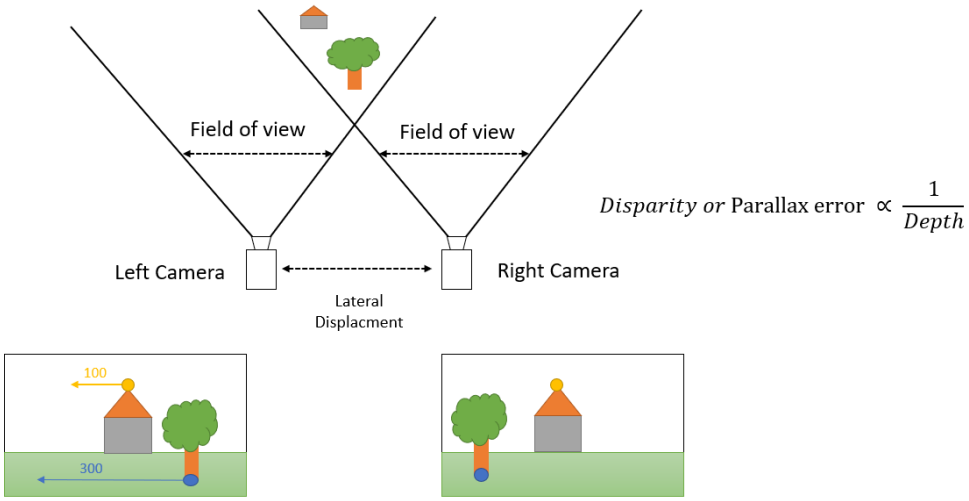

Figure 7.1: *Disparity to Depth relation*

Depth estimation as an image reconstruction is a well-practised method[7]. Before we start getting deep about how to extract depth from image reconstruction we have to understand about the concept of disparity and parallax error which is directly related to the disparity. The disparity can be defined as the displacement(lateral displacement) when viewing from the two different viewpoints. Figure 7.1 we can see that we have two cameras one is on the left-hand side and the other is on the right-hand side. Both the camera has a different viewing angle or we can say the field of view. The parallax error is the measurement of displacement which is inversely proportional to the depth. When we are able to define the parallax error it is easy to recover the depth information. The

multi-viewing image system provides a robust depth estimation as to when compared to monocular cameras.

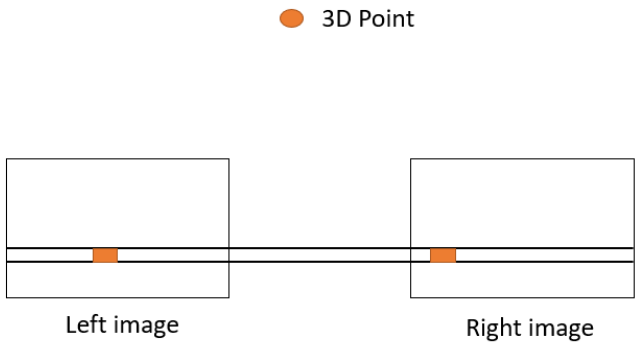

Figure 7.2: *Disparity Calculation*

The parallax error is defined by the difference in the image location of the same 3d point when trying the view from two different view angle and is projected in the same perspective. From figure 7.2 and equation 7.1, we can say that the disparity is the shift of the pixel left to the right image. The depth information is directly related to the baseline which is the distance between the two cameras of a stereo geometry. So when the distance of the baseline is of longest possible length better the depth information we would be able to gather. This gives us a brief idea of the stereo vision of the depth estimation now we move the more mathematical model.

$$d = xleft - xright \tag{7.1}$$

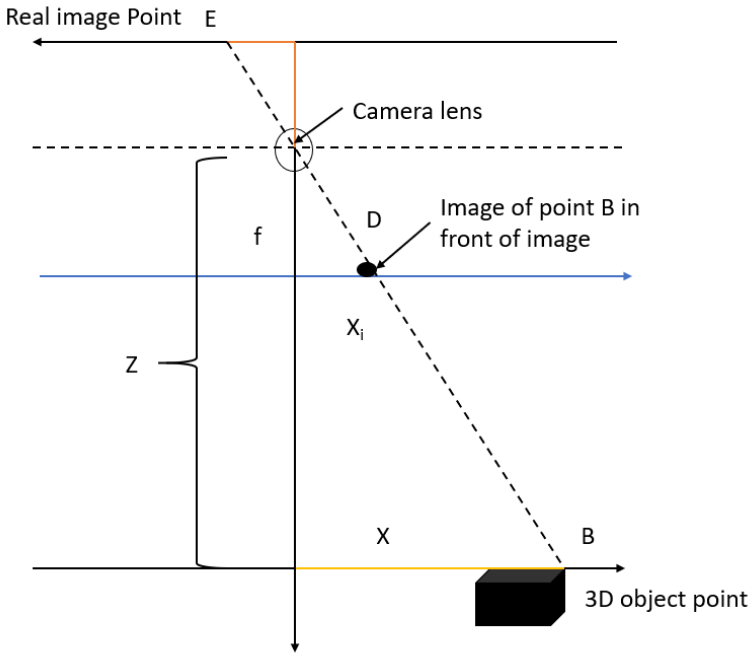

Figure 7.3: *Camera to object depth calculation*

From the figure 7.3

$$\frac{x_i}{f} = \frac{x}{z} \tag{7.2}$$

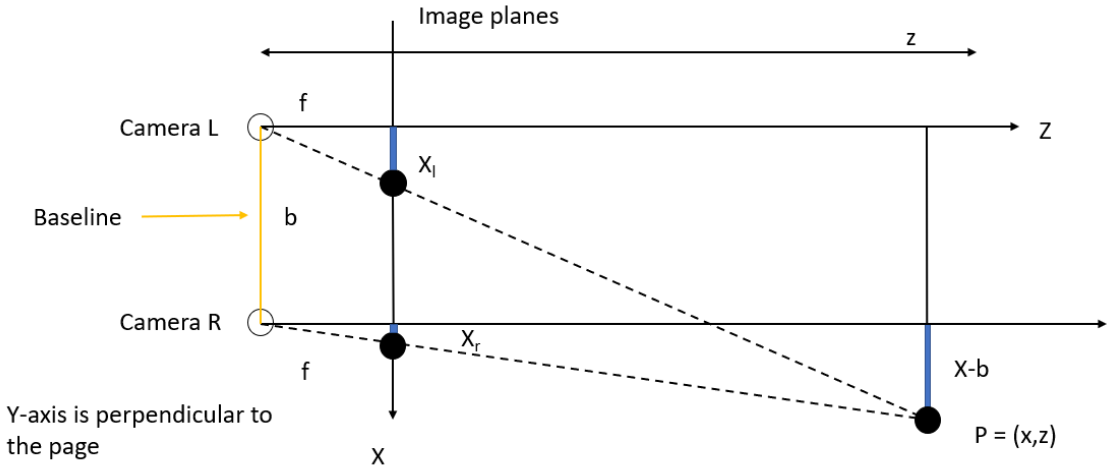

Figure 7.4: *Classic approach for depth estimation using two view camera*

From the similar triangle we obtain the equation 7.3, 7.4 and 7.5 . In the figure Y-axis is perpendicular to the plane.

$$\frac{z}{f} = \frac{x}{x_l} \tag{7.3}$$

$$\frac{z}{f} = \frac{x-b}{x_r} \tag{7.4}$$

$$\frac{z}{f} = \frac{y}{y_l} = \frac{y}{y_r} \tag{7.5}$$

In a stereo setup figure 7.4 and with follow information below we can obtain 3d location of a point(x, y, z) in a scene or the depth. Hence we can say that with the information of disparity we can find the depth.
Optical axes
f : focal length
b : baseline
(x_l, y_l) : image points for left camera
(x_r, y_r) : image points for right camera

$$z = \frac{f \times b}{x_l - x_r} = \frac{f \times b}{d} \tag{7.6}$$

$$x = \frac{x_l \times z}{f} = b + \frac{x_r \times z}{f} \tag{7.7}$$

$$y = \frac{y_l \times z}{f} = \frac{y_r \times z}{f} \tag{7.8}$$

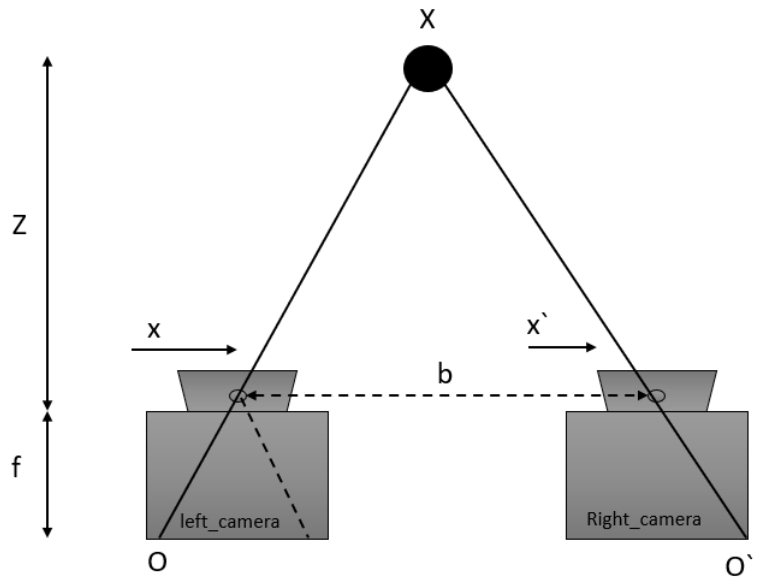

Figure 7.5: *Stereo camera for depth estimation*

In figure 7.5 for both the camera the optical centre is defined by the O and O‘. The main problem to find the depth. As we have seen the depth is dependent on the focal length and the baseline of the cameras. If we are able to discover these parameters we can find the depth. So moving back to the figure 7.5 the x and x‘ define the point on a 2d plane. To depth, we have used the concept of the similar triangle in the figure 7.5 we have two similar triangles Ofx and Ofx‘ and after comparing both to each other we get equation 7.9 for disparity and from disparity we can find the depth information in a scene.

f : focal length
b : baseline
z : depth
d : disparity

$$d = disparity = x - x‘ = \frac{f \times b}{z} \tag{7.9}$$

At this point, we are able to under what is disparity and depth and how we find depth we have some knowledge of intrinsic parameters like focal length and baseline in a stereo setup. The main task for us is to construct a left image from the right image and vice-versa using the intrinsic parameter to find the depth information of a 3d scene. In a given scenario when we have calibrated cameras we can function such a dense function or a wrapping function which can learn to reconstruct one image from one another. The dense function helps us to understand the 3d shape of the scene.

Left image: I^l
Right image:I^r
To try to find some corresponding dense function which relates or help to construct the left image from the right image and vice versa.
Reconstructed left image:I^l_{rec}

$$I^l_{rec} = I^r(d^l) \tag{7.10}$$

$$d^l = \frac{f \times b}{z} \tag{7.11}$$

Reconstructed Right image:I^r_{rec}

$$I^r_{rec} = I^l(d^r) \tag{7.12}$$

$$d^r = \frac{f \times b}{z} \tag{7.13}$$

d^r and d^l : wrapping function give the relation between the (baseline, focal length) to the disparity.
Ω_{lidar}: depth information in lidar image per pixel.
Ω: Pixel in image.

At a given point when we know baseline distance between the cameras and the focal length of the camera, we find the depth from the disparity $d = bf/z$. This part of image reconstruction can be categories into unsupervised learning part and for the supervised part, we use the depth image at the time of training. The depth images are the laser measurement which projects to a sparse subset $\Omega_{lidar} \subset \Omega$ of a pixel in the image and this provides the effective information for depth at the time of training.In the supervised part, we try to measure the variation of the predicted depth $\rho(x)^-1$ from the available ground truth $\Omega_{lidar}(x)$ at each pixel level(x).

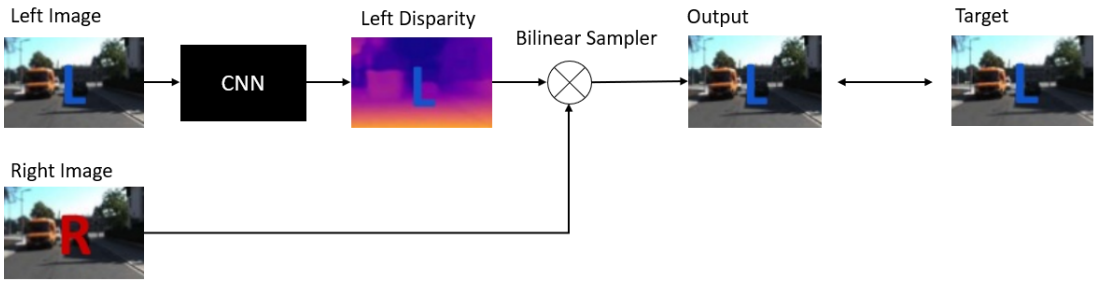

Figure 7.6: *Native approach for depth estimation*

In a traditional approach for depth estimation when only using the stereo pair of images are used at the time of training the target is to predict one the images from the stereo

images. The figure 7.6 we can see that stereo images (left and right) are given at the training time our target is that model produces an output image and using reconstruction loss between the output image and in the target image. At this point also we are unable to get the depth information. To get depth the network produces a disparity and using the input image given to a bilinear sampler which is fully differentiable and provides a good reconstruction quality.

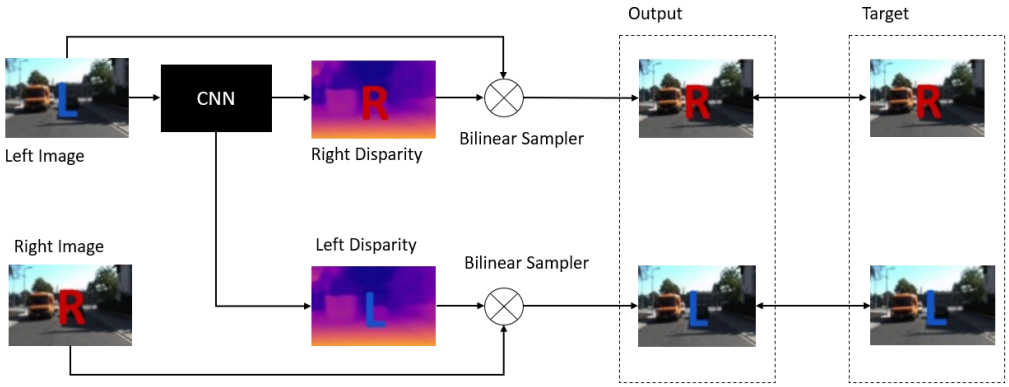

Figure 7.7: *Depth estimation using Left and Right consistency[7]*

Now moving to the research done by Godard et al [7] gives one image to the network and then sample the other image to generate the output. To make a more robust network they tried to predict the two disparities one for the left image and one for the right image and enforcing the produced disparities to be consistent with each other and they train the network with this left-right consistency loss and providing both of the images to the bilinear sampler to produce the output image using the reconstruction loss and smoothness loss to train train the network end to end.

7.2 Depth Estimation Network

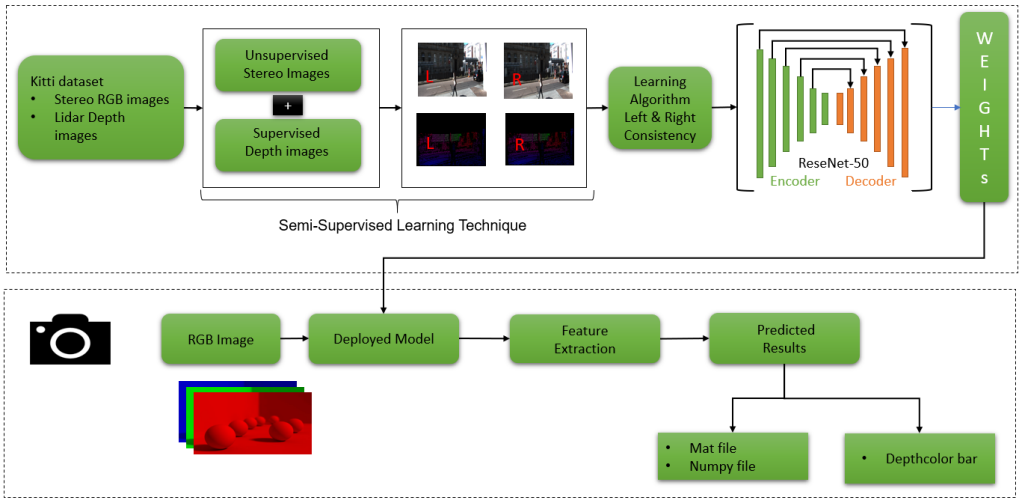

Figure 7.8: *Our approach for Depth Estimation*

So at this point, we know the advantages of unsupervised training with stereo images. So in this master thesis, we try to bring light to semi-supervised learning for depth estimation. To get to the point of semi-supervised learning we introduce the depth images taken from lidar at the time of training to make a more robust model. In past, many types of research have been done get to get the state of art result through a monocular camera. As seen from the figure 7.8 which shows the KITTI dataset provides us with the data for the training. The major data we have taken is Stereo RGB images which correspond to the left and right image. Another major data which we included is depth images taken from lidar which also provided by the KITTI dataset. The figure shows the combination of both supervised and unsupervised cues. The semi-supervised learning is a method to overcome the difference between the lidars field of view and camera, therefore, increase the good dense depth maps. To exploit the benefits of semi-supervised learning we introduced left and right consistency loss for image reconstruction with smoothness loss which helps to bring the state of art results. The major of training is done over the ResNet-50 which is the encoder part of the architecture. The decoder parts consist of the many small convolution blocks which help in getting the dense depth. The architecture uses skip connection in from the encoder block which helps in recovering a high-resolution dense depth maps. The process of training continued until the required weight or stable training weight are achieved. When the training weights have achieved the process of inference can be initiated. For the inference, we take an RGB image and use the trained weight given to the deployed model and extracting the important feature to get the depth. The depth information is in the form of numpy array and mat file which store the information of depth for each pixel in the image. This stored information of depth in an array is then used by the python matplotlib function to visualize the depth.

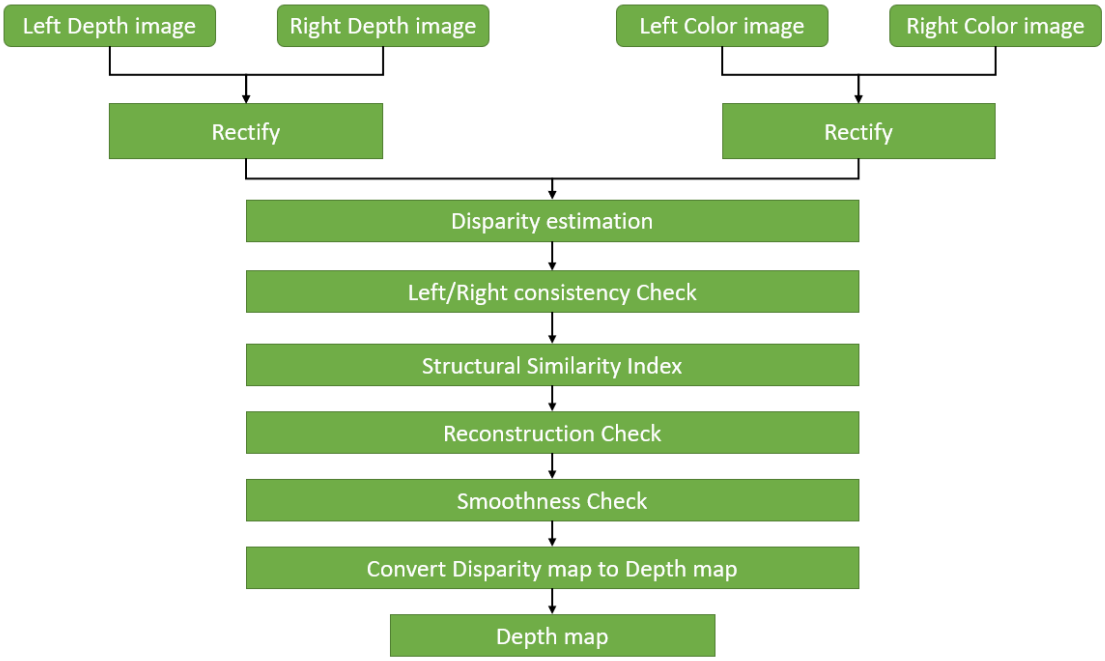

Figure 7.9: *Flow chart for our approach*

Figure 7.9 shows us the flow chart for training. The training uses the stereo image which

is shown in the right-top part of figure 7.9. The other training data comes from depth images which are shown in the left-hand top of the figure 7.9 the images are then rectified to get the images to a common plane. Throughout the training, various losses are being continuously checked like left-right consistency loss which helps to find the consistency loss between the output image to the target image for this we also uses the structural similarity index which is just a function for comparing the structural similarity of the two images. Two main functions are being used one is unsupervised reconstruction loss which checks the loss between the left image and right image are dense using the corresponding dense wrapped function. Another training loss is supervised loss which measures the difference between the generated inverse depth to the ground truth inverse depth. The last loss we use the smoothness loss just get the better depth when the side pixel are in low gradient image regions. All these losses help to find the depth from the disparity.

7.3 Training Loss

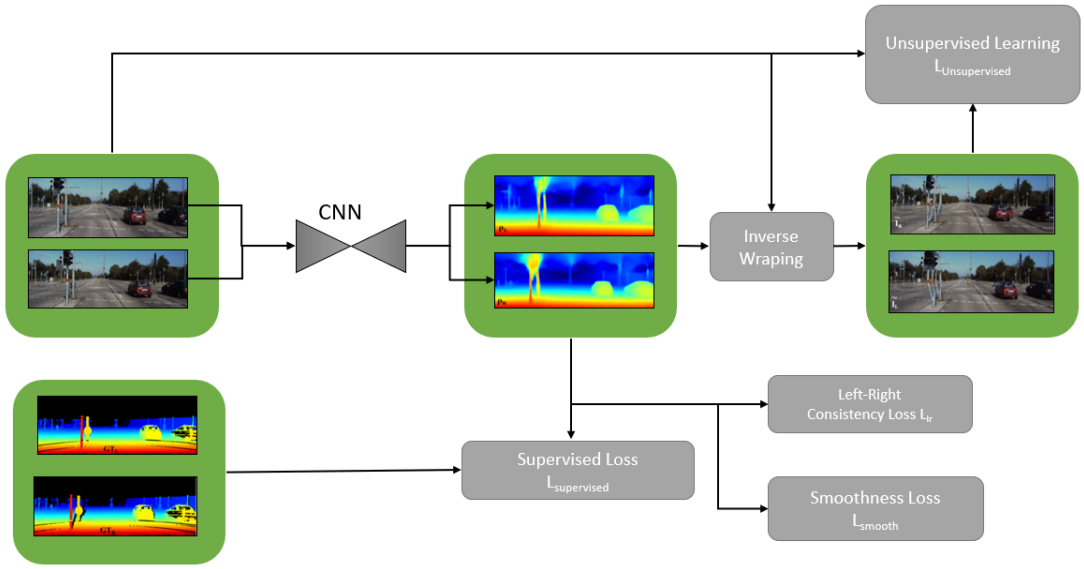

Figure 7.10: *Different Training loss*

One can describe training loss how bad [66] a model can predict for a single input in general. The training loss is the penalty for a bad prediction[66]. The loss defined how the model will predict the results. There is a direct relationship between the relation between training loss and model prediction. When the prediction by the model is perfect or good as required then the training loss is very low or near to zero but when the prediction by the model is not perfect or bad the training loss is greater which the model will not perform well. To build the model for depth estimation we have four major losses supervised loss, the unsupervised loss left and right consistency loss and final the smoothness loss all these losses would be discussed in brief in below section. Figure 7.10 shows us the various loses. We train the CNN to get the inverse depth than one

of the stereo images is given to the sampler and using the inverse wrapping function we try to generate the other corresponding image which known as the predicted image. Then we try to compare the predicted image to the target image and see that prediction is appropriate. This type of loss is known as unsupervised learning. The depth images which are during the training is known as the supervised part of the training. For supervised loss, we try to measure the predicted inverse depth to the available ground truth inverse depth. To check the consistency between the left-right image and vice versa we use the left-right loss it's a type of loss is a forward-backwards which check for every predicted image. the final smoothness loss which also shown in figure 7.10 check for the smoothness in inverse depth sometimes the low gradient region when the shadow of some kind of unclear region the smoothness loss bring more clarity to the depth.

To describe all the losses together we get this equation 7.14. λ_i is constant which can be changed according to required.

$$L_s = \lambda_1 L_{Unsupervised} + \lambda_2 L_{Left-RightConsistencyLoss} + \lambda_3 L_{supervised} + \lambda_4 Lsmooth \quad (7.14)$$

7.3.1 Supervised Loss $L_{supervised}$

In this model the advantage of ground truth data which is depth images taken from a lidar sensor are being used during the training process. One research did by the Kuznietsov et al[54] in 2017 at the RWTH Aachen University also uses depth images for training. To define the supervised loss one can relate to comparing the predicted depth to the ground truth depth for all the pixel which are available in the ground truth. The equation 7.15 describes the supervised loss where the comparison between the predicted depth for each pixel in an image to the ground truth is shown.

$$L_{supervised} = \sum_{k \in (l,r)} \frac{1}{M_k} \sum_{i,j \in \Omega_k} ||\rho_{i,j}^k - Z_{i,j}^{-1^k}||_1 \quad (7.15)$$

$L_{supervised}$: Supervised loss
Ω_l: Points where Ground truth depth are accessible for left image
Ω_r : Points where Ground truth depth are accessible for right image
M_l: For total number of pixel where the ground truth is accessible for left image
M_r : For total number of pixel where the ground truth is accessible for right image
Z^-1: Ground truth inverse depth
ρ : Predicted inverse depth

7.3.2 Un-Supervised Loss $L_{unsupervised}$

The unsupervised loss can be described as a reconstruction loss which uses the photo-consistency between the left and right image. In a stereo setup, we try to generate the left image from the right using a dense inverse wrapping function. The wrapping function is dependent on the baseline of two cameras in a stereo setup, the focal length of the camera and inverse depth. In this loss, the inverse wrapping function is being used to estimate the generated left and right image and then check whether the generated left and

right images are correct to its left and the right target image. In this to check between the left and right generated to left and right target image we have used structural similarity index. To make the model fully differentiable we have proposed a method which uses the bilinear sampler. The equation 7.16 below defines the supervised loss what we have used to train the model.

$$L_{unsupervised} = \sum_{k \in (l,r)} f(I^k, \widetilde{I}^k) \tag{7.16}$$

$$f(I, \widetilde{I}) = \frac{1}{N} \sum_{i,j} \alpha_1 * \frac{1 - SSIM(I_{ij}, \widetilde{I}_{ij})}{2} + \alpha_2 * ||I_{ij} - \widetilde{I}_{ij}||_1 + \alpha_3 * census(I_{ij}, \widetilde{I}_{ij}) \tag{7.17}$$

Reconstructed left image:I^l_{rec}

$$\widetilde{I}^l = I^r(d^l) \tag{7.18}$$

$$d^l = \frac{f \times b}{z} \tag{7.19}$$

Reconstructed Right image:I^r_{rec}

$$\widetilde{I}^r = I^l(d^r) \tag{7.20}$$

$$d^r = \frac{f \times b}{z} \tag{7.21}$$

$L_{unsupervised}$: Unsupervised loss
I^l: left image
I^r : right image
$\widetilde{I}^l$: Reconstructed left image
$\widetilde{I}^r$: Reconstructed right image
N: Total number of pixel
$\alpha_1, \alpha_2, \alpha_2$: Scalar quantity
α_2 : Scalar quantity
d^l : left disparity
d^r : Right disparity
$Census$: ternary census taken form [67]
f : Focal length
b : Baseline
z : depth

Bilinear sampler

It is part of the linear transformation. In the bilinear interpolation, the interpolation occurs in both the direction x and y. When on image any sort of scaling or rotation

is being performed the pixel are being moved. For example before rotation a point in an image it at (x,y) and after transformation the point is changed to (x+a,y+b). This a big problem in the vision-related task which involves images. The location of the pixel in the output image does not map the location to the output image. To overcome this problem we use the bilinear interpolation. In bilinear interpolation try to get the four different points $(Q_{11}, Q_{12}, Q_{21}, Q_{22})$ which diagonal to each other using them the base we try to find the helping points (R_1, R_2). Using the helping we try to find the main point $P = (x, y)$. This approach helps us bring more differentiable image which is realistic to the input image.

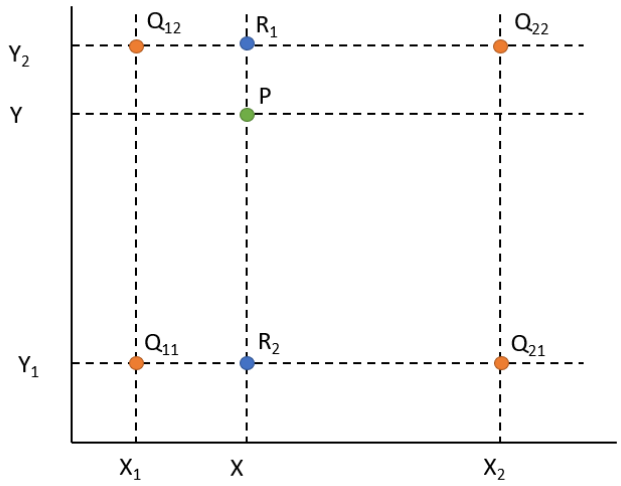

Figure 7.11: *Understanding Bilinear sampler*

$P = (x, y)$: Pixel point
$Q_{11} = (x_1, y_1)$
$Q_{21} = (x_2, y_1)$
$Q_{12} = (x_1, y_2)$
$Q_{22} = (x_2, y_2)$
(Q_{11}, Q_{21}) : Average weighted point
(Q_{12}, Q_{22}) : Average weighted point
R_1 : Unknown pixel point
R_2 : Unknown pixel point

Interpolation in x direction to get $R_1 \& R_2$ point

$$R_1 = \frac{x_2 - x}{x_2 - x_1} Q_{11} + \frac{x - x_1}{x_2 - x_1} Q_{21} \tag{7.22}$$

$$R_2 = \frac{x_2 - x}{x_2 - x_1} Q_{12} + \frac{x - x_1}{x_2 - x_1} Q_{22} \tag{7.23}$$

Interpolation in y direction

$$P = \frac{y_2 - y}{y_2 - y_1} R_1 + \frac{y - y_1}{y_2 - y_1} R_2 \tag{7.24}$$

In one research done at google regarding the spatial transform[38]. They conclude a better way of finding a robust model which is invariant to the input variation in images. During the training time, the main problem is the data. To increase the data we use some sort of transformation like rotation or scaling. These transformations bring more challenges to make the pipeline differentiable and as when using pooling layer which also isolates the important features by reducing the spatial size this can make the model unlikely perform under average.

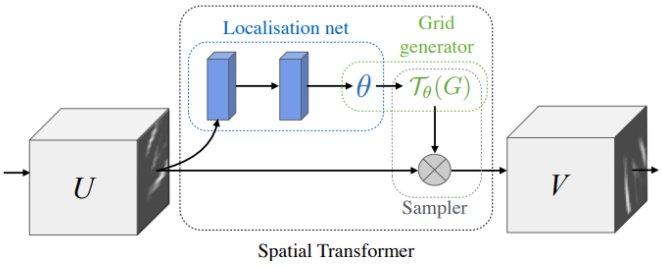

Figure 7.12: *Spaital Transform Model [38]*

To overcome all these challenges of modular[38], differentiable[38] and dynamic[38] the google[38] introduced this spatial transform network. The network comprised of three main parts first to be localization nets[38] which help in getting the affine transformation parameters which would be applied to the input feature maps. The second part of the network is called a grid generator[38] which gives the desired output after applying the transformation. When getting the feature map after the transformation to make it differentiable the bilinear sampler[38] comes into the light. In the figure 7.13 we see the image which is known as U and the output of the image is V. The figures show in the (a) part where the identity function is applied and in the (b) part the affine transformation is being applied to get the output. In our case, we have used a bilinear sampler to make the pipeline for training differentiable and to judge the right feature information.

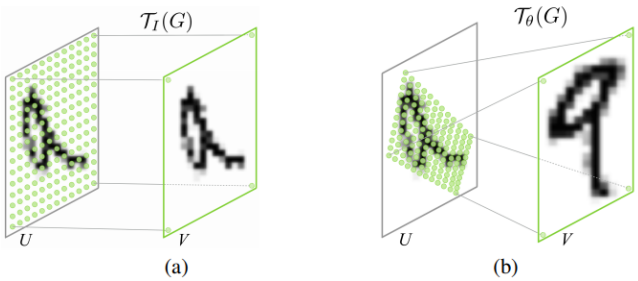

Figure 7.13: *Results when using the Bilinear sampler*

SSIM

SSIM is a method that uses the measures the structural similarity in between the two images. In our case, we have left and right image and the model predict the depth for these two images. Using the inverse wrapping function we produce the corresponding the reconstructed stereo images. To check the appearance of both reconstructed image

to the stereo image SSIM algorithm is used. For example, we have left the image and right to check the similarity between both the image equation 7.25 can be used.

$$E_{ssim} = \frac{1 - SSIM(I_l, \widetilde{I}_r)}{2} \tag{7.25}$$

The basic equation to calculate the structural similarity between the two images. The equation is as follow

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{7.26}$$

μ_x: define average of x
μ_y : define average of y
σ_x^2: define variance of x
σ_y^2 : define variance of y
σ_{xy}: define co variance of x and y
$c_1 = (k_1 L)^2$: variable to help division with not so strong denominator
$c_2 = (k_1 L)^2$: variable to help division with not so strong denominator
L : range of pixel values ($2^{bits_per_pixel} - 1$)
k_1 : 0.01 set as default
k_2 : 0.03 set as default

The equation 7.26 can be written in python code as follow .

```
def SSIM(self, x, y):
C1 = 0.01 ** 2
C2 = 0.03 ** 2

mu_x = slim.avg_pool2d(x, 3, 1, 'SAME')
mu_y = slim.avg_pool2d(y, 3, 1, 'SAME')

sigma_x  = slim.avg_pool2d(x ** 2, 3, 1, 'SAME') - mu_x ** 2
sigma_y  = slim.avg_pool2d(y ** 2, 3, 1, 'SAME') - mu_y ** 2
sigma_xy = slim.avg_pool2d(x * y , 3, 1, 'SAME') - mu_x * mu_y

SSIM_n = (2 * mu_x * mu_y + C1) * (2 * sigma_xy + C2)
SSIM_d = (mu_x ** 2 + mu_y ** 2 + C1) * (sigma_x + sigma_y + C2)

SSIM = SSIM_n / SSIM_d

return tf.clip_by_value((1 - SSIM) / 2, 0, 1)
```

Census with unflow

In this section, we talk about how to make the network robust especially when dealing with the vision-related task. The main problem arises in the vision-related task that relationship between the one sequence to the other sequence or defining one frame to

another frame is generally distorted which means the information is not available or occluded. The main task is to bring more robustness to the model as a continuous parameter. The introduction of charbonnier loss with the ternary census brings more robustness to the algorithm[68]. This is in particular help in bringing more clarity to the image-related task like depth estimation and optical flow estimation.
In a mathematical term, the charbonnier loss can be expressed as

$$\rho(x) = (x^2 + \epsilon^2)^{\alpha} \tag{7.27}$$

The figure 7.14 show us the charbonnier loss when α = 0.45 and 0.25. The charbonnier loss is not the most accurate but consistent.

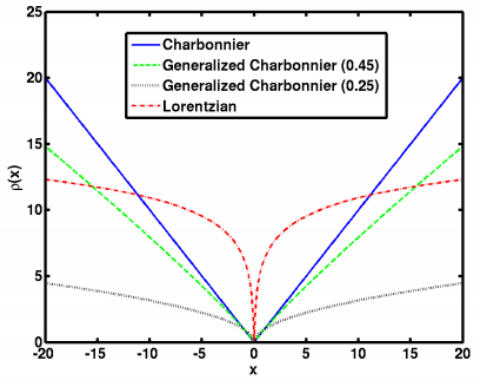

Figure 7.14: *Charbonnier loss when α = 0.45 and 0.25*

In the method of depth estimation, we use a stereo pair of images for training to get the photometric consistency between the corresponding pixels the charbonnier function bring more robustness to the algorithm. The stereo pair of images have a huge variation of illumination which means some of the pixel information are occluded. To realize the change in illumination we have used the ternary census [69] which incorporates to reduce the illumination and gamma [70] changes in the image. Therefore helping in getting a reliable image consistency through the image.

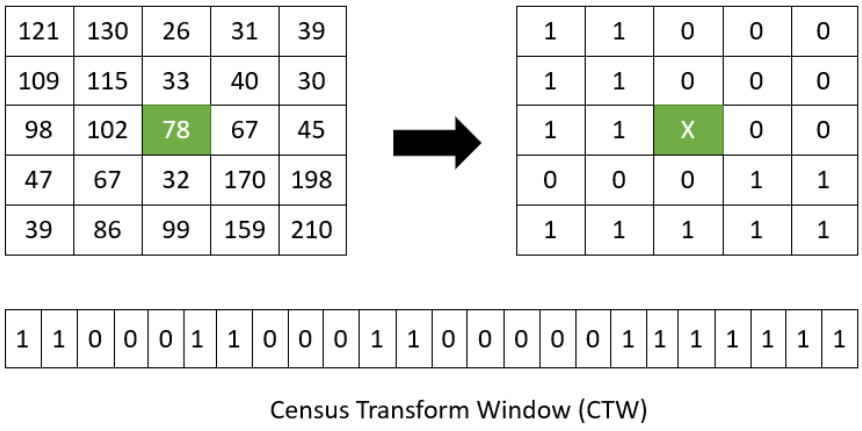

Figure 7.15: *Census Transform Window*

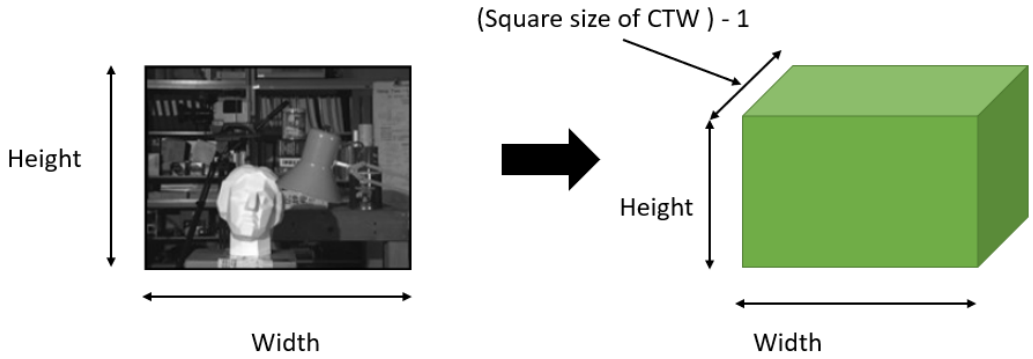

Figure 7.16: *Census Transform*

In the ternary census suppose we have an image and the image has the value for each pixel as shown in the figure 7.15. In the ternary census, the centre pixel is taken as a baseline for checking the intestines. Then the intestines are check for each pixel which are surrounded by the centre pixel. If the intensities are greater than the centre pixel they are assigned one. The intensities which are lower than the centre pixel are assigned zero. This help us in getting the perfect intensities when the stereo pair of images are used. The output of an image through a census transform data (image size * vector size) see figure 7.16. In the stereo setup, we have two images one is the left image and other is the right image and other is the left image. Due to the inconsistency of the illumination and gamma, the information from one pixel for the left image cannot correlate to the pixel in the right image and vice-versa. To avoid this confusion of correlation of pixel from left to right image we apply census transform and we get two images which we call then left census transformed vector and right census transformed vector. To find the right judgment of pixel we use the Hamming distance which helps in finding the right 3D disparity space see figure 7.17.

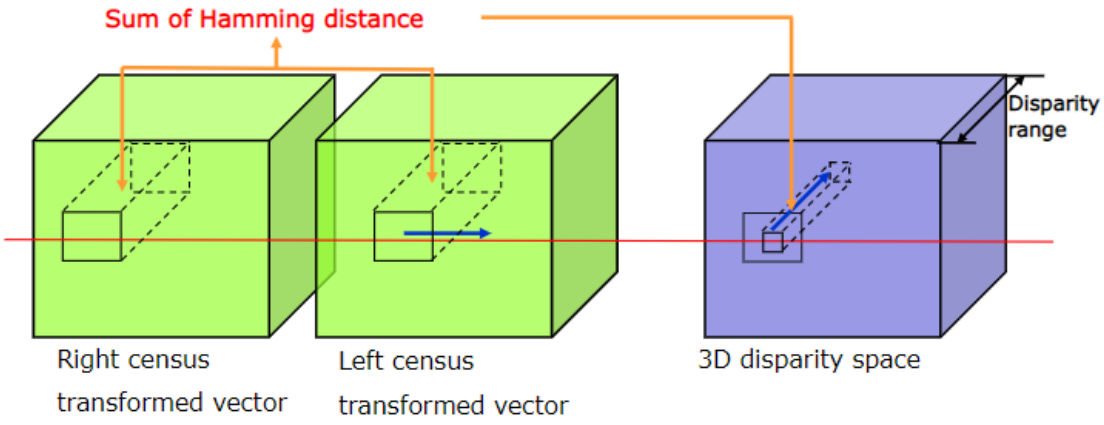

Figure 7.17: *Using Hamming Distance to find Disparity*

7.3.3 Left-Right Consistency Loss L_{lr}

This an important loss to make a robust model for depth estimation. The left and right loss measure the contribution of both the left and right image are equal at the time of training. One can describe the left and right consistency loss to be sure the predicted inverse depth of the left image is consistent to the inverse depth of right image and vice-versa. The left-right loss is defined by equation 7.28.

$$L_{lr} = \frac{1}{N} \sum_{i,j} ||\rho^l_{ij} - \rho^r_{ij+d^l_{ij}}||_1 + ||\rho^r_{ij} - \rho^l_{ij+d^r_{ij}}||_1 \tag{7.28}$$

$$d = baseline * f * \rho \tag{7.29}$$

L_{lr} : Left-Right Consistency Loss
ρ^l: predicted left inverse depth
ρ^r : predicted right inverse depth
d^l: predicted left disparity
d^r : predicted right disparity
N: Total number of pixel
f : Focal length
b : Baseline
z : depth

7.3.4 Smoothness Loss L_{smooth}

In some region of the image where the gradient is low or region is not differentiable. The smoothness loss brings light to cover up the backdrop of the low gradient region. This help to get better depth values in the region where the depth information is hard to get. The smoothness loss can be described as in equation 7.30.

$$L_{smooth} = \frac{1}{N} \sum_{k \in (l,r)} \sum_{i,j} |\delta_x \rho^k_{ij}| \exp^{-|\delta_x I^k_{ij}|} + |\delta_y \rho^k_{ij}| \exp^{-|\delta_y I^k_{ij}|} \tag{7.30}$$

L_{smooth} : Smoothness Loss
δ_x: Gradient x
δ_y : Gradient y
ρ^l: predicted left inverse depth
ρ^r : predicted right inverse depth
N: Total number of pixel

7.4 Edge detection

Edge detection is the process of extracting the important information from an image which very much relevant for determining the relation to the properties and structure of an object in a particular scene[71]. Edge detection is an initial step when we want to

extract the building steps for a model. To understand what does edge detection means one can relate to the slight change in intensity within the image or it can be described as the discontinuity in the image intensity. In a mathematical term, edge detection is a derivative of variation in the intensity of an image. The variation can be of any type like step, ramp, line and roof see figure 7.18.

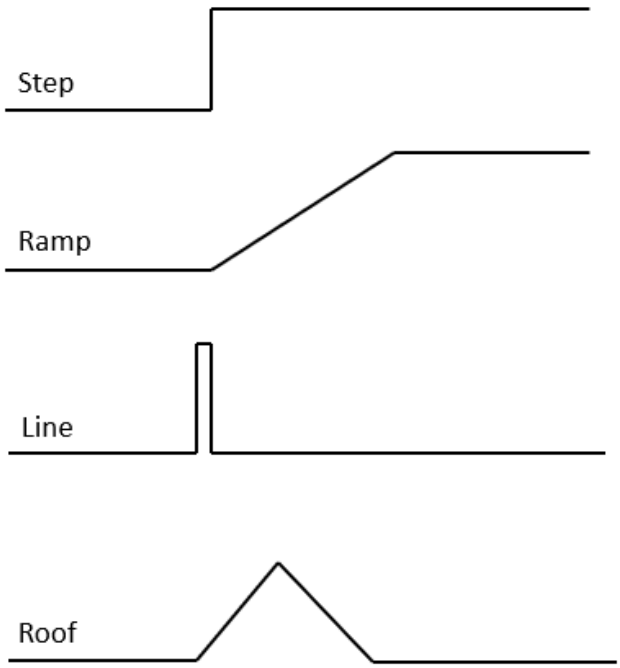

Figure 7.18: *Edge detection using various function*

Now we have a knowledge of what is an edge detection what does it means. To get more understanding the word gradients bring more light to edge detection which measures the alteration in the function which relates to image intensity. The image is a simple collection of an array for each pixel denoting a value which defines intensity in the image. The gradient help in finding the significant alteration in the values in an image. In 2D the gradient can be represented as a vector.

$$G[f(x,y)] = \begin{bmatrix} G_x \\ \\ G_y \end{bmatrix} \tag{7.31}$$

The gradient have two main properties associated with it. One is the above equation $G[f(x,y)]$ which show the relation to the maximum rate of increase of the $f(x,y)$.

$$G[f(x,y)] = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{7.32}$$

Second is the magnitude of the gradient.

$$G[f(x,y)] = \sqrt{G_x^2 + G_y^2} \tag{7.33}$$

To get the direction of the gradient with respect to x axis the alpha α is measured.

$$\alpha(x,y) = tan^{-1}(\frac{G_x}{G_y}) \tag{7.34}$$

To relate to more to a digital image the gradient equation is slightly changed to simple approximation.

$$G_x \widetilde{=} f[i,j+1] - f[i,j] \tag{7.35}$$

$$G_y \widetilde{=} f[i,j] - f[i+1,j] \tag{7.36}$$

In the above equation the j represents to the x direction and i represents to the y direction.

$$G_x = \begin{bmatrix} -1 & 1 \\ \\ -1 & 1 \end{bmatrix} \tag{7.37}$$

$$G_y = \begin{bmatrix} 1 & 1 \\ \\ -1 & -1 \end{bmatrix} \tag{7.38}$$

To make the gradient more approachable to real-world coordinate there is some approximation done to get the real coordinate (x,y) to get the position in the space. The G_x is approximated to the new interpolated point [i,j+0.5] and similarly, the G_y is also approximated to the new interpolated point [i+0.5, j].to understand this might be confusing so to be more accurate there are other approaches like Sobel filter which uses 3*3 neighbourhood filter which helps to the gradient which is relative to centre pixel.

Sobel Filter

This is an innovative approach which overcomes the drawback of the traditional approach of getting the to get the interpolated points[71]. The Sobel filter uses a 3*3 neighbourhood to get the gradient. To understand this see figure 7.19 which shows how the other pixels are arranged around the pixel[i,j]

a_0	a_1	a_2
a_7	[i,j]	a_3
a_6	a_5	a_4

Figure 7.19: *Interpolation in Sobel Filter*

The magnitude of the gradient in sobel filter can be defined by:

$$M = \sqrt{s_x^2 + s_y^2} \tag{7.39}$$

The s_x and s_y are computed as follow in which c is a constant.

$$s_x = (a_2 + c * a_3 + a_4) - (a_0 + c * a_7 + a_6) \tag{7.40}$$

$$s_y = (a_0 + c * a_1 + a_2) - (a_6 + c * a_5 + a_4) \tag{7.41}$$

Both S_x and S_y the gradient operator can be shown as a convolutional mask suppose when c = 2.

$$s_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{7.42}$$

$$s_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{7.43}$$

7.5 Training

The training is a major part of depth estimation. The training is a process in which the important feature is being extracted and integrate the algorithm which is related to the task we want to perform. The training can be of different types, in general, we can classify the training in the three categories. First, be supervised training where the ground truth or label information is available. The second type of training is where the ground truth is not available or the label information is not possible to get. The final type of training we say to the semi-supervised training in this the labelled (ground truth information) and unlabeled data are both being used. This type of training, in general, used where the information about the task is available but not completely. At this point, the semi-supervised learning provides better robust model related to the task.

In our case, we have chosen semi-supervised learning for depth estimation. The main reason to consider this approach is that in the supervised learning to get the ground truth data for training is a big problem. The ground truth data means the depth images taken from lidar which also do not provide the complete depth information and are typically sparse. When only using the unsupervised learning the stereo pair of images do not provide consistent depth information. So we know that both supervised and unsupervised learning have their own shortcoming but when both the approaches are combined some outstanding results can be achieved. In semi-supervised learning, we

introduced the depth images taken from lidar and the stereo RGB images for training. Both the data are taken from KITTI dataset which provides a vast range of data from the scenario for training.

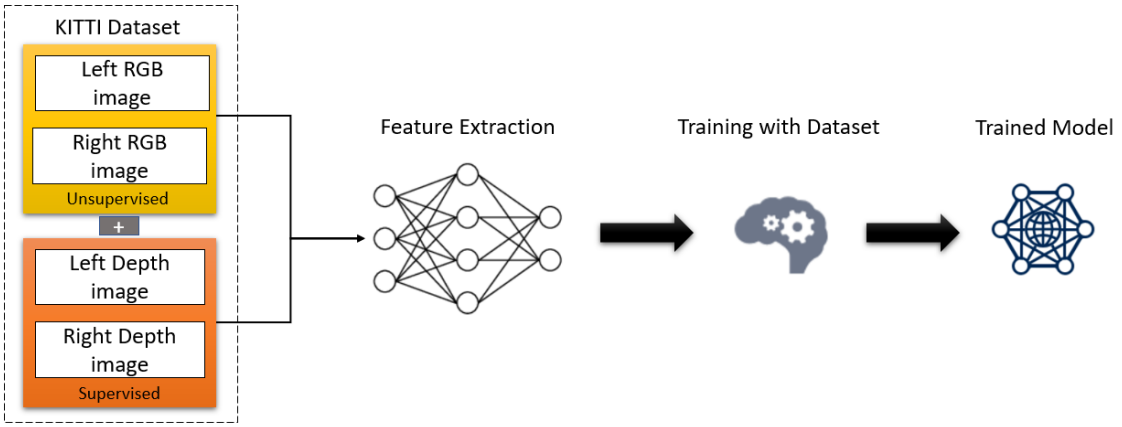

Figure 7.20: *General Split for training*

As we can see from figure 7.20. The general idea of how the data for training has been split. The first section of the figure 7.20 we see the supervised part which has all the depth images (Left depth image, Right depth image) and the unsupervised part we have stereo images (Left RGB image, Right RGB image). From the figure 7.20 at lower level, the feature extraction takes places until the important features are being learned by the convolutional net. After the feature extraction, the training parts are over which provide us with a trained model weight which we can use for depth estimation.

7.5.1 KITTI General Split

The general KITTI split is prospered by[10]. The splits comprised of 28689 Left depth image and Right depth image which is supervised part of the model. For the unsupervised, we have use the 28689 Left RGB image and Right RGB image.

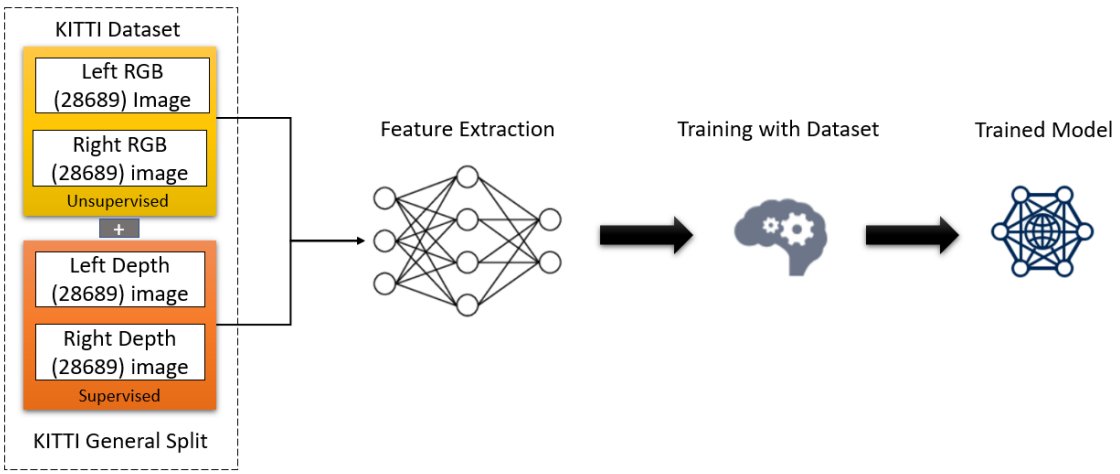

Figure 7.21: *KITTI General Split for training*

The training data is approximate about 200 Gb which pretty much heavy data to train with. The process of feature extraction uses the concept of disparity which is better explained in section 7.1. The training in general means no extra filter are being used to get better depth information. To better outline, the depth information in case of low gradient region no extra feature extraction filter like Sobel filter is taken into account. The figure 7.21 gives an idea of how the training is being done in Kitti general split.

The section below of the code is using the normal gradient to detect the edges in the x direction.

```
def gradient_x(self, img):
        gx = img[:,:,:-1,:] - img[:,:,1:,:]
        return gx
```

7.5.2 KITTI with gradient fix Split

This split have the same amount of training data as described in section 7.5.1. The only major change is the introduction of a sobel filter for edge detection. The filter provides a better definition of depth. The sobel filter is an innovate approach to calculate the gradient. In section 7.3.4, a better description of the Sobel filter with edge detection technique is being discussed.

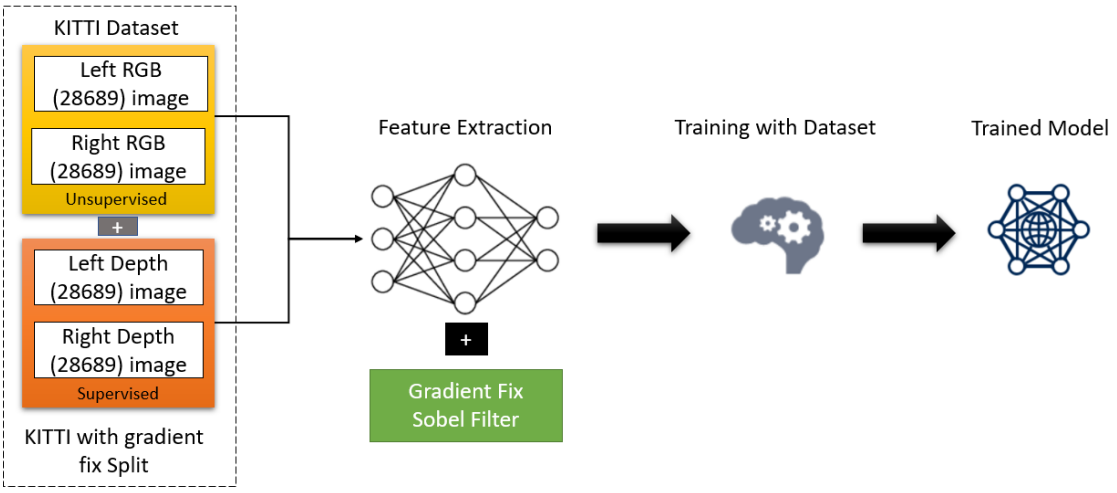

Figure 7.22: *KITTI with gradient fix Split for training*

Below we show a small snippet of the code which we use Sobel filter. The filter provide better edge detection with smoothness to the edges in the low gradient areas.

```
def gradient_x(self, img):

sobel_x_filter = tf.constant([[-1, 0, 1],
                              [-2, 0, 2], [-1, 0, 1]], tf.float32)

if img.get_shape().as_list()[-1] == 1:
```

```
x_filter = tf.reshape(sobel_x_filter, [3, 3, 1, 1])
else:
x_filter = tf.tile(tf.reshape(sobel_x_filter,
           [3, 3, 1, 1]), [1, 1, 3, 1])
gx = tf.nn.conv2d(input=img,
     filters=x_filter, strides=[1, 1, 1, 1],
     padding='SAME')
return gx
```

In the code above, we take the Sobel filter which is defined by the sobel filter then we reshape the Sobel filter. We apply the filter over the image in both the x and y-direction. Sobel operator provides better edge detection and hence better dense depth information. To understand more about the Sobel filter read section 7.3.4.

7.5.3 Eigen General Split

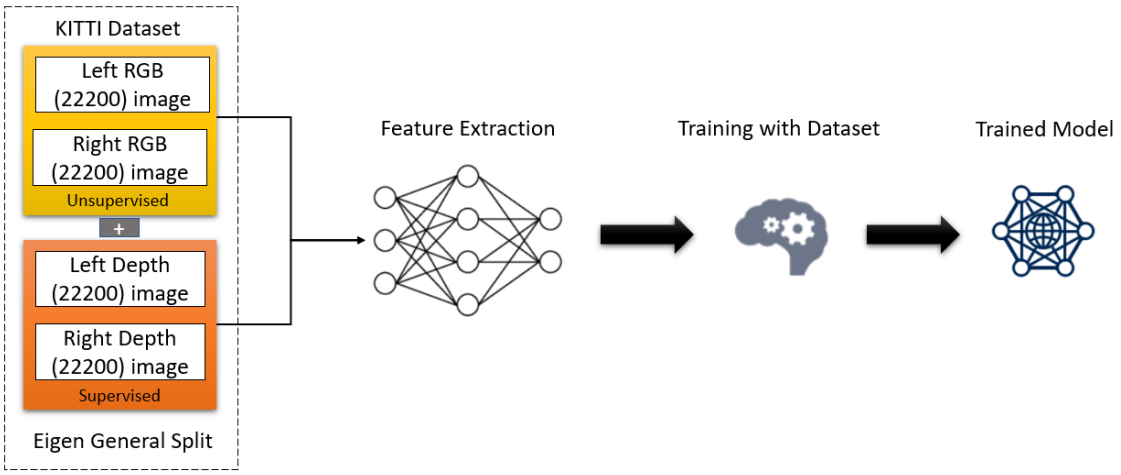

Figure 7.23: *Eigen Split for training*

In this split, we try to change the number of images for training to see the variation in the model but the convolutional nets remain the same. This split is a standard as proposed by [7][10]. The Eigen general splits have around 32 scene which contains about the 23,488 stereo pair images. From 23,488 we have used 22,200 images for training. The network is the same as the kitti split. What we try to do to generate the depth images by using the Velodyne laser images taken from the KITTI data set and give it to our model for tarining . To test the data we use Garg crop[6].

7.5.4 Eigen with gradient fix Split

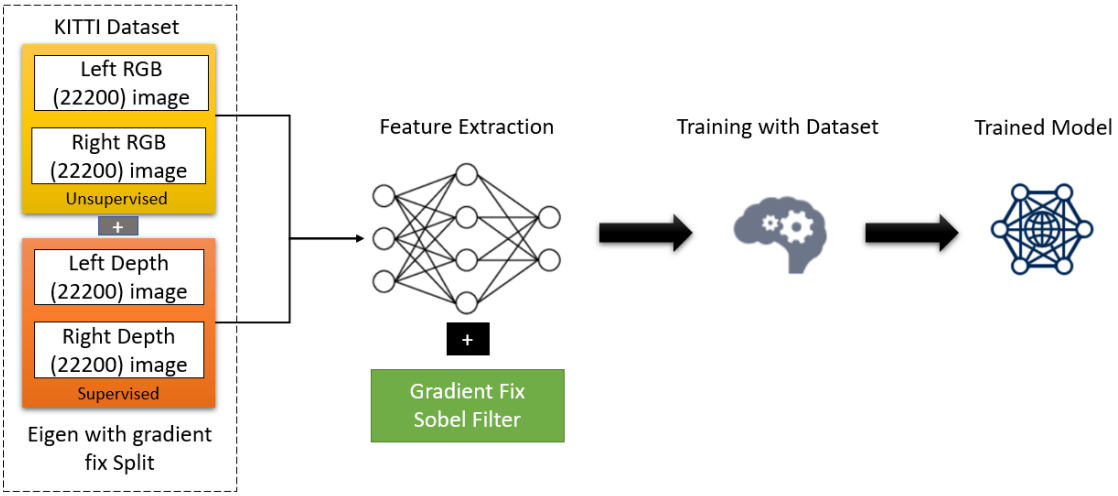

Figure 7.24: *Eigen with gradient fix Split for training*

This split uses the Sobel filter as we have discussed in the section 7.3.4 which helps in getting the better edge detection and to capture better features. In images to get the simple alteration due to the illumination(gradient) is a challenging task. To tackle this task we use the Sobel filter which provides efficient depth information to generate and visualize the depth for every pixel in an image. The number of training images is about 22,200. Which relates to 22,200 stereo pair of images and 22,200 depth images taken from Velodyne laser. The results and evaluation of this split will be discussed in chapter 8.

7.6 Hyperparameters

Hyperparameters is one of the key parts of the trained model. They help to find how the model behaves. They are not fixed they general defined by the coder. They vary from task to task. The hyperparameter includes the learning rate, number of epochs, hidden layers and activation functions. Hyperparameter help in controlling the algorithm and provide a major advancement in the performance. So it's very important to choose the right hyperparameter for the training.
For better optimization, we have used Adam optimizer instead of the traditional stochastic gradient descent. The main part of the Adam optimizer help in getting a better learning rate as it uses a square gradient for scaling the learning rate. In Adam optimizer gives uses the better momentum to define the learning rate instead of taking the direct gradient itself it uses the average of the gradient to get the right momentum. At this point, we have used the word momentum a lot now the main question of what is momentum according to the learning rate. Suppose we have a random variable(N) with its Nth moment. So the momentum can be defined as the expected value of the Nth random variable. In mathematical term, we will be shown below where m is the moment and x is the random variable.

$$m_n = E[x^n] \tag{7.44}$$

Adam optimizer is an intuitive method which involves an adaptive learning rate which in the Adam optimizer uses the first and second moment of gradient estimates to change the learning rate for each neural network weight. The first moments are the simple mean of the small batch of the data which we have decided. The second moment is variance. Adam optimizer is an intuitive method which involves an adaptive learning rate which in the Adam optimizer uses the first and second moment of gradient estimates to change the learning rate for each neural network weight. The first moments are the simple mean of the small batch of the data which we have decided. The second moment is variance.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{7.45}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_1^2 \tag{7.46}$$

$m$and v: moving average
g: gradient on current batch
β beta range from (0.9 to 0.999)

As we have talked about moment and gradient now the task is to use both of them and get a better optimization. The Adam optimizer constructs itself around exponentially moving average as described in the equation 7.45 and 7.46 calculate the gradient on the batch size what the coder has decided.

Description	Value
Optimizer	Adam optimizer
Learning rate	10^{-4}
β_1	0.9
B_2	0.999
ε	10^{-8}
Epoch	25
Batch size	8
Input image size	256*512
Inverse depth range	0 to 1.0 (Sigmoid function)

Figure 7.25: *Implementation Details*

For this master thesis, we have used the set of parameters which can be dived into two categories one are the parameter which is used for the general implementation of the

model. The other are the parameters which are related to the loss function. In figure 7.25 we see the important parameter for the implementation. The input size of the image is about 256*512 pixels which we have decreased as to declare the number of computational parameters. For the activation function, we have used the sigmoid function which helps the network to produce the inverse depth in the range of 0 to 1.0. For regularisation, we used the optimization techniques like Adam optimizer with the beta ranging from 0.9 to 0.99. the learning rate which means at which rate the algorithm will staring learning for the given number of epoch with fixed batch size. The learning rate is an important parameter we have used the stand general learning proposed by[11][7][10].

Description	Value
λ_1	1
λ_2	1.0
λ_3	150.0
λ_4	0.1
α_1	0.85
α_2	0.15
α_3	0.08

Figure 7.26: *Hyperparameters for loss function*

$$L_s = \lambda_1 L_{Unsupervised} + \lambda_2 L_{Left-RightConsistencyLoss} + \lambda_3 L_{supervised} + \lambda_4 Lsmooth \quad (7.47)$$

$$L_{unsupervised} = \sum_{k \in (l,r)} f(I^k, \widetilde{I}^k) \quad (7.48)$$

$$f(I, \widetilde{I}) = \frac{1}{N} \sum_{i,j} \alpha_1 * \frac{1 - SSIM(I_{ij}, \widetilde{I}_{ij})}{2} + \alpha_2 * ||I_{ij} - \widetilde{I}_{ij}||_1 + \alpha_3 * census(I_{ij}, \widetilde{I}_{ij}) \quad (7.49)$$

The hyperparameter for the loss function is scalar quantities. They are decided according to the behaviour of the loss function. They are selected as per which values perform best for getting a lower value for the complete loss. We see the four scalar quantities λ_1 λ_2 λ_3 λ_4 which are used in equation 7.47. From the research done by [7][10] we picked the best values for these four scalar quantities which get a very adaptive lower loss function. The value of λ_1 λ_2 λ_3 λ_4 are defined in the figure 7.26. The α_n in the figure is also a scalar quantities which are used in the unsupervised loss .We use the value of $\alpha_1 = 0.85$ as

proposed in [11] and $\alpha_2 = 0.15$ $\alpha_3 = 0.08$ proposed in [10] . The best part is that we can always play with this scalar values and change according to get lower loss value for the loss function.

7.7 Environment Setup for Depth Estimation

The setting up environment is quite a challenging task. To tackle this we have made it simple with some commands which we will discuss in this section. The major libraries which we have used are TensorFlow, Numpy and OpenCv. To enable the use of the GPU for training and inference purpose we have to set the CUDA tool kit, CUDNN the version varies and we need to update the driver of GPU. As we have talked earlier that the training data is about 200Gb so to train with this huge amount of data we need the higher robust GPU which have sufficient core to handle the training. The most of interference is done on high-end GPU. The GPU we have used is GeForce RTX 2080Ti which take about 25 hours to train the model on average. In this section, we discuss installing the external environment setup for the TensorFlow. The TensorFlow version is 1.4 which is being installed on our machine having window 10 with GPU GeForce RTX 2080Ti.

7.7.1 TensorFlow

The figure gives an overview of the TensorFlow architecture. The hierarchy at each level of TensorFlow APIs is shown in the figure 7.27. The high - level TensorFlow APIs is the main brain which helps to bring the functionality to the layers in the bottom. This help in getting the model to start training, inference and other kind top-level function that the network has to perform. The mid-level APIs is an intermediate layer which help in loading the data and processing it and maintaining the flow of connection between the top layer to the bottom layer. The lower level layer is coder layer mainly the programming are written.

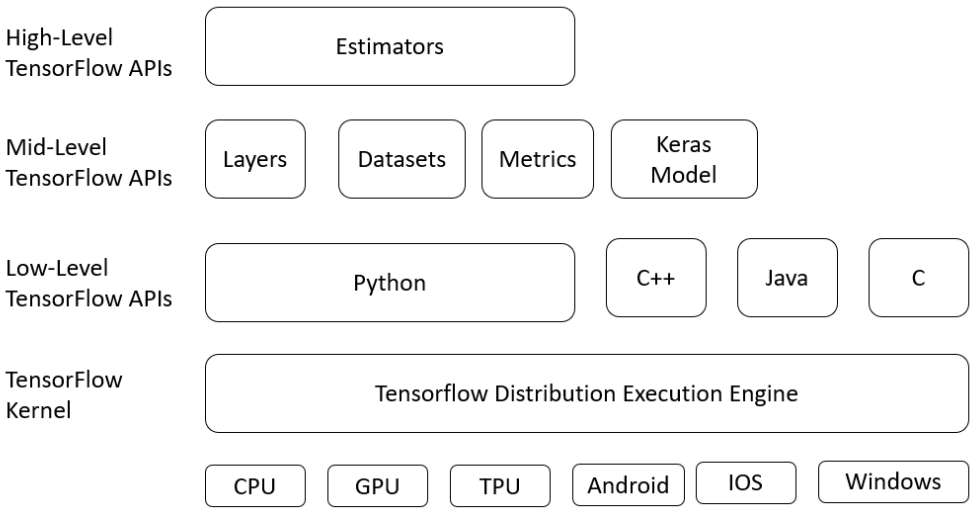

Figure 7.27: *Dependencies*

Installing TensorFlow

- Step1: Download the CUDA tool kit (check for the version for our was 10).
- Step2: Now go start and type “environment variable” and click to path variable and select edit.
  Add the following path

```
C:“Program Files“NVIDIA GPU Computing Toolkit“CUDA“v10.0“bin
C:“Program Files“NVIDIA GPU Computing Toolkit“CUDA“v10.0“libnvvp
C:“Program Files“NVIDIA GPU Computing Toolkit“CUDA“v10.0“extras“CUPTI“libx64
```

- Step3: Download CUDNN for version 10 of CUDA
  - When you download CUDNN version for CUDA tool kit 10 extract the files and place it in C drive.
  - Now go start and type “environment variable” and click to path variable and select edit.
    Add

```
C:“cuda“bin
```

- Step 4: Go to anaconda website download anaconda for python 3.7.
- Step 5: Open command prompt window run the command below.

```
pip install --ignore-installed --upgrade tensorflow-gpu
```

CUDA

For deep learning platform, Cuda help in getting the set of instructions sent to the GPU. To define the CUDA its an interface for parallel computing which enables to uses of GPU for high-level computation. In the figure 7.28, we can see how the instruction is fetched and decoded. As we use the GPU which is capable of doing multiple instructions at a single time. The basic idea of the figure 7.28 when a new instruction is being given to the GPU for computation (deep learning involves many matrix operations) the information is first fetched then the instruction is decoded and the operation related to the decoded instruction is executed and finally write back to the memory.

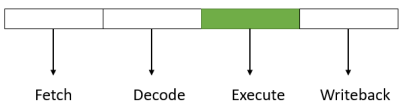

Figure 7.28: *One clock cycle per instruction*

The finally single instruction is now formed to do the parallel operation by stacking the multiple single instructions as shown in the figure 7.29. This help in better computation speed as in the single instruction takes 4 clocks and with parallelism, it’s possible to execute multiple instructions in four cycles. The figure 7.29 shows the takes one clock cycle to finish the three instruction.

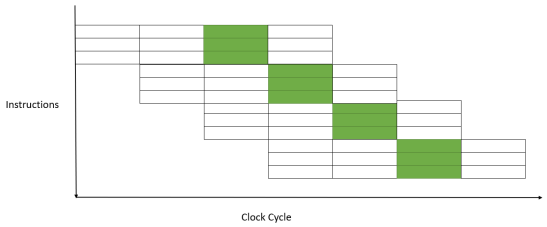

Figure 7.29: *Three instruction per clock cycle*

7.7.2 Libraries

After we have finish with installing TensorFlow and we can start installing all the dependencies.

```
name: thesis_depth
channels:
  - defaults
dependencies:
  - ca-certificates=2019.8.28=0
  - certifi=2019.9.11=py37_0
  - openssl=1.1.1d=he774522_2
  - pip=19.2.3=py37_0
  - python=3.7.4=h5263a28_0
  - setuptools=41.4.0=py37_0
  - sqlite=3.30.0=he774522_0
  - vc=14.1=h0510ff6_4
  - vs2015_runtime=14.16.27012=hf0eaf9b_0
  - wheel=0.33.6=py37_0
  - wincertstore=0.2=py37_0
  - pip:
    - absl-py==0.8.1
    - astor==0.8.0
    - catkin-pkg==0.4.14
    - cycler==0.10.0
    - docutils==0.15.2
    - ffmpeg-python==0.2.0
    - ffmpy==0.2.2
    - future==0.18.2
    - gast==0.3.2
    - google-pasta==0.1.7
    - grpcio==1.24.1
    - h5py==2.10.0
    - keras-applications==1.0.8
    - keras-preprocessing==1.1.0
    - kiwisolver==1.1.0
    - markdown==3.1.1
    - matplotlib==3.1.1
    - numpy==1.17.2
    - opencv-contrib-python==4.1.1.26
    - opencv-python==4.1.1.26
    - pillow==6.2.1
    - protobuf==3.10.0
    - pyparsing==2.4.2
    - pypng==0.0.20
    - python-dateutil==2.8.0
    - pyyaml==5.1.2
    - rospkg==1.1.10
    - scipy==1.3.1
    - six==1.12.0
    - tensorboard==1.14.0
    - tensorflow-estimator==1.14.0
    - tensorflow-gpu==1.14.0
    - termcolor==1.1.0
    - werkzeug==0.16.0
    - wrapt==1.11.2
prefix: C:\Users\sing\Anaconda3\envs\thesis_depth
```

Figure 7.30: *Dependencies*

This command “conda env create -f environment.yml” help to build all the dependencies shown in figure 7.27 with the environment name $thesis\_depth$.

8 Results and Evaluation

For the evaluation part, we have used the used 5 equation 8.1, 8.2, 8.3, 8.4, 8.5. The evaluation metrics are the main benchmark for evaluating the performance of the model. The depth estimation form a monocular camera can be stated as regression problem so RMSE can be used to check evaluation of the model. RMSE metrics calculate the number of information of depth which we produced through our model and compare it ground truth depth for the total number of pixels in that image. In the RMSE metrics, the lower output value shows the model performance is good and when the value is high the model is not a good model.

$$RMSE = \sqrt{\frac{1}{T}\Sigma_{i=1}^{T}||\rho(x_i)^{-1} - Z(x_i)||_2^2} \tag{8.1}$$

$$RMSE(log) = \sqrt{\frac{1}{T}\Sigma_{i=1}^{T}||log(\rho(x_i)^{-1}) - log(Z(x_i))||_2^2} \tag{8.2}$$

$$Accuracy = \frac{|i \in (1, ..., T)|max(\frac{\rho(x_i)^{-1}}{Z(x_i)}, \frac{Z(x_i)}{\rho(x_i)^{-1}}) = \delta < thr|}{T} \tag{8.3}$$

In the ARD and SRD which calculate the absolute and squared relative difference of the generated depth to the ground truth depth. In this scenario, the lower the value of ARD and SRD show a model is performing good and higher value show that the model needs some improvement. The minimum amount of depth which our model can get is zero meter and the maximum depth is about 80 meter in a given scene.

$$ARD = \frac{1}{T}\Sigma_{i=1}^{T}\frac{|\rho(x_i)^{-1} - Z(x_i)|}{Z(x_i)} \tag{8.4}$$

$$SRD = \frac{1}{T}\Sigma_{i=1}^{T}\frac{|\rho(x_i)^{-1} - Z(x_i)|^2}{Z(x_i)} \tag{8.5}$$

$\rho(x_i)^{-1}$: Inverse Depth
$Z(x_i)$: Ground Truth
T: total number of pixel

The model is trained over the Nvidia GPU (GEFORCE® RTX 2080 Ti) with 12 GB RAM configuration. As we have discussed the data set of 200 GB to train there was a requirement for high computation GPU with enough core to handle the processing. The figure 8.1 show us the name and the configuration the GPU which we have used for the training.

Hardaware (GPU)	Quantity	Architecture	RAM
GEFORCE® RTX 2080 Ti	1	Turing-GPU-Architecture	12 GB

Figure 8.1: *Hardware Requirement*

8.1 Kitti Split

The split consists of about 28689 left and 28689 right stereo pair and depth images for training. Figure 8.2 shows us the input image on the left-hand side and the predicted depth image on the right-hand side. The close (blue colour) in this figure 8.2 means zero meters from the viewpoint and the far (dark red colour) means the 80-meter depth information from the viewpoint. The results are good the model is able to calculate the depth information from the input image. As we can see that this model is able to detect persons, cars and surrounding and is able to evaluate to find relative depth information.

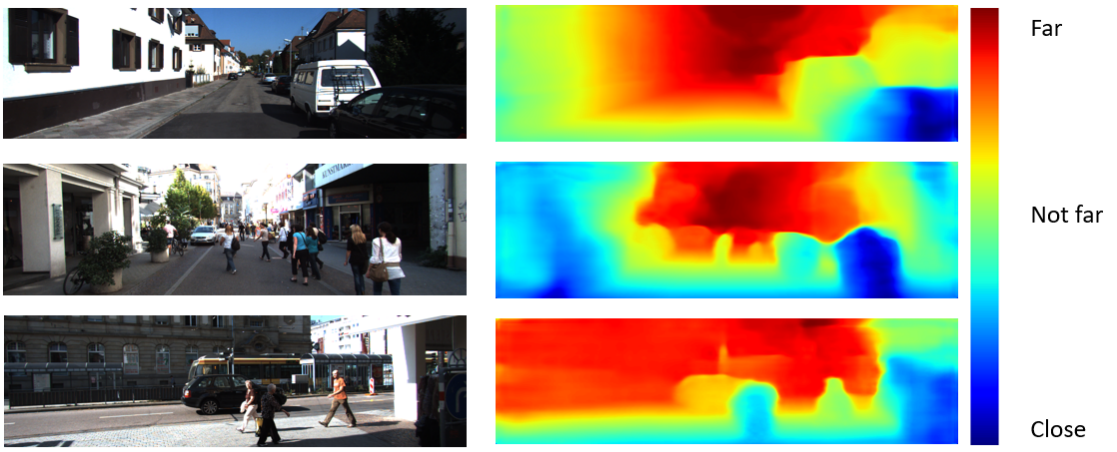

Figure 8.2: *Running Inference over KITTI Split Model*

The figure 8.3 shows us the accuracy of our model with comparison to the other researcher work. From the evaluation metrics, the RMSE value is 4.181 and in comparison to the other results, we are very close to the state of art results.

KITTI Split									
Method	Dataset	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	
Eigen et al. [48] Coarse	KITTI	0.214	1.605	6.563	0.292	0.673	0.884	0.957	
Eigen et al. [48] Fine	KITTI	0.203	1.548	6.307	0.282	0.702	0.89	0.958	
Liu et al [50]	KITTI	0.201	1.584	6.471	0.273	0.68	0.898	0.967	
Liana et al [5]	NYU	0.176		4.46	0.072				
Garg et al [6]	KITTI	0.169	1.08	5.104	0.273	0.74	0.904	0.962	Lower is better
Godard et al [7]	KITTI	0.141	1.369	5.849	0.242	0.818	0.929	0.966	
Godard et al [7] + CS	KITTI	0.136	1.512	5.763	0.236	0.836	0.935	0.968	Higher is better
Godard et al [7] + CS + Post-Processing	KITTI	0.126	1.161	5.381	0.224	0.843	0.941	0.972	
Kuznietsov et al [51]	KITTI	0.089	0.478	3.61	0.138	0.906	0.98	0.995	
Amiri et al [10]	KITTI	0.078	0.417	3.464	0.126	0.923	0.984	0.995	
Our Depth	KITTI	0.086	0.67	4.181	0.143	0.906	0.971	0.99	

Figure 8.3: *Results Comparison*

The figure 8.4 gives information about the learning rate. For the first 50000 iterations, the learning rate was kept steady at 1e-4 and after 50k iteration we changed the learning rate to be around 5e-4. The final learning rate was about 2e-4 which helped the model to learn important feature and bring stable depth information.

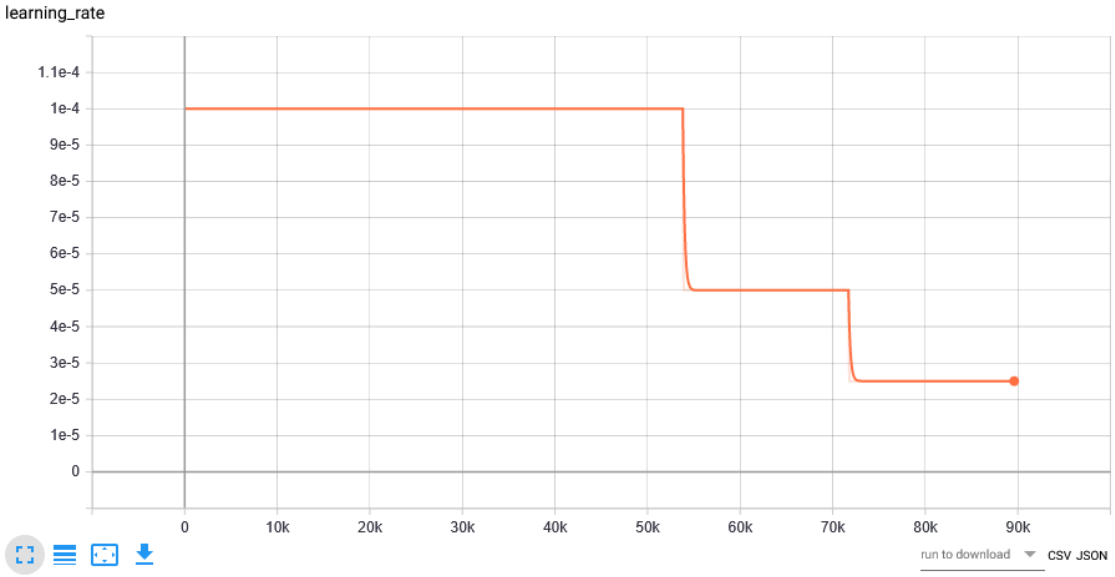

Figure 8.4: *Learning Rate*

Figure 8.5 gives information about the training loss. Using the tensorboard library we were able to see the total loss during the complete training time. The total loss is the combination of supervised, unsupervised, Left-Right consistency loss and smoothness which all discussed in chapter 7.

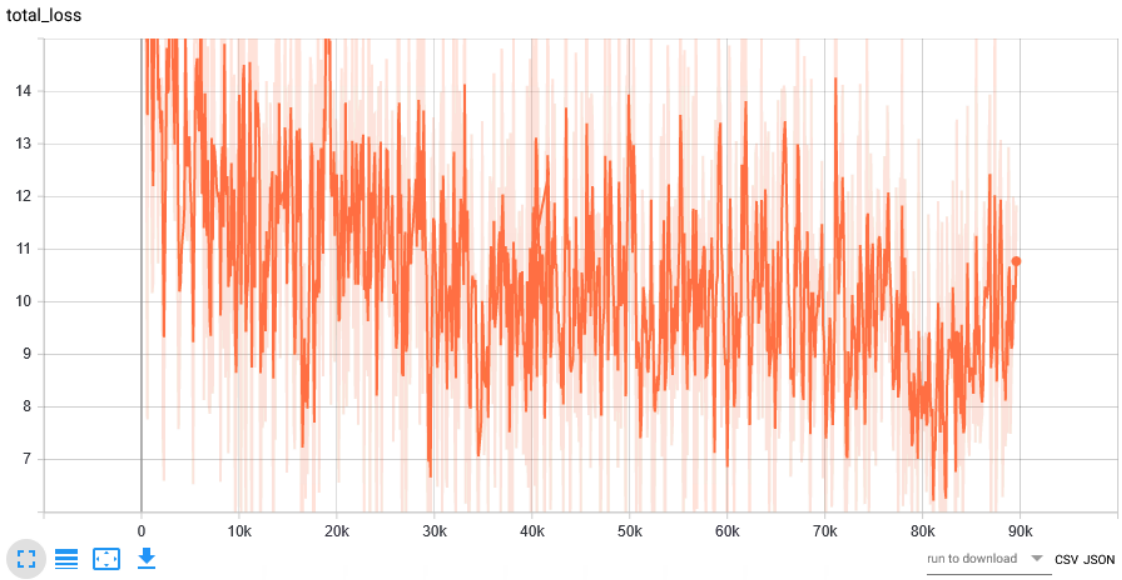

Figure 8.5: *Training Loss*

This is the main figure 8.6 which tells a lot about the model how it will perform. The hyperparameter was set with a batch size of 8. The total number of epochs we have 25 epochs and some scalar hyperparameters for the loss function is shown in figure 8.6 (b) part. The input size of the image is 256 * 512 which was reduced from the original size of 1392 * 512.

Description	Value
Optimizer	Adam optimizer
Learning rate	10^{-4}
β_1	0.9
B_2	0.999
ε	10^{-8}
Epoch	25
Batch size	8
Input image size	256*512
Inverse depth range	0 to 1.0 (Sigmoid function)

(a) General Hyperparameter

Description	Value
λ_1	1
λ_2	1.0
λ_3	150.0
λ_4	0.1
α_1	0.85
α_2	0.15
α_3	0.08

(b) Hyperparameter for loss function

Figure 8.6: *Hyper Parameters*

8.2 Kitti Gradient Fix

The results are stable the model is able to calculate the depth information from the input image. As we can see that this model is able to detect persons, cars and surrounding and is able to evaluate to find relative depth information. The split consists of about 28689 left and 28689 right stereo pair and depth images for training. Figure 8.7 shows us the input image on the left-hand side and the predicted output on the right-hand side. The close (blue colour) in this figure 8.7 means zero meters from the viewpoint and the far (dark red colour) means the 80-meter depth information from the viewpoint.

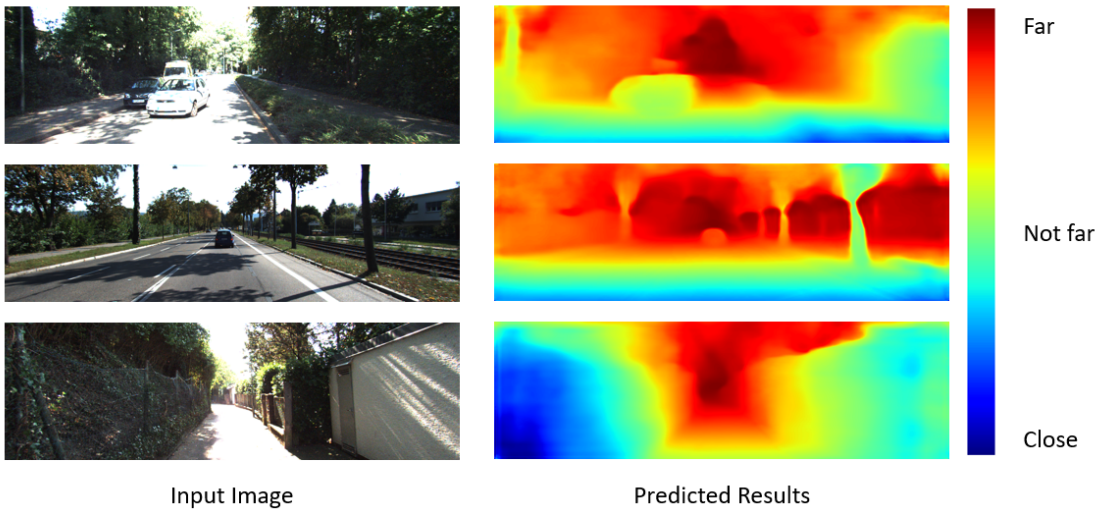

Figure 8.7: *Running Inference over Kitti Gradient Fix split Model*

The figure 8.8 shows us the accuracy of our model with comparison to the other researcher work. From the evaluation metrics, the RMSE value is 3.928 and in comparison to the other results, we are very close to the state of art results. The RMSE define how many per pixel in the image we are able to calculate the depth pixels. In this model evaluation, we are pretty much close to the baseline. In this model, we have used the gradient fix which is explained in chapter 7 section 7.4.

KITTI Split With Gradient Fix									
Method	Dataset	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	
Eigen et al. [48] Coarse	KITTI	0.214	1.605	6.563	0.292	0.673	0.884	0.957	
Eigen et al. [48] Fine	KITTI	0.203	1.548	6.307	0.282	0.702	0.89	0.958	
Liu et al [50]	KITTI	0.201	1.584	6.471	0.273	0.68	0.898	0.967	
Liana et al [5]	NYU	0.176		4.46	0.072				
Garg et al [6]	KITTI	0.169	1.08	5.104	0.273	0.74	0.904	0.962	Lower is better
Godard et al [7]	KITTI	0.141	1.369	5.849	0.242	0.818	0.929	0.966	
Godard et al [7] + CS	KITTI	0.136	1.512	5.763	0.236	0.836	0.935	0.968	Higher is better
Godard et al [7] + CS + Post-Processing	KITTI	0.126	1.161	5.381	0.224	0.843	0.941	0.972	
Kuznietsov et al [51]	KITTI	0.089	0.478	3.61	0.138	0.906	0.98	0.995	
Amiri et al [10]	KITTI	0.078	0.417	3.464	0.126	0.923	0.984	0.995	
Our Depth	KITTI	0.083	0.601	3.928	0.138	0.915	0.973	0.099	

Figure 8.8: *Results Comparison*

The figure 8.9 gives information about the learning rate. For the first 50000 iterations, the learning rate was kept steady at 1e-4 and after 50k iteration we changed the learning rate to be around 5e-4. The final learning rate was about 2e-4 which helped the model to learn important feature and bring stable depth information.

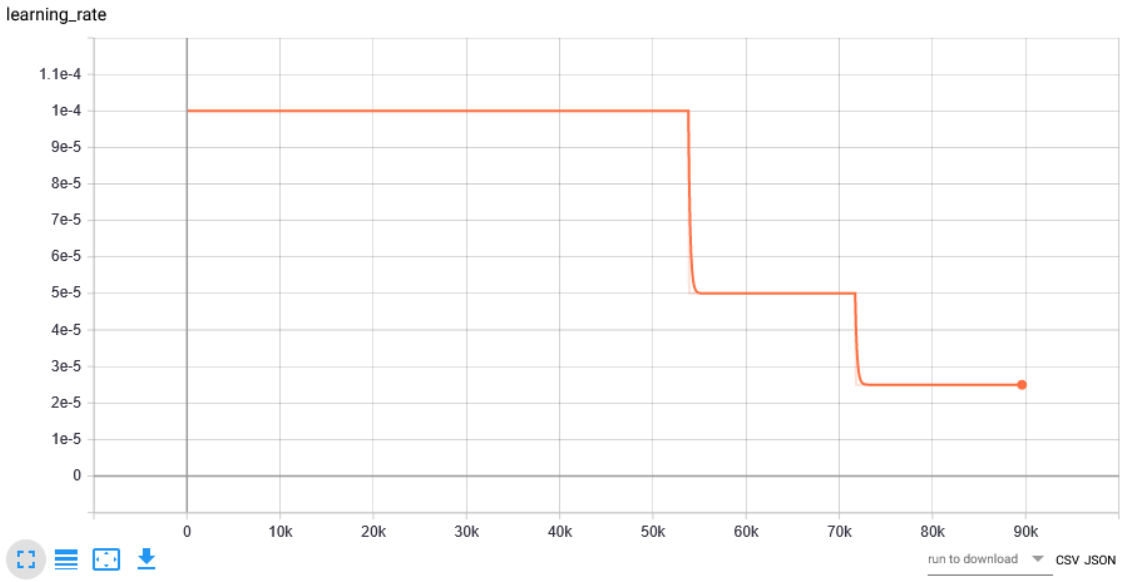

Figure 8.9: *Learning Rate*

Figure 8.10 gives information about the training loss. Using the tensorboard library we were able to see the total loss during the complete training time. The total loss is the combination of supervised, unsupervised, Left-Right consistency loss and smoothness which all discussed in chapter 7. The training loss in this figure 8.10 is pretty stable does not spikes randomly and that help to model to learn at steady pace.

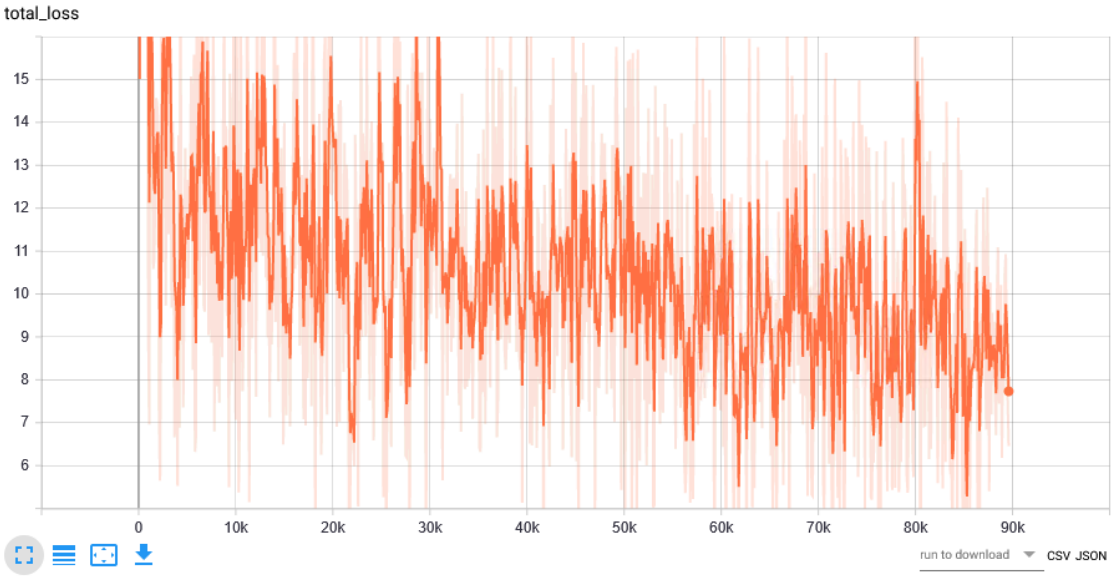

Figure 8.10: *Training Loss*

Figure 8.11 shows us the hyperparameters which we have used for the model. There is two hyperparameters one is the hyperparameter in general which decide the epochs, optimizer, learning rate and batch size etc. The other hyperparameter for the loss function they are the scalar quantity which changed and played to get more practical model. The hyperparameter was set with a batch size of 8. The total number of epochs we have 25 epochs and some scalar hyperparameters for the loss function is shown in figure 8.11 (b) part.

Description	Value
Optimizer	Adam optimizer
Learning rate	10^{-4}
β_1	0.9
B_2	0.999
ε	10^{-8}
Epoch	25
Batch size	8
Input image size	256*512
Inverse depth range	0 to 1.0 (Sigmoid function)

(a) General Hyperparameter

Description	Value
λ_1	1
λ_2	1.0
λ_3	150.0
λ_4	0.1
α_1	0.85
α_2	0.15
α_3	0.08

(b) Hyperparameter for loss function

Figure 8.11: *Hyper Parameters*

8.3 Eigen Split

Eigen split have 22200 left and 22200 right images for training taken from KITTI Dataset. The supervised part uses the LIDAR images which contains the sparse depth information for training. Figure 8.12 shows us the input image on the left-hand side and the predicted output on the right-hand side. The close (blue colour) in this figure 8.12 means zero meters from the viewpoint and the far (dark red colour) means the 80-meter depth

information from the viewpoint. In this split of training, we have chosen some different Hyperparameters. We have tried to play with the hyperparameter we have changed the batch size and number of epochs due to which training time was increased.

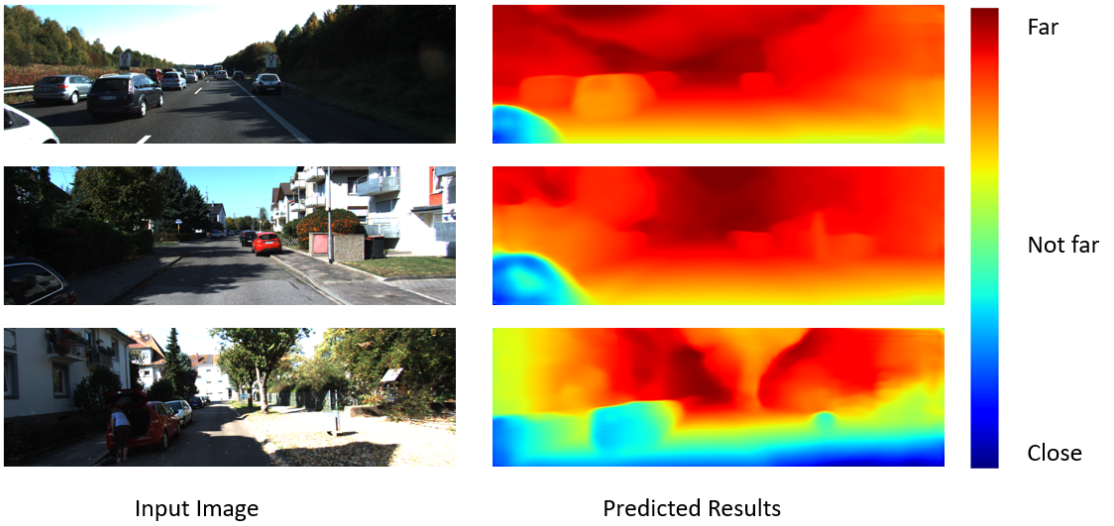

Figure 8.12: *Running Inference over Eigen Split Model*

Figure 8.13 shows us the accuracy of our model with comparison to the other researcher work. From the evaluation metrics, the RMSE value is 4.458 and in comparison to the other results, we are very close to the state of art results. The results from figure 8.13 the RMSE and RMSE log are slighter higher still the model are able to predict relative good depth maps.

Eigen Split									
Method	Dataset	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	
Eigen et al. [48] Coarse	KITTI	0.214	1.605	6.563	0.292	0.673	0.884	0.957	
Eigen et al. [48] Fine	KITTI	0.203	1.548	6.307	0.282	0.702	0.89	0.958	
Liu et al [50]	KITTI	0.201	1.584	6.471	0.273	0.68	0.898	0.967	
Liana et al [5]	NYU	0.176		4.46	0.072				
Garg et al [6]	KITTI	0.169	1.08	5.104	0.273	0.74	0.904	0.962	Lower is better
Godard et al [7]	KITTI	0.141	1.369	5.849	0.242	0.818	0.929	0.966	
Godard et al [7] + CS	KITTI	0.136	1.512	5.763	0.236	0.836	0.935	0.968	Higher is better
Godard et al [7] + CS + Post-Processing	KITTI	0.126	1.161	5.381	0.224	0.843	0.941	0.972	
Kuznietsov et al [51]	KITTI	0.089	0.478	3.61	0.138	0.906	0.98	0.995	
Amiri et al [10]	KITTI	0.078	0.417	3.464	0.126	0.923	0.984	0.995	
Our Depth	KITTI	0.105	0.744	4.458	0.165	0.879	0.968	0.989	

Figure 8.13: *Results Comparison*

Figure 8.14 gives information about the training loss. Using the tensorboard library we were able to see the total loss during the complete training time. The total loss is the combination of supervised, unsupervised, Left-Right consistency loss and smoothness which all discussed in chapter 7. The training loss in this figure 8.14 is pretty stable does not spike randomly and that help to model to learn at a steady pace.

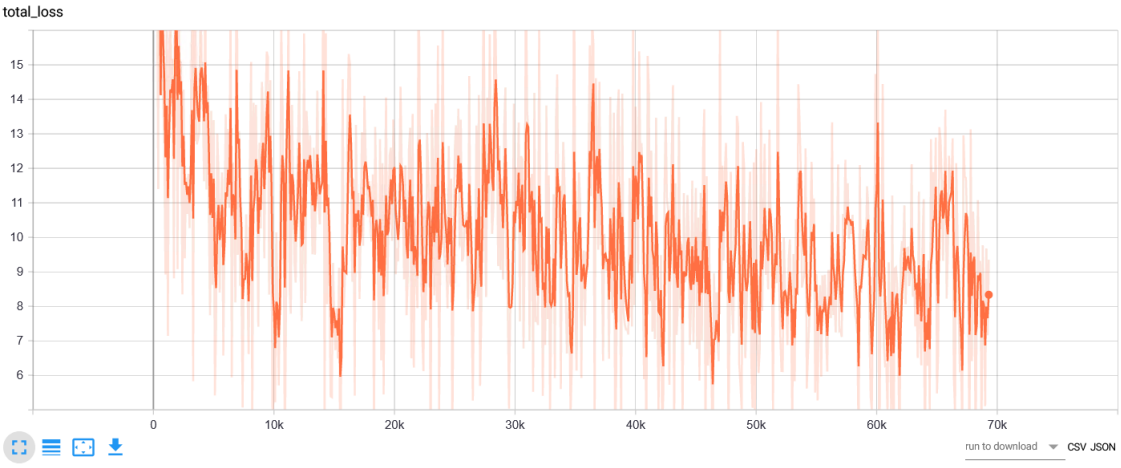

Figure 8.14: *Training Loss*

Figure 8.15 shows us the hyperparameters which we have used for the model. There are two hyperparameters one is the hyperparameter in general which decide the epochs, optimizer, learning rate and batch size etc. The other hyperparameter for the loss function they are the scalar quantity which changed and played to get more practical model. The hyperparameter was set with a batch size of 9 The total number of epochs we have 30 epochs and some scalar hyperparameters for the loss function is shown in figure 8.15 (b) part.

Description	Value
Optimizer	Adam optimizer
Learning rate	10^{-4}
β_1	0.9
B_2	0.999
ε	10^{-8}
Epoch	30
Batch size	9
Input image size	256*512
Inverse depth range	0 to 1.0 (Sigmoid function)

(a) General Hyperparameter

Description	Value
λ_1	1
λ_2	1.0
λ_3	150.0
λ_4	0.1
α_1	0.85
α_2	0.15
α_3	0.08

(b) Hyperparameter for loss function

Figure 8.15: *Hyper Parameters*

8.4 Eigen Gradient Fix

Figure 8.16 shows the results from the training which we did with the split which contains about 44400 depth and stereo RGB images for training. The main changes were made to hyperparameters the batch size was reduced to 7 for training purpose. In this split also we have also changed the number of epochs which was earlier set to 25 to 35. The results from the training are devastating. The correct judgment of the depth maps were completely wrong. As we can see in the figure 8.16 the red part shows us the near to the viewpoint and the blue show us the far away points but the result is not correct.

To fix the problem we have used the Sobel filter to avoid gradient problem but still, this training did not perform well.

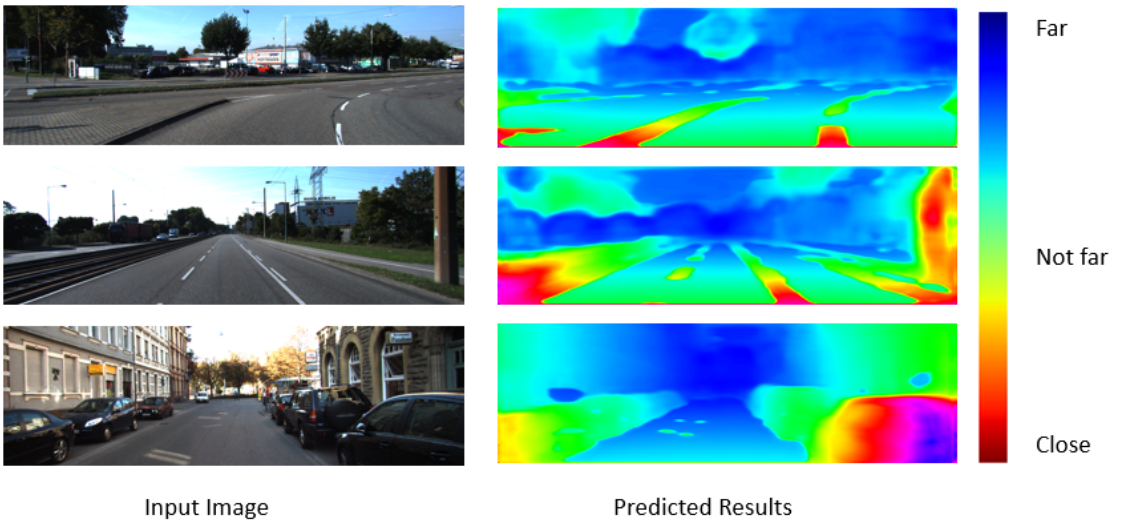

Figure 8.16: *Running Inference over Eigen Gradient Fix Split Model*

From the result, in figure 8.17 we can see the RMSE value of 4.97 which is very poor when comparing to the state of the art results. The model is trained in a semisupervised way when comparing to the ground truth the model does not bring the clear depth.

Eigen Split with Gradient Fix									
Method	Dataset	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	
Eigen et al. [48] Coarse	KITTI	0.214	1.605	6.563	0.292	0.673	0.884	0.957	
Eigen et al. [48] Fine	KITTI	0.203	1.548	6.307	0.282	0.702	0.89	0.958	
Liu et al [50]	KITTI	0.201	1.584	6.471	0.273	0.68	0.898	0.967	
Liana et al [5]	NYU	0.176		4.46	0.072				
Garg et al [6]	KITTI	0.169	1.08	5.104	0.273	0.74	0.904	0.962	Lower is better
Godard et al [7]	KITTI	0.141	1.369	5.849	0.242	0.818	0.929	0.966	
Godard et al [7] + CS	KITTI	0.136	1.512	5.763	0.236	0.836	0.935	0.968	Higher is better
Godard et al [7] + CS + Post-Processing	KITTI	0.126	1.161	5.381	0.224	0.843	0.941	0.972	
Kuznietsov et al [51]	KITTI	0.089	0.478	3.61	0.138	0.906	0.98	0.995	
Amiri et al [10]	KITTI	0.078	0.417	3.464	0.126	0.923	0.984	0.995	
Our Depth	KITTI	0.117	0.895	4.97	0.185	0.852	0.955	0.985	

Figure 8.17: *Results Comparison*

Figure 8.18 gives information about the training loss. Using the tensorboard library we were able to see the total loss during the complete training time. The total loss is the combination of supervised, unsupervised, Left-Right consistency loss and smoothness which all discussed in chapter 7. The training loss in this figure 8.18 is not stable. Spike randomly during the iteration from zero to its first 10000 iterations. This might lead to unstable training which leads to an unstable model. This concludes that the random spikes in the losses can lead to unstable training and an unstable model.

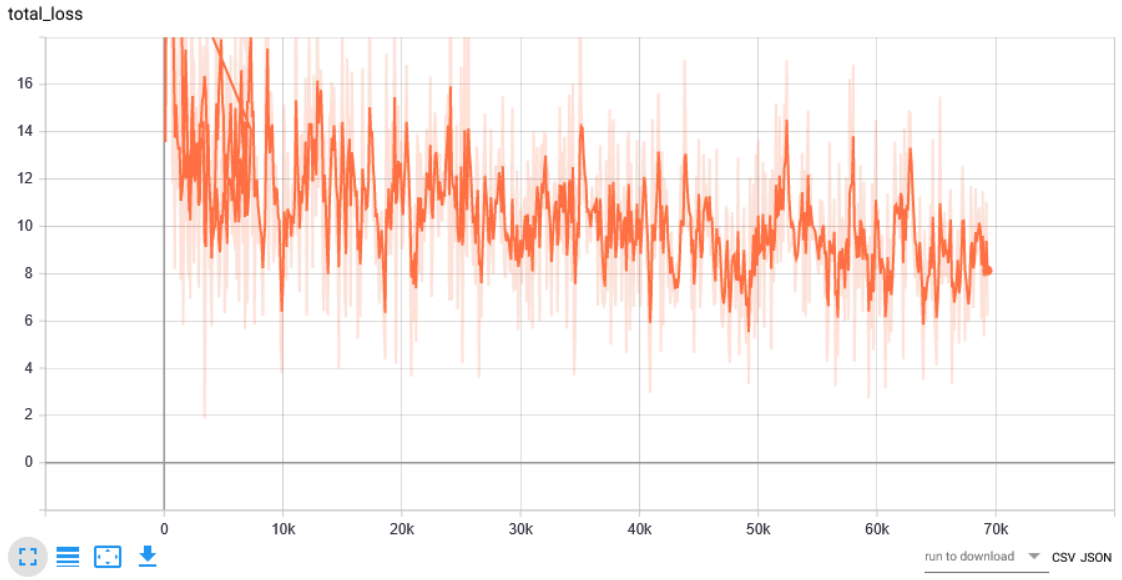

Figure 8.18: *Training Loss*

Figure 8.19 shows us the hyperparameters which we have used for the model. There are two hyperparameters one is the hyperparameter in general which decide the epochs, optimizer, learning rate and batch size etc. The other hyperparameter for the loss function they are the scalar quantity which changed and played to get more practical model. The hyperparameter was set with a batch size of 7 The total number of epochs we have 35 epochs and some scalar hyperparameters for the loss function is shown in figure 8.19 (b) part.

Description	Value
Optimizer	Adam optimizer
Learning rate	10^{-4}
β_1	0.9
B_2	0.999
ε	10^{-8}
Epoch	35
Batch size	7
Input image size	256*512
Inverse depth range	0 to 1.0 (Sigmoid function)

(a) General Hyperparameter

Description	Value
λ_1	1
λ_2	1.0
λ_3	150.0
λ_4	0.1
α_1	0.85
α_2	0.15
α_3	0.08

(b) Hyperparameter for loss function

Figure 8.19: *Hyper Parameters*

9 Conclusion and Future Scope

Depth estimation from an single image without having the knowledge of the environment is a challenging task. In this master thesis using deep learning platform, we are able to find a method that uses various losses to make a robust model for depth estimation. The model is able to generate the depth images only using a single image. The main challenging task was to generate depth each relatable pixel in the image. The main motivation was taken from the research proposed by [7]. In this research, the main challenging task was to find which architecture we should use. The second most challenging task was training the model which require to understand the dataset and split the dataset for training. During the evaluation, we have tested our model with the other results to check how the model will perform. For the evaluation, we have used various evaluation metrics like RMSE, RMSE log. we have trained the model with 4 different splits for each different split we performed the evaluation. The evaluation results for which we have used 697 images taken all from the KITTI dataset and compare to the ground truth depth. We have got the value of RMSE 3.928 and RMSE log 0.138 is pretty good when we compared with other researcher works.

In this master thesis, we conclude that semi-supervised learning for depth estimation is an intuitive way through which the model can learn. The model uses the left and right consistency loss with various other losses like reconstruction loss, smoothness loss which have used by involving the semisupervised learning approach. In this method, we have used the lidar data point taken from KITTI dataset as ground truth for training. The model has been tested and evaluated and compared to the state of the art results. The major work which I can relate to is proposed by [7][10].

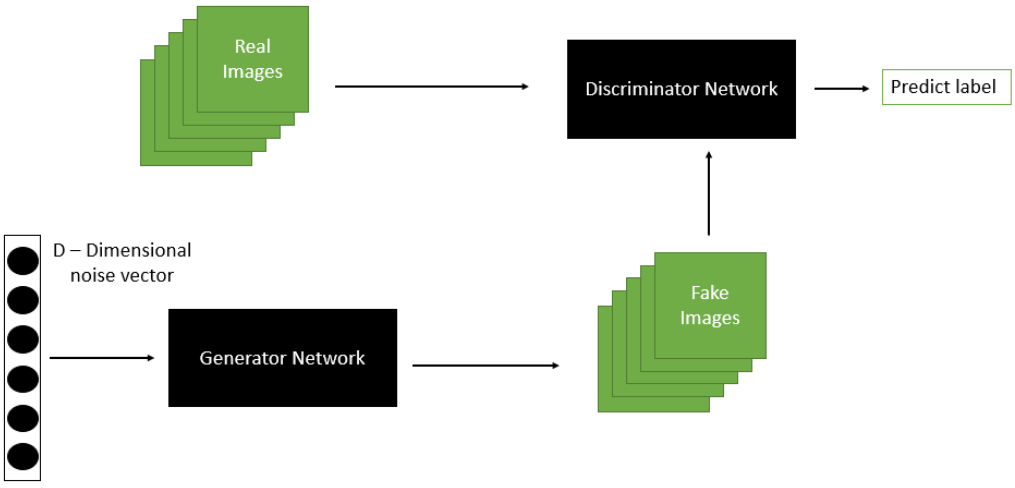

Figure 9.1: *Generative adversarial networks*

For the training purpose, the main problem is the training data. To avoid this problem we can use the Generative Adversarial Networks (GAN) which is the involvement of the synthetic data in the deep learning framework. In figure 9.1 we can see that the GAN is made up of the two networks which are named as generator network which generates the fake images and other is the discriminator network which approves the data. The main task is that the data which GAN creates are similar to the real world training data. Hence it increases the training data set through which model can learn better.

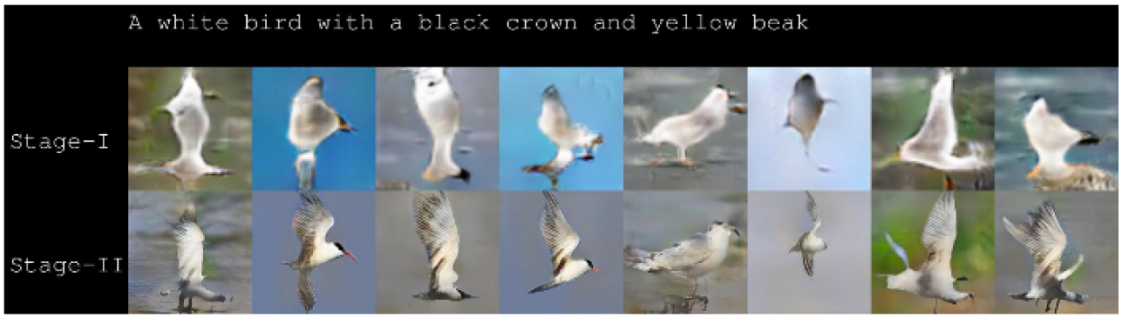

Figure 9.2: *Generative adversarial networks*

The figure 9.2 shows the results which are produced by the GAN at the stage one they produced images which look like the bird but the definition is not clear in the second stage of refinement they produce the results which look like a bird and are more clear. The GAN is the approach which is unique and can counter the problem of training data for a deep learning platform.

Bibliography

[1] Warner McGee. Depth output: Demystified foundry community. https://bit.ly/2mL4sHY. (Accessed on 3/09/2019).

[2] Keyur R. Ranipa and M. V. Joshi. A practical approach for depth estimation and image restoration using defocus cue. *IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6, 2011.

[3] Dineesh Mohan and A Ram. A Review on Depth Estimation for Computer Vision Applications. *Positions*, 4(11):235–239, 2015.

[4] Ronald van Loon. Machine learning explained: Understanding supervised, unsupervised. https://bit.ly/2BCHVyE. (Accessed on 23/09/2019).

[5] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. *Proceedings - 2016 4th International Conference on 3D Vision, 3DV 2016*, pages 239–248, 2016.

[6] Ravi Garg, B. G. Vijay Kumar, Gustavo Carneiro, and Ian Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9912 LNCS:740–756, 2016.

[7] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:6602–6611, 2017.

[8] Fabien Lotte. Principle of semi-supervised learning. https://bit.ly/2lebVz1. (Accessed on 09/09/2019).

[9] Nikolai Smolyanskiy, Alexey Kamenev, Jeffrey Smith, and Stan Birchfield. Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness. *IEEE International Conference on Intelligent Robots and Systems*, 2017-September:4241–4247, 2017.

[10] Ali Jahani Amiri, Shing Yan Loo, and Hong Zhang. Semi-Supervised Monocular Depth Estimation with Left-Right Consistency Using Deep Neural Network. 2019.

[11] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging Into Self-Supervised Monocular Depth Estimation. (1), 2018.

[12] Matan Goldman, Tal Hassner, and Shai Avidan. Learn Stereo, Infer Mono: Siamese Networks for Self-Supervised, Monocular, Depth Estimation. 2019.

[13] David Kriesel. A brief introduction on neural networks. 2007.

[14] Kuldeep Shiruru. An introduction to artificial neural network. *International Journal of Advance Research and Innovative Ideas in Education*, 1:27–30, 09 2016.

[15] Adam Geitgey. Machine learning is fun part 3: Deep learning and convolutional neural networks. `https://bit.ly/2J6coMt`, 06 2016. (Accessed on 18/09/2019).

[16] Dominique Gilleman. Convolutional network (cnn). `https://bit.ly/2Bp9RbU`. (Accessed on 09/08/2019).

[17] ujjwalkarn. An intuitive explanation of convolutional neural networks. `https://bit.ly/2scL7hg`, 08 2016. (Accessed on 25/08/2019).

[18] Andrej karpathy. Convolutional neural networks for visual recognition. `https://bit.ly/31vwXYQ`, 17 2018. (Accessed on 19/10/2019).

[19] Jeffrey A Bonnell. Implementation of a new sigmoid function in backpropagation neural networks. 2011.

[20] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[21] Sebastian. Raschka. Python-machine-learning-book. `https://bit.ly/2pB56cE`, 2015.

[22] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *arXiv preprint arXiv:1901.06032*, 2019.

[23] Muneeb ul Hassan. Vgg16 – convolutional network for classification and detection. `https://bit.ly/2KXZxwI`. (Accessed on 27/11/2019).

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[25] Lei He, Guanghui Wang, and Zhanyi Hu. Learning depth from single images with deep neural network embedding focal length. *IEEE Transactions on Image Processing*, 27(9):4676–4689, 2018.

[26] Jack Zhu and Ralph Ma. Real-time depth estimation from 2d images, 2016.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:770–778, 2016.

[28] Sasha Targ, Diogo Almeida, and Kevin Lyman. Resnet in Resnet: Generalizing Residual Architectures. pages 1–7, 2016.

[29] Koustubh Sinhal. Resnet, alexnet, vggnet, inception: Understanding various architectures of convolutional networks. `https://bit.ly/37KTJQS`. (Accessed on 29/11/2019).

[30] Kourosh Khoshelham. Accuracy analysis of kinect depth data. In *ISPRS workshop laser scanning*, volume 38, 2011.

[31] JM Martínez Montiel, Javier Civera, and Andrew J Davison. Unified inverse depth parametrization for monocular slam. Robotics: Science and Systems, 2006.

[32] Javier Civera, Andrew J Davison, and JM Martinez Montiel. Inverse depth parametrization for monocular slam. *IEEE transactions on robotics*, 24(5):932–945, 2008.

[33] Fangchang Ma. *Algorithms for single-view depth image estimation*. PhD thesis, Massachusetts Institute of Technology, 2019.

[34] Yanchao Yang, Alex Wong, and Stefano Soatto. Dense depth posterior (ddp) from single image and sparse range. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3353–3362, 2019.

[35] Matthew. mj 4 ict: Color depth. `https://bit.ly/39qhlLj`, April 2017. (Accessed on 10/24/2019).

[36] Dhruv Mahajan I. Zeki Yalniz, Hervé Jégou. Billion-scale semi-supervised learning for state-of-the-art image and video classification. `https://ai.facebook.com/blog/billion-scale-semi-supervised-learning/`, October 2019. (Accessed on 12/09/2019).

[37] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research. *The International Journal of Robotics Research*, (October):1–6, 2013.

[38] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial transformer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2017–2025. Curran Associates, Inc., 2015.

[39] Vitor F. Pamplona, Manuel M. Oliveira, and Gladimir V.G. Baranoski. Photorealistic models for pupil light reflex and iridal pattern deformation. *ACM Transactions on Graphics*, 28(4), 2009.

[40] Jonathan T. Barron, Andrew Adams, Yichang Shih, and Carlos Hernández. Fast bilateral-space stereo for synthetic defocus - Supplemental Material. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:4466–4474, 2015.

[41] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[42] Andrew Fitzgibbon and Toby Sharp. Real-Time Human Pose Recognition in Parts from Single Depth Images: Supplementary Material Jamie Shotton Andrew Fitzgibbon Mat Cook Tob. *Communications of the Acm*, 2:1297–1304, 2013.

[43] Danail Stoyanov, Marco Visentini Scarzanella, Philip Pratt, and Guang-Zhong Yang. Real-time stereo reconstruction in robotically assisted minimally invasive surgery. In Tianzi Jiang, Nassir Navab, Josien P. W. Pluim, and Max A. Viergever, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2010*, pages 275–282, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[44] Junyuan Xie, Ross Girshick, and Ali Farhadi. Deep3D: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9908 LNCS:842–857, 2016.

[45] Saleh. S, Khwandah. S, Heller. A, Mumtaz. A, and Hardt. W. (2019). Traffic Signs Recognition and Distance Estimation using a Monocular Camera.In: 6th International Conference Actual Problems of System and Software Engineering. [online] Moscow: IEEE, pp.407-418. Available at: http://ceur-ws.org/vol-2514/.

[46] Saleh. S, Hadi. S, Amin Nazari. M, and Hardt. W. (2019). Outdoor Navigation for Visually Impaired based on Deep Learning. In: 6th International Conference Actual Problems of System and Software Engineering. [online] Moscow: IEEE, pp.397-406. Available at: http://ceur-ws.org/vol-2514/.

[47] Saleh. S. M, A. Khwandah. S, Hardt. W, Hilbrich. M, and Lazaridis. P. I. "Estimating the 2D Static Map Based on Moving Stereo Camera," 2018 24th International Conference on Automation and Computing (ICAC), Newcastle upon Tyne, United Kingdom, 2018, pp. 1-5, doi: 10.23919/IConAC.2018.8749004.

[48] Patrick P.K. Chan, Bing Zhong Jing, Wing W.Y. Ng, and Daniel S. Yeung. Depth estimation from a single image using defocus cues. *Proceedings - International Conference on Machine Learning and Cybernetics*, 4:1732–1738, 2011.

[49] Andreas Wedel, Uwe Franke, Jens Klappstein, Thomas Brox, and Daniel Cremers. Realtime depth estimation and obstacle detection from monocular video. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4174 LNCS:475–484, 2006.

[50] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. 3-D Depth Reconstruction from a Single Still Image. (June), 2007.

[51] Eigen. Eigen 2014. *Nips*, pages 1–9, 2014.

[52] Bo Li, Chunhua Shen, Yuchao Dai, Anton Van Den Hengel, and Mingyi He. Li_Depth_and_Surface_2015_CVPR_paper.

[53] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:5162–5170, 2015.

[54] Yevhen Kuznietsov, Jörg Stückler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:2215–2223, 2017.

[55] Andrew Zisserman. Microsoft powerpoint self_supervision3.pptx. `https://bit.ly/2KSEht2`. (Accessed on 26/09/2019).

[56] Suzana Herculano-Houzel. The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in human neuroscience*, 3:31, 2009.

[57] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[58] Daniel Hugo. A practical approach to convolutional neural networks. `https://bit.ly/2MZrjJe`, 2019. (Accessed on 10/09/2019).

[59] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[60] Mirza Cilimkovic. Neural networks and back propagation algorithm. *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin*, 15, 2015.

[61] Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys*, 27(3):326–327, 1995.

[62] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[63] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[64] O. Chapelle, B. Scholkopf, and A. Zien, Eds. Semi-supervised learning (chapelle, o. et al., eds.; 2006) [book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, March 2009.

[65] Max Welling. Fisher linear discriminant analysis. department of computer science, university of toronto. Technical report, Technical Report, 2005.

[66] google developers. Descending into ml: Training and loss — machine learning crash course. `https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss`. (Accessed on 12/16/2019).

[67] Simon Meister, Junhwa Hur, and Stefan Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[68] Jonathan T Barron. A general and adaptive robust loss function. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4331–4339, 2019.

[69] Fridtjof Stein. Efficient computation of optical flow using the census transform. In Carl Edward Rasmussen, Heinrich H. Bülthoff, Bernhard Schölkopf, and Martin A. Giese, editors, *Pattern Recognition*, pages 79–86, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[70] Oliver Demetz, David Hafner, and Joachim Weickert. The complete rank transform: A tool for accurate and morphologically invariant matching of structures. In *BMVC*, 2013.

[71] Ramesh Jain, Rangachar Kasturi, and Brian G Schunck. *Machine vision*, volume 5. McGraw-Hill New York, 1995.