Convolutional Neural Network Based Monocular Depth Estimation

Author
Iqra Suleman
Fall-2017 MS (CE) 00000203984

Supervisor
Dr. Muhammad Usman Akram

DEPARTMENT OF COMPUTER ENGINEERING
COLLEGE OF ELECTRICAL MECHANICAL ENGINEERING(CEME)
National University of Sciences and Technology (NUST)
Islamabad, Pakistan

April 2020

Convolutional Neural Network Based Monocular Depth Estimation

Author
Iqra Suleman
Fall-2017 MS (CE) 00000203984

A thesis submitted in conformity with the requirements for
the degree of *MS Computer Engineering*
Thesis supervisor
Dr. Muhammad Usman Akram

Thesis Supervisor's signature: _____

DEPARTMENT OF COMPUTER ENGINEERING
COLLEGE OF ELECTRICAL MECHANICAL
ENGINEERING(CEME)
National University of Sciences and Technology (NUST)
Islamabad, Pakistan

April 2020

Declaration

I, *Iqra Suleman* declare that this thesis titled "Convolutional Neural Network Based Monocular Depth Estimation" and the work presented in it are my own and has been generated by me as a result of my own original research. The material that has been used from other sources, it has been properly acknowledged / referred.

_____

Signature of student

Iqra Suleman

Fall-2017 MS (CE) 00000203984

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for Plagiarism. Turnit in report endorsed by Supervisor is attached.

Signature of Student
Iqra Suleman
MS (CE) i00000203984

Signature of Supervisor
Dr. Muhammad Usman Akram

Language Correctness Certificate

This thesis has been read by an English expert and is free of most typing, syntax, semantic, grammatical and spelling mistakes. Thesis is also according to the format given by the university.

Signature of Student

Iqra Suleman
Fall-2017 MS (CE)
00000203984

Signature of Supervisor

Dr. Muhammad Usman Akram

Copyright Notice

- Copyright in text of this thesis rests with the student author. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the author and lodged in the Library of NUST College of EME. Details may be obtained by the Librarian. This page must form part of any such copies made. Further copies (by any process) may not be made without the permission (in writing) of the author.

- The ownership of any intellectual property rights which may be described in this thesis is vested in NUST College of EME, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the College of EME, which will prescribe the terms and conditions of any such agreement.

- Further information on the conditions under which disclosures and exploitation may take place is available from the Library of NUST College of EME, Rawalpindi.

Acknowledgements

All praise and glory to Almighty Allah (the most glorified, the highest) who gave me the courage, patience, knowledge and ability to carry out this work and to persevere and complete it satisfactorily. Undoubtedly, HE eased my way and without HIS blessings I can achieve nothing.

I would like to express my sincere gratitude to my advisor Dr. Muhammad Usman Akram for boosting my morale and for his continual assistance, motivation, dedication and invaluable guidance in my quest for knowledge. I am blessed to have such a co-operative advisor and kind mentor for my research.

Along with my advisor, I would like to acknowledge my entire thesis committee: Dr. Sajid Gul Khawaja, Dr. Arslan Shaukat from CEME and Dr. Usman Qayyum from NESCOM for their cooperation and prudent suggestions.

My acknowledgement would be incomplete without thanking the biggest source of my strength, my family. I am profusely thankful to my beloved parents who raised me when I was not capable of walking and continued to support me throughout every phase of my life and my loving siblings who were with me through my thick and thin.

I owe thanks to a very special person, my husband, for his continued and unfailing love, support and understanding during my pursuit of Master's degree that made the completion of thesis possible. I greatly value his contribution and deeply appreciate his belief in me. I appreciate my baby, my little boy for abiding my

ignorance and the patience he showed during my thesis writing. Words would never say how grateful I am to both of you.

This research work was funded by The National Engineering and Scientific Commission (NESCOM) under Agreement No.00xx-RAC-IV/UNIV/2017.

Finally, I would like to express my gratitude to all my friends and the individuals who have encouraged and supported me through this entire period. This thesis is dedicated to my beloved parents.

Abstract

For a long time, stereo cameras have been deployed in visual simultaneous location and mapping (SLAM) systems to obtain 3D information. Although stereo cameras show good performance, the main drawback is the complex and expensive hardware setup it requires, which restricts the use of the system. Monocular cameras are a simpler and cheaper alternative. Recent work has shown that access to depth maps in the monocular system is beneficial as they can be use d to improve 3D reconstruction. This work proposes a deep neural network that pr edicts dense high-resolution depth maps from monocular RGB images. Network architecture follows an encoder-decoder structure in which multi-scale information is captured and skip-connections are used to retrieve details. The network is trained and evaluated on a NYU v2 dataset with results comparable to state-of - the-art methods. The problem of depth estimation is an important component for understanding the geometry of a scene and for navigating through space. More understanding of the environment, such as recognition activities, contributes to changes in other fields.

In many applications, accurate measurement of depth from images is a crucial task involving interpretation and restoration of the scene. Existing methods for calculating depth often yield fuzzy approximations with low resolution. This thesis describes a convolution neural network to use transfer learning to compute a high resolution map of depth with one single RGB image. Using a typical encoder-decoder model, when initializing our encoder, we exploit features derived using high performance pre-trained networks along with extension and testing techniques that result in more accurate results.

We demonstrate how our approach can achieve accurate, high-resolution depth maps, even for a very simple decoder. We train dataset on three models i.e. densenet 169, 201 and Resnet50. Our network conducts results with state of the art on two datasets with less parameters and training iterations, and also offers qualitatively better tests that reflect human boundaries more accurately. Our algorithm gives state of the art results and it gives rms value of 0.4611

Keywords: *depth estimation*, *stereo*, *monocular*, *cameras*, *convolutional neural networks*, *NYU Depth v2*.

Contents

List of Figures

List of Tables

Chapter 1: Introduction

In many applications, including scene recognition and reconstruction[[1], [2], [3]], depth estimation from 2D images is a fundamental task. In applications such as navigation and scene recognition, virtual reality[1], image refocusing[2], and seg mentation[3], providing a complex depth map of the real-world can be very useful. Recent depth estimation advances concentrate on the u se of convolutional neural networks (CNNs) to perform 2D to 3D reconstruction. While the performance of these methods has steadily increased, the quality and r esolution of these estimated depth maps still present major problems. Recent imp lementations in augmented reality, virtual depth-of-field, and other image effects [[4], [5], [6]] allow a quick calculation of high-resolution 3D reconstructions to be effective. It is important for such applications to faithfully recreate discontinuity in depth maps and avoid large disturbances th at are often present in depth estimates calculated using current CNNs.

It is important for such applications to faithfully recreate discontinuity in depth m aps and avoid.Large disturbances that are often present in depth estimates calc ulat ed using current CNNs.

Estimating depth using monocular images is important in many practical scenari os such as images on social media, and also challenging as no image corresponde nces available. Some detail clues, such as perspective and object size, can be used from monocular images, and CNN is the ideal tool to use this information.

So, how does the predicted scene measure distance and understand our 3D envir onment? The mechanism at work here is that our brain begins to reason

about the incoming visual signals by identifying patterns such as the scale, shape, and motion of the **Depth Cues** scene. The image does not contain distance information, but in some way we could easily interpret and recover depth data. We see the near and farther distance of the scene. Such indications also allow us to view objects and surfaces supposedly in 3D shape on flat images [7].

1.1 Why is it so difficult to measure depth?

Finally, let's try to grasp some of the basic difficulties of estimating depth. The key culprit is to project 3D views into 2D images. Another question is rooted deep when there are objects moving and in motion. Depth Estimation is ill-posed problem. Sometimes, when performing work in monocular depth estimation, several authors point out that the problem of estimating depth from a single RGB image is an ill-posed inverse problem. What this implies is that many of the 3D scenes seen in the world will actually refer to the same 2D plane (figs 1.1).

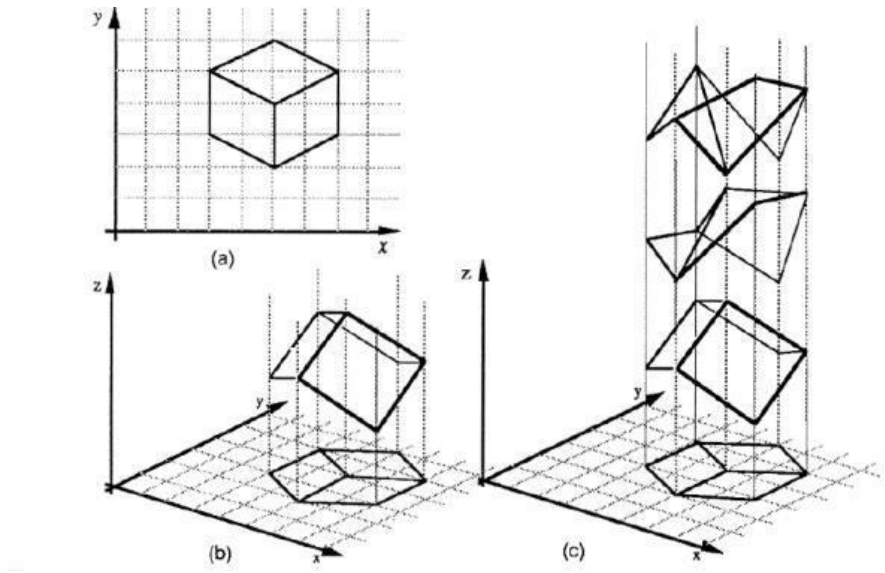

Figure 1.1 Vision has to solve an ill posed problem

1.2. Motivation

The overwhelming performance of computer vision's deep neural networks has prompted researchers to look into the issue of estimating depth from an image. Due to the increasing level of automation and artificial intelligence, precise computer vision and image representation is also required. The most important element in computer vision is depth estimation.. Being able to predict the depth maps of monocular images in a systematic manner would make these systems more reliable, easier and cheaper. In addition, this study provides an hint of the possibility of replacing monocular cameras with cheaper, lighter and lower-power cameras. In other applications, such as robotic navigation, 3D-modeling [8] and virtual reality systems , accurate depth estimation is also a key function.

1.3. Problem Statement

To explore the performance of single image based depth estimation with three network architectures i.e. Resnet50, densenet169 and densenet201 by giving singl e RGB image as an input which gives depth map as output and evaluate the perf ormances on different evaluation metrics.

1.6. Aim

This master thesis work focuses on the implementation, training and evaluation of a deep convolutional neural network that predicts depth from RGB monocular images. This work explores the possibility of modifying a state-of - the-art network in order to improve its performance. The system implemented is measured quantitatively by acceptable error parameters, but also qualitatively by visual inspection. The system is trained and tested in a public NYU Depth v2 and KITTI depth data set.

1.5. Objectives

Major objectives of the research are as follow:

1. To accurately estimate depth information.
2. Precision for the boundaries of closer objects.
3. To create an accurate algorithm of the depth estimation.
4. To set a milestone for Future's Specific Applications.

1.6. Structure of Thesis

This work is structured as follows:

Chapter2 covers the basics of depth estimation and gives review of the literature and the significant work done by researchers in past few years.
Chapter3.Explains dataset.

Chapter4 consists of the proposed methodology in detail.

Chapte5 includes all the experimental results accompanied by relevant figures.

Chapter6 Discuss potential directions for future work, and concludes the thesis.

Chapter 2: Literature Review

2.1 Depth Estimation

Depth estimation is a vision task for the computer to determine 2D image depth. The function requires an RGB image input and outputs an image of depth. The deep image contains details on the distance from the perspective of the objects that are normally the camera capturing the image. The following two photos clear ly illustrate the depth evaluation in practice.

Figure 2.1 Input image and depth estimation at output

2.2 Depth Is Important For 3D Vision

Knowing how much objects are connected to a camera remains challenging but completely necessary for exciting applications like robots, autonomous driving, 3D reconstruction of the scene and AR. Depth is a crucial prerequisite in robotics for performing multiple tasks such as perception, navigation, and planning. If we want to create a 3D diagram, the computing depth enables us to back up project

images taken in 3D from multiple views. Then all points can be completely reorganized the scenne by registration and matching.

2.3 Depth Estimation Methods

Depth is derived from 2 common methodologies in computer vision. Namely, **the depth of the monocular images** (static or sequential) **or the depth of the stereo images** by using epipolar geometry.

2.4 Depth Estimation in Computer Vision

The aim of depth estimation is to obtain a representation of the spatial structure of the scene, to restore the 3D shape and position of the objects in the image also known as the inverse problem [9], where we are trying to retrieve certain unknow ns , due to the lack of knowledge required to completely determine the solution. A dding that mapping between the 2D view and the 3D view is not special (Fig. 1.1)

And how do computers really feel the depth? Could we somehow pass any of the ideas mentioned above? The early algorithm with an impressive result starts with an estimate of depth using stereo vision back in the 1990s. Much progress has been made on complex stereo correspondence algorithms [10-12]. Scientists have been able to use geometry to constrain and reproduce the concept of stereopsis both mathematically and in real-time.

As far as monocular depth estimation is concerned, it has recently begun to gain p opularity by using neural networks to learn a representation that explicitly distils depth [13]. In addition, great progress has been made in self-monitored depth estimation [14-16].

2.5 Convolutional Neural Network (CNNs)

Neural network convolution (CNNs) run in one or more layers of a convolutional network. The three key components of CNN are convolutions, activation function and pooling [17], and the CNN typically includes several layers involving these operations. CNNs are also especially well suited for solving image issues [18]. In s ection 2.2.1-2.2.3, the main components are described. CNNs also require two otherforms of batch normalization and drop-out operations, see section 2.3.3. These perations allow the network to acquire the important features needed to overcome the task during the training process [19].

2.5.1 Convolutional Layers

The input, which is often an image, is converted into a filter sliding over the full photo in the convolutional layer. This connects each neuron in the hidden layer to the various neuron regions in the previous layer. The region is defined by the filter size and is referred to as the receptive field [19]. Stride [18] is the number of pixels that the filter moves during each new step. Figure 2.3 provides an example of a normal convolution. As it captures the image features or patterns, the product of the convolution is called an image map. For example, a feature may be a horizontal edge or line but a more complex structure as well. The weights and biases of the filters are learned during training and determine the type of features to be extracted.

All neurons in the network, which is called parameter sharing, are associated with the same weights and biases. This allows the same filter to detect the same feature at the same position throughout the entire image. This is because features useful in one area are likely to be useful in other areas of the picture [18].

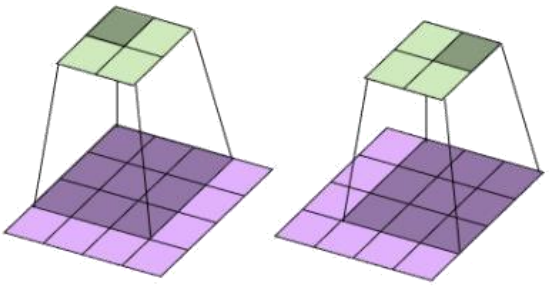

Figure 2.2 Use stride 1 and no zero-padding to convolve a 3x3 kernel over a 4x4 input. The output is smaller than the input if zero-padding is used but an output of the same size as the input can be generated.

2.5.2 Activation Function

There are different kinds of activation functions and ReLU [18] is a popular option the most widely used in CNNs today. Such functions are mentioned in Figure 2.3 and outlined in the following sections.

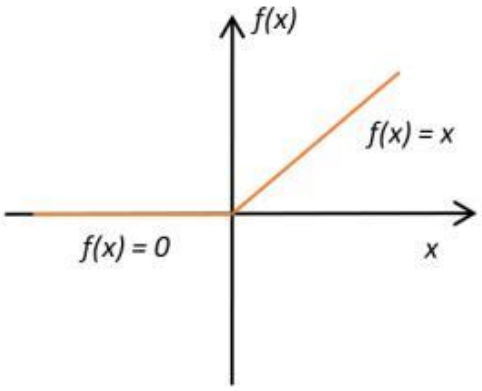

Figure 2.3 Activation function ReLU. Here,x is the input and the factor a is a learnable parameter that is modified during the training cycle

2.5.3 Pooling Layers

Usually a pooling layer is added after the feature map is sent through the activation function to reduce its spatial size. This reduces the amount of parameters that the

network requires to know and thus increases network computational efficiency [17]. It also improves the receptive field and makes the network invariant to limited input translations. There are no learnable parameters for the pooling layers but rather a fixed function is implemented separately on each of the feature maps [19]. A small filter, typically 2x2 or 3x3, slides over the feature map during this operation, and returns a summary statistics of the nearby values. The most popular choices are max-pooling and average pooling as shown in figure 2.4.

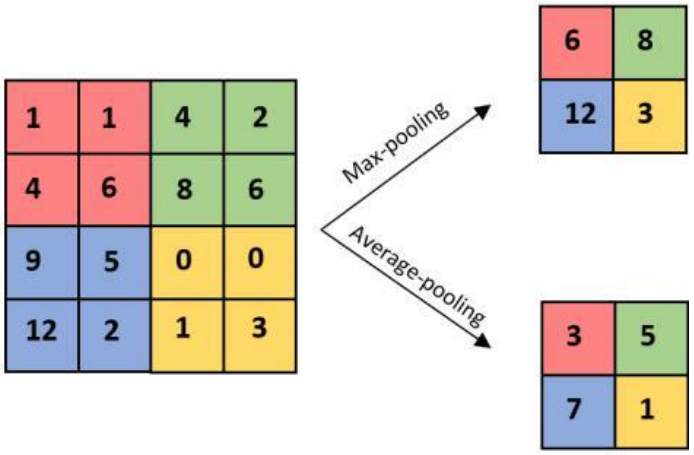

Figure 2.4: Top-and average pooling using a 2x2 size filter over a 4x4 feature map with stride 2 and no zero-padding. The effect is an output of 2x2 in size.

Max-pooling requires taking maximum values within the area of the filter.

Since the same feature is identified in the entire feature map, returning the maximum value in a given region provides an indication of whether or not the feature is present in that area. In this way the network is invariant to small input image translations, since the maximum value is always extracted within a small area. That also means that the feature's exact spatial position is lost, however. Ave rage-pooling is similar to max-pooling, but returns the sum of all values within the filter area instead.

Because only one value is returned for a area containing multiple values, the spatial size of the function map is reduced. How much is that, i.e. the down sampling

factor, is calculated by the size of the filter and the stride. You can use zero padding to get the desired down sampling factor.

The below section presents current solutions to the challenge of estimating depth using neural networks in convolution.

2.6 Related Work

A number of researchers have already work on depth estimation. We look at some of the works from RGB reference images that relate to depth estimation and 3D reconstruction problem. We are looking, in particular, at recent approaches that depend on deep neural networks.

Laina et al. [20] proposes a completely convolutional architecture to address the problem of estimating the depth map of a scene given an RGB picture. Residual learning models the ambiguous mapping between monocular images and depth maps. For optimization, the reverse Huber loss is used. The model works on images or videos in real time.

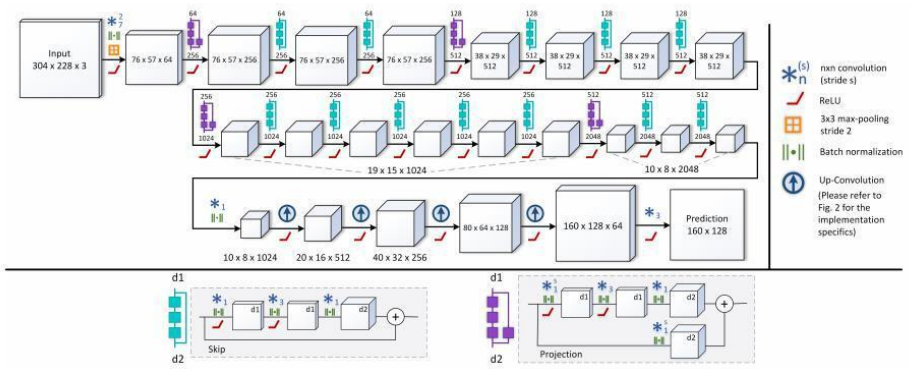

Figure.2.5. Laina et al. Network Architecture. [20]

Laina et al. [20] compare various network architectures: AlexNet, VGG-16, and ResNet-50. Due to its limited area of reception at the

last convolutional layer, AlexNet is outperformed by all other networks and does not collect enough global details. With theVGG-16, the receptive field is greater, the depth prediction is better, while ResNet-50 gives the highest output for all error measurements.

The results of this work illustrate how well up-projections are used in the decoder to get high resolution complex output predictions. It also demonstrates how ResNet can be used as the extractor for features rather than VGG or AlexNet

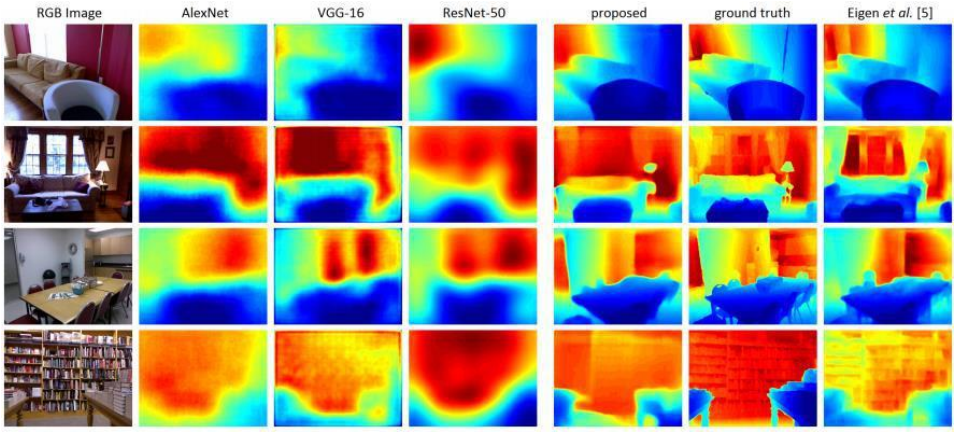

Figure 2.6 depth prediction on NYU dataset by Laina et al. [20]

Godard et al. [16] suggested an unsupervised monocular depth measurement method without annotated ground truth accuracy and allowed the network to train with left-
right consistency checks on calibrated stereo images. They discussed different net work configurations in their other work [21], and exchanged weights between net work depth and pose estimation.

Godard et al. [16] uses a loss of reconstruction with a left - right consistency. They show that only taking into account the loss of reconstruction can

provide a good image of reconstruction, but result in poor depth predictions. Using epipolar geometry constrains, their network can simultaneously generate disparity maps for both images (left to right and right to left) using only the left i mage. The predictions of depth become more reliable and precise by applying consistency between the disparity maps during training.

This research demonstrates how a network can be equipped for depth estimation using only pairs of stereo images in an unsupervised manner

Figure 2.7. Depth estimation by Godard. [16]

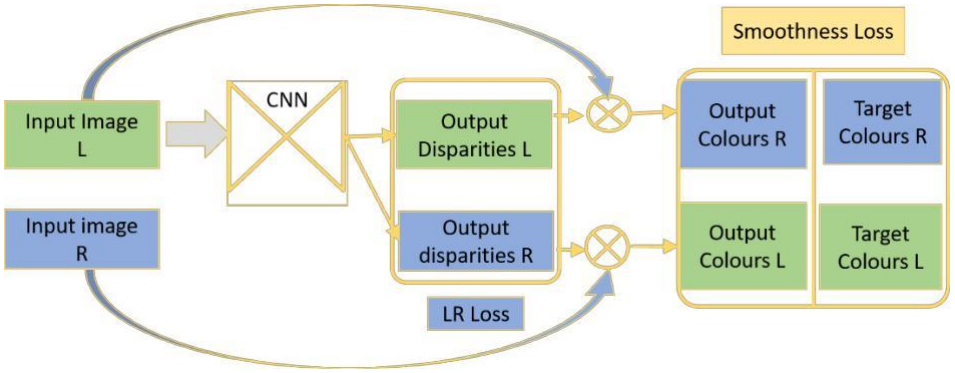

Figure2.8. Godard et al. [16] Architecture.

Lee [22] presented a method on unsupervised depth estimation. They utilize a dual CNN based model to produce the corresponding disparity

map. The models proposed are tested through KITTI driving and urban software Cityscapes. They evaluated their model on measurement criteria these include Absolute Relative Differences (Abs Rel), Squared Relative Differences (Sq Rel), RMSE log and d1-all.

They suggest a novel monocular depth estimation algorithm using relative depth maps. Next, they estimate relative depths between pairs of regions, as well as ordinary depths, at different scales using a convolutional neural network. Secondly, they restore relative depth maps based on the rank-1 property of pairwise comparison matrices from selectively estimated results. Third, they break down ordinary and relative maps of depth into components, and optimally recombine them to recreate a final map of depth.

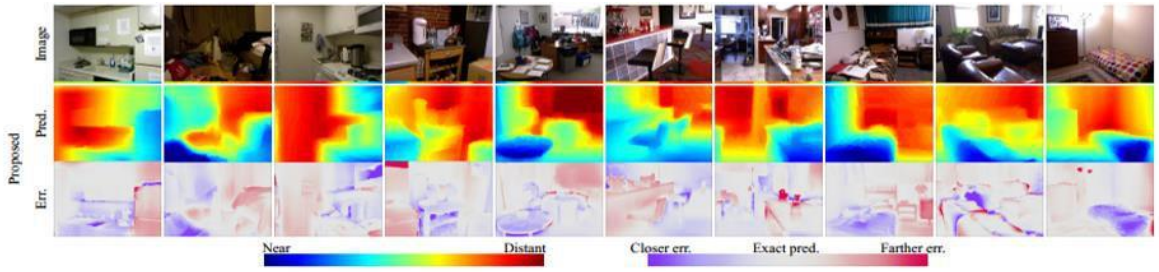

Figure.2.9. Lee [22] predicted depth & error maps

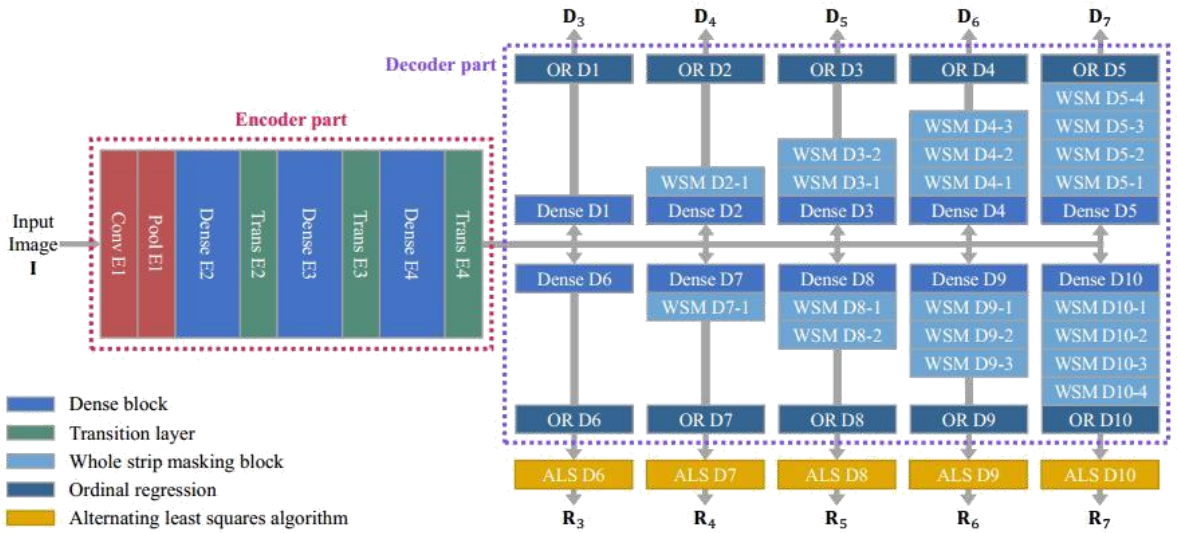

Figure 2.10. Lee [22] proposed depth network

Xu, D., et al. [23] introduced a new approach for the monocular depth measurement. Their approach utilizes a continuous CRF to combine multi-scale information from different layers of a Convolutionary Neural Network (CNN) front-end. On the KITTI benchmark and NYU Depth V2 dataset, they tested their methods. We measure relative mean error (rel), root mean squared error (rms), mean error log10 (log10), and precision.

They are proposing a new deep learning model for calculating depth maps from still images that seamlessly combine front-end CNN and multi-scale CRF, our architecture does not only consider prediction maps as inputs but operates directly at feature level. Our approach benefits from a new attention process that allows multi-scale features to be robustly fused and organized information incorporated. Our approach demonstrates state-of - the-art results on the NYU Depth V2 dataset [24] and is among the top performers on the KITTI benchmark's more demanding outdoor scenes [25].

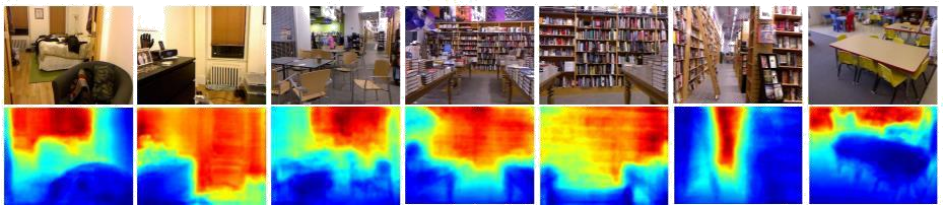

Figure 2.11 Xu, D et al [23] predicted depth maps.

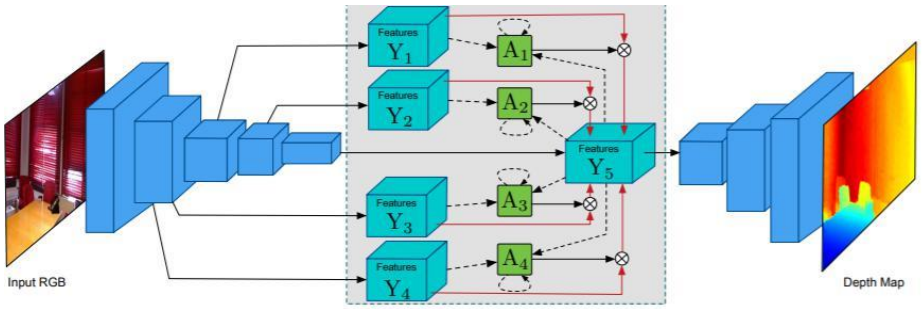

Figure 2.12 Xu, D et al [23] proposed network for monocular depth estimation

Liu et al. [26] presented a profoundly convolutional neural field model for depth estimation of a single image, testing their algorithm on two popular online datasets: the NYU v2 dataset and the Make3D dataset. Standard metrics used for quantitative assessments are average relative error (rel), root mean squared error (rms), average log10 error (log), and accuracy.

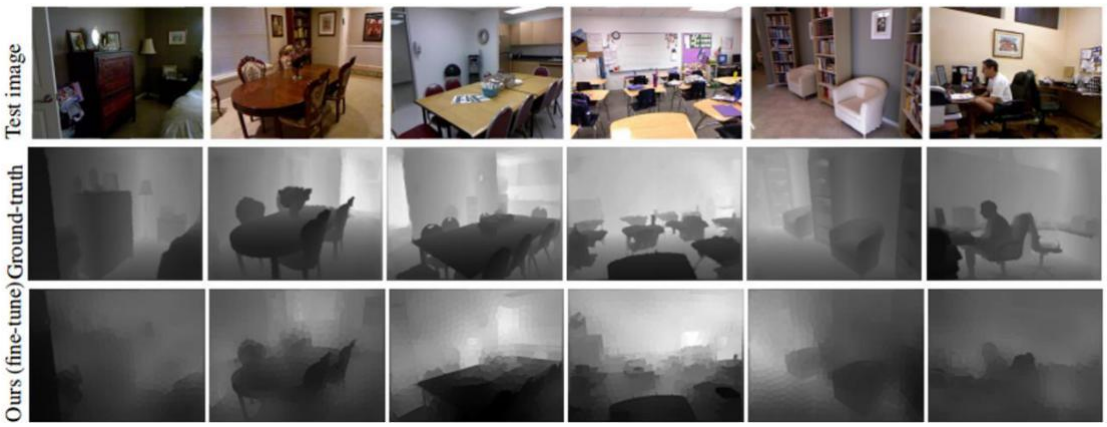

Figure 2.13 Liu et al [26] prediction on NYU v2 dataset

Interpretation of model by liu et al. [26] The input image is first over-segmented into superpixels, then crop the picture patch based around its centroid for a superpixel p, then resize and feed it to a CNN (5 convolution layers and 4 completely linked layers). Consider K types of similarities, fed into a fully-connected layer for a pair of neighboring superpixels (p, q). The outputs are then fed to the loss layer structured by CRF, which minimizes the probability of negative logs.

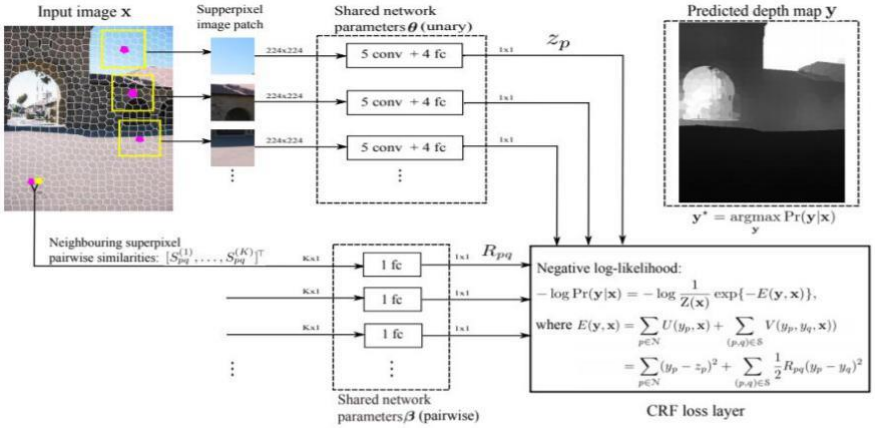

Figure.2.14. Liu et al. [26] model for depth estimation.

Zhou et al. [15] Using single depth view and multiview pose networks, their method evaluates their model on Make3D and KITTI image datasets. We compare their estimate of ego-motion with two versions of monocular ORB-SLAM 1) ORB-SLAM (full) which recovers odometry using all frames of the driving series , and 2) ORB-SLAM (short)

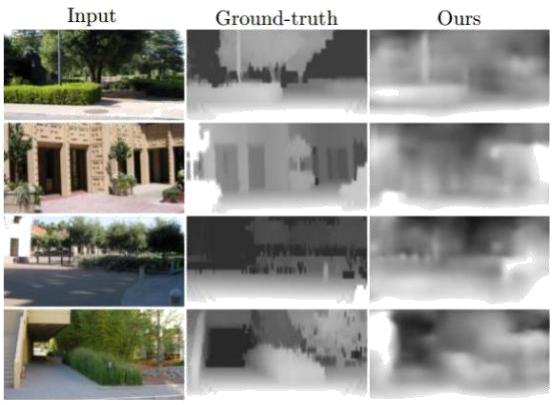

Figure 2.15 Zhou et al [15] prediction on Make3D dataset

The core principle of our architecture is to use different types of data comparison t o improve the depth plus ego-motion and flow-prediction network. The proposed architecture trains the network without a paired dataset by using sequential stereo scenes to estimate depth, optical flow, and ego-motion. We train each network under coupled consistency conditions, thus sharing the parameters. The block of f low consistency produces a flow using depth plus ego-motion and a network of stand-alone flow estimation, called FlowNet. To penalize the photometrically dissimilar regions, the induced optical flow from the depth plus ego-motion is compared with the network-based predicted optical flow.

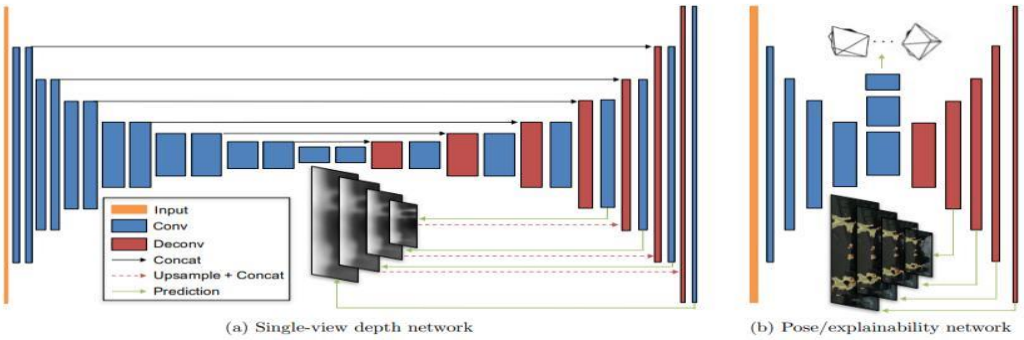

Figure 2.16. Zhou et al. [15] network Architecture.

S.#	Author	Year	Technique	Dataset	RMS
1	D. Eigen, C. Puhrsch, and R. Fergus.	2014	Depth map prediction from a single image using a multi-scale deep network.	NYU v2, KITTI	0.907
2	H. Fu, M. Gong, C. Wang, N. Batmanghelich, and D. Tao. [2]	2018	Deep ordinal regression network for monocular depth estimation.	KITTI,Make3D and NYU v2	0.509
3	I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. [20]	2016	Deeper depth prediction with fully convolutional residual networks.	Make3D and NYU v2	0.573
4	Z. Hao, Y. Li, S. You, and F. Lu.	2018	Detail preserving depth estimation from a single image using attention guided networks.	NYU v2	0.555
5	P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille.	2015	Towards unified depth and semantic prediction from a single image.	NYU v2	0.745
6	B. Li, C. Shen, Y. Dai, A. V. den Hengel, and M. He.	2015	Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs.	Make3D, NYU v2	0.821
7	Zhao, S., Fu, H., Gong, M., & Tao, D.	2019	Geometry-Aware Symmetric Domain Adaptation for Monocular Depth Estimation.	KITTI	3.846
8	Clement Godard ´1 Oisin Mac Aodha2 Michael Firman3 Gabriel Brostow3	2019	Digging Into Self-Supervised Monocular Depth Estimation	KITTI Eigen	4.863
9	C. Godard, O. M. Aodha, and G. J Brostow.[16]	2017	Unsupervised monocular depth estimation with left-right consistency.	KITTI and Cityscapes	4.935
10	Lee, J.-H. and C.-S. Kim. [22]	2019	Monocular depth estimation using relative depth maps.	NYU v2	0.538
11	Godard, C., et al. [21]	2019	Digging intoself-supervised monocular depth estimation.	KITTI Make3D	0.4863
12	Xu, D., et al. [23]	2018	Structured attention guided convolutional neural fields for monocular depth estimation.	NYU v2 KITTI	4.677
13	Liu, F., C. Shen, and G. Lin. [26]	2015	Deep convolutional neural fields for depth estimation from a single	NYU v2 KITTI	0.824

			image.		
14	Zhou, T., et al. [15]	2017	Unsupervised learning of depth and ego-motion from video.	KITTI, Make3D	10.47

Table 2.1. Overview of different techniques of depth estimation

Zhou, T., et al. [15] current system does not directly quantify the dynamics and occlusions of scenes, all of which are crucial factors in 3D perception of scenes. 2) Our system assumes that the object is intrinsic, which prevents the use of random Internet videos of unknown object types / calibration. In C. Godard, O. M. Aodha, and G. J. Brostow.[16] both left - right consistency test and post-processing enhance the quality of the results, certain artifacts are still apparent at occlusion boundaries due to the fact that the pixels in the occlusion area are not apparent in both images . Our system needs rectified and temporarily synchronized stereo pairs during testing, which means that it is not currently possible to use existing single-volume sets. The fully convolutional model proposed by Laina et al. [20] significantly enhances the accuracy of the edge and the description of the structure in the predicted depth maps. The network proposed is completely convolutional, containing up-projection layers that allow much deeper configurations to be trained, while significantly reducing the number of parameters to be learned and the number of training samples required. The algorithm proposed by Lee [22] provides the output and an ablation analysis showed that relative depth maps are more efficient than ordinary maps in maintaining the depth of a scene. Fu et al. [27] , on the other hand, formulated depth estimation as a classification problem. The depth range is first discretized into intervals. The network then learn to classify each image pixel into one of the depth intervals. However, instead of using uniform discretization, Fu et al. [27] used a spacing increasing discretization.

Chapter 3: Datasets

High demand for multiple scene understanding technologies such as autonomous driving, robot navigation, or virtual reality devices. Convolutional neural network (CNN) architectures have been most widely used to substitute or improve conventional approaches and have been shown effectively to infer geometric details only from the monocular RGB or intensity images provided.

Developing and evaluating depth estimation algorithms requires a large amount of representative data, particularly for learned estimation methods. Scharstein and Szeliski [28] provided the Middlebury dataset as an early test environment for quantitative assessment of stereo algorithms, where the ground truth was obtained by structured light. Facilitated by consumer depth cameras such as Microsoft Kinect [29], a number of indoor depth data sets have been proposed [30] . In particular, the NYUdepthv2 data set [24] is a widely used data set with approximately 1500 samples. However, due to the limitations of consumer depth cameras in severe ambient light and modulation frequency limitations, these data sets only cover indoor scenarios with limited ranges. The KITTI Stereo 2015 benchmark [31] has introduced 400 images of street scenes with ground-level truth acquired by the lidar system.

The Kitti dataset [32] includes scenes from streets captured from a moving vehicle, see Figure 3.0. The dataset includes RGB images from a Velodyne laser scanner along with the depth maps. Depth maps are provided, however, only in a very low resolution which also suffers from irregularly and sparsely spaced points. The most

frequently used dataset is the NYU depth v2 dataset [24] which contains 464 indoor scenes with aligned RGB and video sequence depth images obtained from a Microsoft Kinect v1 sensor. A subset of this dataset is mainly used for deep network testing, while a further 654 image and depth pairs are used for evaluation

Fig 3.0 Test vehicle equipped with a RGB stereocamera (Aptina AR0230, 1920x1024, 12bit) and a lidar(Velodyne HDL64-S3, 905nm)

3.1 Data Augmentation

The use of data increases relevant to the task is essential for the network to benefit from the increase. It is necessary to feed the images on the network which it will actually find in the real world. Some popular and helpful increases are for the depth estimation task; small random rotations, gaussian noise, random horizontal flips, color increases and translations.

Applying a procedure named data augmentation is one of their considerations for avoiding overfitting. This technique is primarily used for the purpose of increasing image data by transforming input images, such as cropping, flipping and/or rotating [33]. Other methods, like transferring the style of your image, are available from night to day, winter to summer, sunny to cloudy and vice versa.

This technique increases the amount and variety of data that an algorithm for machine learning needs to improve its efficiency and evaluation. The only valid data augmentation transformation is the horizontal flip of images, because of the existence of the task in DenseDepth. Other transformations which alter the scene's geometry are counterproductive. For example, a vertical flip will put the road and cars at the top of the frame, while the sky would be at the bottom, a scenario that does not suit any future inference input picture, nor any real world scenario. The following table provides explanations for explanation of some of those transformations.

Method	Input image	Output image
Deep Style Transfer (seasons)		
Deep Style Transfer (time of day)		
Horizontal flipping		
Vertical flipping		

Table 3.1. Illustration of Data Augmentation techniques [34]

3.2 NYU Depth v2

Nathan Silberman and Fergus submitted the NYU Depth V2 dataset at ECCV 201 2 Video sequences of 464 indoor scenes captured with Microsoft Kinect [24]. compose NYU Depth v2. v2 [24]. The collection of data includes two components: the labeled 1449 (795 in a training set, 654 in a test set) RGB matched and the 640x480 resolution depth pictures with textual marking for each object. For each object. The second component contains raw camera data and Mi crosoft Kinect output. This results in a total of 407,024 unlabeled videos. The data set also contains MATLAB scripts for the analysis of the raw data, and for the synchronization, alignment and full depth maps. The data set includes the profundity values are 0-10 meters for both components.

Figure.3.1. NYU v2 Sample Images

3.3 KITTI

The dataset consists of about 93,000 images from outdoor scenes, divided into five categories; area, residential, path, campus, and person. The data was collected using a high-resolution stereo camera rig in color and a Velodyne 3D Laser Scanner (LiDAR), which captured the scene depth. A total of 151 sequences have been registered, and the raw data from the left and right camera are given for each frame. A rectified version of the RGB images is available for download along with the raw LiDAR scans.

Depending on the calibration parameters, the resolution of the rectified RGB images varies slightly but is around 1242x375.

The raw LiDAR scans are sparse, indicating that not all pixels contain values. However, Uhrig et al. [35] provided denser depth maps, deriving a large-scale dataset of depth-annotated RGB images from the sparse scan results. Their research was inspired by showing that conventional convolutional neural networks typically perform poorly when trained on sparse data. The raw LiDAR scans are sparse, i.e. They contain as many pixels as four times [35].

KITTI [36] includes video clips from a LIDAR range moving vehicle. In case of RGB images of 1392x512 resolution, LIDAR point cloud data must be taken from depth maps. These are only given for the lower part of the picture. Dataset consists of 56 scenes and has a total of approximately 20,000 pictures. In 2013, the

[37] KITTI dataset was submitted at IJRR. Instead of using the entire KITTI dataset, the Eigen division is generally pursued. This divide includes 23,488 photographs from 32 preparation scenes and 697 photographs from 29 test rooms.Figure 3.2 an example of an input images from the KITTI datasets.

Figure.3.2. KITTI Sample images

3.4 Make3D

Make3D [38] is an outdoor collection of 534 images of open air, 400 pictures are used to test, 134 for testing. The depth information is collected using Laser and the

resulting RGB pictures have a resolution of 2272x1704. To decrease the preparation time, it is additionally usual to down-example these images.

Figure 3.3. Make3D Sample Images

Data record	Figures	Notation Available	Scene
NYUD-V2 [1]	1449 + 407K RAW	Depth + Segmentation	Indoor
KITTI [3]	94K Frames	Depth aligned with RAW data	Street
Make3D [4]	500 Frames	Depth	Outdoor

Table 3.1: Data record for Depth Estimation [8]

3.5 Metrics for evaluating performance of depth estimation on datasets

To benchmark output and compare existing solutions, it is important to evaluate results quantitatively. In Table 3.2, we present the specific metrics used to assess the performance of systems performing depth estimation. This is the estimated depth of the pixel and the target depth of the pixel. At the end of this chapter, Table 2.2 compares the performance of current solutions using these metrics.

Relative error (rel)	$\frac{1}{\lvert N\rvert}\sum_{i\in N}\frac{\lvert d_i-d_i^*\rvert}{d_i^*}$
Square relative error (sqr rel)	$\frac{1}{\lvert N\rvert}\sum_{i\in N}\frac{\lvert d_i-d_i^*\rvert^2}{d_i^*}$

Root mean squared error (rms)	$\sqrt{\frac{1}{\lvert N\rvert}\sum_{i\in N}\lvert d_i - d_i^*\rvert^2}$
Root mean squared error log (rms-log)	$\sqrt{\frac{1}{\lvert N\rvert}\sum_{i\in N}\lvert \log(d_i) - \log(d_i^*)\rvert^2}$
Log10 error (log10)	$\frac{1}{\lvert N\rvert}\sum_{i\in N}\lvert \log_{10}(d_i) - \log_{10}(d_i^*)\rvert$
Threshold of accuracies (Ai) for A ≺ thr	$\%$ of d_i s.t. $\max(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}) < thr$, where $thr \in \{1.25, 1.25^2, 1.25^3\}$

Table 3.2: Metrics used for evaluating performance in depth estimation

Higher values mean better results for each metric except for the threshold metric.

Higher values mean lower error for metric thresholds.

Chapter 4: Proposed Methodology

We define our method of estimating a depth map from a single RGB image in this section. We first define the architecture of the encoder-decoder used. Next we explore our findings on both the encoder and decoder complexity and its output relationship. First, we suggest a fitting loss function for the task in question. Finally, we identify effective policies for expansion that significantly help the training process.

4.1 Depth information:

Depth can be stored in meters for every pixel in the image frame as distance from the camera. The following figure shows depth map for a single RGB image. The depth map is to the right where the actual depth was converted to relative depth using this room's maximum depth.

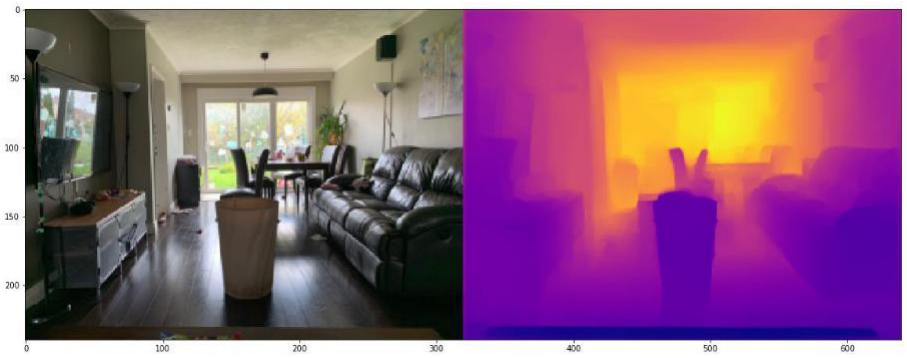

Figure 4. RGB Image and its corresponding depth map

The most widely used approach for real application is a combination of a so-called pre-trained network operating on powerful computer hardware. The general structure for using the pre-trained NN is illustrated in the figure below.

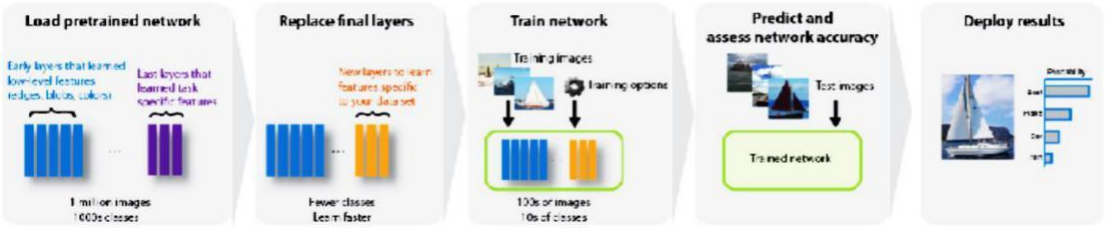

Figure.4.1 General classification of pre-trained network

We propose an architecture that connect all layers (with corresponding feature map sizes) directly to each other to ensure maximum information flow between layers in the network. To preserve the feed-forward nature, each layer gets additional feedback from all preceding layers and passes to all subsequent layers on its own feature-maps as shown below.

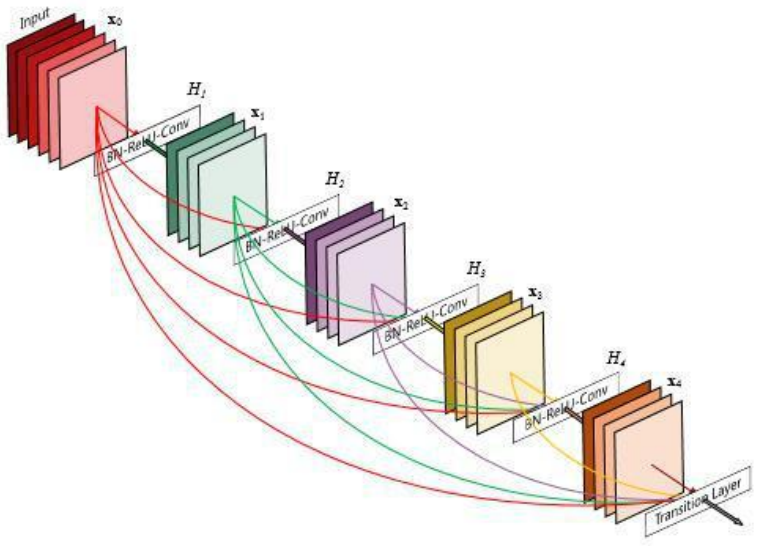

Figure 4.2 Each layer takes all of the previous features-maps as input.

Unlike ResNets, we never combine features by summing them up before they pass into a layer; instead, we combine features by concatenating them. Due to its dense connectivity pattern, DenseConvolutional Network (DenseNet) was approached. The result of this dense communication pattern is that it needs fewer parameters than traditional convolution networks, because redundant feature maps need not be relearned. Traditional feed-forward architectures can be seen as state algorithms which are transferred from layer to layer. Every layer reads the state from its previous layer, and writes to the layer that follows. It changes the state but it also transmits information that must be protected. We evaluate DenseNets on two highly competitive datasets for comparisons (NYU Scope v2, and KITTI). Our models tend to allow far fewer parameters.

4.2 Proposed Network Architecture

DenseDepth relies on transfer learning, a method that makes more effective use of previous information derived from a learning problem — image classification in this case — to help solve another — depth estimation —[39]. Transfer learning has helped this approach to produce similar or even better results to a simpler and modular architecture than other methods. However, it is a strictly supervised form of learning that requires ground-truth-depth data like those from 3D laser scans or other costly hardware.

Our network architecture follows a framework of encoder decoders. The encoder is where the transfer learning takes place, specifically using DenseNet-169 pre-trained on ImageNet [40], an image database for classifying images and recognizing objects. Having some pretrained section allowed this method to reduce validation

loss compared to an initialization of totally random weights. The decoder segment consists of core convolutional layers.

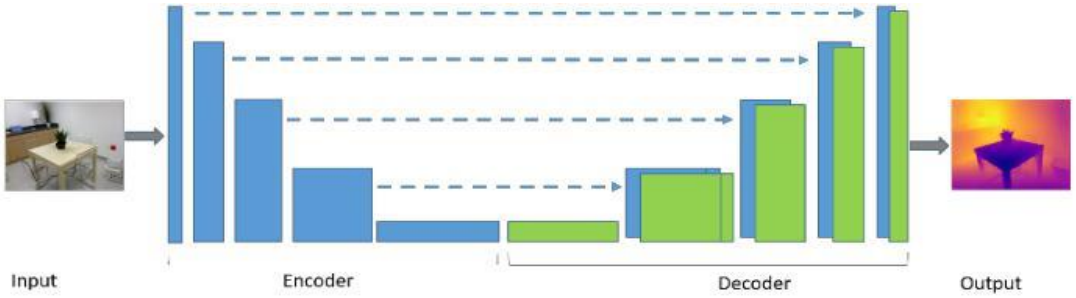

Figure.4.3.(a) An architectural overview of our network.

4.3 **Modifying network architecture**

The encoder presented is a pretrained DenseNet 169.The encoder consists of 4 dense blocks that occur before the fully connected layers in the DenseNet 169 model. It is different from other depth models in that it uses a very simple decoder. Each decoder block consists of a single bi-linear upsampling layer followed by 2 convolution layers. Following another standard practice in encoder decoder architecture, the up sampling layers are concatenated with the corresponding layers in the encoder. Figure below, explains the architecture in more detail.

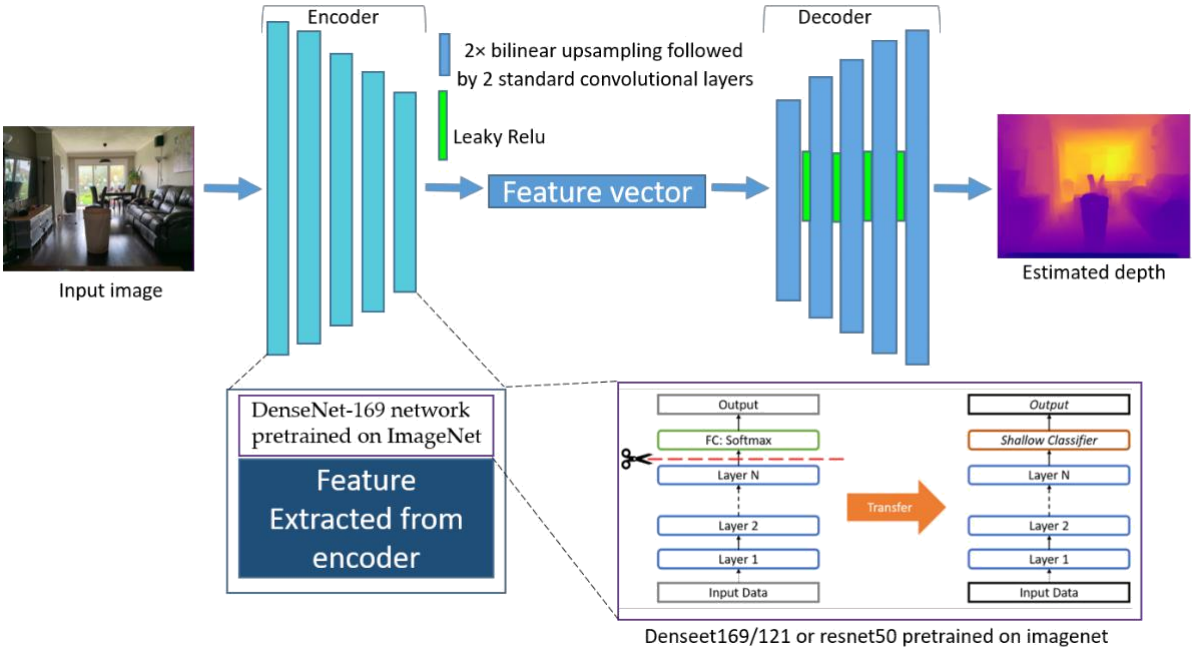

Figure 4.3.(b) Densenet architectural summary [41]

In DenseDepth the mirrored images (horizontal flipping) are used with its horizontally flipped version to average the depth values of the original image. Fig.4.3 (a) displays an overview of the depth estimate of our encoder network. The first part of the decoder consists of four learnable upsampling blocks, which returns an output about half the size of the input image. A drop-out layer is then added with a 50 percent probability, convolution, and a ReLU activation. The output is eventually upsampled by bilinear interpolation, resulting in a final estimate of the same spatial scale as the data.

Using the DenseNet-169[7] network pre-trained on ImageNet [8], the input RGB image is encoded into a function vector for our encoder. This vector is then fed into a successive sequence of up-sampling layers [9] to create the final depth map at half of the input resolution. Our decoder is created by these up sampling layers and their related skip connections. Our decoder includes no suggested batch

normalization [10] or other advanced layers in recent state-of - the-art methods [11, 12].

LAYER	OUTPUT	FUNCTION
INPUT	480 ×640×3	
CONV1	240 ×320×64	DenseNet CONV1
POOL1	120 ×160×64	DenseNet POOL1
POOL2	60 ×80×128	DenseNet POOL2
POOL3	30 ×40×256	DenseNet POOL3
. . .	. . .	. . .
CONV2	15 × 20 × 1664	Convolution 1 × 1 of DenseNet BLOCK4
UP1	30 × 40 × 1664	Upsample 2 × 2
CONCAT1	30 × 40 × 1920	Concatenate POOL3
UP1-CONVA	30 ×40×832	Convolution 3 × 3
UP1-CONVB	30 ×40×832	Convolution 3 × 3
UP2	60 ×80×832	Upsample 2 × 2
CONCAT2	60 ×80×960	Concatenate POOL2
UP2-CONVA	60 ×80×416	Convolution 3 × 3
UP2-CONVB	60 ×80×416	Convolution 3 × 3
UP3	120 × 160 × 416	Upsample 2 × 2
CONCAT3	120 × 160 × 480	Concatenate POOL1
UP3-CONVA	120 × 160 × 208	Convolution 3 × 3
UP3-CONVB	120 × 160 × 208	Convolution 3 × 3
UP4	240 × 320 × 208	Upsample 2 × 2
CONCAT3	240 × 320 × 272	Concatenate CONV1
UP2-CONVA	240 × 320 × 104	Convolution 3 × 3
UP2-CONVB	240 × 320 × 104	Convolution 3 × 3
CONV3	240 ×320×1	Convolution 3 × 3

Table 4.1 Network architecture.

Tab.4.1 shows the structure of encoder-decoder with skip connections network. Encoder is based on the DenseNet-169 [13] network where the top layers removed that are related to the original ImageNet [42] classification task. For decoder, start with a 1 × 1 convolutional layer with the same number of output channels as the

output of truncated encoder. Then successively add up sampling blocks each composed of a 2× bilinear up sampling followed by two 3 × 3 convolutional layers with output filters set to half the number of inputs filters, and were the first convolutional layer of the two is applied on the concatenation of the output of the previous layer and the pooling layer from the encoder having the same spatial dimension. Each up sampling block, except for the last one, is followed by a leaky ReLU activation function [26] with parameter α = 0:2. The input images are represented by their original colors in the range [0; 1] without any input data normalization. Target depth maps are clipped to the range [0:4; 10] in meters. Layers up to CONV2 are exactly those of DenseNet-169 [13]. Up sampling is bilinear up sampling. We follow each CONVB convolutional layer by a leaky ReLU activation function [26] with parameter α = 0:2. Note that in this table we use the output shapes corresponding to the spatial resolution of the dataset NYU Depth v2 (height× width× channels).

4.4 Changing feature extractor to DenseNet 169/121 or Resnet 50

ResNet-50 [20] is used as an encoder to extract features from the input image. ResNet has shown great potential by applying skip connections that make it possible to skip steps that facilitate deep network training [43]. In some of the works [20, 27, 44], different architectures were evaluated against each other, with ResNet outperforming all the others. According to Cao et al. [44] and Eigen et al. [45], deeper architectures also seem to perform slightly better than shallow ones when predicting depth.

While ResNet has shown great results, there are a range of other networks that can be used to extract features and that may function better for this particular task. DenseNet, proposed by Huang et al. [42], is a very large yet narrow network in

which all layers are directly connected to each other. In DenseNet, each layer receives input from all previous layers and transfers its own function maps to all future layers. It ensures the full flow of knowledge between layers and encourages preparation. In comparison to ResNet, where the functions are summed together, DenseNet functions maps are concatenated.

More info on the DenseNet architecture can be found in [42]. Dense Convolutionarl Network (DenseNet), $\frac{L(L+1)}{2}$ which connects each layer in a feed-forward fashion with each other. Whereas conventional convolutional L layer networks have L connections one between each layer and its subsequent layer — our network has direct co direct connections. The feature maps of all preceding layers are used as inputs for each layer, and their own feature maps are used in all subsequent layers as inputs. A possible counter-intuitive effect of this dense pattern of connectivity is that it requires fewer parameters than traditional convolutional networks, as there is no need to re-learn redundant feature maps. Traditional feed-forward architectures can be seen as state algorithms, which are moved from layer to layer. Every layer reads the state from its previous layer, and writes to the layer that follows it. It changes the state but also transmits information that requires preservation. DenseNet layers are very narrow (e.g., 12 filters per layer), add only a small set of features-maps to the "collective knowledge" of the network and keep the remaining features-maps unchanged — and the final classifier makes a decision based on all the features-maps in the network. In addition to improved parameter efficiency, DenseNets' improved flow of information and gradients across the network make it easy for them to train. Each layer has direct access to the gradients of the loss function and the original input signal, resulting in implicit deep supervision [46]. This helps to train deeper

network architecture. Furthermore, we also observe that dense connections have a regularizing effect, which reduces over-fitting for tasks with smaller training sizes.

As DenseNet improves the flow of information, enhances the propagation of features across the network, and uses a method of skip-connections similar to those that have been shown to be useful in ResNet, DenseNet was presumed to have potential as a good feature extractor. At the same time, using DenseNet instead of ResNet would reduce the number of parameters available.

4.5 Network Training

The segment explains how network training has been carried out. Implementation, training and evaluation were carried out in Python using the Anaconda. The Networks have been equipped on NVIDIA GeForce RTX 1080 GPU. Our encoder is a DenseNet-169[17] preformed on ImageNet[5]. The decoder weights are initialized at random after [11]. In all tests, the ADAM[20] optimizer was used with a learning rate of 0:0001 and a parameter value of $\beta 1 = 0:9$, $\beta 2 = 0:999$. The batch size is set to 2. The total number of training parameters for the entire network is approximately 21M. Training is done for 20 epochs on NYU Depth v2, taking 2 hours and 40 minutes per epoch to finish.

4.6 Training details

Training on NVIDIA GeForce GTX 1080 took 2 hours per epoch for the NYU Depth v2[54] data collection . The performance in the NYU Depth v2 dataset is impressive, reaching the highest accuracy in most of the metrics (89.5-99.6%), making it state-of-the-art. All networks are trained by stochastic gradient descent (SGD). We train models for 50 epochs with a batch size of 2 on Densenet169. The learning rate is initially set to 0.1 and is reduced by 10 times in epochs 30 and 50. It took 2 hours 40 minutes per epochs to complete the training.

4.7 Training a neural network

In this section the general training principles of supervised neural networks are described.

4.7.1 Loss Function

The loss function is used to calculate the difference between the value of the ground truth, **y**, and the value of the expected value, *y^*. The difference is also referred to as the error of prediction and provides a numerical measure to how well the network is able to perform its function. When predictions are made pixelwise, the error of prediction is also measured pixelwise. During the training cycle, weights and biases are modified with the goal of reducing the role of loss[20]. Therefore, the option of an appropriate loss function is important for the network to properly learn its task. Our approach is to define a loss function that balances the reconstruction of depth images by reducing the difference in depth values while also penalizing distortions of high-frequency information in the depth map image domain. Usually, these details correspond to the boundaries of the objects in the scene.

Loss L:

For the training of our network, we define the loss L between y and y^ as the weighted sum of three loss functions:

$$L(y, \hat{y}) = \lambda L_{depth}(y, \hat{y}) + L_{grad}(y, \hat{y}) + L_{SSIM}(y, \hat{y}) \quad (1)$$

L_1 loss

The first loss L_{depth} term depth is the point-wise L1 loss described by the depth values:

$$L_{depth}(y, \hat{y}) = \frac{1}{n} \sum_{p}^{n} |y_p - \hat{y}_p|. \qquad (2)$$

The second loss term Lgrad is the loss of L1 defined by the image gradient g of the depth map:

L_2 loss

Second L_{grad} loss is the L1 loss defined by the gradient g for the image depth:

$$L_{grad}(y, \hat{y}) = \frac{1}{n} \sum_{p}^{n} |\boldsymbol{g}_{\mathbf{x}}(y_p, \hat{y}_p)| + |\boldsymbol{g}_{\mathbf{y}}(y_p, \hat{y}_p)| \qquad (3)$$

where the difference between x and y components for depth gradients of y and y^ is determined by gx and gy.

Structural Similarity (SSIM) Loss

L_{SSIM} uses the term "Structural Similarity" (SSIM)[47] which is a commonly used metric for image reconstruction tasks.

$$L_{SSIM}(y, \hat{y}) = \frac{1 - SSIM(y, \hat{y})}{2}. \qquad (4)$$

Note that we only define a weight parameter λ for the loss term L_{depth}. In empirical terms, we have identified and established λ = 0:1 as a reasonable weight for this term.

Chapter 5: Experimental Results

5.1. Environment

Next a description of the area in which the experiment was performed is provided for reference and reproducibility purposes. Models were tested on an NVIDIA GeForce GTX 1080 graphics card (8 GB of Memory) on the same desktop computer. Monodepth as run using TensorFlow [55], a Google-developed software library for machine learning and neural network research.

Hardware environment	
GPU	GeForce GTX 1080
Software Environment	
Programming language	Python 3.6 + Anaconda(Spyder 4.0)
Framework	Tensorflow 1.9.0

Table 5.1 Summary of operational environment

5.2. Pre-Trained Models

Monodepth implementation provide open source pre-trained models for those attempting to replicate their experiments. That is, the authors have already done training on large data sets, and a model has been selected as the best performer for a given dataset. The best weights can be applied to the network with a pre-trained model such that the depth inference can be determined without the need for network retraining, a very challenging method due to the high complexity of convolutional neural network architecture.

Only two pretrained models are available: one trained on the NYU Depth V2 dataset and the other trained on the KITTI dataset. A description of the pretrained models available is given in the table below:

Name	Dataset	Description	Fine-tuned
NYU	NYU Depth V2	Interior, Rooms	NO
KITTI	KITTI	Exterior, city road	NO

Table 5.2 Summary of available pre-trained models

Pretrained models are expected to do better to extract depth from images taken in similar environments as the dataset used to train the models. For example, using the models trained with KITTI or Cityscapes would be better for the task of depth perception in an autonomous vehicle driving through a city. However, EIGEN or NYU Depth V2 are more suitable for 3D reconstruction or navigation of interiors.

5.3. Results

In this section, a quantitative and qualitative analysis using dataset images has been done. Color-coded maps of the depth / disparity of this model are presented with the plasma color map as shown below in figure 5.1.

Figure 5.1 Plasma color map

Available in two pre-trained models, one using KITTI dataset and the other using NYU v2 data collection. The best results were obtained by using the NYU pre-trained model to extract depth maps from roads and cities. The best results from this experiment can be seen in the following figure with the monocular input image that was fed to the algorithm. The following figure displays the best depth maps, along with the corresponding monocular inputs, from the random collection of indoor images.

Figure 5.2 Results from Dense Depth using NYU pre-trained model

Dense Depth is capable of extracting depth information from these high-detailed, monocular indoor images that can be seen in many features such as the boundary of the object and the many levels of the depth of the real world that it displays.

5.3.1. Training and testing Results on depth estimation models

Our architecture is capable of extracting depth information from these high-detailed, monocular indoor images, which can be seen in many features such as the boundary of the object and the many levels of the depth of the real world.

Dense Depth was trained in a 50 K sample of NYU-v2 data set. The input was a 640x480 resolution RGB image and the output was a 320 x 240 resolution depth map. The model was trained in Keras using the Adam optimiser.

Even though ResNet has shown great results, a number of other networks exist that can be used to extract features and may function better for this specific task. DenseNet, proposed by Huang et al. [42], is a very large but narrow network to which all layers are directly connected. -- layer receives inputs from all previous layers in DenseNet, and transfers its own feature maps to all future layers. It ensures optimal flow of knowledge between layers, and ease training. The

feature maps in DenseNet are concatenated in comparison to ResNet, in which the features are summed together.

Since DenseNet improves the flow of information, strengthens the propagation of the feature through the network, and uses a type of skip-connections similar to those proved useful in ResNet, DenseNet was assumed to have potential as a good extractor feature. At the same time, using DenseNet instead of ResNet will decrease the number of necessary parameters. ResNet backbone has, therefore, been replaced with DenseNet-201.Three different model architectures (Densenet169,201 and resnet 50) have been trained:

Training Results using Densenet 169

1. An implementation of the DensetNet 169 encoder. This model has been trained for 50 epochs (2 hours per epoch on NVIDIA 1080) as shown in below figures.

Figure 5.3 Epoch 1 training result using densenet169

Figure 5.4 Epoch 10 training result using densenet169

Figure 5.5 Epoch 20 training result using densenet169

Figure 5.6 Epoch 30 training result using densenet169

Figure 5.7 Epoch 40 training result using densenet169

Figure 5.8 Epoch 50 training result using densenet169

Training Results using Densenet 201

2. An implementation of a DenseNet 121 encoder which has fewer parameters than DenseNet 169. This model was trained for 20 epochs (2 hours per epoch on GPU) as validation loss had stabilized by this point.

Figure 5.9 Epoch1 training result using densenet 201

Figure 5.10 Epoch5 training result using densenet 201

Figure 5.11 Epoch10 training result using densenet 201

Figure 5.12 Epoch 15 training result using densenet 201

Figure 5.13 Epoch 20 training result using densenet 201

Training results using Resnet50

3. Implementing a Resnet 50 encoder with more parameters than DenseNet 169.This model was trained for 20 epochs (2 hours per epoch on GPU) and the training was discontinued as the model had begun to be over fitted.

Figure 5.14 Epoch 1 training result using Rsnet50

Figure 5.15 Epoch 5 training result using Rsnet50

Figure 5.16 Epoch 10 training result using Rsnet50

Figure 5.17 Epoch 15 training result using Rsnet50

Figure 5.18 Epoch 20 training result using Rsnet50

In above figures we can see as the number of epochs are increases the fine details of depth maps becomes more visible.

Testing Results During the test period, we calculate the complete test image depth map prediction and Then measure by 2 to suit the ground truth resolution and test Eigen et al. on the center crop predefined [16]. At test time, we calculate the final performance by taking a picture's average prediction and its mirror image prediction.

Figure 5.19. Testing Results

5.4 Evaluation Results:

Densenet169 performance in indoor scenarios is significantly better than densenet 201 and resnet50. The following figure shows a few examples of the difference between the outputs produced by the three methods compared to the input images. Three performance metrics were used in these experiments to evaluate densenet169, 201 and Resnet50 as shown below in tables 5.3,5.4 and 5.5.

Epochs	a1	a2	a3	Rel	rms	log_10
1	0.7998,	0.9589,	0.9922,	0.1459,	0.5221,	0.0622
2	0.8154,	0.9671,	0.9935,	0.1337,	0.5083,	0.059
3	0.8250,	0.9678,	0.9933,	0.1307,	0.4892,	0.0577
4	0.8309,	0.9697,	0.9936,	0.1301,	0.4899,	0.0569
5	0.8422,	0.9718,	0.9931,	0.1268,	0.4711,	0.0548
6	0.8389,	0.9712,	0.9930,	0.1303,	0.4719,	0.0551
7	0.8350,	0.9711,	0.9936,	0.1266,	0.4783,	0.0555
8	0.8383,	0.9713,	0.9935,	0.1263,	0.4736,	0.0548
9	0.8331,	0.9691,	0.9930,	0.1268,	0.4868,	0.0561
10	0.8386,	0.9709,	0.9935,	0.1272,	0.4741,	0.055
11	0.8383,	0.9707,	0.9935,	0.1259,	0.4801,	0.0551
12	0.8419,	0.9710,	0.9930,	0.1259,	0.4718,	0.0547
13	0.8411,	0.9717,	0.9936,	0.1252,	0.4708,	0.0545
14	0.8381,	0.9706,	0.9933,	0.1259,	0.4744,	0.0553
15	0.8425,	0.9719,	0.9934,	0.1252,	0.4679,	0.0545
16	0.8398,	0.9710,	0.9938,	0.1265,	0.4708,	0.0546
17	0.8353,	0.9703,	0.9937,	0.1264,	0.4781,	0.056
18	0.8417,	0.9720,	0.9936,	0.1248,	0.4691,	0.0542
19	0.8418,	0.9722,	0.9937,	0.1259,	0.4678,	0.0542
20	0.8347,	0.9706,	0.9932,	0.1266,	0.4782,	0.0556
21	0.8369,	0.9707,	0.9932,	0.1261,	0.4768,	0.0552
22	0.8383,	0.9705,	0.9935,	0.1265,	0.4720,	0.0547
23	0.8381,	0.9702,	0.9933,	0.1267,	0.4741,	0.055
24	0.8398,	0.9703,	0.9933,	0.1259,	0.4747,	0.0548
25	0.8382,	0.9713,	0.9934,	0.1264,	0.4724,	0.0549
26	0.8373,	0.9702,	0.9935,	0.1268,	0.4780,	0.0553
27	0.8356,	0.9696,	0.9934,	0.1263,	0.4796,	0.0554
28	0.8405,	0.9706,	0.9934,	0.1261,	0.4702,	0.0546

29	0.8368,	0.9699,	0.9932,	0.1255,	0.4785,	0.0549
30	0.8380,	0.9699,	0.9935,	0.1262,	0.4761,	0.0552
31	0.8390,	0.9698,	0.9931,	0.1262,	0.4771,	0.0549
32	0.8372,	0.9699,	0.9929,	0.1265,	0.4775,	0.0551
33	0.8357,	0.9689,	0.9930,	0.1264,	0.4840,	0.0558
34	0.8412,	0.9704,	0.9932,	0.1264,	0.4712,	0.0545
35	0.8388,	0.9703,	0.9935,	0.1260,	0.4740,	0.0549
36	0.8384,	0.9702,	0.9931,	0.1265,	0.4744,	0.0553
37	0.8386,	0.9702,	0.9933,	0.1265,	0.4765,	0.0551
38	0.8359,	0.9690,	0.9928,	0.1268,	0.4818,	0.0555
39	0.8394,	0.9702,	0.9931,	0.1265,	0.4738,	0.0548
40	0.8359,	0.9695,	0.9933,	0.1265,	0.4788,	0.0556
41	0.8363,	0.9690,	0.9929,	0.1261,	0.4826,	0.0555
42	0.8409,	0.9705,	0.9930,	0.1252,	0.4724,	0.0546
43	0.8381,	0.9692,	0.9933,	0.1266,	0.4774,	0.0554
44	0.8392,	0.9687,	0.9931,	0.1264,	0.4784,	0.0551
45	0.8380,	0.9696,	0.9935,	0.1262,	0.4766,	0.0551
46	0.8393,	0.9700,	0.9935,	0.1268,	0.4733,	0.0548
47	0.8377,	0.9684,	0.9931,	0.1261,	0.4769,	0.0552
48	0.8388,	0.9695,	0.9935,	0.1259,	0.4760,	0.0551
49	0.8372,	0.9687,	0.9933,	0.1264,	0.4814,	0.0555
50	0.8405,	0.9697,	0.9933,	0.1272,	0.4695,	0.0545

Table 5.3 Densenet169 Evaluation results on performance metrics

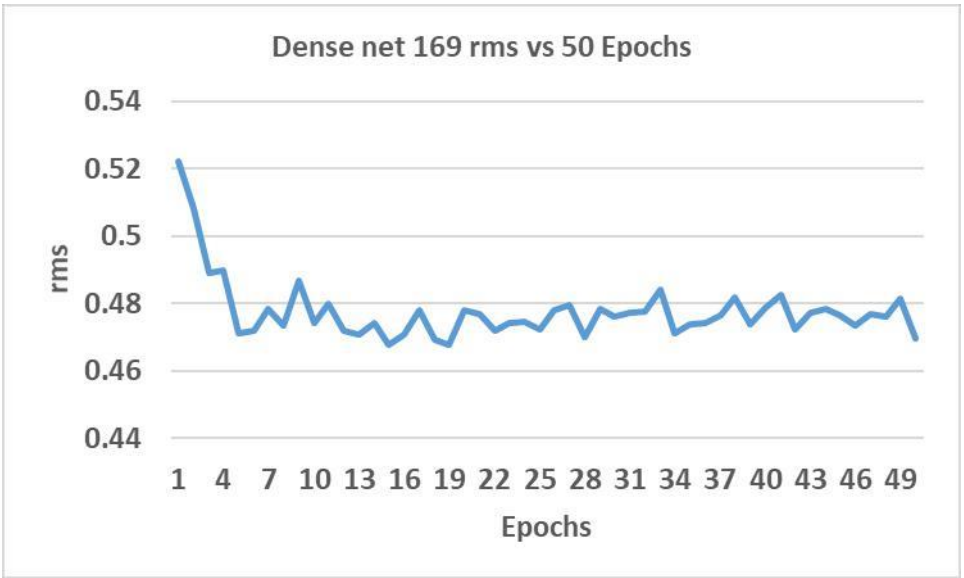

Figure 5.20

After 50th epoch results becomes decaying. Therefore, we stop on 50th epoch.

Epochs	a1,	a2,	a3,	rel,	rms,	log_10
1	0.7449,	0.9542,	0.9904,	0.1811,	0.5855,	0.0717
2	0.7845,	0.9564,	0.9911,	0.1437,	0.5576,	0.065
3	0.8133,	0.9662,	0.9928,	0.1366,	0.5028,	0.059
4	0.8168,	0.9665,	0.9932,	0.1369,	0.4952,	0.0587
5	0.8205,	0.9660,	0.9928,	0.1321,	0.4952,	0.0581
6	0.8187,	0.9673,	0.9930,	0.1342,	0.4955,	0.058
7	0.8210,	0.9655,	0.9925,	0.1327,	0.4962,	0.0581
8	0.8168,	0.9652,	0.9925,	0.1328,	0.5040,	0.0587
9	0.8257,	0.9678,	0.9930,	0.1310,	0.4883,	0.0568
10	0.8239,	0.9665,	0.9926,	0.1336,	0.4894,	0.0574
11	0.8207,	0.9651,	0.9922,	0.1313,	0.4980,	0.0579
12	0.8215,	0.9659,	0.9924,	0.1312,	0.4954,	0.0577
13	0.8248,	0.9676,	0.9928,	0.1308,	0.4889,	0.0569
14	0.8183,	0.9651,	0.9927,	0.1319,	0.5046,	0.0583
15	0.8217,	0.9659,	0.9925,	0.1310,	0.4935,	0.0576
16	0.8251,	0.9650,	0.9923,	0.1343,	0.4877,	0.057
17	0.8232,	0.9660,	0.9922,	0.1322,	0.4927,	0.0579
18	0.8250,	0.9672,	0.9928,	0.1309,	0.4876,	0.0569
19	0.8300,	0.9677,	0.9923,	0.1310,	0.4843,	0.0565
20	0.8241,	0.9663,	0.9924,	0.1306,	0.4909,	0.057

Table 5.4 Densenet 201 Evaluation results on performance metrics

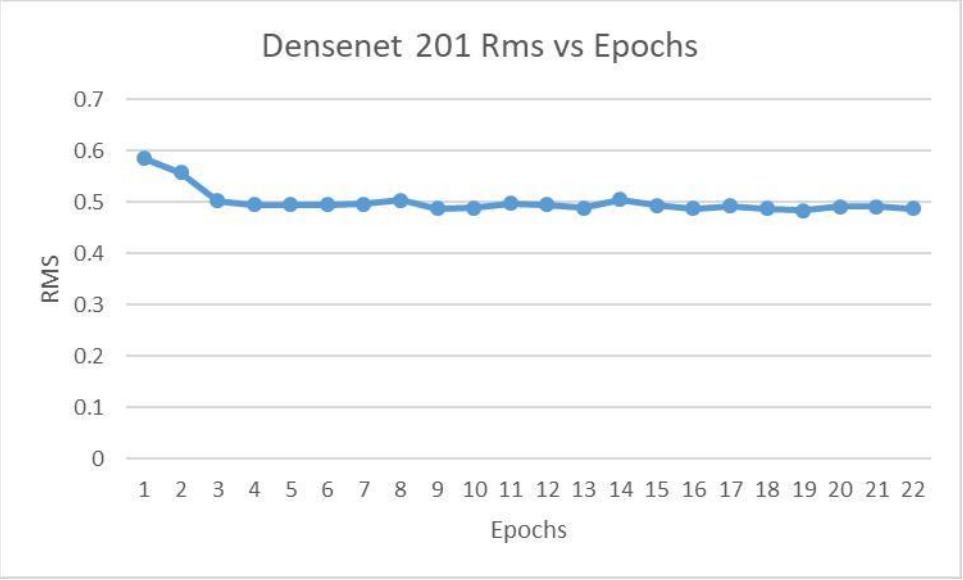

Figure 5.21

Epochs	a1,	a2,	a3,	rel,	rms,	log_10
1	0.7363,	0.9379,	0.9850,	0.1674,	0.5920,	0.0731
2	0.7501,	0.9424,	0.9870,	0.1673,	0.5799,	0.0710
3	0.7622,	0.9485,	0.9884,	0.1586,	0.5683,	0.0685
4	0.7645,	0.9492,	0.9887,	0.1587,	0.5607,	0.0679
5	0.7701,	0.9483,	0.9874,	0.1592,	0.5589,	0.0674
6	0.7781,	0.9511,	0.9891,	0.1574,	0.5417,	0.0659
7	0.7644,	0.9463,	0.9885,	0.1590,	0.5698,	0.0686
8	0.7761,	0.9514,	0.9896,	0.1562,	0.5427,	0.0661
9	0.7744,	0.9504,	0.9895,	0.1551,	0.5516,	0.0667
10	0.7689,	0.9491,	0.9895,	0.1549,	0.5635,	0.0675
11	0.7736,	0.9489,	0.9891,	0.1543,	0.5596,	0.0670
12	0.7788,	0.9513,	0.9893,	0.1535,	0.5463,	0.0659
13	0.7733,	0.9487,	0.9889,	0.1543,	0.5591,	0.0672
14	0.7694,	0.9484,	0.9892,	0.1559,	0.5628,	0.0677
15	0.7773,	0.9505,	0.9894,	0.1563,	0.5450,	0.0661
16	0.7797,	0.9497,	0.9890,	0.1557,	0.5470,	0.0660
17	0.7698,	0.9485,	0.9891,	0.1550,	0.5580,	0.0676
18	0.7817,	0.9496,	0.9887,	0.1542,	0.5410,	0.0656
19	0.7828,	0.9506,	0.9882,	0.1562,	0.5368,	0.0653
20	0.7731,	0.9477,	0.9884,	0.1572,	0.5558,	0.0670

Table 5.5 Resnet50 Evaluation results on performance metrics

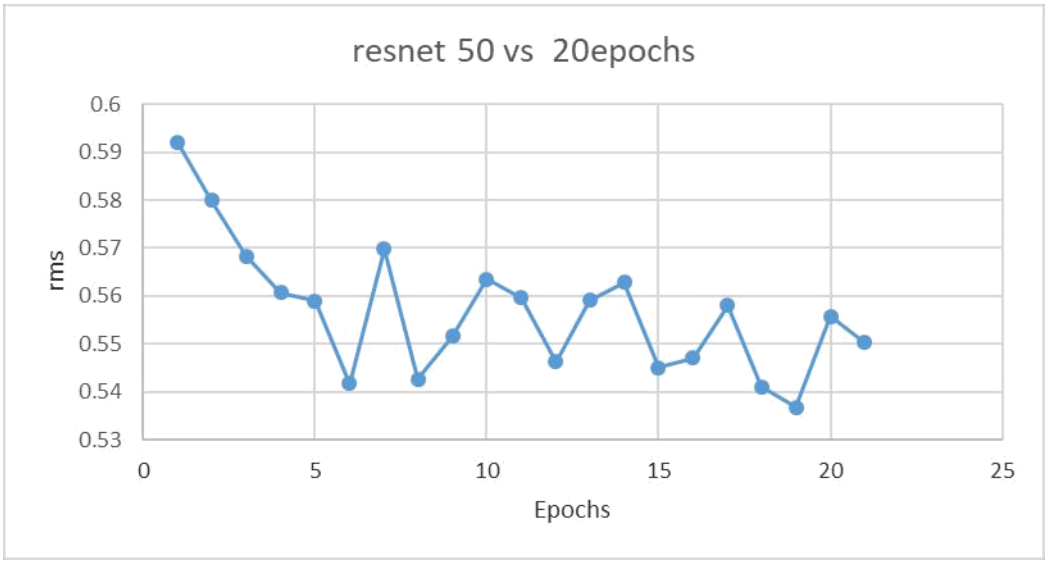

Figure 5.22

DenseNet 169 model performs better than DenseNet121 and ResNet 50 as seen in the figure below.

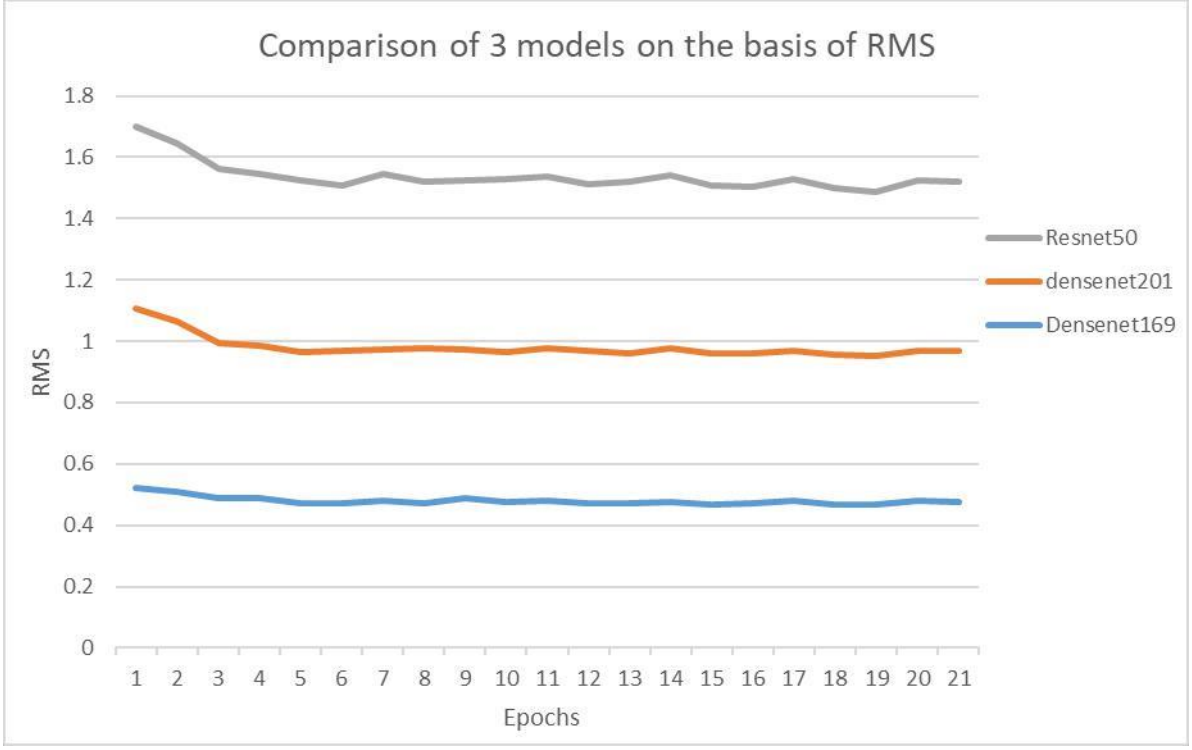

Figure 5.23 Comparison of 3 models on the basis of RMS

We use three different metrics to compare three models (i.e Densenet 169 , 201 and resnet50) output— average relative error (rel) in predicted and actual depth, RMSE (rms) — root mean square error in actual and predicted depth, and average log error (log) between the two depths and threshold of accuracies (Ai) for A <thr (for thr = 1.25,1.252,1.253)as shown in Table 5.6. For all those metrics, lower values imply a stronger model as shown in Figure 5.19.

A1	A2	A3	Rel	Rms	Log_10
0.8471	0.9731	0.9937	0.1234	0.4678	0.0535

Table 5.6. Accuracies1, 2, 3, relative error, root mean square and logarithmic

	Previous Papers	Ours Model
Parameters	Lee 63M DORN 123.4M Laina 218 M	42.6M
Training data	120kSamples(DORN)	50K Samples
RMSE	Dorn 2.727 Laina 0.573 Eigen 7.156	0.4678
Threshold Accuracy	DORN (0.964) Laina (0.967)	0.9937
Depth Map Resolution	Santos 120x160 Laina 160x128	320x420
Training Iterations	Dorn 3M Laina 5M	750K(15 Epochs)

Table 5.7. Comparison of our model with previously published papers

We compare the model performance in terms of rms with Laina, Eigen et al, Godard et al and Fu et al. As shown in table 5.6 ours model densenet 169 give lowest rms value which implies that densenet169 model is stronger.

Here,are the visual resuts of densenet169 model on images from CityScapes dataset are presented.Note that our model had only been trained on NYU v2 depth dataset not on CityScape,it also gives acceptable results on cityscape dataset that is rms=0.5224 which is nearest to our model value.

Figure 5.24 Testing results on CityScape dataset

Previous papers	RMS	A1	A2	A3
Godard	5.093	0.879	0.962	0.989
Dungbo Min	3.162	0.901	0.969	0.986
Ours Model	0.5224	0.7845	0.9564	0.9911

Table 5.8 Result comparison of our model with previous work done on images from CityScapes dataset

Chapter 6: Conclusion and Future work

In this section conclusion and future work is discussed.

6.1 Conclusion

For this thesis the question of monocular depth estimation was investigated, which was ill posed.problem. This study presents a novel, deep convolutional neural network, which predicts depth from monocular RGB images by deploying an encoder-decoder framework to solve the issue. The suggested network architecture is inspired by state-of-the-art networks estimating the depth. The network achieves results comparable to the state-of - the-art methods on the NYU V2 dataset by integrating dilated convolutions to capture context at different scales, skip-connections to recover high level details, and ReLU activations to improve model fitting.

We have presented a convolutional neural system for estimating depth map of individual RGB images through the use of recent development in architecture of network and high-performance pre-trained model availability.

We have a pipeline for a strong Depth Estimation model that is simple and easy to learn. A DenseNet encoder with fewer parameters was introduced. We use three different metrics for compare model performance. The lower values indicate a stronger model for all these metrics.

To summarize, the main goal was achieved (obtaining a depth map from a single RGB image). It was possible at the writing of this thesis thanks to the other sub-objectives developed and completed. Some potential future lines of further research linked to monocular depth perception are presented and addressed in the next and final section based on the results obtained during this review.

6.2 Future Work

The computer vision area is continually developing, improving and discovering new methods, solutions and applications for the most important tasks in the area. The challenge of extracting 3D information from a single image is a fascinating line of research for future related work.

We plan to finding improved data augmentation policies and their probability values for the problem of depth estimation is an interesting topic for future work.[48]

References

1. Lee, W., N. Park, and W. Woo. *Depth-assisted real-time 3D object detection for augmented reality*. in *ICAT*. 2011.
2. Moreno-Noguer, F., P.N. Belhumeur, and S.K. Nayar, *Active refocusing of images and videos.* ACM Transactions On Graphics (TOG), 2007. **26**(3): p. 67-es.
3. Hazirbas, C., et al. *Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture*. in *Asian conference on computer vision*. 2016. Springer.
4. Hedman, P. and J. Kopf, *Instant 3d photography.* ACM Transactions on Graphics (TOG), 2018. **37**(4): p. 1-12.
5. Cao, C., et al., *Stabilized real-time face tracking via a learned dynamic rigidity prior.* ACM Transactions on Graphics (TOG), 2018. **37**(6): p. 1-11.
6. Wang, L., et al., *DeepLens: shallow depth of field from a single image.* arXiv preprint arXiv:1810.08100, 2018.
7. Anderson, B.L., *Can computational goals inform theories of vision?* Topics in Cognitive Science, 2015. **7**(2): p. 274-286.
8. Bhoi, A., *Monocular depth estimation: A survey.* arXiv preprint arXiv:1901.09402, 2019.
9. Szeliski, R., *Computer vision: algorithms and applications*. 2010: Springer Science & Business Media.
10. Okutomi, M. and T. Kanade, *A multiple-baseline stereo.* IEEE Transactions on pattern analysis and machine intelligence, 1993. **15**(4): p. 353-363.
11. Boykov, Y., O. Veksler, and R. Zabih, *A variable window approach to early vision.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998. **20**(12): p. 1283-1294.
12. Birchfield, S. and C. Tomasi, *Depth discontinuities by pixel-to-pixel stereo.* International Journal of Computer Vision, 1999. **35**(3): p. 269-293.
13. Eigen, D., C. Puhrsch, and R. Fergus. *Depth map prediction from a single image using a multi-scale deep network*. in *Advances in neural information processing systems*. 2014.
14. Garg, R., et al. *Unsupervised cnn for single view depth estimation: Geometry to the rescue*. in *European Conference on Computer Vision*. 2016. Springer.
15. Zhou, T., et al. *Unsupervised learning of depth and ego-motion from video*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.
16. Godard, C., O. Mac Aodha, and G.J. Brostow. *Unsupervised monocular depth estimation with left-right consistency*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.
17. Goodfellow, I., Y. Bengio, and A. Courville, *Deep learning*. 2016: MIT press.
18. Nielsen, M.A., *Neural networks and deep learning*. Vol. 2018. 2015: Determination press San Francisco, CA, USA:.
19. Altenberger, F. and C. Lenz, *A non-technical survey on deep convolutional neural network architectures.* arXiv preprint arXiv:1803.02129, 2018.

20. Laina, I., et al. *Deeper depth prediction with fully convolutional residual networks*. in *2016 Fourth international conference on 3D vision (3DV)*. 2016. IEEE.
21. Godard, C., et al. *Digging into self-supervised monocular depth estimation*. in *Proceedings of the IEEE International Conference on Computer Vision*. 2019.
22. Lee, J.-H. and C.-S. Kim. *Monocular depth estimation using relative depth maps*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.
23. Xu, D., et al. *Structured attention guided convolutional neural fields for monocular depth estimation*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.
24. Silberman, N., et al. *Indoor segmentation and support inference from rgbd images*. in *European conference on computer vision*. 2012. Springer.
25. Geiger, A., et al., *Vision meets robotics: The kitti dataset.* The International Journal of Robotics Research, 2013. **32**(11): p. 1231-1237.
26. Liu, F., C. Shen, and G. Lin. *Deep convolutional neural fields for depth estimation from a single image*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
27. Fu, H., et al. *Deep ordinal regression network for monocular depth estimation*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.
28. Scharstein, D. and R. Szeliski, *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms.* International journal of computer vision, 2002. **47**(1-3): p. 7-42.
29. Zhang, Z., *Microsoft kinect sensor and its effect.* IEEE multimedia, 2012. **19**(2): p. 4-10.
30. Song, S., S.P. Lichtenberg, and J. Xiao. *Sun rgb-d: A rgb-d scene understanding benchmark suite*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
31. Menze, M. and A. Geiger. *Object scene flow for autonomous vehicles*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
32. Geiger, A., P. Lenz, and R. Urtasun. *Are we ready for autonomous driving? the kitti vision benchmark suite*. in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012. IEEE.
33. Perez, L. and J. Wang, *The effectiveness of data augmentation in image classification using deep learning.* arXiv preprint arXiv:1712.04621, 2017.
34. Luan, F., et al. *Deep photo style transfer*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.
35. Yang, N., et al. *Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry*. in *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
36. Geiger, A., et al., *Vision meets robotics: The kitti dataset. International Journal of Robotics Research (IJRR).* 2013.
37. Uhrig, J., et al. *Sparsity invariant cnns*. in *2017 International Conference on 3D Vision (3DV)*. 2017. IEEE.

38. Saxena, A., M. Sun, and A.Y. Ng, *Make3d: Learning 3d scene structure from a single still image.* IEEE transactions on pattern analysis and machine intelligence, 2008. **31**(5): p. 824-840.
39. Ventura, D. and S. Warnick, *A theoretical foundation for inductive transfer.* Brigham Young University, College of Physical and Mathematical Sciences, 2007.
40. Challenge, I.L.S.V.R., *Available online: http://www.* image-*net. org/challenges.* LSVRC/(accessed on 26 February 2019), 2014.
41. Alhashim, I. and P. Wonka, *High quality monocular depth estimation via transfer learning.* arXiv preprint arXiv:1812.11941, 2018.
42. Huang, G., et al. *Densely connected convolutional networks*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
43. He, K., et al. *Deep residual learning for image recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
44. Cao, Y., Z. Wu, and C. Shen, *Estimating depth from monocular images as classification using deep fully convolutional residual networks.* IEEE Transactions on Circuits and Systems for Video Technology, 2017. **28**(11): p. 3174-3182.
45. Eigen, D. and R. Fergus. *Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture*. in *Proceedings of the IEEE international conference on computer vision*. 2015.
46. Lee, C.-Y., et al. *Deeply-supervised nets*. in *Artificial intelligence and statistics*. 2015.
47. Wang, Z., et al., *Image quality assessment: from error visibility to structural similarity.* IEEE transactions on image processing, 2004. **13**(4): p. 600-612.
48. Cubuk, E.D., et al. *Autoaugment: Learning augmentation strategies from data*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019.