University of Southampton

Faculty of Engineering and Physical Sciences

Electronics and Computer Science

Review of Depth Estimation methods using Deep Learning along with development and comparison of three Depth estimation end to end approaches

By

Srivenkata Srikanth

30/08/2021

Supervisor: Dr Shiyan Hu

Second Examiner: Dr Son Hoang

A dissertation submitted in partial fulfilment of the degree of

MSc Computer Science

ABSTRACT

Depth Estimation has a variety of applications and demands in several domains. The process is used by self-driving cars to build a three-dimensional map of the surroundings which is then used by the other modules. Demand for depth estimation is also growing in the domain of Robotics for applications that include robot-assisted surgery and automated relocation of cargo. The current depth estimation solution used is LIDAR. While it is accurate in certain situations, it is expensive for the amount of value it provides. Hence, there is a demand for an alternative solution. Deep learning can be used to solve this problem and can provide stunningly accurate depth maps. This article focuses on the development and evaluation of quick and efficient approaches with low computational complexity, in order to closely adhere to the motivation of using deep learning, which is to reduce the cost. It first starts off with an introduction followed by a thorough literature survey, including a brief review and comparison of the current state of the art approaches, along with the concepts involved.

The KITTI and the NYU datasets were used. The article then moves on to the detailed explanation and testing of the approaches. Three monocular depth estimation approaches were developed, namely the Pix2pix model, the U-net with DenseNet encoder and the U-net with MobileNetv2 encoder. The architectures of these approaches were illustrated and their results on both datasets were evaluated. The structural similarity (SSIM) and the Mean Squared Errors were used for this. A new metric was also engineered for the fair and more apposite comparison of the approaches, which factored in the computational complexity of the approaches. It was determined that the Pix2pix model performed the best with an SSIM and Engineered metric scores of over 0.95. The testing process involved both, indoor and outdoor scenes. The U-nets with Densenet and MobileNetv2 encoders scored relatively less on these metrics. However, the generated depth maps were further evaluated and explained through inspection to even out the differences caused by various factors such as colour encodings, between the ground truth maps and the depth maps generated by the approaches. Here, it was determined that the U-net with MobileNetv2 also generated accurate depth maps. Its score was low solely due to the aforementioned differences.

Finally, a Graphical user interface was developed to facilitate the hybridised and appropriate use of the engineered models. Then, the future prospects were stated. These included the generation of a new dataset with both indoor and outdoor scenes and integration with smartphones and other hardware.

ACKNOWLEDGEMENTS

This dissertation helped me gain immense exposure along with hands-on experience and helped me boost my knowledge in the domain of depth estimation and image to image translation. I had learnt deep learning before but that involved simple classification and regression tasks. The concept of image-to-image translation and the usage of deep learning for such tasks was completely new to me. Through this dissertation, I now have a very good idea of the field. It has taken my experience and knowledge of deep learning and especially convolutional neural networks to the next level. It has also made me passionate about Generative Adversarial Networks and I shall continue learning more and working on them even after this project.

I would like to thank my guide, Dr Shiyan Hu, for his incredible support and guidance throughout this dissertation. He guided me when I started to vacillate. He was kind throughout the course of this project and addressed all the questions and considerations that I had.

In addition, I would like to thank Dr Son Huang, my second examiner. It was a pleasure presenting my work to him. He asked a lot of brilliant and relevant questions and gave me great suggestions about the future scope of this project.

I am very grateful to the Electronics and Computer science school at the University of Southampton for the stellar MSc course and for giving me this amazing research opportunity to learn and grow.

Finally, I would like to thank my parents for their guidance, financing my education and always being there and supporting me throughout my studies.

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

You must change the statements in the boxes if you do not agree with them.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

I have not used any resources produced by anyone else.

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

CONTENTS

Abstract 2

Acknowledgements 3

List of figures 6

List of tables 7

1 Introduction 8

1.1 Motivation for deep learning in Depth Estimation 8

1.2 Report Structure 9

2 Literature Survey 9

2.1 Background and Theory 9

2.1.1 Types of Depth Estimation 9

2.1.2 CNNs 10

2.1.3 U-Nets 10

2.1.4 Transfer learning 11

2.1.5 GAN 12

2.1.6 SSIM and MSE loss 13

2.2 Related work 13

2.3 Shortlisting and critical analysis of the state of the art approaches 15

3 Data and Requirement Specification 17

3.1 Data used 17

3.1.1 KITTI Depth Dataset 17

3.1.2 NYUv2 depth dataset 18

3.2 Technology used 19

4 Methodology and implementation 19

4.1 Data Pre-processing 19

4.2 Modelling 20

4.2.1 The Pix2pix model 20

4.2.2 The U-net with Densenet as encoder 24

4.2.3 The U-net with MobileNetv2 as encoder 28

4.3 Stereo Depth Estimation 33

4.4 The Graphical User Interface 35

5 Results 37

5.1 The Engineered Metric 37

5.2 Scores secured by all the three approaches on the KITTI dataset 38

6 Evaluation 41
6.1 MobileNetv2 vs DenseNet for the encoder 41
6.2 Pix2Pix vs the rest 44
7 Conclusion and Future work 47
8 References 48
Appendix 50

LIST OF FIGURES

Figure 1: The idea behind LIDAR 8
Figure 2: A typical Convolutional Neural Network 10
Figure 3: A sample U-net 11
Figure 4: Overview of working of a GAN 13
Figure 5: Samples of images (left) and depth maps (right) pairs from the KITTI dataset 18
Figure 6: Samples of image and depth map pairs from the NYUv2 dataset 19
Figure 7: The appearance of a U-net 21
Figure 8: Generator Architecture 22
Figure 9: Discriminator architecture 23
Figure 10: Images(left), generated depth maps (centre) and ground truth maps (right) 24
Figure 11: Architecture of Densenet169 model 25
Figure 12: Architecture of approch 2- U-net with DenseNet encoder 26
Figure 13: MobileNetv2 architecture 29
Figure 14: Approach 3- U-net with MobileNetv2 encoder 30
Figure 15: SGBM architecture and data flow 33
Figure 16: stereo image pairs (top) and generated depth map (bottom) for stereo pair 1 34
Figure 17: stereo image pairs (top) and generated depth map (bottom) for stereo pair 2 34
Figure 18: stereo image pairs (top) and generated depth map (bottom) for stereo pair 3 34
Figure 19: stereo image pairs (top) and generated depth map (bottom) for stereo pair 4 35
Figure 20: stereo image pairs (top) and generated depth map (bottom) for stereo pair 5 35
Figure 21: Graphical user interface 36
Figure 22: Outputs stored in the output folder 37
Figure 23: Each row consists of the input image, ground truth depth map, output by pix2pix model, output by U-net with MobileNetv2 and output by U-net with Densenet encoder, respectively from right to left 40
Figure 24: Example scene 1 for comparison of approaches 2 and 3 - Input image (left), Depth map generated by U-net with MobileNetv2 encoder (centre) and depth map generated by U-net with Densenet (right) 42
Figure 25: Example scene 2 for comparison of approaches 2 and 3 - Input image (left), Depth map generated by U-net with MobileNetv2 encoder (centre) and depth map generated by U-net with Densenet (right) 43
Figure 26: Example scene 3 for comparison of approaches 2 and 3 - Input image (left), Depth map generated by U-net with MobileNetv2 encoder (centre) and depth map generated by U-net with DenseNet (right) 43

Figure 27: Example scene 1 for analysis of approach 1 - Input image (left), Ground truth depth map (centre) and depth map generated by Pix2Pix model (right) 44
Figure 28: Example scene 2 for analysis of approach 1 - Input image (left), Ground truth depth map (centre) and depth map generated by Pix2Pix model (right) 45
Figure 29: Example scene 2 for analysis of approach 1 and comparison with approaches 2 and 3- Input image (left), Depth map generated by U-net with MobileNetv2 encoder (centre) and depth map generated by U-net with Densenet (right) 45
Figure 30: Depth estimator GUI dataflow..... 50
Figure 31: Depth estimator GUI welcome screen..... 50

LIST OF TABLES

Table 1: Comparison of the state of the art approaches in the NYU and KITTI datasets 16
Table 2**:** MSE and SSIM values along with Ground truth (left), generated depth map in greyscale (centre) and generated depth map in RGB (right) for images 1-5 from Test 1 for the U-net with Densenet encoder..... 28
Table 3: MSE and SSIM values along with Ground truth (left), generated depth map in greyscale (centre) and generated depth map in RGB (right) for images 1-5 from Test 1 for the U-net with MobileNetv2 encoder 33
Table 4: Scores on the KITTI dataset..... 39
Table 5: Comparison of approaches 4.2.2 and 4.2.3 on the KITTI and NYU datasets..... 41
Table 6: Comparison of the Pix2pix model and the U-net with MobileNetv2 encoder 46

1 INTRODUCTION

Depth estimation is a fundamental consideration that is taken into account in numerous domains. Self-driving cars need accurate and fast depth information of their surroundings. They use depth information to reconstruct three-dimensional maps of their environment. The speed and accuracy of this information is key to the operation and safety of these cars.

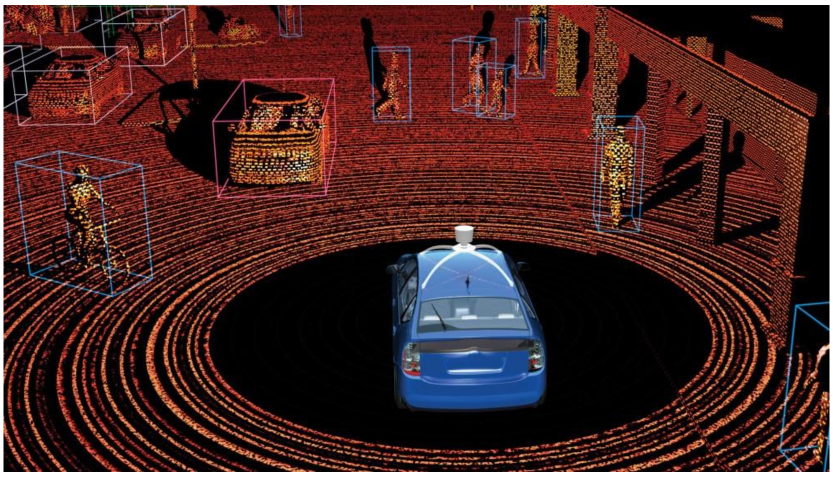


Figure 1: The idea behind LIDAR

Often, LIDAR is used for this purpose. However, LIDAR is very expensive and hence is one of the key problems that encumber the worldwide flourishing of self-driving cars by directly contributing to compromising their affordability.

Furthermore, depth estimation is also used extensively in the domain of robotics. This could be just for analysis and positioning in their surroundings or for a specific reason such as depth estimation for robot-assisted surgery. The depth information needs to be precise here as any wrong reading could potentially result in the loss of human lives.

Another interesting example of depth estimation is the automated 2D to 3D conversion for movies. Shadow mapping and ray tracing in computer graphics and games is also a similar use case where the usage of deep learning can save a lot of time and money.

1.1 MOTIVATION FOR DEEP LEARNING IN DEPTH ESTIMATION

At the moment, LIDAR is the most widely used depth estimation technique. LIDAR, also known as, Light Detection and Ranging, is a depth sensing technology that uses light to measure depth. It has a receiver that measures the time the light takes to reflect off the target and return to it. This time is used to calculate depth information.

Despite being expensive, LIDAR systems are not accurate in cloudy and dark weather conditions and are affected by high sun angles and reflections. Moreover, the technology itself offers very low value for money. It works in low altitude only and collects a lot of complex data which can get very tricky to analyse and gather information [4].

Deep Learning methods, on the other hand, eradicate all these problems. They use cameras which are a lot cheaper comparatively. They're a lot faster as well. Sufficient training and data can also make them stunningly accurate. Hence, well-developed methods can potentially give better results than LIDAR and even eliminate it completely in the future.

There are some deep learning depth estimation approaches out currently. However, these are far from perfect. They require a large amount of training data and expensive hardware for training and production of results. Their architectures are also very complex, making them even harder to adopt.

In this project, I introduce methods, that were generated using low-end hardware and free services and manage to generate depth maps similar in quality and precision, to those generated by the state of the art model.

1.2 Report Structure

The Report starts with a thorough study and review of the domain. The currently available approaches are discussed and ranked. Following this, the data and technology used are discussed. The design and implementation and a few outputs along with results are shown in the methodology section. The Results section consists of a table with metric values of all three approaches. The Evaluation section then discusses these metric values from the results sections and other observations. A few more comparisons were also performed for specific evaluations and hence included in the evaluation section rather than the result sections. It first compares approaches 4.4.2 and 4.2.3 and then compares the better one with the Pix2pix model. Then the conclusion is stated along with the potential future developments of this project.

2 Literature Survey

2.1 Background and Theory

2.1.1 Types of Depth Estimation

The task of depth estimation can be accomplished in predominantly two ways. These are Monocular and Stereo depth estimations. Stereo Depth estimation is the process of estimating depth using a pair of images or stereo images. The ideology behind this is analogous to the way our eyes perceive depth. Monocular depth estimation on the other hand involves estimating depth using a single image.

It is well-known that we cannot determine depth with just one eye. This is due to the fact that it is impossible to estimate depth accurately with just one image. However, we can approximate depth information with one eye closed as we have a lot of experience with estimating depth. This is the idea behind Monocular depth estimation. The deep learning models are shown and made to learn from a large collection of images and corresponding depth maps. The model's weights are updated using these samples and consequently, they are able to generate astoundingly accurate depth maps with just one image.

More than two images can also be used to estimate depth. However, increasing the number of input images to more than two for just the generation of just one depth map might dramatically increase the data collection costs while not having any significant impact on the output. Even Monocular approaches manage to produce stunning outputs as shall be shown later, and ergo, there is no real reason to venture beyond the Stereo approaches.

To fully understand the work done in this project, involves a thorough understanding of concepts from a variety of domains. A brief overview of these domains and concepts has been covered in the next few sub-sections.

2.1.2 CNNs

Convolutional Neural Networks are a type of Artificial Neural Network that are commonly applied to visual tasks. These networks could be pictured as a regularized version of Multi-layer perceptron networks. This is because the neurons in CNN receive only a small subset of adjacent inputs (inputs in a convolution) rather than receiving all the inputs due to the fully connected layers. This helps in optimization and helps combat overfitting [26]. A typical CNN architecture is shown below in figure 2 [27].

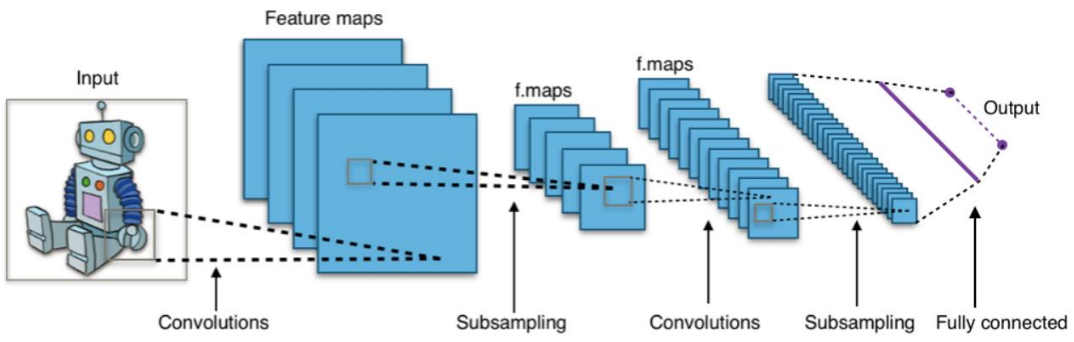


Figure 2: A typical Convolutional Neural Network

2.1.3 U-Nets

All the monocular depth estimation approaches include a U-net architecture, either as a component or the whole architecture itself. U-net is simply a type of convolutional neural network which was tailored to deal with image segmentation tasks, specifically in the biomedical domain [16]. Using a U-net architecture greatly brings down the large number of training samples that would be otherwise required. This is done by using a contracting path for capturing the context. Also, an expanding path that is symmetric is used for decoding, facilitating accurate localization.

The U-net architecture is divided into two major components. These are the encoder and the decoder. The fundamental concept behind U-net is that the feature mapping that is learnt when the encoder converts the image into a vector is the same mapping that is used by the decoder to convert the vector back to an image [17]. Doing so helps retain the integrity of its structure, consequently bringing down the distortion and enhancing the structural similarity score.

A U-net can be visualized as follows in figure 3 [20].

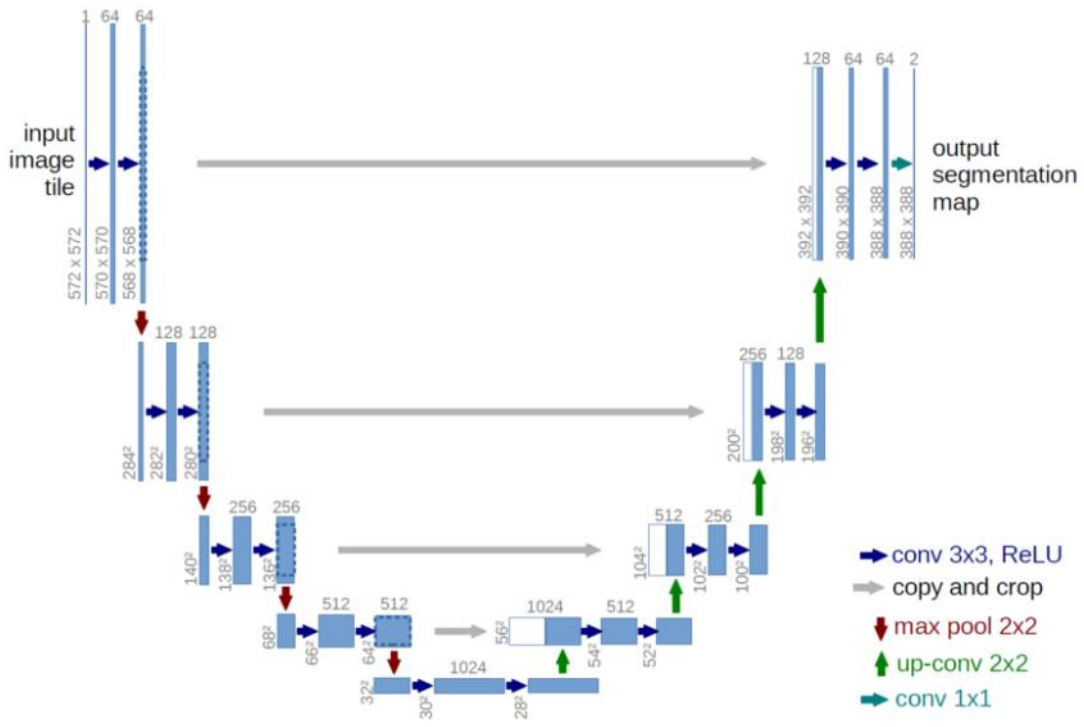


Figure 3: A sample U-net

The input is first contracted step by step. This is then passed through the mediating layer at the bottom, known as the bottleneck layer. Following this, the intermediate representation is expanded through upsampling blocks and eventually, the result is produced.

All the three monocular depth estimation approaches in this work use U-net architectures which vary to a great extent in the architecture of their encoders and decoders.

2.1.4 Transfer learning

Transfer Learning is an innovative technique that has gained popularity in recent years. It involves using pre-trained model weights as a starting point rather than using techniques such as Xavier initialization or simply using zeroes or random values as the initial weights. The idea behind this is to basically transfer knowledge from one task to another analogous task.

Doing so has given better results in numerous situations. The outputs were more accurate and the training time and consequently the computational expenses were also greatly reduced.

It is also perfect for those scenarios where sufficient training data is available only in another domain of interest and not exactly available for the task at hand. Transfer learning can then be used to transfer knowledge rather than spending a lot of resources on data collection and/or synthesis [19].

However, it is to be noted that if the source and the target domain are unrelated, it may lead to negative transfer learning, which would cause the performance and accuracy to plummet [18].

To use transfer learning in the development of a deep learning solution, first, the weights of the pre-trained model are loaded. Following this, the model is fine-tuned using samples that are similar to the data that the model is intended to be used on later. The advantage of doing this is that the number of epochs used, need not be as high as it needs to be when being trained from scratch. Moreover, during the fine-tuning phase, the weights are updated for an arbitrary set of layers. This could be just a few of the output layer, the last few layers, or the whole network, thus enhancing the flexibility.

This concept is employed for two approaches in this project, namely the U-nets with MobileNetv2 and Densenet as the encoders respectively.

2.1.5 GAN

GANs or Generative Adversarial networks are a category of artificial intelligence algorithms called implicit generative models and are aimed at solving the problem of Generative modelling. The objective of Generative modelling is to predict the probability distribution that was used to generate the given set of samples [21].

The advantage of GANs is that they provide a way to learn deep representations without having to use data that is extensively annotated. This is achieved in a game-theoretic approach in which two networks are trained and made to compete against each other [23]. The backpropagation values are then obtained based on this competitive process [22].

These two models are called the Generator and the discriminator. The task given to the generator is to produce samples that closely resemble the given data, in order to deceive the discriminator and make it think that the generated sample was from the given data. The objective of the discriminator is to classify a given example as real or fake and to not be deceived by the samples generated by the generator. Before being given samples from the generator, it is first trained on a given training dataset until its accuracy reaches a satisfactory amount.

Despite being powerful and highly accurate, GANs are prone to a problem known as mode collapse. In this, the GANs end up skipping a subset of input spaces. The cause of this is weak generative models that fail to notice this omission. Hence, this problem can simply be solved by better and scrupulous training of the discriminator before it is made to compete with the generator

The working of a GAN is depicted in figure 4 below.

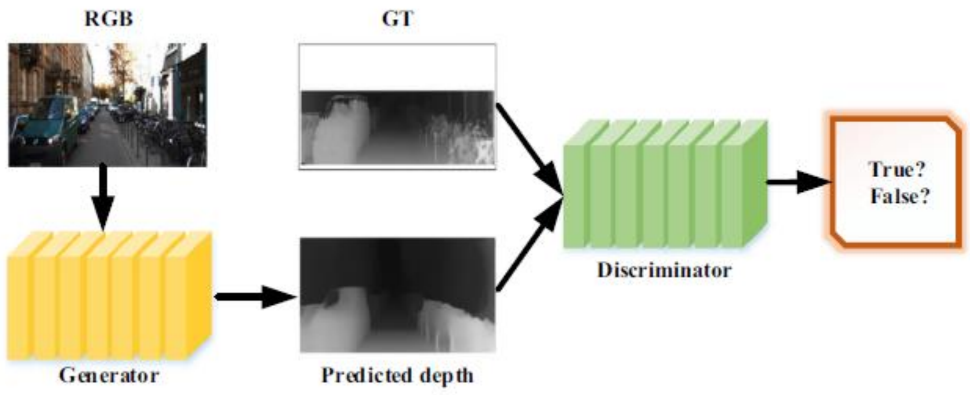


Figure 4: Overview of working of a GAN

2.1.6 SSIM and MSE loss

SSIM or Structural Similarity Index is a metric that perceives the preservation of an image's structural quality when it is transformed. Rather than judging simply based on the pixel values like Mean Squared Error and Peak Signal to Noise Ratio, SSIM takes into account, the fact that the spatially near pixels have interdependencies [24]. Hence, it is a more stable and reliable metric when compared to other similar ones.

The SSIM for any images x and y is calculated using their means and covariances as follows:

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

Mean Squared Error, as the name implies, is simply the mean of squares of error of a model and is the most commonly used absolute error metric. Taking squares instead of the absolute errors ensures that the result is always positive, making it easier to analyse and compare approaches [25].

The MSE for two images is calculated as follows:

$$\mathrm{MSE}=\frac{1}{n}\sum_{i=1}^{n}(Y_i-\hat{Y}_i)^2$$

PNSR is also an absolute error metric. However, it has not been used in this study as MSE has been used and using another absolute error metric would make it redundant.

2.2 RELATED WORK

Some research has already been done in the domain of depth estimation. Based on the training procedure employed, these research works can be divided into three types, namely, supervised, semi-supervised and unsupervised approaches. In the supervised learning domain, the work can be further classified into vanilla CNN based methods, RNN based methods and GAN based methods. In the Vanilla CNN bases methods, Li et al. developed a double streamed approach that made use of the VGG 16 Convolutional Neural Network. The first stream was engineered for depth regression and the second, for depth gradient information. A depth gradient fusion block was used to combine these steams and the final depth map was generated [28]. Depth loss and gradient loss were both considered to strengthen each stream's generalization power.

Alhashim et al. used Densenet to estimate depth. In this, an encoder-decoder architecture was used, with the DenseNet pre-trained model as the encoder. The decoder consisted of four upsampling blocks. Their approach gave great results and had a lower error rate when compared with models that were trained from scratch. The main contribution and claim by this work were that it produced sharp and detailed depth maps which were much sharper than the other state of the art approaches. The training time was also lesser as a pre-trained model was used and fine-tuned, rather than being trained from scratch [10]. This approach was one of the best approaches discovered during the literature study and the U-net with DenseNet approach in the Monocular depth estimation approaches of this dissertation, was in fact inspired by this work.

Lee et al. came up with an approach that used relative depth maps at different time scales. These depth maps were then decomposed and recombined to generate a final accurate depth map [29]. Despite the fact that relative depth estimation approaches are much less accurate when compared to the conventional absolute monocular depth estimation approaches mentioned above, this particular work managed to come very close to other absolute depth estimation works and even managed to outperform some of them. However, the other relative depth estimation approaches simply could not match up to the conventional absolute depth estimation approaches in terms of accuracy. But the former did prove to be slightly more robust to data homography [30].

The concept of CRF, which stands for Conditional Random Fields, was also applied to depth estimation by a few research works. CRF is a method of statistical modelling that takes context information into account rather than directly predicting the output by considering adjacent information. The basic idea is that the networks learn from patches. This information is then sent to a CRF module to obtain the final depth maps [30]. This is known as CRF cascading. Xu et al., in their paper titled 'Structured attention guided convolutional neural fields for monocular depth estimation' employed this CRF cascading concept to an attention model that picked robust features that were multiple scaled, on its own. This aided in reducing the root mean squared error [31].

Recurrent Neural Networks have also been used for the task of depth estimation. ConvLSTM, which is a type of Recurrent Neural Network, is a network engineered for Spatio-temporal data prediction. Kumar et al. engineered the DepthNet, which was build using ConvLSTM components. The encoding block consisted of eight ConvLSTM layers which enabled the network to sequentially use temporal data. Moreover, the convolutions helped preserve inter-space dependencies [32].

The final and the most apposite type of network for the task of depth estimation is GAN. GAN or a Generative Adversarial Network was used by Li et al. Sparse point clouds in the input were converted to dense depth maps by using Conditional Adversarial Networks [33].

Richard Chen et al. further innovated and improved this approach by employing adversarial learning of non-local loss function that is contextually aware [34].

It is a known fact that all the approaches above, being supervised learning methods, have one thing in common, which is that they all require a large amount of training data. Obtaining stereo pairs or image and ground truth map pairs is also a highly expensive and long process. It demands a large number of personnel, costly equipment such as LIDAR (for gathering data), time and a lot of appropriate scenes to capture. Privacy concerns also come up quite often, making this process even more time consuming and challenging.

Hence, Zheng et al. used synthetic image pairs to tackle this problem. They developed two networks, the first of which was an image to image translation with the objective of enhancing realism and the second was a final depth prediction network. The first network produced marginally modified real images by taking in synthetic images and the second network was then trained on loss pertaining to the synthetic image and ground truth depth pairs. They claim that this technique gave great results and even outperformed a good number of networks that were trained on actual pragmatically gathered data rather than synthetic data [35].

2.3 Shortlisting and critical analysis of the state of the art approaches

The domain of depth estimation was further explored extensively before the development phase. This was done by reviewing and analyzing other research studies and research works included in them. KITTI and NYU depth datasets were mainly looked at while comparing approaches based on numerical metrics. Three approaches stood out during this process. The first of these was the Dense Prediction Transformer by Ranftl et al. [37], which was a recent solution. A Vision Transformer was used in this work rather than a CNN, making the approach stand out from the rest. The approach collected intermediate representations from various stages and then used a Convolutional decider which made of use the collected information and put together a complete depth map. The vision transformer maintained high resolution throughout its operations and had a receptive field after stages. This helped generate sharper and predictions that were collectively coherent, when compared to conventional Convolutional Neural Network approaches. Ranftl et al. claimed that the approach had improved over the performance of the state of the art Convolutional Neural Network based approaches by over 28%. Its performance on the NYU dataset was record-breaking and it is the best approach in terms of raw numerical metric scores. Specifically, this was achieved using the DPT-Hybrid variant which generated feature maps with 256 dimensions. Its scores on the KITTI dataset were also outstanding.

Ada Bins, short for Adaptive bins, is another reputed approach that produced state of the art results for the task of Monocular depth estimation [38]. At its core, Ada Bins used an encoder-decoder architecture. However, an interesting addition was made to it. Bhat et al. used a transformer block that divided the ranges of depth into ‘bins’. The central values of these bins were evaluated adaptively, for each image. The final depth map was obtained by taking a linear combination of all the central values of the bins. Hence the name Ada Bins.

Bhat et al. claimed that their approach surpassed and showed significant improvement over the other state of the art approaches at the time of publication, which was near the end of 2020. The approach managed to produce the best results on the KITTI dataset, superseding all the other state of the art models. The performance on the NYU Depth v2 dataset was also top-notch and it was seconded by none other than the work done by Ranftl et al.

The third shortlisted approach used the U-net architecture as its core. Several additions were made to this U-net model. Wu et al. added an optical layer before the input reached the U-net [39]. This optical layer consists of a model that focuses on the physical aspect. It aided with the Point Spread Functions, taking in a phase mask that is learnable. These Point Spread Functions were then used with the input to give the image that will be used as input to the U-net. The U-net then generated the depth map using the output of the optical layer.

Another outstanding approach chosen was the work by Alhashim et al. [10]. This approach used a U-net architecture and used a DenseNet as the encoder. Due to this change, the depth maps produced by the model were sharper and clearer than the other state of the art approaches. Moreover, this approach was much less computationally intensive compared to the other approaches in the state-of-the-art shortlist. The reason behind this was that the approach used a pre-trained model of DenseNet as the encoder and hence, the training process was shorter when compared to training from scratch. The approach produced relatively sharper depth maps for much lesser training time and hence the value of time spent training was much higher, making this perfect for low budget hardware and projects requiring quick development. Due to these advantages, the first approach of the project, described in the later sections, was inspired by this work by Alhashim et al.

A few other Monocular depth estimation solutions, which used the U-net, were also explored and studied such as the work by Zhou et al., who used a U-net and enhanced the same with a spatial pyramid along with a Super Resolution Net. A clip loss function was also used along with this. However, they were all superseded by the solution developed by Wu et al.

The performance of the above three state of the art approaches is shown in the table below.

Approach	**NYUv2 (RMSE)**	**KITTI (Absolute Relative error)**
DPT-Hybrid [39]	0.357	0.058
Ada Bins [38]	0.364	0.062
U-net	0.382 (Wu et al. [39])	0.093 (Alhashim et al. [10])

Table 1: Comparison of the state of the art approaches in the NYU and KITTI datasets

The DPT-Hybrid and Ada Bins were both highly innovative approaches and produced stellar results. They are seconded by none in NUUv2 and KITTI datasets respectively. However, their architecture is highly complex and computationally intensive. Ergo, these approaches come with inherent high development time and costs, making them relatively unsuitable for low budget and one time use projects. The whole idea behind using deep learning solutions is to make the development phase cheaper and quicker. Despite performing exceptionally, these approaches void the whole point of deep learning solutions due to their complexity.

The U-net approach, on the other hand, gave results that were outstanding. The results produced were so close to the DPT-Hybrid approaches and the Ada Bins approaches, that the difference was negligible. The U-nets were also less computationally expensive. Alhashim et al. further brought down the development time by using a pre-trained DenseNet model. This brought down the training time as there was no need to train from scratch and it was shown that the performance was the same, if not better.

Hence, this approach was chosen for development and further research in this project. The approach in 4.2 is inspired by this work by Alhasim et al.

On further exploring, the Pix2pix model was discovered. This work by P. Isola et al. performed better than any other solution on the Cityscapes dataset. The computational and overall complexity was also low, especially considering the accuracy and precision of the results [5]. Hence, this approach was also chosen and implemented as shown in the later sections.

Finally, an interesting concept termed multi-task learning was also applied to depth estimation. The idea behind this is that, if a neural network is trained on two tasks rather than one, it sometimes performs better on both tasks, when compared to training separately. It was discovered that using Multi-task learning and training a network for depth estimation and semantic segmentation would generate a strong model [41]. However, only the performance of semantic segmentation was improved, but the difference in the performance of depth estimation was negligible. Due to this reason, this approach was not implemented as it would just increase the computational and overall complexity without any significant boost in the quality of results.

3 Data and Requirement Specification

3.1 Data used

The first key step to implementing a Deep Learning solution is to find the right data. Gathering data especially for depth estimation is a very complex task and can take years to do. However, this has been done by other researchers and certain datasets with outstanding quality are available online, to academic researchers, for free. Two of these datasets namely KITTI depth prediction [6] and NYUv2 depth Dataset [7] have been used for this dissertation.

3.1.1 KITTI Depth Dataset

This dataset consists of sequences captured from a vehicle in motion, along with depth maps that were obtained using LIDAR. It consists of over 90 thousand maps and hence is perfect for training and evaluating deep learning models. A sample from the dataset is shown below.

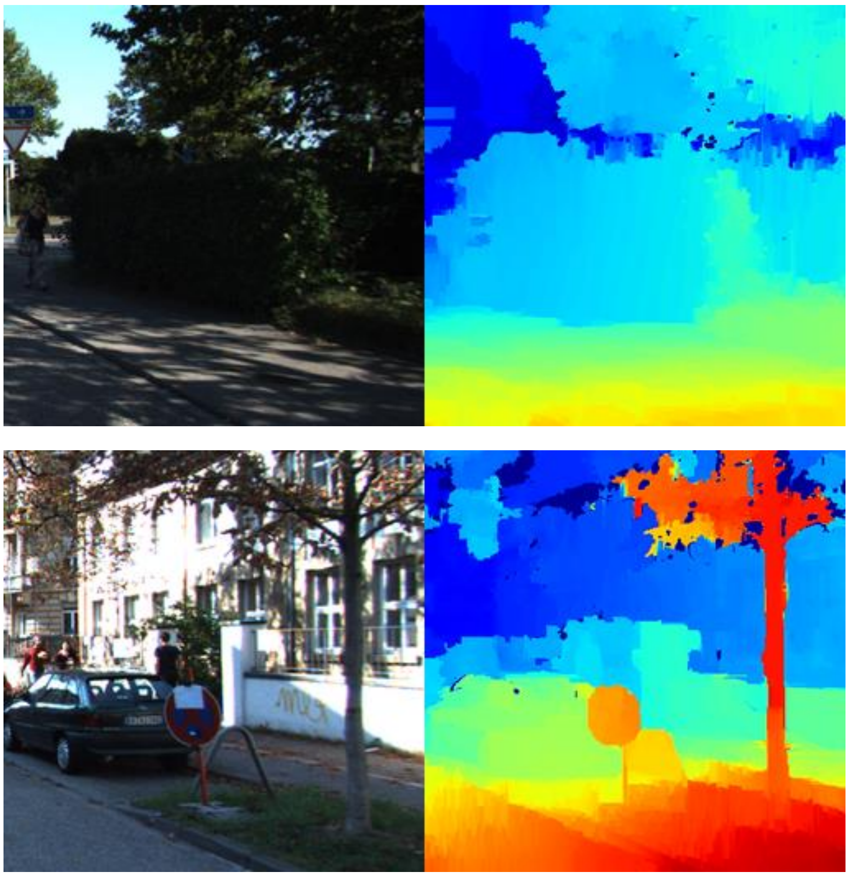
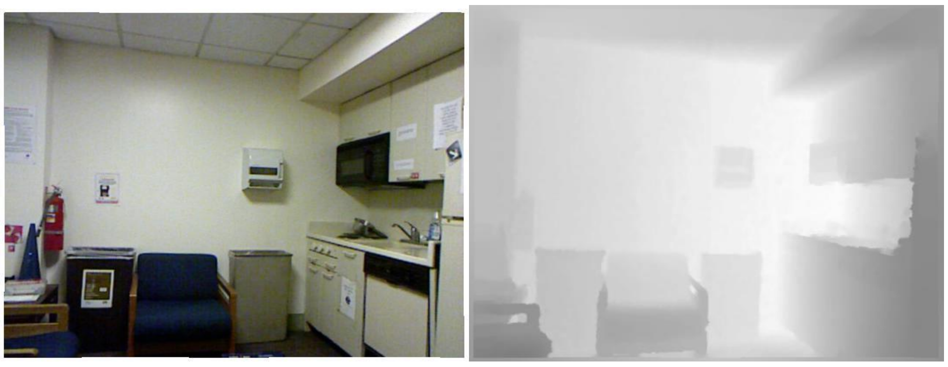


Figure 5: Samples of images (left) and depth maps (right) pairs from the KITTI dataset

3.1.2 NYUv2 depth dataset

The NYU dataset, on the other hand, consists of sequences recorded from eclectic indoor scenes. The depth maps and the RGB images were captured using Microsoft Kinect. The dataset consists of 1449 labelled dense pairs of aligned RGB and depth maps which were also pre-processed to fill in the missing depth information. These were used for training purposes. Samples have been shown below.



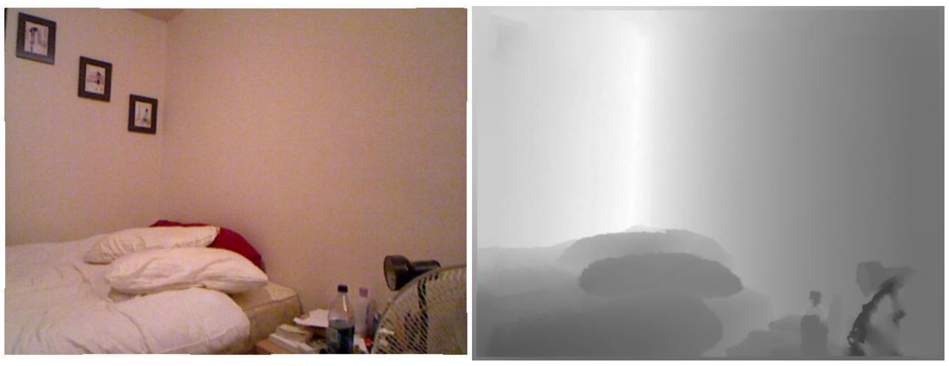


Figure 6: Samples of image and depth map pairs from the NYUv2 dataset

In addition to these, the dataset also contains over 400,000 unlabelled frames. The raw depth along with accelerometer and depth data from Kinect has also been provided coupled with a toolbox for easy manipulation.

3.2 TECHNOLOGY USED

A list of software and other technology used for this work has been stated below.

- Python
- Tensorflow
- Keras
- Pytorch
- Jupyter Notebook (local and Kaggle kernels/Google Colabs)
- Laptop with Ryzen 7 4800 H and Nvidia GeForce GTX 1650 Ti

4 METHODOLOGY AND IMPLEMENTATION

4.1 DATA PRE-PROCESSING

In most deep learning projects, the data cleaning and pre-processing phase is quite an elaborate and time consuming one. However, The KITTI and the NYUv2 Depth datasets used for this project are maintained outstandingly well and hence, minimal data pre-processing was required. The NYUv2 Depth dataset comes in a '.mat' file format. A python function was used to extract this and separate the images and corresponding depth maps into separate folders while preserving the names.

The images from the KITTI dataset were used to train the Pix2Pix model. This model, being relatively more computationally intensive, was trained online. Hence, the images were converted and stored in numpy array files for convenience. A few images were set aside for testing purposes and were not included in the training sets. These images were used for testing and the results have been shown in later sections.

It is to be noted that the KITTI dataset consists of outdoor images and the NYU dataset consists of indoor images. Hence, both these datasets were used to cover a wider spectrum of use cases for depth estimation.

4.2 Modelling

Three separate models were developed to tackle the task at hand. These are listed below.

1. The Pix2pix model
2. U-net with a pre-trained MobileNetv2 as the encoder incorporated using transfer learning
3. U-net with a pre-trained DenseNet169 as the encoder incorporated using transfer learning

4.2.1 The Pix2pix model

4.2.1.1 Theory

The no-free lunch theorem states that there is no single model that will always do better than all the other models, without having substantial information about the modelling problem. Despite this fact, the Pix2pix model can be thought of as the best image to image translation algorithm as it is widely used, fast and accurate.

Pix2pix model is basically a Conditional Generative Adversarial network that was specifically designed for image to image translation. The architecture consists of two components. First, a generative model that generates fake examples that closely resemble the samples in the training set. Second is the discriminative model, which decides if the image given to it is real or fake.

Both, the discriminator and the generator are trained simultaneously in an adversarial manner. The generator aims to generate examples the are so close to the training set that they deceive the discriminator into thinking that they are real. Meanwhile, the discriminator tries its best to not be deceived by the generator. In the image to image translation scenario, the generator is given an image and it generates the output (depth map) based on this. The discriminator is given the original image and the generated output, and it determines whether the pair is plausible. It is to be noted that the generative model's updating and learning depend on the discriminator model. The generator is then evaluated and updated using a combination of adversarial loss and L1 loss between the generated sample and the ground truth.

This competitive adversarial training approach was proven to be an effective technique and gave similar, if not better, results to training them separately, while eliminating the need for hand engineering the involved functions [5].

4.2.1.2 Implementation specifics

The architecture of this neural network was inspired by the work done by P. Isola et. Al [5]. Keras was used for the implementation of this paper rather than Pytorch which was used by the authors of the paper. Keras was chosen instead of PyTorch due to its concise architecture and much better readability. Keras is slightly slower than PyTorch as it is a high-level API. However, this model was trained on Kaggle utilizing the free GPU allocation. This service is computationally very fast and hence deemed the difference in training time as negligible. The architecture was made to generate colour images based on the input images.

As mentioned earlier, the model consists of a discriminator, which is an image classifying convolutional neural network, and a generator which is a U-net. The discriminator classifies a given image as real or fake. The fundamental principle here is that each prediction is mapped to a small patch in the input. This adds flexibility to the input image size, which greatly helps make pre-processing and experimenting with different datasets easier. The mean of all outputs corresponding to all patches is evaluated to determine the final result produced by the discriminator.

Since there are only two classes, binary cross-entropy was used to evaluate the loss and optimize the model. Moreover, the beta parameter of the Adam optimizer was used to reduce the intensity of the updates to the weights. This was done to ensure thorough and steady learning. Setting beta to 0.5 gave the best results.

On the other hand, the generator is based on a U Net encoder-decoder architecture. This works by downsampling the input image till the bottleneck layer dimension is reached and then the image is upsampled in the decoder. The encoder and decoder architectures are similar. Every encoding layer has a corresponding decoding layer, and these are connected via skip connections for better results. Removing the skip connections negatively affects the performance.

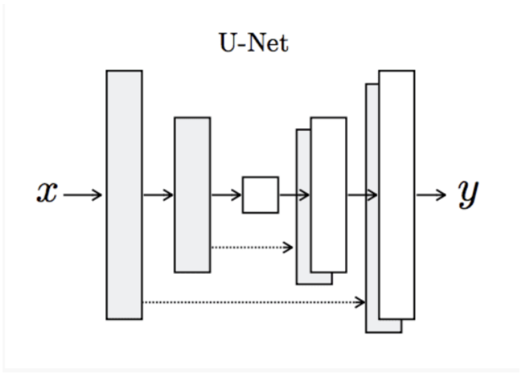


Figure 7: The appearance of a U-net

The encoder and decoder could be further modularized into blocks. Separate block functions were made for the encoder and the decoder. Tanh activation function was used for the output as it proved to be much more effective than the sigmoid activation function and is faster for training than the ReLU function [8]. The complete architecture of the Generator and the Discriminator is illustrated in figures 8 and 9 below.

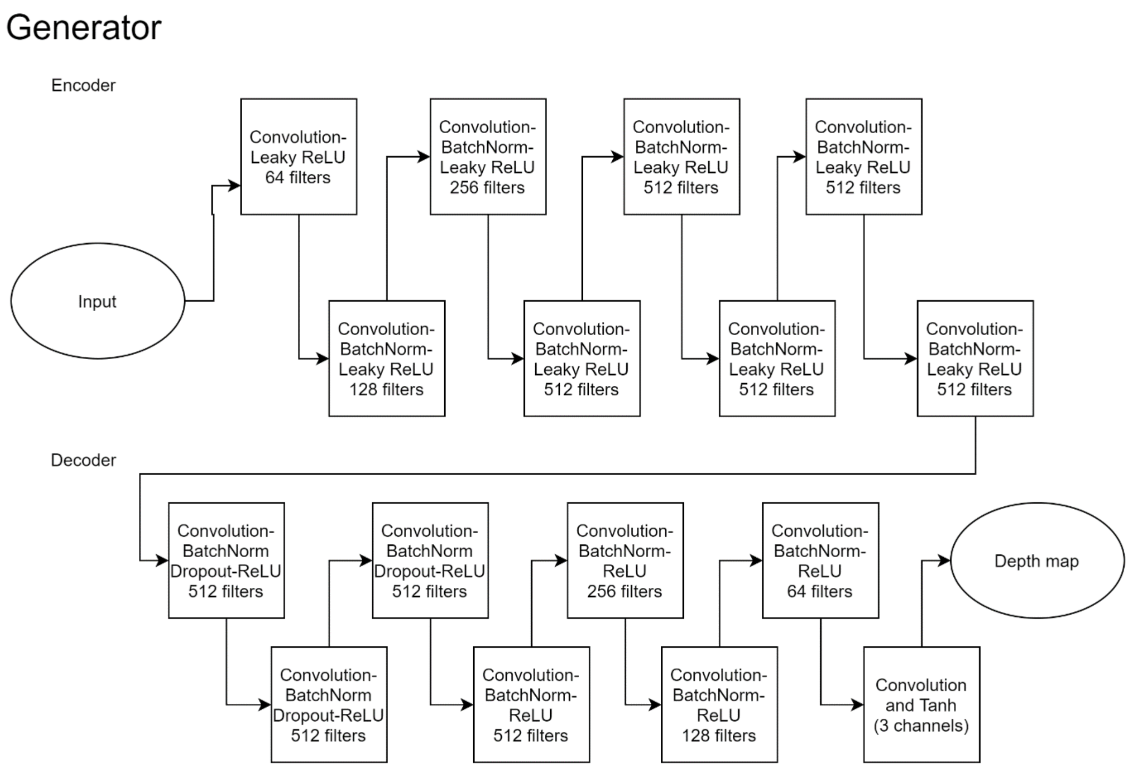

Generator
Encoder
Input
Convolution-Leaky ReLU 64 filters
Convolution-BatchNorm-Leaky ReLU 128 filters
Convolution-BatchNorm-Leaky ReLU 256 filters
Convolution-BatchNorm-Leaky ReLU 512 filters
Convolution-BatchNorm-Leaky ReLU 512 filters
Convolution-BatchNorm-Leaky ReLU 512 filters
Convolution-BatchNorm-Leaky ReLU 512 filters
Convolution-BatchNorm-Leaky ReLU 512 filters
Decoder
Convolution-BatchNorm Dropout-ReLU 512 filters
Convolution-BatchNorm Dropout-ReLU 512 filters
Convolution-BatchNorm Dropout-ReLU 512 filters
Convolution-BatchNorm-ReLU 512 filters
Convolution-BatchNorm-ReLU 256 filters
Convolution-BatchNorm-ReLU 128 filters
Convolution-BatchNorm-ReLU 64 filters
Convolution and Tanh (3 channels)
Depth map

Figure 8: Generator Architecture

The architecture of the generator is quite complex. It starts off with a Convolutional layer with 64 filters. This is followed by seven convolutional layers with leaky Relu and batch normalization. In order to downsample the image, the filters increase gradually. The next two layers have 128 and 256 filters respectively, followed by five layers with 512 filters. Then the intermediate representation is sent to the decoder. The decoder starts off with three convolutional layers with 512 filters along with batch normalization, dropout and Relu activation functions. Following this, the image is upsampled through four convolutional layers with 512, 256, 128 and 64 filters respectively, along with Relu activation functions and batch normalizations. Finally, the result is passed through a 3-channel convolutional layer with a tanh activation function to produce the final depth map.

The discriminator, being a classification CNN has a relatively simpler architecture. It starts with a 64 filter convolutional layer with Leaky Relu activation. This is followed by three convolutional layers with a doubling number of filters (128, 256 and 512 respectively), along with Leaky Relu activations and batch normalizations. Finally, a single channelled convolutional layer with Sigmoid activation is used to produce the output depth map.

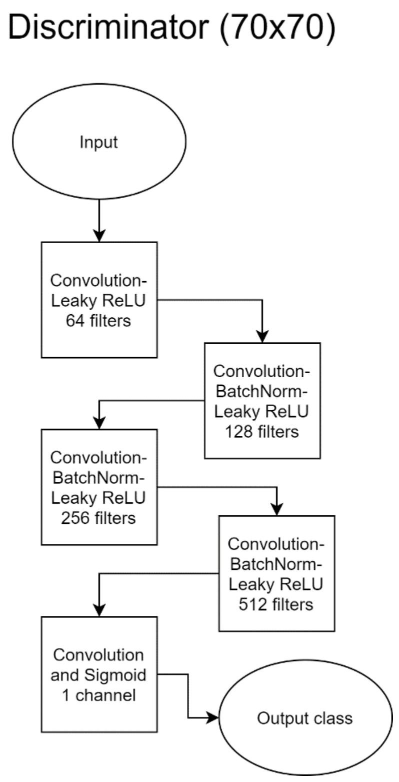

Discriminator (70x70)
Input
Convolution-Leaky ReLU 64 filters
Convolution-BatchNorm-Leaky ReLU 128 filters
Convolution-BatchNorm-Leaky ReLU 256 filters
Convolution-BatchNorm-Leaky ReLU 512 filters
Convolution and Sigmoid 1 channel
Output class

Figure 9: Discriminator architecture

The Pix2Pix model was then trained. Code was written such that the weights are saved every 10 epochs. 200 epochs were used by the authors of the Pix2Pix paper. However, 50 epochs were used for this work to aid in reducing the computational complexity. The generated results were highly accurate as shown in figure 10. The model was trained online using Kaggle Notebook services.

The steps taken in each training step are listed below:

1. Batch of real examples was selected
2. Generator generates a corresponding batch of fake samples
3. These batches of images are used, and the discriminator is then updated
4. The generative model is then updated with the class labels
5. Loss of each epoch is calculated to evaluate the progress. Two losses for the discriminator (on real and fake examples) and one loss for the generator (weighted average sum of adversarial and L1 loss) is calculated.

Just after the first 10 epochs, the generated images had a good amount of depth information. After 30 epochs, the images started to resemble the ground truth. After all the 50 epochs, it was hard to tell the difference between the ground truth and the generated depth maps on manual inspection. The MSE values were very low and the SSIM scores went as high as 0.96 and did not drop below 0.84.

A few samples generated by the approach are shown below in figure 10.

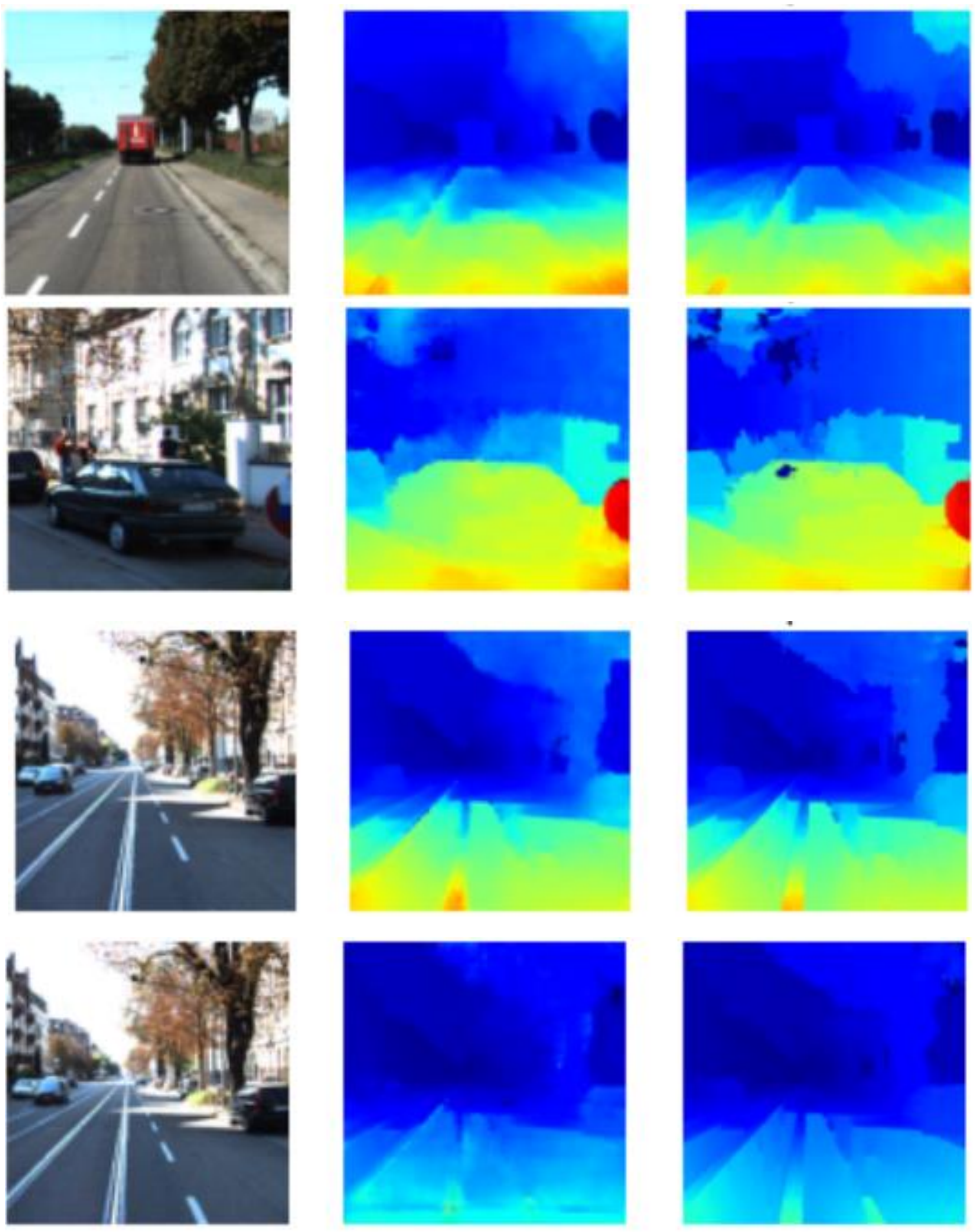


Figure 10: Images(left), generated depth maps (centre) and ground truth maps (right)

4.2.2 The U-net with Densenet as encoder

4.2.2.1 Theory

The existing depth estimation solutions produce blurry and unclear depth maps. To combat this problem, this approach uses a U-net architecture engineered using transfer learning, that closely resembles the architecture used by Alhashim et. al. [10]. The DenseNet169 pre-trained model was used as the encoder.

A Densenet is a type of Convolutional Neural Network which solves the problem of Exploding and Vanishing gradients by simplifying the connectivity patterns between layers. This is done by simply connecting every layer directly with the others and ensures maximum information flow. This way every layer can access the gradients from the loss function directly. The layer connections that facilitate this are known as Dense connections. Densenet has relatively narrow layers which contribute to a small set of fresh feature maps. The potential of the network is exploited through feature reuse which is facilitated by the aforementioned access to preceding depth maps, enabling the concept of collective knowledge [14]. All these developments enable the generation of sharp and clear depth maps.

Conventional Convolutional Neural Networks sum the output feature maps with incoming ones. But whereas this is not the case with Densenets. Densenet layers concatenate the output and corresponding input maps instead. Furthermore, Densenet is modularized into Dense blocks, which have the same dimension of feature maps throughout but differ in the number of filters between them.

In order to retain the feed-forward nature of Conventional Convolutional Neural Networks, the layers obtain additional inputs from all preceding layers and pass on the output to all subsequent layers. The architecture of DenseNet can be seen below in figure 11.

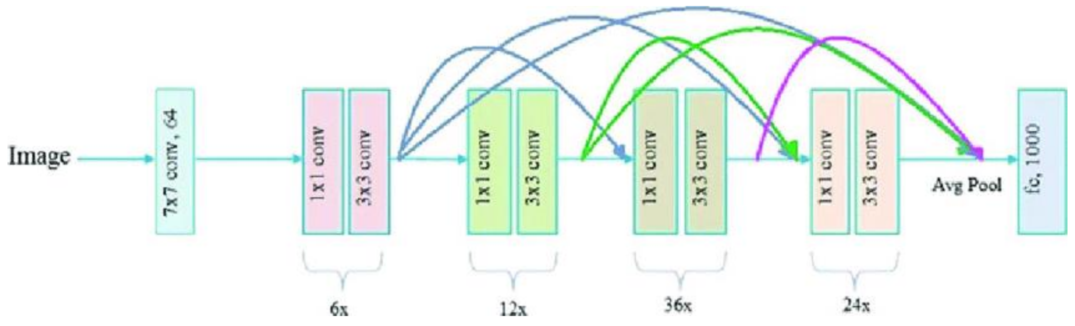


Figure 11: Architecture of Densenet169 model

4.2.2.2 Approach Architecture

The working of this approach is quite interesting. The input image is first taken in by the input layer and then given to the DenseNet pre-trained model. The Densenet encodes the image, and the intermediate representation is then passed through a convolutional layer. Following this is the decoding phase. The image is up sampled four times through up sample blocks. Each up sample block consists of an upsample layer. After each up-sample layer, the outputs are joined and passed through two convolutional layers, each of with are followed by a leaky ReLU function. The architecture can be seen in the diagram below in figure 12.

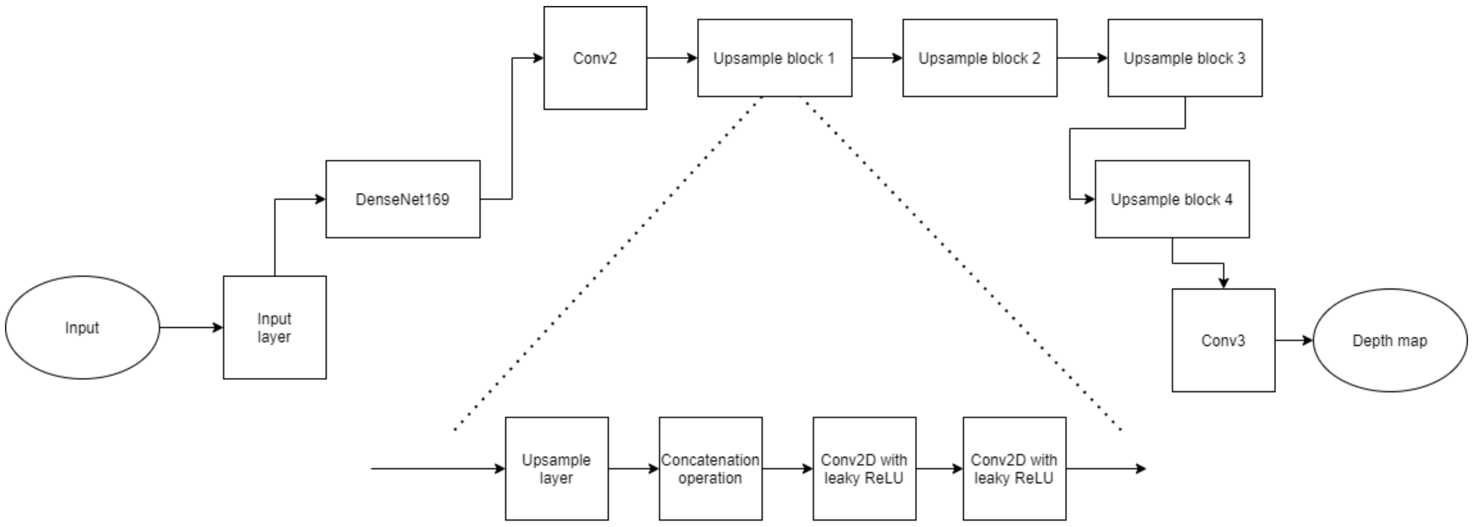


Figure 12: Architecture of approch 2- U-net with DenseNet encoder

For evaluation of loss and updating during training, two loss parameters were evaluated and given a weight to compute the final loss. These were the depth loss and the SSIM loss. Further information and explanation for this have been given in the next section on the U-net with MobileNetv2 encoder.

Adam optimizer was used to optimize the model while training. The reason for this is that the Adam optimizer was proved to be the best as it consistently performed better than the others such as SGD and RMSprop. Moreover, it is much less sensitive to the learning rate. Also, its adaptive optimizer for sparse data was perfect for the task at hand which was constrained by low computational power and development time [13].

4.2.2.3 Training specifics and evaluation

The training was done on a machine with the processor Ryzen 7 4800H and Nvidia GTX 1650 Ti as the GPU. The training took about 10 hours for 100 epochs. The weights were saved after each epoch for testing and experimenting purposes.

The exact parameters used are as follows:

1. Batch size of 32 was used. The loss was much lower when compared to other batch sizes. The samples also appeared much better than those of other batch sizes, via manual inspection
2. Learning rate was set to 0.0001. 0.001 and 0.0001 were tested as well. However, 0.0001 appeared to strike the right balance between speed and accuracy. Moreover, Adam optimizer is much less sensitive to the learning rate as mentioned above. Hence, the learning rate did not require further tuning.
3. The model was trained for 100 epochs. There was hardly any difference in RMSE and on manual inspection after this. Hence the training was stopped.
4. The model was then tested, and the results are shown below in table 2.

Image 1:

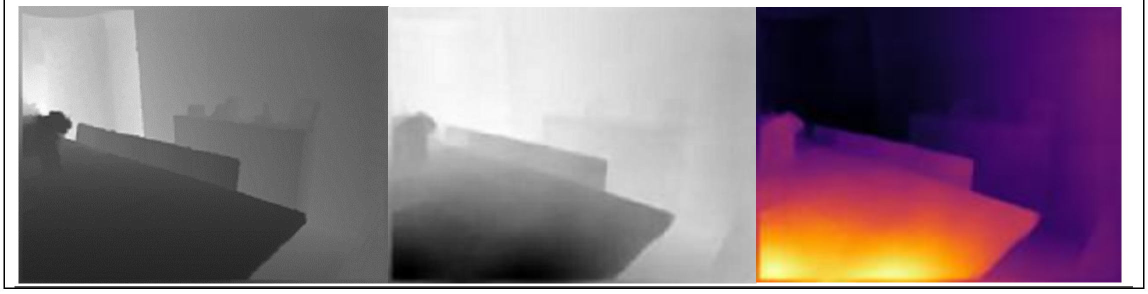
Epoch	30	50	70	90	100
RMSE	5918.8970345052085	5214.095361328125	5979.976715494791	6494.561259765625	6033.72638671875
SSIM	0.8060400393911671	0.7961140308441151	0.8050693938221732	0.7922766376035698	0.7966691116124766
					

Image 2:

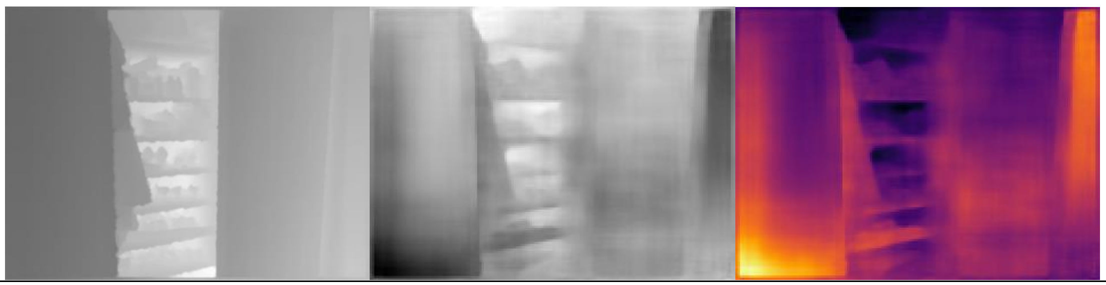
Epoch	30	50	70	90	100
RMSE	1752.5736295572917	2052.8995084635417	1936.5605794270834	2870.330556640625	1944.2880989583334
SSIM	0.8393851838054432	0.8140242195596431	0.822805085183414	0.8114409316284211	0.833819932114343
					

Image 3:

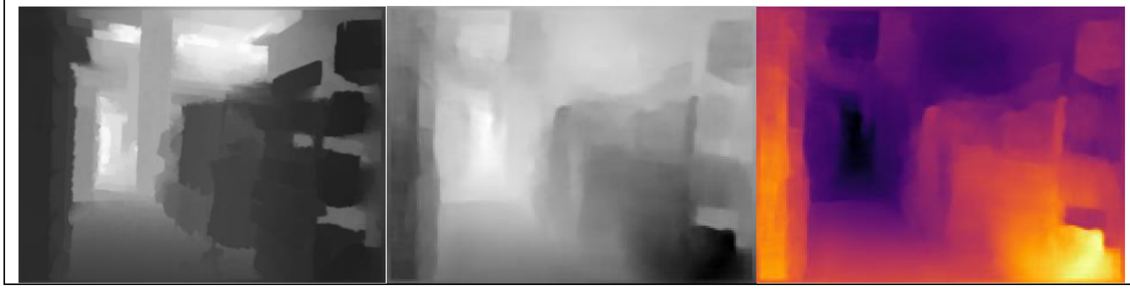
Epoch	30	50	70	90	100
RMSE	9303.484928385416	3443.49078125	9054.157457682291	6007.212265625	4160.359088541667
SSIM	0.6418728126221875	0.7298026591485959	0.6451461068670956	0.6905998949991458	0.7249024636535434
					

Image 4:

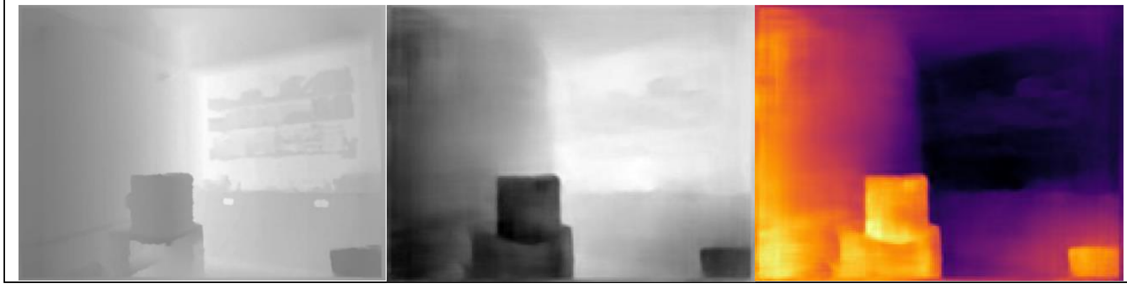
Epoch	30	50	70	90	100
RMSE	1598.702734375	1671.5722493489584	2095.683072916667	1830.2821647135418	2179.623955078125
SSIM	0.8774789385815166	0.8544342424885082	0.8413427480182176	0.8583259552425065	0.844887741091571
					

Image 5:

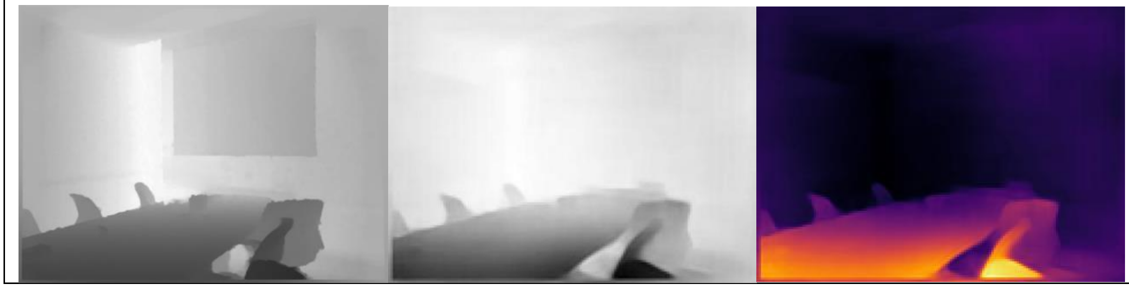
Epoch	30	50	70	90	100
RMSE	1746.82529296875	820.5782975260416	888.7268033854167	1806.2269108072917	1588.941787109375
SSIM	0.8936935485526499	0.9007481116452359	0.9132562522460091	0.9063656930479539	0.9072377006757703
					

Table 2: MSE and SSIM values along with Ground truth (left), generated depth map in greyscale (centre) and generated depth map in RGB (right) for images 1-5 from Test 1 for the U-net with Densenet encoder

4.2.3 The U-net with MobileNetv2 as encoder

4.2.3.1 Theory

This approach architecture is analogous to that in the previous approach. However, a pre-trained MobileNetv2 was used as the encoder instead of Densenet. Doing this enabled the reduction of training and validation loss by a significant margin when compared to training from scratch and relative to using the DenseNet encoder.

MobileNetv2 is a popular and widely used object detection network. It uses Linear Bottlenecks and Inverted residuals [9]. It was engineered to work on mobile devices, making it less computationally intensive. However, it has proven to be a great object detection network in non-mobile use cases as well. Being, an object detection network, it has demonstrated outstanding generalizing capability and hence has the potential to be a near-perfect encoder for depth estimation. The boundaries are captured better by it. Hence, using MobileNetv2 as the encoder resulted in sharper and clearer depth maps. The architecture of MobileNetv2 is shown below in figure 13.

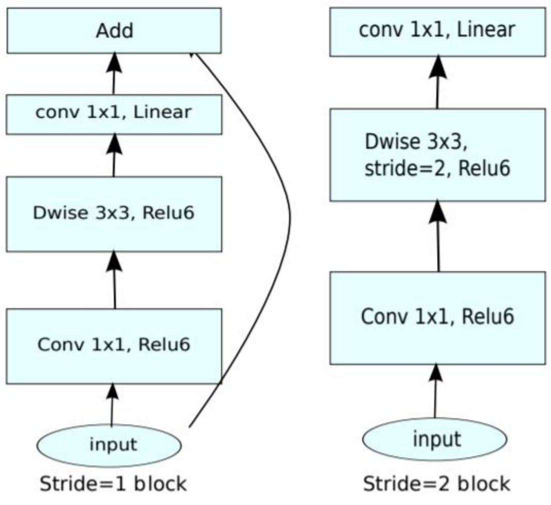


Figure 13: MobileNetv2 architecture

4.2.3.2 Approach Architecture

The encoder's architecture, as described above, just consists of the pre-trained MobileNetv2 with the extreme layers truncated. Pytorch was used to accomplish this. The encoders working involves simply taking in the input image and passing the same to the MobileNetv2. The Mobilenet then encoded the image. The encoded information was then passed through a convolutional layer. This Intermediate representation was then passed to the decoder for the generation of a depth map.

The decoder architecture is relatively more complex. It starts off with Conv2D 1x1 convolutional layer. The number of input channels was defined based on the input and the number of output channels was the product of the number of input channels and the specified width of the decoder which was a fraction set to 0.6.

The upsampling blocks followed this layer. A total of six upsampling blocks were used to upsample and generate a depth map. Each upsample block was initialized with two Conv2D layers with leaky ReLU activation function. Interpolation layer and leaky ReLU was used for forward pass. Following the sequence of upsample blocks was one convolutional Conv2D layer with kernel size specified as 3. This generated the depth map corresponding to the

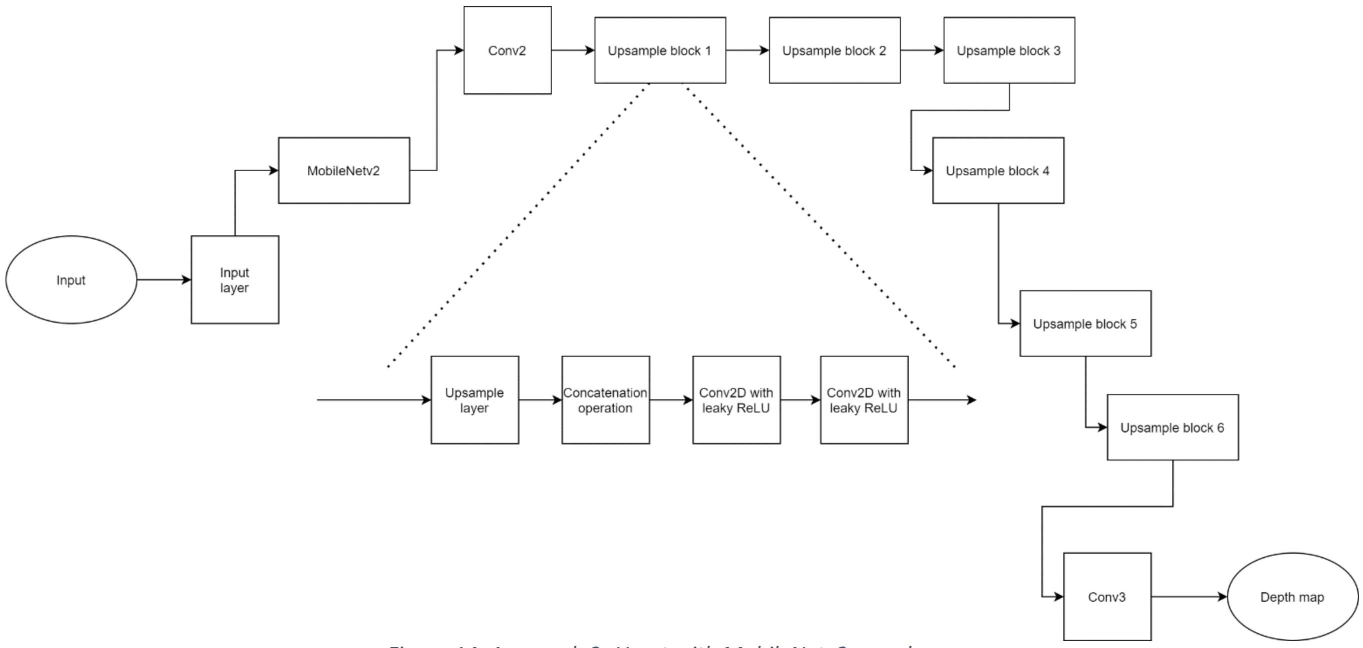


Figure 14: Approach 3- U-net with MobileNetv2 encoder

input which could then be saved using the matplotlib python library. The architecture can be seen in the diagram below in figure 14.

In the U-net architecture, it was determined that highly complex convolutional blocks did not necessarily enhance the results or the performance. It was determined that two convolution layers following bilinear upsampling layers performed outstandingly well and hence was the best choice [10].

Two losses were computed, namely the SSIM loss and the depth loss. The final loss was computed by a weighted sum of these losses.

The depth loss is a pointwise L1 loss that is based on the depth values. It was calculated as follows.

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

The Structural Similarity or SSIM was considered as it is a widely used and reliable metric for image translation tasks. This was verified for convolutional neural networks for the task of depth estimation [11]. The SSIM loss was the most informative and hence was given a larger weight**.** It was observed that giving a smaller weight to SSIM severely degraded the quality of the results. This was simply due to the fact that the approaches were trained on different datasets and hence the specifics of their objectives and perception factors, such as depth intensity, varied while learning.

Moreover, to reduce the dependence of loss on the absolute ground truth values, the reciprocal of the depth was considered instead [12].

Adam optimizer was again used for optimization due to its low dependence on learning rate and superiority over the other optimizers in most scenarios.

4.2.3.3 Training specifics and evaluation

The training was done on a machine with the processor Ryzen 7 4800H and Nvidia GTX 1650 Ti as the GPU. The training took about 8 hours for 100 epochs. The weights were saved after each epoch for testing and experimenting purposes.

The exact parameters used are as follows:

1. Similar to the last approach, a batch size of 32 was used.
2. Learning rate was set to 0.0001 and gave great results.
3. The model was trained for 100 epochs to match the training procedure of the U-net with Densenet encoder
4. Testing was then performed, and the results are as follows in table 3.

Image 1:

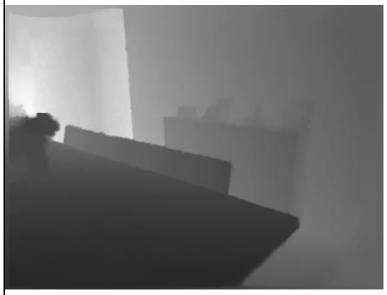
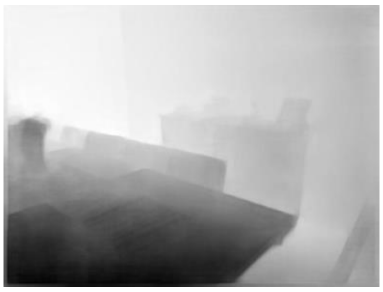
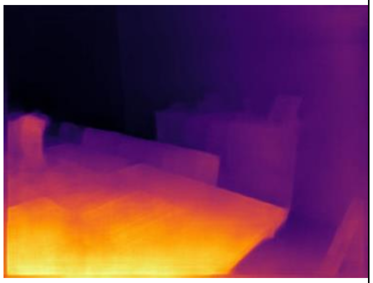
Epoch	**30**	**50**	**70**	**90**	**100: BEST**
RMSE	5907.083544921875	5676.389410807292	7497.601276041667	6342.832747395833	5453.151907552084
SSIM	0.7994239935698918	0.8072490163630696	0.7787230304149872	0.8022623566132152	0.8142396354363118
  					

Image 2:

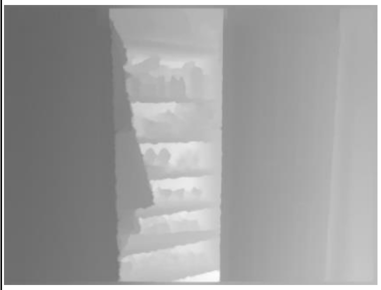
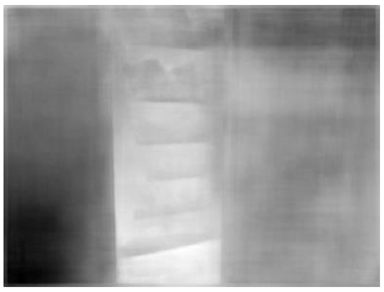
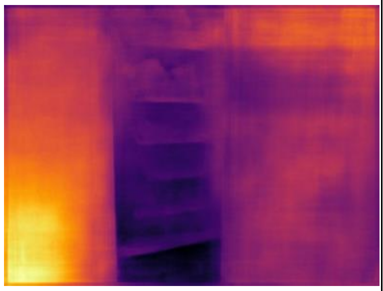
Epoch	**30**	**50**	**70**	**90**	**100**
RMSE	1338.5737662760416	1620.180390625	1371.4947786458333	1624.0465266927083	1238.3652897135416
SSIM	0.8696215676057024	0.8229568604946199	0.8389892590275926	0.8460594506751112	0.8417519161323566
  					

Image 3:

Epoch	**30**	**50**	**70**	**90**	**100**

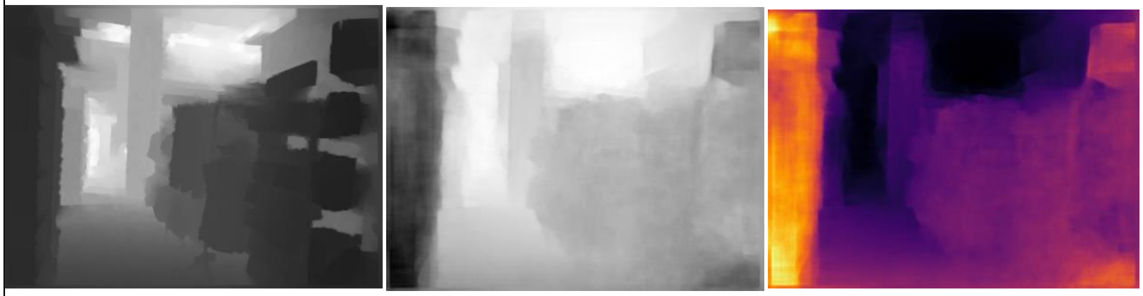
RMSE	5358.693235677083	6943.60294921875	8788.344342447917	6037.229140625	7650.837109375
SSIM	0.6503336762187644	0.6418648367410008	0.628560855790575	0.6630884347859459	0.6538859175754772
					

Image 4:

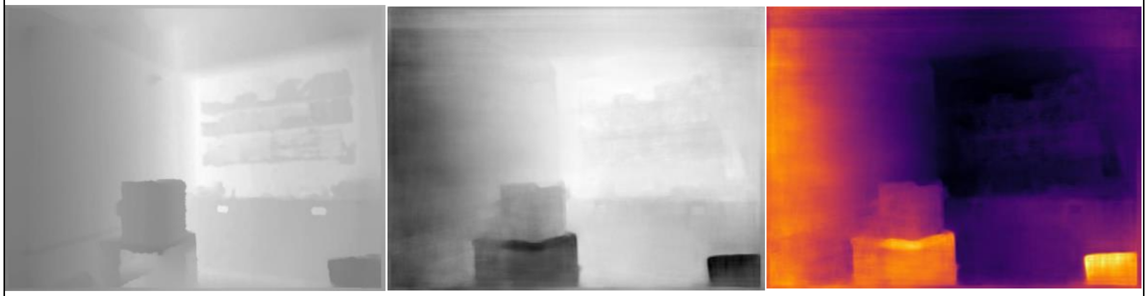
Epoch	30	50	70	90	100
RMSE	882.9916015625	827.1744661458333	891.6036328125	1775.451640625	946.408125
SSIM	0.8787261971872601	0.8830084632137888	0.881805717314765	0.861732325183184	0.8832446943704686
					

Image 5:

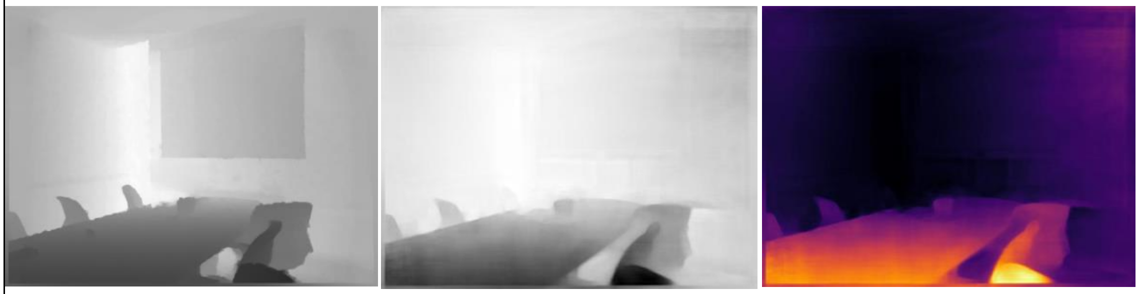
Epoch	30	50	70	90	100
RMSE	1616.5084928385418	1514.4728645833334	2174.97818359375	1955.9937174479167	1413.7335807291668
SSIM	0.8944981600887195	0.8891590437655021	0.8971249132821469	0.8993457373735034	0.9126853495442888
					

Table 3: MSE and SSIM values along with Ground truth (left), generated depth map in greyscale (centre) and generated depth map in RGB (right) for images 1-5 from Test 1 for the U-net with MobileNetv2 encoder

4.3 Stereo Depth Estimation

The Monocular depth estimation approaches described in the previous section manage to produce stunningly accurate depth maps and capture the relative depth extremely well. This eliminates the need for a Stereo depth estimation approach. However, Deep Learning solutions require high computational power and require a large volume of data, specifically depth maps that need to be generated using LIDAR or by other means. This process of data collection is usually expensive and hence is the biggest challenge to deep learning solutions.

To combat this, the stereo depth estimation approach eliminates the data collection phase altogether. This greatly reduces the development costs and encourages more low-end businesses to adopt this solution. The only trade-off is that this approach requires two images of the scene in order to produce a depth map. This process is known as Stereo depth estimation. The aforementioned trade-off is worth it when the development time is very less, and the project budget is low.

This approach uses the SGBM (Stereo Processing by Semiglobal Matching and Mutual Information) algorithm, which is a modified version of the H. Hirschmuller algorithm [42]. The algorithm first takes as input, a pair of stereo images that have been rectified and their epi-polar lines have been matched to the horizontal axis. Following this, the three main modules, namely matching cost calculation, directional cost calculation and post-processing, are executed in that order. The post- processing step generates the depth map that can be encoded and saved as required [43].

The detailed architecture diagram of the approach has been shown in the figure below in figure 15 [42].

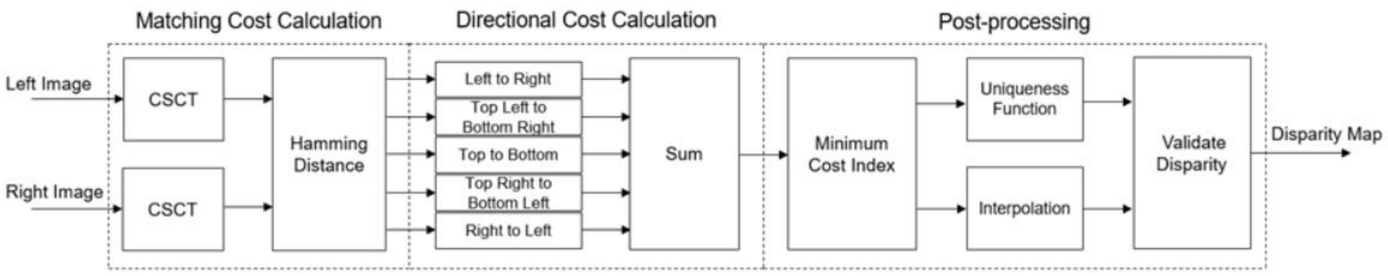
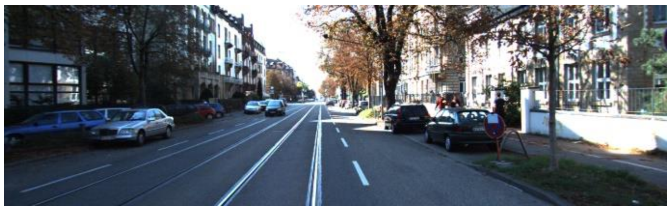


Figure 15: SGBM architecture and data flow

A few depth maps generated by the approach are shown below along with the input images.

Scene 1:



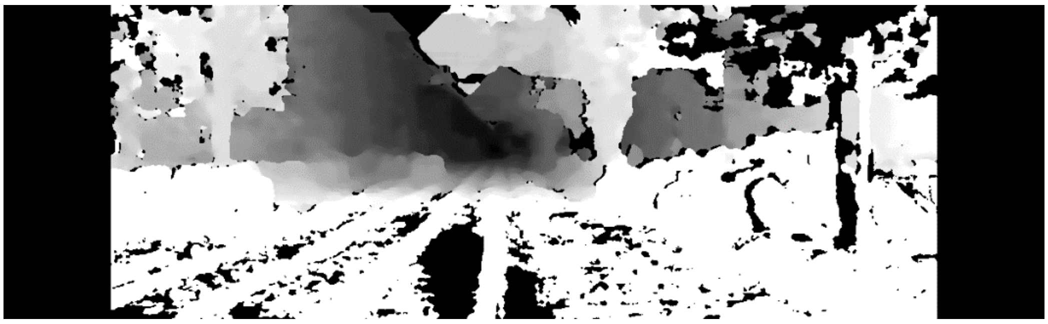


Figure 16: stereo image pairs (top) and generated depth map (bottom) for stereo pair 1

Scene 2:

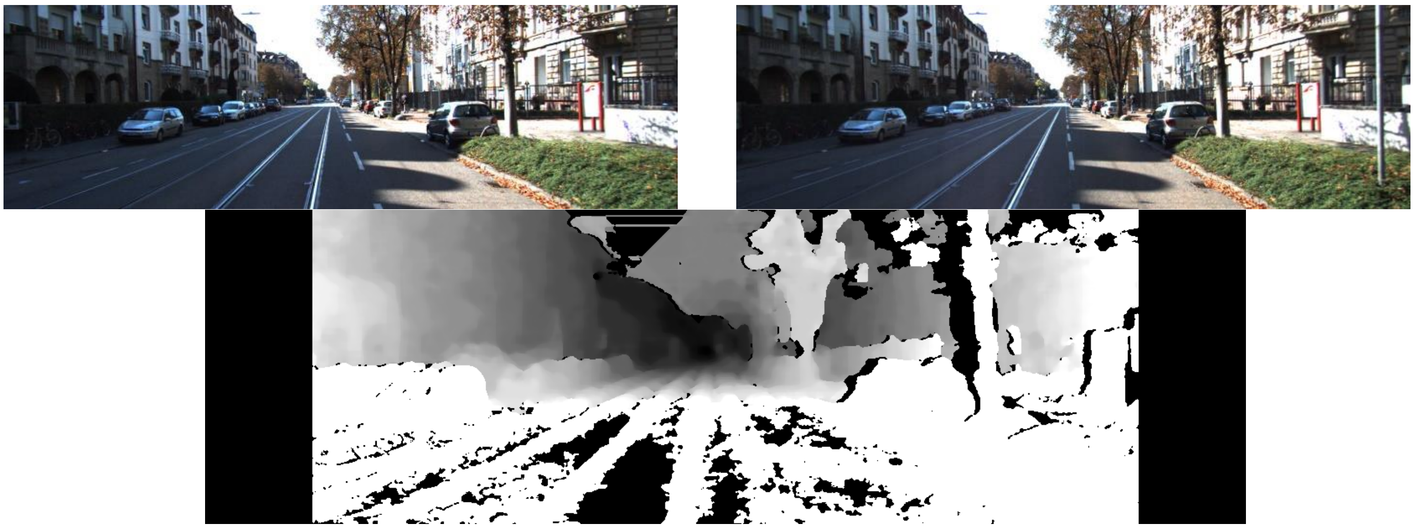


Figure 17: stereo image pairs (top) and generated depth map (bottom) for stereo pair 2

Scene 3:

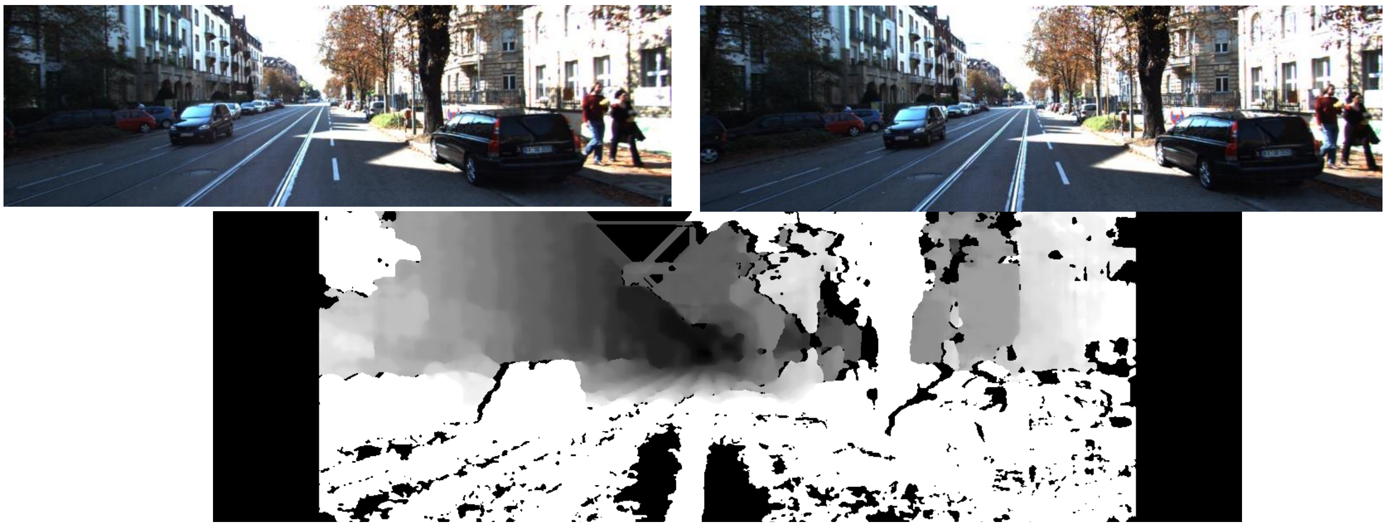


Figure 18: stereo image pairs (top) and generated depth map (bottom) for stereo pair 3

Scene 4:

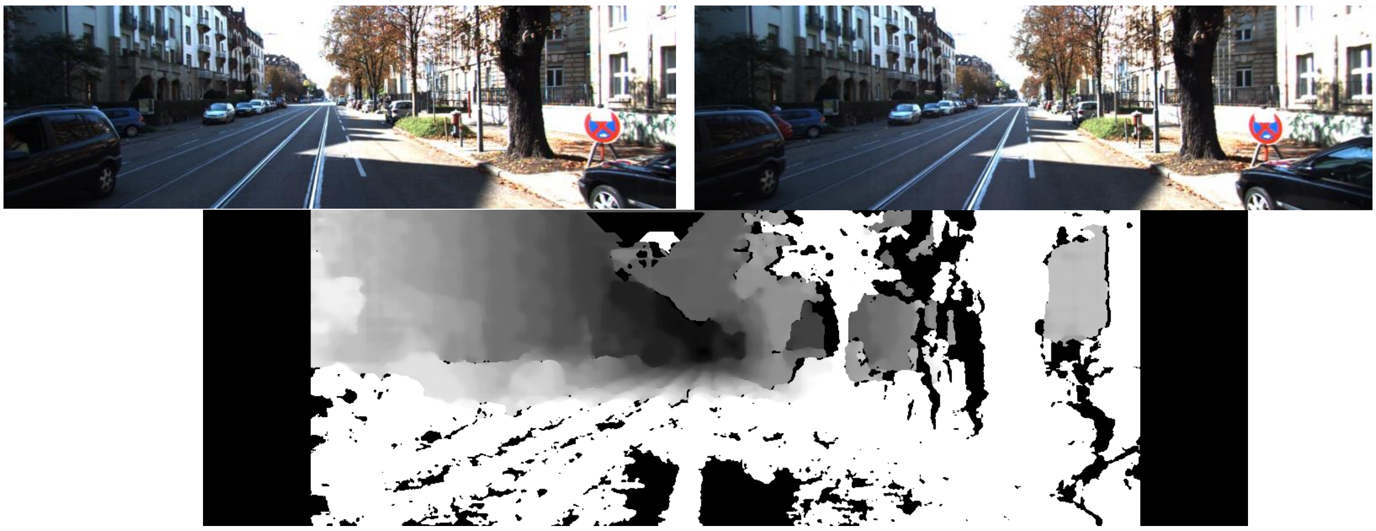


Figure 19: stereo image pairs (top) and generated depth map (bottom) for stereo pair 4

Scene 5:

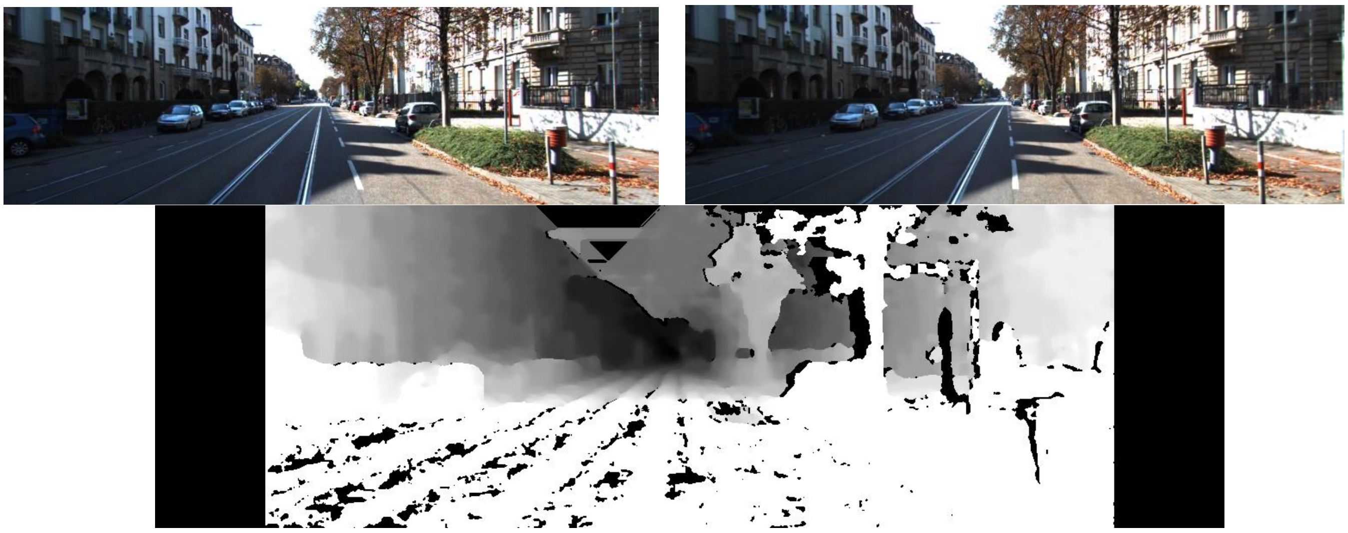


Figure 20: stereo image pairs (top) and generated depth map (bottom) for stereo pair 5

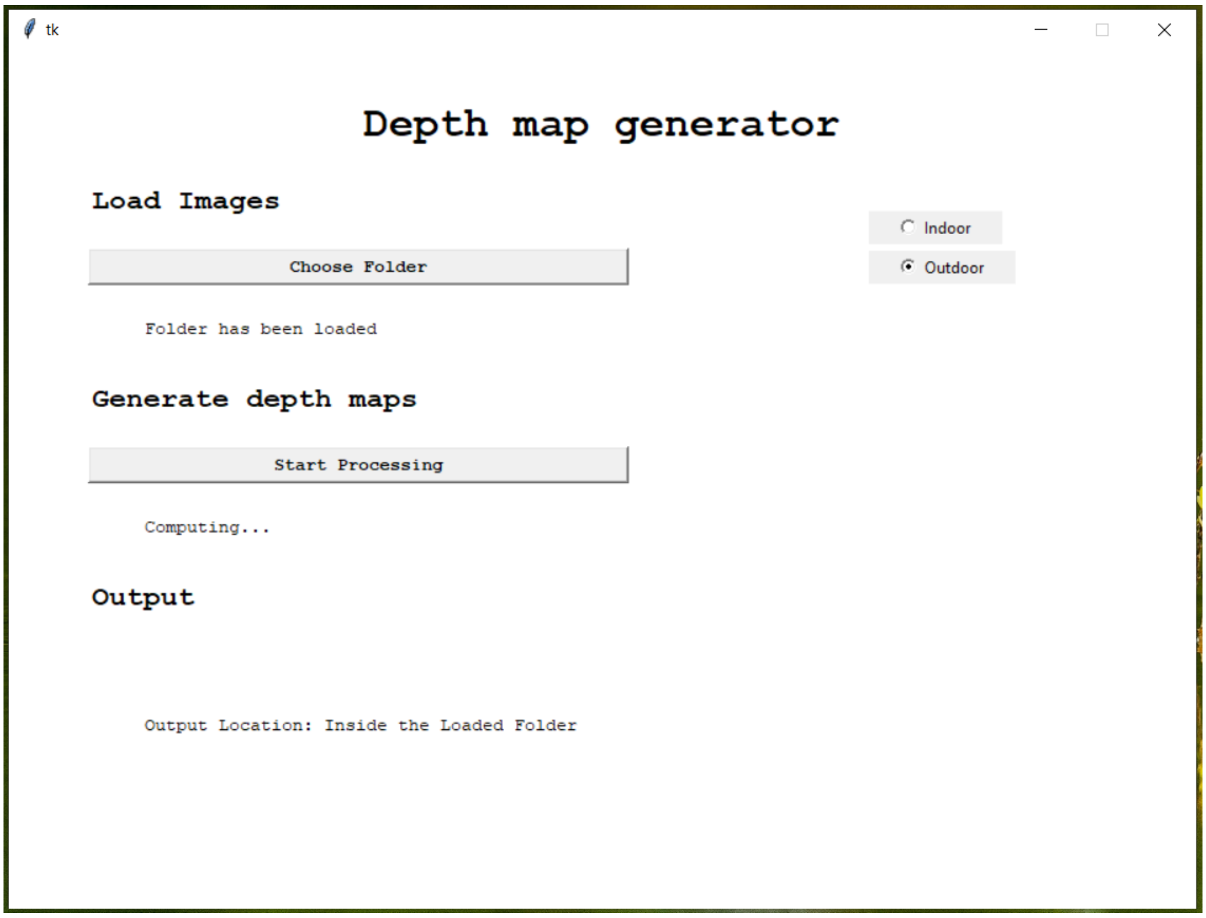
It can be observed that the depth is successfully detected to a satisfactory extent. Despite the depth not being as accurate as the one generated by the Deep Learning Models, the approach was still included as the maps were generated without the usage of any Deep Learning techniques and hence involved no training and negligible data collection. The computational burden was also drastically reduced.

4.4 The Graphical User Interface

As mentioned in the previous subsections, three separate approaches were designed for this dissertation. The pix2pix model was the best for outdoor data and the U-net with MobileNetv2 encoder gave the best results for indoor data. This shall be experimentally demonstrated in the later sections.

However, practically using these approaches involved running complex test script files and a lot of manual adjustments had to be made while running them. Also, the test files had to be run through the command prompt, making the work less user friendly.

To combat these problems, a Graphical User Interface was developed for the simple and user-friendly usage of the models for depth map generation. The GUI was made with the python programming language, specifically making use of the Tkinter library. The interface enables the user to conveniently load the data and generate depth maps based on its type. The input images are to be kept in a folder and can then be loaded onto the application with a single click. If the images involve an indoor scene, the U-net with MobileNetv2 is used to generate depth maps. On the other hand, if outdoor images are to be processed, the pix2pix model is used to generate the corresponding depth maps. The output is stored in a folder titled 'output' to ensure that the chosen folder remains pristine. These depth maps can then be conveniently accessed by the user. A snapshot of the graphical user interface is shown in the figure below.



Indoor [✓ Outdoor]

Folder has been loaded

Computing...

Output Location: Inside the Loaded Folder

Figure 21: Graphical user interface

The depth maps are stored in the output folder numerically, as follows.

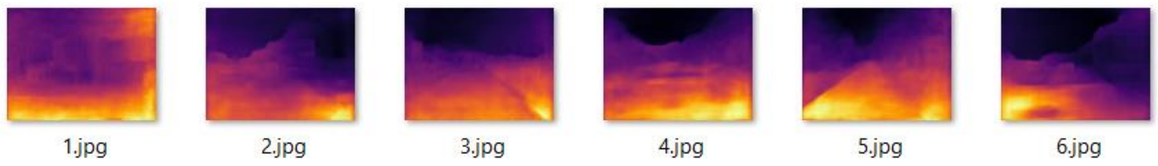


Figure 22: Outputs stored in the output folder

The code for all the models and the graphical user interface can be found at this Github repository [44]. The architecture is explained further in the Appendix at the end.

5 RESULTS

5.1 THE ENGINEERED METRIC

Using a single metric, or a single numbered score for comparing approaches is advocated as a good and appropriate practice. Having multiple metrics, like the previous section where MSE and SSIM scores were both shown, makes the process of comparing and choosing a better approach, complex and ambiguous. Ergo, a single metric is a better option.

The metric engineered for comparison is a hybrid of two other standard metrics, namely MSE and SSIM and another metric namely CC (Computational complexity) that was created just for the sake of this project. MSE or Mean Squared error is the average of the sum of squares or errors. The squared error is taken in order to eliminate the sign and use the absolute values of errors. SSIM on the other hand is a number between zero and one which represents the structural similarity between the two images that are being tested. These are both popular and widely used metrics for image evaluation. The Computational Complexity metric or CC is a fraction with its value between zero and 1, which will be assigned to an approach based on its training time per epoch. In order to ensure correct values, the training time per epoch shall be measured on the same machine (with Ryzen 4800H as the CPU and Nvidia GeForce GTX 1650 Ti as the GPU). The exact value shall be calculated as follows:

$$CC_i = \frac{t_i}{\sum_{j=1}^{n} t_j}$$

where n is the total number of approaches being considered

It is to be noted that the value will vary every time the list of approaches in consideration is altered.

The final score of the engineered metric is the weighted average sum of the metrics above. MSE is given lesser weightage than SSIM. This is because the colour scheme added to the depth maps may give rise to pixel value differences when compared to the ground truth depth maps in the original dataset. This might result in a high MSE even for accurate depth maps in certain situations. SSIM on the other hand evaluates the structure and hence is a more pertinent metric for the task at hand.

The CC value is given a very low weightage because the fact is that the results matter more than the computational burden. However, this project focuses to an extent on reducing computational power and hence this parameter was not be omitted.

The formula is shown below:

$$s_i = 1 - \left(\frac{MSE_i}{\sum_{j=1}^{n} MSE_j} * 0.4 + (1 - SSIM_i) * 0.55 + 0.05 * CC_i\right)$$

where n is the total number of approaches being considered.

5.2 Scores secured by all the three approaches on the KITTI dataset

The images used for performing the experiments recorded below were taken from the KITTI depth dataset. Despite the Pix2Pix model being trained on the KITTI dataset, the chosen images were unseen to all three models. This was possible because they were removed from the training set while training the Pix2Pix model and the other two models were trained on the NYUv2 Depth dataset.

Image no.	Model	MSE	SSIM	MSE fraction	Engineered metric
1	U-net (MobileNetv2)	7917.253173828125	0.591591063307058	0.40666	0.602711085
	U-net (DenseNet)	10890.87873840332	0.5599261954162658	0.55938	0.519207407
	Pix2Pix	660.6629372427983	0.8733671390958214	0.03396	0.891767927
2	U-net (MobileNetv2)	18060.35499572754	0.4418821940902094	0.44989	0.503079207
	U-net (DenseNet)	21981.07521057129	0.32073664107037875	0.54756	0.392381153
	Pix2Pix	102.18738747427983	0.9526240663316693	0.00254	0.947927236
3	U-net (MobileNetv2)	24257.086486816406	0.32210181854555464	0.44274	0.44006
	U-net (DenseNet)	28951.06460571289	0.2829745991925744	0.52841	0.37927203
	Pix2Pix	1580.061567644033	0.8705269444004058	0.02883	0.892257819
4	U-net (MobileNetv2)	22398.405700683594	0.3260680899649298	0.50270	0.418257449
			0.3043845311594511		0.404143492

	U-net (DenseNet) Pix2Pix	22085.63690185547 72.74429601766118	0.9598084364937529	0.49567 0.00161	0.95225064
5	U-net (MobileNetv2) U-net (DenseNet) Pix2Pix	18851.576278686523 18131.319107055664 174.27755272633746	0.3439106141922331 0.3759582059648839 0.9399489056584808	0.50734 0.48796 0.00468	0.426214838 0.446593013 0.940099898
6	U-net (MobileNetv2) U-net (DenseNet) Pix2Pix	16680.329986572266 15720.550582885742 1107.683623221022	0.5463130248525669 0.5561901393305531 0.8779111331135782	0.49780 0.46915 0.03303	0.541352164 0.553244577 0.894639123

Table 4: Scores on the KITTI dataset

The images used for the table above, along with the depth maps generated by the models and the ground truths are shown below in figure 23.

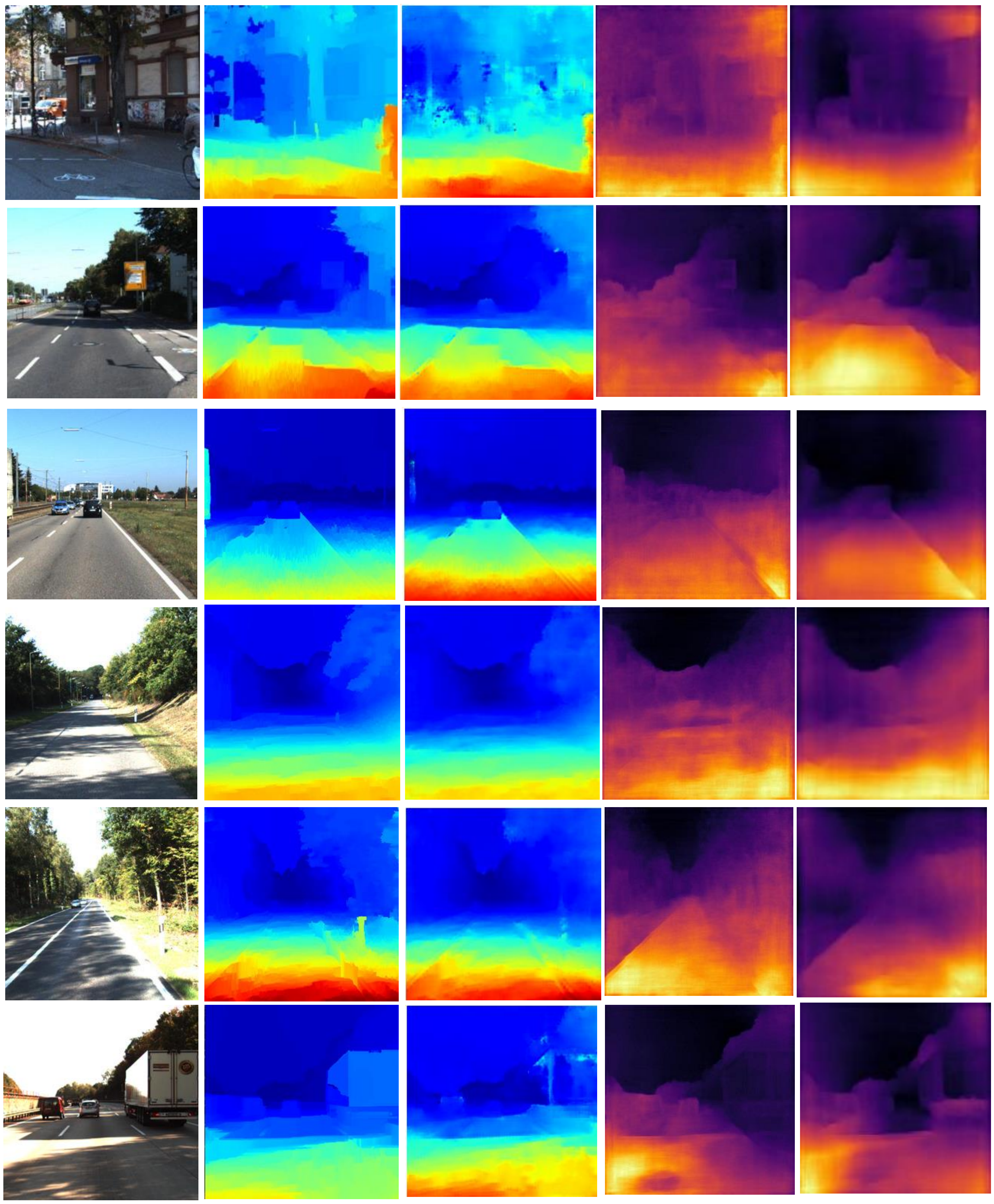


Figure 23: Each row consists of the input image, ground truth depth map, output by pix2pix model, output by U-net with MobileNetv2 and output by U-net with Densenet encoder, respectively from right to left

It is to be noted that the U-nets with MobileNetv2 and DenseNet were trained on a different dataset. Hence, the way they perceive depth and generate depth maps is different. Thus, the numerical metric values are not high for these two models. However, on manual inspection, it can be clearly seen that the Depth is detected clearly by all the approaches.

6 EVALUATION

6.1 MOBILENETV2 VS DENSENET FOR THE ENCODER

Since the U-net with MobileNetv2 and the U-net with DenseNet have similar fundamental architectures, eleven tests were performed to compare them. The results of the first set of tests have been tabulated in the table below. The results correspond to the ones generated by the best models. T1 stands for Test 1 which was performed on the NYU dataset, which the models were trained on. T2 stands for Test 2, which was performed on the KITTI dataset. This Test 2 data was completely new and strange to both models.

Image	U-net MobileNetv2: SSIM	MSE	U-net DenseNet: SSIM	MSE
T1:1	0.8142396354363118	5453.151907552084	0.7966691116124766	6033.72638671875
T1:2	0.8417519161323566	1238.3652897135416	0.833819932114343	1944.2880989583334
T1:3	0.6538859175754772	7650.837109375	0.7249024636535434	4160.359088541667
T1:4	0.8832446943704686	946.408125	0.844887741091571	2179.623955078125
T1:5	0.9126853495442888	1413.7335807291668	0.9072377006757703	1588.941787109375
T2:1	0.591591063307058	7917.253173828125	0.5599261954162658	10890.87873840332
T2:2	0.4418821940902094	18060.3549957275	0.32073664107037875	21981.07521057129
T2:3	0.32210181854555464	24257.086486816406	0.2829745991925744	28951.06460571289
T2:4	0.3260680899649298	22398.405700683594	0.3043845311594511	22085.63690185547
T2:5	0.3439106141922331	18851.576278686523	0.3759582059648839	18131.319107055664
T2:6	0.5463130248525669	16680.329986572266	0.5561901393305531	15720.550582885742
Mean	0.607061302	11351.59115	0.591607933	12151.58768

Image	U-net MobileNetv2: Engineered metric	MSE fraction	U-net DenseNet: Engineered metric	MSE fraction
T1:1	0.687940281	0.474728796	0.64805953	0.525271204
T1:2	0.7373242	0.389098384	0.634240316	0.610901616
T1:3	0.53053269	0.647761411	0.677800919	0.352238589
T1:4	0.794684337	0.302750612	0.605788502	0.697249388
T1:5	0.743647082	0.470824651	0.707310596	0.529175349
T2:1	0.586995722	0.420948407	0.49633877	0.579051593
T2:2	0.492618524	0.451041706	0.376821835	0.548958294
T2:3	0.424799831	0.455890423	0.357992199	0.544109577
T2:4	0.407931243	0.503515517	0.388817699	0.496484483
T2:5	0.415255755	0.509737707	0.430672096	0.490262293
T2:6	0.52454776	0.514811008	0.53182898	0.485188992
Mean	0.576934312		0.532333767	

Table 5: Comparison of approaches 4.2.2 and 4.2.3 on the KITTI and NYU datasets

It can be inferred that the U-net with MobileNetv2 outperforms the U-net with DenseNet in most situations. This includes both indoor and outdoor scenes. U-net with MobileNetv2 has a lower Mean Squared Error and a higher SSIM for the majority of the images. This is inferred from the mean values in the tables. Moreover, the engineered metric gives a small boost to the U-net with MobileNetv2 as it was less computationally intensive when compared to the other. CC values assigned to U-net with MobileNetv2 and U-net with DenseNet were 0.4 and 0.6 respectively, for this particular comparison. The U-net with MobileNetv2 performs almost 4.5% better than the U-net with DenseNet encoder.

Even on manual inspection, it is clearly seen that the U-net with MobileNetv2 produces better and sharper results. This difference is clear in figure 24.

The pre-trained MobileNetv2 network was trained on the ImageNet Dataset. This dataset is made for object detection and contains over a thousand classes of images with over a million images put together [36]. In order to be used with such a vast number of classes, the network must be able to encode and pass on the semantic and boundary information clearly throughout its layers and through the involved downsampling process. Ergo, it comes as no surprise that MobileNetv2 seems to have picked up outstanding semantic segmentation capabilities due to its training and hence is able to clearly encode the key contents of the image. This thorough distinguishing capability can be clearly seen in the pictures shown below in figure 24.

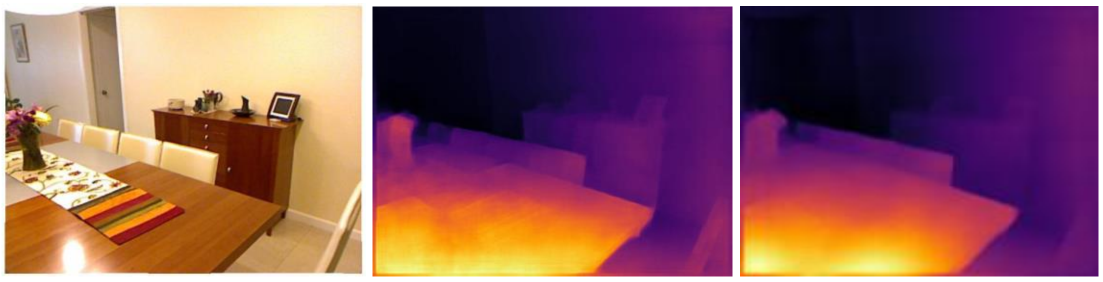


Figure 24: Example scene 1 for comparison of approaches 2 and 3 - Input image (left), Depth map generated by U-net with MobileNetv2 encoder (centre) and depth map generated by U-net with Densenet (right)

The U-net with MobileNetV2 captures the outlines of the chairs and the table perfectly. It thoroughly differentiates between the first two chairs and the table with the chest of drawers behind it and clearly portrays the depth difference between them. The depth information for even the small objects such as the vase on the table has been identified successfully with minimal error. The small region to the left of the top part of the vase has been shown to be a little closer than it actually was. However, the depth for the region beside the lower half of the vase has been detected well. The generated depth maps are also very sharp. In fact, these are sharper than the ones generated by the U-net with DenseNet encoder. The reason behind this is the addition of two more upsampling layers to the decoder.

Coming to the depth maps generated by U-Net with the DenseNet encoder, they are also quite accurate. The depth maps are softer, and the boundaries are not as distinguishable as they were with the U-net with MobileNetv2 encoder, but this simply could be the result of having fewer upsampling layers. However, in some situations, even the depth information produced by the U-net with DenseNet approach is partially wrong. An instance of this is shown below.

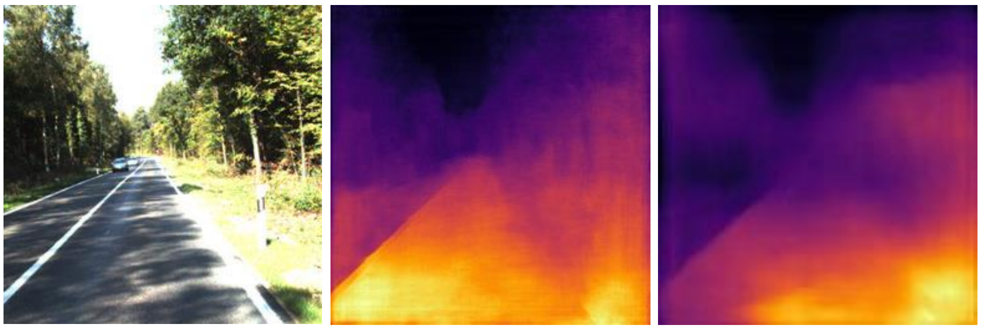


Figure 25: Example scene 2 for comparison of approaches 2 and 3 - Input image (left), Depth map generated by U-net with MobileNetv2 encoder (centre) and depth map generated by U-net with Densenet (right)

Both the approaches were trained on indoor data and had never seen roads before. Hence, both had trouble detecting depth correctly with the left side of the road due to the high contrast difference between the white lines and the rest of the road. However, the U-net with MobileNetv2 successfully detects the depth for the side of the road that the picture was taken from. The U-net with Densenet encoder on the other hand falls short here. It perceives the left half of the current lane to be much farther than it actually is and produced accurate information gradually as we move to the right side of the picture.

Despite all these shortcomings, the U-net with Densenet does manage to produce better depth maps in certain situations. This is illustrated below in figure 26.

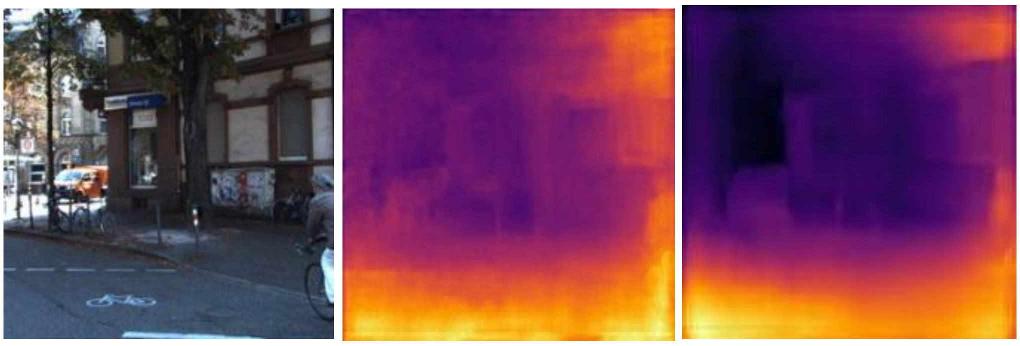


Figure 26: Example scene 3 for comparison of approaches 2 and 3 - Input image (left), Depth map generated by U-net with MobileNetv2 encoder (centre) and depth map generated by U-net with DenseNet (right)

This is an extremely complex scenario. There are too many objects and the lighting is bad. The surfaces also have high colour contrast differences such as the dark road with bright white lines and the bright walls with dark brown lines. These features confused the models. The Pix2Pix model which was trained on outdoor data also struggled with this scene.

The U-net with MobileNetv2 encoder did manage to detect the road depth quite well. It also managed to do a satisfactory job around the edges of the image. However, it went wrong near the centre of the image and showed everything to be closer than it actually is. The U-net with DenseNet encoder on the other hand also produced a depth map that was far from the ground truth. But it managed to convey that there was a significant depth difference after the edge of the brown building on the right. The U-net with MobileNetv2 failed to represent the magnitude of this depth difference correctly.

To recapitulate, both the models performed very well on indoor scenarios. The U-net with MobileNetv2 however, did manage to produce sharper and relatively more accurate depth maps in most scenarios. But, the U-net with DenseNet represented the depth more accurately in some poorly lit and complex situations. This could be the result of the skip connections in the Densenet. Finally, both approaches struggled in complex outdoor situations. However, they did surprisingly well in well lit outdoor scenes. This was remarkable because none of the models had been exposed to outdoor image data before.

6.2 Pix2Pix vs the rest

The Pix2Pix model was trained on the KITTI dataset, which consisted of all outdoor images. Hence, on being tested with outdoor images, the approach produced depth maps with stunning accuracies. In some cases, it was hard to spot any difference between the generated images and the ground truth. This is the result of how the Pix2Pix architecture is fundamentally built. The generator works to generate samples that are so close to the actual data, so as to deceive the discriminator. This pushes the generator to produce depth maps as close as possible to the ground truth. The effectiveness of this approach is seen in the sample below.

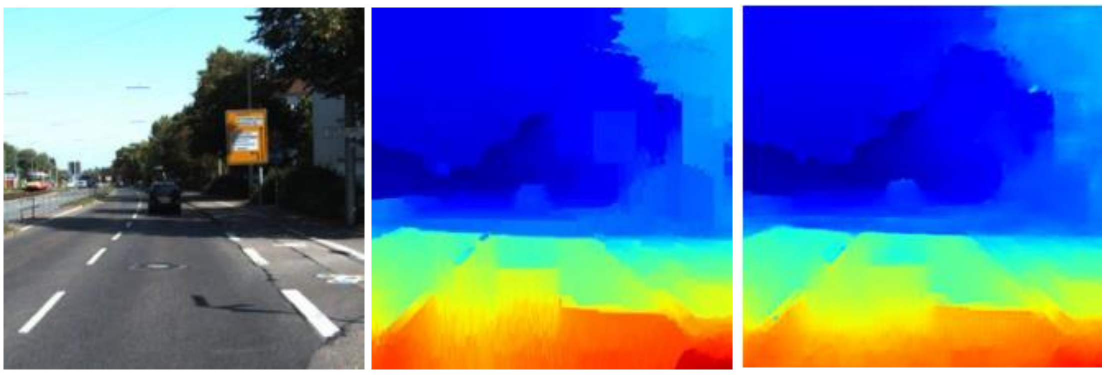


Figure 27: Example scene 1 for analysis of approach 1 - Input image (left), Ground truth depth map (centre) and depth map generated by Pix2Pix model (right)

In this particular scenario, the generated depth map is so close to the ground truth that it's hard to tell the difference between them even for a human, on manual inspection. The part near the banner specifically is a very tricky one. But the model clearly detected the depth disparity between the banner, the tree to its right and the building behind the tree. The model was trained on similar images (KITTI dataset). However, the sample above was never seen by the model before. Another example of an outstanding depth map produced by the Pix2Pix model is shown below.

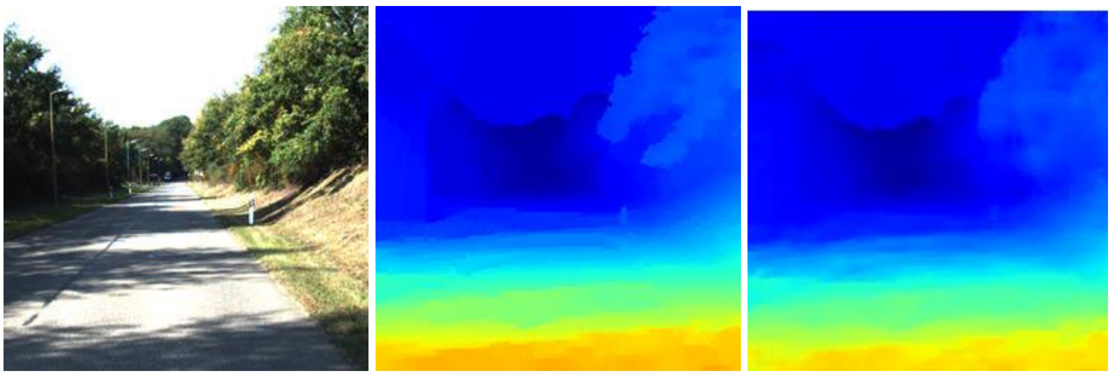


Figure 28: Example scene 2 for analysis of approach 1 - Input image (left), Ground truth depth map (centre) and depth map generated by Pix2Pix model (right)

The depth map is highly accurate with a Structural Similarity score of over 0.95 and an MSE of just over 72 as compared to over 22,000 for the U-nets. The tree line was detected clearly and the depth on the road has also been produced precisely, despite the high colour contrast differences due to the shadows.

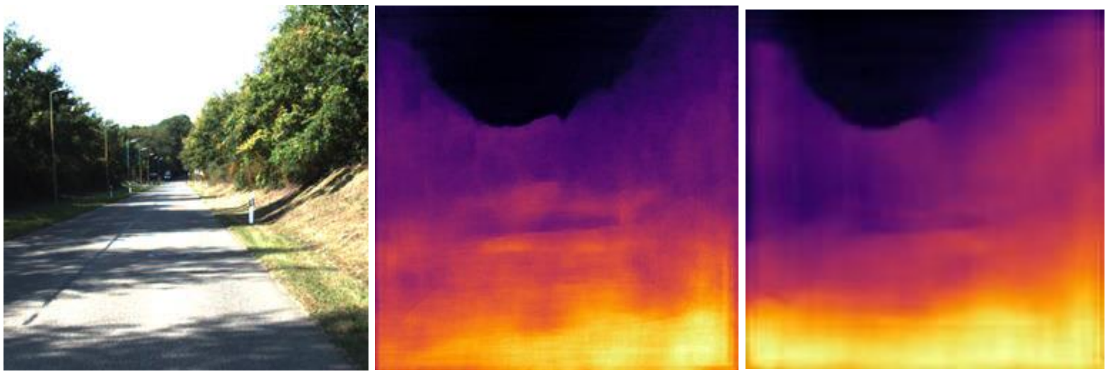


Figure 29: Example scene 2 for analysis of approach 1 and comparison with approaches 2 and 3- Input image (left), Depth map generated by U-net with MobileNetv2 encoder (centre) and depth map generated by U-net with Densenet (right)

The U-net with MobileNetv2 and Densenet encoders also managed to show the depth surprisingly well despite never having seen similar or any outdoor data prior to the test. They managed to tackle the high colour contrast differences well. However, the central part of the depth maps was not recreated accurately. The SSIM values were low and MSE values were high for both the U-nets. However, these metrics cannot be solely considered to judge these approaches on the KITTI dataset. This is because the exact perception of relative depth difference and the intensity of the depth maps vary drastically, not to mention the differences in conditions. Hence, manual inspection is a much better option when judging the U-nets on an unfamiliar scenario. Through manual inspection, it was inferred that their perception of depth, albeit a little different to the ground truth in the KITTI dataset, is still highly accurate and plausible.

When compared with the engineered metric in the outdoor scene conditions, the U-net with DenseNet encoder performs poorly. The U-net with MobileNetv2 encoder and the Pix2Pix models perform relatively better. The effect of including CC (Computational Complexity) in the engineered metric is demonstrated in this comparison. For Image 1, the Pix2pix model and the U-net with MobileNetv2 secure the same score of 0.41, despite the

Pix2Pix model have much higher SSIM and lower MSE scores. This is because the Pix2Pix model is highly computationally intensive. It took over twice as long as the U-net with MobileNetv2 to train. Factoring this in, the U-net with MobileNetv2 is also a quick and stellar solution for outdoor depth estimation, and triumphs over the U-net with DenseNet.

Since the Pix2Pix model gave the best results for outdoor data, it was tested on indoor images as well. Since the U-net with MobileNetv2 gave better results than the U-net with DenseNet for indoor images, as proved in the previous section, on the NYU dataset, it was compared with the Pix2pix model. The test results are shown in the table below.

Image	**U-net MobileNetv2: SSIM**	**MSE**	**Pix2Pix: SSIM**	**MSE**
T1:1	0.8142396354363118	5453.151907552084	0.8055766906820525	2741.8794661458332
T1:2	0.8417519161323566	1238.3652897135416	0.6295929908154684	6839.490035807292
T1:3	0.6538859175754772	7650.837109375	0.7987741427435131	2793.28103515625
T1:4	0.8832446943704686	946.408125	0.7385004585688127	5107.9494010416665
T1:5	0.9126853495442888	1413.7335807291668	0.6648300312541919	9657.813037109376
Mean	0.821161503	3340.499202	0.727454863	5428.082595

Image	**U-net MobileNetv2: Engineered metric**	**MSE fraction**	**Pix2Pix: Engineered metric**	**MSE fraction**
1	0.61291311	0.665421724	0.72798587	0.334578276
2	0.832892065	0.153303722	0.426347634	0.846696278
3	0.497867317	0.732549843	0.751095716	0.267450157
4	0.854507179	0.156318506	0.487452655	0.843681494
5	0.882150661	0.127690704	0.435482799	0.872309296
Mean	0.736066066		0.565672935	

Table 6: Comparison of the Pix2pix model and the U-net with MobileNetv2 encoder

The U-net with MobileNetv2 has a slight edge over the Pix2Pix model when tested on the NYU dataset consisting of Indoor images. However, the former was trained on the same type of data. Moreover, the Pix2pix model secured scores that were very close to the scores obtained by the U-net with the MobileNetv2 encoder. The difference in average SSIM scores was just over 0.1. There was a significant difference in terms of the engineered metric because the CC (computational complexity) parameter was considered. This parameter gave a small advantage to the U-net with Mobilenetv2 encoder as it was much less computationally intensive compared to the Pix2pix model.

In the tables above and in the table from the results section, the Pix2pix model and the U-net with Mobilenetv2 encoder were compared on both, the KITTI (outdoor images) dataset and the NYU (indoor images) dataset. It was established that both the models performed well on the datasets that they were trained on respectively. However, they also produced good results on new unseen data of the other kind. This shows that both models have great generalization capabilities. They were both able to pick up depth differences even in complex situations. The pix2pix model gave the best results overall. It came very close to U-net with MobileNet v2 encoder in indoor data and dominated with a huge margin on outdoor data. The latter however could not 'dominate' the former even on indoor data. It just produced slightly better results on the NYU dataset and lagged behind by a huge margin on the KITTI dataset, even after factoring in the computational complexity.

Hence, it was established that the Pix2Pix architecture is suitable for projects with a high budget, demanding state of the art accuracy and results. The U-net with MobileNetv2 encoder is perfect for low budget and single-use projects that demand quick and cheap development and deployment. If computational complexity is the biggest concern and gathering stereo pairs of images are not a problem, the SGBM would be an ideal choice.

7 CONCLUSION AND FUTURE WORK

This thesis starts off by reviewing the current depth estimation approaches. Following this, three end to end approaches, namely Pix2pix model, U-net with Densenet encoder and U-net with MobileNetv2 encoder, were developed and compared. A metric was engineered specifically for this thesis and used for the comparison of these approaches. The metric took the computational complexity of models as a key consideration. It was experimentally inferred that the Pix2Pix model performed the best as it was specifically made for image-to-image translation. However, it was relatively more computationally intensive. Amongst the other two approaches, the U-net with MobileNetv2 Encoder was experimented on more, and tweaks such as more upsampling layers were added. Ergo, It outperformed the U-net with Densenet and produced great results. Hence, the U-net with MobileNetv2 was deemed perfect for low budget projects and the Pix2pix model was deemed best for projects with more time and money. Stereo depth estimation was also explored. The SGBM algorithm was used for the same. The advantage of this approach was that it did not need any training data. Hence, the approach could be deployed without needing any development time.

Finally, the advantage of training different networks on different datasets was that they could be used to engineer a hybrid approach that could be used for any kind of data. This was facilitated by the Depth estimator GUI. It used the appropriate model for a corresponding type of data to generate accurate depth maps.

In future work, the models shall be included in a smartphone app and integrated with the camera. Moreover, a new dataset shall be generated manually consisting of a good number of indoor and outdoor images. Ideally, the data from NYU and KITTI datasets could simply be clubbed. However, since their perception of depth is different due to being captured by different technologies, the training process would be negatively affected. Hence data must be collected whilst following the convention adopted by one of the datasets. Since KITTI (outdoor) dataset is larger, indoor images can be captured using the same technology and added to the former to make a large dataset that will help the models to generalize better. This would involve the manual collection of data using LIDAR equipment. It is an expensive process but doing so can help researchers and companies worldwide in making progress in the domain of depth estimation using Artificial Intelligence and potentially even replace LIDAR and other expensive solutions altogether.

8 REFERENCES

1. Depth Estimation: Basics and Intuition, Daryl Tan, https://towardsdatascience.com/depth-estimation-1-basics-and-intuition-86f2c9538cd1
2. NYU depth v2 Dataset: https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html
3. Research Guide for Depth Estimation with Deep Learning, https://heartbeat.fritz.ai/research-guide-for-depth-estimation-with-deep-learning-1a02a439b834
4. LIDAR, https://lidarradar.com/info/pros-and-cons-of-lidar
5. P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image to image translation with conditional adversarial networks. In CVPR, 2017.
6. KITTI depth dataset: http://www.cvlibs.net/datasets/kitti/eval_depth.php?benchmark=depth_prediction
7. NYUv2 depth dataset, https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html#:~:text=The NYU-Depth V2 data,scenes taken from 3 cities
8. Activation Functions in Neural Networks, https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d
9. Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
10. Alhashim, Ibraheem, and Peter Wonka. "High quality monocular depth estimation via transfer learning." arXiv preprint arXiv:1812.11941 (2018).
11. C. Godard, O. M. Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6602–6611, 2017
12. B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5622–5631, 2017
13. 7 tips to choose the best optimizer, https://towardsdatascience.com/7-tips-to-choose-the-best-optimizer-47bb9c1219e
14. Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
15. SGBM Algorithm, https://uk.mathworks.com/help/visionhdl/ug/stereoscopic-disparity.html
16. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.
17. UNet, https://towardsdatascience.com/u-net-b229b32b4a71
18. Pan, Sinno Jialin, and Qiang Yang. "A survey on transfer learning." IEEE Transactions on knowledge and data engineering 22.10 (2009): 1345-1359.
19. Qureshi, Aqsa Saeed, et al. "Wind power prediction using deep neural network based meta regression and transfer learning." Applied Soft Computing 58 (2017): 742-755.
20. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.
21. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2020. Generative adversarial networks. Communications of the ACM, 63(11), pp.139-144.
22. A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta and A. A. Bharath, "Generative Adversarial Networks: An Overview," in IEEE Signal Processing Magazine, vol. 35, no. 1, pp. 53-65, Jan. 2018, doi: 10.1109/MSP.2017.2765202.

23. Bengio, Y., Goodfellow, I. and Courville, A., 2017. Deep learning (Vol. 1). Massachusetts, USA:: MIT press.
24. Wang, Z., Bovik, A.C., Sheikh, H.R. and Simoncelli, E.P., 2004. Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4), pp.600-612.
25. Mean Squared Error, https://en.wikipedia.org/wiki/Mean_squared_error
26. Vedaldi, A. and Lenc, K., 2015, October. Matconvnet: Convolutional neural networks for matlab. In Proceedings of the 23rd ACM international conference on Multimedia (pp. 689-692).
27. Convolutional neural networks, https://en.wikipedia.org/wiki/Convolutional_neural_network
28. J. Li, R. Klein, A. Yao, A two-streamed network for estimating fine-scaled depth maps from single rgb images, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 3372–3380.
29. J. Lee, C.S. Kim, Monocular depth estimation using relative depth maps, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2019, pp. 9729–9738
30. Ming, Y., Meng, X., Fan, C. and Yu, H., 2021. Deep Learning for Monocular Depth Estimation: A Review. Neurocomputing.
31. D. Xu, W. Wang, H. Tang, H. Liu, N. Sebe, E. Ricci, Structured attention guided convolutional neural fields for monocular depth estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3917–3925
32. A.C. Kumar, S.M. Bhandarkar, M. Prasad, Depthnet: a recurrent neural network architecture for monocular depth prediction, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2018, pp. 283–291
33. Li, Y., Qian, K., Huang, T. and Zhou, J., 2018. Depth estimation from monocular image and coarse depth points based on conditional gan. In MATEC Web of Conferences (Vol. 175, p. 03055). EDP Sciences.
34. Chen, R., Mahmood, F., Yuille, A. and Durr, N.J., 2018. Rethinking monocular depth estimation with adversarial training. arXiv preprint arXiv:1808.07528.
35. Zheng, C., Cham, T.J. and Cai, J., 2018. T2net: Synthetic-to-realistic translation for solving single-image depth estimation tasks. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 767-783).
36. ImageNet dataset, https://www.image-net.org/
37. Ranftl, R., Bochkovskiy, A. and Koltun, V., 2021. Vision transformers for dense prediction. arXiv preprint arXiv:2103.13413.
38. Bhat, S.F., Alhashim, I. and Wonka, P., 2021. Adabins: Depth estimation using adaptive bins. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 4009-4018).
39. Wu, Y., Boominathan, V., Chen, H., Sankaranarayanan, A. and Veeraraghavan, A., 2019, May. Phasecam3d—learning phase masks for passive single view depth estimation. In 2019 IEEE International Conference on Computational Photography (ICCP) (pp. 1-12). IEEE.
40. L. Zhou and M. Kaess, "Windowed Bundle Adjustment Framework for Unsupervised Learning of Monocular Depth Estimation With U-Net Extension and Clip Loss," in IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 3283-3290, April 2020, doi: 10.1109/LRA.2020.2976301.
41. M. Klingner and T. Fingscheidt, "Online Performance Prediction of Perception DNNs by Multi-Task Learning With Depth Estimation," in IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 7, pp. 4670-4683, July 2021, doi: 10.1109/TITS.2021.3054437.

42. StereoBinarySGBM, https://docs.opencv.org/4.5.2/d1/d9f/classcv_1_1stereo_1_1StereoBinarySGBM.html
43. Hirschmuller, H., 2007. Stereo processing by semiglobal matching and mutual information. IEEE Transactions on pattern analysis and machine intelligence, 30(2), pp.328-341.
44. Depth Estimation for low budget projects, https://github.com/skth5199/depth_gen

APPENDIX

The working of the Depth estimator GUI is shown in the figure below. This GUI enables the models developed during this project, to be used by other researchers and companies with ease, without having to download and read through large complex blocks of code. The application also chooses the best model based on the type of data (be it indoor or outdoor), and generates the best depth map it can with the available models.

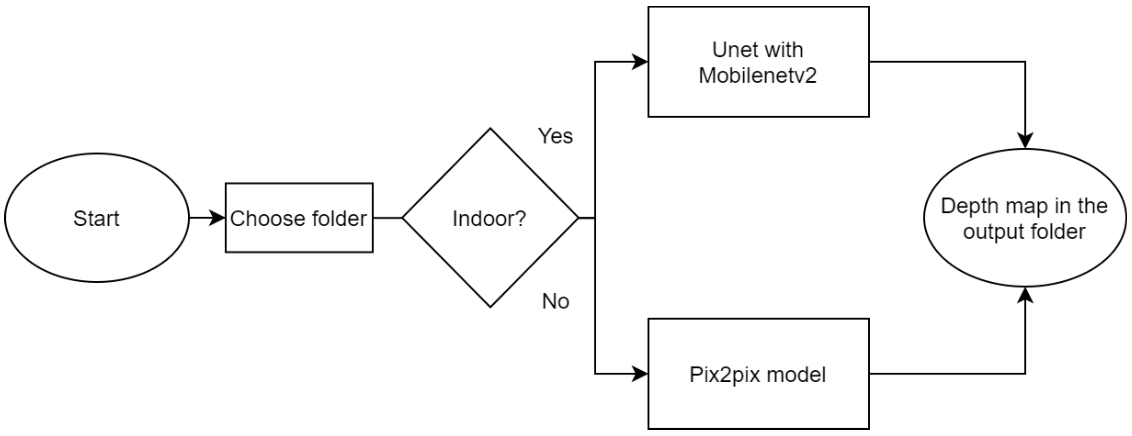


Figure 30: Depth estimator GUI dataflow

The GUI was developed using python and Tkinter library and first greets the user with a welcome screen as shown below.

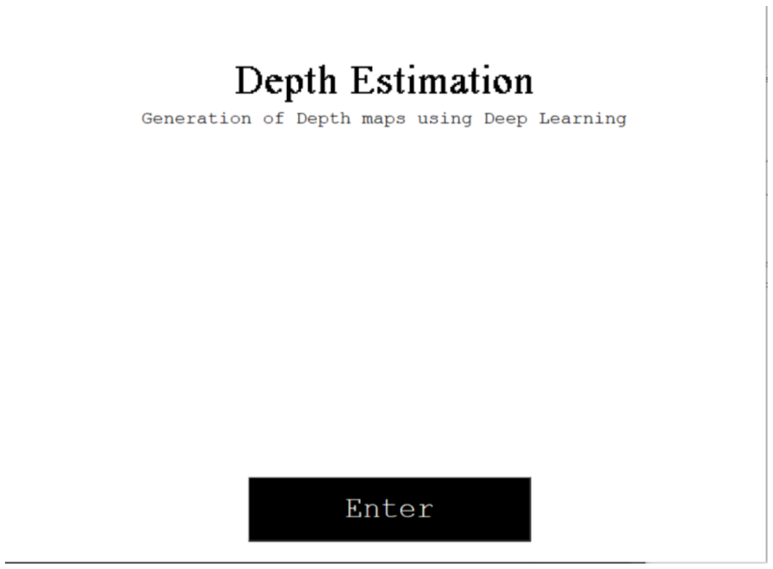


Figure 31: Depth estimator GUI welcome screen