Transfer Learning for Depth networks In Outdoor Environments

Presented by:

Kevin Pogrund

Prepared for:

Paul Amayo

February 5, 2023

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for the degree of Mechatronics.

Abstract

Estimating depth from a single image is becoming increasingly well researched in a structured environment with current state of the art, achieving impressive results. There is very little research in monocular depth estimation in unstructured environments. This project proposes the use of transfer learning to adapt the current state of the art networks in structured, outdoor environments to unstructured outdoor environments using transductive transfer learning, more commonly known as domain adaption. This was done by freezing various layers of the depth networks and then only training some of the later, more specific layers on the unstructured dataset. This allowed the depth networks to be trained using less data and computational resources. This project shows that this method is effective at adapting the depth networks however it would likely be more effective to train the networks on a large, unstructured dataset should the computational resources be available.

Acknowledgements

Firstly I would like to thank Dr Paul Amayo, for reassuring me when I was hesitant to switch to this topic and encouraging me when I thought I may have made a mistake. Thank you for guiding me during the troubles involving GPU and then with the copious amount of debugging that was required. Thanks to you I have my foot in the door of the field of machine learning.

Thank you Sasha, for helping me with debugging as well as reassuring me that what I was going through is not uncommon in machine learning research.

Thank you Eden, Levi and Seth, for your understanding and encouragement despite having minimal social interaction for the last 13 weeks. I thoroughly look forward to seeing you all again soon.

Thank you to my lecturers and tutors throughout my years at UCT. It goes without saying that I would not be here if not for you. You are the giants upon whose shoulder I will hopefully stand.

Finally, I would like to thank my family. Mom and Dad, thank you for always supporting me even when I may not deserve it. Your guidance and reassurance has been a constant throughout my life and for that I am grateful.

Plagiarism Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.

2. I have the IEEE convention for citation and referencing. Each contribution to, and quotation in, this final year project report from the work(s) of other people, has been attributed and has been cited and referenced.

3. This final year project report is my own work.

4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as there own work or part thereof.

.....

Kevin Pogrund

February 5, 2023

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Depth estimation is a computer vision task of taking a two-dimensional image and deriving three-dimensional depth information from it. Autonomous vehicles use depth estimation for tasks such as scene reconstruction, obstacle detection, obstacle avoidance [1] and path planning [2]. Common approaches in the past use the differences between two images to predict depth, imitating how eyes work. In recent times, there has been advancements in the methods used for depth estimation due to the improvements in deep learning. Thanks to neural networks, depth estimation from a single RGB image, called monocular depth estimation, has become relatively effective.

1.2 Problem statement

Most of the current research on depth estimation is done in structured environments, either indoors or in urban areas. This study will focus on depth estimation

in outdoor, unstructured environments. One of the main differences between structured and unstructured environments is that unstructured environments have little to no straight lines or other geometric structures. This lack of structure makes depth estimation in unstructured environments more challenging.

Due to the large amounts of ground truth data required as well the amounts of computer resources required to train a depth estimation network, depth estimation in unstructured environments is not researched particularly often. Because of this, it would be much more economical in terms of computer resources as well as data required to adapt a depth network trained in a structured environment to unstructured environments. This will be attempted with the machine learning technique called transfer learning

1.3 Objectives

The main objective of this project is to adapt existing depth networks, trained in structured environments, to unstructured environments. In order to do this, they must first be evaluated in structured environments to provide a benchmark. This will be done using transfer learning. As such, the efficacy of transfer learning for adapting depth networks will also be assessed.

1.3.1 Sub-Objectives

- Evaluate the current networks on unstructured data
- Train these networks on the unstructured data using transfer learning
- Evaluate the trained networks on unstructured data and compare with the original networks

Evaluate the current networks on unstructured data

In order to have a benchmark, the current depth networks, which have been trained on structured data, will be evaluated.

Train these networks on the unstructured data using transfer learning

The networks will then be trained on the unstructured dataset using transfer learning. This will be done on the same unstructured dataset as was used for evaluation.

Evaluate the trained networks on unstructured data and compare with the original networks

The trained networks will then be evaluated, on the same images as the untrained depth networks. These results will then be compared in order to determine whether transfer learning successful in this case.

1.4 Scope and Limitations

The scope of this project is mainly the analysis and adaption of three state of the art monocular depth networks in unstructured environments. The performance of the networks will be done qualitatively and quantitatively. Due to a limitation in both time and computational power, as well as not all the networks being adapted in the same environment, only the results in terms of accuracy of depth estimation will be examined. Speed and computational resource requirements are beyond the scope of this project.

In order to ensure a fairer comparison, all three depth networks chosen are based on an encoder-decoder architecture. They are also all supervised networks as such,

other training methods like unsupervised and semi-supervised are beyond the scope of this project.

Finally, the quality of training is largely dependant on the volume of ground truth data. this is a limitation of this project seeing as there are a total of 500 images being used for training and evaluation.

Chapter 2

Literature Review

2.1 Introduction

The development of autonomous vehicles (AV), specifically cars, has driven innovation across an array of fields including machine learning and computer vision. One of the key developments has been depth estimation. Depth estimation traditionally has been done by comparing the images of two cameras and using the disparity to estimate depth in an example of biomimicry.

The majority of these depth networks are trained on data that has been captured in a structured environment, either urban or indoors. This is due to the availability of large, publicly available datasets in these settings. Structured environments are environments made up of man made structured. They are generally made up a lot of straight lines which are not found in nature.

This review will analyse the previous methodologies used for depth perception as well as the current state of the art in terms of monocular depth perception. It will also go over transfer learning techniques and how they can be used to adapt depth networks to an unstructured environment.

2.2 Depth Estimation

Accurate depth estimation is an important task in the field of autonomous vehicles. An RGB image is transformed into a pixel-map containing a value for each pixel [3]. Often, these pixels are given a colour and the display is a colour map which visually represents the depth of each pixel shown in Figure 2.1. This can assist the vehicles with task such as obstacle detection and avoidance [1] and path planning [2].

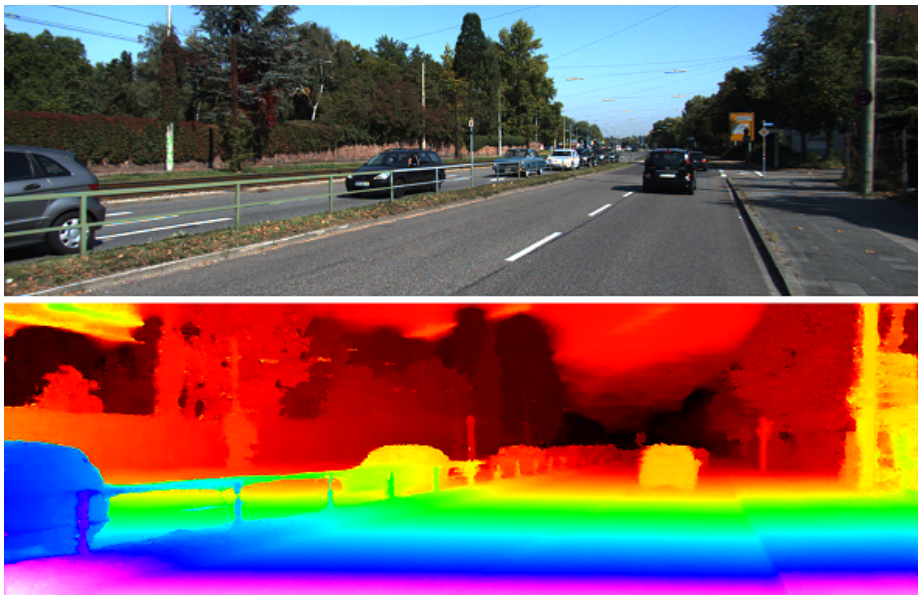


Figure 2.1: Example of a Depth Estimation Heatmap [4]

Object tracking, image segmentation, 3D scene reconstruction, simultaneous localization, and mapping (SLAM) all require accurate depth estimation [5]. These tasks are vital for perception and navigation of an unfamiliar environment for autonomous vehicles.

2.3 Previous Methodologies

Depth estimation has been done using binocular vision [6] his is often done by using a stereo camera, and therefore called stereopsis, and depth estimation algorithms which perform geometrical analysis using discrepancies in the two images. This is an example of bio-mimicry, copying how humans perceive depth using their visual cortex [7] and is called stereo matching [8]. The underlying principles for stereo depth estimation is shown in Figure 2.2 below. Structure from motion (SFM) had also been used for depth perception using a set of images captured from a camera moving within an environment [9]. In both techniques, the disparity between the images is used to estimate the depth. Stereo is based on spatial disparity between the points and SFM is based on the temporal disparity of these points [10].

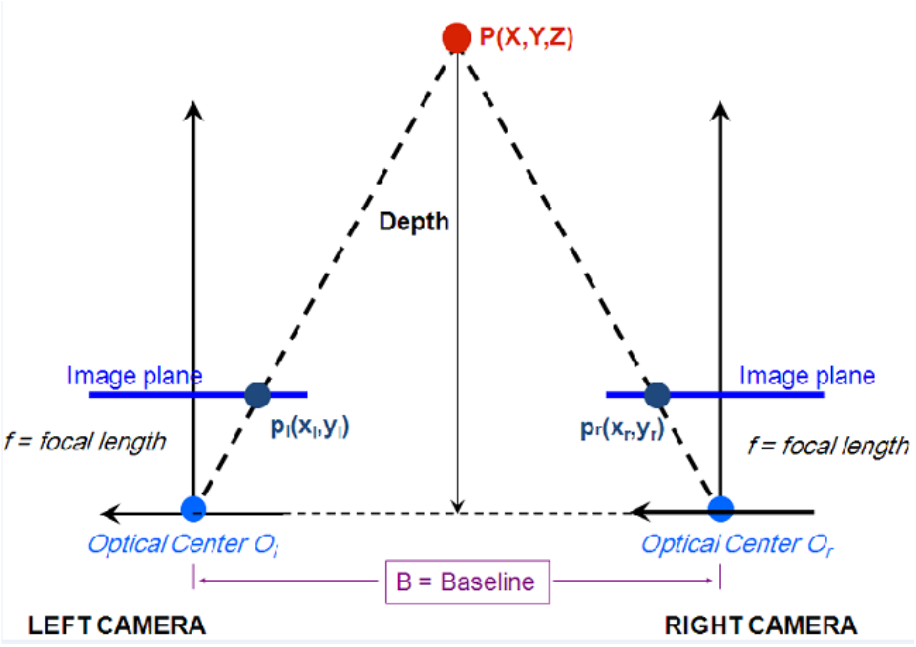


Figure 2.2: Visual representation of how Stereo Matching is done [11]

Stereo matching has been developed to be highly accurate in controlled environments. However, it is difficult to achieve this accuracy in real-world situations. In the real world, textureless areas, reflective surfaces, and repetitive patterns, are difficult for stereo algorithms to accurately map [12]. Because stereo matching is done using two or more images of the same scene captured from different angles [13], it is necessary to have accurate information about the location of each camera relative to the others. By looking at Figure2.2 a change in the distance between cameras will result in an incorrect estimation of depth. There is also the issue of depth discontinuities and occlusion [14], shown in 2.3.

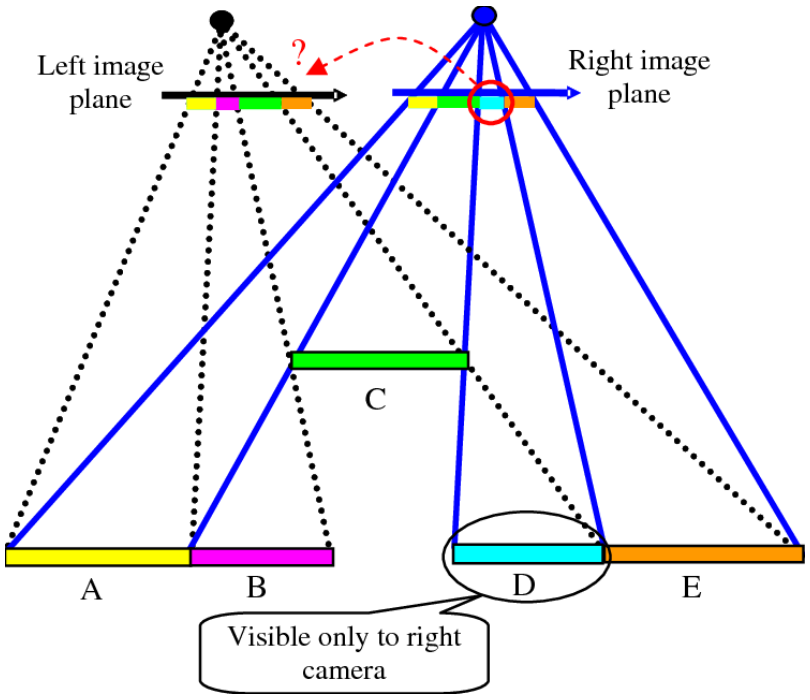


Figure 2.3: Visual representation of Occlusion

Since there are situations where multiple cameras are not available, as well as the potential cost benefit of using fewer cameras, there has been an increase in the research into monocular depth estimation. The purpose of monocular depth

estimation is to estimate the depth of each pixel in a single RGB image. Because it is using a single image, monocular depth estimation relies on pictorial clues within the image. Texture variations and gradients, defocus colour/haze contain useful and important depth estimation [15].

2.4 Current Stage of Development

2.4.1 Stereo

Stereo matching is comparing two images and matching their features, thereby converting it from two 2D images into a 3D depth map. The semi-global matching (SGM) algorithm reported by Hirschmüller [16] is one of the top-performing algorithms in the Middlebury benchmark [17]. Deep learning has been applied to stereo depth estimation to increase the accuracy of the predictions. Žbontar and Le-Cun were the first to propose training a convolutional neural network on pairs of small image patches were the true disparity is known (from LIDAR or structured light) [18]. Instead of a matching cost calculation, they used a Siamese network called MC-CNN to predict the similarity between image patches. The use of deep learning was increased when Mayer et al developed an end-to-end disparity network, called DispNet [19]. Figure 2.4 below highlights the improvements that deep learning made on stereo matching.

Figure 2.4: Comparison of Stereo Matching methods [19]

2.4.2 Monocular Depth Estimation

Due to the limitations of stereo matching, monocular depth estimation has garnered a lot of interest. Research is being done to find out the most cost effective, efficient, and accurate methods of estimating depth from a single image. This is done with two kinds of machine learning, supervised and unsupervised.

Unsupervised Learning

Unsupervised learning does not require large amounts of labelled data. Supervised learning is generally done using view synthesizers as the supervisory signal and a photometric warp loss to replace the loss calculated from ground truth depth maps [20]. View synthesis is the creation of a new view of the same subject within an image. View synthesis can be done in conjunction with stereo matching data by using pairs of neighbouring camera views to create and render arbitrary virtual views on a specified camera path between them [21].

Supervised Learning

Supervised learning requires labelled data. Depth estimation is therefore a problem of minimising the difference between the predicted and labelled depth for each pixel. This requires that the data is accurately labelled to the correct depth for each pixel. This heavily relies on a large amount of labelled data so methods of reducing the volume of labelled data required have been developed. This is called semi-supervised learning.

ADABINS

Bhat et al. [22] have recently proposed a new architecture building block, called AdaBins, for depth estimation. AdaBins comes from adaptive bins. The bin width

is adaptively computed for each image as opposed to trained bins which vary for each dataset. They evolved the approach used by Fu et al. [23] who observed that performance was improved if the depth regression task was transformed into a classification task. They used a fixed number of bins of fixed width, whereas Bhat et al. computed adaptive bins that dynamically change depending on the features of the input scene.

Furthermore, Bhat et al. realised that by changing the order of the blocks within the architecture, they were able to achieve better results. The state-of-the-art architecture is encoder-attention-decoder and Bhat et al. used encoder-decoder-attention.

When calculating pixel depth, Bhat et al. did not predict the depth as the bin centre of the most likely bin. This resulted in smoother depth maps without discretization artifacts which were found in the work of Fu et al. On all datasets used AdaBins outperformed what Bhat et al. considered to be their main competitors, BTS [24] and DAV [25] when using metrics introduced by Eigen et al. [26].

Laplacian

Song et al. [27] proposed the use of a Laplacian pyramid-based decomposition technique applied to the decoding process. This was inspired by the ability of encoders the Laplacian pyramid in successfully emphasizing the difference across scale space. The encoded features are fed into stacked convolution blocks and sub-band depth residuals are generated at each pyramid level. The depth map is progressively restored from course to fine scales by combining the depth residuals at each pyramid level. This process improved the performance of predicting depth boundaries. Song et al. proposed to guide the decoding process with residuals of the input colour image and combine the predicted results from course to fine scales to reconstruct the final depth map.

Song et al. also propose to apply weight standardization to pre-activation convolution blocks, to improve the flow of gradients and makes the convergence stable without loss of performance.

Visually, the method used by Song et al. produced sharper images with less blurring at the edges of objects. It was also better at measuring the boundary of thin objects like traffic signs.

Using metrics introduced by Eigen et al. [26] the method proposed by Song et al. outperformed other methods on the KITTI dataset [28] overall. In the metric $\delta < 0.125$, the proposed method was equal to the one used by Lee et al. [29] while outperforming them in all other metrics used. In the NYU dataset [30], the proposed method outperformed other methods used in all metrics except the method used by Lee et al. [29]. In the metric $\delta < 0.125$ and Abs Rel, the method used by Song et al. was better, in RMSE the method used by Lee et al. was better and in $\delta < 1.25^2$, $\delta < 1.25^3$ and log10, the methods produced the same result. Overall, the method used by Song et al. achieved the best results over both datasets.

Vision Transformers

Ranftl et al. [31] proposed the use of dense vision transformers in place of convolutional networks as a backbone for dense prediction tasks while maintaining the encoder-decoder structure.

The vision transformer operates on a bag-of-words representation of an image [32]. Image patches that are individually embedded in feature space or deep features extracted from an image, take the role of 'words'. These are known as 'tokens'. The set of tokens are transformed using sequential blocks of multi-headed self-attention (MHSA) [33] by the transformer. These tokens are related to each other to transform the representation. Notably, a transformer keeps the number of tokens for all computations. Also, MHSA is a global operation seeing as every token can influence

every other token. Therefore, the transformer has a global receptive field at every stage after the embedding.

The vision transformer extracts a patch embedding from the image by processing all non-overlapping square patches from the image. These patches are flattened into vectors and individually embedded. Seeing as transformers are set-to-set functions, they do not store the information of the spatial positions of individual tokens. Therefore, the image embeddings are concatenated with a learnable position embedding to add this information to the representation. A special token which is not grounded in the input image and serves as the final, global image representation, which is used for classification, known as the readout token, is added.

The decoder assembles the set of tokens into image-like feature representations at differing resolutions. These are then fused together, using a three-stage Reassemble operation, into a final dense prediction.

Ranfl et al. used three architectures of Dense prediction transformers, DPT-Base, DPT-large and DPT-Hybrid. Using the KITTI dataset [28] and metrics introduced by Eigen et al. [26] he DPT-Hybrid architecture outperformed other methods across all metrics.

2.4.3 Transfer Learning

Due to the limited amount of labelled data in unstructured environments, transfer learning aims to solve this problem of limited labelled data in unstructured environments by extracting useful information from data in a related domain and transferring them to be used in the target domain [34]. In the case of this paper, there is data about structured environments, such as the KITTI dataset [28], and the Oxford Radar RobotCar Dataset [35], [36] which will be used to estimate depth in unstructured environments using transfer learning algorithms.

Pan et al. [37] have divided transfer learning into three distinct types, inductive

TL, transductive TL and unsupervised TL as shown in Figure 2.5.

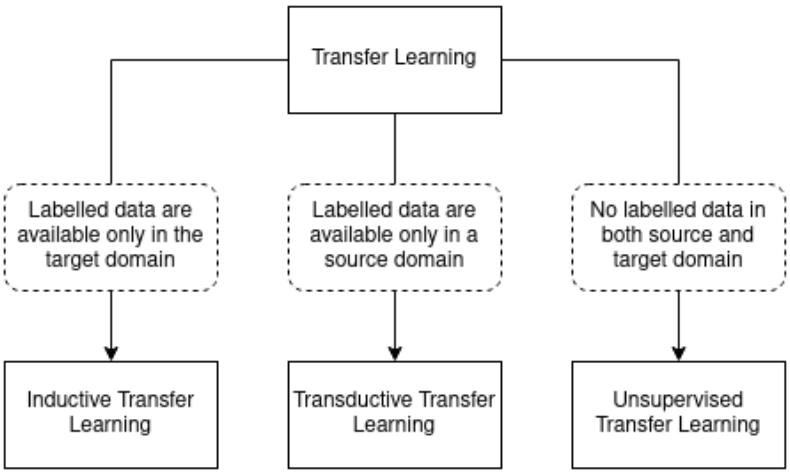


Figure 2.5: Different types of Transfer Learning, edited from [37]

Seeing as there is labelled data in the source domain, being structured environments, but not in the target domain, unstructured environments, transductive transfer learning is the approach that is best suited to this project. Seeing as our goal is depth estimation, another name for transductive learning will be used, domain adaption [37].

Transfer Learning for Depth Estimation

Transfer learning has been utilised for depth estimation. The DenseNet-169 [38] network has been used by Alhashim et al. [39] as feature encoder, while using a standard encoder-decoder architecture They showed that a well-constructed encoder can outperform other methods that rely on multistage depth estimation networks or that require multiple feature encoding layers [39]. Yeh et al. [40] also used the DenseNet-169 network as well as a simple encoder-decoder architecture. They too were able to achieve state-of-the-art performance for both indoor and outdoor depth

estimation. These serve to highlight the performance advantages of transductive transfer learning

There have also been studies specifically using domain adaption for depth estimation. One of the more recent being done by Basak et al. [41] who used a single encoder-decoder based network and were able to outperform existing state-of-the-art methods with fewer trainable parameters and small iterations. They used the KITTI dataset [28] as well as NYU Depth V2 [30] o training and testing purposes and the DenseNet-169 [38] architecture pretrained on ImageNet [42] as the encoder.

Chapter 3

Theory and Problem Formulation

3.1 Depth Network Architecture

Network architecture is an important choice in the context of monocular depth network design. The encoder-decoder design, despite only being pioneered in 2014, is one of the best performing architectures for neural networks [43]

3.1.1 Encoders

In standard encoder-decoder design, shown in Figure 3.1, the encoder performs the role of feature extractor. The encoder is generally pretrained on an image classification dataset and then repurposed as a feature extractor used for depth estimation. Once trained, the encoder will give *feature vectors* that can be used by the decoder. Some of the more popular choices for feature extractors include VGG [44], ResNet [45], ResNext [46] and DenseNet [38] with EfficientNet [47] also being used.

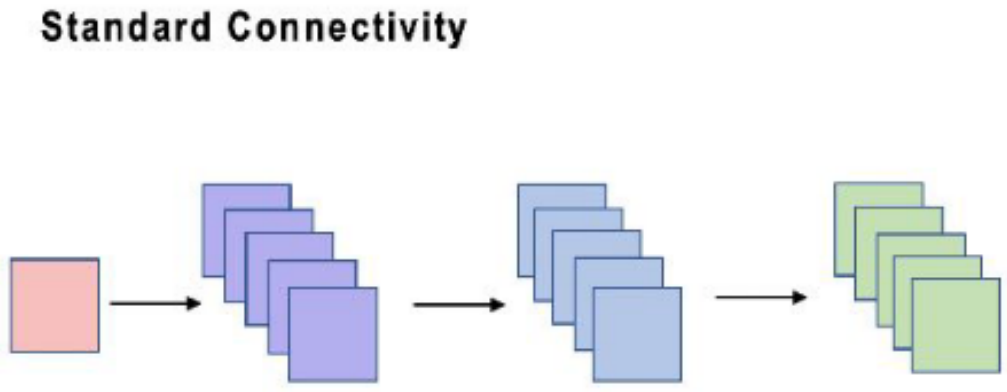


Figure 3.1: Standard Encoder Architecture [48]

ResNet

Residual networks (ResNets) use identity shortcut connections to reduce the vanishing-gradient problem[1]. These identity shortcut connections bypass one or more layers, as shown in Figure 3.2. The addition of these identity connections allows more layers to be added without saturation or loss of performance [45]

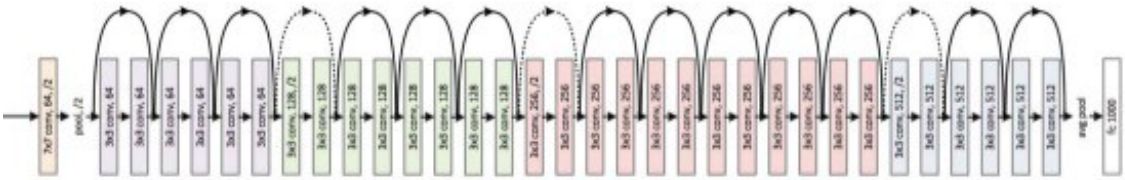


Figure 3.2: Architecture of the ResNet encoder [45]

[1] As more layers are added to a network the gradient of the loss function approaches zero [49]

ResNeXt

ResNeXt is a highly modularized network architecture for image classification. The network is constructed by repeating a building block which aggregates a set of transformations with the same topology [50]. This design results in a homogeneous, multi-branch architecture which has a few hyper-parameters to set. This creates a new dimension, named *cardinality*, which is the size of the set of transformations, as can be seen in Figure 3.3. Cardinality is an essential factor in addition to the dimensions width and depth [46].

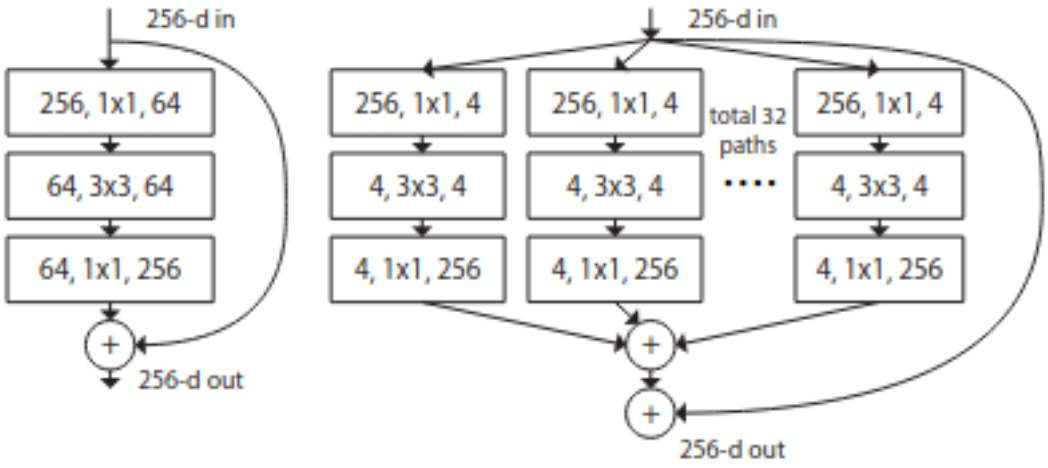


Figure 3.3: **Left**: A block of ResNet [45]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity [46]

This is the encoder architecture used in LapDepth.

DenseNet

The Dense Convolutional Network (DensNet) connects each layer to every other layer which follows it. For each layer, the feature-maps of all the preceding layers are used as inputs and its own feature-maps are used as inputs for all subsequent

layers as shown in Figure 3.4. This greatly reduces the number of parameters as well as alleviate the vanishing-gradient problem, strengthens feature propagations and encourages feature reuse [38].

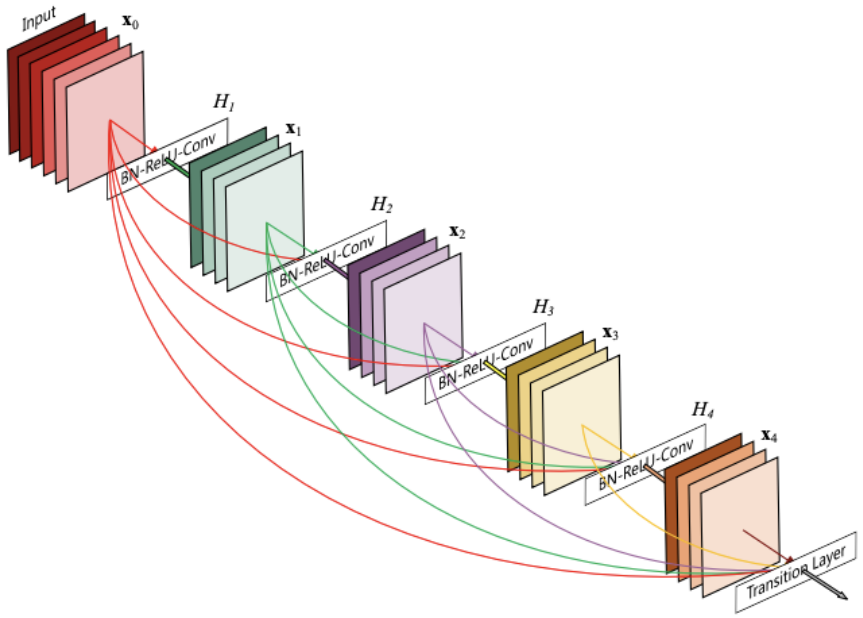


Figure 3.4: Architecture of the DenseNet encoder with each layer taking all preceding feature maps [38]

This is the encoder architecture used in BTS.

EfficientNet

EfficientNet is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of width/depth/resolution using a compound coefficient. Conventional practice is to arbitrarily scale these factors but EfficientNet uniformly scales the network width, depth and resolution with a set of fixed scaling coefficients. This method is justified by the intuition that if the input image is bigger,

then the network needs more layers to increase the receptive field and more channels to capture more fine-grained patters on the bigger image. The layout of this network can be seen in Figure 3.5

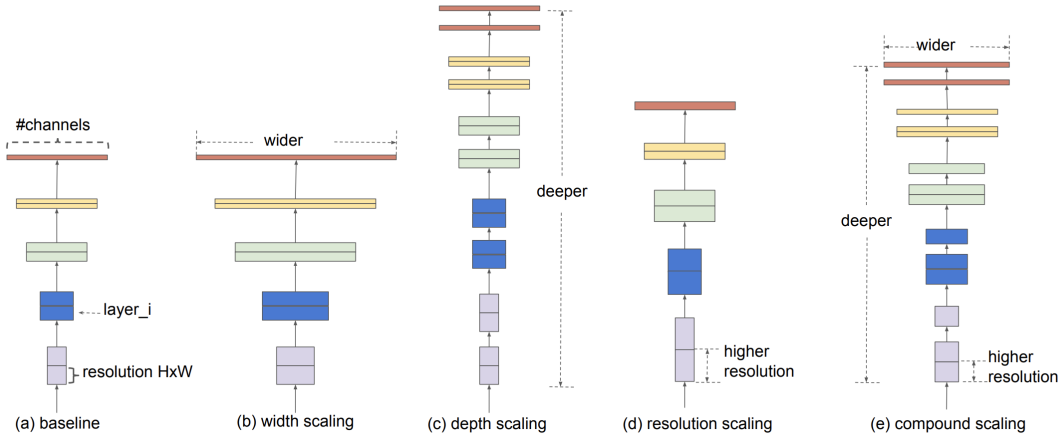


Figure 3.5: (a) is the baseline network; (b)-(d) are conventional scaling that only increase one dimensions of the network width, depth or resolution; (e) is the proposed compound scaling method that uniformly scaled all three dimensions with a fixed ratio

This is the encoder architecture used in AdaBins.

3.1.2 Decoders

The function of the decoder is to take *feature vectors* received from the encoder and construct the a predicted image with features that matter the most to make the reconstructed input recognizable as the original input. In the context of depth estimation, the decoder produces a predicted depth output.

Upsampling

Upsampling is a widely used method of semantic segmentation, to restore the low resolution feature maps received from the encoder. Generally, some form of interpolation is used for the reconstruction of pixels in image processing with each point being generated based on its neighbouring pixels by using interpolation [51]. Three common interpolation techniques used in upsampling are Nearest interpolation, Bilinear interpolation and cubic interpolation. Nearest interpolation uses the nearest pixel to generate. Bilinear interpolation estimates the pixel value by averaging the values of the two nearest pixels. Cubic interpolation estimates the pixel by using the volumes of the neighbouring pixels [52]. This is the decoder type which is used in AdaBins

Atrous Spatial Pyramid Pooling

Atrous Spatial Pyramid Pooling (ASPP) is a method which makes three main contributions to the task of semantic image segmentations. Firstly, atrous convolution allows more explicit control of the resolution that feature responses are computed at. Secondly, ASPP is able to robustly segment objects at multiple scales by probing the incoming convolutional feature layer with filters at differing sampling rates and effective fields-of-view. Thirdly, it improves localization of object boundaries [53]. A visual representation of this can be seen in Figure 3.6.

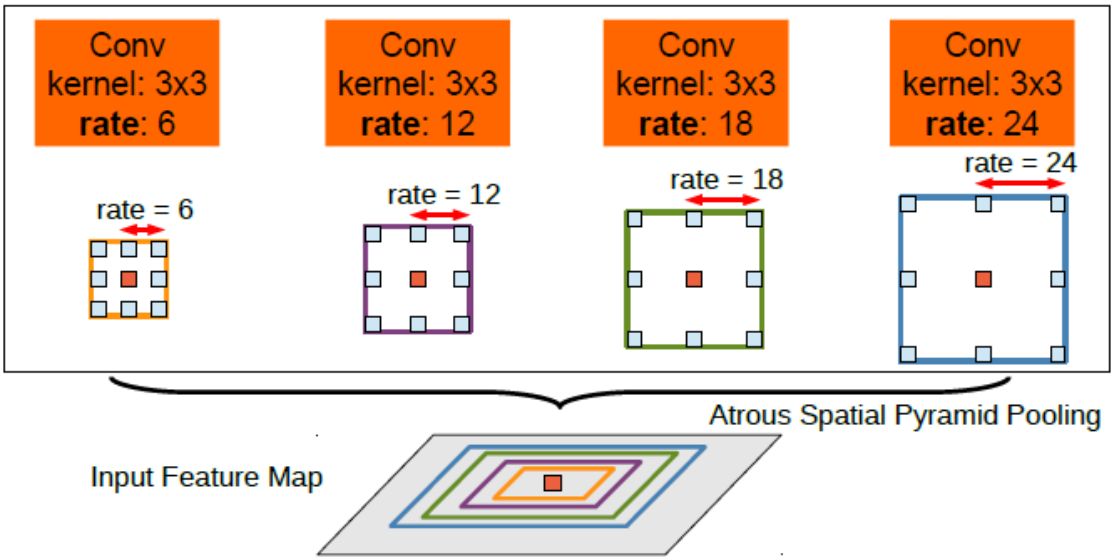


Figure 3.6: To classify the central pixel (orange), ASPP exploits multi scale features by employing multiple parallel filters with different rates. These are then concatenated to form a single image [53]

This is the decoder type which is used in BTS

Laplacian Pyramid

The Laplacian Pyramid is a method developed by Song et al [27] and is used in the LapDepth network. As was explained in Chapter 2, it used an ASPP block as well as bilinear upsampling to produce an estimated depth map. This can be seen in Figure 3.7.

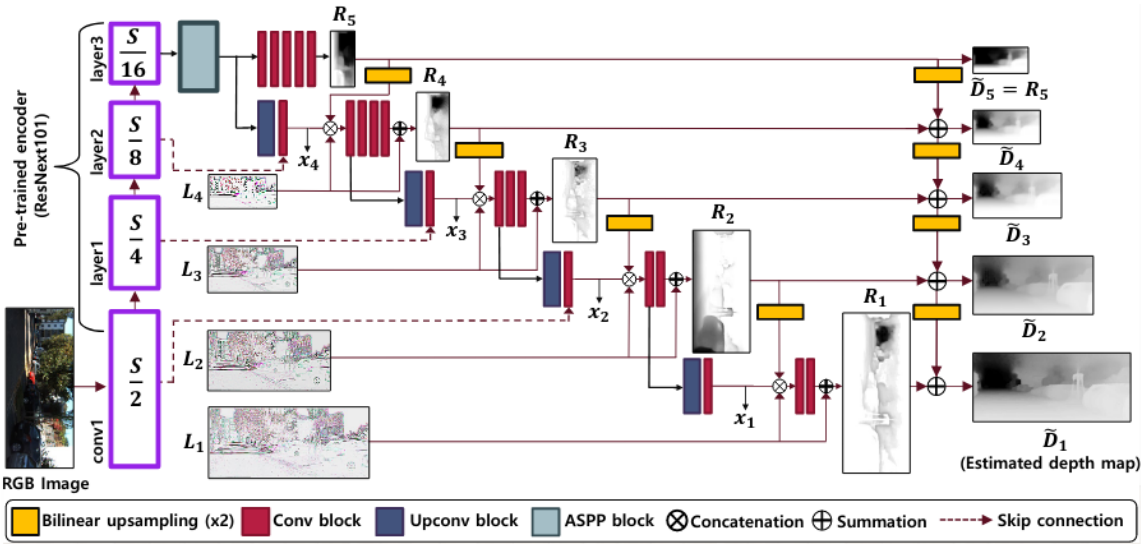


Figure 3.7: The architecture used in LapDepth to implement the architecture of Laplacian Pyramid-Based Depth Residuals [27]

3.2 The Depth Network Selection

In order to investigate the performance of state of the art monocular depth perception networks in unstructured environments, three depth networks were chosen with different encoders, decoders and loss functions will be analysed. The networks were chosen based on state of the art performance, as well as how recently they were developed. Furthermore, all three networks selected are based on an encoder-decoder architecture, thus ensuring a fair and accurate comparison of network designs.

The three depth networks that were chosen are Adaptive Bins (AdaBins) by Bhat et al. [22], Laplacian Pyramid-Based Depth Residuals (LapDepth) by Song et al [27] and Big to Small (BTS) by Lee et al. [24]. The reason that these three networks were chosen is that, at the time of writing, they are some of the best performing networks on the KITTI [28] dataset which do not require extra training data. AdaBins is ranked first, LapDepth is ranked second and BTS is ranked third. BTS is also the

best performing network in the project **Depth Networks in Unstructured Environments** [48], of which this is a continuation. These networks also have models trained on the NYU Depth v2 [30] however this is an indoor dataset. With KITTI being an outdoor structured dataset, it is closer to the desired outdoor unstructured environment and therefore the models trained on the KITTI dataset will be used.

A summary of these models can be seen in Table 3.1 showing the name, author and the model that was used for evaluation

Depth Network	**Author**	**Model used**	**Ranking on the KITTI Eigen split**
AdaBins [22]	Bhat et al.	tf_efficientnet_b5_ap	1
LapDepth [27]	Song et al.	LDRN_KITTI_ResNext101_pretrained_data	2
BTS [24]	Lee et al.	bts_eigen_v2_pytorch_densenet161	3

Table 3.1: State of the art depth networks which were selected. All were trained on the KITTI dataset

All the networks, while using encoder-decoder architectures, have different encoders and decoders. All the networks use different methodologies to determine depths. AdaBins uses adaptive bins to discretize the depth interval into N bins where the interval is fixed for a given dataset. LapDepth applies the Laplacian pyramid-based decomposition technique to the decoding process. Specifically Laplacian residuals to the colour image guide encoded features to be generated as the depth residual containing local details. These details appropriately represent depth properties of different scale spaces. The BTS network uses specialised Local Planar Guidance modules to provide local geometric information. Therefore, these three networks include a variety of architectures while representing the state of the art in monocular depth perception.

3.2.1 Evaluation

The first thing that will be done is evaluate the current depth networks qualitatively and quantitatively. This will be done on the unstructured data. The qualitative analysis will constitute producing predicted depth heat maps which can be compared with each other as well as the quantitative analysis which will be done by calculating the performance metrics based on ground truth data. This will enable analysis of all three depth networks to take place on both structured and unstructured environments both before and after they have been trained.

3.2.2 Adaption

The selected depth networks will be adapted to work in unstructured environments by the use of transfer learning. These depth networks will then be fine tuned on the unstructured dataset. They will then be evaluated in order to determine how effective the transfer learning was.

3.3 Transfer Learning

In order to adapt the depth networks, which have been trained in structured environment, to unstructured environment, transfer learning will be used. Transfer learning is a method used in machine learning where a model developed for a task is used as a staring point for a model of a different but similar task. This reduces the need for both time and computing resources which would be required to develop a depth network. It also allows the new model to be trained with significantly less input data than if it were to be trained from scratch

3.3.1 Fine Tuning and Domain Adaption

Fine tuning is a commonly used transfer learning technique whereby some layers of the pretrained network are retrained on the new data. This is done by freezing some of the model parameters of the network before training, therefore only a part of the new network is trained on the new data. In most architectures, the initial layers of a network are more general while the later layers are more specific. As such, these initial layers are the ones which are frozen and the later layers are trained, seeing as they are more task specific. The number of layers to retrain will differ depending on the task and will be decided on through experimentation. It may be different depending on which depth network is used.

As shown in Figure 2.5, the specific type transfer learning that will be used is Transductive Transfer Learning, more commonly referred to as domain adaption. Domain adaption is when the task performed is the same however the source and target domains are different. In the context of this project, the task of depth perception is being adapted from the structured domain to the unstructured domain. The change in source and target domains is what will require fine tuning.

3.3.2 Loss Functions

Loss functions are used in machine learning to determine whether a prediction is good or not. When training, optimization functions are used in order to reduce minimize the loss and therefore make better predictions. The loss functions used by the chosen depth networks are as follows:

Scale Invarient Loss

Scale-invariant Loss was proposed by Eigen et al [26] and uses their *scale-invariant mean squared error* (in log space). They realised that finding the average scale of the

scene accounted for a large fraction of the error so they implemented scale-invariant error. This measures the relationships between the points in a scene irrespective of the absolute scale [26]. The function for Scale-invariant loss can be seen in Equation 3.1

$$\mathcal{L}(y, \hat{y}) = \frac{1}{n}\sum_{i} d_i^2 - \frac{\lambda}{n^2}(\sum_{i} d_i)^2 \tag{3.1}$$

where d_i = logy_i - log$\hat{y}_i$, $\lambda \in [0, 1]$, y is the predicted depth and $\hat{y}_i$ is the ground truth depth.

LapDepth [27] adapts this loss function and takes the square root in order to alleviate the problem of imbalance. The resulting equation can be seen in Equation 3.2

$$\mathcal{L}(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{i} d_i^2 - \frac{\lambda}{n^2}(\sum_{i} d_i)^2} \tag{3.2}$$

The balancing factor, λ, was set to 0.85.

Both AdaBins [22] and BTS [24] take this further and add a scaling factor α resulting in Equation 3.3

$$\mathcal{L}(y, \hat{y}) = \alpha\sqrt{\frac{1}{n}\sum_{i} d_i^2 - \frac{\lambda}{n^2}(\sum_{i} d_i)^2} \tag{3.3}$$

Where α was set to 10 in both depth networks.

Bin-center density loss

AdaBins [22] due to its architecture also has a loss function called bin-center density loss. This loss term encourages the distribution of bin centers to follow the distribution of depth values in the ground truth [22]. This equation can be seen in

Equation 3.4

$$\mathcal{L}_{bins} = chamfer(X, c(\mathbf{b})) + chamfer(c(\mathbf{b}), X) \tag{3.4}$$

Where the bin centers are $c\mathbf{b}$, the set of all depth values in the ground truth image are X and bi-directional Chamfer Loss [54] is used as a regulizer

This is then added to the loss function to give Equation 3.5 as the loss function used by AdaBins [22]

$$\mathcal{L}_{total} = \mathcal{L}(y, \hat{y}) + \beta \mathcal{L}_{bins} \tag{3.5}$$

where β is set to 0.1

Chapter 4

Evaluation of State of the Art

4.1 The Datasets

4.1.1 Unstructured Dataset

The unstructured dataset was captured using a Husky Unmanned Ground Vehicle. A Bumblebee®2 FireWire camera, to capture the raw images, and a Velodyne Lidar® HDL-32E to capture the ground truth data. This setup can be seen in Figure 4.1 on the right. The images were captured in the surrounds of the University of Cape Town.

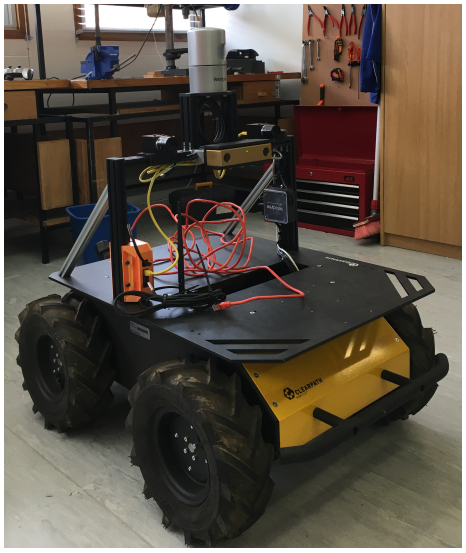


Figure 4.1: Image of the vehicle that was used to capture the images for the dataset with all its attachments

4.1.2 KITTI dataset

The KITTI Vision Benchmark Suite [28] is the most commonly used dataset for computer vision tasks such as object detection

and tracking, sceneflow and depth estimation. The KITTI dataset is made up of many scenes from outdoor, structured, urban environments such as building, pedestrians, vehicles, tress and roads. The KITTI dataset is a commonly used by state of the art networks for training and evaluation.

4.2 Evaluation

As mentioned in Chapter 3, the evaluation is divided into two parts, namely, qualitative and quantitative.

4.2.1 Qualitative

All of the networks being used are able to take in an image of a scene, both structured or unstructured, and return a converted image giving a visual representation of the depths of the items in the image. These images are in different formats to each other which can make comparison unnecessarily difficult. Therefore, they will all be converted to the same colour scheme so it is easier to compare them. The colour scheme chosen is the "jet" colour scheme.

4.2.2 Quantitative

In order quantitatively evaluate the networks, evaluation metrics are used. Many of these metrics calculate some form of pixel-wise error by utilising the difference between the ground truth depth, y, and the predicted depth $\hat{y}$ of every pixel in the image. The error metrics that will be used, were popularised by Eigen et al. [26] and are divided into two categories, **Accuracy metrics** and **Inlier metrics** and they are listed below.

Accuracy metrics

Absolute Relative error (Abs Rel):

$$\text{Absolute Relative error (Abs Rel)} = \frac{1}{n}\sum_{i}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \tag{4.1}$$

Squared Relative error (Sq Rel):

$$\text{Squared Relative error (Sq Rel)} = \frac{1}{n}\sum_{i}^{n}\frac{(y_i - \hat{y}_i)^2}{y_i} \tag{4.2}$$

Root Mean Squared Error (RMSE):

$$\text{Root Mean Squared Error (RMSE)} = \sqrt{\frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2} \tag{4.3}$$

Root Mean Squared Log Error (RMSE Log):

$$\text{Root Mean Squared Log Error (RMSE Log)} = \sqrt{\frac{1}{n}\sum_{i}^{n}(\log(y_i) - \log(\hat{y}_i))^2} \tag{4.4}$$

Inlier metrics

Threshold Accuracy (δ_i):

$$\text{Threshold Accuracy } (\delta_i) = \%\text{ of y such that } \max(\frac{y_p}{\hat{y}_p}, \frac{\hat{y}_p}{y_p}) = \delta < thr \tag{4.5}$$

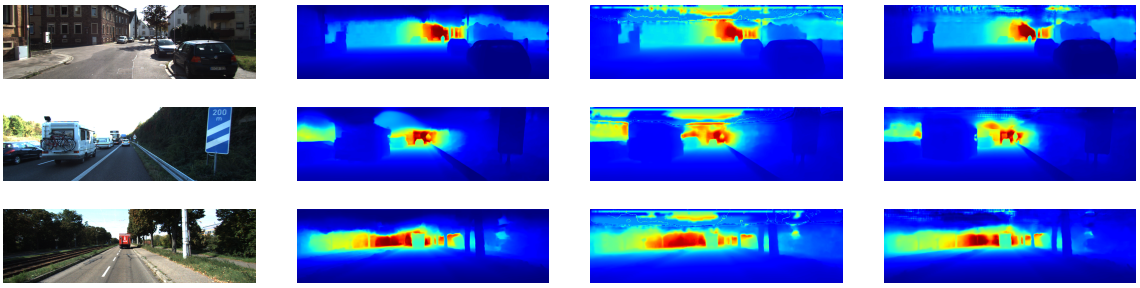
for thr = 1.25, 1.25^2, 1.25^3, where n is the number of pixels, $\hat{y}_i$ is the predicted depth value and y_i is the ground truth depth.

Accuracy metrics calculates the average error over all the pixels while inlier metrics measure the proportion of pixels in the depth map which fall within a specific threshold. The metrics expounded upon in Equations 4.1 to 4.5 will be used for all quantitative analysis which takes place in this report as these are commonly used in monocular depth perception

4.3 Results on the KITTI dataset

The depth networks will first be evaluated on the KITTI dataset. This will act as a benchmark when comparing results on the unstructured dataset.

4.3.1 Qualitative Results



(i) Raw Image (j) AdaBins [22] (k) LapDepth [27] (l) BTS [24]

Figure 4.2: Images showing the depth map produced by the selected Depth Networks on the KITTI [28] dataset

As is expected, Figure 4.2 shows that all the depth networks perform well on the KITTI [28] dataset. This is to be expected as they are the state of the art when trained on this dataset.

4.3.2 Quantitative Results

	Higher is better			Lower is better			
Method	δ_1<1.25	δ_2<1.25^2	δ_3<1.25^3	AbsRel	SqRell	RMSE	RMSE *log*
AdaBins [22]	0.964	**0.995**	**0.999**	**0.058**	**0.190**	**2.360**	**0.088**
LapDepth [27]	**0.965**	**0.995**	**0.999**	0.059	0.201	2.397	0.090
BTS [24]	0.955	0.993	0.998	0.060	0.249	2.798	0.096

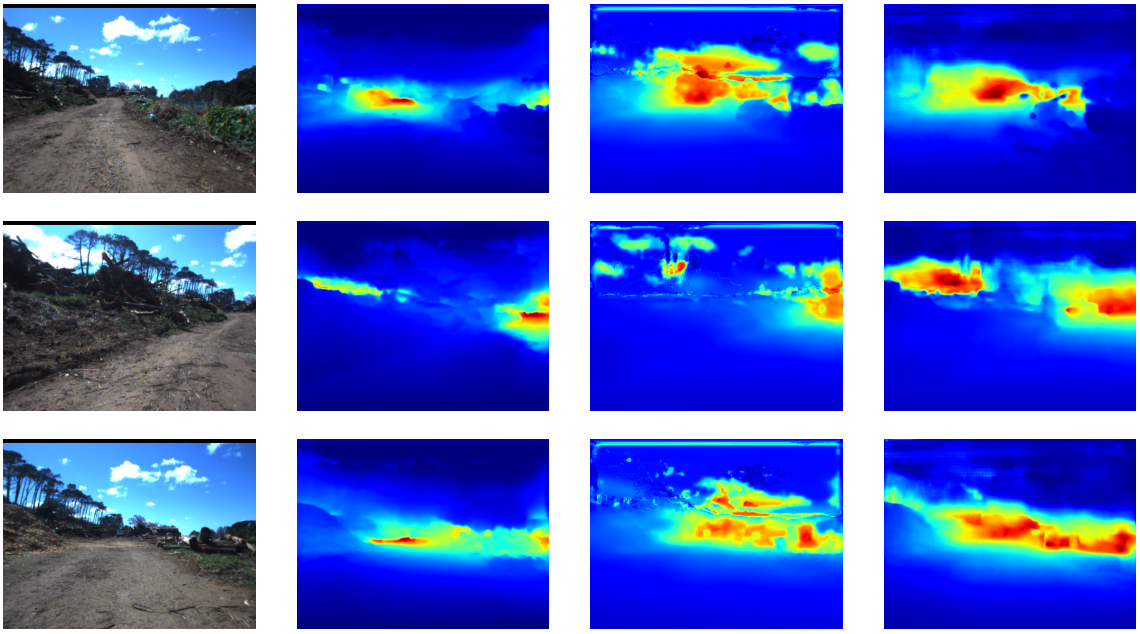
Table 4.1: Table showing the metrics achieved by the selected Depth Networks on the KITTI [28] dataset

The quantitative results on the KITTI dataset confirm what is already known. AdaBins has the best performance, followed by Lapdepth with BTS trailing the two. AdaBins performs the best on all of the accuracy metrics. When it come to the Inlier metrics, AdaBins and LapDepth are equal for $\delta_1 < 1.25^2$ and $\delta_2 < 1.25^3$ and LapDepth narrowly outperforms AdaBins for $\delta_3 < 1.25$. This can all be seen in Table 4.1

4.4 Unstructured

In order to adapt the networks from use with the KITTI [28] dataset to use the unstructured datasets, the ground truth data had to be scaled by multiplying it by 256.0 for all three depth networks.

4.4.1 Qualitative Results



(i) Raw Image (j) AdaBins [22] (k) LapDepth [27] (l) BTS [24]

Figure 4.3: Images showing the depth map produced by the chosen Depth Networks on the unstructured dataset

As can be seen in Figure 4.3, AdaBins appears to have an ok performance while BTS and LapDepth do not perform particularly well. in BTS and LapDepth here is no minimal distinction between the tress, it mostly looks like one blur whereas AdaBins seems to have sharper edges where the trees are. It also looks like the clouds cause problem for LapDepth. This could be because in the KITTI dataset, due to it mostly being captured in urban environments, has very little outside of the maximum depth. This could be causing LapDepth to register the clouds as the maximum depth as opposed to outside of the scope of the range. It should also be noted that the images produced by LapDepth had distortion. This is consistent throughout models and datasets and will therefore not be taken into consideration

when results are analyzed.

4.4.2 Quantitative Results

	Higher is better			Lower is better			
Method	δ_1<1.25	δ_2<1.25^2	δ_3<1.25^3	AbsRel	SqRell	RMSE	RMSE *log*
AdaBins [22]	**0.519**	**0.793**	**0.860**	**0.273**	**6.613**	24.833	**0.551**
LapDepth [27]	0.035	0.070	0.121	2.007	18.038	10.583	1.102
BTS [24]	0.051	0.131	0.274	1.234	6.714	**6.670**	0.822

Table 4.2: Table comparing the quantitative results achieved selected depth networks on the unstructured dataset

As can be seen in Table 4.2, BTS [24] outperforms LapDepth [27] on every metric. This is unexpected seeing as LapDepth performed better on the KITTI [28] dataset. The δ metrics were surprisingly low considering that in Figure 4.3 it would appear that more than 3.5% and 5.1% of the predicted depths were in range of 1.25 of ground truth. AdaBins [22] far outperforms the other networks in all metrics except RMSE. This is in line with what can be seen in Figure 4.3. It is also expected that it would perform the best seeing as it is the best performing network on the KITTI dataset [28].

Chapter 5

Adaption of the Depth Networks and Evaluation

5.1 Transfer Learning

A key reason for the popularity of transfer learning is the fact that is takes less time and fewer resources than would be required to train a new network. As mentioned in Chapter 3, a combination of domain adaption, whereby task the encoder will changed from image classification to depth estimation, and fine tuning will be used.

5.2 Implementation

The methods used on Laplace [27] and BTS [24] are similar in nature and will be expounded upon simultaneously. The architecture of the ResNet [45] and ResNeXt [46], which are the encoders used in these two networks, are similar enough that the methodology can be explained at the same time.

Freezing and unfreezing parameters

In order to freeze and then selectively unfreeze layers, the code in Listing 5.1 was used. First, all the parameters in the network were frozen. Then, some of the parameters were unfrozen, going from most specific to more general. The freezing was done by disabling the gradient of each parameter and then the gradient was re-enabled for the parameters that were going to be trained.

Listing 5.1: Code used to implement transfer on the Lapdepth network

```
unfreeze_layers = ['layer3.22', 'layer3.21', 'layer3.20',
                   'layer3.19', 'layer3.18']
# a list of the layers that I want to unfreeze
for param in Model.module.parameters():
    param.requires_grad = False  # freeze everything

for name, child in Model.module.named_children():
    if 'encoder' not in name:
        continue
    for name2, parameters in child.named_parameters():
        # print(name, name2)  # this prints out the
            # name of the layers
        # then the layers that I want unfrozen I add
        # to the list unfreeze_layers
        if any(x in name2 for x in unfreeze_layers):
            parameters.requires_grad = True
            # unfreeze the chosen layers
```

It should be noted that when implementing for BTS, "Model.module" was replaced with "model"

AdaBins [22] is a depth network which already implements transfer learning. It does this by setting the global pool and the classifier of the basemodel to an identity operator which is argument-insensitive. This can be seen in Listing 5.2

Listing 5.2: code in AdaBins that was used to implement transfer learning

```
basemodel.global_pool = nn.Identity()
basemodel.classifier = nn.Identity()
```

Implementation Details

The unstructured dataset was split into a training/evaluation split of 80/20 which gave 400 training images and 100 evaluation images. The training for LapDepth and BTS was implemented on a PC running Ubuntu 18.04 with an NVIDIA RTX2070 GPU. A batch size of 3 was used as this was the maximum batch size that the GPU could process and the number of epochs was 25. The training for AdaBins was implemented on the Google Colab environment due to specific software requirements. However, when attempts to save the trained model were made, the runtime would crash. As such, only the quantitative results will looked at for AdaBins post training.

5.2.1 Training Results

When trying to determine the best number of layers to train, the depth network was trained numerous times with a different amount of frozen layers. The training loss for each depth network was then recorded, tabulated and graphed.

5.2.2 AdaBins

At the completion of training, the total loss of the network was 1.289. This would imply that the training was effective as this the sum of both the normal function as

well as bin-center density loss.

BTS

Layer(s) trained	Number of Learning parameters	Training Loss
1	502 368	3.857
5	2 418 720	1.024
10	4 604 640	0.897
15	6 557 760	0.879
20	8 278 080	0.849
Full (24)	9 486 720	0.776

Table 5.1: The relationship between the number of layers frozen, the number of learning parameters and training loss when the BTS depth network is trained

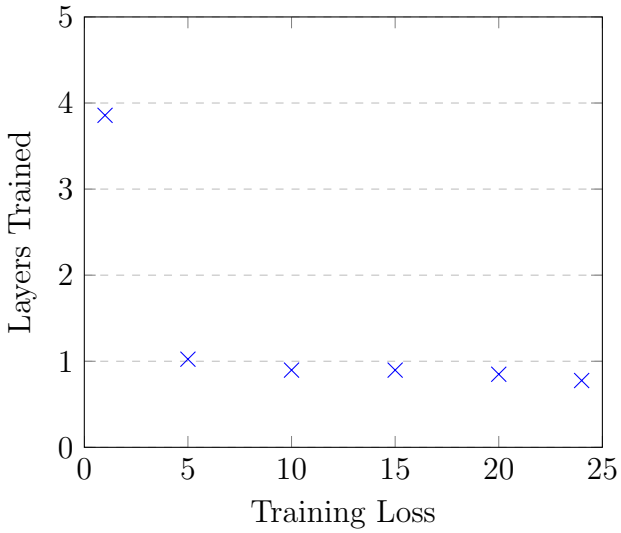


Table 5.2: The relationship between the number of layers trained and the training loss in the BTS depth network

As can be seen in Table 5.1 and Graph 5.2 above, the relationship between training loss and layers trained is clearly non-linear. In fact, it looks like it could be approximated by and inverse exponential function. From Graph 5.2 it can be seen that the difference in training loss with 1 layer frozen is significantly greater than when 5 layers are frozen. The difference in loss is 2.833 which is a reduction of 73.5% decrease. In comparison, the difference between 10 layers and a full 24 layers is 0.121 which is a reduction of 13.5%. This is much lower, however, this difference in training loss is not insignificant.

Laplace

Layer(s) trained	Training Loss
1	1.658
5	1.404
10	1.360
15	1.323
Full (22)	1.137

Table 5.3: The relationship between the number of layers frozen, the number of learning parameters and training loss when the LapDepth network is trained

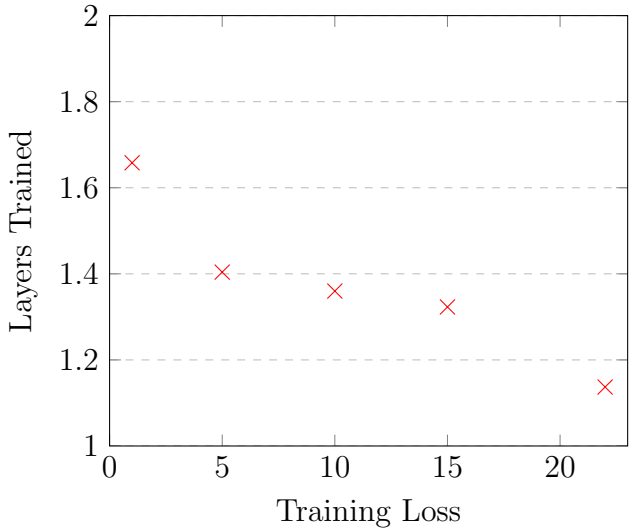


Table 5.4: The relationship between the number of layers trained and the training loss in the LapDepth network

As can be seen in Table 5.3 and Graph 5.4 above, the relationship between training loss and layers trained is slightly non-linear. However, it appears to be more linear than in BTS, especially from 5 layers frozen onward where it appears to be linear. The difference in training loss with 1 layer frozen and when 5 layers are frozen is 0.254 which is a reduction of 15.3%. In comparison, the difference between 10 layers and a full 24 layers is 0.223 which is a reduction of 16.4%.

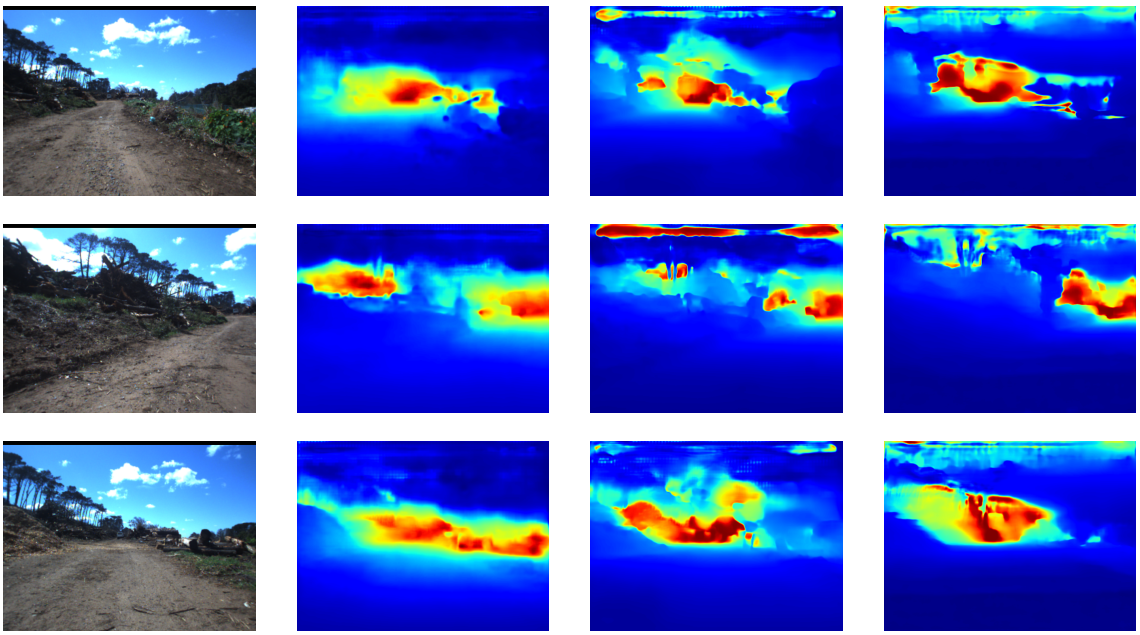
5.3 Evaluation

The results from BTS [24] and LapDepth [27] will be examined at 1 frozen layer, 10 frozen layers and fully frozen. The results from AdaBins [22] will be examined only when it was trained seeing as it is a network which already implements transfer

learning when training. This will be done only quantitatively for reasons mentioned above

5.3.1 BTS

Qualitative



(i) Raw Image (j) Untrained (k) 1 layer (l) Fully Trained

Figure 5.1: Image comparisons of the BTS [24] depth network when trained to different levels on the unstructured dataset

From Figure 5.1 it looks like there is minimal improvements when 1 layer was trained but much greater improvements when all the layers were trained. When 1 layer was trained there was some improvement in the detection of the trees but it is not great. When all the layers were trained, the detection of the trees was much better. The road directly in from of the Husky was also interpreted to be closer than

when the depth network was untrained and when 1 layer was trained.

Quantitative

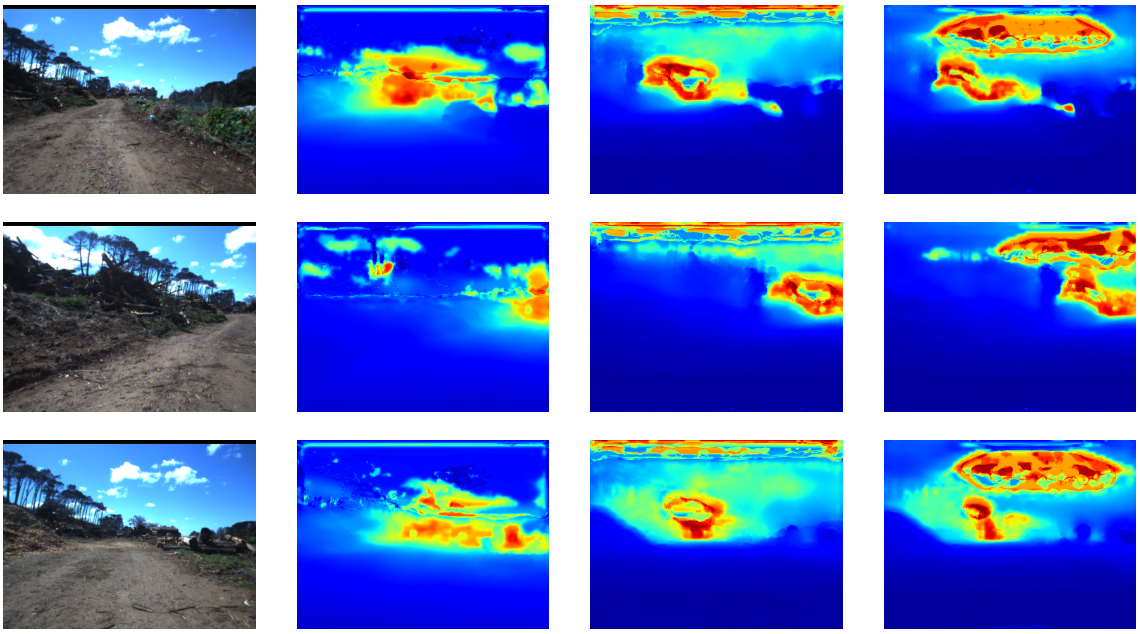
	Higher is better			Lower is better			
Layers Frozen	δ_1<1.25	δ_2<1.25^2	δ_3<1.25^3	AbsRel	SqRell	RMSE	RMSE *log*
Untrained	0.051	0.131	0.274	1.234	6.714	6.670	0.822
1	0.045	0.136	0.347	1.106	5.684	6.441	0.756
10	0.918	0.981	**0.993**	0.093	0.305	2.423	0.145
Full (24)	**0.927**	**0.982**	**0.993**	**0.083**	**0.287**	**2.289**	**0.137**

Table 5.5: Quantitative results achieved on the BTS [24] network when trained at different levels

As was expected, when the network was trained fully (24 layers), the results were the best across all metrics except δ_3 <1.25^3 where it was the same as when 10 layers were frozen. It should also be noted that none of the error metrics improved significantly from when 10 layers were frozen and 24 layers. This falls in line with what was seen in Table 5.1 and Graph 5.2 where the reduction of losses was also very small.

5.3.2 Laplace

Qualitative



(i) Raw Image (j) Untrained (k) 1 layer (l) Fully Trained

Figure 5.2: Image comparisons of the LapDepth [27] network when trained to different levels on the unstructured dataset

The results are surprising especially when compared with Table 5.6. The main visual difference is that the road in front of the vehicle is a darker blue, implying it is closer, after training. Seeing as this accounts for more than half the image in each case, this could be the reason for the large jump in quantitative results while the difference is not visually obvious. On top of that, when fully trained, it seems like the clouds are being picked up in all three examples. There are some clear improvements though. The edges of the trees on the side of the road are definitely sharper after training in all three images.

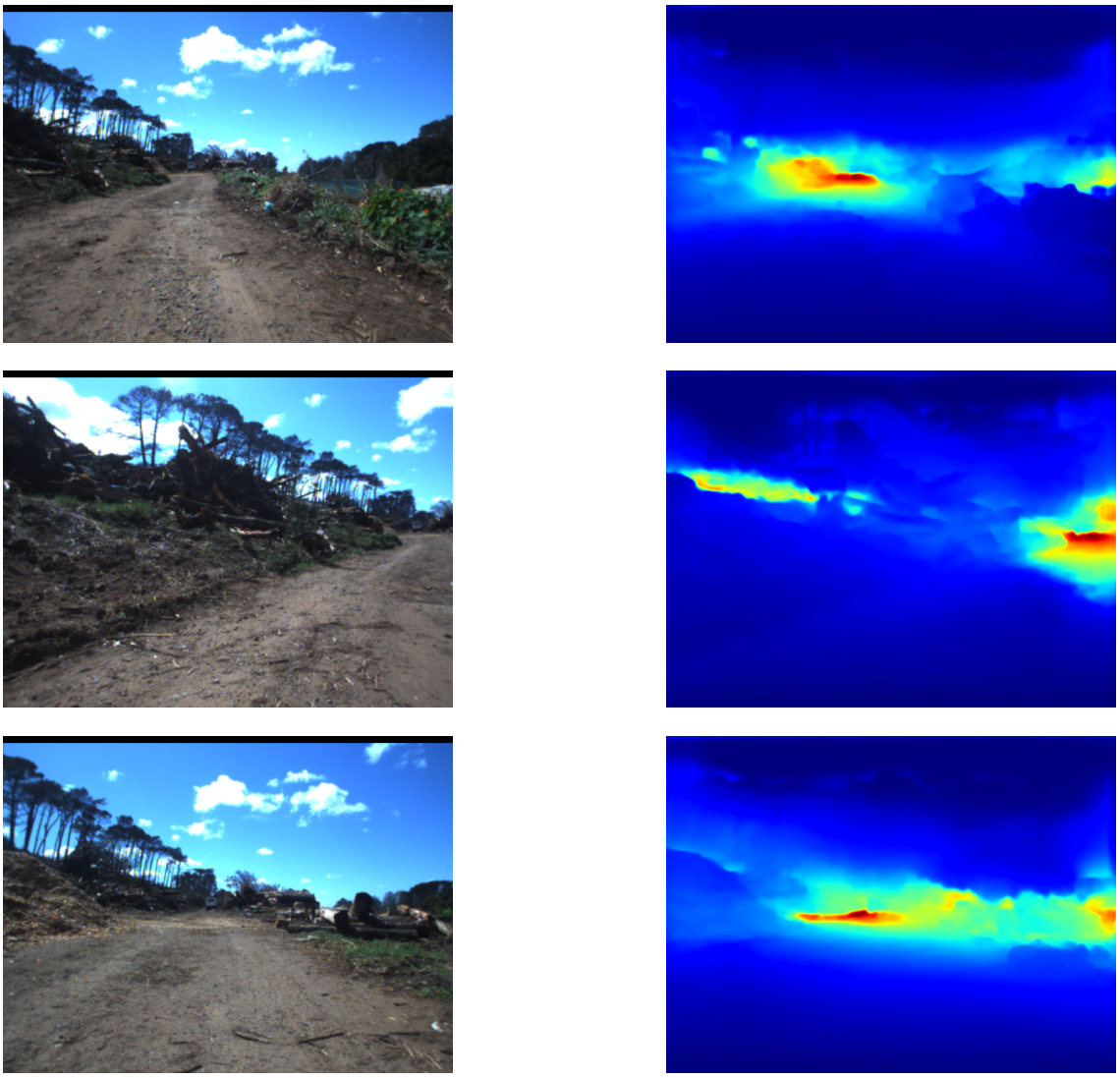
Quantitative

	Higher is better			Lower is better			
Layers Frozen	δ_1<1.25	δ_2<1.25^2	δ_3<1.25^3	AbsRel	SqRell	RMSE	RMSE *log*
Untrained	0.035	0.070	0.121	2.007	18.038	10.583	1.102
1	0.804	0.975	0.992	0.171	0.418	2.665	0.194
10	0.907	0.982	0.993	0.122	0.322	2.394	0.157
Full (22)	**0.913**	**0.983**	**0.994**	**0.116**	**0.291**	**2.318**	**0.155**

Table 5.6: Quantitative results achieved on the LapDepth [27] network when trained at different levels

Unsurprisingly, when LapDepth was trained fully (22 layers) it had the best performance in every metric. However, the difference in the metrics between 10 layers and a full 22 layers is very small. This is to be expected when looking at Table 5.3 and Graph 5.4. Even the difference between 1 layer and 22 layers is not very large. It is interesting to see that the majority of the improvements happen in the training of the first layer, after which the improvements are much smaller. This is different to BTS where the majority of the improvements come when more layers are frozen.

5.3.3 AdaBins



(e) Raw Image

(f) AdaBins [22]

Figure 5.3: Images showing the depth map produced by AdaBins [22] on the unstructured dataset

The images in Figure 5.3 show the depth map produced by AdaBins [22] before it was trained. There are depth maps produced by the trained networks so these are just a reference point.

5.3.4 Quantitative Results

	Higher is better			Lower is better			
Method	δ_1<1.25	δ_2<1.25^2	δ_3<1.25^3	AbsRel	SqRell	RMSE	RMSE *log*
Untrained	0.519	0.793	0.860	0.273	6.613	24.833	0.551
Trained	0.840	0.970	0.989	0.104	0.400	3.227	0.178

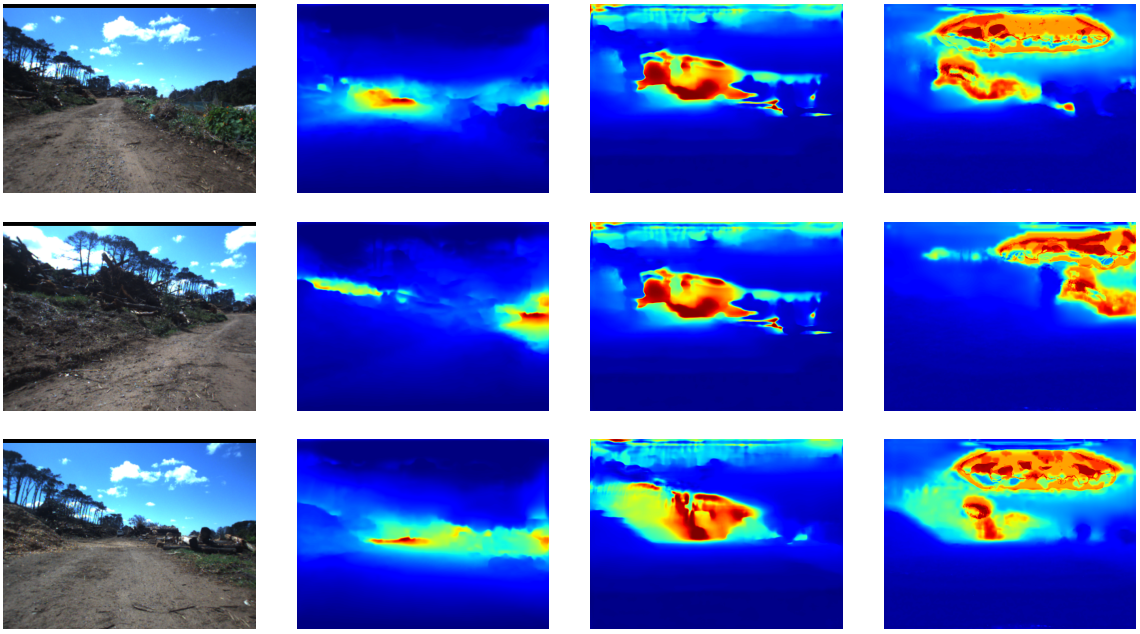
Table 5.7: Quantitative results achieved on AdaBins [22] before and after being trained

As can be seen in Figure 5.7, the depth network makes significant improvements across all metrics. δ_1 has increased by 32.1% to 84% and both δ_2 and δ_3 are 97% and above. The greatest improvement in the accuracy metrics was seen in RMSE. The trained value is over 7.6 times less than the value in the untrained network.

5.4 Fully Trained Network Comparison

In the chapter above, it has been shown how transfer learning has improved each of the network's performance on the unstructured data. The fully trained depth networks will now be compared with each other and the results will be analysed.

Qualitative Results



(i) Raw Image (j) AdaBins [22] (k) BTS [24] (l) LapDepth [27]

Figure 5.4: Images comparisons of the trained depth networks on the unstructured dataset (The images for AdaBins [22] are untrained)

All of the networks appear to have an acceptable performance. Comparing BTS and LapDepth it would appear that LapDepth is slightly better when the distortion is not taken into account. The mapping of the obstacles and trees on the side of the road looks sharper.

5.4.1 Quantitative Results

	Higher is better			Lower is better			
Method	δ_1<1.25	δ_2<1.25^2	δ_3<1.25^3	AbsRel	SqRell	RMSE	RMSE *log*
AdaBins [22]	0.840	0.970	0.989	0.104	0.400	3.227	0.178
BTS [24]	**0.927**	0.982	0.993	**0.083**	**0.287**	**2.289**	**0.137**
LapDepth [27]	0.913	**0.983**	**0.994**	0.116	0.291	2.318	0.155

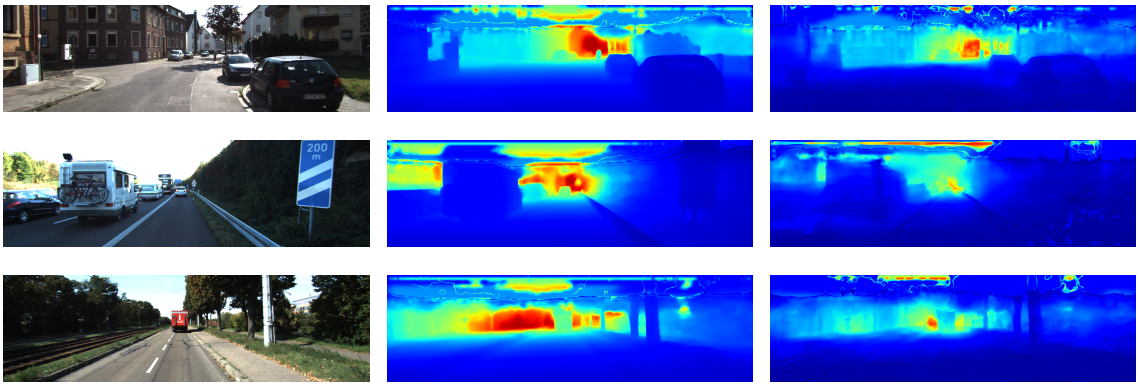
Table 5.8: Quantitative comparison of the trained depth networks on the unstructured dataset

As can be seen in Table 5.8, BTS outperforms both networks in all the accuracy metrics as well as in δ_1. In δ_2 and δ_3 it is only outperformed by LapDepth 0.1%. AdaBins is the worst performing network. It has the worst value for every metric except AbsRel. These results are unexpected. The ranking of performance is completely flipped from the performance on the KITTI [28] dataset.

5.5 Evaluation of Trained Depth Networks on the KITTI Dataset

5.5.1 LapDepth

5.5.2 Qualitative Results



(g) Raw Image (h) Untrained (i) Trained

Figure 5.5: Image comparisons of the LapDepth [27] depth network on the KITTI [28] before and after it was trained

The results from Figure 5.5 show that there is a significant drop in performance after the network has been trained. The signs, poles and vehicles are much blurrier and there doesn't appear to be much structure after the network has been trained.

5.5.3 Quantitative Results

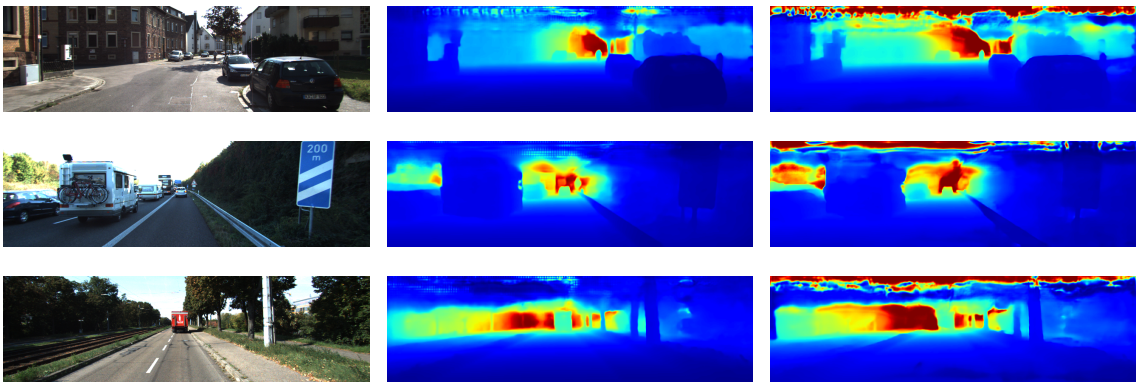
	Higher is better			Lower is better			
Method	δ_1<1.25	δ_2<1.25^2	δ_3<1.25^3	AbsRel	SqRell	RMSE	RMSE *log*
Untrained	0.965	0.995	0.999	0.059	0.201	2.397	0.090
Trained	0.124	0.299	0.524	0.443	2.919	7.471	0.698

Table 5.9: Quantitative comparison of LapDepth [27] on the KITTI [28] dataset before and after it was trained

The results shown in Table 5.9 confirm what can be seen in Figure 5.5. The drop in performance is much more severe than what was expected from only looking at the figure. The only inlier metric to go above 50% was δ_3 and all of the accuracy metrics increased quite a lot

5.5.4 BTS

5.5.5 Qualitative Results



(g) Raw Image (h) Untrained (i) Trained

Figure 5.6: Image comparisons of the BTS [24] depth network on the KITTI [28] dataset before and after it was trained

There does seem to be a bit of drop in performance but not a lot. The hard lines of the signs and structures do look slightly fuzzier in the trained network but it still looks passable. The main difference is at the top of the images where the trained networks perceives it at max depth while the untrained network does not.

5.5.6 Quantitative Results

	Higher is better			Lower is better			
Method	δ_1<1.25	δ_2<1.25^2	δ_3<1.25^3	AbsRel	SqRell	RMSE	RMSE *log*
Untrained	0.955	0.993	0.998	0.060	0.249	2.798	0.096
Trained	0.844	0.937	0.979	0.117	0.485	3.202	0.184

Table 5.10: Quantitative comparison of BTS [24] depth network on the KITTI [28] dataset before and after it was trained

The results in Table 5.9 match with what can be seen in Figure 5.5. There is a reduction in performance on the trained network but it is small. In terms of the Inlier metrics, δ_1 was reduced by 0.111 to be 84.4%. Both δ_2 and δ_3 both remained above 90% which is quite good considering it was trained on a different domain.

5.6 Comparison of metrics

In this section, the quantitative metrics will be compared when the depth networks are evaluated on the same dataset they were trained on

	Higher is better			Lower is better			
	δ_1<1.25	δ_2<1.25^2	δ_3<1.25^3	AbsRel	SqRell	RMSE	RMSE *log*
AdaBins							
Kitti	0.964	0.995	0.999	0.058	0.190	2.360	0.088
Unstructured	0.840	0.970	0.989	0.104	0.400	3.227	0.178
LapDepth							
Kitti	0.965	0.995	0.999	0.059	0.201	2.397	0.090
Unstructured	0.913	0.983	0.994	0.116	0.291	2.318	0.155
BTS							
Kitti	0.955	0.993	0.998	0.060	0.249	2.798	0.096
Unstructured	0.927	0.982	0.993	0.083	0.287	2.289	0.137

Table 5.11: Quantitative comparison of the depth networks when evaluated on the same dataset on which they were trained

As can be seen in Table 5.11, the metrics are brought into what looks like an acceptable range. This would suggest that the transfer learning as been effective in adapting the depth networks to unstructured environments.

Chapter 6

Conclusions

6.1 Conclusions

This project aimed to adapt current start of the art, monocular depth networks which have been trained in structured environments to work in unstructured environments. This was done by using transfer learning in order to train the network with the small dataset that was available for unstructured environments.

Chapter 3 analysed the architecture of the depth networks as well as the loss function that were used for training. It also described the various metrics that were used and how they were calculated.

As shown in Chapter 4, the depth networks perform well in structured environments but their performance drops of significantly when evaluated on unstructured environments. It was also shown in Chapter 5 that the metrics of the trained networks, when compared to the state of the art, were comparable. This shows that *transfer learning* has been an effective method of adapting the depth networks from the structured domain to the unstructured domain. It is noteworthy that BTS [24], which was the worst performing network of the three, had the best results once transfer learning was implemented.

Chapter 5 explained how the networks were going to be adapted as well as went over the results that were achieved, both qualitatively and quantitatively.

It has been shown that transfer learning is a viable method for adapting depth networks. Various ways of further progressing this idea have been put forward in the following section.

6.2 Future Work and Possible Improvements

Naturally, the first thing I would do given more time is to try and set up a computer to run AdaBins so that I can save the models after it has been trained. That would allow qualitative analysis to be performed.

It would be interesting to dive deeper into the structure of the networks and figure out why some adapted to the unstructured environment better than others e.g why did BTS adapt better than AdaBins even though it had worse performance on the KITTI dataset.

The ground truth data for the unstructured dataset is sparse. It would be interesting to see how the depth networks would adapt if the ground truth data was dense. It would also be nice to have larger datasets on which to train.

A deep dive into BTS and LapDepth in order to try find some relationship between the number of layers trained and the losses incurred or the accuracy and inlier metrics achieved.

Bibliography

[1] H. Yoo, J. Son, B. Ham, and K. Sohn, "Real-time rear obstacle detection using reliable disparity for driver assistance," *Expert Systems with Applications*, vol. 56, pp. 186–196, 2016.

[2] Y.-M. Han, J.-B. Jeong, and J.-H. Kim, "Quadtree based path planning for unmanned ground vehicle in unknown environments," in *2012 12th International Conference on Control, Automation and Systems*, 2012, pp. 992–997.

[3] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2650–2658.

[4] N. Smolyanskiy, A. Kamenev, and S. Birchfield, "On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1007–1015.

[5] P. Hambarde, A. Dudhane, P. W. Patil, S. Murala, and A. Dhall, "Depth estimation from single image and semantic prior," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 1441–1445.

[6] A. Saxena, S. H. Chung, A. Y. Ng *et al.*, "Learning depth from single monocular images," in *NIPS*, vol. 18, 2005, pp. 1–8.

[7] J. C. Read and B. G. Cumming, "Visual perception: Neural networks for stereopsis," *Current Biology*, vol. 27, no. 12, pp. R594–R596, 2017.

[8] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1, pp. 7–42, 2002.

[9] D.-J. Lee, P. Merrell, Z. Wei, and B. E. Nelson, "Two-frame structure from motion using optical flow probability distributions for unmanned air vehicle obstacle avoidance," *Machine Vision and Applications*, vol. 21, no. 3, pp. 229–240, 2010.

[10] Y. Salih, A. S. Malik, and Z. May, "Depth estimation using monocular cues from single image," in *2011 National Postgraduate Conference*, 2011, pp. 1–4.

[11] A. A. Fahmy, O. Ismail, and A. Al-Janabi, "Stereo vision based depth estimation algorithm in uncalibrated rectification," *Int J Video Image Process Netw Secur*, vol. 13, no. 2, pp. 1–8, 2013.

[12] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 66–75.

[13] B. Xu, S. Zhao, X. Sui, and C. Hua, "High-speed stereo matching algorithm for ultra-high resolution binocular image," in *2018 IEEE International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, 2018, pp. 87–90.

[14] e. Etriby, Sherif Said Aly, "3-d surface reconstruction using spatial frequency-based approaches under influence of perspective distortion,"

Ph.D. dissertation, Otto-von-Guericke-Universität Magdeburg, 2008. [Online]. Available: http://dx.doi.org/10.25673/4870

[15] A. Saxena, J. Schulte, A. Y. Ng *et al.*, "Depth estimation using monocular and stereo cues." in *IJCAI*, vol. 7, 2007, pp. 2197–2203.

[16] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, 2005, pp. 807–814 vol. 2.

[17] D. Scharstein and R. Szeliski, "The middlebury stereo pages," 2012.

[18] J. Zbontar, Y. LeCun *et al.*, "Stereo matching by training a convolutional neural network to compare image patches." *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2287–2318, 2016.

[19] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[20] R. Garg, V. K. Bg, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *European conference on computer vision*. Springer, 2016, pp. 740–756.

[21] N. Abd Manap and J. J. Soraghan, "Novel view synthesis based on depth map layers representation," in *2011 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, 2011, pp. 1–4.

[22] S. F. Bhat, I. Alhashim, and P. Wonka, "Adabins: Depth estimation using adaptive bins," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4009–4018.

[23] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2002–2011.

[24] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, "From big to small: Multi-scale local planar guidance for monocular depth estimation," *arXiv preprint arXiv:1907.10326*, 2019.

[25] L. Huynh, P. Nguyen-Ha, J. Matas, E. Rahtu, and J. Heikkilä, "Guiding monocular depth estimation using depth-attention volume," in *European Conference on Computer Vision*. Springer, 2020, pp. 581–597.

[26] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *arXiv preprint arXiv:1406.2283*, 2014.

[27] M. Song, S. Lim, and W. Kim, "Monocular depth estimation using laplacian pyramid-based depth residuals," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2021.

[28] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[29] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, "From big to small: Multi-scale local planar guidance for monocular depth estimation," *arXiv preprint arXiv:1907.10326*, 2019.

[30] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[31] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," *arXiv preprint arXiv:2103.13413*, 2021.

[32] J. Sivic and A. Zisserman, "Efficient visual search of videos cast as text retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 591–606, 2009.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[34] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1019–1034, 2015.

[35] D. Barnes, M. Gadd, P. Murcutt, P. Newman, and I. Posner, "The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Paris, 2020. [Online]. Available: https://arxiv.org/abs/1909.01300

[36] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017. [Online]. Available: http://dx.doi.org/10.1177/0278364916679498

[37] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[38] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[39] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," *arXiv preprint arXiv:1812.11941*, 2018.

[40] C.-H. Yeh, Y.-P. Huang, C.-Y. Lin, and C.-Y. Chang, "Transfer2depth: Dual attention network with transfer learning for monocular depth estimation," *IEEE Access*, vol. 8, pp. 86 081–86 090, 2020.

[41] H. Basak, S. Ghosal, M. Sarkar, M. Das, and S. Chattopadhyay, "Monocular depth estimation using encoder-decoder architecture and transfer learning from single rgb image," in *2020 IEEE 7th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, 2020, pp. 1–6.

[42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[43] J. Brownlee, "Encoder-decoder recurrent neural network models for neural machine translation," Aug 2019. [Online]. Available: https://machinelearningmastery.com/encoder-decoder-recurrent-neural-network-models-neural-machine-translation/

[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[46] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[47] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

[48] S. Ball and P. Amayo, "Depth networks in unstructured environments," 2020.

[49] C.-F. Wang, "The vanishing gradient problem," Jan 2019. [Online]. Available: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

[50] "Papers with code - resnext explained." [Online]. Available: https://paperswithcode.com/method/resnext

[51] Y. Li, W. Cai, Y. Gao, and X. Hu, "More than encoder: Introducing transformer decoder to upsample," *CoRR*, vol. abs/2106.10637, 2021. [Online]. Available: https://arxiv.org/abs/2106.10637

[52] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE transactions on acoustics, speech, and signal processing*, vol. 29, no. 6, pp. 1153–1160, 1981.

[53] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *CoRR*, vol. abs/1606.00915, 2016. [Online]. Available: http://arxiv.org/abs/1606.00915

[54] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," *CoRR*, vol. abs/1612.00603, 2016. [Online]. Available: http://arxiv.org/abs/1612.00603

[55] M. Shaw, "Writing good software engineering research papers," in *Software Engineering, 2003. Proceedings. 25th International Conference on*. IEEE, 2003, pp. 726–736.

[56] B. Paltridge, "Thesis and dissertation writing: an examination of published advice and actual practice," *English for Specific Purposes*, vol. 21, no. 2, pp. 125–143, 2002.

[57] U. Eco, *How to write a thesis*. MIT Press, 2015.

[58] I. Graessler, J. Hentze, and T. Bruckmann, "V-models for interdisciplinary systems engineering," in *DS 92: Proceedings of the DESIGN 2018 15th International Design Conference*, 2018, pp. 747–756.

[59] W. Royce, "The Software Lifecycle Model (Waterfall Model)," in *Proceedings of WESTCON*, Aug. 1070.

[60] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, May 1988.

[61] J. Taylor and J. G. Hoole, "Robust Protocol for Sending Synchronisation Pulse and RS-232 Communication over Single Low Quality Twisted Pair Cable," in *Proceeding of ICIT*. Taiwan: IEEE, Mar. 2016.

[62] T. Oetiker, H. Partl, I. Hyna, and E. Schlegl, "The Not So Short Introduction to LaTeX 2_ε," https://tobi.oetiker.ch/lshort/lshort.pdf, Jul. 2015, version 5.05.

[63] "IEEE Conference Paper Templates," http://www.ieee.org/conferences_events/conferences/publishing/templates.html.

[64] A. Baboon, B. Charles, D. Ester, and F. Generalson, "An Amazing Title," Their Not-so-awesome University, Technical Report, Apr. 1492.

[65] B. van der Zander, J. Sundermeyer, and T. Hoffmann, "TeXstudio – A LaTeX Editor," https://www.texstudio.org/.

[66] "InkScape Website," http://www.inkscape.org/.

[67] "Voluntary Page and Overlength Article Charges,"
http://www.ieee.org/advertisement/2012vpcopc.pdf, 2014.

[68] Y. Li, W. Cai, Y. Gao, and X. Hu, "More than encoder: Introducing transformer decoder to upsample," *CoRR*, vol. abs/2106.10637, 2021. [Online]. Available: https://arxiv.org/abs/2106.10637