Advanced Ranking Sampling Strategies for Monocular Depth Estimation

---

Praneeth Balakrishna

December 10, 2021
Version: My First Draft

PADERBORN UNIVERSITY
The University for the Information Society
Department of Electrical Engineering,
Computer Science and Mathematics
Warburger Straße 100
33098 Paderborn

INTELLIGENT SYSTEMS
Intelligent Systems Group (ISG)

Master's Thesis

Advanced Ranking Sampling Strategies for Monocular Depth Estimation

Praneeth Balakrishna

1. Reviewer Prof. Dr. Eyke Hüllermeier
Department of Computer Science
Paderborn University

2. Reviewer Prof. Dr. Marco Platzner
Department of Computer Science
Paderborn University

Supervisors Mr. Julian Lienen

December 10, 2021

Praneeth Balakrishna
Advanced Ranking Sampling Strategies for Monocular Depth Estimation
Master's Thesis, December 10, 2021
Reviewers: Prof. Dr. Eyke Hüllermeier and Prof. Dr. Marco Platzner
Supervisors: Mr. Julian Lienen
Intelligent Systems Group (ISG)
Paderborn University
Department of Computer Science
Pohlweg 51
33098 and Paderborn

Abstract Todo

Acknowledgement TODO

Contents

1 Introduction

> *You can't do better design with a computer, but you can speed up your work enormously.*
>
> — **Wim Crouwel**
> (Graphic designer and typographer)

1.1 Postcards: My Address

Praneeth Balakrishna
Vogeliusweg 23D.2.5
33100 Paderborn
Germany

1.2 Motivation

Depth estimation is a well studied problem in computer vision. It is an important step in the 3-D reconstruction of scenes and is crucial in fields like robot-vision and self driving cars. Monocular depth estimation aims to solve this problem using images obtained from a single camera. Humans can easily infer depth from single images with sufficient samples learnt(how near and far objects appear) over the lifetime. In contrast, for a computer vision system, monocular depth estimation is an ill posed problem due to its inherent ambiguity in mapping RGB pixel values to depth.

Typically, this is solved as a regression problem where the learning model learns to predict the metric depth map of the given RGB image. Such a depth map contains the metric depth of every pixel/segment within the image. Deep learning models, especially Convolutional Neural Networks(CNN) have achieved good success

in learning such models, as demonstrated in [EPF14]. Such a technique requires training data which also contains a metric depth map which is obtained using much complex techniques such as methods utilizing sensors like Lidar and Kinect.
However, not all applications require exact metric depth prediction. Instead, prediction of an ordinal relation is sufficient in such cases, *e.g.*, to detect occlusion boundaries in the image for an augmented reality application. This also provides an opportunity for models to learn from pseudo depth data [ZIKF15, CFYD16]. Pseudo depth maps only provide relative depth information such as object A 'is closer' than object B or object B 'is closer' than object C, but does not quantify the distance between them. Furthermore, such an ordinal classification problem can be solved as a ranking problem [Liu11]. [XSC+18, XZW+20] employ learning-to-rank methods on pairwise training samples, minimizing pairwise ranking loss in order to predict a dense pseudo-depth map. However, pairwise prediction leads to loss of information, particularly information about the transitivity of the order relation [LHEN21]. In order to overcome this issue, list-wise ranking was proposed [CQL+07], where an arbitrary number of samples are chosen as a list and the model is trained to rank the samples in the order of their relative depth [LHEN21]. This is achieved by minimizing a permutation loss function which computes the inconsistency between the output ranking predicted by the model and the ground truth permutation. Such a model aims to solve a much complex ranking problem wherein the transitivity of all elements in the list of an arbitrary length is to be considered. In the monocular depth estimation methods described in [XSC+18, CFYD16, LHEN21], sampling methods utilized for training the model is predominantly random sampling of pairs or lists. Random sampling leads to two potential issues. Firstly, it leads to imbalanced ordinal relations [XSC+18] and secondly, it misses out the fact that certain pixels in the image provide better depth cues than others.

In an attempt to further improve the performance of list-wise learning-to-rank method for monocular depth estimation [LHEN21], two potential methods are explored in this thesis. Firstly, an informativeness score is computed over the training data by making use of the visual depth cues in the images and the learning model is trained on highly informative samples that provide better cues for depth estimation. Secondly, a query strategy on a partially trained model is developed such that, the model queries samples from the unlabeled training data about which it is most uncertain with respect to relative depth prediction. Thereby, choosing to learn from training samples that it is most uncertain of. Such a technique lies in the realm of active learning [Set09] where the current model hypothesis is utilized within the query selection process for labelling samples for further training.

1.3 Problem Statement

The fundamental goal of the thesis is to solve the monocular depth estimation (of pixels) problem as a learning-to-rank task. However, the challenge is to achieve this by weakly supervised training data where the learning model does not learn from accurate metric depth, but instead learn from relative-depth training data. The methods described in Sec Todo !!!!! achieve laudable results in solving such a problem. The focus here is to improve those methods in order to achieve better results.

1.3.1 Improved sampling strategy

In methods [XSC+18, CFYD16, LHEN21], the learning models are trained with samples(pairs or lists) that are randomly sampled from the training images. There are certain aspects of the image, like depth-edges, texture, occlusion, etc. that provide depth cues and random sampling misses out on entirely utilizing such information and hence, the model learns mostly from less informative samples. Therefore, the aim is to propose an information measurement score on the training samples and develop a superior sampling strategy such that informative samples are more likely picked during training.

1.3.2 Active learning

Active learning is a case of machine learning where the learning algorithm chooses the data on which it learns [Set09]. The model poses queries on unlabeled data to an oracle(eg. a human annotator), for the label. By this method, the learning model can learn effectively on fewer number of training samples.
Therefore, the second aim of the thesis is to develop an active learning algorithm to query uncertain samples for labeling, during prediction. Additionally, an oracle which can label the queried samples is to be formulated.

Finally, an analysis of the performance improvement with both the above techniques is to be conducted in comparison to the earlier methods.

1.4 Results : TODO

1.5 Thesis Structure

Chapter 2

Chapter 2 gives an overview of most of the necessary topics and various terminologies that are needed to understand the chapters that follow. It begins with describing the fundamental concepts of computer vision and also traces the various methods in which features can be extracted from images. This is followed by the topic Deep Learning which has resulted in a revolution in the field of artificial intelligence. Popular deep learning techniques pertaining to monocular depth estimation are discussed here. A brief description of the programming tools and frameworks needed to realize such models is also provided.

Chapter 3

Chapter 4

Chapter 5

Chapter 5

Chapter 6

Fundamentals

2

> *A picture is worth a thousand words. An interface is worth a thousand pictures.*
>
> — **Ben Shneiderman**
> (Professor for Computer Science)

2.1 Computer Vision

Computer vision is a field of artificial intelligence which aims to develop techniques that aid machines/computers to see and make meaningful inferences out of the content present in digital images. Computer vision is the core idea in various applications namely Autonomous driving: for lane and pedestrian detection, in Healthcare: for medical imaging and cancer/tumour detection, in Manufacturing: for defect inspection, OCR/bar-code reading and preventive maintenance and in Agriculture: for crop yield monitoring, livestock health monitoring and crop disease detection.

Although the applications of computer vision is manifold, there exists a sequence of distinct steps to process and analyze images. Such a process can be termed as the vision pipeline. [Man]. The following figure 2.1 illustrates a typical vision pipeline followed in computer vision applications.

2.1.1 Input Data

Computer vision applications typically deal with image or video data. A simple grayscale image can be represented as a 2-Dimensional function $F(x, y)$ over x and y. where $F(x, y)$ is the intensity of the pixel at position (x, y)
In colour images, a third dimension is added which represents the intensities of

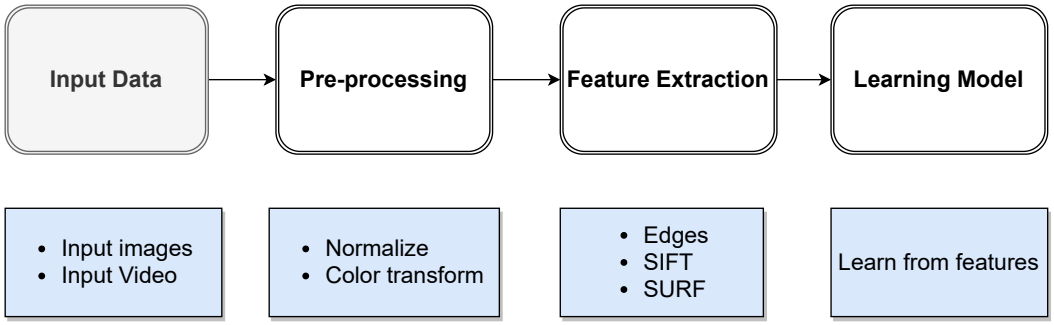

Fig. 2.1.: Computer vision pipeline

different colour channels in the image. A typical color model is the RGB color model which stands for "Red, Green and Blue". A colour image in RGB format would be:

$$\text{RGB image} = F(x, y) = [Red(x, y), Green(x, y), Blue(x, y)]$$

which consists of three channels $Red(x, y), Green(x, y), Blue(x, y)$ each of dimensions mxn. Each channel represents the intensity map of the particular colour at every position (x, y) in the image.
Each Color (that is a triplet of values (R, G, B)) has a depth of 24 bits (3 image planes times the number of bits per plane). This results in $(2)^{24}$ different colour [GW02].

2.1.2 Image Preprocessing

The images acquired usually come from various sources and are often not clean. There could be a varied amount of noise, illumination differences and different contrasts. In order to feed them to a machine learning application, they need to be standardized and cleaned up [Man]. It is nearly impossible to clean up each sample individually. Therefore a general procedure is followed to achieve this task.

1. **Convert images to grayscale to reduce complexity:** Certain applications like edge detection and depth estimation do not require information from all the three channels(in the case of RGB) in order to perform. In such cases it is useful to remove unwanted information from the image to reduce space and computational complexity [Man]

2. **Standardize images:** Algorithms involving learning models have a fixed input shape requirement. In order to satisfy this requirement, the images from

various sources must be scaled and reshaped to match the input dimensions of the learning models [Man] .

3. **Data augmentation:** Data augmentation is a method to increase the amount of training samples by making copies of the existing samples but however with slight changes in either orientation, translation, color intensity, etc. This exposes the learning model to a wider range of variations in the images and hence acts as a regularizer [Man].

2.1.3 Feature Extraction

Machine learning models perform only as good as the features that they are trained on. Therefore, identifying and quantifying good quality features is crucial. Feature extraction consists of two steps: Feature detection and feature description [GW02]. Feature detection refers to finding useful features in the image where as feature description assigns quantitative attributes to to the detected features [GW02]. Traditional computer vision problems relied heavily on manual feature engineering and selection. Such a task relied on the domain knowledge of the respective domain experts. The extracted features are then used to train learning models to realize a computer vision objective such as object detection, etc. Some examples of feature sets are:

- Histogram of Oriented Gradients
- Haar Cascades
- Scale Invariant Feature Transform (SIFT)
- Speeded Up Robust Feature (SURF)

A detailed description of machine learning models used in computer vision applications are provided in the next section.

2.2 Deep Learning

The inception of Machine learning had a primary objective of solving the problems faced by systems that rely on hard-coded information in order to function. Hence Machine learning emerged as a subset of Artificial Intelligence(AI), giving systems the ability to acquire their own knowledge by extracting information patterns from data without being explicitly programmed. However the performance of simple machine learning algorithms depends heavily on the representation of data that they are trained on [GBC16]. These representations of the raw data are called features and machine learning algorithms only learn to map such features to an output. For many tasks however, identifying what features are to be extracted is a difficult task by itself. For example, suppose we need a system to identify human faces in images. Here, one would like to use the presence of eyes as features. Unfortunately, it is very difficult to describe what eyes exactly look like in terms of image pixel values. Therefore, identifying and extracting features is a challenging task in most non trivial cases.

Deep Learning solves this central problem in representation learning by introducing representations that are expressed in terms of other, simpler representations [GBC16]. Deep learning technique uses abstract elements called neural networks to learn the representations of data and also models such high level abstractions in data using multiple non-linear transforms. Figure 2.2 demonstrates how a deep learning model can represent an image as multiple abstract feature representations like contours and object parts which are in turn represented as edges at a lower abstraction. In this manner, deep learning class of algorithms eliminate the need of manual feature engineering by employing techniques which inherently perform hierarchical feature extraction from the given data. It is based on cascaded layers of non-linear processing units, where the output of one layer is provided as an input to the next layer. The initial layers are responsible for extracting the lower level features from the data. This learned information is then fed to the next layer, which attempts to learn the higher-level features to form a hierarchical representation. These algorithms can operate in both supervised as well as the unsupervised manner and therefore can be used in applications like pattern recognition, classification tasks, etc.

The quintessential example of a deep learning model is the Multilayer Perceptron (MLP) [GBC16]. A multilayer perceptron is a mathematical function with certain non-linearity which maps a set of inputs to output values. Each application of such a

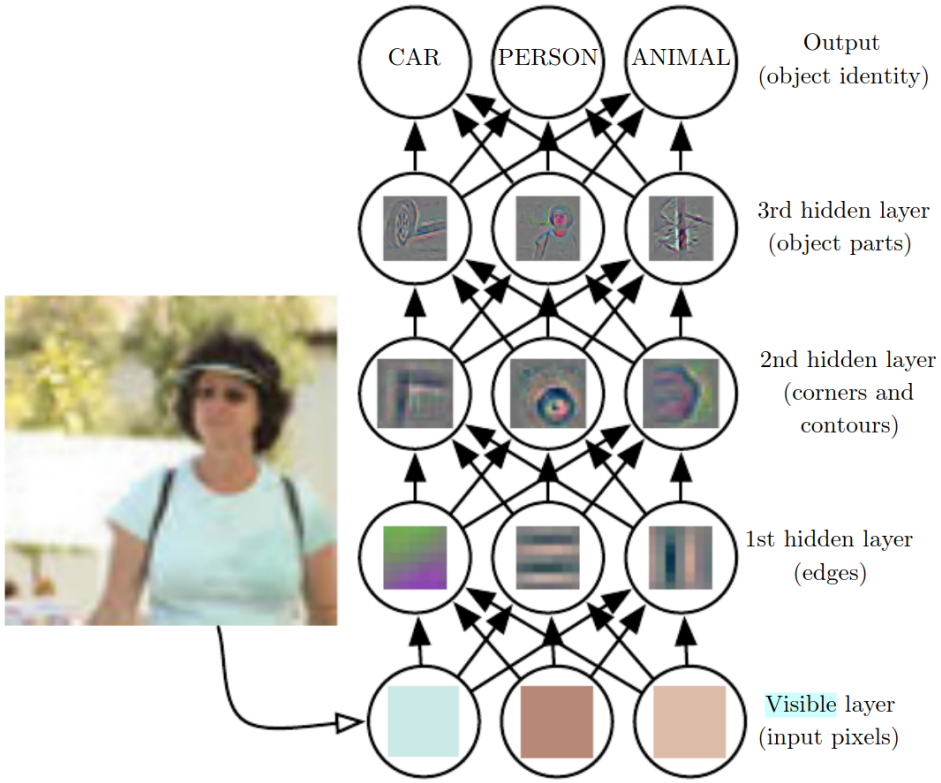

Fig. 2.2.: Illustration of deep learning model [GBC16]

function can be thought of as providing a new representation of the input data. Deep learning further breaks the complicated task of generating representations into series of nested mappings, each described by a different layer of the model [GBC16]. As shown in Figure 2.2, the input data consisting of observable variables are presented at the 'Visible layer'. This is followed by a series of 'hidden layers' which form abstract representations from the input. Each layer in the network is comprised of multiple nodes which perform the computations. Each node or neuron performs a linear combination of the set of inputs at its level and a set of weights and passes it through an activation function. The weights signify the relevance of the input component it acts upon. The sum-of-products of the input and weights are then passed through an activation function, which determines the output of the particular node. The activation function has an optional threshold value. If the sum-of-products is greater than the threshold, the node produces an output and is said to be *activated*. Figure 2.3 summarizes the operations in a single node/neuron. Mathematically such an operation for the output of the j'th node at the i'th layer can be describes as:

$$o_j = \varphi\left(\sum X_i \cdot W_{ij}, \Theta_j\right)$$

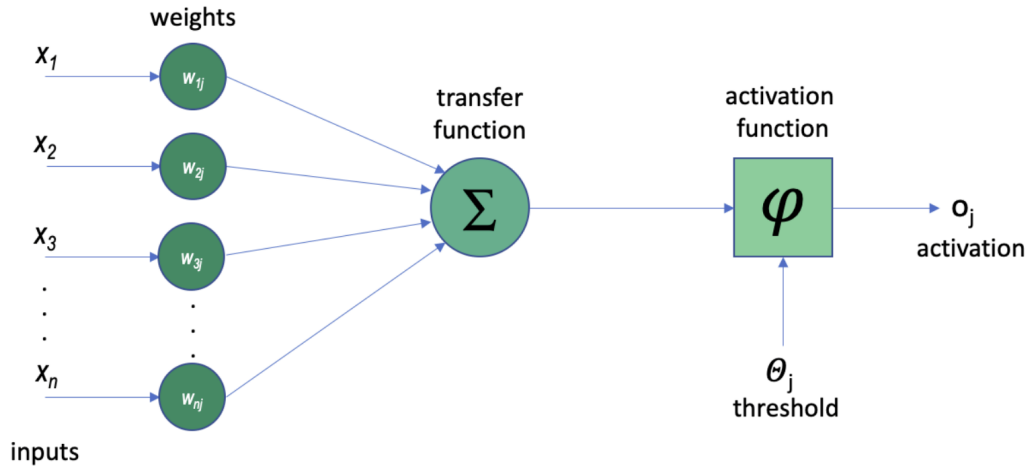

Fig. 2.3.: A single node of a neural network

Training a neural network essentially seeks to make the neural network learn the data representation behind the given raw input such that a predefined application specific cost function is minimized. This means that neural networks are trained using iterative, gradient-based optimizers that merely drives the cost function to a very low value [GBC16]. Variants of gradient descent and backpropagation are the crucial steps behind such iterative cost reduction technique used in deep learning. The derivative of the cost computed for a given prediction is computed with respect to each of the trainable parameters (weights) and subsequently updated in the direction of the negative gradient in an attempt to reduce the cost value. Using such an iterative optimization approach, neural networks are capable of learning any arbitrary function that maps the input to a suitable output.

In order to formally describe a deep neural network, it is essential to understand the following terms:

- **Number of layers:** The number of layers in a deep learning model determines the depth of the model. It is a model architecture choice and purely depends on the complexity of the task at hand. A higher number of layers result in a more complex model which might cause problems like over-fitting and vanishing gradients. On the other hand, having very less number of layers results in a very simple model which might not even be able to learn the underlying representations in the data. This phenomenon leads to a problem called under-fitting.

- **Cost Function:** The cost function is the criterion for optimization in all deep learning techniques. The cost function for deep neural networks is more or less similar to that of simpler parametric models such as linear models [GBC16]. In most cases the parametric model defines a distribution $p(y|x;\theta)$ where y is the prediction, x is the input and θ is the model parameter. In such cases the principle of maximum likelihood can be used and the negative likelihood can be used as the cost function. A classic example is the binary cross entropy loss used in binary classification tasks.

- **Activation function:** Popular activation functions used in deep learning are ReLU, sigmoid and tanh functions. In instances where a real number of any value is desired as an output, linear activation is used. The choice of activation function depends on the task and size of the network. However, sigmoid and tanh activation functions display the vanishing gradient problem for extreme values of their inputs. This factor must be considered while deciding for activation functions.

- **Optimizer:** Optimizers are algorithms to iteratively minimize the cost function by altering the model parameters in order to improve the accuracy of the model. Therefore the choice of an optimizer is crucial in order to converge faster and avoid unfavorable local minima. Optimizers are typically gradient based techniques. Popular choices include Adam, Stochastic Gradient Descent(SGD) and RMSProp.

- **Learning rate:** The learning rate determines the size of the step taken in the direction of the optima in optimization algorithms. The choice of the learning rate determines the speed of convergence and the final optimum reached. A very low learning rate may make the training process very slow and hence takes a long time to converge. At the same time, a higher learning rate may cause the optimizer to overshoot the desired minima. Therefore, a balanced choice is to keep it adaptive, such that a higher learning rate is used in the beginning and subsequently decays over the training process.

- **Regularization:** Deep learning models are complex models with even millions of parameters in certain cases. This gives the model an enormous amount of flexibility to fit the training data. At times this can lead to overfitting to the training data. Hence regularization is essential to produce better generalization of the model. In L1 and L2 regularization, an additional regularization term is added to the cost function that penalizes large weights. However, Dropout

regularization is the most popular technique for deep learning models. In dropout, at every training step each node has a probability p of being turned off. The network learns accordingly, adapting to this change, thus reducing the chances of overfitting.

- **Epochs:** An epoch signifies one complete pass of the training data during the learning process. If the number of epochs is too large, it may cause the model to overfit to the data and may result in poor generalization on the test data. Whereas, if the number of epochs is too small, the model may not be able to learn the representations completely and may result in under-fitting.

2.3 Software Tools

This section briefly describes some of the most necessary software tools and frameworks used in realizing a deep learning based monocular depth estimation system for the thesis.

2.3.1 Python programming language

Python is the world's fastest growing and popular programming language not just among software engineers and also among mathematicians, data analysts, scientists and accountants. Surveys like [bus] [Row18] done worldwide also indicate that python is the most commonly used programming language for machine learning and data analysis. People from different disciplines use python for variety of different tasks such as data analysis and visualization, machine learning models and other automation.

Python, which was originally developed in the late 1980's is an object oriented, high-level general purpose programming language [VR+07]. It provides a wide array of built-in data structures and abstract data types which eases application development. One of the distinguishing factors of python is that it is an interpreted language, meaning it does not have a compilation step. This feature along with modularity and simple syntax makes debugging simpler, thus reducing development time through increased efficiency.

Artificial intelligence and machine learning has penetrated into most applications in today's world. Python has become the go-to language for machine learning and deep learning applications. Python comes with a myriad amount of libraries and frameworks that make coding easy. This also saves a significant amount of time. Python is an open-source programming language and enjoys excellent support from many resources and quality documentation along with a large and active community. Where speed is a concern, it provides alternatives like Cython, which is capable of producing performances similar to the C programming language. Also, being popular for the data analysis related tasks, Python provides very efficient interfaces to all major commercial as well as open-source databases. All these features allow Python to outshine in the field of data science compared to traditional languages like C++ and Java.

2.3.2 Machine Learning Frameworks: TensorFlow

TensorFlow [AAB+16] is a computation library developed by Google which has become very popular among the machine learning community. Though it was originally developed for numerical computation, it proves to be useful for deep learning applications too. The fundamental data structure used in TensorFlow are multidimensional arrays called tensors. Each computation is represented graphically as a data flow graph [Ten19] where each node in the graph represents a mathematical operation.

Some of the key features of tensor-flow are as follows:

- Though python is the preferred language for developing applications in TensorFlow, most of the underlying implementation is done in C++ and CUDA.
- TensorFlow provides comprehensive GPU support. This is crucial for training deep learning models faster and also facilitates training on distributed clusters.
- In contrast to sequential programming, a static computation graph consisting of nodes and edges is created and the graphs are executed in the form of sessions.
- *tf.data* facilitates building of data pipelines efficiently including data preprocessing.

- Keras integrates tightly with TensorFlow [Ten19]. This offers a variety of high-level API's which can be used to build complex models on top of low-level TensorFlow functions.

todo : Additional buffer for more stuff if needed.

Related Work

3

> *A picture is worth a thousand words. An interface is worth a thousand pictures.*
>
> — **Ben Shneiderman**
> (Professor for Computer Science)

3.1 Related Work Section 1

3.2 Related Work Section 2

3.3 Related Work Section 3

3.4 Conclusion

4 System

> *Innovation distinguishes between a leader and a follower.*
>
> — **Steve Jobs**
> (CEO Apple Inc.)

4.1 System Section 1

Fig. 4.1.: Figure example: *(a)* example part one, *(c)* example part two; *(c)* example part three

4.2 System Section 2

Fig. 4.2.: Another Figure example: *(a)* example part one, *(c)* example part two; *(c)* example part three

4.3 System Section 3

4.4 Conclusion

5 Concepts: This text is here to test a very long title, to simulate the line break behavior, to show that an extremely long tilte also works

> *Users do not care about what is inside the box, as long as the box does what they need done.*
>
> — **Jef Raskin**
> about Human Computer Interfaces

5.1 Concepts Section 1

5.2 Concepts Section 2

5.3 Concepts Section 3

5.4 Conclusion

6 Conclusion

6.1 System Section 1

6.2 System Section 2

6.3 Future Work

Example Appendix

A

A.1 Appendix Section 1

Alpha	Beta	Gamma
0	1	2
3	4	5

Tab. A.1.: This is a caption text.

A.2 Appendix Section 2

Alpha	Beta	Gamma
0	1	2
3	4	5

Tab. A.2.: This is a caption text.

Bibliography

[AAB+16] ABADI, Martín; AGARWAL, Ashish; BARHAM, Paul; BREVDO, Eugene; CHEN, Zhifeng; CITRO, Craig; CORRADO, Greg S.; DAVIS, Andy; DEAN, Jeffrey; DEVIN, Matthieu et al.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. In: *arXiv preprint arXiv:1603.04467* (2016)

[bus] *Programming Languages Most Used and Recommended by Data Scientists*

[CFYD16] CHEN, Weifeng; FU, Zhao; YANG, Dawei; DENG, Jia: Single-image depth perception in the wild. In: *arXiv preprint arXiv:1604.03901* (2016)

[CQL+07] CAO, Zhe; QIN, Tao; LIU, Tie-Yan; TSAI, Ming-Feng; LI, Hang: Learning to rank: from pairwise approach to listwise approach. In: *Proceedings of the 24th International Conference on Machine learning*, 2007, S. 129–136

[EPF14] EIGEN, David; PUHRSCH, Christian; FERGUS, Rob: Depth map prediction from a single image using a multi-scale deep network. In: *arXiv preprint arXiv:1406.2283* (2014)

[GBC16] GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron: *Deep Learning*. MIT Press, 2016. – `http://www.deeplearningbook.org`

[GW02] GONZALEZ, R.C.; WOODS, R.E.: *Digital Image Processing*. Prentice Hall, 2002. – ISBN 9780201180756

[LHEN21] LIENEN, Julian; HULLERMEIER, Eyke; EWERTH, Ralph; NOMMENSEN, Nils: Monocular Depth Estimation via Listwise Ranking Using the Plackett-Luce Model. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, S. 14595–14604

[Liu11] LIU, Tie-Yan: Learning to rank for information retrieval. (2011)

[Man] MANNING. *Free Content Center*

[Row18] ROWE, Martin. *Survey says python is top programming language*. Aug 2018

[Set09] SETTLES, Burr: Active learning literature survey. (2009)

[Ten19] TENSORFLOW. *What's coming in TensorFlow 2.0*. Jan 2019

[VR+07] VAN ROSSUM, Guido et al.: Python Programming Language. In: *USENIX annual technical conference* Bd. 41, 2007, S. 36

[XSC+18] XIAN, Ke; SHEN, Chunhua; CAO, Zhiguo; LU, Hao; XIAO, Yang; LI, Ruibo; LUO, Zhenbo: Monocular relative depth perception with web stereo data supervision. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, S. 311–320

[XZW+20] XIAN, Ke; ZHANG, Jianming; WANG, Oliver; MAI, Long; LIN, Zhe; CAO, Zhiguo: Structure-guided ranking loss for single image depth prediction. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, S. 611–620

[ZIKF15] ZORAN, Daniel; ISOLA, Phillip; KRISHNAN, Dilip; FREEMAN, William T.: Learning ordinal relationships for mid-level vision. In: *Proceedings of the IEEE International Conference on Computer Vision*, 2015, S. 388–396

List of Figures

List of Tables

Colophon

This thesis was typeset with LaTeX 2_ε. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

Declaration

You can put your declaration here, to declare that you have completed your work solely and only with the help of the references you mentioned.

Paderborn, December 10, 2021

Praneeth Balakrishna