Deep Learning for Image and Video Processing

Academic Year: 2023-2024

Single Image Depth Estimation

Final project

Ismael Gomez Garrido i6334270

Table of Contents

1. Introduction

Single Image depth estimation or monocular depth estimation is a known task in the field of computer vision that consists of estimating the depth value for every pixel of a given RGB image. The image below is a single image depth estimation example by DINOv2, a model from Meta AI [1].

This problem is not trivial and presents many difficulties, which can be reduced to a big variance in the structure, composition, and perspective of the input images. Just to give an idea of how the humans do this complex task here are seven indications or signals that we use for depth estimation [2]:

- Occlusion: Objects that are partially covered by other objects are considered farther away.
- Perspective: Parallel lines that appear to meet in the distance provide information about depth.
- Size: The size of an object on the retinal image varies with its distance, influencing depth perception.
- Texture Gradient: The texture of a surface appears denser as the distance from the observer increases.
- Atmospheric: Objects farther away may appear blurry and bluish due to atmospheric effects.
- Light and Shadows: Patterns of light and shadows, such as objects casting shadows or having shadows attached to their surfaces, are used to perceive depth.
- Height: Objects closer to the horizon seem farther away, contributing to depth perception.

With these indicators and all our background knowledge this task is very easy for every human but becomes very challenging to perform it successfully with a computer vision algorithm. Depth estimation algorithms find application in various fields, playing a crucial role in computer vision tasks and enhancing the capabilities of machines to perceive and understand their surroundings. In autonomous vehicles, these algorithms contribute to scene understanding, enabling the vehicle to navigate safely by recognizing the distance to objects and obstacles. In the field of robotics, depth estimation facilitates object manipulation and interaction by providing insights into the spatial distribution of the environment. Other applications of this technique can be found in other fields such as Augmented Reality, Virtual Reality, or medical imaging.

This project focuses on single image depth estimation in the wild, this means that the images do not have a similar structure or common elements. The great variance between the training images makes this problem very complicated because it demands a high generalization ability.

In the next section some related works are exposed, and then the data selected for this project and the methodology, along with the experiments carried out and the final conclusions.

2. Related work

Due to the complexity of the task and the number of applications in the real world in the past years there have been many contributions to this field. In this section I will try to expose a general view of how these implementations have evolved and how is the current state of the art with respect to this task.

At the early stages of machine learning one of the main approaches was to group sets of pixels into superpixels and estimate the relative depth between these superpixels [3]. An issue of this idea is the difference between indoor and outdoor images, due to their different structure the performance can be easily affected. Therefore, it is common to separate them and train the model only with one of these kinds.

Until the recent use of vision transformers, the encoder-decoder neural network was the most common architecture used for this problem. This bottle-neck structure forces the model to learn only the most relevant features of the images. One example of this implementation is [4], where the network is built using inception modules proposed by Google [5] to predict the image depth.

Other work based of an encoder-decoder idea is [6]. The authors first use a common encoder (they test several pretrained ones) and then they implement a multi-level decoder using Laplacian pyramids to decode the image and get the depth. They obtained state-of-the-art results on benchmarks datasets containing both indoor and outdoor images.

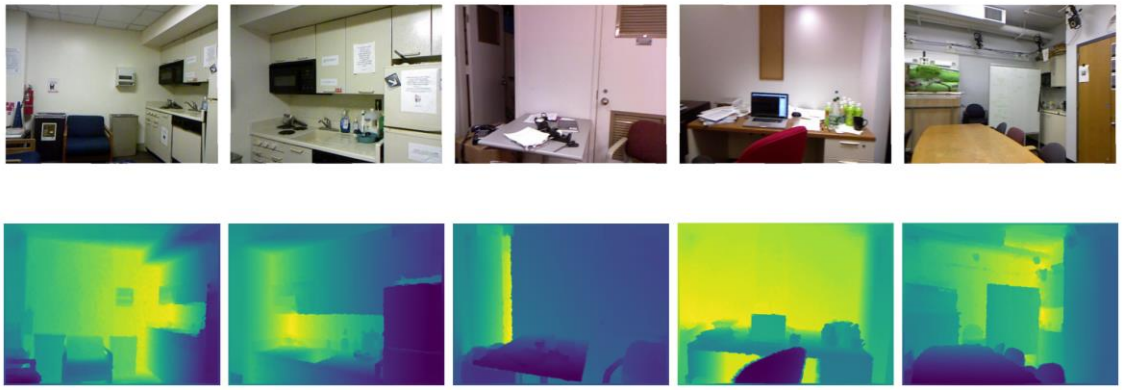
The most recent works in this field involve more complicated models. On the one hand, vision transformers have become popular, and, in many benchmarks, they have obtained very promising results [7]–[9]. On the other hand, diffusion-based models also have been used for depth estimation achieving considerable success as well [10].

3. Data

NYU Depth V2

For this project I have used two different datasets. The first one is NYU Depth V2, it consists of video sequences from a variety of indoor scenes as recorded by both the RGB and Depth cameras from the Microsoft Kinect [11]. The dataset contains 1449 RGB images with resolution 480 x 640 and its corresponding depth maps.

NYU Images and Depths

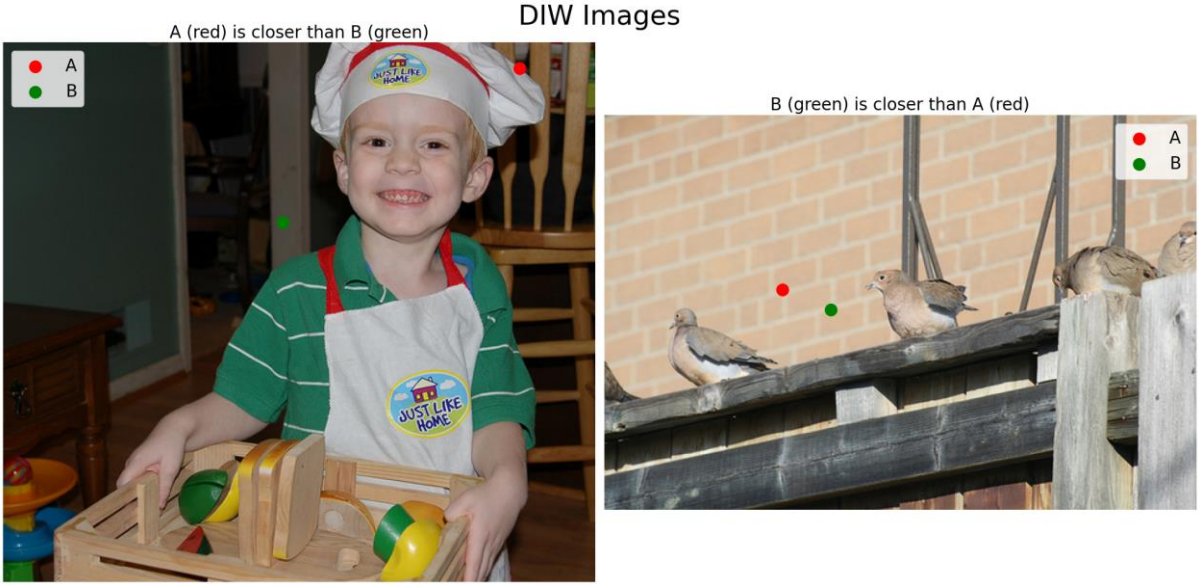


The main reason to choose this dataset is that it has been widely used as a benchmark for single image depth estimation in the last years. Therefore, there are many publications using this dataset to train models or test performance. Besides, the small number of images makes the training faster.

Depth in the Wild

The Depth in the Wild (DIW) dataset is a dataset created from random images sampled using random query keywords in Flickr [4]. Instead of having the full depth map of the images, the annotation consists of the depth relation between two points in the image (which point is closer or if they are equal). The authors remark that the points are not completely randomly chosen in order to avoid bias like assuming the lower point is closer or, if both have the same height, the point closer to the center will be also the closest in depth. This results in a total of 500.000 RGB images with variable resolutions, which range between 250 x 200 and 500 x 500.

DIW Images



The reason to choose this dataset is the challenge that it represents learning to estimate the depth of an image using only the relative depth information from the pairs of points. In [4] the authors obtained solid results training only with DIW, but they obtained their best performance by training first on NYU Depth V2 and then using DIW.

4. Methodology

Data preprocessing

Before the implementation of any model, it was necessary to analyze and preprocess the data. In order to make the training faster all the images from both datasets were resized to a resolution of 200 x 200, it is an efficient size and still captures the structure and details from the original image.

Due to the huge number of images in the DIW and the limited available computational power I selected a subset of this dataset. The selected subset corresponds to the original test partition from the repository, and it contains about 74000 images.

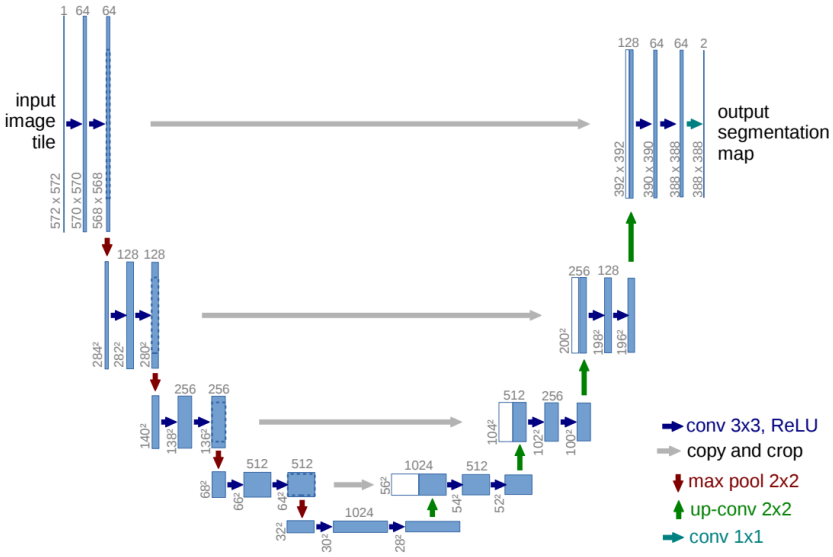
Training process

Learning directly from the DIW dataset would require a lot of iterations because every training example only focuses on 2 points from the predicted depth image. To make that faster the models will be trained first on the NYU dataset and then on the DIW. In this way, when the model receives the input from the pairs of points it will have some previous background knowledge of estimating depth and will be easier to adapt the output.

To increase the performance of the model and reduce the training iterations I decided to use pretrained models as a basis for the architectures. The majority of the available pretrained models for depth estimation were too big for my computational resources, so the models selected are not specifically trained for depth estimation. However, the models have been trained in computer vision tasks and possess some background knowledge in image analysis and feature extraction. Several pretrained models have been used in this project but here I present only the 2 which offered the best performance while keeping a reasonable training and inference time. The discarded models are Google InceptionV3, MiDaS and GPLN[6], [7]

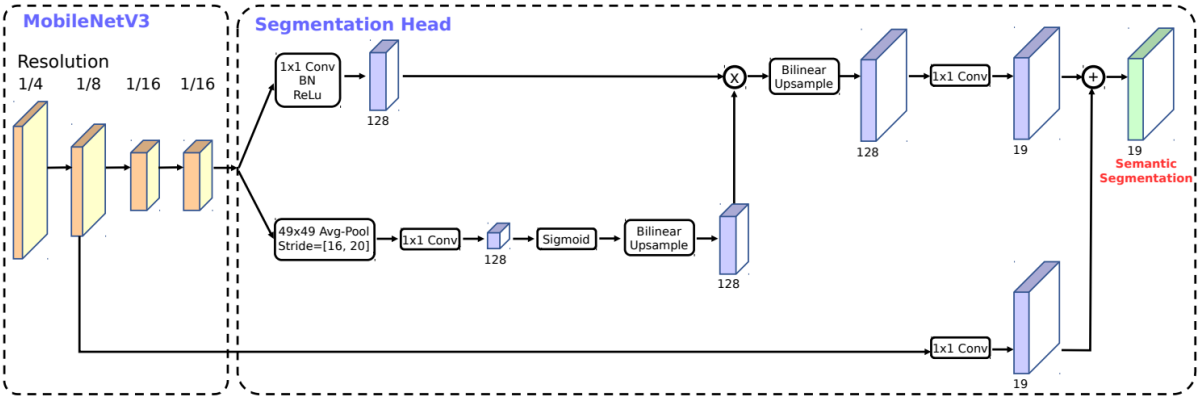
Model I: U-Net ++

The U-Net is a well-tested and widely used architecture type in computer vision, specially in semantic segmentation [12]. Image semantic segmentation is well related to depth estimation because the detection of the different classes in an image can help in the understanding of the spatial distribution. As it can be seen in the image below, the structure of the network is similar to an encoder-decoder but with connections between the intermediate levels. The chosen encoder pretrained weights were '***resnext50_32x4d'*** and the total number of parameters of the model is 48M.



Model II: Transfer Learning with LR-ASPP

Another image semantic segmentation and object detection model is LR-ASPP (Little Reduced Atrous Spatial Pyramid Pooling). This model uses a MobileNetV3 encoder and a new architecture LR-ASPP as a segmentation head for the decoder. The Atrous Spatial Pyramid Pooling module resamples a given feature map at multiple rates prior to convolution, helping to capture objects at different scales thanks to the Atrous convolutions. The Little Reduced idea refers to using a large pooling kernel with large stride and 1x1 convolution, this helps to reduce the number of weights in the decoder [13], [14].



Metrics

NYU Depth V2 Loss function

The loss function chosen to evaluate and train the models with the NYU data is the Mean Squared Error. Both depth maps and model predictions are normalized so their range is [0,1] and then the loss is calculated by comparing the images pixelwise.

DIW Loss function

To train the model based only on the depth relation of two points a new loss function was developed in [4]. This function encourages the model to predict a higher value for the closest point:

$$loss(I, a, b, z) = \begin{cases} \log(1 + \exp(-z_a + z_b)) & if\ r = +1 \\ \log(1 + \exp(z_a - z_b)) & if\ r = -1 \\ (z_a - z_b)^2 & if\ r = 0 \end{cases}$$

Where ***I*** is the training image, ***a*** and ***b*** are the two points and z is the predicted depth image. When ***r*** = 1 it means that the point ***a*** is closer than ***b*** and when ***r*** = -1 it means that ***b*** is closer than ***a***. In the case of both points being at the same or very similar distance we have ***r*** = 0. For simplicity, in this project we have considered that every image is associated with only one pair of points, and we have assumed that always one point is closer than the other.

WKDR - Accuracy

Another used metric to quantify the performance of the model in the DIW dataset is the Weighted Kinect Disagreement Rate (WKDR), that is the disagreement between the ground truth and the model predictions. This metric depends on the parameter delta, which determines a threshold to consider when two points are at the same depth [3]. In the subset chosen in this project there is no pair or points with the equal relationship, hence, for measuring the performance of the model the accuracy of the predictions (rate between the number of correct predictions and the total number of predictions) will be used.

5. Experiments & Results

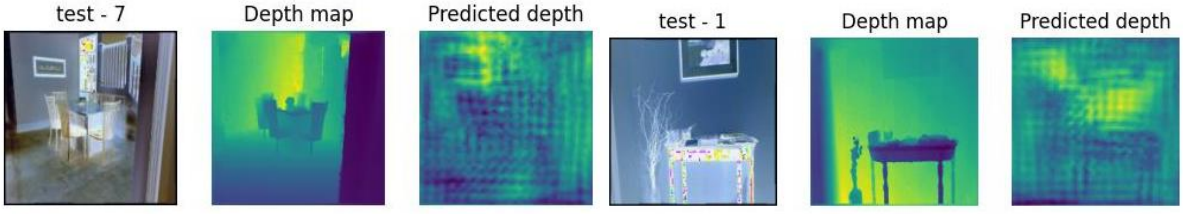
LR-ASPP

The hyperparameters for this experiment were:

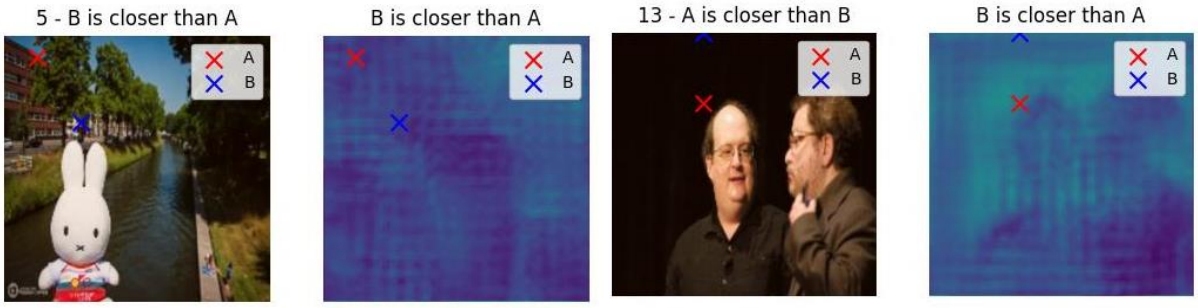
Item	Value
model weights	LRASPP_MobileNet_V3_Large_Weights
Loss	Mean Squared Error
Optimizer	Adam
Weight decay	0.02
Learning rate	1e-3
Epochs	100
Parameters	3.2 M

The output of the pretrained model had 21 layers, I performed some experiments on how to add fine-tuning layers to the model in order to have only one layer as an output. Finally the best result was obtained by just combining all the 21 layers in 1 step (other alternatives were adding more convolutional filters or a bottle-neck structure).

The results after training with the NYU dataset are shown below. The main depth structure is hardly shown and it's only able to recognize some shapes.



After training with the DIW dataset:



U-Net++

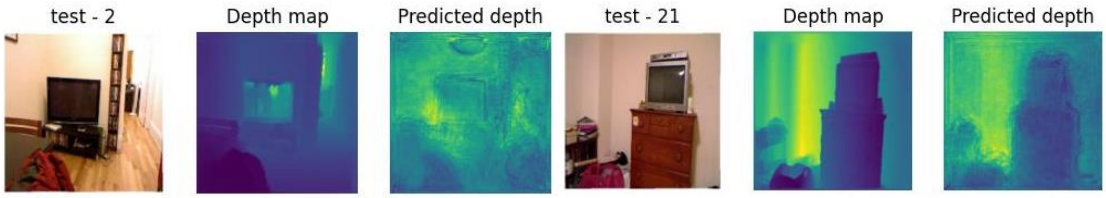
The hyperparameters for this experiment were:

Item	Value
model weights	resnext50_32x4d
Loss	Mean Squared Error
Optimizer	Adam
Weight decay	0.02
Learning rate	1e-3
Decoder epochs	2
Full epochs	8
Parameters	48 M

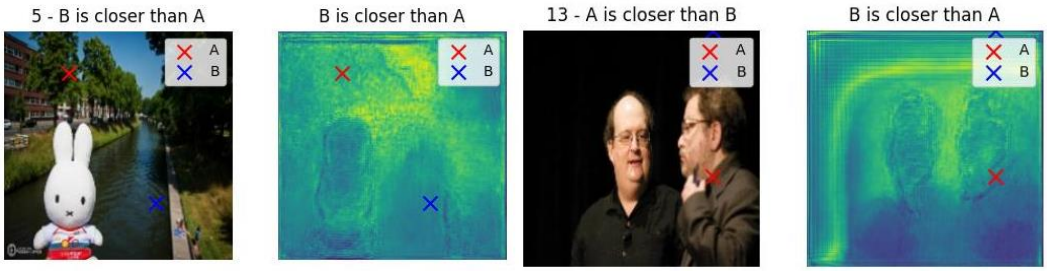
The reasoning behind the decoder epochs and the full epochs is that in the first two training epochs the encoder weights were frozen. This freezing allows the random initialized weights from the decoder to learn without harming or disturbing the encoder weights. After these 2 epochs, all the model weights are trained.

Due to the big difference in the number of parameters with respect to the LR-ASPP, this model was trained for less epochs. However, the training time was notably larger with this model.

After training the U-Net on the NYU dataset, the following predictions were made on the test data:



Then, after training the model with the DIW dataset the predictions were like this:



In both datasets the main structure of the image and the depth distribution is well captured, although in the DIW examples the shapes and details are less defined. This can be caused because the model was training less epochs on the DIW dataset due to the big number of instances.

Comparation

In the table below there is a comparison between the performance of both models. The U-Net clearly outperformed the LR-ASPP model in both datasets, as was demonstrated with the depth images examples.

	LR-ASPP	U-Net++
NYU - Test MSE loss	0.058	0.032
DIW - Test accuracy	58.4	65.32

6. Conclusions & possible improvements

I think that the big difference between the number of parameters had a lot of influence on the performance in both datasets. Although the LR-ASPP is a more innovative and effective architecture with that relatively small size is not possible to capture all the information that the U-Net is able to retain.

The difference between the performance in BYU and DIW can be explained by two factors. On the one hand, the NYU dataset is providing on every training step feedback over all the predicted pixels while when training with the DIW the error is only calculated based on two points. Therefore, the learning in this second data set is much slower. On the other hand, the difference in the number of instances made it impossible to train on both datasets the same number of epochs, so the images from NYU were seen more times by the models than the ones from DIW. Besides, the big variance present in the DIW images makes it harder to learn the depth structure of the image.

Possible lines of work for this project are comparing models with a similar number of parameters or trying other different architectures like vision transformers or diffusion models.

7. References

[1] M. Oquab *et al.*, “DINOv2: Learning Robust Visual Features without Supervision,” Apr. 2023, [Online]. Available: http://arxiv.org/abs/2304.07193

[2] A. Mertan, D. J. Duff, and G. Unal, “Single Image Depth Estimation: An Overview,” Apr. 2021, doi: 10.1016/j.dsp.2022.103441.

[3] D. Zoran CSAIL, P. Isola CSAIL, D. Krishnan Google, and W. T. Freeman Google, “Learning Ordinal Relationships for Mid-Level Vision.”

[4] W. Chen, Z. Fu, D. Yang, and J. Deng, “Single-Image Depth Perception in the Wild,” Apr. 2016, [Online]. Available: http://arxiv.org/abs/1604.03901

[5] C. Szegedy *et al.*, “Going Deeper with Convolutions,” Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.4842

[6] 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*, vol. 31, no. 11, pp. 4381–4393, Nov. 2021, doi: 10.1109/TCSVT.2021.3049869.

[7] R. Birkl, D. Wofk, and M. Müller, “MiDaS v3.1 -- A Model Zoo for Robust Monocular Relative Depth Estimation,” Jul. 2023, [Online]. Available: http://arxiv.org/abs/2307.14460

[8] X. Yang *et al.*, “PolyMaX: General Dense Prediction with Mask Transformer,” Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.05770

[9] R. Ranftl, A. Bochkovskiy, and V. Koltun, “Vision Transformers for Dense Prediction,” Mar. 2021, [Online]. Available: http://arxiv.org/abs/2103.13413

[10] B. Ke, A. Obukhov, S. Huang, N. Metzger, R. C. Daudt, and K. Schindler, “Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation,” Dec. 2023, [Online]. Available: http://arxiv.org/abs/2312.02145

[11] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, “Indoor Segmentation and Support Inference from RGBD Images.”

[12] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, “UNet++: A Nested U-Net Architecture for Medical Image Segmentation,” Jul. 2018, [Online]. Available: http://arxiv.org/abs/1807.10165

[13] A. Howard *et al.*, “Searching for MobileNetV3,” May 2019, [Online]. Available: http://arxiv.org/abs/1905.02244

[14] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” Jan. 2018, [Online]. Available: http://arxiv.org/abs/1801.04381