Depth Estimation Project: Monocular Depth Estimation trained on a given data sets

Yerdaulet Kumisbek
Nazarbayev University
yerdaulet.kumisbek@nu.edu.kz

Abzal Aidakhmetov
Nazarbayev University
abzal.aidakhmetov@nu.edu.kz

Merey Zhumayeva
Nazarbayev University
Merey.Zhumayeva@nu.edu.kz

December 6, 2022

Abstract

The calculation of depth is crucial for many artificial intelligence tasks, including mapping, localization, and obstacle avoidance. As a result of some features of monocular cameras, namely, comparatively cheap cost and small size, depth estimation from a single RGB picture has garnered substantial and rising attention. Modern single-view depth estimate techniques, however, are too sluggish for real-time inference on an embedded platform, such as one installed on a micro aerial vehicle, since they are built on very large deep neural networks. In our project work, considering several depth estimation papers, we came to the one conclusion of using a MobileNet pre-trained model for the encoder and upsampling methods for the decoder, making them relatable to each other. So, using the right network architecture, we could further minimize computational complexity and latency. By reaching almost the same loss (0.0022) and MSE (148) as complex analog research methods, we could reach runtime for only 0.08 seconds, meaning a high rate that can be used in embedded systems.

1 Introduction

Numerous artificial intelligence activities, such as mapping, localization, and obstacle avoidance, depend on how depth is calculated. Existing devices and sensors for depth estimation are usually hard to install, heavy, and highly power-consuming ones (e.g., Lidar, structured-light sensors, etc.). Because of such drawbacks, they are not suitable as wanted for small robotic platforms (such as micro aerial and mini ground vehicles). Concerning this problem, monocular camera usage for depth calculation is established, as it usually inexpensive, small and energy-efficient.

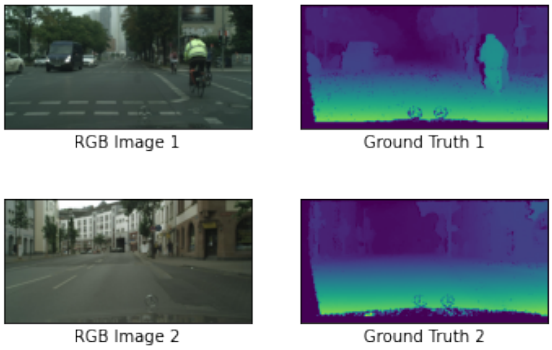

Figure 1: RGB Image and ground truth

Due to this day, the majority of previous research on enhancing the reliability and accuracy of monocular depth estimation has led to computationally costly techniques that are difficult to integrate into robotic systems. So, even though research papers show Monocular Depth utilities' usefulness, this topic is still problematic with weak constraints. It requires the use of several, occasionally subtle vi-

sual signals, extensive context, and prior information in order to be solved. So, it leads us to focus on computer vision's main object - learning-based techniques. Diverse training data is required in order to develop models and trustworthy vision techniques that can be applied to a variety of inputs. Even while gathering data on a large enough scale with variables like light intensity and objects in photos might be difficult, there are a number of available data sets online that we were able to utilize for this model.

Regarding the recent research papers, many of them address the increasing efficiency of deep learning-based methods at the cost of increased computational complexity perspective. Also, as the FastDepth paper shows, recent papers exceptionally focus on creating fast and effective encoder networks, that are used for identification and picture classification. So, in regarding applications, pixel-based pictures are used as input, whereas the output is a label, which is an object type and location.

Considering such limitations, in this paper, we have used the "Transfer learning and fine-tuning" Tensorflow technique, where Transfer learning means taking the learned features from one dataset and utilizing them in another data set. So, this process consists of several stages: 1) Taking layers from previous datasets features and trained model, 2) Save them, or freezing them, to not destroy data while using, 3) adding new trainable layers to a frozen one, and finally 4) use it for a new dataset. After transfer learning occurs, fine-tuning, which is an optional step has proceeded, where after unfreezing the previous model occurs, it is applied to new types of datasets, making their loss smaller and efficiency higher.

Addressing the Transfer learning and fine-tuning process, MobileNet encoder-decoder network architecture, where low latency is in focus was taken. [1] Also, for a network pruning process, NetAdapt is considered the best one [1], with the usage of the TVM compiler stack to reduce the runtime that is crucial for embedded systems using monocular depth estimation devices. During this project work, the low-latency designed model shows 0.009 percent loss, meaning one of the best score for the monocular depth calculation program.

In the architecture of the module, notably, Resizing, rescaling, and flipping methods are used to make input images fittable to the filtering process. Also, the "weights" of the initial pre-trained module are saved and not changed initially, to not affect it while training new input data sets. After the training process occurred, new "weights" of the new input data sets, namely, street images, and industrial images' features.

In conclusion, this study shows monocular depth estimation trained on given datasets and used a pre-trained model on MobileNet network architecture, taking an output of a very low percentage of loss.

2 Related Work

2.1 Literature review

Over the course of the past few years, a wide variety of approaches have been applied to the problem of monocular depth estimation. Traditional methods that use multiple cameras or sensor-based systems such as LIDAR do not lose their relevance today, but these methods have been improved with specialized hardware which helps provide an accurate 3D point cloud. In spite of the widespread adoption of these techniques, monocular depth estimation—the process of constructing a depth map of a scene using only a single image as input—remains a very active field of research.

Deep convolutional neural networks (CNN) have been trained in recent years by researchers using currently accessible datasets such as KITTI [3], Make3D [10], MegaDepth [9], NYU Depth V2 [11], and Cityscapes[2]. For example, the research on MegaDepth aimed to train a highly accurate monocular depth estimation CNN and also curate a 200GB depth dataset of camera shots. Other successful methods such as Monodepth [4] and MonodepthV2 [5] are of notable importance because they use self-supervised learning to exploit the left-right consistency constraint of stereo images. The researchers introduced a method for developing extremely accurate depth maps from largely unlabeled (or partially labeled) datasets. They trained the network with binocular stereo images. The results showed that the

Unlabeled data can make it feasible to gather and use vast amounts of data that can be used for network training.

Transfer learning is another principal method to perform efficient pre-trained encoder [1]. For instance, the research paper on DenseNet [8] demonstrated that with a sufficiently precise and pre-trained encoder, even a simple decoder architecture can produce good depth estimation results. Using transfer learning on a pre-trained dataset, this research reduces the need for extremely large labeled datasets. In addition, the most widely used architectures of transfer learning in the computer vision field are the ResNet [6], MobileNet [7] and EfficientNet [12]. More specifically, MobileNet models were designed to maximize accuracy with limited device resources. They are small, low-latency, and low-power models that can be used as a foundation for challenges such as classification, detection, and segmentation tasks. As a result, such designs are of paramount significance because they contributed to the definition of a basic technique to compact previously established architectures for mobile devices. In this frame of reference, it is significant to highlight the FastDepth [13] research which attempted to design a fast, compact, and highly-effective network architecture that is capable of compiling on-device inference efficiently. The presented network architecture employed MobileNet encoder architecture which utilizes a depth-wise decomposition.

In general, the literature is replete with the encoder model schemes to address depth estimation problems and successful CNN system realizations with enhanced performance. With this in mind, this paper studies the decoder-encoder architecture and attempts to develop a training model with enhanced prediction and minimum loss.

2.2 Datasets

Regarding the existing research papers, there are several datasets that can be used in modeling Monocular Depth estimation, and the main of them are early mentioned KITTI, NYU Depth, Cityscapes, Make3D having RGB images with corresponding features, depth annotation having different parameters and objects in them. The main difference is in the environment (indoor/outdoor, dynamic scenes), depth annotation type (abs/relative depth type), quality of the given datasets, and accuracy laser, human annotation taking them as a parameter.

In this project work, the dataset is given, therefore, it is decided to split the dataset for training and for testing separately, dividing 80 percent for training one, and 20 percent for testing accordingly. Provided Dataset has a relatively large stereo baseline and relatively static scenes.

2.3 Evaluation Metrics

Mean Square Error was chosen as the loss estimation metric as it was stated in the guidelines of the project:
$MSE = \frac{1}{n-2}\sum_{i=1}^{n}(Y_i - Y_i)^2$
In order to save time, early stopping callback was used in all of the fit functions, which stopped any calculations when the minimum loss was reached. Therefore, in the loss graphs, the number of epochs differs for distinct models.

3 CNN Architecture

In this section, we outline our suggested network architecture, the reasons for our design decisions, and the measures we take to shorten inference runtime and reduce the loss.

Fig. 2 depicts the fully convolutional encoder-decoder network for this project. It was based on FastDepth network design that has been adapted specifically for mobile and embedded applications and introduced originally in [13]. From the input image, the encoder extracts high-level low-resolution features. The final high-resolution output depth map is created by gradually upsampling, refining, and merging all of these features as they are fed into the decoder. The FastDepth paper performed a high-speed depth estimation with an efficient network design. It includes a low-latency decoder along with the efficient MobileNet encoder which applies depth-wise decomposition supporting low-complexity convolutional layers due to the single input channel. Therefore, in this study, the problem of monocular

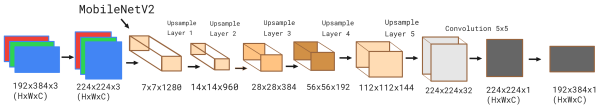

Figure 2: The network architecture design used in this work: The intermediate feature maps have the following dimensions: height, width and the number of channels.

depth estimation for a given dataset was resolved by using the network architecture based on the Fast-Depth design.

3.1 Data Preprocessing

Before proceeding to the construction of encoder-decoder architecture for our model, data augmentation and data exploration were accomplished. More specifically, the input image of 192 $\times$ 324 was horizontally flipped, randomly rotated, rescaled in the range of $[-1, 1]$, and ultimately, resized to the following dimensions: 224 $\times$ 224. Resizing was performed to match the encoder design because the proper input shape to the pre-trained MobileNet has to be 224 $\times$ 224 $\times$ 3.

3.2 Encoder Network

Mainly, ResNet-50 [6] and EfficientNet [12] have been common options among the encoder types due to their increased accuracy, but these networks tend to have high latency. In contrast, the MobileNetV2 encoder implements the depthwise decomposition by dividing a standard convolutional layer into depthwise layers and resulting in reduced complexity of a convolutional layer [13]. Thereby, in this study, the base model MobileNet was instantiated and with its pre-trained weights. Before compiling the model, as suggested in our model code, the *base_model.trainable* was set to *False* making all the layer's weights to freeze or, in other words, become non-trainable. This implies that during the training process, the condition of a frozen layer will not be modified. The total number of parameters for the encoding layer was accounted for $2,257,984$ and the output shape of 7 $\times$ 7 $\times$ 1280 was obtained as illustrated in Fig. 2. Initially, MobileNetV2 was designed for classification, but the layers responsible for the former were deleted.

3.3 Decoder Network

The subsequent step was to design a decoding layer, where the output of the encoder would be merged and upsampled to generate a dense prediction result. The upsampling process is an essential component of the decoder's design, and it includes operations such as unspooling, transpose convolution, and interpolation combined with convolution. The decoder design for this project was composed of the five cascading layers and a single point-wise layer, as demonstrated in Fig. 2. Every upsample layer was responsible for completing a 5 $\times$ 5 convolution and cutting the number of output channels by a factor of 0.5. After the convolution step, the round (or, in other words, nearest-neighbor) interpolation was performed, which increased the intermediate feature maps' resolution by 2. Furthermore, LeakyRelu was used as the activation function, subsequently adding the BatchNormalization layer for a better learning rate. As a result, the decoding layer's output image of 224 $\times$ 224 $\times$ 1 was resized to the original image shape of 192 $\times$ 324 $\times$ 1 by using bilinear interpolation. The paper tried to maintain the mirrored architecture of the encoder and decoder to get better results. Therefore, the number of filters and window sizes decreased and increased according to the MobilNetV2 architecture, which can be seen in Fig. 2.

3.4 Fine Tuning

After training the model with the training dataset and after reaching the convergence, fine-tuning was implemented by unfreezing the part of the base model. However, to avoid overfitting, a low learning rate of 10^{-5} was taken. In order to apply fine-tuning, *base_model.trainable* was set to $True$, and it is significant to do it after training the dataset on frozen layers of the base model. Furthermore, for unfreezing a model that contains BatchNormalization layers in order to do fine-tuning, *training* was set to *False*

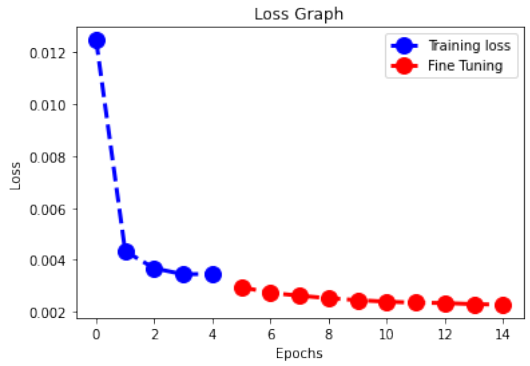

Figure 3: The Graph of the Loss during Training and Fine Tuning for MobilNetV2

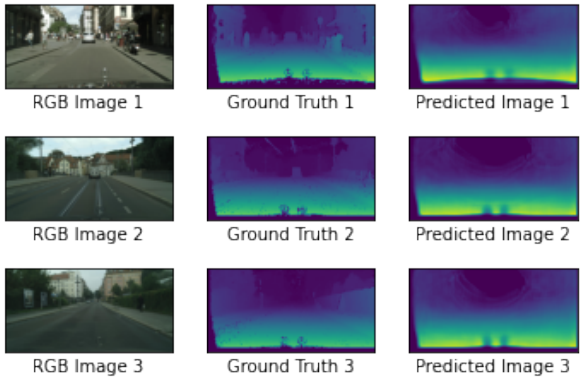

Figure 4: Results of the Predicted images and Pre-trained model ground truth

when calling the base model to maintain the Batch-Normalization layers in inference mode.

3.5 Experimental Results

Experimental results are presented in this section to demonstrate our approach to the problem. Train test split was utilized to train the introduced network and evaluate it in terms of its loss and runtime. The optimizer algorithm was set to *Adam* and loss was designated as *mse* to compute the mean of squares of errors between true labels and predictions. The default value for the learning rate is 10^{-3}. With such parameters, the model was trained for 25 epochs on the training dataset with a batch size of 8.

Specifically, Fig. 3 depicts the loss graph after implementing fine-tuning. It can be observed that the fine-tuning gave us incremental improvements in our training model. Furthermore, visualized results of the monocular depth estimation produced by the suggested model on the curated set of data are represented in Fig.4.

It is significant to mention that along with the MobileNetV2 encoder design, the ResNet50 was also applied in this project to compare the performance of the training models. The loss graph for ResNet50 is shown in Fig. 5. Even though ResNet50 is popular for its high-accuracy results, in this scenario, the MobileNetV2 showed superior performance in terms

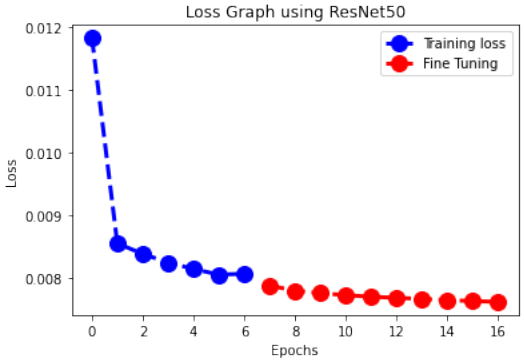

Figure 5: The Graph of the Loss during Training and Fine Tuning for ResNet50

of the loss, MSE, and runtime owing to its low-complexity architecture design. The comparison results are summarized in Table I.

Comparison of Different Encoders

Encoder	Loss	MSE	Runtime
MobileNetV2	0.0022	148	0.08
ResNet50	0.0076	210	0.10

Comparison of Encoders

4 Conclusion

In this project work, we tried to train and test high-speed depth estimation that will fit embedded devices. Focusing on low latency and low complexity, we have tried ResNet and Mobilenet pre-trained models as promising network architectures and reached results that do not monopolize inference runtime even when paired with a tiny encoder in order to achieve high frame rates. Although the work here focuses on depth calculation, we think that comparable techniques may be applied to other dense prediction tasks, including picture segmentation, to attain real-time performance using deep-learning-based methods. And Finally, there are several ways to achieve higher results by adding cutting-edge pruning, and hardware-specific compilation-like methods. Therefore, for further development, the project has to be continued.

References

[1] Vincent Casser, Sören Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:8001–8008, 07 2019. 3

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. 06 2016. 2

[3] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012. 2

[4] Clement Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6602–6611, United States, Nov. 2017. Institute of Electrical and Electronics Engineers (IEEE). 2017 IEEE Conference on Computer Vision and Pattern Recognition , CVPR 2017 ; Conference date: 21-07-2017 Through 26-07-2017. 2

[5] Clement Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3827–3837, United States, Feb. 2020. Institute of Electrical and Electronics Engineers (IEEE). International Conference on Computer Vision 2019, ICCV 2019 ; Conference date: 27-10-2019 Through 02-11-2019. 2

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 06 2016. 3, 4

[7] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017. 3

[8] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017. 3

[9] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. *CoRR*, abs/1804.00607, 2018. 2

[10] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):824–840, 2009. 2

[11] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. volume 7576, pages 746–760, 10 2012. 2

[12] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 05 2019. 3, 4

[13] Diana Wofk, Fangchang Ma, Tien-Ju Yang, Sertac Karaman, and Vivienne Sze. Fastdepth: Fast monocular depth estimation on embedded systems. 05 2019. 3, 4