Monocular Depth Estimation with ResNet-34 and a Custom Upsampling Decoder

Giovanni Lopopolo
Deep Learning and Robot perception
Ingegneria Informatica e Robotica
Università degli Studi di Perugia

***Abstract*—Monocular depth estimation (MDE) is a fundamental task in computer vision, used in applications ranging from autonomous navigation to robotics. This paper proposes a supervised MDE model that combines a pre-trained ResNet-34 encoder with a customized upsampling decoder to generate Depth Maps from RGB images. Key contributions include freezing the first levels of the encoder for regularization and efficiency and using a combined loss function based on RMSE and SSIM metrics. The experiments, conducted on a dataset generated with the Unreal Engine graphics simulator, highlight the model's ability to estimate Depth Maps. The study demonstrates the potential of deep learning in addressing challenges traditionally solved by geometric techniques.**

I. Introduction

Despite the progress made in deep learning, succeeding in obtaining accurate depth predictions while ensuring computational efficiency, remains one of the most complicated challenges. Indeed, the absence of stereoscopic or temporal signals makes the work of monocular depth estimation (MDE) (which extracts depth information via a single RGB image, relying solely on spatial patterns and contextual signals within a single frame, itself becoming inherently ambiguous as a result) very difficult. The project under examination, therefore, is concerned with estimating the depth of each pixel present in an RGB image through a supervised approach, producing a Depth Map in which each pixel encodes the distance from the object. This supervision integrates a ResNet-34 encoder supported by a decoder upsampling customized to improve the spatial resolution of the Depth Maps. A crucial question that emerges in this work is: what is the most appropriate Loss Function to tackle the task of monocular depth estimation? A combined loss function based on RMSE (Root Mean Squared Error) and SSIM (Structural Similarity Index) was chosen, designed to balance the global accuracy with the structural consistency of the produced Depth Maps. This is a mechanism to exploring the potential of deep neural networks in order to emulate certain abilities, such as the human capability to perceive depth, as it relies on understanding context, object proportions and shadows. This is advantageous because it represents an effective trade-off between computational complexity and accuracy compared to other known approaches.

II. Related Work

Monocular depth estimation (MDE) has made significant progress in recent years, with approaches mainly divided into supervised and self-supervised methods. Traditional geometric approaches to depth estimation, such as stereo matching or multiview reconstruction, rely on the use of multiple images or cameras to resolve depth-related ambiguities. These methods exploit correspondences between different views to reconstruct 3D scenes more accurately. However, in monocular configurations, the intrinsic loss of dimensional information during 2D projection makes these techniques inapplicable without additional constraints, limiting their effectiveness in scenarios where only one view is available. This limitation has stimulated the development of modern methods, including supervised and self-supervised learning. Supervised methods, such as those introduced by Eigen et al. [1] and Laina et al. [2], achieve high accuracy by exploiting the ground truth of Depth Maps during training. Although these approaches excel in controlled environments, they require labeled datasets, which makes these approaches expensive and difficult to produce. To reduce this dependence, self-supervised approaches, such as those proposed by Godard et al. [3] and Zhou et al. [4], exploit self-learning signals, such as photometric coherence between consecutive video frames. Although this reduces dependence on labeled data, it often compromises accuracy, particularly in regions with dynamic objects or poor textures. This distinction between the two paradigms highlights a fundamental trade-off between accuracy and data accessibility that is a central challenge in the field of MDE. Recently, approaches based on advanced deep networks have led to further significant advances. MiDaS [5] uses a pre-trained backbone to create generalized and highly accurate Depth Maps over several domains. This method, while robust in unseen scenarios, has a high computational cost during inference. EfficientNet [6], on the other hand, demonstrated that uniform scalability of model width, depth, and resolution can optimize the relationship between performance and computational complexity, making it particularly suitable for high-resolution images. However, both of these architectures suffer from high computational complexity, limiting their adoption in scenarios where efficiency is critical. It is interesting to note that humans are able to infer depth from a single viewpoint by exploiting contextual features such as proportions, occlusions and shadows, as well as previous experience. This ability has inspired the development of deep learning-based approaches that aim to emulate human depth perception. By learning from large datasets, deep neural networks can capture intricate relationships between visual features and depth, enabling Depth Maps to be estimated directly from RGB images. Inspired by these observations, our approach proposes a methodology that balances accuracy and computational efficiency. We use a ResNet-34 encoder and a customized upsampling decoder to address MDE. Our architecture aims to optimize computational complexity without compromising accuracy by harnessing the power of supervised learning to resolve the inherent ambiguities of depth in a single RGB image

III. Methodology

A. Model Architecture

An encoder-decoder architectural model was used in this work.

1) Encoder Architecture: On the Encoder side, we employ a partial fine-tuning approach for ResNet 34 that was chosen for its ability to extract hierarchical image features through its residual block architecture [7]. Pre-trained on ImageNet, it provides a robust feature extractor for moderate-sized datasets, achieving a balance between complexity and accuracy. The convolutional layers responsible for extracting general features such as edges and textures were frozen to maintain their pre-trained weights, which is part of the strategy developed by Yosinski et al. [8], which highlights the utility of freezing the first layers to preserve generalizable features. It was then decided to remove the fully-connected layers of ResNet-34, as they were designed for classification tasks with 1000 classes in ImageNet. The removed layers were replaced by a custom decoder that enables a regression task. This brings us even more computational efficiency and task-specific adaptation (Simonyan and Zisserman, 2015) [9]. The modification was done in order to adapt the pre-trained models to nonclassification tasks (Razavian et al. 2014 [10] and Laina et al. 2016) [2].

2) Decoder Architecture: Concerning the decoder on the other hand, it is designed to progressively increase the sampling of feature maps extracted from the ResNet-34 encoder, reconstructing a dense Depth Map at 224×224 resolution. It employs transposed convolutions (`ConvTranspose2d`) to increase spatial resolution, with kernels of size 4×4, stride 2 and padding 1. On the one hand, this design ensures a smooth transition from the ResNet-34 encoder to the extracted feature map; on the other hand, it ensures a smooth transition from the low-resolution feature maps (7×7) to the high-resolution Depth Map, which corresponds to the size of the input image.

The output dimensions of a transposed convolution layer are calculated as:

$$\begin{aligned} H_{\text{out}} &= (H_{\text{in}} - 1) \cdot \text{stride} - 2 \cdot \text{padding} \\ &\quad + \text{kernel\_size} + \text{output\_padding}, \\ W_{\text{out}} &= (W_{\text{in}} - 1) \cdot \text{stride} - 2 \cdot \text{padding} \\ &\quad + \text{kernel\_size} + \text{output\_padding}. \end{aligned}$$

This combination ensures that each decoder level gradually increases in spatial resolution until the desired size is reached.

3) BatchNormalization: Batch normalization is a technique introduced to solve the problem of internal co variant displacement during training of deep neural networks. This technique normalizes the activations of one layer per mini-batch, improving the stability and speed of training convergence. It also acts as a regularizer, eliminating the need for Dropout in some cases [11].

4) Activation Function: The activation function used is ReLU [12] to stabilize training and introduce nonlinearities, which improve the model's ability to learn complex spatial relationships.

5) Intermediate Convolution: Intermediate convolutions 1×1 , used after the second, fourth layers and at the end of the decoder itself, act as channel transformations, combining information between feature maps without altering spatial resolution, in line with strategies used in modern architectures [2].

6) Evolution of Decoder: The decoder reduces the number of channels progressively ($512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 1$) while increasing spatial resolution, achieving an efficient trade-off between computation and accuracy. This architecture was chosen over more complex designs (e.g., U-Net-style skip connections) to prioritize computational efficiency and simplicity while maintaining strong reconstruction capabilities.

TABLE I
Evolution of Feature Map Dimensions through the Decoder

Layer	**Feature Map Dimensions**	**Number of Channels**
Input	7×7	512
Transposed Conv 1	14×14	256
Transposed Conv 2	28×28	128
Conv	28×28	128
Transposed Conv 3	56×56	64
Transposed Conv 4	112×112	32
Conv	112×112	32
Transposed Conv 5	224×224	1
Conv	224×224	1
Output	224×224	1

7) Comparison with Alternative Choices: While ResNet 50 or ResNet-101 could have been used as encoders, their higher complexity and higher risk of overfitting made ResNet-34 a more suitable choice for a moderate-sized dataset such as DepthEstimationUnreal. In addition, DenseNet or EfficientNet architectures, known for their efficiency, could be explored in the future to improve model performance.

Concerning the decoder, the inclusion of jump connections, as seen in U-Net [13], could further improve fine detail preservation. However, the additional computational cost associated with such architectures was avoided in this implementation to maintain efficiency.

B. Dataset and Preprocessing

At the beginning of the project, the training and validation datasets were provided, but the test dataset that was then used by Professor Costante to evaluate the final model was not. Specifically, the dataset provided is DepthEstimationUnreal, generated using Unreal Engine, a graphical simulator capable of producing ground truth Depth Maps from RGB images. The dataset includes thousands of RGB images and paired Depth Maps, divided into training and validation subsets. The input is RGB images, stored in .jpeg format and normalized to the interval [0, 1], while the Depth Maps, i.e., the paired target, are stored in .npy format and are clipped to the interval [0.0, 20.0]. In the encoding of the Depth Maps, lower intensity values (black, blue, purple) indicate closer objects, while higher intensity values (yellow, orange) represent objects farther from the camera. Since ResNet34 was developed to receive input images with a resolution of 224×224, it was necessary to preprocess both the training dataset, the validation dataset, and the test dataset used later by the Professor.

Two transformations were applied to convert the resolution of the RGB input and Depth Maps target from 144 × 256 to 224 × 224:

1) **Resize**$(224, 224)$: This transformation scales the image to the specified dimensions, in this case 224×224.
2) **CenterCrop**$(224, 224)$: This transformation crops a central area of the image to the specified size, ensuring that the central part of the resized image remains meaningful.

Table II lists the dataset preprocessing steps.

TABLE II
PREPROCESSING PARAMETERS

Parameter	Value
Depth Clipping	[0.0, 20.0]
Normalization	RGB: [0, 1]
Transformations	Resize, CenterCrop
Image Resize	224x224

1) Data Augmentation: Although we recognize the importance and effectiveness of data augmentation in supervised depth estimation (Jaipuria e al. 2020) [14], this work does not include any part dedicated to it, because the results obtained during the process did not improve the generalization of the model, worsening, on the other hand, the recognition of features within the images. Data augmentation generally involves altering the brightness of an image, or going to affine, rotation, translation and scaling transformations. In the specific case of this project, the use of these transformations resulted in black borders that compromised the context of the image. The random nature of the data augmentation also prevented optimal comparison as that performed on the input images was different from that of the respective targets.

C. Loss Function

The issue raised in the introduction of the paper about which is the best Loss Function to use, finds more explanation below: in order to identify the appropriate Loss Function, it is necessary to align model optimization with the most relevant evaluation metrics. Hence, RMSE (Root Mean Squared Error) and SSIM (Structural Similarity Index), commonly used to assess the quality of estimated Depth Maps, were chosen because they are able to capture complementary aspects of model performance, In fact, RMSE measures overall numerical accuracy by penalizing large discrepancies; SSIM focuses on preserving visual and perceptual structure, ensuring local consistency in estimates. The simultaneous use of both in the Loss Function context not only improves the learning process, but also steers the model toward the same characteristics required for successful estimation. Such a combination preserves both numerical and perceptual consistency, proving effective in managing task complexity [15].

As already theorized, the loss function used in this work combines RMSE and SSIM, whose function is defined:

$$\mathcal{L} = \alpha \cdot \text{RMSE} + \beta \cdot (1 - \text{SSIM}), \tag{1}$$

where:

- α and β are weighting factors, both set to 1.0 in this work to give equal importance to both terms,
- RMSE measures the overall numerical difference between the predicted and ground truth depths,
- $1 - \text{SSIM}$ measures the structural dissimilarity between the predicted and ground truth Depth Maps.

The RMSE loss is computed as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}, \tag{2}$$

where N is the number of pixels, $\hat{y}_i$ is the predicted depth value for the i-th pixel, and y_i is the ground truth depth value. RMSE focuses on minimizing global errors across the entire Depth Map.

The SSIM-based loss is computed as:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{3}$$

where μ_x, μ_y are the means, σ_x^2, σ_y^2 are the variances, and σ_{xy} is the covariance between the predicted Depth Map (x) and the ground truth (y). Constants C_1 and C_2 stabilize the computation. The loss term is calculated as:

$$\text{SSIM Loss} = 1 - \text{SSIM}(x, y),$$

The combined loss is implemented as:

```
ssim_loss = 1 - ssim(output, depth)
rmse_loss = torch.sqrt(F.mse_loss(output,
    depth))
loss = alpha * rmse_loss + beta * ssim_loss
```

where `output` is the predicted Depth Map and `depth` is the ground truth Depth Map.

D. Optimizer and Learning Rate Scheduler

1) Optimizer: The Adam optimizer [16] was used to train the proposed model because it is able to combine the advantages of both the AdaGrad and RMSProp algorithms. It also dynamically adjusts the learning rate for each parameter based on the estimates of the first and second moments of the gradients. This makes it particularly suitable for problems with sparse gradients or noisy targets, such as depth estimation. The optimizer is configured with the following parameters:

- **Learning Rate:** $\texttt{lr} = 0.001$
- **Weight Decay:** 1×10^{-5}

2) Learning Rate Scheduler: To improve convergence and generalization, a cyclic learning rate scheduler (CyclicLR) [17] was used. CyclicLR periodically varies the learning rate between a minimum ($\texttt{base\_lr} = 0.001$) and maximum ($\texttt{max\_lr} = 0.005$) value over the course of training. The schedule follows a triangular2 policy, where the learning rate oscillates in a triangular pattern with a decreasing amplitude after each cycle. This helps the model escape local minima and improves training stability.

The scheduler was configured with the following parameters:

- **Base Learning Rate (`base_lr`):** 0.001
- **Maximum Learning Rate (`max_lr`):** 0.005
- **Step Size Up:** 1000 iterations
- **Step Size Down:** 2000 iterations
- **Mode:** triangular2

The Adam optimizer was chosen for its ability to handle noisy gradients and dynamic adjustments of the learning rate, ensuring robust training even in the presence of non-stationary objectives.

The CyclicLR scheduler complements Adam by introducing periodic variations in the learning rate, which encourages the model to explore a wider solution space, potentially avoiding overfitting and improving generalization [17].

E. Execution Pipeline

The ‘main.py‘ script defines key parameters (Table III) and orchestrates the training and evaluation processes.

TABLE III
HYPERPARAMETERS

Parameter	Value
Learning Rate	0.001
Batch Size	32
Max Epochs	100
Eval Frequency	Every 2 epochs
Visualization	Every 100 steps

IV. EXPERIMENTS

A. Metrics

Performance is evaluated using RMSE and SSIM. Qualitative analysis includes visualizing Depth Maps with the ‘visualize_img‘ function (Fig. 3).

B. Results

During the training, 100 epochs were used and a continuous decrease in Loss was evident, arriving at epoch 96 at the value of Loss: 0.7666. In addition, the RMSE and SSIM values also tended to improve, however a discrepancy was observed due to overfitting, but still able to generalise as evidenced by the visual results and test results provided by Professor Costante.

- Loss: 0.7666
- Training set:
  - RMSE on TRAIN : 0.579466572321883
  - SSIM on TRAIN: 0.856304222290669

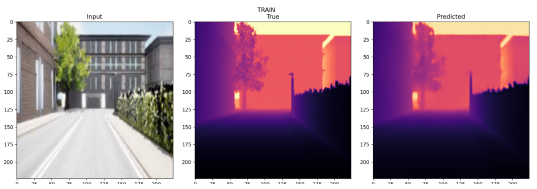

Fig. 1. Visualization of input RGB image, ground truth, and predicted Depth Map with Training Dataset

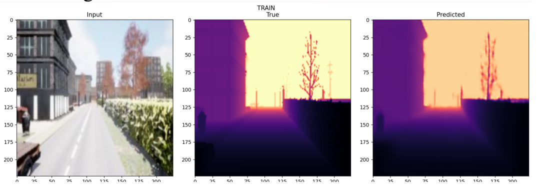

Fig. 2. Visualization of input RGB image, ground truth, and predicted Depth Map with Training Dataset

- Validation set:
  - RMSE on VALIDATION : 2.810505590940777
  - SSIM on VALIDATION: 0.5213211429746527

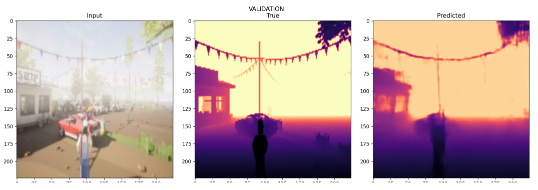

Fig. 3. Visualization of input RGB image, ground truth, and predicted Depth Map with Validation Dataset

- Test set:
  - RMSE on TEST : 2.218269580288937
  - SSIM on TEST : 0.6457219939482847

V. CONCLUSION

This paper presented a supervised MDE approach combining a ResNet-34 encoder and a custom upsampling decoder. While the model achieved promising results on the training set, validation performance highlights the need for better regularization and data augmentation.

The human ability to perceive depth with one eye relies on contextual understanding, object dimensions, and the interplay of light and shadow. This work demonstrates that deep neural networks can emulate such capabilities by learning from experience, highlighting their potential to tackle tasks traditionally solved by geometric techniques. Future work will explore advanced encoders, skip connections, and domain adaptation to improve generalization.

References

[1] D. Eigen, C. Puhrsch, and R. Fergus, ”Depth Map Prediction from a Single Image using a Multi-Scale Deep Network,” NeurIPS, 2014.

[2] I. Laina et al., ”Deeper Depth Prediction with Fully Convolutional Residual Networks,” 3DV, 2016.

[3] C. Godard, O. Mac Aodha, and G. J. Brostow, ”Unsupervised Monocular Depth Estimation with Left-Right Consistency,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6602–6611. Available: https://arxiv.org/abs/1609.03677

[4] T. Zhou et al., ”Unsupervised Learning of Depth and Ego-Motion from Video,” CVPR, 2017.

[5] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, ”Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 44, no. 3, pp. 1623–1637, 2022. Available: https://arxiv.org/abs/1907.01341

[6] M. Tan and Q. Le, ”EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” in *Proceedings of the International Conference on Machine Learning (ICML)*, 2019, pp. 6105–6114. Available: https://arxiv.org/abs/1905.11946

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, ”ImageNet: A large-scale hierarchical image database, 2009, pp. 248–255.

[8] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, ”How Transferable Are Features in Deep Neural Networks?”, vol. 27, 2014, pp. 3320–3328.

[9] K. Simonyan and A. Zisserman, ”Very Deep Convolutional Networks for Large-Scale Image Recognition”, in International Conference on Learning Representations (ICLR), 2015. Available: https://arxiv.org/abs/1409.1556.

[10] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, ”CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2014, pp. 512–519.

[11] S. Ioffe and C. Szegedy, ”Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015, pp. 448–456. Available: https://arxiv.org/abs/1502.03167.

[12] V. Nair and G. E. Hinton, ”Rectified Linear Units Improve Restricted Boltzmann Machines,” in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.

[13] O. Ronneberger, P. Fischer, and T. Brox, ”U-Net: Convolutional Networks for Biomedical Image Segmentation,” MICCAI, 2015.

[14] R. Jaipuria, S. A. Kadambi, and M. Tomizuka, ”Depth Estimation from Monocular Images and Sparse Radar using Deep Ordinal Regression,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10218–10225. Available: https://doi.org/10.1109/IROS45743.2020.9340844

[15] Z. Wang et al., ”Image Quality Assessment: From Error Visibility to Structural Similarity,” IEEE TIP, 2004.

[16] D. P. Kingma and J. Ba, ”Adam: A Method for Stochastic Optimization,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. Available: https://arxiv.org/abs/1412.6980.

[17] L. N. Smith, ”Cyclical Learning Rates for Training Neural Networks,” in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 464–472. Available: https://arxiv.org/abs/1506.01186.