Monocular Depth Estimation

Riccardo Nicolini

Abstract

This work addresses the problem of Monocular Depth Estimation, that is, the prediction of depth maps from a single RGB image. MDE can be considered a dense prediction task, as it involves associating an estimated distance value from the camera to each pixel of the image. The goal is to train a deep learning model that minimizes the RMSE error compared to the ground truth depth maps and, at the same time, maximizes structural quality metrics such as SSIM. The performance will be evaluated on unseen data during training and compared with that of other models, in order to assess the effectiveness of the proposed model.

Introduction

Depth estimation from a single image is one of the most important problems in computer vision, with critical applications in autonomous driving, robotics, and augmented/virtual reality. Instead of traditional geometric methods, such as stereo vision and visual SLAM, which require multiple viewpoints to reconstruct the scene, deep learning-based methods learn a direct mapping from an RGB image to a depth map, leveraging monocular cues such as texture, blur, and shadows. A depth map is a dense prediction, that is, a matrix in which each pixel of the image has a corresponding estimated distance value from the camera. From a visual perspective, a depth map can be represented with intensity gradients: light images correspond to distant objects, while dark ones correspond to nearby objects. Despite recent advances, MDE remains a challenge due to the presence of ambiguities and the generalization of unseen scenes during training.[1]

Related Work

The use of convolutional neural networks has allowed overcoming the limitations of traditional stereo vision-based systems, enabling the accurate generation of depth maps from a single image. Modern models such as MiDaS [2], ZoeDepth, and PatchFusion are now state-of-the-art in MDE. Some architectures originally designed for other dense prediction tasks have inspired approaches to MDE. UNet was originally proposed for biomedical segmentation; the encoder-decoder scheme with symmetric skip connections has proven excellent in preserving spatial detail. The residual connections introduced by ResNet have been fundamental in stabilizing training and improving gradient flow in decoders. Finally, modern models like ConvNext have improved classic models like ResNet; thanks to depthwise convolutions, wider kernels, LayerNorm, inverted bottlenecks, and the use of GELU, ConvNext combines the simplicity of convolutional networks with features typical of transformers, enhancing CNN performance.

Proposed Approach

The proposed model uses a fully convolutional Encoder-Decoder architecture that consists solely of convolutional layers. This structure is capable of maintaining the spatial integrity of the input image, providing as output a depth map with an estimate for each pixel. Since applying convolutions directly on the full-resolution image would be computationally expensive, the encoder performs progressive downsampling, reducing spatial dimensions and extracting features, while the decoder reconstructs the original resolution through upsampling operations, thus returning the final depth map.

The dataset used consists of 3469 input images of resolution 256×144 pixels paired with the corresponding ground truth depth maps of the same resolution; a validation set allows measuring the model's generalization performance during training. The starting model, used as a baseline, was designed with a relatively simple encoder-decoder structure. The encoder consists of a pre-trained ResNet50, stripped of the fully connected layers.

Connected finally, which progressively reduces the spatial resolution of the input image. The decoder consists of a sequence of four transposed convolutions, each followed by a ReLU activation function, with the task of gradually bringing the representation back to the original dimensions.

Finally, a 1×1 convolutional layer reduces the channels to a single value per pixel, thus producing the final depth map.

To overcome the limitations of the baseline, the architecture has been modified by introducing three components: U-Net and symmetric skip connections.

U-Net style architecture

A first improvement was the insertion of a U-Net style structure, originally proposed by Ronneberger et al. [3] for the segmentation of biomedical images. The U-Net architecture is characterized by a symmetric encoder-decoder shape, in which each downsampling level corresponds to a direct connection (skip connection) to the corresponding upsampling level.

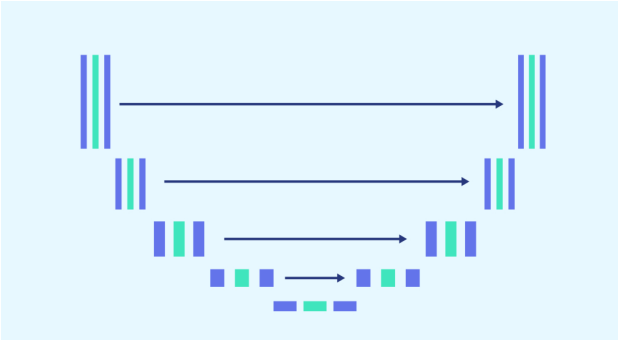

Figure 1: *U-Net style architecture.*

These connections play a fundamental role in dense prediction tasks: they allow the reintegration of low-level information, such as edges, textures, and spatial details, which would otherwise be lost during the dimensionality reduction process performed by the encoder. In this way, the model is able to preserve the structure of the image and generate more detailed depth maps. [4] [5]

Residual blocks in the decoder

A second improvement introduced concerns the use of residual blocks within the decoder. Residual blocks, first introduced in ResNet, aim to simplify gradient propagation during training, reducing the vanishing gradient problem and allowing for the training of deeper networks.

From an architectural point of view, a residual block includes a skip connection that directly links the input of the block to its output, summing the original signal with the convolutional transformation. In this way, if the convolutional layer is not useful, the model can simply leave the input unchanged, preventing the signal from being excessively attenuated. [6]

In the proposed decoder, after each upsampling phase and concatenation with the features coming from the encoder (U-Net), a residual block is applied, consisting of two 3×3 convolutions with Batch Normalization and ReLU activation function, accompanied by a skip connection, containing a 1×1 convolution to align the channel dimensions. 2

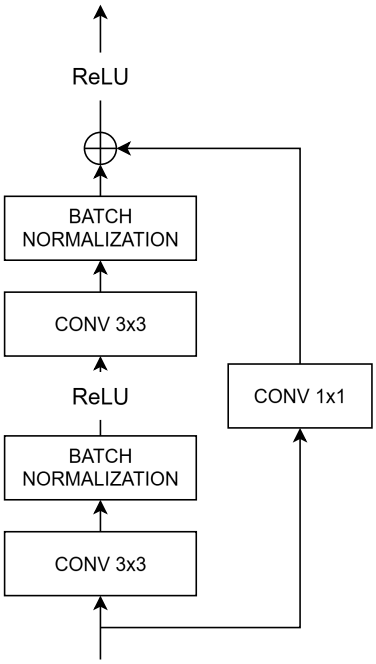

Figure 2: *Schema of the residual block.*

ConvNext as encoder

An additional improvement over the baseline was introduced by replacing ResNet50 with ConvNext as a pre-trained encoder. ConvNext, presented by Liu et al. in the paper ConvNet for the 2020s [7], represents a modernization of ResNet architectures, inspired by Vision Transformers but maintaining the typical structure of convolutional networks.

The main innovations of ConvNext include:

- **Depthwise convolution**: Each filter operates on a single channel, thus imitating transformers that alternate spatial attention and channel fusion in MLP.
- **Kernel Ampio**: ConvNext adopts a larger kernel (7x7 instead of 3x3).
- **Inverted Bottleneck**: Unlike ResNet, ConvNext first performs a 1x1 expansion convolution, applies the depthwise convolution 7x7, and finally applies a 1x1 compression convolution to return to the original size. In this way, the largest number of channels is present in the core of the block, rather than at its extremes, offering more expressive capacity.
- **GELU Activation function**: ConvNext replaces ReLU with GELU (Gaussian Error Linear Unit), which is a softer activation that does not truncate

negative values but attenuates them. Additionally, the activations are reduced compared to ResNet, simplifying the residual path and reducing distortions due to too many nonlinearities.
- **Layer Normalization**: ResNet uses Batch Normalization after each convolution, ConvNext introduces Layer Normalization that normalizes the activations per channel on each individual sample, unlike BatchNorm which uses all batch samples. This change makes the model less dependent on batch size. Again, the normalization layers are reduced.

In the proposed model, ConvNext was used in features only mode, in order not to extract the final prediction but exclusively feature maps at different scales of resolution (1/4, 1/8, 1/16, and 1/32) of the input. The use of a modern encoder significantly impacted the performance of the final model. [8]

The final architecture is viewable in Figure 3.

Experiments

The experimental phase was conducted on a virtual machine equipped with an NVIDIA L4 GPU, which ensured sufficient computing power to handle the training of the network.

To monitor the model's progress, the Weights & Biases (wandb) platform was used, which allowed logging of the main indicators during training. In particular, the evaluation was performed every 2 epochs, recording the RMSE and SSIM metrics on both the training and validation sets. This allowed real-time visualization of the model's evolution and identification of any overfitting phenomena.

Two batch sizes were tested: With a batch size of 32, the model showed a more oscillating gradient behavior (more "nervous" loss), while with a batch size of 64, it showed a more "smooth" descent without obvious signs of overfitting. For this reason, a batch size of 64 samples was used for the final training.

The training was performed for a total of 80 epochs, a value that proved sufficient to ensure the convergence of the model.

Loss As a loss function, a composite function was used, defined by the combination of three different losses [9]:

- **MSE (Mean Squared Error)**: Aiming to minimize the validation RMSE metric.

  $MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$

- **SSIM loss**: $L_{SSIM} = 1 - SSIM(\hat{y}, y)$

- **Edge loss**: Introduced to preserve contours and improve the definition of transitions between objects. This loss is calculated as the sum of the absolute differences between the derivatives (x and y) of prediction and ground truth.

  $L_{Edge} = \frac{1}{N}\sum_{i=1}^{N}(|\partial_x\hat{y}_i - \partial_x y_i| + |\partial_y\hat{y}_i - \partial_y y_i|)$

Finally, the overall loss is expressed as a weighted combination of the three terms:

$$L = MSE \ + \ 0.15 \cdot L_{SSIM} \ + \ 0.05 \cdot L_{Edge}$$

Optimizer As an optimizer, AdamW was used, a variant of Adam that is more commonly used today [10]. Unlike traditional Adam, which applies weight decay as part of the loss, AdamW explicitly separates the regularization term, avoiding unwanted interactions with the optimization process.

To further improve the training phase, two different learning rates were adopted, a smaller one for the parameters of the ConvNeXt encoder (1e-4), which being pre-trained requires only gradual fine-tuning; a larger one (1e-3) for the parameters of the decoder, initialized from zero and therefore needing a faster update.

This separation was implemented by selecting all network parameters that start with ConvNext.

Scheduler For managing the learning rate, the OneCycleLR scheduler was adopted. [11].

The One Cycle strategy involves starting the learning rate from a very low value, increasing it to a predefined maximum value (maxlr), and then decreasing it again to low values, following a cyclical pattern during training.

The scheduler is defined by the function:

$$lr(t) = lr_{min} + (lr_{max} - lr_{min}) \cdot f(\frac{t}{T})$$

in which the parameters correspond to:

- lr_{min}: minimum value of the learning.
- lr_{max}: maximum value of the learning.
- t: current iteration.
- T: total number of iterations.
- f: cosine function that defines the trend of the cycle.

The final graphs that express the trend of RMSE and SSIM on the training and on the validation are reported below:

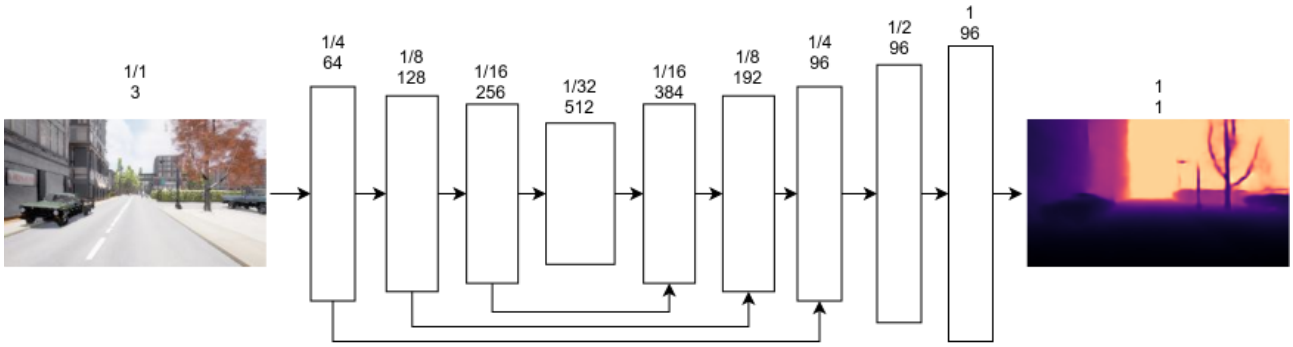

Figure 3: *Architecture of the final neural network.*

Figure 4: *RMSE on the Training Set.*

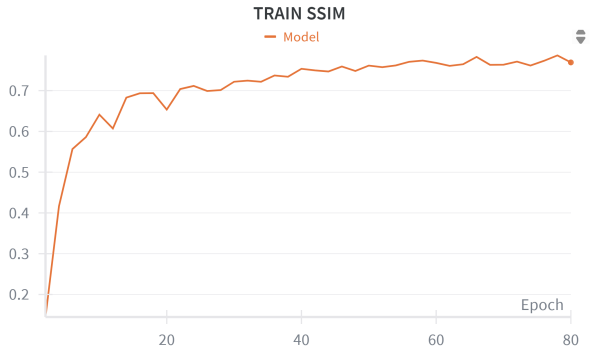

Figure 5: *SSIM on the Training Set.*

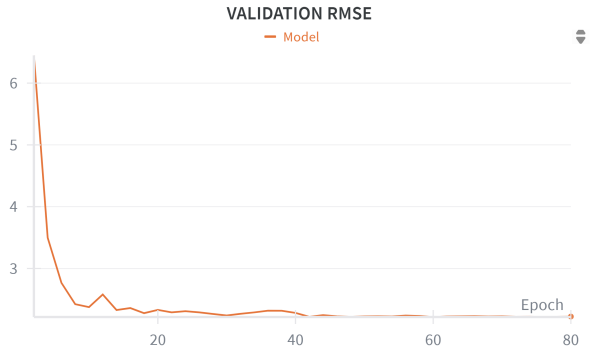

Figure 6: *RMSE on the Validation Set.*

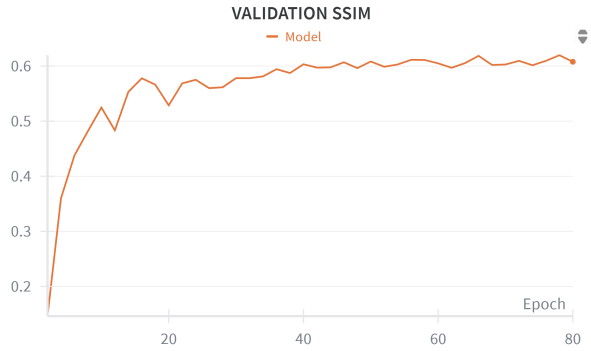

Figure 7: *SSIM on the Validation Set.*

Finally, the parameters of the best checkpoint (60) were tested on the validation test, obtaining the following metrics:

- RMSE: 2.3156763076782227
- SSIM: 0.5699696362018585

In Figure 8 it is possible to visualize an example of output provided by the network.

Conclusion

The architecture proposed in the report showed good performance on the validation set, demonstrating the effectiveness of combining various techniques.

In particular, the use of ConvNext as a modern encoder, integrated with U-Net decoders and residual blocks, improved the quality of predictions compared to the initial baseline.

During training, the choice of an appropriate loss, the use of AdamW with two different learning rates for the encoder and decoder, and the use of a OneCycle scheduler favored stable and effective convergence.

As future developments, the model could be further improved by using data augmentation techniques to enhance generalization.

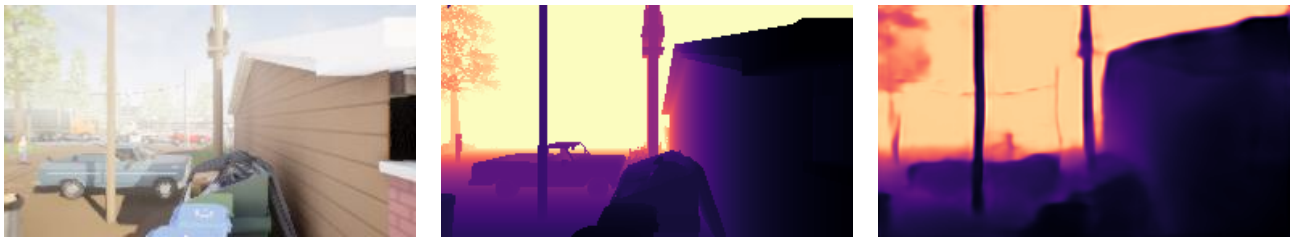

Figure 8: *Visualization of the network output compared to the depth map and the provided input.*

From an architectural perspective, it would be interesting to experiment with the use of Vision Transformers (ViT) in the encoder or hybrid CNN-Transformer models, in order to compare their ability to capture global relationships against purely convolutional approaches.

References

[1] Jiuling Zhang. *Survey on Monocular Metric Depth Estimation*. 2025. arXiv: 2501.11841 [cs.CV]. URL: https://arxiv.org/abs/2501.11841.

[2] Reiner Birkl, Diana Wofk, and Matthias Müller. *MiDaS v3.1 – A Model Zoo for Robust Monocular Relative Depth Estimation*. 2023. arXiv: 2307.14460 [cs.CV]. URL: https://arxiv.org/abs/2307.14460.

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV]. URL: https://arxiv.org/abs/1505.04597.

[4] Zhitong Lai et al. *Rethinking Skip Connections in Encoder-decoder Networks for Monocular Depth Estimation*. 2022. arXiv: 2208.13441 [cs.CV]. URL: https://arxiv.org/abs/2208.13441.

[5] Diana Wofk et al. *FastDepth: Fast Monocular Depth Estimation on Embedded Systems*. 2019. arXiv: 1903.03273 [cs.CV]. URL: https://arxiv.org/abs/1903.03273.

[6] Hoang-Thanh Duong, Hsi-Min Chen, and Che-Cheng Chang. "URNet: An UNet-Based Model with Residual Mechanism for Monocular Depth Estimation". In: *Electronics* 12.6 (2023). ISSN: 2079-9292. DOI: 10.3390/electronics12061450. URL: https://www.mdpi.com/2079-9292/12/6/1450.

[7] Zhuang Liu et al. *A ConvNet for the 2020s*. 2022. arXiv: 2201.03545 [cs.CV]. URL: https://arxiv.org/abs/2201.03545.

[8] Duc To. *From Resnet to ConvNeXt : Modernizing a Vanilla ResNet*. https://tokudayo.github.io/from-resnet-to-convnext-2/. 2022.

[9] Muhammad Hafeez et al. "Depth Estimation Using Weighted-Loss and Transfer Learning". In: *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2024, pp. 780–787. DOI: 10.5220/0012461300003660. URL: http://dx.doi.org/10.5220/0012461300003660.

[10] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG]. URL: https://arxiv.org/abs/1711.05101.

[11] Ayman Al-Kababji, Faycal Bensaali, and Sarada Prasad Dakua. *Scheduling Techniques for Liver Segmentation: ReduceLROnPlateau Vs OneCycleLR*. 2022. arXiv: 2202.06373 [cs.CV]. URL: https://arxiv.org/abs/2202.06373.