Project Summary — Monocular Depth Estimation Using CNN-Based UNet++ on NYU Depth v2

The purpose of this project is to develop a deep learning model capable of producing high-quality monocular depth maps from a single RGB image using a convolutional encoder–decoder architecture. Monocular Depth Estimation is one of the central problems in computer vision and robotics because it enables an understanding of scene geometry without requiring stereo images or additional depth sensors. This capability is essential for tasks such as visual SLAM, autonomous navigation, AR/VR spatial reconstruction, 3D mapping, and high-level scene interpretation. The core challenge is that depth must be inferred from visual cues alone, making the learning process heavily dependent on large annotated datasets and expressive neural architectures. In this project, the NYU Depth v2 dataset is used as the primary training source, providing a rich set of aligned RGB–Depth image pairs captured in diverse indoor environments.

The dataset is provided in the form of a CSV file, where each entry corresponds to the path of an RGB image and the path of its corresponding depth map. These entries are mapped to full directory paths and organized into a DataFrame for efficient indexing. The dataset contains approximately fifty thousand aligned samples, which are then split into training, validation, and test subsets in a ratio that ensures both diversity and statistical reliability: around forty-five thousand samples for training, four and a half thousand for validation, and several hundred for testing. This large-scale dataset allows the model to learn a broad spectrum of geometric and structural patterns, such as object boundaries, orientation changes, surface smoothness, lighting variations, and global spatial layout.

A substantial part of the project involves designing a robust and meaningful pre-processing pipeline. Since depth estimation requires the model to capture underlying scene geometry rather than superficial texture, strong and diverse augmentations are applied during training. These include horizontal flips, various types of blur, Gaussian noise, brightness and contrast variations, randomized color adjustments, hue and saturation changes, geometric transformations, and random resized cropping. These augmentations are implemented using Albumentations, and all transformations are applied in a paired manner to ensure that each RGB image and its depth map remain aligned. After augmentations, images are resized to a standard resolution of 224×224, normalized using ImageNet statistics, and converted into tensors suitable for GPU computation. Validation and test transformations are strictly limited to resizing and normalization to avoid introducing artificial distortions during evaluation. A custom PyTorch Dataset class is created to handle image loading with OpenCV, conversion to RGB or grayscale format, execution of augmentations, normalization of depth maps to the [0, 1] range, and construction of final tensors representing each training example.

DataLoaders are then constructed for the training, validation, and test sets, with batch sizes of sixty-four for training and validation and a smaller batch size for testing. This ensures efficient GPU utilization and stable throughput for the training process. The large batch size helps smooth gradient updates, while the shuffling mechanism used in the training loader introduces randomization that reduces overfitting and ensures better generalization.

For visualization and debugging, the project includes custom utilities to convert depth maps into colored images using an Inferno colormap, which provides a perceptually meaningful representation of geometric structure. Another utility merges RGB images with their ground truth depth maps for initial visual inspection. A more advanced visualization tool is provided later to display RGB images, ground truth depth, and predicted depth maps side by side. These visual inspection tools are essential in depth estimation projects because numerical metrics alone often fail to fully capture the perceptual quality and structural correctness of predictions.

The core model of the project is based on UNet++, implemented via the segmentation_models_pytorch library. UNet++ extends the classical U-Net architecture with dense skip-connections and nested decoder pathways that improve gradient flow and enable the model to capture fine structural details more effectively. The chosen encoder backbone is ResNeXt-50, pretrained on ImageNet, which provides a strong hierarchical representation of visual features. The encoder processes the RGB image and extracts multi-scale context, while the decoder progressively upsamples the features and reconstructs a dense depth map. The output of the model is a single-channel depth map with the same spatial resolution as the input. A training strategy is applied where the encoder is initially frozen for the

first two epochs, allowing the decoder to learn stable low-level depth predictions. After this initial phase, the entire network, including the encoder, is unfrozen for joint fine-tuning. This strategy stabilizes training and leads to better generalization.

The model is optimized using the AdamW optimizer with weight decay, which helps control overfitting. A OneCycleLR learning rate scheduler is integrated to accelerate early training, facilitate efficient exploration of the parameter space, and guide the model toward optimal convergence in later epochs. The loss function used for supervising the depth regression task is mean squared error, which encourages precise numerical alignment between predicted and ground truth depth values. Because depth maps are continuous and represent real-valued geometric measurements, the task is naturally formulated as a pixel-wise regression problem rather than a classification or segmentation problem.

To evaluate the performance of the model, two complementary metrics are used: MSE and SSIM. MSE provides a strict mathematical measure of the numeric difference between predictions and ground truth. SSIM, in contrast, evaluates perceptual and structural similarity by examining luminance, contrast, and geometric structure. SSIM is particularly important in depth estimation because even if the numeric values are slightly off, a model that reproduces accurate geometric structure is far more useful in applications such as SLAM and 3D reconstruction. The combination of SSIM and MSE offers a balanced and informative assessment of the model's performance.

Training proceeds for five epochs. During the first two epochs, only the decoder learns, allowing it to capture initial depth patterns without disrupting the pretrained encoder. From the third epoch onward, both encoder and decoder are trained together. Each epoch computes loss and evaluation metrics for both training and validation datasets. The best performing model is selected based on the highest validation SSIM value. Throughout training, automatic mixed precision is used to speed up computation and reduce memory consumption by employing half-precision floating point operations when possible. Gradient clipping ensures stability in backpropagation, especially during the periods when the model is learning multi-scale geometric representations.

Once training is complete, the best checkpoint is loaded and used to evaluate the model on the unseen test set. Predictions are generated for all test images, and final SSIM and MSE scores are computed. Additionally, a large visualization grid is produced to qualitatively assess the model’s ability to reproduce global depth gradients and local geometric details. The results indicate strong consistency between predicted and ground truth depth, with clearly preserved structures such as walls, furniture edges, floor boundaries, and depth gradients in indoor scenes. The validation SSIM reaches approximately 0.91, demonstrating high perceptual fidelity.

Overall, this project establishes a complete and reliable monocular depth estimation pipeline using a modern encoder–decoder architecture, strong data augmentation, mixed precision training, and robust evaluation metrics. It demonstrates the ability to convert RGB images into meaningful depth predictions and shows how deep learning can replicate geometric reasoning from visual cues. The resulting system is not only academically relevant but also directly applicable to robotics, AR/VR systems, real-time mapping, navigation, and 3D understanding tasks, making it an ideal model for perception components in advanced robotic systems.