DL4DS: Monocular Depth Estimation on Low Power Systems

Neel Gangrade, Rachita Singh

May 5, 2025

Abstract

This project aims to develop a lightweight, efficient model for monocular depth estimation that runs smoothly on mobile systems. By using techniques like quantization, pruning, and knowledge distillation, along with efficient architectures such as MobileNetV3 and depth-wise separable convolutions, we strive to achieve fast inference without compromising depth accuracy.

Introduction

Deep learning has significantly advanced computer vision, enabling applications like object detection, segmentation, and depth estimation. However, these advancements often come at a cost - state of the art models require extensive computational resources, making them difficult to deploy on real-world devices like smartphones, AI-powered PCs, and embedded systems. Many depth estimation models rely on large architectures that perform well in controlled environments but struggle with efficiency when running on edge devices with limited processing power and memory.

This project focuses on monocular depth estimation, which involves predicting depth from a single image rather than using stereo vision or LiDAR. Existing depth estimation models are often too resource-intensive for mobile deployment, and the challenge is to strike a balance between depth accuracy and computational efficiency. As part of Track 3 of the Low-Power Computer Vision Challenge (LPCVC), this project aims to develop an optimized monocular depth estimation model that can run efficiently on Snapdragon-powered devices while maintaining high accuracy. To achieve this, we will explore model compression techniques such as quantization and pruning, efficient neural network architectures, and depth-aware loss functions. The model will be benchmarked using industry-standard depth estimation metrics (MAE, RMSE, AbsRel, Chamfer Distance, and IoU) while ensuring an inference speed of 34ms per image on the target hardware.

By combining deep learning research with real-world hardware constraints, this project will contribute toward making monocular depth estimation more accessible and practical

for edge AI applications. The ability to efficiently estimate depth from a single image on mobile devices could enable breakthroughs in AR, robotics, and smart camera systems, making AI-driven vision technology more widely available.

Related Work

Monocular depth estimation has become an active field in computer vision in recent decades. Its fundamental task is to recover the corresponding depth information from a single RGB three-channel image captured by a monocular camera. Eigen et al. [2] first utilized the power of deep learning to estimate depth maps and get accurate results for depth estimation. , Eigen et al. [2] also present their scale-invariant loss function aiming to measure depth relations and accuracy, irrespective of the absolute global scale. Yang et al. [3] came up with Depth Anything V2, which changed the way monocular depth estimation models were trained on a large scale. They went away from real-world sensor data (LiDAR, stereo cameras) to synthetically generated images. Their three-stage pipeline—(1) training a DINOv2-Giant teacher on 595K synthetic images, (2) generating pseudo-labels for 62M unlabeled real images, and (3) distilling compact student models.

It is also worth noticing that some methods also take the inference time and model complexity into account, which makes them applicable to mobile devices like smartphones or embedded systems. Ranftl et al. [4] achieve great success dealing with the trade-off between accuracy and speed. Their Midas-v2.1 small model achieved 30 FPS on the iPhone 11. Efficient backbones like MobileNetV3 and NAS-optimized networks dominate edge deployments. Cai et al. [5] reduced energy consumption to 0.5W on Edge TPUs using MobileNetV3LMin, while Hornauer [6] validated FastDepth's viability on NVIDIA Jetson.

Reducing the model complexity and computation overhead while maintaining the performance has long been a popular topic. One feasible way is to simplify the model, e.g., pruning the redundant parameters [7], model quantization [8]. Knowledge distillation aims at transferring knowledge from a cumbersome teacher network to a compact student network. To balance accuracy and speed, Wang et al. [9] distilled knowledge from a teacher network (EfficientNet-B7) into a lightweight student (MobileNet), achieving 94.7% parameter reduction with only a 10% si-RMSE drop. Their student model runs at 20 FPS on Snapdragon 888 CPUs, making it viable for real-time mobile applications. There have been lots of efforts in designing efficient networks for mobile devices. These networks with extraordinary architectures pave the way for many real-time tasks on mobile devices. MobileNet-V1 [10] reduces the parameters by 31.9 times compared to VGG16.

Wang et al. [9] propose an interesting solution using knowledge distillation which has not been implemented at a large scale. We will not be reproducing the results of their architecture but will use similar ideas to build on top of Depth-Anything V2 [3], which has

produced SOTA results in recent years for monocular depth estimation.

Proposed Work

Our approach focuses on optimizing both inference speed (less than 34ms per image) and depth estimation accuracy. For optimizing inference speed, we plan to explore methods such as quantization and pruning.

Model quantization is a technique used to reduce model size and improve computational efficiency by converting model weights and activations into lower-precision formats (e.g., from 32-bit floating-point (FP32) to 16-bit (FP16) or 8-bit (INT8)). This helps in achieving faster inference speeds while keeping the accuracy drop minimal and enables real-time processing by decreasing latency. There are essentially two methods for quantization -

- Post training quantization (convert the trained model into a lower precision version without retraining)
- Quantization aware training (train the model while simulating quantization effects to retain accuracy)

Pruning is a method to speed up inference by removing less significant or redundant model parameters. It can be done by removing entire neurons, channels, or layers (structured pruning), thereby reducing the model size, or by removing individual weights with smaller magnitudes (unstructured pruning), making the model sparse. Along with this, downsizing the model or replacing certain architectural components can also help meet the speed constraint. Since we'll be working on the Depth-V2 model as our base, we'll start by using a lighter version of the network instead of the heavier Large or Giant versions. To further speed things up, we'll swap out standard convolutions for depthwise separable convolutions, which help reduce computation without sacrificing too much accuracy. We'll also explore lightweight backbone architectures like MobileNetV3, EfficientNet-Lite, and GhostNet, all designed to strike a balance between performance and efficiency.

Apart from these hardware-accelerated solutions, our proposed solution will also focus on knowledge distillation. It will be a lightweight framework for monocular depth estimation optimized for edge devices, combining knowledge distillation from Depth Anything V2 and hardware-centric optimizations. By distilling a 100M-parameter teacher model into a 25M-parameter student and applying 8-bit quantization.

We will be using a Teacher-Student architecture similar to Depth Anything V2 [3]. The implementation of knowledge distillation used by Wang et al. [9] will be implemented on top of the training methodology of [3] to train a monocular depth estimation model on low-resource systems. The Teacher-Student architecture will look similar to this:-

- Teacher Model: Encoder is Depth Anything V2-Large (ViT-Base backbone, 100M

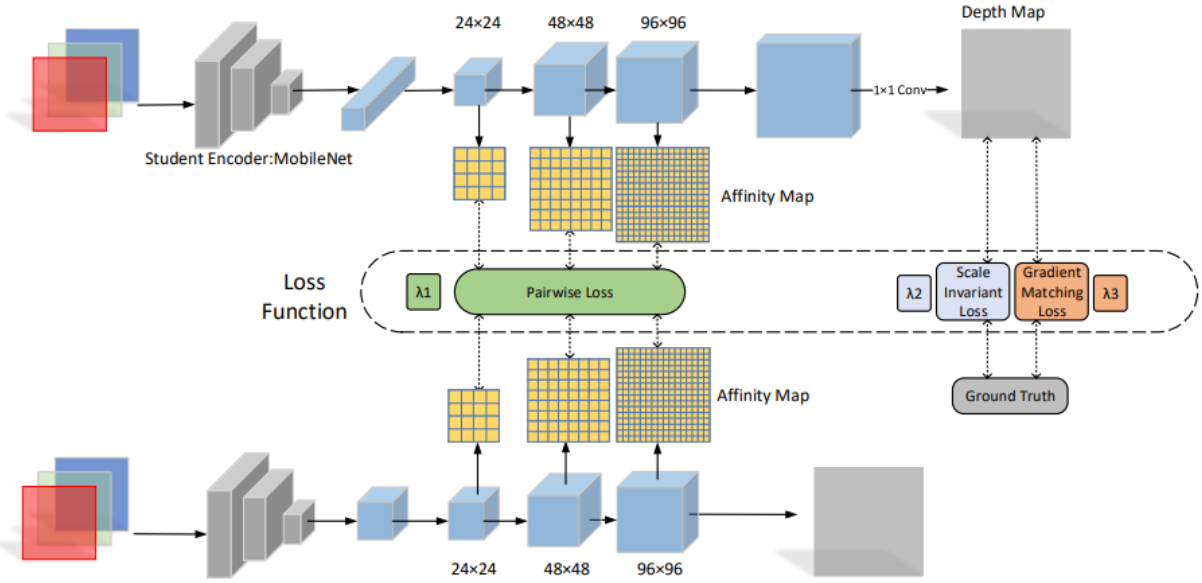

Figure 1: Teacher-Student Training Architecture. The above network represents the student model while the below stands for the teacher model [9].

parameters), pre-trained on 62M pseudo-labeled images, with custom decoder layers inspired from Wang et al. [9]

- Student Model: A typical encoder-decoder structure with skip connections. We adopt MobileNet [10] as the backbone to extract features, which uses depthwise and pointwise convolution to reduce the computation overhead and number of parameters.

During training, we adopt 3 different loss functions: pairwise loss, gradient matching loss, and scale invariant loss. Under the supervision of the pairwise loss between the feature maps of two networks, the representation ability of our teacher network is transferred to the student network via knowledge distillation. Scale-invariant loss and gradient matching loss are adopted to measure the discrepancy between the estimated depth map and the ground truth depth map. During inference, only the light-weight student network is needed.

Datasets

The NYU Depth V2 dataset provides a comprehensive collection of real-world indoor scenes, captured using a Microsoft Kinect sensor. It consists of 1,449 densely annotated RGB images and corresponding depth maps, spanning a variety of indoor environments such as bedrooms, offices, and kitchens [13]. The dataset is composed of pairs of RGB and Depth frames that have been synchronized and annotated with dense labels for every

image. In addition to the projected depth maps, we have included a set of preprocessed depth maps whose missing values have been filled in using the colorization scheme of Levin et al.

The evaluation dataset comprises 2,000 RGB images captured in various indoor and outdoor scenes under different lighting conditions (normal and low light) using a range of mobile devices. As detailed in the table below, the dataset includes 500 indoor images with normal light, 500 indoor images with low light, 500 outdoor images with normal light, and 500 outdoor images with low light. 10

Each image is accompanied by a corresponding Depth Map and Confidence Map. The Depth Map indicates the depth of each pixel in the RGB image, while the Confidence Map shows the confidence level of the corresponding pixel in the Depth Map. The following picture displays some samples (indoor normal light, indoor low light, outdoor low light). Each row contains the RGB image, Visualized Depth Map, and Visualized Confidence Map from left to right. The inherent challenges in capturing reliable depth information in cluttered, low-light indoor settings make NYU Depth V2 a valuable benchmark for evaluating depth estimation models.

Complementing the real-world data, the SYNTHIA dataset offers a large-scale synthetic environment for outdoor scene analysis [14]. This dataset comprises a broad range of urban and suburban scenarios, rendered using computer graphics techniques to generate photo-realistic RGB images along with precise ground-truth depth maps. The controlled nature of the synthetic data facilitates systematic evaluation and model development, enabling researchers to isolate and analyze specific challenges in depth estimation under varying conditions of illumination, occlusion, and scene geometry.

Methodology

Exploratory Data Analysis: We analyzed two critical datasets for monocular depth estimation: the NYU Depth V2 dataset and the SYNTHIA dataset. Our objective was to thoroughly characterize the data and derive insights that inform subsequent model training and development.

Analysis of the NYU Depth V2 Dataset

- The NYU Depth V2 dataset, comprising real-world indoor scenes, was first organized by indoor category (e.g., basement, bedroom) with each subfolder containing paired RGB images (in JPG format) and depth maps (in PNG format). Custom functions were used to traverse these subfolders, collecting paired RGB images (in JPEG format) and depth maps (in PNG format).

  Subsequently, we visualized (Figure 2) randomly selected RGB - depth pairs to verify

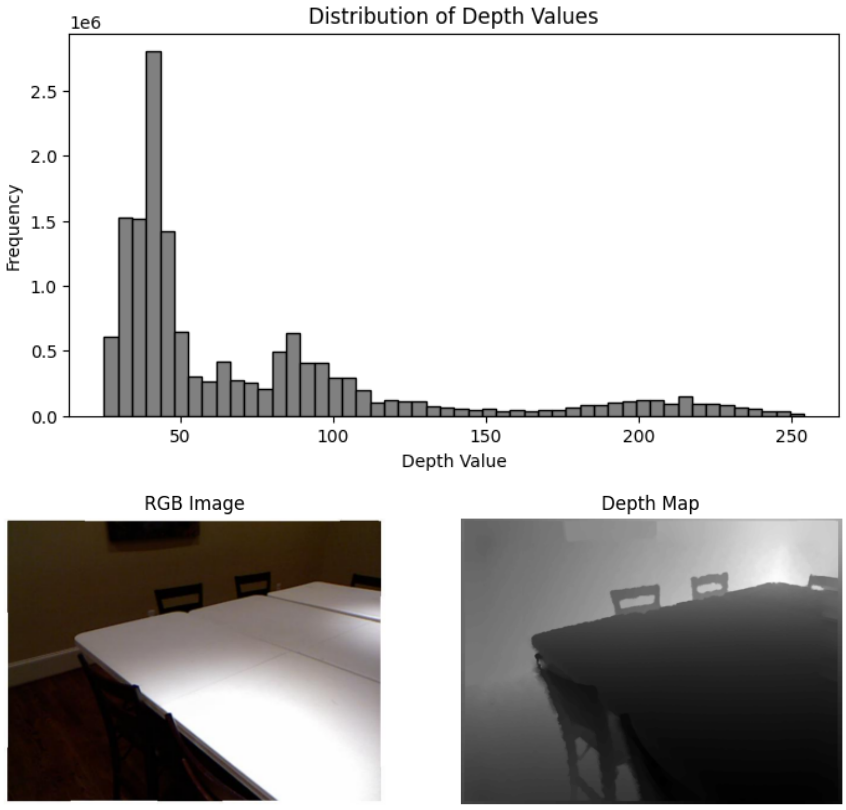

Figure 2: Top: Depth Distribution; Bottom: Input RGB and Ground Truth Depth Map

the integrity and alignment of the data. Detailed statistical analysis was performed on the depth maps, where we computed key metrics - including minimum, maximum, mean, median, and standard deviation and generated histograms to illustrate the overall depth distribution. Moreover, an outlier detection mechanism was employed to flag depth maps with significant deviations from the norm, which suggests that additional preprocessing steps, such as outlier filtering and normalization, may be necessary.

Analysis of the SYNTHIA Dataset

- Parallel analysis was conducted on the SYNTHIA dataset, a collection of synthetic outdoor scenes. Adjustments were made to account for its distinct organizational structure, ensuring accurate pairing of RGB images with their corresponding depth maps. Visual assessments of representative samples were conducted to compare synthetic scene characteristics with those observed in the NYU dataset.

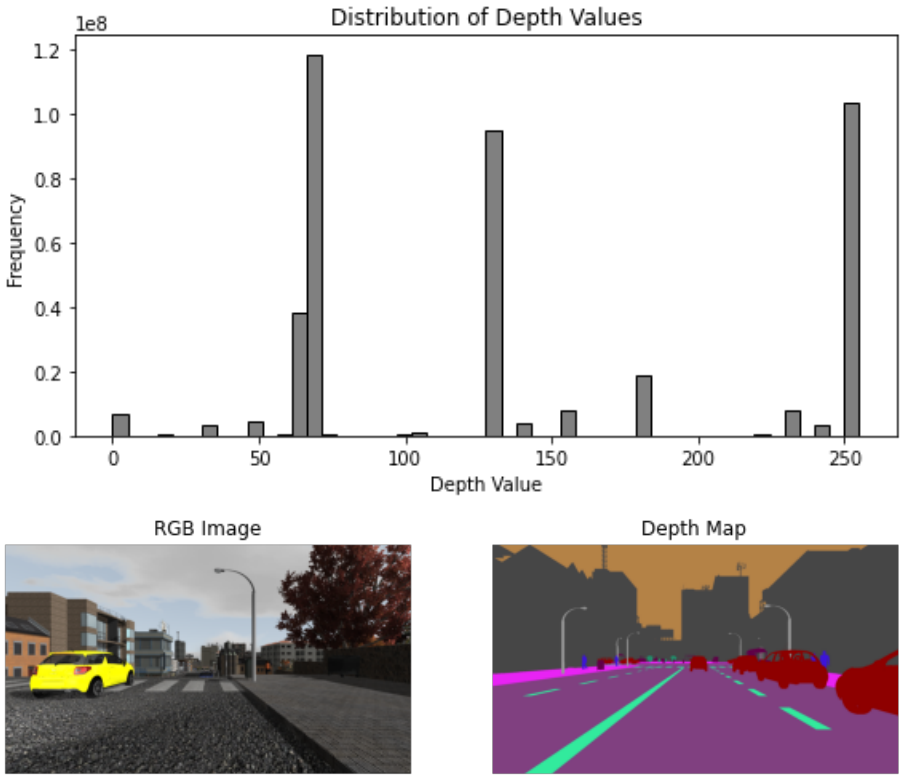

Figure 3: Top: Depth Distribution; Bottom: Input RGB and Ground Truth Depth Map

Statistical evaluation (Figure 3) of the depth values in the SYNTHIA dataset revealed a more uniform distribution compared to its real-world counterpart. However, the inherent differences in scene composition and lighting conditions between synthetic and real data underscore the need for specialized preprocessing strategies to address domain-specific nuances.

Our baseline model leverages a U-Net architecture with a pretrained ResNet-18 backbone for the encoder. This architecture balances computational efficiency with performance, making it suitable for establishing a foundation upon which more sophisticated models can be built.

The baseline model follows an encoder-decoder design with skip connections:

- Encoder: A pretrained ResNet-18 network serves as the feature extractor, leveraging transfer learning to benefit from features learned on large-scale image classification tasks. The encoder consists of five stages:

  - Initial layer: 7×7 convolution, batch normalization, ReLU activation
  - Four ResNet blocks with increasing channel dimensions (64, 128, 256, 512)
- Decoder: The decoder pathway reconstructs the spatial information and generates the depth map through a series of upsampling and convolutional blocks:
  - Four decoder blocks, each consisting of:
    * Transposed convolution for upsampling
    * Skip connection from the corresponding encoder level
    * Double convolutional block with batch normalization and ReLU
  - Final 1×1 convolution layer and sigmoid activation to produce depth values in [0,1]
- Skip Connections: Feature maps from the encoder are concatenated with the upsampled features in the decoder, helping to preserve fine-grained spatial information that might otherwise be lost during downsampling.
- Loss Function: The loss function is a combination of L1 and Gradient loss computed using the ground truths and the predictions.

During our experimentation, the model was trained for 50 epochs (Figure 4) using Adam (lr=1e-4). Early overfitting was observed due to the minimal changes in the validation loss, likely due to the pretrained ResNet-18 encoder's transferred knowledge. This baseline deliberately excludes hyperparameter optimization and data augmentation, techniques that could potentially enhance generalization performance and provide clearer benchmarks for future improvements.

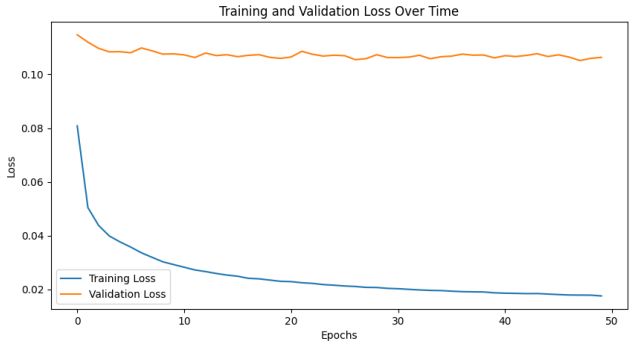

Figure 4: Train and Validation loss over 50 epochs

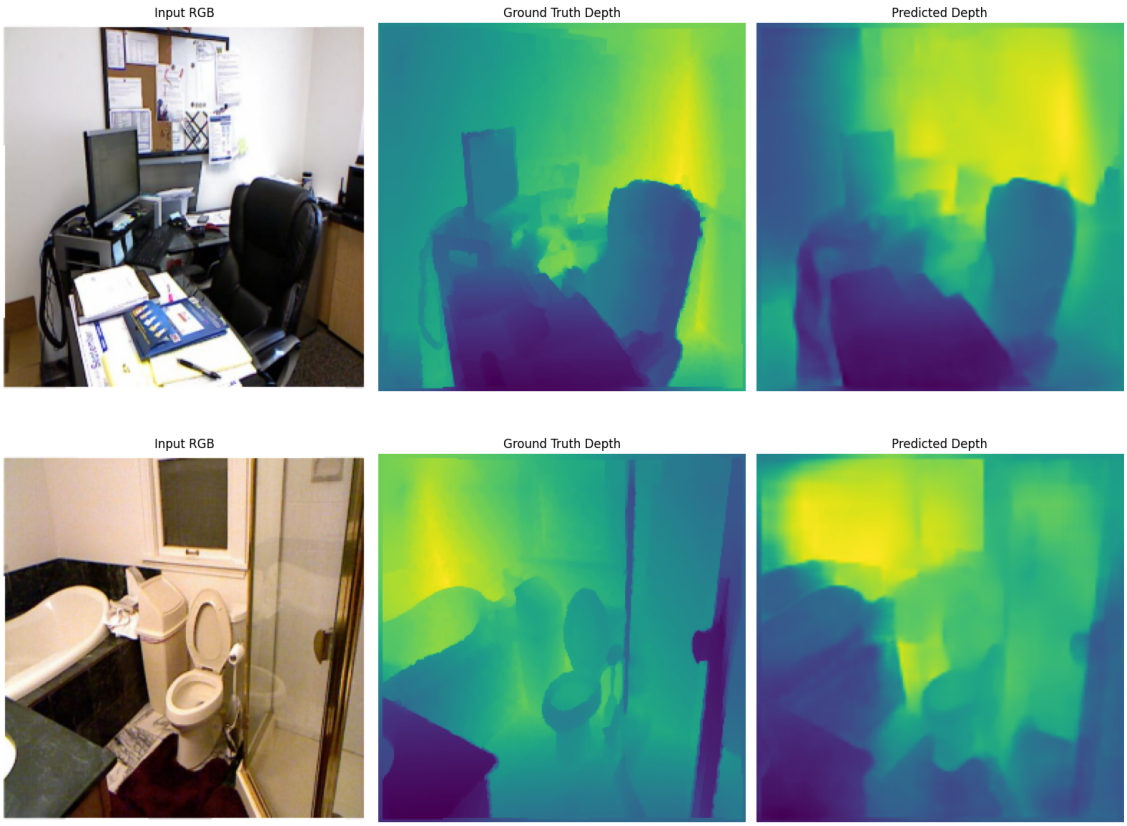

Figure 5: Results from the baseline model (U-Net architecture) after 50 epochs (Left to Right: Input RGB, Ground Truth, Predicted Depth)

The results above (Figure 5) demonstrate that despite overfitting tendencies, the model captures fundamental spatial relationships on unseen validation data. While imperfect, the predicted depth maps indicate successful learning of basic monocular depth cues, establishing a foundation for more sophisticated approaches. The ground truth and predicted depth maps in these results have been color graded using a color map for visualization purposes.

Building upon our initial model and architectural proposal, we have implemented a comprehensive approach that balances depth estimation accuracy with computational efficiency. The use of three complex loss functions in our initial proposal highly hindered our training process and prevented the model from converging. Our final model architecture leverages knowledge distillation alongside hardware-oriented optimizations to achieve real-time performance on low-power systems.

We implemented a knowledge distillation framework for efficient monocular depth estimation using a Teacher-Student paradigm. This approach enables deployment of accurate depth estimation models on resource-constrained devices while maintaining high-quality predictions.

- Teacher Model: Leverages the pre-trained Depth Anything [3] Large architecture, accessed via the HuggingFace transformers library. This model provides high-fidelity depth maps with strong generalization capabilities across diverse scenes but is computationally intensive, making it unsuitable for real-time applications on mobile devices.
- Student Model: The student model employs an efficient encoder-decoder architecture:
  - Encoder: MobileNetV3-Small as the feature extraction backbone, which employs depthwise separable convolutions to minimize computational complexity while maintaining representational capacity. The encoder progressively reduces spatial dimensions while increasing feature depth, capturing hierarchical representations at multiple scales.
  - Decoder: Decoder comprises three sequential blocks, each containing:
    - Depthwise separable convolutions to maintain efficiency.
    - Batch normalization and ReLU6 activation functions.
    - Bilinear upsampling (scale factor=2) for feature map resolution recovery.
  - Skip Connections: Skip connections between encoder and decoder stages to preserve fine-grained spatial information. Features from encoder layers 3, 8, and 11 are projected via 1×1 convolutions to match the corresponding decoder feature dimensions, then integrated into the decoding process.
- Loss Functions:

– Scale-Invariant Loss: Addresses the scale ambiguity inherent in monocular depth estimation by computing log-space differences between predictions and ground truth. It focuses on relative relationships between depth values rather than absolute depth accuracy, which is particularly valuable when ground truth depth may have different scaling than predictions.

– L1 Loss: Calculates the mean absolute error between predicted depth and ground truth depth. It enforces direct depth value correspondence while being robust to outliers.

– Knowledge Distillation Loss: Implemented as an L2 loss between the student model's predictions and the teacher model's predictions, facilitating the transfer of knowledge from the teacher to the student.

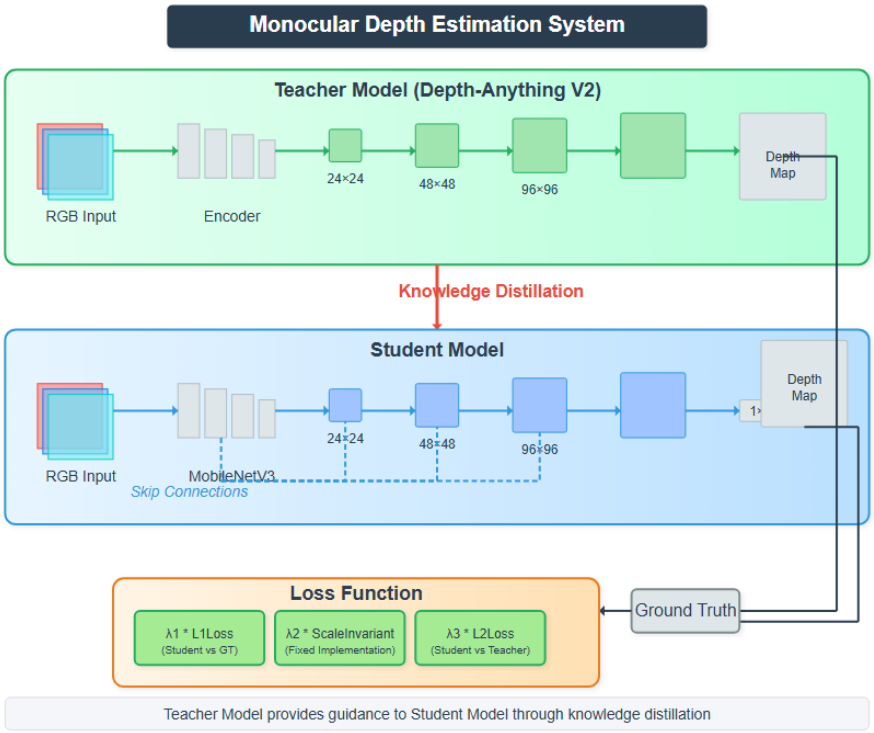

Figure 6: Teacher-Student Model Architecture with MobileNetV3 as Student Backbone (Image Generate by Claude)

To further enhance the model's efficiency for deployment on low-power systems, we implemented more optimization techniques:

- Quantization: We applied both post-training quantization and quantization-aware training to convert model weights and activations from 32-bit floating-point (FP32) to 8-bit integer (INT8) format. Specifically, we used PyTorch's 'quantize_dynamic' on Convolution and Linear layers, which resulted in a significant reduction in model size and CPU inference latency.
- Pruning: We implemented unstructured weight-based pruning during our training process to make the model sparse. This approach removes individual weights with smaller magnitudes while preserving the overall network structure, effectively reducing computational demands during inference.

Evaluation

Model Input: During model evaluation, only the RGB images will be fed into the submitted model. The Depth Map and Confidence Map will not be accessible. Therefore, the submitted model should only take one image as input. All RGB images are in VGA resolution (640x480), so the input tensor shape will be (Batch, Channels, Height, Width) = (Batch, 3, 480, 640) in the PyTorch format. All images will be loaded as RGB channel order and ranging from [0, 255] in float. No input normalization will be applied, so each submitted model should include normalization operations, such as (image-mean)/std or image/255, at the beginning of the model if needed.

Model Output: The model is expected to predict a relative depth (ranging from 0 to 1, from near to far) based on its input. The model should produce a single output with one channel, and the expected output tensor shape is (Batch, 1, Height, Width) = (Batch, 1, 480, 640).

To rigorously assess our model's performance, we conducted comprehensive evaluations focusing on both computational efficiency and depth estimation accuracy. Our evaluation framework was designed to simulate real-world deployment scenarios on low-power systems while maintaining high standards for depth prediction quality.

We evaluated our model's efficiency using the following metrics:

- Inference Speed: Our final optimized model achieves approximately 22 frames per second on a low-power CPU. This performance level enables real-time applications in resource-constrained environments such as mobile devices and embedded systems.
- Model Size: Through the application of quantization and pruning techniques, we achieved a substantial reduction in model size compared to the original teacher model. The INT8 quantization reduced the memory footprint by approximately 75% relative to the FP32 model, while maintaining acceptable depth estimation accuracy.

To evaluate the quality of depth predictions, we utilized standard metrics widely adopted in the monocular depth estimation literature:

- Mean Absolute Error (MAE): Our model achieves an MAE of approximately 10 on the test dataset. The MAE provides a direct measure of the average prediction error across all pixels, with lower values indicating better performance.
- Root Mean Squared Error (RMSE): We recorded an RMSE of approximately 20, which penalizes larger errors more heavily than smaller ones. This metric is particularly valuable for assessing the model's handling of outliers and difficult regions in the depth map.
- Absolute Relative Error (AbsRel): Our model demonstrates an AbsRel of approximately 0.5, indicating relatively small errors between predicted depth and ground truth when considered in proportion to the true depth values.

Timeline

- Week 1: Research & Setup
  - Finalize project scope and evaluation criteria.
  - Study Depth-Anything V2, knowledge distillation, and Snapdragon AI Hub.
- Week 2: Dataset Preparation
  - Download and preprocess KITTI, NYU Depth V2, DIW, MegaDepth datasets.
  - Implement data augmentation (brightness, contrast, noise, cropping).
- Weeks 3-4: Baseline Model Training and Architectural Refinements
  - Train Depth-Anything V2-Small as a baseline.
  - Experiment with MobileNetV3, EfficientNet-Lite, and GhostNet backbones.
- Weeks 5-6: Model Optimization - Quantization & Pruning
- Week 7: Knowledge Distillation
  - Train a Teacher-Student model (Depth-Anything V2-Large $\rightarrow$ Small).
  - Use scale-invariant loss & gradient matching loss for learning.
- Weeks 8-10: Fine-Tuning, Confidence Map Filtering, and Evaluation
  - Optimize hyperparameters and depth estimation metrics.
  - Implement confidence map-based filtering for better accuracy.

- Week 11: Finalizing results, project report preparation

Conclusion

This project aims to develop a fast, efficient, and accurate monocular depth estimation model optimized for Snapdragon-powered edge devices. By incorporating quantization, pruning, and knowledge distillation alongside lightweight architectures like MobileNetV3 and depthwise separable convolutions, our model will strike a balance between speed (less than 34ms per image) and depth accuracy. Through training on diverse depth datasets and evaluation with industry-standard metrics, we aim to ensure robust performance across various real-world conditions. More than just optimizing a model, this project has the potential to advance AI-driven depth estimation for real-world applications. From augmented reality and robotics to smart camera systems and autonomous navigation, enabling high-quality depth perception on low-power devices could unlock new possibilities for mobile AI. By bridging the gap between cutting-edge research and real-world deployment, this work contributes to the future of efficient, scalable, and accessible computer vision solutions.

References

[1] Lee, Jae-Han, and Chang-Su Kim. "Monocular depth estimation using relative depth maps." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.

[2] Eigen, David, Christian Puhrsch, and Rob Fergus. "Depth map prediction from a single image using a multi-scale deep network." Advances in neural information processing systems 27 (2014).

[3] Yang, Lihe, et al. "Depth anything: Unleashing the power of large-scale unlabeled data." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024.

[4] Ranftl, René, et al. "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer." IEEE transactions on pattern analysis and machine intelligence 44.3 (2020): 1623-1637.

[5] Cai, Hong, et al. "Real-Time and Accurate Self-Supervised Monocular Depth Estimation on Mobile Device." NeurIPS 2021 Competitions and Demonstrations Track. PMLR, 2022.

[6] Hornauer, J., Nalpantidis, L., Belagiannis, V. (2021). Visual domain adaptation for monocular depth estimation on resource-constrained hardware. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 954-962).

[7] He, Yihui, Xiangyu Zhang, and Jian Sun. "Channel pruning for accelerating very deep neural networks." Proceedings of the IEEE international conference on computer vision. 2017.

[8] Polino, Antonio, Razvan Pascanu, and Dan Alistarh. "Model compression via distillation and quantization." arXiv preprint arXiv:1802.05668 (2018).

[9] Wang, Yiran, et al. "Knowledge distillation for fast and accurate monocular depth estimation on mobile devices." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

[10] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

[11] Uhrig, Jonas, et al. "Sparsity invariant cnns." 2017 international conference on 3D Vision (3DV). IEEE, 2017.

[12] KITTI Depth Prediction Evaluation Dataset

[13] NYU Depth Dataset V2

[14] Synthia Dataset