Aishwarya Muttineni (50537179)
Chandu Challagonda (50547890)

Autofocus by Depth Estimation

Overview

The goal of the Auto Focus via Depth Estimation project is to replicate camera auto-focus functionality by using deep learning techniques to predict depth information from RGB photos and emulate depth-of-field effects. The project creates precise depth maps with pixel intensities corresponding to object distances using the ResNet-18 and ResNet-50 designs. A dataset of RGB images and the related depth maps is used to train and assess the models. ResNet-50 performs better in terms of loss, Mean Squared Error (MSE), and Peak Signal-to-Noise Ratio (PSNR). The depth maps are refined using post-processing methods including Gaussian blurring and median, and then depth regions are segmented using K-Means clustering. By selectively blurring areas according to depth, these clusters produce focus-enhanced photos that mimic the auto-focus effect. The research demonstrates how deep learning can be used to improve and automate depth estimates for use in computer vision, photography, and other fields.

Throughout the project's development, both contributors (Aishwarya and Chandu) contributed equally in terms of duties and work. Together, we worked on the coding portion, making sure that the models, preprocessing pipelines, and post-processing methods were implemented. We also each made an equal contribution to the report's documentation, which included draughting the literature review, outlining the methods, and evaluating the findings. We were able to successfully utilize each other's strengths to accomplish the project's objectives since the workload was divided in a balanced way.

Prior Work

The area of depth estimation from images has been thoroughly studied, with many methods concentrating on deep learning, hardware-based sensors, or stereo vision. Among the noteworthy earlier works are:

- **Eigen et al. (2014): "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network" [1]**
    - This groundbreaking study pioneered the field of learning-based depth estimation by introducing CNNs for single-image depth prediction.
    - Limitation: Post-processing methods to enhance depth maps for practical uses like auto-focus were absent.

- **Laina et al. (2016): "Deeper Depth Prediction with Fully Convolutional Residual Networks" [2]**

    - For depth prediction, fully convolutional residual networks were suggested, greatly increasing scalability and accuracy.
    - Limitation: Focus simulation and other depth-based applications were not investigated.

- **Godard et al. (2017): "Unsupervised Monocular Depth Estimation with Left-Right Consistency" [3]**
  - For depth estimation using stereo pictures, the emphasis was on unsupervised learning, which reduced the need for labelled datasets.
  - Reliance on stereo vision and lack of depth maps for artistic effects such as changing focus are limitations.

- **Kendall et al. (2017): "End-to-End Learning of Geometry and Context for Monocular Depth Estimation" [4]**
  - CNNs and geometric constraints were used to provide precise monocular depth estimate.
  - Limitation: Accuracy was prioritised over downstream uses like depth-based picture modifications.

- **Yin et al. (2019): "Enforcing Geometric Constraints for Monocular Depth Estimation" [5]**
  - By implementing geometric priors, depth estimation was enhanced and robust performance was attained in intricate scenarios.
  - Limitation: Does not investigate applications such as focus simulation or downstream processing.

Novelty of this Project

The Auto Focus by Depth estimate project distinguishes itself from other efforts by highlighting a useful application of depth maps in producing focus-enhanced photos, whereas earlier work has mostly concentrated on increasing depth estimate accuracy. What's new is:

1. **Post-Processing Refinements**:

   - In contrast to earlier research, our effort refines depth maps using a combination of Gaussian and median blurring techniques, improving their usefulness for applications further down the line.

2. **K-Means Clustering for Depth Segmentation**:

   - A new layer of depth comprehension is introduced by using clustering to divide depth maps into distinct regions, allowing for accurate modulation of depth-based focus.

3. **Reordering Depth Classes**:

   - A step not covered in earlier publications, the reordering of depth classes according to intensity levels guarantees constant focus effects across different situations.

4. **Integration with Focus Simulation**:

- The research imitates camera auto-focus by directly using depth maps to produce focus-enhanced photos. No previous literature has examined this practical application.

5. **Architecture Comparison**:

- The study offers a clear roadmap for future implementations by comparing ResNet-18 and ResNet-50 for depth prediction, shedding light on the trade-offs between computing efficiency and prediction quality.

This study demonstrates the adaptability of deep learning in real-world situations by bridging the gap between theoretical depth prediction and real-world applications by integrating depth estimation, post-processing, and focus simulation into a single pipeline.

Approach

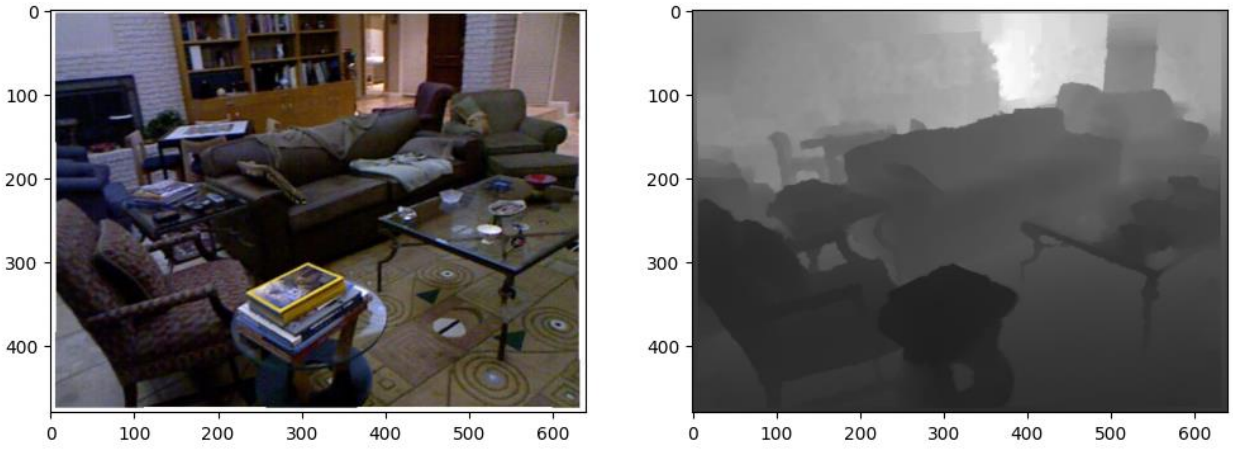
Every stage of the procedure is explained in great depth, supported by relevant images and insights into its importance.

Step 1: Read the Dataset

What Happens: The RGB images and matching depth maps are included in the loaded dataset. A sample pair of images and labels is displayed.

- A pandas DataFrame with 50,688 image-label pairs is produced.
- Simple checks make sure no entries are missing or incorrect.

Image 1: Processed Image-Label Pairs



This displays the dataset's raw input, with the depth map on the right and the original RGB image on the left. Lighter regions on the greyscale depth map signify objects that are closer, while darker regions suggest objects that are farther away. This was produced after an example

image-label pair was visualised by loading and reading the dataset.

Step 2: Initialize the Dataset and DataLoader

What Happens:

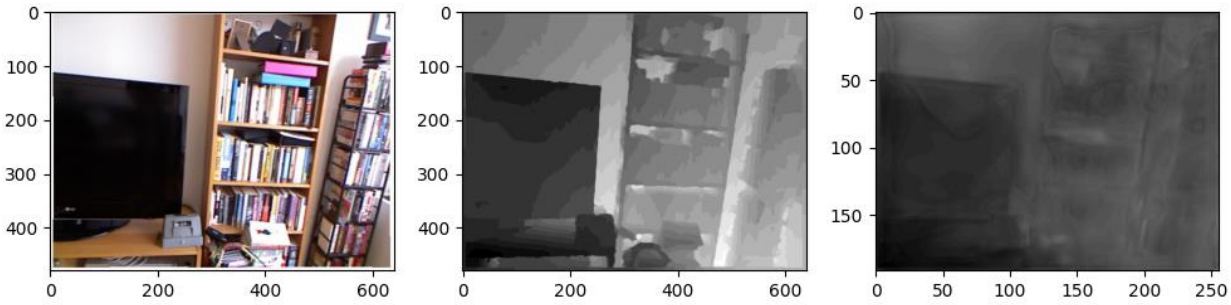
- To manage data transformations, such as extracting, normalising, and reshaping images and labels, a dataset class (NYU2_dataset) is developed.
- To iterate over the dataset, a DataLoader is initialised.

Step 3: Train the ResNet-18 Model

What Happens:

- The dataset is used to train the ResNet-18 model over five epochs. Plotting is done for loss and metrics like MSE and PSNR.
- Following training, the trained model is used to produce predictions for test images.
    - The model is trained for 5 epochs.
    - Loss function: BCELoss
    - Optimizer: Adam with a learning rate scheduler.

Image 2: Depth Prediction from ResNet-18



- The three panels here include:
    1. Left: The original RGB input image.
    2. Middle: A greyscale representation of the depth map that the ResNet-18 model predicted.
    3. Right: a smoothed or normalised version of the depth map after processing.

    Following the execution of the ResNet-18 model, this image was produced as part of the "Preview" function.

Step 4: Train the ResNet-50 Model

What Happens:

- With comparable setups, the ResNet-50 model is trained across ten epochs. Following training, predictions are produced for test photographs.

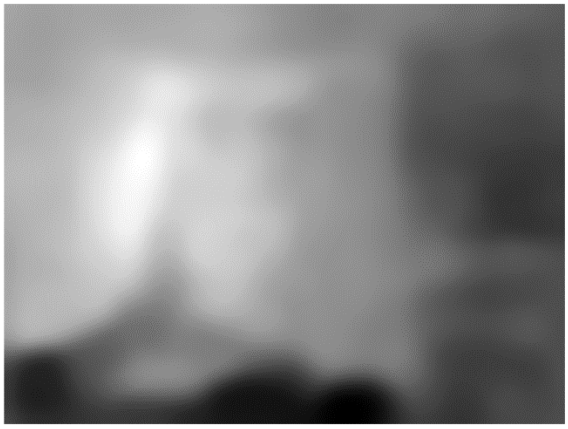
Image 3: Grayscale image

- The RGB source image used as the ResNet-50 model's input is displayed in the left panel.
- The ResNet-50 model's anticipated depth map is shown in greyscale in the right panel. Compared to the prior ResNet-18 forecast, the depth map might contain more precise data.

Step 5: Perform Feature Extraction with Blurring

- **What Happens**:
  - The anticipated depth map is subjected to Gaussian and median blurs in order to improve the regions of interest and minimise noise. Smoother depth representations are produced by these blurring approaches.

Image 4: Smoothed Depth Map



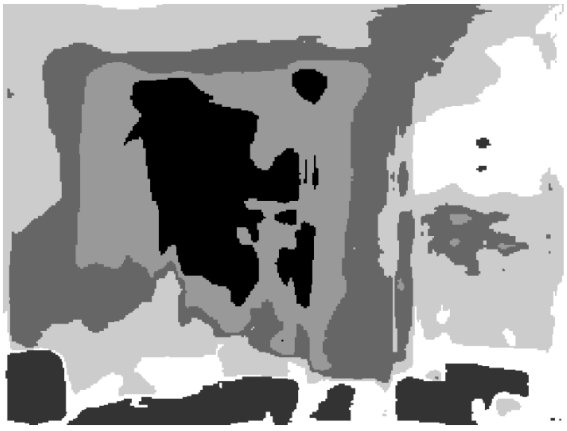
- After applying median blurring, which softens sharp edges and lowers noise in the depth map, the expected depth map is shown in the left panel.
- The depth map is displayed in the right panel following the application of Gaussian blurring with different kernel sizes to improve smoothness even more.

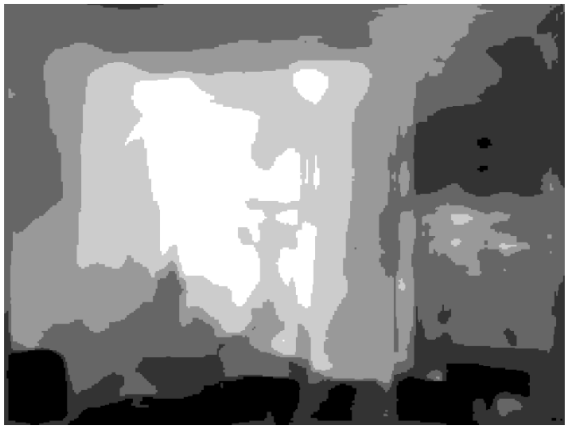
Step 6: Apply K-Means Clustering

What Happens:

- To divide the image into discrete depth-based sections, K-Means clustering is used to both the predicted and smoothed depth maps.

Image 5: Clustered Depth Map





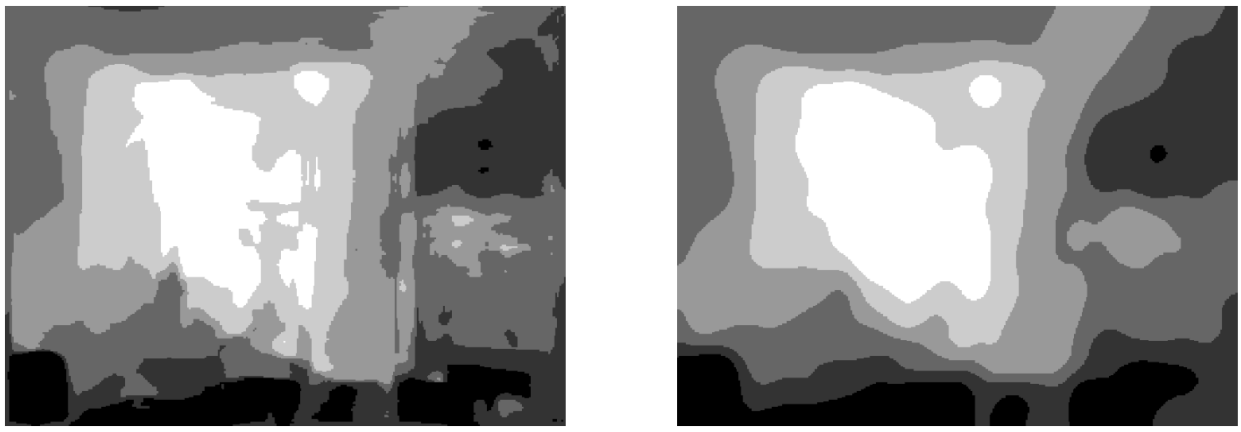
- K-Means clustering was applied to the original depth map features to produce the clustered depth map shown in the left panel. A different depth level is represented by each cluster.

- The smoothed (blurred) image's clustered depth map is shown in the right panel. Prior to clustering, blurring aids in the formation of more uniform depth zones.

Step 7: Reorder K-Means Classes

- **What Happens:**
  - To guarantee a constant depth-ordering from close to distant objects, K-Means classes are rearranged according to average intensity.

Image 6: Reordered Clusters



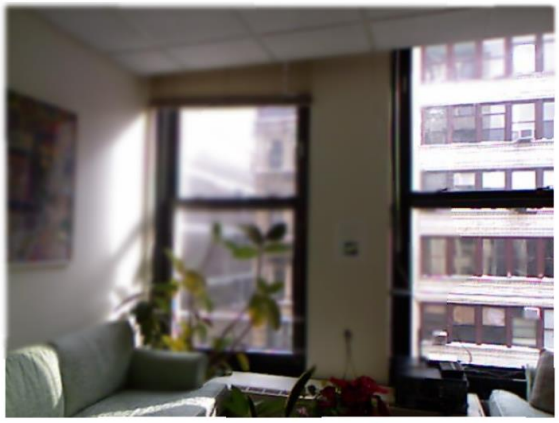
- The original image's rearranged K-Means clustered depth map, with the clusters arranged according to their intensity values, is displayed in the left panel.
- It shows the smoothed image's rearranged clustered depth map in the right panel.

Step 8: Generate Focus-Enhanced Image

What Happens:

- The rearranged K-Means clusters are used to selectively apply blurs to areas according to their depth, resulting in a focus-enhanced image. This simulates the effect of auto-focus.

Image 7: Focused Depth Map



- The left panel is the original RGB image.
- The focus-enhanced image, which simulates an auto-focus effect by blurring different regions according to their depth, is displayed in the right panel.

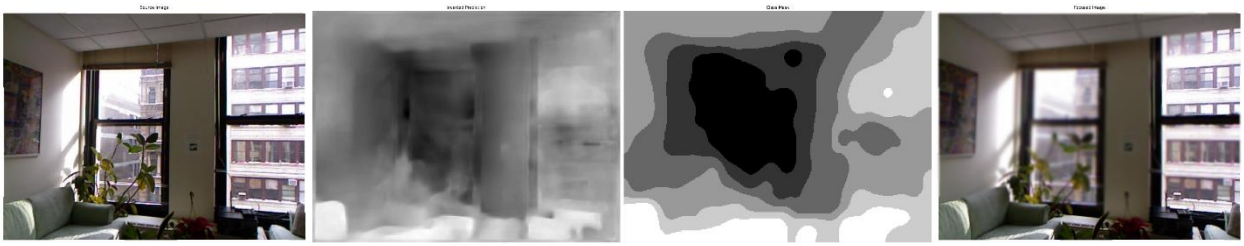
Step 9: Final Visualization

What Happens:

- A thorough visual representation of the entire procedure is produced:

1. The source RGB image.
2. The inverted depth prediction.
3. The class mask generated by K-Means.
4. The final focus-enhanced image.

Image 8: Comprehensive Visualization



- The first panel is the source image.
- The second panel is the inverted depth prediction.
- The third panel is the class mask showing clustered depth levels.
- The fourth panel is the focus-enhanced image.

Aishwarya Muttineni (50537179)
Chandu Challagonda (50547890)

What aspects of the algorithms have you coded on your own?

1. **Custom Data Preprocessing Pipeline**:
   - We built the full data preprocessing pipeline ourselves, including image and depth label reading, scaling, normalisation, and reshaping.
   - **Relevance**: This preprocessing guaranteed compatibility with the ResNet-18 and ResNet-50 models, optimised the input format, and increased training efficiency.
   - **Contribution**: This phase was crucial for maintaining the integrity of the data pipeline and making sure that depth maps were appropriately associated with input images.
2. **Post-Processing of Depth Maps**:
   - To decrease noise and smooth the anticipated depth maps, we independently used approaches including median and Gaussian blurring.
   - **Relevance**: Smoothing depth maps improved visual quality and enabled better depth-based segmentation for grouping and focus formation.
   - **Contribution**: This update directly improved the project's performance in generating realistic images with enhanced focus.
3. **K-Means Clustering and Depth Class Reordering**:
   - To achieve consistent depth ordering, we built custom code to apply K-Means clustering to the depth maps and rank the clusters depending on intensity values.
   - **Relevance**: These procedures supplied the structured depth segmentation required for implementing depth-based blurring in the focus simulation stage.
   - **Contribution**: This clustering was critical in enabling the formation of segmented depth areas for focus generation.
4. **Focus Simulation Algorithm**:
   - We independently created the focus simulation technique, which applied different blurs to regions based on their depth classes, replicating a camera's depth-of-field effect.
   - **Relevance**: This was the project's main feature, translating depth estimates into practical applications by producing visually pleasing focus-enhanced images.
   - **Contribution**: This self-coded component is the foundation of the project's uniqueness and practical use.

What aspects of the algorithms have you used from online resources?

1. **Model Architectures (ResNet-18 and ResNet-50)**:
   - We used pre-trained ResNet-18 and ResNet-50 architectures from PyTorch for depth prediction [6].
   - **Relevance**: These well-tested designs provided dependable frameworks for creating high-quality depth maps.
2. **Loss Functions and Metrics**:
   - To assess model performance, we employed PyTorch's standard implementations of Mean Squared Error (MSE) loss and Peak Signal-to-Noise Ratio (PSNR) [6].
   - **Relevance**: These metrics and loss functions made it possible to optimise the models robustly and evaluate them objectively.
3. **Data Augmentation and Transformations**:

- PyTorch's torchvision.transforms library was used for resizing, normalization, and augmentations like flipping [7].
- **Relevance**: These changes enhanced the dataset's generalizability and variety.

4. **K-Means Clustering Implementation**:
   - We made use of the Scikit-learn library's K-Means clustering implementation [8].
   - **Relevance**: By dividing the depth maps into discrete areas, the clustering technique made depth-based focus simulation possible.
5. **Conceptual Guidance**:
   - Shir Gur's paper "UnsupervisedDepthFromFocus" [9] provided us the conceptual understanding of how to combine depth prediction with focus simulation.
   - **Relevance**: High-level guidelines for connecting depth maps to focus effects were supplied by this study, although each implementation in this project was created separately.

Experimental Protocol

1. **Dataset Summary:**
   - The NYU Depth v2 dataset comprises 50,688 depth maps and related RGB pictures.
   - Input images resized to 480x640 pixels.
2. **Evaluation Metrics:**
   - Loss: Measures prediction error.
   - MSE: Quantifies pixel-level accuracy.
   - PSNR: Evaluates depth map quality.
3. **Compute Resources:**

   - Effective training and inference are possible with the Google Colab environment and an A100 GPU.

Results

Metrics:

Metric	ResNet-18	ResNet-50
Loss	**0.5496**	**0.5486**
MSE	**0.0010**	**0.0006**
PSNR	**32.9079**	**34.5172**

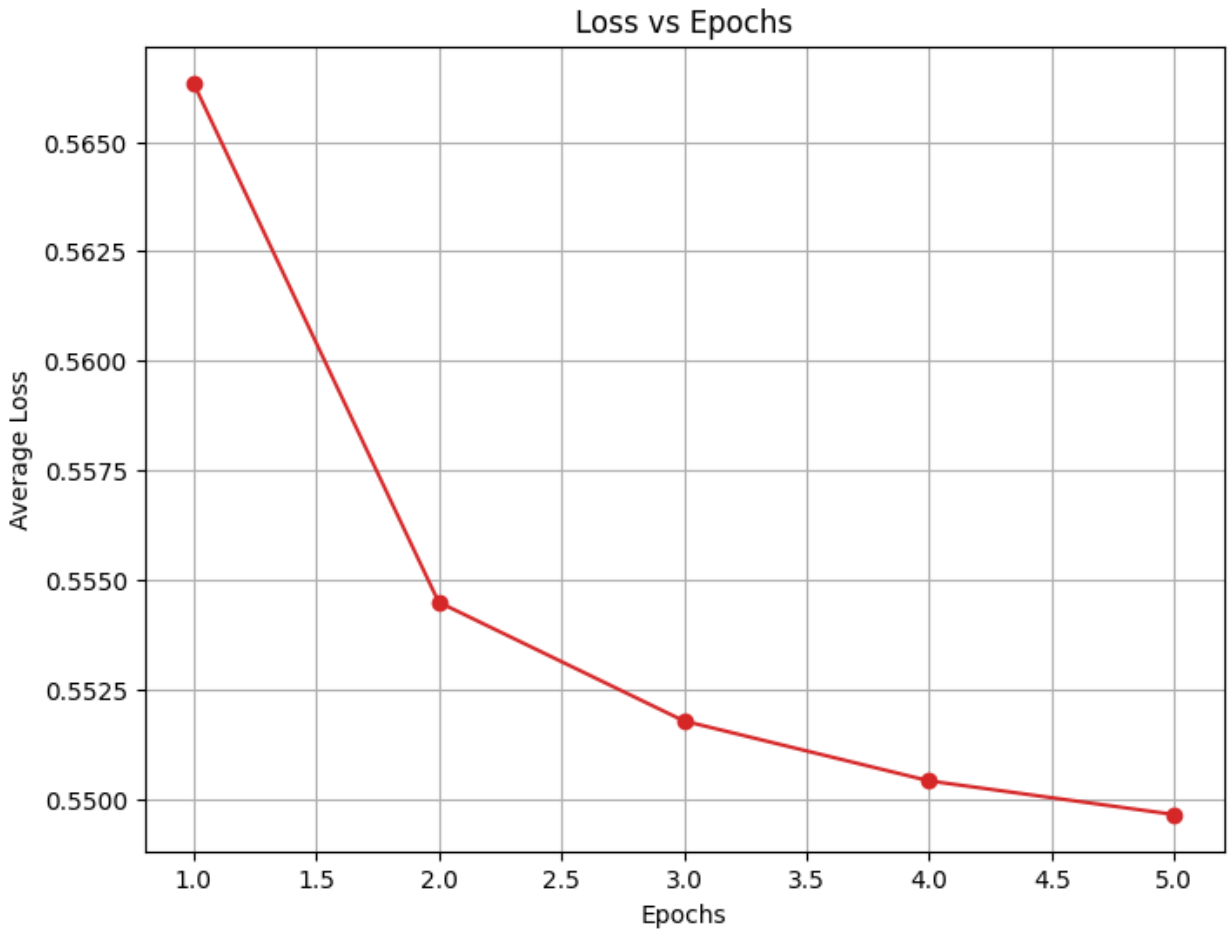
Explanation: According to the findings, ResNet-50 performed better than ResNet-18 across all evaluation metrics. With a little reduced loss (0.5487 as opposed to 0.5497), ResNet-50 demonstrated superior training optimization. The projected depth maps appear to be closer to the ground truth, as seen by its much reduced Mean Squared Error (MSE) of 0.0006 when compared to ResNet-18's 0.0011 value. A higher Peak Signal-to-Noise Ratio (PSNR) of 34.3302 was also attained by ResNet-50, indicating better depth map reconstruction quality with fewer aberrations.

These gains are attributable to ResNet-50's deeper design, which enhances overall performance in depth estimation tasks by enabling it to extract finer details and more complicated features from the data.

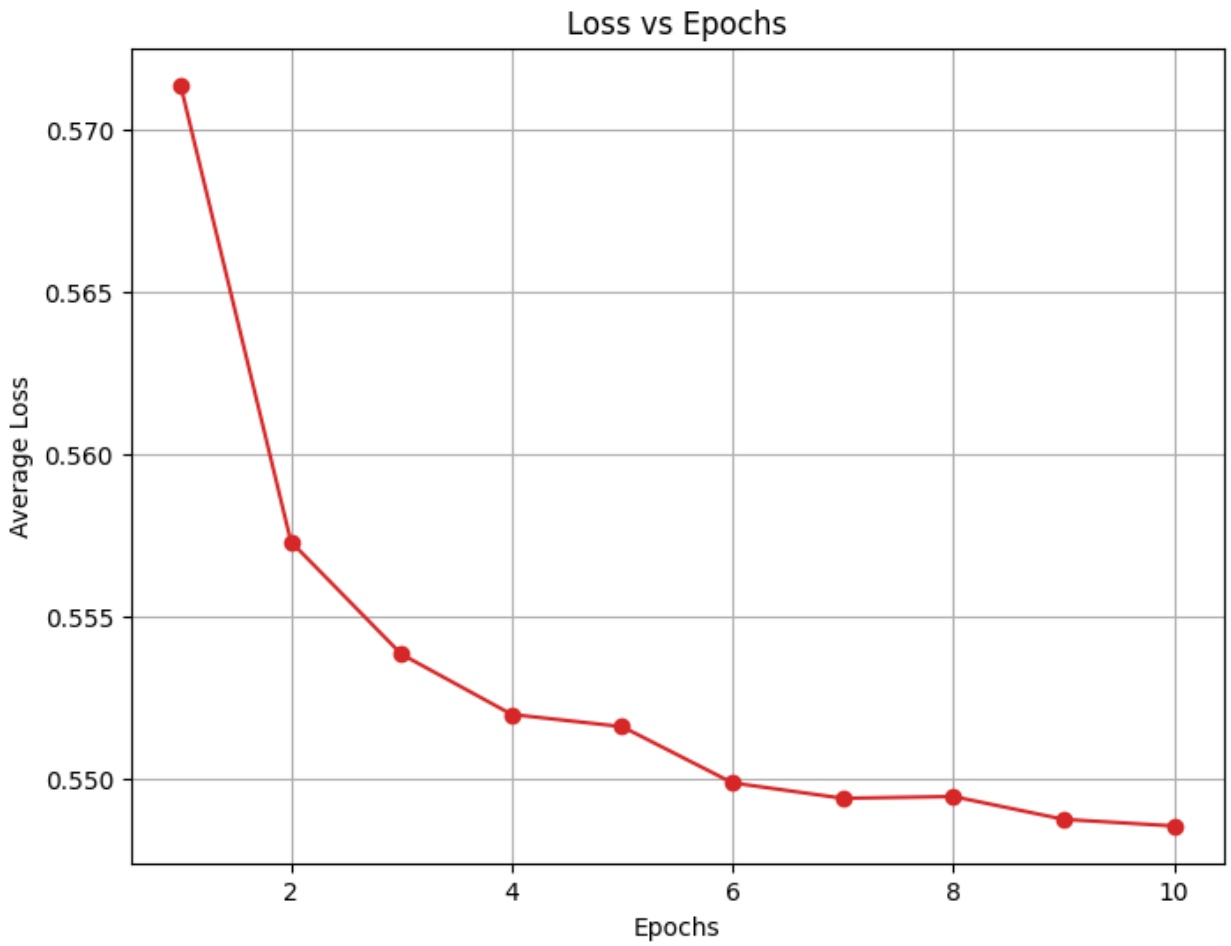
Plots:

Loss vs Epoch: The training process's convergence behaviour is demonstrated. ResNet-50 has superior learning as evidenced by its lower final loss values when compared to ResNet-18.

- **ResNet-18**

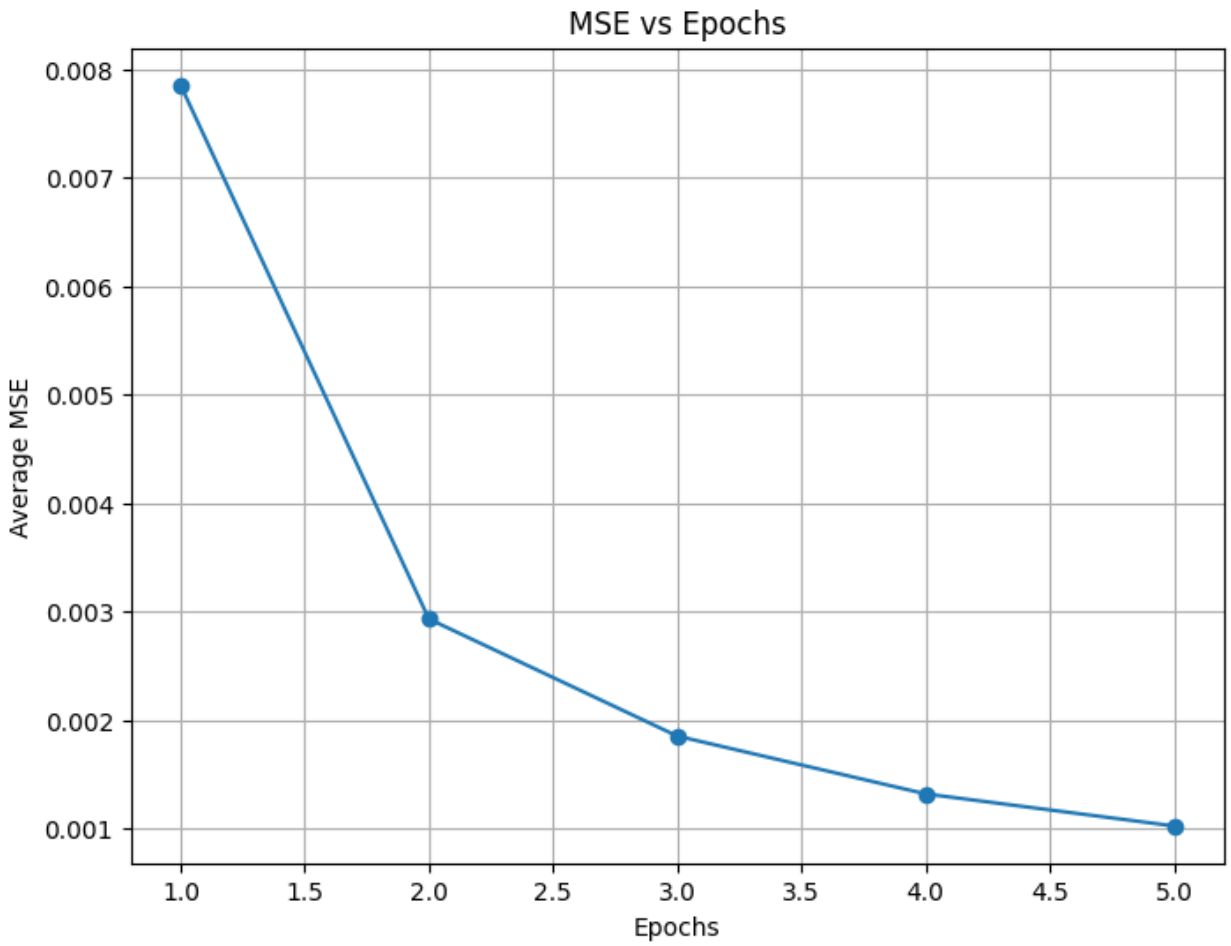


- **ResNet-50**

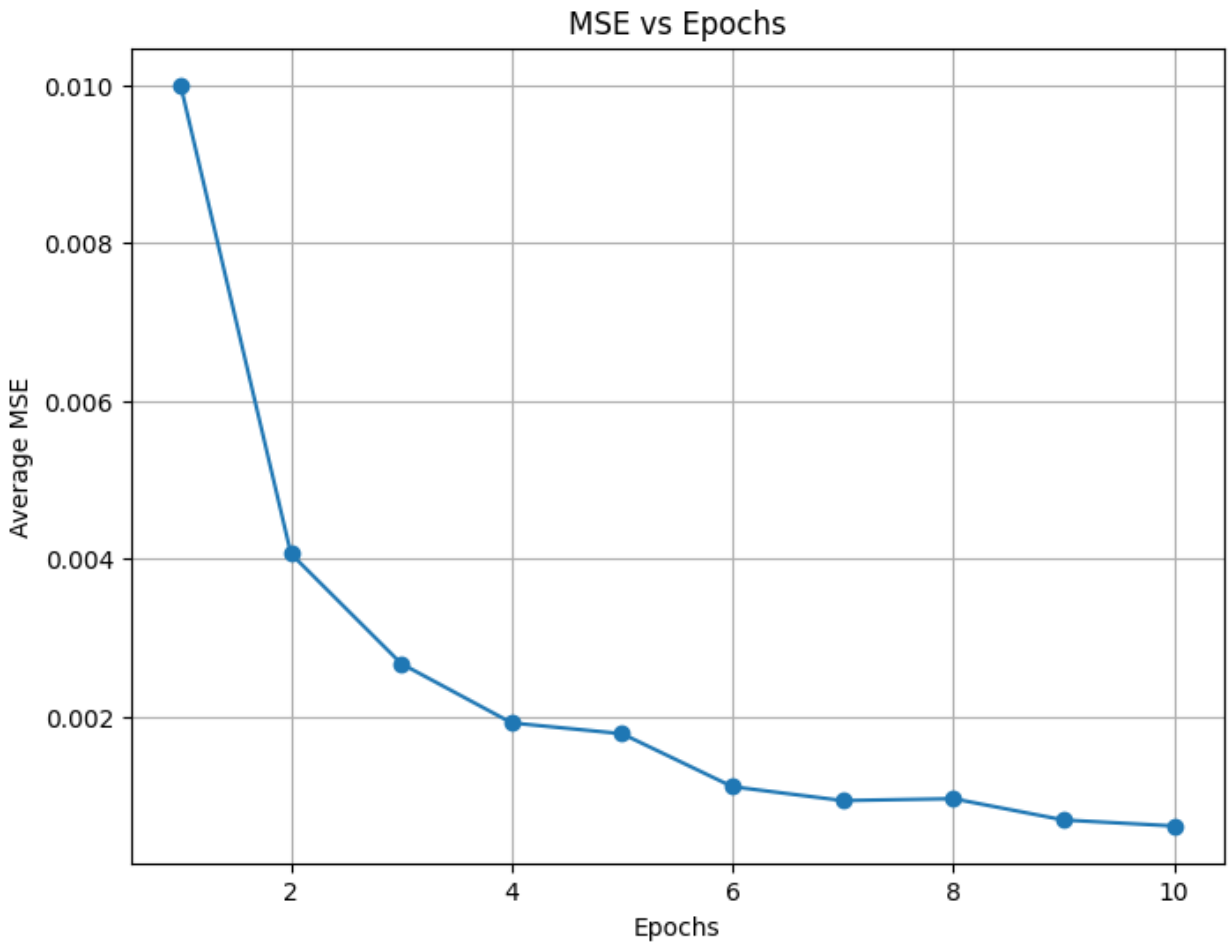


MSE vs Epoch: Increases pixel-level accuracy as training advances. The consistent drop in MSE for ResNet-50 demonstrates its greater predictive power.

- **ResNet-18**

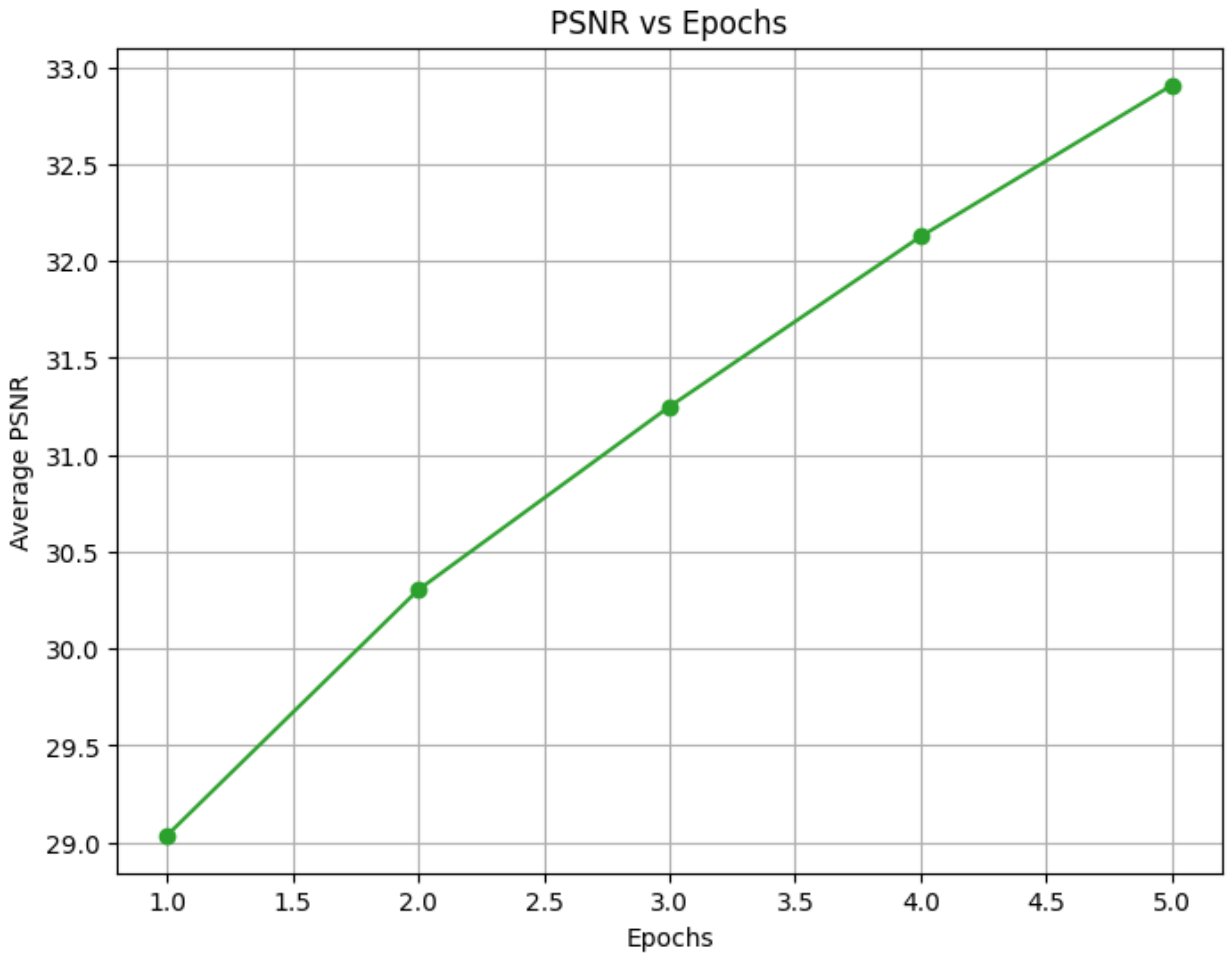

MSE vs Epochs
Average MSE
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Epochs

- **ResNet-50**



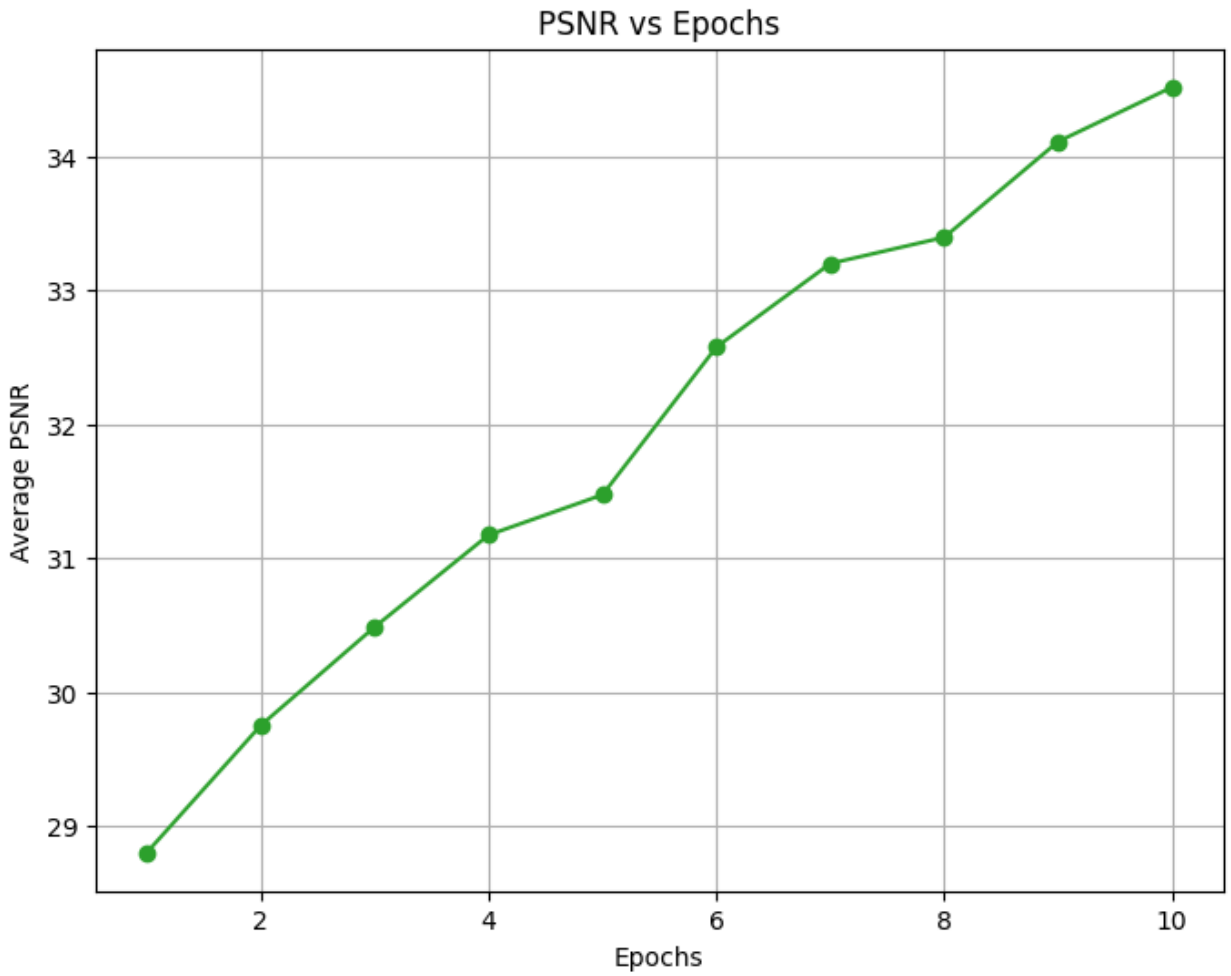
PSNR vs Epoch: This indicates an improvement in the perceptual quality of depth maps. ResNet-50's increased PSNR illustrates its capability to provide visually cohesive outputs with less artefacts.

- **ResNet-18**


PSNR vs Epochs
Average PSNR
Epochs
33.0
32.5
32.0
31.5
31.0
30.5
30.0
29.5
29.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

- **ResNet-50**

Aishwarya Muttineni (50537179)
Chandu Challagonda (50547890)



Analysis

1. **Strengths:**
   - ResNet-50's deeper design allows for sharper and more accurate depth predictions, which leads to improved feature extraction.
   - Gaussian and median blurring are powerful post-processing techniques for reducing noise and producing visibly smoother depth maps.
   - K-means clustering reliably splits depth maps into meaningful sections, which may then be used for applications such as object detection and 3D scene interpretation.
2. **Limitations:**
   - ResNet-18, as a shallower network, struggles to capture tiny details in complex scenarios, resulting in noisier predictions.
   - Over-smoothing during post-processing might conceal important details, especially in areas with abrupt depth changes.
   - The clustering process is susceptible to parameter selection, such as the number of clusters, which might influence segmentation quality.

3. **Insights from Input Complexity:**
    - Images with heavy occlusion or overlapping objects present issues for both models, as evidenced in areas with less distinct depth delineation.
    - Scenes with uniform backgrounds or less texture perform better, demonstrating how depth models rely on input variety.
4. **Comparison of ResNet Models:**
    - ResNet-50 outperforms ResNet-18 on all evaluation criteria, demonstrating the value of deeper feature representations in depth estimation tasks.

Discussion and Lessons Learned

- **Key Insights:**
    - Deeper designs, such as ResNet-50, dramatically improve depth prediction quality by learning complicated feature hierarchies. This is seen in the sharpness and accuracy of anticipated depth maps.
    - Smoothing and grouping are important post-processing processes for fine-tuning raw prediction results. To avoid over-smoothing, a careful balance of noise reduction and detail retention is required.
    - Reordering in K-means clustering enhances the interpretability of depth areas, making the results more useful for downstream applications.
- **Challenges Faced:**
    - To achieve optimal convergence, ResNet-50 training required careful hyperparameter optimisation, including learning rate and batch size.
    - Handling noisy ResNet-18 predictions proved problematic, particularly in complicated indoor scenarios with many objects and occlusions.
    - The computational demands for training deeper networks were greater, necessitating the utilisation of Google Colab's GPU capabilities.
- **Future Directions:**
    - To further improve the quality of depth prediction, attention methods like transformer-based structures or self-attention can be incorporated.
    - To assess the robustness of the model, the dataset will be expanded to include outside scenes with different lighting conditions.
    - To lessen reliance on huge annotated datasets, unsupervised or semi-supervised learning approaches are being investigated.
    - Making use of multi-scale feature extraction methods to record precise features and global context in depth maps.

- **Lessons for Practical Implementation:**
    - Model performance is significantly impacted by standardising dimensions and intensity ranges and carefully preparing input data.
    - It is easier to analyse outcomes and pinpoint areas for improvement when a clear pipeline for visualisation and evaluation is in place.
- **Key Insights:** Performance is much enhanced by deeper architectures (like ResNet-50), but it's critical to strike a balance between noise reduction and detail preservation.
- **Future Work:** Integrate attention-based techniques and test on outdoor datasets.

Bibliography

[1] Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems* (Vol. 27). https://doi.org/10.48550/arXiv.1406.2283

[2] Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., & Navab, N. (2016). Deeper depth prediction with fully convolutional residual networks. In *Proceedings of the 4th International Conference on 3D Vision (3DV)* (pp. 239–248). https://doi.org/10.1109/3DV.2016.32

[3] Godard, C., Mac Aodha, O., & Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 6602–6611). https://doi.org/10.1109/CVPR.2017.699

[4] Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., & Bry, A. (2017). End-to-end learning of geometry and context for monocular depth estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (pp. 66–75). https://doi.org/10.1109/ICCV.2017.17

[5] Yin, W., Liu, Y., Shen, C., & Yan, Y. (2019). Enforcing geometric constraints for monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 2002–2011). Seoul, Korea (South). https://doi.org/10.1109/ICCV.2019.00578

[6] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). *PyTorch: An imperative style, high-performance deep learning library*. Advances in Neural Information Processing Systems, 32, 8026–8037

[7] Torchvision. (n.d.). PyTorch's Computer Vision Library. Retrieved from https://pytorch.org/vision/stable/index.html.

[8] Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12, 2825–2830

[9] Gur, S., & Wolf, L. (2019). Single image depth estimation trained via depth from defocus cues. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, 7683–7692. https://doi.org/10.1109/CVPR.2019.00787