CS231N Final Project: Estimating Depth From RGB Monocular images

Gadiel Sznaier Camps
Stanford University
gsznaier@stanford.edu

Tanran Zheng
Stanford University
trzheng@stanford.edu

Raymond Guo
Stanford University
raygou@stanford.edu

Abstract

In this paper, we present our exploration into using Deep Convolutional Neural Networks (CNNs) to perform Monocular Depth estimation. This is a well known and challenging problem due to the natural ambiguity that stems from trying to reconstruct 3D scenes from 2D information. Recent methods have shifted towards using CNNs to learn these features, therefore, we explore the following four models: Transfer learning using ResNet backbone, transfer learning using UNet backbone, transfer learning using Feature Pyramid Network backbone, and a Multi-Scaled Depth model. Due to the simplicity of the approach, the ResNet model was used as the baseline when examining each method. To perform the comparisons, the NYUv2 Dataset was used and augmented through the use of rotations, flips and grey scale conversions to aid model generalization. The trained models were then compared against each other using the Relative Absolute Difference (L_1), the Mean Squared Difference (L_2), and the Smooth L_1. Using these metrics, it was determined that the method using a UNet backbone performed the best. Meanwhile, it was discovered that the Multi-Scaled Depth model was found to perform the worse on these metrics, but was still able to approximate the relative depth between objects well.

1. Introduction

For humans, identifying the depth in scene from a single picture is a simple and intuitive task as illustrated by Figure 1. However, for computers, this deceptively simple problem represents a significant challenge. Past approaches have attempted several different techniques to overcome this challenge. For example, one technique has been to use geometric assumptions to determine the layout of a room. However, using geometric assumptions to obtain 3D reconstructions is often not accurate and is usually limited to simple scenes. Instead, a different approach is to use user annotations. However, this type of information is typically not readily available and is inconvenient for some applications to obtain. Alternatively, a different method is to use a non-parametric approach in which a group of similar images with known depths are used to find the depth of an image with unknown depth. However, this technique is also often inconvenient as similar images with known depths may not be available.

Figure 1: Image taken from the NYUv2 data set [13]. This is an example of how well known objects and depth cues, such as humans and chairs, can help estimate depth. Base on the relative size it is easy to estimate that the chair is closer to the camera than the human.

In comparison CNNs can be easily taught to extract features from images and have been improving at an incredible pace, hitting new milestones in the field of computer vision each year. As such great strides has been made in estimating depth from stereo vision. However, monocular depth estimation still remains a challenging unsolved problem that often arises in the field of robotics when performing tasks such monocular based SLAM, where depth information is not available due to missing visual cues, or when trying to extract 3D information out of 2D images found online, where additional depth information can provide better context understanding for the scenes in the images.

Therefore, we wish to implement a deep neural network in this project to perform depth estimation. To do so we assume that we have labeled RGB images where the labels are the per pixel depth for each image. The models then take these RGB images as their input and output a corresponding

image where each element in pixel value is the estimated depth for the input image.

2. Related Work

In computer vision, depth estimation is used to estimate depth from a 2D image by using the input of an RGB image and to output an image with information about the distance from the objects to the camera.

Depth perception is an integral component when interacting with the environment. This ability allows humans and animals to manipulate objects, avoid obstacles, determine travel time and perform scene comprehension [1]. Similarly, in the field of robotics, depth perception is integral for performing even the simplest of tasks. Normally, when the topic of depth estimation is considered, it is often assumed that multiple cameras or range sensors will be available for stereo perception. However, this approach is often not possible for small drones, in deep space missions where having multiple cameras is expensive, or in autonomous driving. Instead, one solution has been to been to attempt to estimate depth through the use of a single image, also known as Monocular Depth Estimation (MDE). One issue with this approach is the natural ambiguity that results from attempting to reconstruct 3D scenes from 2D images. Historically, this has been tackled by using algorithms that exploit depth cues from well known features, such as texture information, known object sizes, and occlusions [1]. One common example, is a traditional stereo matching method uses left and right images to calculate depth value and estimates a disparity map between the two images. The new image is made based on the right image and disparity map [15]. However, more recently, the work has shifted towards using Deep Convolutional Neural Networks (CNNs) to instead learn the relevant features for depth estimation [1, 4, 12, 8, 2, 3].

Eigen et. al. [4] proposed using a multi-scale CNN to predict depth, estimate surface normals and semantically label objects. The model was able to switch between tasks with only minor changes to the structure and was designed so that they could directly regress from their input image to their output pixel mapping. To achieve this they passed their input to three convolution blocks which learn features at three different scales. The output of each block was then passed as the input to next higher scaling convolution block, allowing them to generate features at different scaling values and refine their predictions. An advantage of this design was that they could capture image details without having to use super pixels or low level segmentation. In addition, this approach allowed them to return an estimation in real time.

Rather than use three convolutional blocks at different scaling values and then performing regression on the depth, Liu et. al. [12] formulate the challenge of learning depth from a single image as a continuous conditional random field (CRF) learning problem. They then proposed a structured learning scheme that allows them to learn unary and pairwise potentials of the CRF during back-propagation, which in turn allowed them to analytically solve the log likelihood optimization problem. Through their exploration, they found that they could achieve comparable results and faster training time by using superpixels in conjunction with a basic CNN. The main drawback to this solution however is that it requires using superpixels, which reduce the level of fine detail that the model can provide when predicting depth.

Cao et. al. [2] instead treated the problem as a classification problem. To accomplish this, they discretized that ground truth depth into bins of fixed depth ranges. They then trained a residual CNN over the new discrete space. The benefit of this approach is that it allowed them to obtain a confidence value on the model's depth prediction. Furthermore, they noted that this formulation simplifies the problem making it easier for the network to train. However, this formulation also comes at the cost that the depth prediction is less precise.

In contrast, Liu et. al. [11] proposes a deep neural network for piece-wise planar depth map reconstruction from a single RGB image. By using piece-wise planar depth maps for training and testing, the evaluations outperform baseline methods in terms plane segmentation and depth estimation.

These studies represent the related work for single RGB input and how they have approached the unique issue of the lack of information in motion and different shooting angles.

3. Dataset

The NYUv2 dataset was used to train and verify our implementation by splitting it into training (1040 images), validation (240 images) and test data (249 images) [13]. The dataset contains 1449 RGB-D labeled images of household environments obtained through the use of a Microsoft Kinect across 464 different scenes from 3 different cities. Each image sample has dimension $3 \times 480 \times 680$; and each the depth image has dimension $1 \times 480 \times 680$, where the value of the each pixel is the depth.

3.1. Data Augmentation

The training data was augmented through the following steps to help the model learn general features:

- *Rotation*: The input and target images were rotated randomly with angle, $\theta \in [-5, 5]$ degrees with 0.5 probability.

- *Flips*: The input and target images were horizontally flipped with 0.6 probability.

- *Grey Scale*: The input image was converted into a grey scale image with 0.2 probability.

- *Normalization*: The input images were normalized along each channel based on the mean and variance of the training images.

The Rotation and Horizontal Flipping transformations were chosen based on the suggestions given by [5], as they demonstrated that these augmentations help improve depth estimation for novel unseen images. In addition, The grey scale transformation was chosen to reduce the model's dependence in using pixel color, which can depend on environmental conditions such as lighting, to learn features while the normalization was done to remove any anomalies or artifacts from the images.

4. Methodology

We explored four different models for estimating depth from a single image: Transfer Learning using ResNet backbone, Transfer Learning using a UNet backbone, Feature Pyramid Network (FPN), and a Multi-Scaled Depth model based on the work presented by [13]. Because it was the simplest to implement, the Transfer Learning approach using ResNet backbone was used as our baseline model. Furthermore, each model was then trained using an Adam optimizer due to its robustness to choosing incorrect hyperparameters. The learning rate was then chosen to be between $1x10^{-3}$ and $1x10^{-5}$, as higher learning rates introduced undesirable artifacts while lower learning rates cause the model to not train a satisfactory amount. The Architecture of each approach is described in more detail in the following sections below.

4.1. Evaluation Metrics:

We used three evaluation metrics to evaluate the performance of our models: Absolute relative difference (L_1), Mean Squared relative difference (L_2), and Smooth L_2 loss function. Out of the three possible metrics, the smooth L_1 Loss function was used to train all our models as it has better outlier rejection than L_2 [6], and is more sensitive than L_1.

4.1.1 L_1 Loss

L_1 Loss is one of the most popular evaluation metrics for depth estimation, which is calculated by equation 1.

$$\frac{1}{n}\sum(|x_i - y_i|) \tag{1}$$

4.1.2 L_2 Loss

TheL_2 Loss is another popular evaluation metrics for depth estimation, which is calculated by equation 2.

$$\frac{1}{n}\sum_i(x_i - y_i)^2 \tag{2}$$

4.1.3 Smooth L_1 Loss

To train these models we used the smooth L_1 loss given by equation 3.

$$loss(x, y) = \frac{1}{n}\sum_i z_i \tag{3}$$

where n is the batch length and z_i is given by

$$z_i = \begin{cases} \frac{1}{2\beta}(x_i - y_i)^2 & \text{if } |x_i - y_i| < \beta \\ |x_i - y_i| - \frac{1}{2}\beta & \text{otherwise} \end{cases} \tag{4}$$

and beta determines when the loss switches between the L_2 loss and the L_1 loss. For the all the models we trained, β was chosen to be 2 as this corresponded with the mean depth of the dataset.

4.2. Baseline:

A pretrained ResNet18 network provided by the PyTorch library was used as the model backbone for the baseline. This model was chosen due to the size constraints of the hardware used to run depth estimation experiments. The final fully connected layer of the ResNet model was then removed and replaced with five fully connected layers of 512, 512, 128, 4800 and 38400 parameters respectively. Between each layer, ReLU activation functions were used to provided the necessary non-linearity. The output of the model was then up sampled once using bilinear interpolation to bring the output to the same dimensions as the input image. In addition, the weights of the convolutions layers were frozen as it was deemed that the features learned by ResNet for classification were still relevant for depth estimation.

4.3. UNet:

UNet was originally proposed by Olaf Ronneberger et al. [14] for biomedical image segmentation. The network is based on the fully convolutional network and is able to output a segmentation map with the same size as the input image, therefore it is ideal for the depth estimation task. The UNet consists of three major parts: the encoder, the bottleneck, and the decoder, see figure 2. The encoder consists of multiple 2D convolutional layers and max pooling layers, which can extract the features of different spatial resolution of the input images, just like the traditional convolutional network. The extracted features will be fed to the

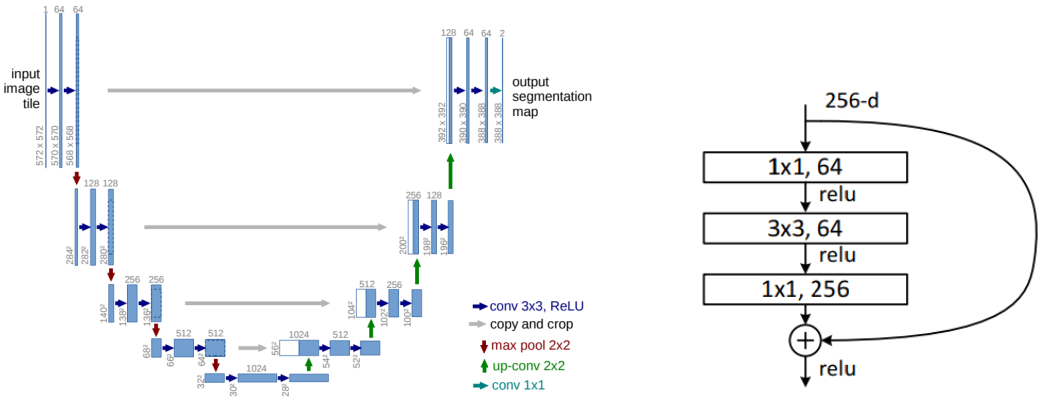


Figure 2: **Letf:** UNet Architecture proposed by Ronneberger et. al. Instead of performing segmentation, final layers are retrained to give the depth estimation for a single image.[14]. **Right:** A building block of ResNet-152, which is the encoder we used for the UNet.[7]

decoder through the bottleneck as well as the skip connections (the grey arrows in figure 2). With the feature representation of the images, the decoder will perform upsampling (up-convolution) and make the final prediction. By using well-designed skip connections between encoder and decoder as well as adding a large number of channels in the decoder, this network is able to propagate context information to higher resolution layers, resulting in higher prediction accuracy [14].

In order to use UNet to tackle our depth estimation task, we made the following changes to the original architecture. First, we used ResNet-152 [7] trained on ImageNet-5K as the encoder, as our task requires learning far more image features than that of the original task in the UNet paper. Next, we added 2D batch normalization and dropout layers to each block of the decoder in order to make it generalize better with fewer training examples. Finally we replaced the loss function with the smooth L_1 loss function described in section 4.1, because the original loss function were designed to focus on the edges of the biomedical images (edges of the cells), which is not suitable to our task. Moreover, during the training of UNet, we kept the parameters of the encoder unchanged and only performed updates to the parameters of decoder and the the final layer. In this way, our model is able to extract the features of the input images better with fewer training examples.

4.4. FPN:

The Feature Pyramid Networks (FPN) was originally proposed by Tsung-Yi Lin et al. [10] in 2016 for object detection task. It was further modified by Alexander Kirillov et al. [9] for instance and semantic segmentation task. It has similar architecture, which consists of encoder (top-down pathway), decoder (bottom-up pathway), and skip connections, see left Figure 3. The major difference between FPN and UNet is that in each of the decoder block of FPN, the model will make a prediction with different resolution ($\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$, and $\frac{1}{32}$ of the size of the original input image) through different segmentation heads, see right Figure 3. Each segmentation head consists of a small 5×5 MLP to predict 14×14 masks and object scores in a fully convolutional fashion. These predictions will be concatenated to output a single prediction map at the end[10]. In this way, the FPN model can also generate high resolution predictions, just like the UNet, but with higher accuracy.

Similar to our UNet implementation, we applied a similar modification for the FPN model. More specifically, we used a ResNeXt-101 model [16] trained on ImageNet-5K as the encoder and added 2D batch normalization and dropout layers to each block of the decoder. Finally, we replaced the loss function with the smooth L_1 loss function. During the training, we then kept the parameters of the encoder and segmentation heads unchanged and only performed updates to the parameters of decoder and the final concatenation layer.

4.5. Multi-Scaled Depth Model:

The final model explored was a variation of The Multi-Scaled Depth model proposed by Eigen et. al. [5]. Rather than employ transfer learning techniques to learn features for depth estimation, the Multi-Scaled Depth Model instead estimates the depth for a single image by learning to take into account local and global information during training. To achieve this, the model utilizes two deep network blocks which train in parallel. The first network block learns a coarse global representation of the scene by down-sampling the image to $1/8$ its original size through the use of max-pooling. Fully connected layers are then used to learn cor-

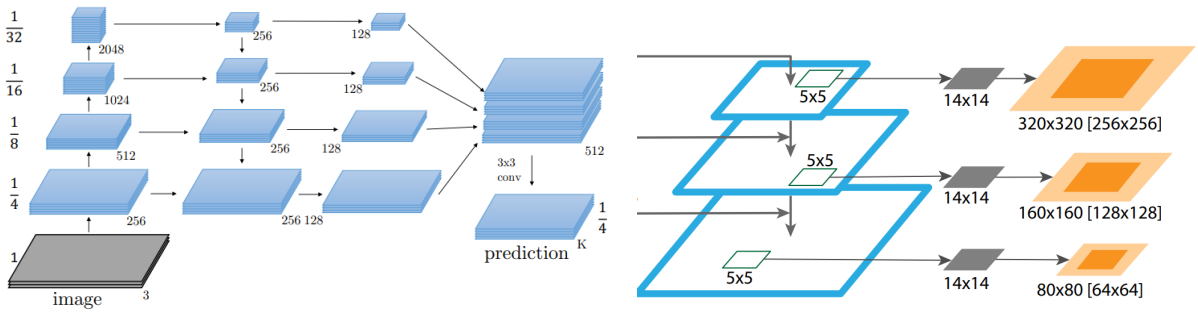


Figure 3: **Left:** FPN architecture proposed by Alexander Kirillov et al. (2017) for sementic segmentation task. [9] **Right:** Details architecture of the segmentation head of FPN. [10]

respondences between each pixel in the down-sampled image. The resulting representation is then up-sampled back to the original image's size and used, along with the original image, as an input for the second network block. The second network then uses the coarse image by passing it as an additional intermediate filter in the middle of the network block. The model is then used to learn a fine grade representation of the depth map by learning local features from a scene by using convolution layers without downsampling. We then modified this approach by introducing batch normalization between each convolution layer for both network blocks to improve the model's ability to generalize. A more detailed explanation of the model architecture is described in figure 4.

5. Results

For this project, two experiments were performed. In the first experiment, the models were trained using three different loss functions (L_1, L_2, and Smooth L_1) and then evaluated using the validation dataset to determine which loss function is the most suitable for our models. It was found that using the smooth L_1 as the loss function resulted in the most consistent training. This stems from the fact that the L_2 loss is sensitive to outliers, as they can create exploding gradients, which in turn causes the training loss to be much more volatile during training. Although using L_1 loss will not cause this issue during the training process, the resulting models perform poorly in predicting the detailed depth. This is because the L_1 loss is not sensitive enough, and as a result the models are not encourage to make predictions that show the edges and objects in the input image. Thus, the Smooth L_1 loss is used as the loss function for training all our models, as it combines the L_1 loss and L_2 loss in a way that allows the models to be encouraged to learn the details while behaving more consistent during the training process.

In the second experiment, we endeavored to obtain a performance comparison across the four models by training them using the Smooth L_1 loss over the training set. The trained models were tested on an unseen image from the test set, as shown in Figure 5 on the left, to visually compare which model was the best at estimating depth of the image. The test loss for the models are shown in Table 1. It was discovered that UNet had the best results across all the metrics. This is most likely due to the fact that the model uses an encoder-decoder scheme with a bottleneck that allows it to learn good features and uses skip connections to skip unnecessary layers if needed. Interestingly, the Multi-Scaled Depth model performed the worse across the various metrics in comparison to the other models even though it was able to return reasonable depth estimations as shown in Figure 5d. This could stem from two possibilities. One possibility is that the model is over fitting to the training data. The second possibility is that while the model does well at approximating relative depth between features, it estimates a greater range of possible depths than the other models, therefore causing a greater discrepancy between the ground truth and the prediction. A more detailed exploration of predicted depth estimations of each model are described in the following sections.

Table 1: Results using various metrics.

	L_1	L_2	Smooth L_1
Baseline	3.7644	6.4166	1.4912
UNet	0.7652	1.0223	0.4042
FPN	0.7933	1.0685	0.4213
Multi-Scaled	3.9355	7.3719	1.6107

5.1. Baseline:

As mentioned previously, for our baseline, we chose to perform transfer learning using a ResNet backbone. Once the model was trained, it was then given the chosen test image to determine is ability to correctly estimate the depth on a novel image. The result of this experiment is illus-

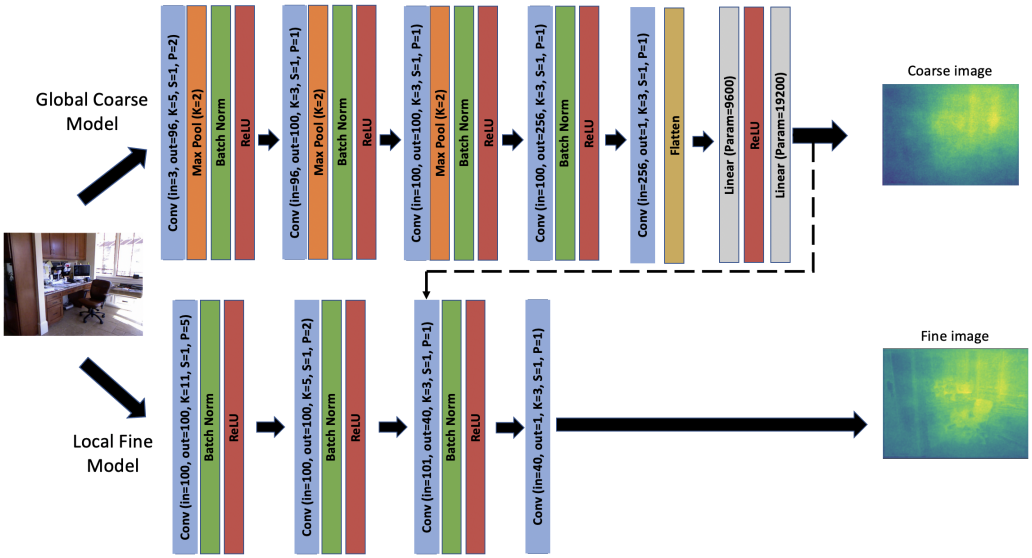


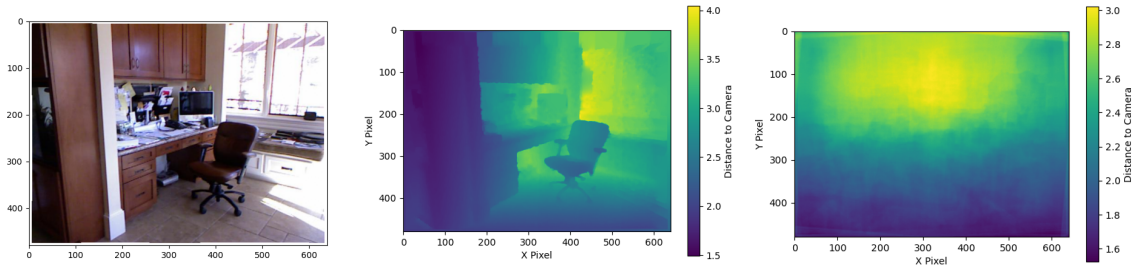
Figure 4: Architecture of the Multi-Scaled Depth Model. Top network block learns a coarse global depth representation while the bottom network block learns a detailed local depth representation. The top network has three blocks of convolution, max-pooling and batch normalization layers. The output is then passed through another block without max-pooling before being passed to a final convolution layer. The output is then flattened and pass to two fully connected layers. The bottom network is three blocks of convolution and batch normalization layers. A final convolution layer is applied with a single filter is returned as the model output along with the coarse image.

trated in Figure 5a. As shown in the figure, the model is approximately able to determine the correct location in the image where the greatest depth should be. Furthermore, the maximum depth value predicted by the model is reasonable and approximately matches the ground truth's value of 4 meters. However, from the figure, it is clear that there is high uncertainty in the model's prediction as it gives the same depth value to large regions of the image. In addition, the model is unable to learn any object features, as evidence by it ignoring the wall on the left of the image. This result was unexpected as the pre-trained ResNet backbone should have provided good learned features weights. However, after some exploration it was determined that this was caused by the combination of the flatten operation and the fully connected layers destroying the learned local features as fully connected layers are poor feature extractors. Furthermore, it was observed that the model was under fitting as shown by validation loss being lower than the training curve in Figure 7 in the supplemental section. Another issue with this approach that was discovered during training was that it is very susceptible to dataset bias. It was found that if data augmentation was not performed, then the model would learn to always assume that the location of greatest depth was in the top center of the image and the location of lowest depth was in the bottom center image regardless of what image was given as input.

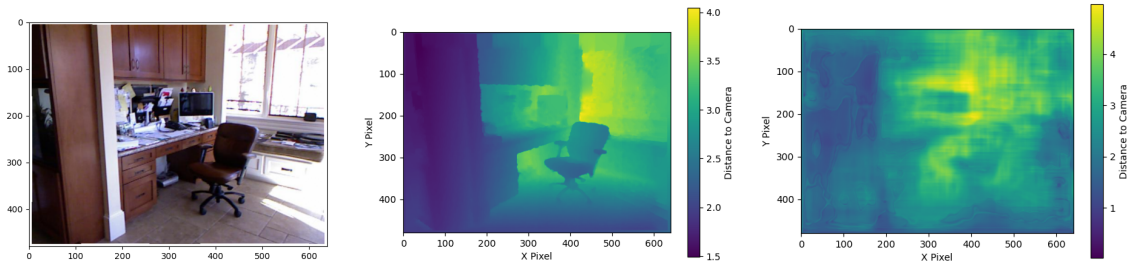
5.2. UNet Model:

According to the results shown in Table 1, the UNet performs the best among all our models under all evaluation metrics. The visualization of the prediction of UNet model is shown in Figure 5b. The UNet performs well in estimating the depth of the input image, as it is able to detect the shape of major objects, such as the chair, table, and monitor in the figure, and accurately predict their depths. It does the best job in recognizing the edges of the major objects in the image among the models. This is because the encoder, decoder, and skip connection architecture allows the model to preserve the high resolution features of the original input image, as a result the model is able to reconstruct these details during the prediction. The result is reasonable because the UNet was originally designed to perform segmentation of high resolution biomedical images and this ability to detecting edges is also useful for the depth estimation tasks as well. Moreover, the pre-trained ResNet encoder allows the UNet to detect the major objects in the image. The UNet model can detect these major objects which are also included in the ImageNet-5k dataset and tends to give them the same depth estimation during the predict process.

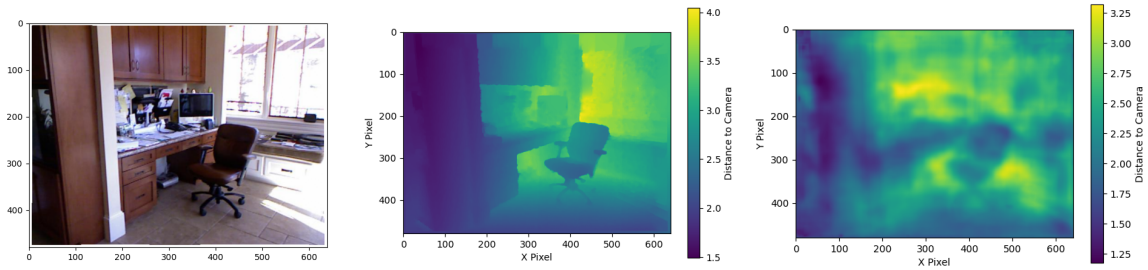
However, when compared to the ground truth, the prediction of UNet still misses the details of smaller objects or edges that are not part of larger objects. For example, the UNet fails to predict the edges of the cabinet and shelf



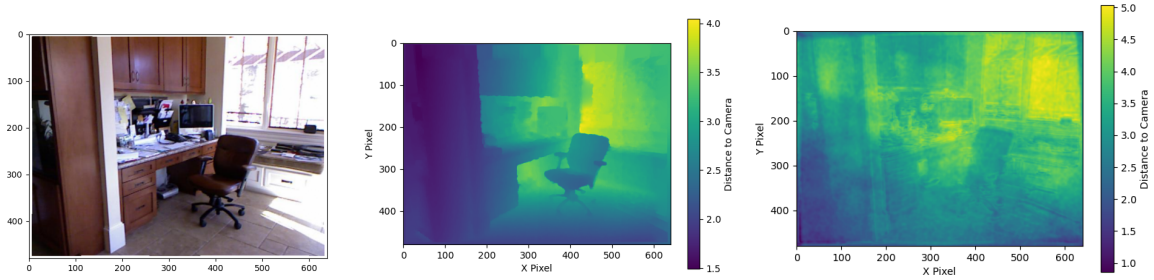
(a) ResNet Transfer Model



(b) UNet Transfer Model



(c) FPN Model



(d) Multi-scaled Depth Model

Figure 5: **Left:** Input image passed into the model. **Middle:** Ground truth depth map. **Right:** Predicted depth map from the model. Dark blue represents locations closest to the camera while bright yellow represents locations furthest from the camera. The depth is in meters.

in Figure 5b. This is because the features of these less-common objects are not captured by the encoder (because either they were not in the ImageNet-5k dataset, or they were harder recognized because they are partially blocked). In addition, some edges and shapes in the original figure are seen distorted in the prediction, which might be caused by the model failing to generalize the impacts of light and shadow. As a result, the UNet transfer model needs to be further fine tuned so that it can leverage both the pre-trained weights from ImageNet-5k and learn to predict smaller de-

tails and lights that were not included in the dataset.

5.3. FPN:

According to Table 1, the FPN transfer model performs slightly worse than the UNet transfer model, but the results are still good. The visualization of the prediction of FPN model is shown in Figure 5c. Similar to the UNet, the FPN also performs well in estimating the depth of the input image by detecting the shape of major objects and predicting their depths. This is because the FPN also has the encoder, decoder, and skip connection architecture, just like the UNet, and it has a even more powerful pre-trained encoder (ResNeXT-152 on ImageNet-5k dataset).

However, comparing to the ground truth and the result of UNet, the prediction of FPN is less clear and losses more information about the details and edges. It also tries to separate the images into different segmentation instead of focusing on the depth only, which makes the FPN performs worse than the UNet. One of the most important reason is that the FPN makes multiple prediction in the decoder and then combine them to generate the final output, while UNet only makes one prediction at the very end of the decoder. Since some of the predictions at the early layers of the decoder have lower resolution than the final output, these predictions will not have the same level of detail. As a result, during the combination of these predictions, some details of the small objects or edges will be lost. This is good for image segmentation task (which is the original purpose of the FPN model) because the model can focus more on the major edges of the image, but might not be good for depth estimation. Another reason why the FPN model performs slightly worse than the UNet model is that the FPN model is more complex, i.e. has more layers and parameters, than the UNet model and the NYU dataset is too small for the FPN model. Thus, the FPN model is slightly overfitted. This can be justified by the training loss curve, shown in Figure 9.

As a result, the FPN transfer model needs to be trained on a larger dataset (or more and better data augmentation techniques). Moreover, the segmentation heads need to be further fine tuned so that they can learn to preserve the details or smaller edges in the image better.

5.4. Multi-Scaled Model:

The final model explored was the Multi-Scaled Model using the model architecture described in Figure 4. The result of the model's prediction for the coarse and fine depth estimation given the chosen test image are shown in Figure 6 and Figure 5d respectively. From Figure 6 we can see that the first network block allows the model to do a decent job at predicting a rough approximation of the depth for the given image. Furthermore, similar to the baseline, the coarse global model is unable to retain any learned object features because of the fully connected layers used. Yet,

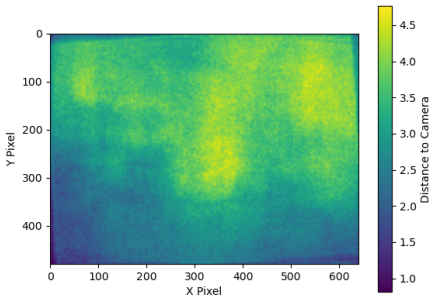


Figure 6: The coarse predicted depth predicted by the Multi-Scaled Depth model given the input image shown in Figure 5d.

when this output is passed to local fine model block, as illustrated by Figure 5d, the model is now able to estimate the depth while still keeping the learned object features without generating any major artificial artifacts. This suggests that learning a down-sampled depth representation can be used as a useful prior for predicting the depth for a single image. To verify this hypothesis, we ran the same model without this component and found that while the model was still able to learn object features, it was also now unable to accurately estimate the pixel-wise depth and would set all pixels to be the mean depth, therefore supporting our hypothesis. Although the combined global and local model was able to estimate depth relative to objects well, it mistakenly predicts a greater depth range than what is seen in the ground truth. This might account for why the model performed worse than the other models on the evaluation metrics as shown in Table 1. This greater depth is also focused on the location where the window appears, which implies that the model learned to associate high depth with windows, which also lends to the idea that model was over-fitting to the small dataset. This hypothesis is further supported by Figure 10 shown in the supplemental materials as the validation loss plot remains constant after 8 epochs while the training loss still continues to decrease during the entire training period.

6. Conclusion

In this paper we implemented four models to perform depth estimation from RGB monocular images. These models are trained with NYU Depth V2 dataset and then are evaluated with different evaluation metrics. Based on the quantitative evaluation and the visualization of output images, the major conclusion is that the UNet transfer model performs the best. To further improve upon these results, we would like to further fine tune these models and improve the size and quality of the training dataset. We would also like to explore more loss functions that might be more suitable for the depth estimation task.

7. Contributions

Tanran wrote the code for the UNet and FPN models and wrote the corresponding sections in the methodology and results. In addition, Tanran wrote the evaluation Metrics section. Gadiel wrote the code for the ResNet baseline and the Multi-Scaled Model and wrote the corresponding sections in the methodology and Results. Furthermore, Gadiel wrote The dataset section. Raymond analyzed existing research and depth estimation techniques, scoped research problem, contributed to data analysis, wrote the Introduction and created the presentation for the project video. Everyone helped write the Related Work section, the Abstract, and proof read the paper.

8. Code

Project Github repository can be found at https://github.com/zhengtr/CS231N-Final-Project

A. Supplemental Results

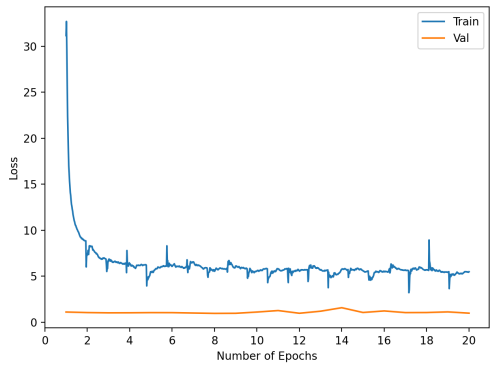


Figure 7: The training and validation loss of ResNet baseline model.

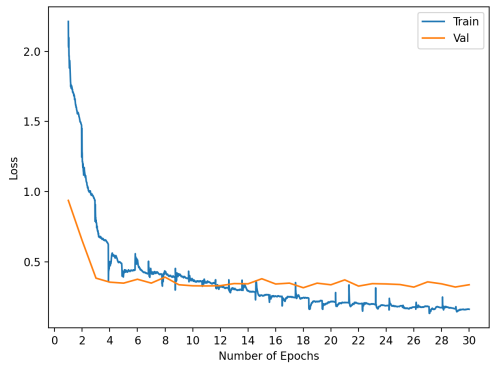


Figure 8: The training and validation loss of UNet transfer model.

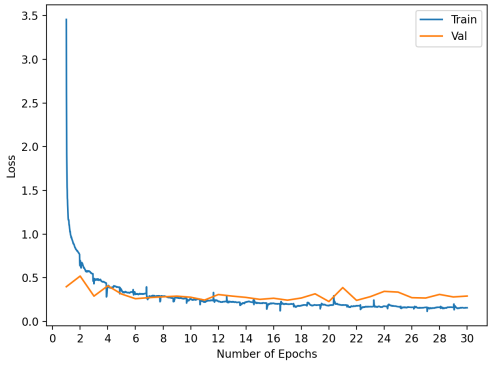


Figure 9: The training and validation loss of FPN transfer model.

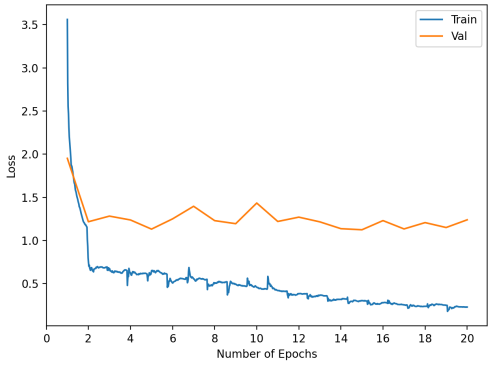


Figure 10: The training and validation loss of Multi-Scaled model.

References

[1] Amlaan Bhoi. Monocular depth estimation: A survey. *arXiv preprint arXiv:1901.09402*, 2019. 2

[2] Yuanzhouhan Cao, Zifeng Wu, and Chunhua Shen. Estimating depth from monocular images as classification using deep fully convolutional residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(11):3174–3182, 2017. 2

[3] Zhao Chen, Vijay Badrinarayanan, Gilad Drozdov, and Andrew Rabinovich. Estimating depth from rgb and sparse sensing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 167–182, 2018. 2

[4] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015. 2

[5] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *arXiv preprint arXiv:1406.2283*, 2014. 3, 4

[6] Ross Girshick. Fast r-cnn, 2015. 3

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 4

[8] Lei He, Guanghui Wang, and Zhanyi Hu. Learning depth from single images with deep neural network embedding focal length. *IEEE Transactions on Image Processing*, 27(9):4676–4689, 2018. 2

[9] Alexander Kirillov, Kaiming He, Ross Girshick, and Piotr Dollár. A unified architecture for instance and semantic segmentation, 2017. 4, 5

[10] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017. 4, 5

[11] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. Planenet: Piece-wise planar reconstruction from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2588, 2018. 2

[12] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2024–2039, 2016. 2

[13] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 1, 2, 3

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 3, 4

[15] Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. Stereo matching using belief propagation. *IEEE Transactions on pattern analysis and machine intelligence*, 25(7):787–800, 2003. 2

[16] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks, 2017. 4