Combining 2D Lidar and Monocular RGB Vision for Accurate Depth Estimation

[redacted]
Department of Computer Science
[redacted]
[redacted]

***Abstract*—3D lidar can be used to get accurate measurements of depth in a full horizontal and vertical field of view, but is heavy and expensive. 2D lidar is cheaper and lighter, but only gets a single horizontal line scan. Neural networks can estimate depth from a single RGB photo, but fail in many cases. This project attempts to combine 2D lidar and neural networks to approach the accuracy of 3D lidar, without the downsides. This depth estimation system would be useful in many situations, including autonomous aerial vehicles where weight is a major concern. We propose a U-net based on ResNet-50 that performs depth estimation from scratch. We find that an auxiliary network that injects a 2D lidar scan into the model decreases error metrics by roughly 50%. We also find that a very simple interpolation system can outperform our neural network by a small but significant margin, implying room for significant improvement.**

***Index Terms*—depth estimation; neural nets; lidar**

I. Introduction

In the context of deep learning, *depth estimation* is the conversion of a monocular image to a dense depth map, with an estimated depth value for each pixel. This has many applications, ranging from robotics and self-driving cars to other fields such as 3D modeling. A closely related problem is *depth completion*, where a sparse depth map is combined with an image to obtain a dense (or *complete*) depth map.

In the general context, there are currently many approaches to depth estimation of a scene. We will be focusing on two that are relevant to our project: lidar and deep learning. 3D lidar is accurate, but can be expensive, heavy, and slow. Deep learning on monocular RGB images is usually cheap, light, and fast, but can fail in many scenarios. This project aims to combine the two in such a way that they cover each other's weaknesses. We will add a 2D lidar scanner to a neural network depth estimation system to increase the accuracy of its estimations.

Depth estimation with deep learning has been lightly studied since 2014. Eigen et al. [1] introduce a 10-layer convolutional neural network (CNN), the first such network to be applied to depth estimation. Later works [2] [3] deepen the network considerably, building upon ResNet [4] to achieve much better results.

Depth completion takes some sparse ground truth depth information and combines it with an RGB image, producing a dense depth estimation. The first papers on depth completion simply took randomly-distributed ground truth samples [5], [6]. However, many more recent papers [7]–[9] take input from a sparse 3D lidar system, which usually gets ground truth scan lines covering a small percentage of the image.

To the best of our knowledge, there is one prior published work that attempts to combine 2D lidar with a monocular RGB camera for depth estimation [10]. Liao et al. take a single horizontal 2D lidar scan of a scene, stretch it vertically, and then predict the delta between the lidar scan and the true depth at every given pixel. Our solution is similar, but adds on spatial propagation techniques that attempt to make use of the lidar scan in a more intelligent manner.

II. Background

A. Neural Networks

A complete explanation of neural networks (abbreviated nets or NNs) is out of scope of this paper, but we will attempt to briefly explain enough to provide an intuition of data flow. Readers already familiar with neural nets can skip this subsection.

A neural net is, to some extent, a black box. One feeds in data, and out comes a result. Inside the box is a collection of layers; the data flows through these layers, usually in sequence. Each layer transforms the data in a relatively simple way, but the product of all the layers combined can be complex. A basic net will feed the data through the layers one by one, but more complex nets can recombine the data in useful ways.

At the top (the start) of a Convolutional Neural Net (CNN), the data consists of the three channels of an RGB image. As it travels through the net, the concepts represented by the data become more and more abstract, going from color to edges to real-life objects to broad generalizations of the scene as a whole. By combining all these different levels of abstraction, we can make depth predictions that take into account the large-scale features of a scene, without losing the small details.

B. Lidar

Lidar (**li**ght **d**etection **a**nd **r**anging) is a technology that can get accurate depth information for a single point. By making the laser scan side-to-side, one can obtain a scan that gives information for a line. This is known as 2D lidar, as it can construct a 2D cross-section of a scene. By making the laser also scan up and down in addition to side-to-side, one can obtain a full scan of a scene. This is known as 3D lidar, since it can construct a full 3D representation of the scene.

Unfortunately, the additional dimensions come at a cost. The hardware to redirect the laser typically increases the weight and price of the lidar unit by a significant amount. Additionally, it also increases the time to collect a complete scan, since the lasers need to move to cover the full area being scanned. To offset this, some lidar units incorporate multiple lasers into their designs, to collect data on multiple points at once. However, this also increases their price and weight even more.

In contrast to lidar, RGB cameras are quite cheap, and are usually already present in situations where 2D or 3D lidar might be used. While they cannot directly obtain depth readings, they can give information about the composition of a scene. Our hypothesis is that we can combine 2D lidar with an RGB camera to approximate the capabilities of 3D lidar, at a much lower price and weight or higher scan rate.

III. Related Work

A. Depth Estimation

Make3D, by Saxena et al. [11], is commonly cited as one of the first works performing depth estimation on a single image. However, their algorithm does not perform very well relative to more recent neural-net-based solutions.

Eigen et al. [1] were the first to use a neural net for single image depth estimation. Their technique consisted of predicting a coarse (i.e., blurry) depth prediction, then refining this prediction in a subsequent step. Their neural net has 10 layers in total. For the coarse prediction, there are five convolutional (conv) layers followed by two fully connected (dense) layers. The small size of these layers creates a bottleneck that enforces a maximum level of detail in this area, resulting in a blurry prediction. For the fine prediction, there is one conv layer, after which the result is concatenated with the coarse prediction. Finally, two more conv layers process this concatenation, creating the final result. When applied to the NYU Depth and KITTI datasets, this model obtains mediocre results. However, the fine output is not particularly useful, getting metrics that are barely better—and in a few cases, worse—than the coarse output. Overall, they demonstrate that single image depth estimation is possible, but there is still ample room for improvement.

Laina et al. [2] make a few notable contributions. Their system is similar to [1], but a bit more complex in most regards. The primary conceptual difference is the lack of a long skip connection between the start and end of the network. They use the ResNet-50 model [4] for the downsampling portion of their network, and then complex novel *up-projection* blocks for the upsampling portion. Their loss is the piecewise reverse Huber loss (*berHu* loss): $\mathcal{L}_1$ for close distances, and $\mathcal{L}_2$ for farther distances. With these techniques combined, they achieve good results for a solution that is relatively straightforward.

Cao et al. [3] propose an interesting technique. Rather than formulating the problem as regression, they treat it as pixel-wise classification. They create 10-120 discrete depth bins, and then classify each pixel into one of these bins using a fairly standard neural net classification process. They use a pretrained ResNet model for part of their network, and also try conditional random fields and markov random fields at the end to refine the predictions. Perhaps surprisingly, this discretization does improve the model's evaluation metrics.

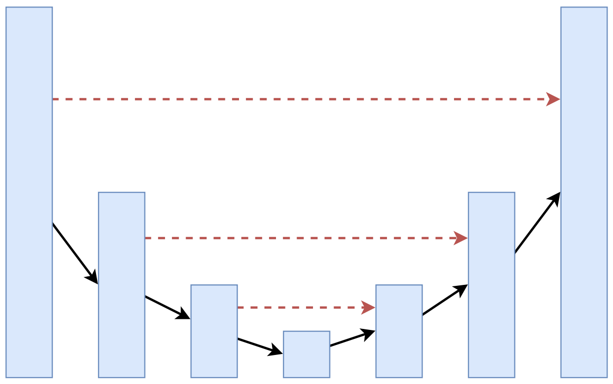
Fig. 1. Generic U-net. Skip connections (dashed lines) bypass the lower-resolution sections of the model.

B. Depth Completion

While it isn't the first to do depth completion, Ma and Karaman 2018 [6] is one of the first to obtain high-quality results. They base their model off of [2], using ResNet, up-projection blocks, and berHu loss. The sparse depth information consists of a number (generally around 50-500) of randomly selected points, uniformly distributed across the image. They do not specify exactly how they feed this data into the neural net, but we can infer that they likely put it in as a fourth channel in the input image, and all non-selected points are set to zero. Due to the bottleneck that is the last layer of ResNet, the output is fairly blurry and imprecise. However, the overall accuracy metrics are quite good, and the most significant failure mode of depth estimation (catastrophic failure to understand an object) is not present.

Most works in recent years use architectures based off of, or similar to, U-Nets [12]. A U-Net is a network that is shaped like a "U". The top of the two sides of the U are each full-resolution, and the bottom is low-resolution but high feature depth. The critical component of a U-Net is the horizontal skip connections between the two sides of the U, as seen in figure 1. These let high-resolution information pass through, without getting diminished at the bottleneck at the bottom of the U.

Ma et al. 2019 [7] and Guizilini et al. [9] both use architectures based off of U-Nets. They both perform depth completion with 3D lidar scans, which are sparse, but have good coverage of the whole scene. [7] puts the depth information in at the top of the network, and beyond that has a standard U-net architecture. [9] creates Sparse Auxiliary Network (SAN) downsampling net that works adjacent to, but is not connected to, the regular downsampling portion of the U-Net. The initial depth information is put into the top of this network, which is then connected to the skip connections of the U-Net. This means that the downsampling portion is performing feature extraction alone, and only the upsampling portion performs the actual depth estimation. This technique likely works due to the relative density of the depth information. When the problem is formulated this way with 3D lidar input, the task of depth completion is more similar to “spread out the input depth ground truth in an intelligent fashion” rather than the “estimate depth based on input RGB” that exists with pure depth estimation.

Cheng et al. [5] propose a spatial propagation technique that takes advantage of this formulation. They use data similar to [6], and a U-Net architecture similar to [7]. However, at the end of their model, they add a novel Convolutional Spatial Propagation Network (CSPN). This propagates the depth information, but does so in a way that looks at the RGB values. Intuitively, if two adjacent pixels have similar RGB values, their depth values are likely similar. In contrast, if their RGB values are very different, their depth values are more likely to be significantly different. This technique outperforms [6] by a large margin.

C. 2D Lidar Completion

2D lidar completion conceptually lies in between depth estimation and depth completion. Depth estimation uses RGB information to estimate depth from scratch. Most formulations of depth completion use 3D lidar depth information to estimate the depth of nearby pixels, and use RGB information merely to assist with accurate propagation. Depth completion from 2D lidar has to estimate depth almost from scratch far away from the single scan, but can propagate the depth information from the scan to more accurately estimate the depth of nearby pixels. In addition, it can resolve the scale ambiguity that may be present, which can still assist in the estimation of all pixels regardless of where they are.

To the best of our knowledge, Liao et al. [10] is the only attempt to perform depth completion using only a 2D lidar scan. Their problem is formulated exactly as ours is. There is an RGB image of a scene and a 2D lidar scan horizontally across the image, and the objective is to estimate the depth of all pixels in the scene. Their architecture is very similar to that of Laina et al. [2]. They use ResNet-50 as their main downsampling network, and then only a few transposed convolutional layers for upsampling. The main difference is the addition of the depth information. The 2D scan is filtered and interpolated, and then stretched vertically to fill the whole area of the image. This is then concatenated to the input RGB. It is also added (as in arithmetic summation) to the output of the network. In this way, it forces the net to learn to predict the delta between the reference depth scan and the ground truth depth. The performance of this network is compared to several previous works, including [11], [1], [3], and [2]. It outperforms the best of these by a small but comfortable margin.

IV. Methodology

In this section, we describe the dataset used, an overview of notable models we attempted, and the full details of our final model architecture and hyperparameters.

A. Data

Three datasets were considered for this project: KITTI [13], NYUv2 [14], and DIODE [15].

The KITTI dataset [13] consists mostly of video data gathered from a vehicle driving along city streets. There is a large quantity of data, but it has several issues for our purposes. The depth maps are not dense, since they are gathered at 10 frames per second, and the lidar unit used to make the dataset is not capable of gathering dense scans at that rate. Since the data is video, most frames are redundant and do not provide a significant amount of additional information for the purposes of model training. Even disregarding this, there is still little diversity in the data, as it is almost entirely views of similar-looking roads. The dataset could have been used for our experiments, but doing so would have required a significant amount of effort to prepare the data, and would have used more disk space than was available to us at the time.

The NYUv2 dataset [14] is videos from a handheld unit indoors. It has most of the same problems as the KITTI dataset, including low-quality depth maps and lack of diversity relative to the disk size of the dataset. There is a small subset of the data that is high-quality, diverse, and not too large to handle; however, it has less than 2000 samples, which would likely not be enough to train a deep neural net adequately without overfitting.

The DIODE dataset [15] contains many individual scans of indoor and outdoor locations. The depth maps are high-quality, there is less redundancy in the samples, and the size of the dataset is manageable within our environment’s constraints. This is the dataset we use for our experiments.

DIODE consists of high-resolution scans of several scenes, which are defined here as a cohesive area with similar features, such as a single room or group of adjacent similar rooms. Each scene was scanned from multiple locations, with each 360-degree scan resulting in dozens of individual crops with a small field of view. Each crop has a 1024x768 RGB image and a corresponding dense depth map of the same resolution. Given the accuracy and resolution of the lidar scanner used, the depth map is entirely ground truth—no interpolation was used. However, due to the inherent limitations of lidar, some of the data points are unreliable. Examples of things that can cause unreliable data points include:

- Long distances. The lidar scanner has a max range of 350 meters.
- Highly reflective surfaces. These can cause the laser to bounce more than once, and thus overestimate the distance to the surface.
- Surfaces at a very sharp angle to the sensor.

The authors of the dataset [15] use median filtering to find data points that are likely to be erroneous, and provide a mask for each crop.

We performed two preprocessing steps. The first was downsampling each crop by a factor of 4, resulting in a resolution of 256x192. The second was to create the simulated 2D lidar scans. We took the middle row of each depth map,

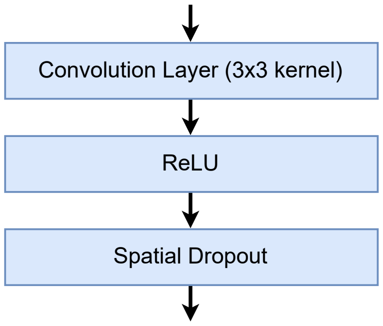

Fig. 2. Our standard convolutional block used in our models.

filtered out unreliable data points using the provided masks, and refilled those points using simple linear interpolation and extrapolation. The masked ground truth maps were used for optimizing our models, and the interpolated 2D lidar scans combined with the RGB images were used as input to our models.

The DIODE dataset has train, validation, and test sets, of which only train and validation are available for download. To prevent data leakage, no scene is in multiple sets. This means that our models cannot memorize a scene that is used in evaluation, and gives us more informative metrics. However, it does so at the expense of decreasing diversity in the training set.

B. Attempted Models

All our models are based on the U-net architecture [12]. The purpose of a U-net is to provide output that has the same (or similar) resolution as the input, as opposed to an image classifier, for example, which is designed to take high-resolution input and reduce it to a small number of output values. See figure 1 for a diagram of a generic U-net. To allow processing of high-level coarse features and low-level fine details, the input starts at the left of the "U" shape, gets reduced to its minimum resolution at the bottom of the U, and then increases in resolution on the right side of the U. The architecture's most unique and notable feature is the skip connections between the sides of the U. These skip the coarser sections of the model, and allow fine details in the data to flow from the input to the output without going through a bottleneck that would discard information.

This architecture is very useful for depth estimation and completion. The middle of the model can recognize large-scale features and relationships in the scene, such as the existence of certain objects or the layout of a room. The sides of the model—aided by the skip connections—can process details like the exact location of a doorway's edge or the texture of a surface.

Our first notable model attempted had 29 convolutional layers, most of which were part of a block. Each block has, in order:

- A convolutional layer with a kernel size of 3x3, and L2 kernel & activity regularizers. Some layers have a stride of 2 when downscaling or upscaling the data. When upscaling, a transposed convolutional layer [16] is used.
- A ReLU activation layer [17].
- A spatial dropout layer [18].
- A batch normalization layer [19] with batch renormalization, which has been found to perform better, especially with smaller batches [20]. We removed this layer in most of the models.

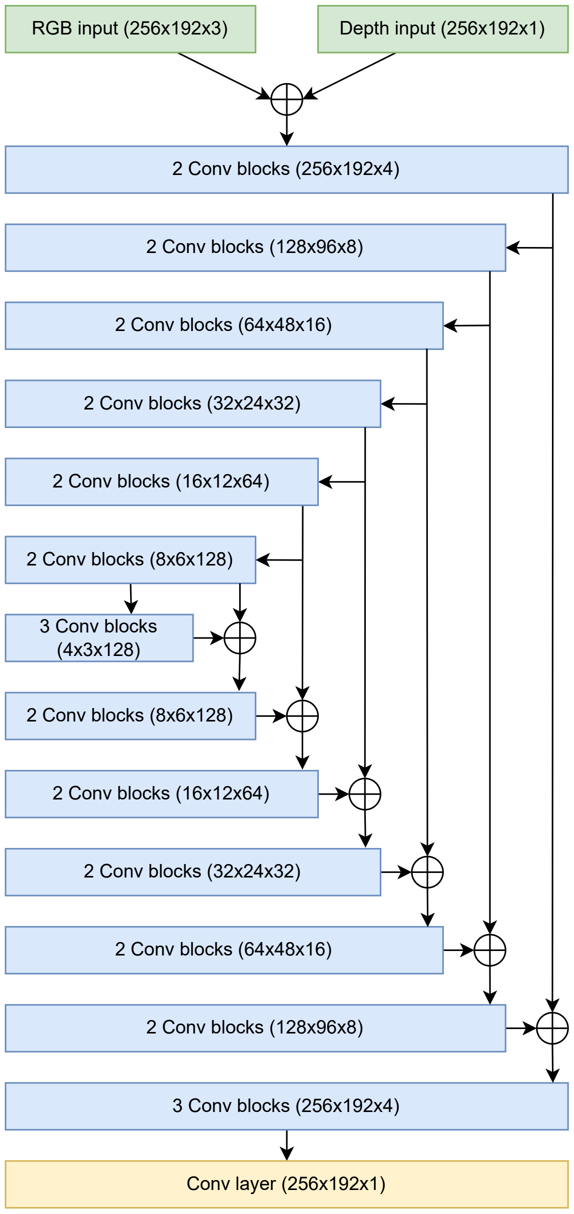

Fig. 3. First notable model attempted. Each block is repeated 2 or 3 times. The $\oplus$ symbol indicates featurewise concatenation.

See figure 2 for a diagram of these blocks. This 29-layer model was essentially a standard generic U-net, with the depth being merely a 4th channel in the input. To simulate a 2D lidar scan, the entire depth map was set to 0 except for a single line in the middle. See figure 3 for a diagram of this model's

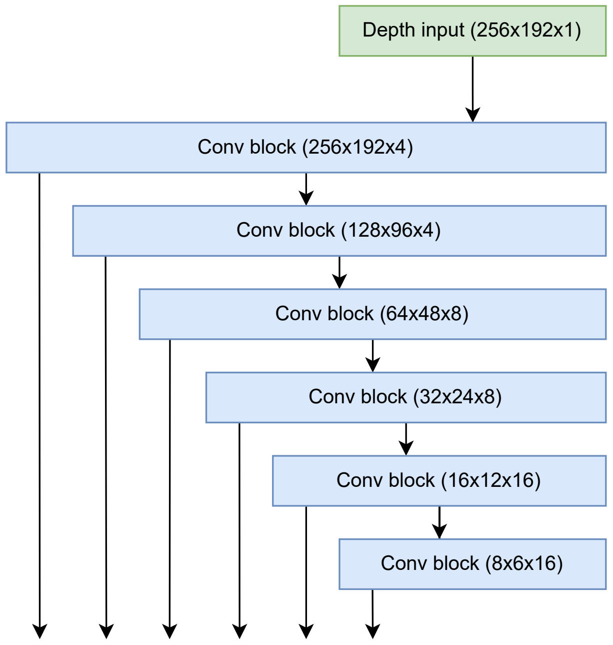

Fig. 4. Second notable model attempted. The arrows at the bottom attach to the concatenation points in figure 3

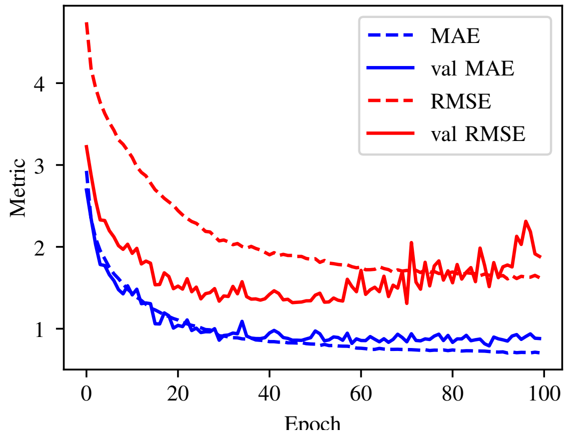

Fig. 5. Metrics during the training of our final model with 2D lidar.

architecture.

It did not perform well. It appeared to learn basic heuristics such as darker pixels are likely to be further away, but did not otherwise appear to learn anything of value. In particular, it did not appear to make use of the input depth scan. We hypothesized that the depth channel was being overshadowed by the RGB channels, and thus little to no depth information existed in the later stages of the model.

To attempt to fix this, we designed our second model. This was nearly the same as the first, but instead of feeding the depth map directly into the main network, we created an auxiliary network to process the depth. This auxiliary network, inspired by the sparse auxiliary network in [9], is shown in figure 4. Notably, there is no depth information in the first half of the model. The processed depth maps from the auxiliary network are concatenated along with the skip connections, so only the second half of the model receives depth information. This 35-layer model did not show noticeable improvement over the 29-layer model. We speculate that the sparse auxiliary network works well in [9] since it has much more well-distributed depth information from 3D lidar as input, rather than our single line from a 2D lidar scan.

C. Final Model

Our main final model is based on a pretrained ResNet-50 model [4]. ResNet is a fairly generic convolutional neural network (CNN) originally designed to classify ImageNet [21]. It uses standard techniques for its time, including 3x3 convolutions, ReLU activations [17], and batch normalization [19]. ResNet-50 serves as a good foundation for many systems, as it is performant, tends to give good results, and uses fewer parameters than competing models.

The 2D lidar model adds on to ResNet in two places. The first is that we remove the fully connected layer at the top of the model and replace it with transposed convolutional layers. The output of ResNet before upscaling is 8x6, so we include five 3x3 transposed convolutional layers with a stride of 2 to upscale back to the original 256x192 resolution. Every layer in our model (here and elsewhere) has the main convolutional layer, ReLU activation, and 0.1 spatial dropout [18]. We also add skip connections between the intermediate layers of ResNet and the upscaling layers at the top of the network, similar to the U-net architecture proposed in [12].

The second addition to ResNet is an auxiliary network to process the depth information, similar to [9]. While we use normal convolutional layers rather than complex blocks that include 2D Minkowski convolutions, the high-level architecture is the same. No depth information goes into the ResNet section of our model; it only goes into the auxiliary network. The starting, ending, and intermediate values of this network are concatenated to the skip connections, so that the upscaling section of the model combines and recombines the low-resolution, high-resolution, and depth information. See figure 6 for a diagram of this model.

All our conv blocks use L2 regularization with $\lambda = 10^{-5}$. We experimented with including batch renormalization [20] in our conv blocks, but found that this destabilized the model, possibly due to the long tail in the distribution of depth values.

We also created an identical model that lacks the auxiliary depth network and 2D lidar input. This model was created as a baseline for comparison, so that we can evaluate how much the model's results are improved by the addition of 2D lidar.

Both of these models were implemented using Keras [22] in TensorFlow [23] version 2.4.1, and trained using a Tesla P4 GPU. The batch size was set to 16 due to GPU memory limitations. The Adam optimizer was used [24] with the default Keras parameters ($\beta_1 = 0.9$ and $\beta_2 = 0.999$) and an initial

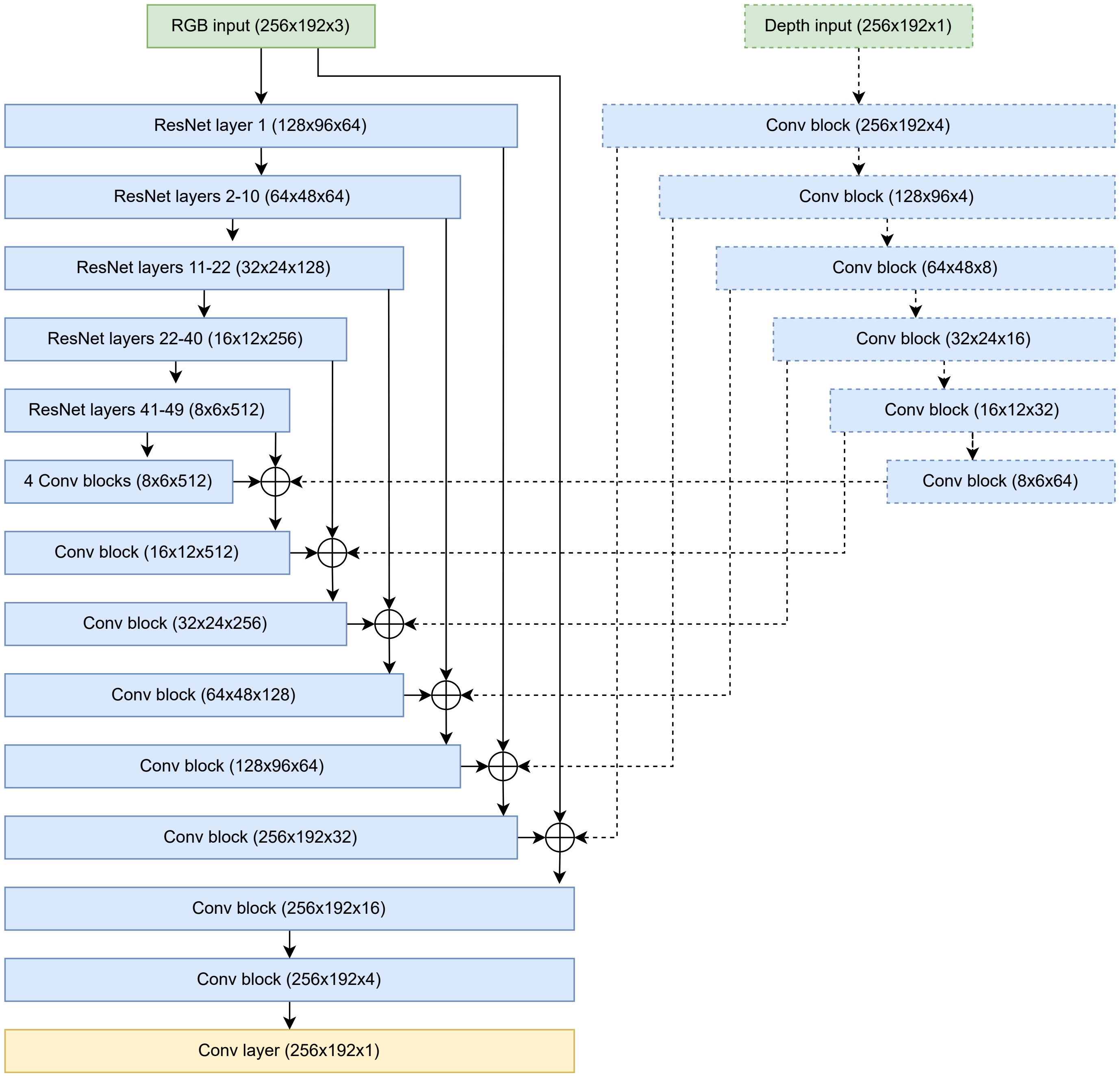

Fig. 6. Final model. The auxiliary depth network is in dotted lines, and can be omitted for a depth estimation model with no lidar input.

learning rate of 10^{-4}. This was trained for 100 epochs, with the learning rate schedule decreasing by a factor of 3 every 30 epochs. Several training metrics can be seen in figure 5. Most metrics continued to decrease slightly over time, but RMSE on the validation set started to increase and become volatile around epoch 50. This suggests the possibility of overfitting. We made several attempts to retrain the model with early stopping, but each attempt either stopped before the model was fully trained or stopped after overfitting already occurred. The loss function used for training was Mean Squared Error (MSE) in $\log_{10}$ space:

$$\mathcal{L}(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(\ln(y_i + 1) - \ln(\hat{y}_i + 1))^2$$

where N is the total number of pixels in a batch, ignoring masked pixels. All depth values are nonnegative, so adding 1 to them guarantees that they are nonnegative in log space. This learning rate schedule was reached after attempting higher learning rates, and finding that these cause the loss to explode, permanently destabilizing the training run.

TABLE I
METRICS.

Model	MAE	RMSE	AbsRel	log MAE	log RMSE
Interpolation Baseline	0.588	1.565	0.173	0.046	0.097
Vasiljevic et al. [15]	1.502	1.695	0.331	0.158	0.178
Ours - no lidar	1.785	3.146	0.455	0.147	0.201
Ours - 2D lidar	0.908	2.405	0.228	0.064	0.105

V. RESULTS

In this section, we define the metrics we will use for comparison, compare the metrics, and then give a qualitative analysis of several output samples.

A. Metrics

We tracked several metrics for our models. Finding helpful holistic metrics is difficult due to two competing factors. The first factor is that it is important to exponentially penalize larger errors more than smaller errors. Intuitively, this is because it is usually desirable to have a model with few weaknesses, rather than a model that has many strengths at the expense of also having many weaknesses. This is why mean squared error is usually used in most regression problems, instead of mean absolute error. The second factor is that in the context of depth estimation, it is desirable for error metrics to be scale invariant, meaning that a 10-meter error in a large-scale outdoor scene should be similar to a 1-meter error in a small indoor scene. The easy way to do this is to penalize large errors less than smaller errors, but doing so goes against the first factor mentioned. There is no standard solution to this problem, so we use several metrics that have different tradeoffs when it comes to these factors.

We used most of the metrics that were given as baselines for the DIODE dataset [15]. Mean Absolute Error (MAE) gives the mean error between the predictions and ground truth in meters. It is defined as:

$$\mathrm{MAE}(y,\hat{y}) = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Root Mean Squared Error (RMSE) penalizes errors exponentially, but it may also overrepresent the errors in samples with large depth values. It is defined as:

$$\mathrm{RMSE}(y,\hat{y}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Mean Absolute Relative Error (AbsRel) measures the error as a fraction of the ground truth depth value. It is fully scale-invariant, but can overrepresent small errors in the depth of objects very close to the camera/lidar. It is defined as:

$$\mathrm{AbsRel}(y,\hat{y}) = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{y_i}$$

Mean Absolute Error in $\log_{10}$ space (logMAE) is the same as MAE, but first converts depth values to log space. This can balance the representation of errors in small-scale scenes and large-scale scenes [25]. It is defined as:

$$\mathrm{logMAE}(y,\hat{y}) = \frac{1}{N}\sum_{i=1}^{N}|\log_{10}(y_i+1) - \log_{10}(\hat{y}_i+1)|$$

Root Mean Squared Error in $\log_{10}$ space (logRMSE) combines RMSE and logMAE. It provides a good balance between scale-invariance and penalization of larger errors, so it is likely the best single metric to refer to. It is defined as:

$$\mathrm{logRMSE}(y,\hat{y}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log_{10}(y_i+1) - \log_{10}(\hat{y}_i+1))^2}$$

B. Comparisons

We evaluate our final depth estimation model ("Ours - 2D lidar") alongside three other models: a basic interpolation model, the baseline metrics from Vasiljevic et al. [15], and our main model without the depth input or auxiliary network ("Ours - no lidar").

Our first comparison is between [15] and our no-lidar model. Samples outputs were not provided by [15], but we can compare the metrics, as seen in table I. Our model performs marginally worse on most metrics, but is similar enough to indicate that our models are roughly comparable. The RMSE for our model is significantly higher, but we believe that this is due to a relatively small number of outliers rather than a fundamental difference in the models.

Our next comparison is between those two models and our model with 2D lidar. Our lidar model, as expected, performs better by a large margin. Most error metrics show a 30% to 50% reduction. This shows promise; it is perhaps obvious that the inclusion of lidar will improve it, but it is valuable to get confirmation of this, and particularly of how much better it performs.

The last comparison is an interesting one. Our naive baseline model, which we call our interpolation model, is very simple. It takes in the 2D lidar scan, filters out the unreliable data points, fills the gaps using basic linear interpolation and extrapolation as usual, and stretches the scan vertically—which can be thought of as nearest-neighbor interpolation. There is absolutely no RGB information present anywhere in this process. Surprisingly, this model outperforms all other models in all metrics. It outperforms by the largest margin in MAE, and by the smallest margin in log RMSE.

The implication of this is clear: there is significant room for improvement in our method. The interpolation model is used as a preprocessing step for the input to our 2D lidar model, so in theory our model could get the same results as the interpolation model by just passing the depth scan through unchanged. However, the depth information is getting overshadowed by other sources of information (RGB image) and noise (in the data and in the randomness of the model). This supports the method used by [10], where the depth scan is arithmetically added to the end of the model, rather than merely being concatenated and passed into a few more convolutional layers. In theory, our model should be able to outperform interpolation by a decent margin. Both the lidar and no-lidar models can estimate many depth features correctly, so at minimum they should be able to refine the interpolated scan. We hypothesize that one of the main issues is a lack of training data.

Note that these metrics do not show the whole picture. Depending on the specific application of the depth estimation system, it may be more important to predict structural relationships between different areas of a scene, rather than predict every pixel independently as accurately as possible. We will go into a qualitative analysis in the next section.

C. Qualitative Analysis

See figure 7 for 8 selected samples from the validation set, and the predictions of our three models on those samples.

In the first row, we see a sample where the interpolation model really shines. Inside a building, there will frequently be little variation vertically, so it is not surprising that simply stretching a horizontal scan vertically can sometimes give excellent results. This sample is one of the best examples of that. Our lidar model gets similar results, but with a bit more noise. Our model with no lidar has no idea that there is a doorway, and is thus completely incorrect.

In the second row, we see a sample where our model with no lidar performs well. Unlike the model with lidar, it accurately predicts the boundaries of the window/door frame, and gives reasonable estimates for the relative distance between the near and far walls. The interpolated scan does not see the top of the doorframe, and the full lidar model does not correct it.

The third, fourth, and fifth rows are examples of the lidar model performing well. It is not perfect, but it recognizes the structure of the room it is in. Of particular note is the fourth row, where it accurately predicts the depth of the fern.

The sixth row shows a fundamental issue with the no-lidar model. Since it is indoors, the dataset is biased towards small distances of just a few meters. Without any ground truth to to correct it, the model leans on that bias, and severely underestimates the depth of most of the room. The lidar model is still biased in a similar way, but the ground truth readings cause this bias to have a much lesser effect.

The seventh row is an example where all models fail in varying ways. On the left side of the ground truth, there are three very clear and distinct areas of depth. The interpolation model, as expected, predicts everything to be as far as the center of the image. The two neural networks are able to accurately predict the top of the doorframe, but do not accurately predict the bottom of the window.

In the last row, we see an example of severe failure in the lidar model. The extreme values in a repetitive pattern are an artifact of the convolutional architecture, and indicate an issue with training. We suspect that this failure mode is caused by the overfitting, and could be completely eliminated by stopping the training just before it starts overfitting.

VI. Conclusion

In this work, we demonstrate the viability of the inclusion of 2D lidar to improve the accuracy of a depth estimation neural network. We see a roughly 40% reduction in error metrics, which can be worth the added cost and weight of a 2D lidar unit in many situations. However, there is also significant room for improvement. A basic interpolation model outperforms our neural net, and a lack of a large enough high-quality dataset makes it difficult to train our model for long without overfitting. With a better dataset, more time spent tuning hyperparameters, and experimentation with new techniques, we could see vast improvement.

In the future, this work could be extended by training on other datasets such as KITTI [13] and NYUv2 [13], or by training and/or evaluating on the outdoor half of the DIODE dataset. The model architecture could be changed to include spatial propagation techniques like in [5], and other more minor changes could be made, such as berHu loss [2] or some form of batch or layer normalization. Lastly, training and evaluating on video data has the potential to drastically improve results, and more accurately simulate what is possible in the real world.

Acknowledgment

Thank you to my advisor, Dr. [redacted], for his advice during the project. Thank you to my friends and family for their support throughout, and for proofreading the final drafts.

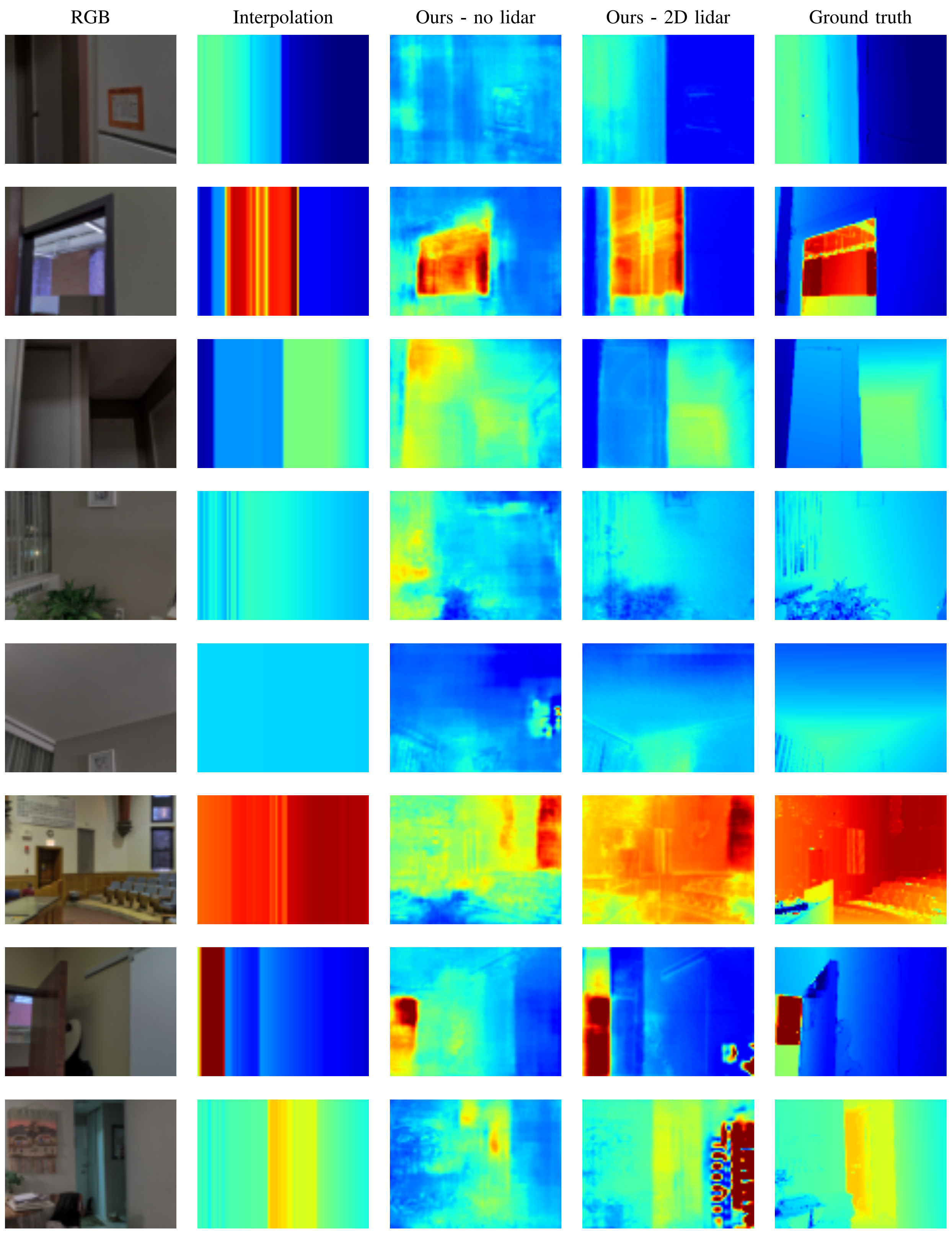

Fig. 7. Selected samples. The first column is the RGB input to the model. The next three columns are the depth predictions from our models, and the last is the ground truth depth map. The depths are indicated by color: blue is near, red is far, and green/yellow/orange are in between.

References

[1] D. Eigen, C. Puhrsch, and R. Fergus, "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[2] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," *Proceedings - 2016 4th International Conference on 3D Vision, 3DV 2016*, pp. 239–248, dec 2016.

[3] Y. Cao, Z. Wu, and C. Shen, "Estimating Depth from Monocular Images as Classification Using Deep Fully Convolutional Residual Networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 11, pp. 3174–3182, nov 2018.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] X. Cheng, P. Wang, and R. Yang, "Depth estimation via affinity learned with convolutional spatial propagation network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 103–119.

[6] F. Mal and S. Karaman, "Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4796–4803, sep 2018.

[7] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-supervised sparse-to-dense: Self-supervised depth completion from LiDAR and monocular camera," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 3288–3295, may 2019.

[8] S. Lee, J. Lee, D. Kim, and J. Kim, "Deep Architecture with Cross Guidance between Single Image and Sparse LiDAR Data for Depth Completion," *IEEE Access*, vol. 8, pp. 79 801–79 810, 2020.

[9] V. Guizilini, R. Ambrus, W. Burgard, and A. Gaidon, "Sparse Auxiliary Networks for Unified Monocular Depth Prediction and Completion," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 11 073–11 083, mar 2021. [Online]. Available: https://arxiv.org/abs/2103.16690v1

[10] Y. Liao, L. Huang, Y. Wang, S. Kodagoda, Y. Yu, and Y. Liu, "Parse geometry from a line: Monocular depth estimation with partial laser observation," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5059–5066, jul 2017.

[11] A. Saxena, M. Sun, and A. Y. Ng, "Make3D: Learning 3D scene structure from a single still image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 824–840, 2009.

[12] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[13] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[14] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European conference on computer vision*. Springer, 2012, pp. 746–760.

[15] I. Vasiljevic, N. Kolkin, S. Zhang, R. Luo, H. Wang, F. Z. Dai, A. F. Daniele, M. Mostajabi, S. Basart, M. R. Walter, and G. Shakhnarovich, "DIODE: A Dense Indoor and Outdoor DEpth Dataset," *CoRR*, vol. abs/1908.00463, 2019. [Online]. Available: http://arxiv.org/abs/1908.00463

[16] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *2010 IEEE Computer Society Conference on computer vision and pattern recognition*. IEEE, 2010, pp. 2528–2535.

[17] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[18] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 648–656.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[20] S. Ioffe, "Batch renormalization: Towards reducing minibatch dependence in batch-normalized models," *Advances in neural information processing systems*, vol. 30, 2017.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[22] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[23] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] A. Mertan, D. J. Duff, and G. Unal, "Single Image Depth Estimation: An Overview," *Digital Signal Processing*, vol. 123, p. 103441, apr 2021. [Online]. Available: http://arxiv.org/abs/2104.06456http://dx.doi.org/10.1016/j.dsp.2022.103441