**University of Toronto**

**Faculty of Applied Science & Engineering**

**APS360: Artificial Intelligence Fundamentals**

**Project Progress Report**

Monocular Depth Estimation

**Team 1**

Akram Khan

Brandon Wu

Darie Roman

Nada Al Aker

Table of Contents

[**1.0 Project Description**](#_bxshq0gpgkgm) **2**

[**2.0 Summary of Work Done**](#_77nzxnii9yxx) **3**

[2.1 Data Collection and Processing](#_vz1ejat4immp) 3

[2.2 Baseline Model Design](#_luhxiqws7we9) 5

[2.3 Model Architecture](#_nzaxqwu2h7x9) 6

[**3.0 Results**](#_ocvaar1eb7ot) **7**

[**4.0 Project Progress**](#_z3dt089i8q9w) **9**

[**5.0 Next Steps**](#_f43fznv8rn0t) **10**

[**6.0 Summary**](#_dne8z39cdsry) **10**

[**7.0 References**](#_xjatjofnevjj) **11**

### 

### 

### 1.0 Project Description

*Table 1: Overview of Project*

| Motivations | Depth estimation is particularly relevant in the perception of environments by autonomous systems. One commonly used method is stereo depth estimation, which utilizes two cameras to triangulate and estimate distances. However, stereo depth estimates tend to be inaccurate for larger distances, since small triangulation errors can result in large errors in distances [1]. On the contrary, monocular depth estimation utilizes visual cues such as texture variations and gradients, and produces results of higher accuracy when compared with stereo depth estimation in situations where separation between the two cameras is limited [1]. Our team undertook monocular depth estimation through machine learning as it has a significant cost advantage over traditional monocular depth measurement techniques such as LIDAR and RGB-D cameras [2]. Additionally, the large power consumption and size constraints of depth sensors (LIDAR and RGB-D cameras) render them ineffective for small-scale applications such as drones and small robots [2]. Thus, machine learning can serve as a useful framework for addressing these concerns. |
| --- | --- |
| Goal | The goal of the project is to train a neural network that will generate a depth map from an RGB image. The input will involve an RGB image with the corresponding depth map as the output (See Figure 1). |
| Importance | Estimating the depth of RGB images can be useful in determining their geographic representation [3]. The applications of such a project are especially interesting, as depth estimation is utilized in robotic systems, augmented reality (AR) and autonomous vehicles. Depth is a major requirement for tasks such as perception, navigation and planning [4]. |
| Relevance | With recent improvements in machine learning networks, monocular depth estimation using deep learning has seen significant improvements in accuracy. Machine learning is a reasonable approach as variations on this project have been explored in the past, and there tends to be global and local cues such as object locations and alignments that a neural network could use to generate the depth map. |



*Figure 1: An RGB image with its corresponding output depth map to be generated through machine learning*

### 2.0 Summary of Work Done

#### 2.1 Data Collection and Processing

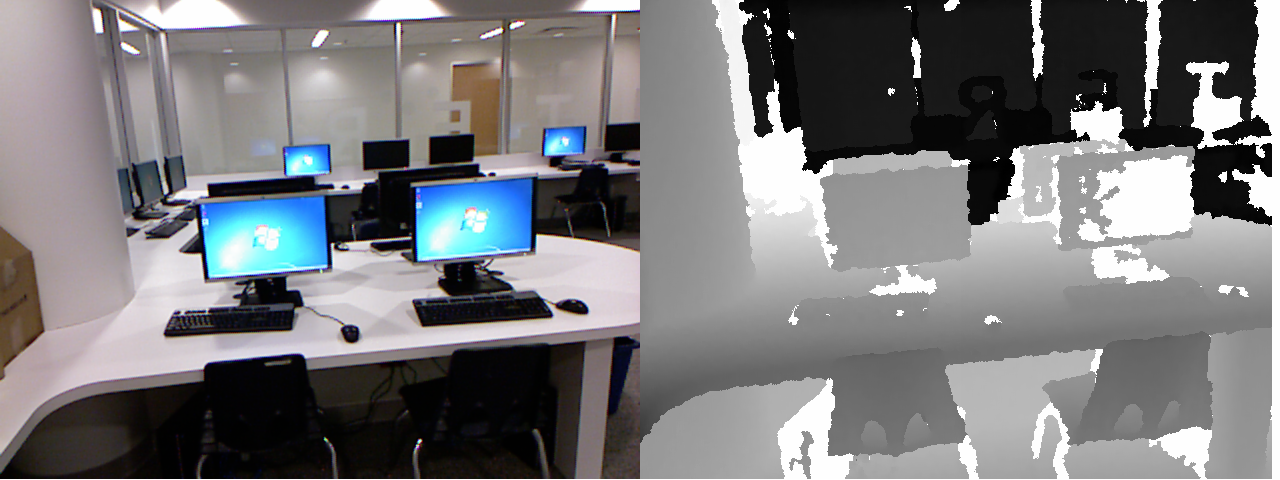
Our training dataset consists of the NYU-Depth V2 data set, which contains over 400,000 images and depth

maps for different indoor scenes captured at a resolution of 640 x 480 [5]. Since the scope of this project

only involves raw depth generation, we cleaned the NYU dataset to exclude additional processing.

Additionally, the NYU dataset contains images of different environments including basements, bedrooms, etc.

Images pairs were selected from a diverse range of environments in order to help the model better generalize.



*Figure 2: An RGB (480x640) image with its corresponding depth map, where darker is closer. The depth sensor is shown to struggle with transparent, reflective, and emissive surfaces.*

*Table 2: NYU Dataset Scenes*

| **Scene** | **# of Image Pairs** |
| --- | --- |
| Computer Labs | 1905 |
| Conference Rooms | 1428 |
| Dentist Office | 3046 |
| Dinette | 740 |
| Exercise Room | 903 |
| Foyers | 724 |

The RGB and Depth images were converted into a single HDF5 binary file in order to compress the data and increase loading speeds. Each individual image is stored with its own key that includes a marker for input and output, as well as an index that is shared with the pair. For example, the RGB image at key “rgb10035” would be paired with the Depth image at key “depth10035”.

The PyTorch Dataset class was overwritten to extract the pictures from this file and to convert them into a usable format, as well as normalizing all pixel values between 0 and 1. This was done by storing the RGB and depth keys separately in lists, sorting them, and correlating each one with an index in the custom dataset. Whenever a data point is requested from the custom Dataset object, the index provided is used to fetch a matching RGB and Depth image pair, normalize it, and then return it.

After the dataset object is created, it is split with numpy’s random\_split by the specified ratios into training, validation, and testing datasets. Three DataLoaders are then created from these with shuffling and a specified batch size in order to optimize GPU memory utilization.

It is worth noting that the NYU dataset contained significant noise and had missing features in the ground truth depth maps. Additionally, the depth images struggled to accurately represent depth with transparent, reflective, and emissive surfaces, which was observed to seriously hinder any training progress. To address this, we plan on replacing the NYU dataset with the DIML dataset, which contains depth images generated from both a Kinect v2 and a Zed stereo camera [5]. The DIML dataset exhibits significantly improved depth images with reduced noise. To clean the dataset, we intend to select images from a diverse range of environments to avoid generalization on a particular environment. Additionally, the RGB and depth images will be stored in a HDF5 format to compress data and reduce loading time.

#### 

#### 2.2 Baseline Model Design

The intuition behind the baseline model was to make it as simple and efficient as possible, whilst conserving the final image’s dimensions. It is merely constructed from 4 convolutional layers. The padding complements the kernel size to maintain the dimensions.

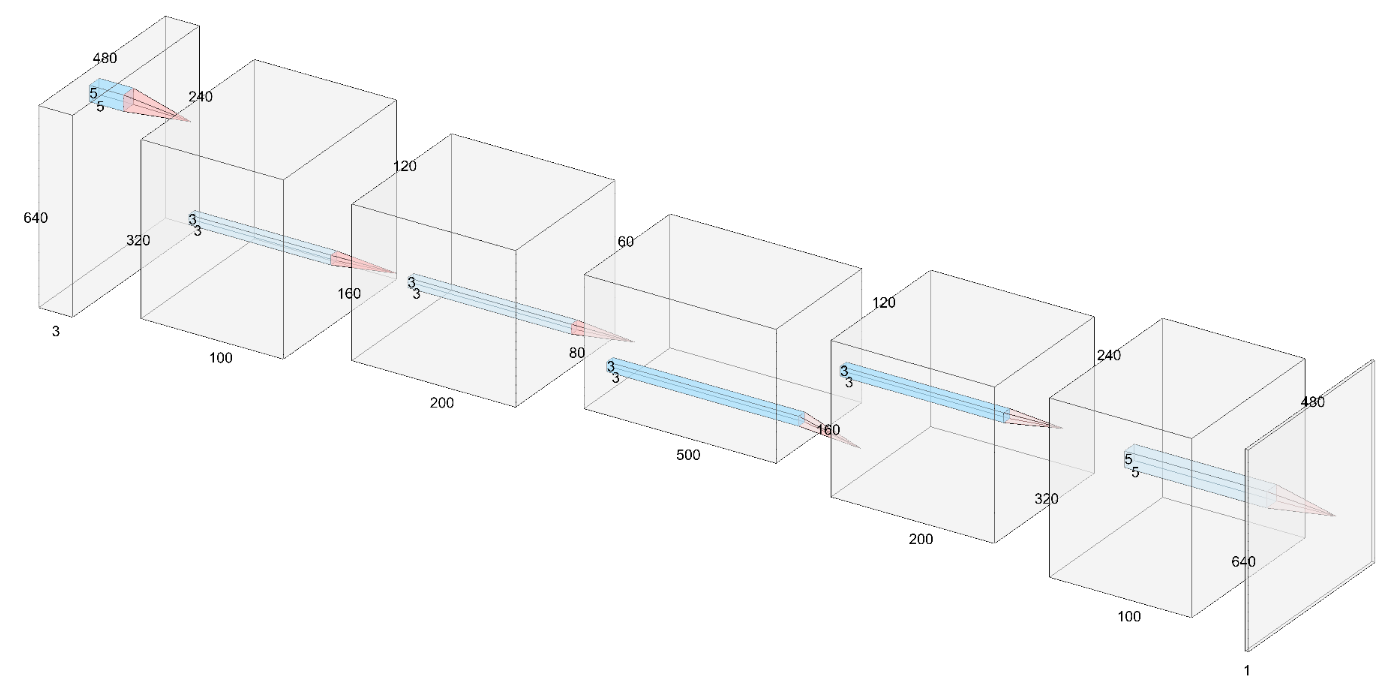
*Figure 3: Baseline model code*

The baseline model constitutes a reasonable architecture for this problem, as it is capable of detecting patterns within images efficiently, without any extra complicated architecture additions.

*Table 3: Baseline Model Results*

| **RMSE** | **RGB** |
| --- | --- |
| Lowest error (validation): 0.370 |  |
| **True Depth** | **Predicted Depth** |
|  |  |

#### 2.3 Model Architecture



*Figure 4: Diagram of model architecture, composed of Convolutional and Convolutional Transpose layers, along with SELU activation function in between.*

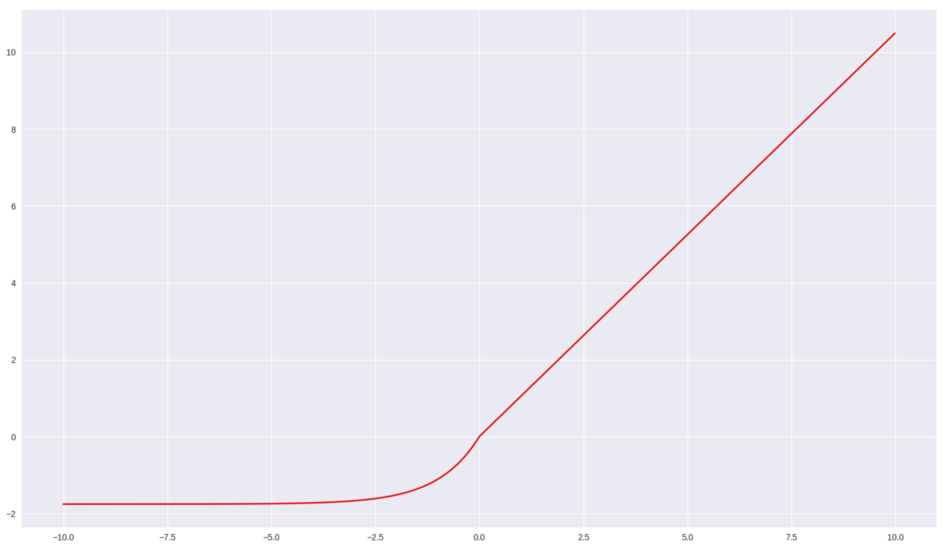


*Figure 5: Example input and output images of model in Figure 4.*

The chosen network architecture (Model A) is a convolutional autoencoder where the compression and subsequent expansion are done using stride. There are three layers of compression using Conv2d with a stride of 2 where the image dimensions are halved every time, and three steps of expansion using ConvTranspose2d, also with a stride of 2, resulting in a doubling of the dimensions every time. The number of channels are smoothly increased and decreased to allow the network to learn an efficient embedding. A kernel size of 5 is used for the first and last layers, where the image is larger. For all other layers, a kernel of size 3 is sufficient. The padding is always set in such a way as to counteract any sizing changes caused by the kernel size, making the dimensions easy to divide and then multiply back: . This results in the following image dimensions throughout the layers, where N is the batch size:

[Nx480x640x3] -> [Nx240x320x100] -> [Nx120x160x200] -> [Nx60x80x500] ->

[Nx120x160x200] -> [Nx240x320x100] -> [Nx480x640x1]



Between the layers, a SELU activation function is applied, which is similar in behaviour to RELU, but switches to an exponential function instead of 0. This activation function has been observed to self-normalize the parameters of CNNs and lead to improved performance [6].

**Total parameters: 2.17M**

*Figure 6: SELU activation function*

### 3.0 Results

As can be seen in Table 4 on the next page, Model A was able to achieve a mediocre result with the lowest validation error reaching a value of 0.246, which was a 33% reduction in error in comparison with the baseline model. However, from a qualitative standpoint, there were visible discrepancies between the generated depth maps and ground truth; Model A appeared to capture the shapes present in the original image, but occasionally failed to accurately gauge the depths of objects that were too dark or too bright. Observing the training curve of Model A, it appears that the data is unrepresentative and may have inconsistencies. Judging by the noise and other issues in ground truth depth maps, this hypothesis is further confirmed. Many hyperparameter combinations were attempted, but the training curve remained sporadic in all instances.

*Table 4: Comparison of Model Results*

| **Architecture**  **Model** | **Model B: MaxPool, Interpolation, ReLU** | **Model A : Double stride, SELU** |
| --- | --- | --- |
| **RMSE**  **Graph** | Lowest error (validation): 0.280 | Lowest error (validation): 0.246 |
| **RGB** |  |  |
| **True**  **Depth** |  |  |
| **Predicted Depth** |  |  |

### 4.0 Project Progress

Thus far, we have found that we often missed our internal deadlines by a day or two, although this can partially be attributed to some deadlines falling in the exam period. By working in 4 separate Google Collab files, the team can branch out and explore many architectures without overwriting each others’ work. Through the usage of Facebook messenger and a shared Google Drive folder, we are able to share the parts of our code that are reusable (i.e. data processing and visualization) and keep team members up to date on what we are currently working on.

*Table 5: Internal deadlines set by the team over the course of the project*

| **Task** | **Internal Deadlines** | **Person(s) Responsible** | **Date Finished** | **Notes/Status** |
| --- | --- | --- | --- | --- |
| Collect Data | June 15th | Darie, Nada | June 20th | Done |
| Clean Dataset | June 18th | Darie | June 20th | Done; dataset has visible noise |
| **Baseline Model Design Alternatives** | | | | |
| Baseline Model Design A | June 28th | Nada, Darie | June 26th | Done |
| Baseline Model Design B | June 28th | Brandon, Akram | - | Scrapped because Baseline Model A was satisfactory |
| **Design Alternatives** | | | | |
| Model A: Convolutional Autoencoder with double stride and Transfer Learning | July 7th | Nada, Darie | July 10th | Finished model architecture; have yet to fully incorporate transfer learning |
| Model B: Convolutional Autoencoder with Progressive Resizing and Interpolation | July 7th | Brandon, Akram | July 12th | Base model is done, progressive resizing has yet to be implemented |
| **Progress Report** | | | | |
| Initial draft of Progress Report | July 9th | Nada (Started) | July 10th | Document Skeleton, with comments on section contents. |
| Final Revision of Progress Report | July 11th | Team | July 12th | Done |

### 

### 5.0 Next Steps

Our next steps will involve processing the new dataset to have more consistent training, and incorporating advanced techniques into the existing model architectures.

*Table 6: Updated plan for the month of July*

| **Task** | **Person(s) Responsible** | **Deadline** | **Redundancies** |
| --- | --- | --- | --- |
| Obtain DIML Dataset, clean and convert to .h5 file | Darie | July 13th | If Darie is unable to complete the cleaning and conversion process, Brandon will take over. |
| Implement ResNet encoder on Model A | Nada, Darie | July 20th | If ResNet feature extraction fails to improve results, can attempt training the weights of a smaller network (i.e. MobileNet) instead of using pretrained weights. |
| Implement progressive resizing on Model B | Brandon, Akram | July 20th | If progressive resizing fails to yield any improvements, may attempt to switch network architecture to U-Net |
| Initial Draft of Project Report | Nada (outline) | July 25th | If Nada can’t write the outline, Darie will take over. |
| Prepare Initial Presentation | Team | July 25th | If one team member is unable to complete their portion, the work will be redistributed to the other members of the team |

### 6.0 Summary

The scope of this project involves depth map generation from RGB input images by utilizing neural networks. Machine learning serves as a useful framework for monocular depth estimation as it eliminates the cost, size and weight constraints of traditional depth estimation techniques such as LIDAR and RGB-D cameras. A baseline model was created consisting of a simplistic CNN architecture to extract visual cues in the dataset. Additionally, two model architectures were created based on an autoencoder structure. Model A, consisting of a SELU activation function and double stride achieved significantly better results with a validation error of 0.246 over model B, which consisted of max pooling layers, a RELU activation and interpolation and achieved a validation error of 0.280. Future work for this project involves implementing a ResNet encoder on model A for improved feature extraction and progressive resizing on model B to improve depth map estimation.

### 7.0 References

[1] S. Ashutosh, “Depth Estimation Using Monocular and Stereo Cues,” Cornell. [Online]. Available: <http://www.cs.cornell.edu/~asaxena/learningdepth/DepthEstimation_Saxena_IJCAI.pdf>[Accessed: 8-Jul-2020]

[2] C. Zhao, “Monocular Depth Estimation Based on Deep Learning: An Overview.” [Online]. Available: <https://arxiv.org/pdf/2003.06620.pdf> [Accessed: 8-Jul-2020]

[3] D. Mwiti, “Research Guide for Depth Estimation with Deep Learning,” Medium, 11-Feb-2020. [Online]. Available: https://heartbeat.fritz.ai/research-guide-for-depth-estimation-with-deep-learning-1a02a439b834. [Accessed: 8-Jul-2020].

[4] D. Tan, “Depth Estimation: Basics and Intuition,” 01-Mar-2020. [Online]. Available: https://towardsdatascience.com/depth-estimation-1-basics-and-intuition-86f2c9538cd1. [Accessed: 8-Jul-2020].

[5] “DIML,” DIML / CVLAB RGB D Dataset. [Online]. Available: https://dimlrgbd.github.io/#section-overview. [Accessed: 13-Jul-2020].

[6] Medium. 2020. SELU — Make Fnns Great Again (SNN). [Online] Available at: https://towardsdatascience.com/selu-make-fnns-great-again-snn-8d61526802a9 [Accessed 9 July 2020].