**University of Toronto**

**Faculty of Applied Science & Engineering**

**APS360: Artificial Intelligence Fundamentals**

**Project Final Report**

Monocular Depth Estimation

**Team 1**

Akram Khan

Brandon Wu

Darie Roman

Nada Al Aker

Word Count: 2253

Table of Contents

[**1.0 Introduction**](#_z564n6xb0d1j) **3**

[1.1 Motivation](#_ll02hhwlnnsn) 3

[1.2 Importance and Goals](#_60y74aflqzgx) 3

[**2.0 Background and Related Work**](#_z564n6xb0d1j) **3**

[**3.0 Data Collection and Processing**](#_z564n6xb0d1j) **4**

[3.1 Data Acquisition and Sorting](#_pfy7pugnw5h5) 4

[3.2 Data Augmentation](#_8hggspvie9sq) 5

[**4.0 Architecture**](#_z564n6xb0d1j) **5**

[4.1 Baseline Model](#_sattfe81j7i) 5

[4.2 Primary Model](#_l4r9cqsarip) 6

[**5.0 Results**](#_araif4vtps8a) **7**

[5.1 Quantitative Results](#_8x13cmstytwc) 7

[5.2 Qualitative Results](#_8x13cmstytwc) 8

[**6.0 Testing on New Data**](#_ko0zn3gbzuua) **9**

[**7.0 Ethical Considerations and Conclusion**](#_r39j45py5x0l) **10**

**8.0 References 11**

**9.0 Contributions 12**

9.1 Individual Contributions  **12**

9.2 Final Work Distribution  **15**

### 1.0 Introduction

#### 1.1 Motivation

Depth estimation is relevant in situations where the perception of the nearby spatial environment is needed, such as in robotics and autonomous systems. Stereo depth estimation 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].

In contrast, 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 infrared dot matrix systems [2]. Additionally, the large power consumption and size constraints of depth sensors render them ineffective for small-scale applications such as drones and small robots [2]. A common alternative is stereo depth reconstruction, but this method suffers from degraded performance when reconstructing a scene with reflective surfaces. [1] Thus, machine learning can serve as a useful framework for addressing these concerns.

#### 1.2 Importance and Goals

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.

The primary goal of the project is to train a neural network to generate a depth map output from an RGB image as an input. 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 [3].

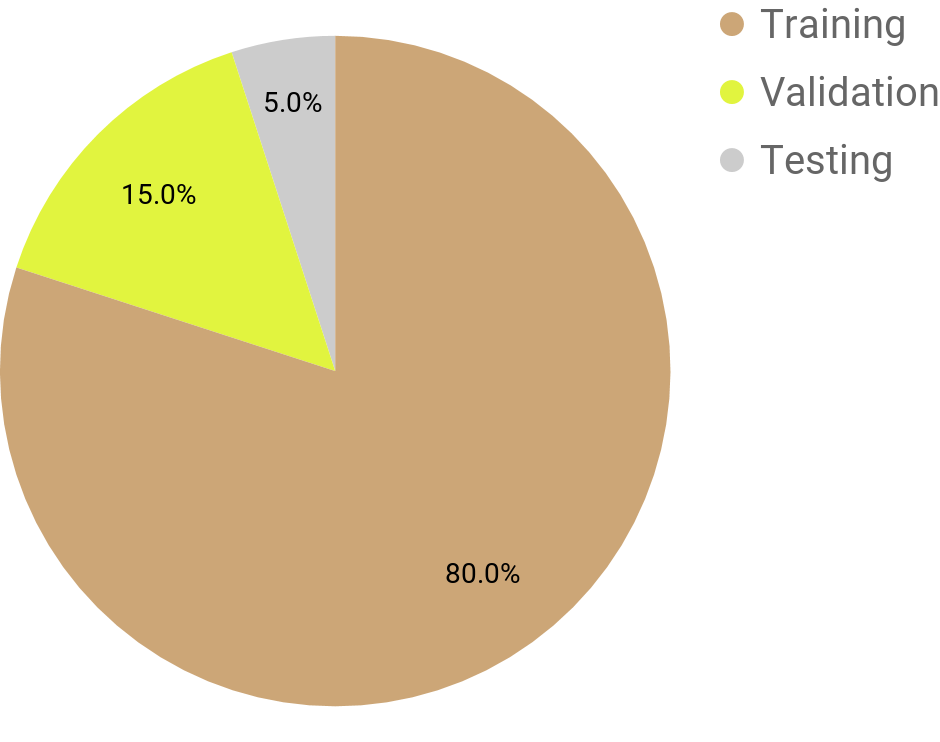
### 2.0 Background and Related Work

To obtain the dataset for depth mapping on RGB images, the NYU-Depth data set is widely utilized. The NYU depth dataset consists of RGB and depth video sequences from several indoor scenes recorded by the Microsoft Kinect [3]. However, in our experience, the NYU dataset contained significant noise and performed poorly in regions with shadows and reflective surfaces. Further research revealed the usage of the DIML (Digital Image Media Lab) RGB-D dataset to train convolutional neural networks for semi-supervised monocular depth estimation [4]. The DIML dataset contains over 200 indoor and outdoor scenes, with RGB-D frames captured using both a Kinect v2 and a Zed stereo camera [5].

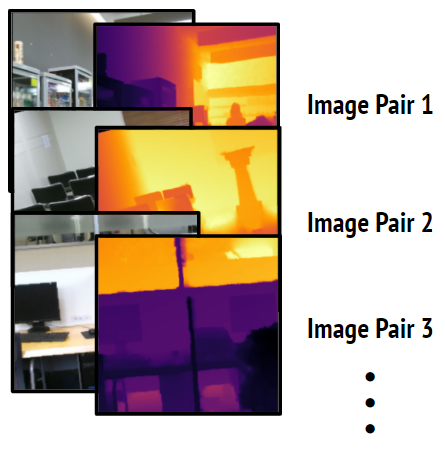
Based on our research into monocular depth estimation, several proposed solutions make use of a convolutional architecture to generate a depth map given an RGB image as input. One such implementation utilizes DenseNets or deeply connected convolutional layers with pretrained weights to power the encoder. The decoder consists of 2x bilinear upsampling layers followed by two convolutional layers [6]. The loss function considered the difference between the ground truth map and the predicted depth map. Since the loss function for depth generation can have significant variations, the paper defined the loss as a weighted sum of three loss functions; the first loss function described the loss between the actual and predicted depth, the second loss term was defined over the gradient of the actual and predicted depth and the last term used structural similarity (SSIM) to define the loss term [6].

### 3.0 Data Collection and Processing

#### 3.1 Data Acquisition and Sorting

The dataset we used was downloaded from the Digital Image Media Lab from Yonsei University in South Korea, which provides RGB videos of various indoor and outdoor scenes with corresponding depth maps in the form of individual image frames. In order to decrease the required storage size, five scenes from the indoor section were used. Additionally, since frames that are close together tend to be almost identical, we discarded four out of every five frames, resulting in six frames of depth per second of source footage.

| **Scene** | **No. of Image Pairs** |
| --- | --- |
| Bathroom | 228 |
| Church | 991 |
| Computer Room | 897 |
| Store | 1251 |
| Warehouse | 244 |
| **Total** | 3611 |

*Table 1: DIML dataset scenes Figure 1: Dataset splitting ratio*

Every RGB image was paired with its corresponding depth map, after which the entire dataset was encoded into a single hdf5 binary file in order to improve loading performance while compressing the data. One problem we initially encountered when loading the data was that the shuffling process would place similar video frames across the training, validation, and testing subsets, thus inaccurately reflecting the model’s ability to generalize during testing as it may have already been trained on a similar frame. The solution was to first split the chronologically sorted dataset into its three subsets, and shuffle each individual subset. This way, each subset receives different scenes, and in the case where one scene exists in two subsets, the frames are from two separate times and thus have different compositions.

#### 3.2 Data Augmentation

To circumvent the issue of having a relatively low number of remaining video frames, data augmentation was performed on the training images, allowing many new frames to be created, and increasing the model’s robustness on rotated videos. The process is as follows:

1. Random rotation with max angle of 15°
2. Random horizontal flip with probability of 50%
3. Centre crop of size 400x533 (just enough to remove black areas resulting from rotation)
4. Resize to 225x300 (allows crop in next step to have a wider field of view)
5. Random crop an area of 224x224

Any random numbers generated were shared between both RGB and depth in order to ensure identical frame placement in every image pair. A simulation of this data augmentation process can be seen in Figure 3.

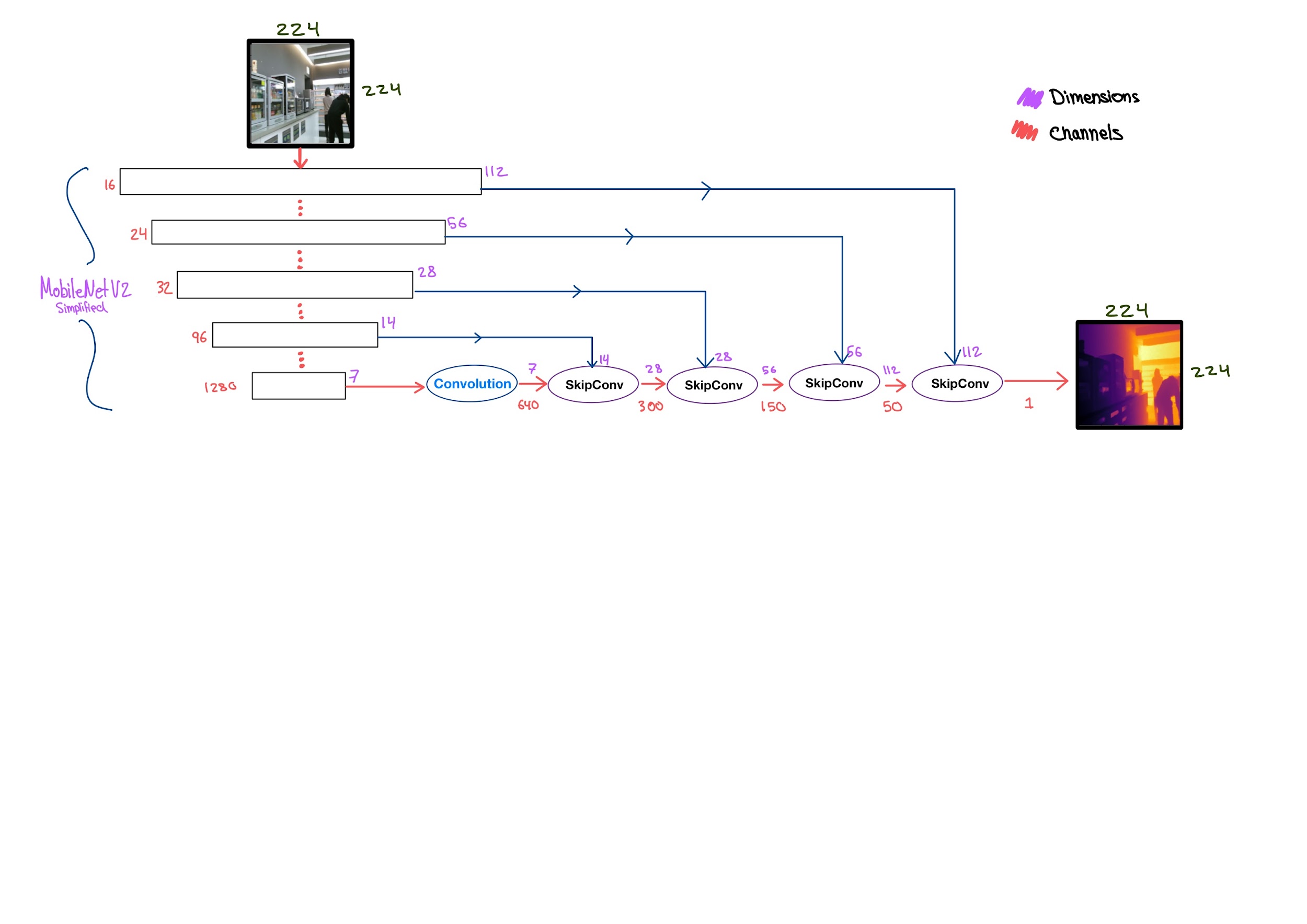
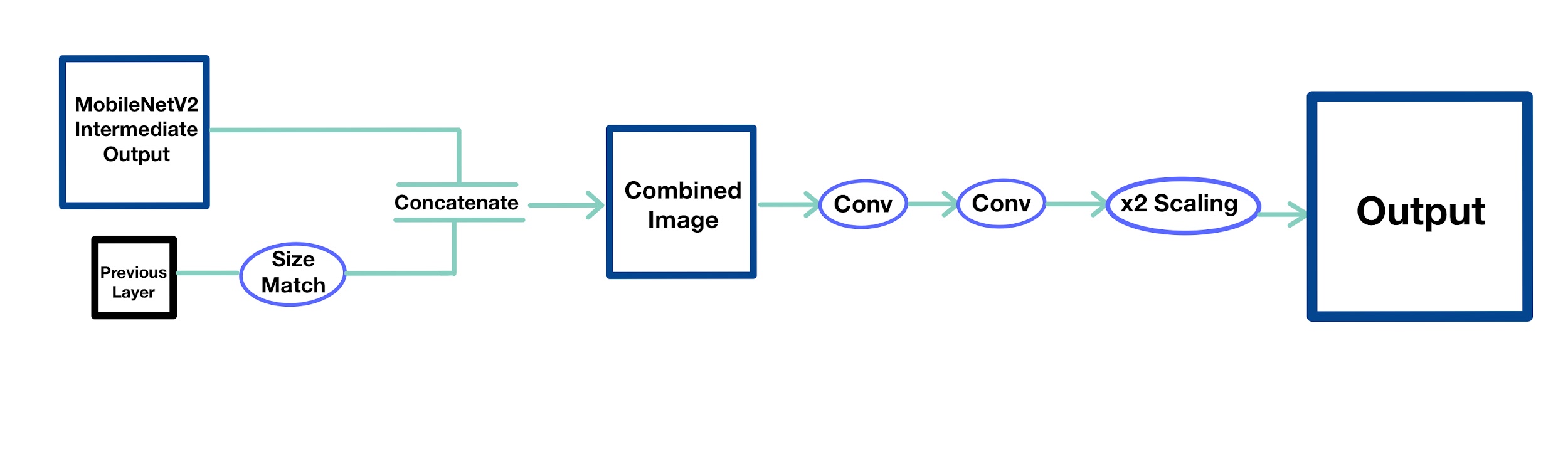
*Figure 3: Frame extraction from original image during data augmentation*

### 4.0 Architecture

#### 4.1 Baseline Model

Producing a depth map from RGB requires a system that can transform an image, while taking into account various patterns and structures to understand depth. Therefore, the baseline chosen to compare our final model against is a simple 4-layer CNN that preserves the image size in order to output an appropriately sized depth map. The increasing and then decreasing number of channels allows plenty of different patterns to influence the transformation from RGB to depth, and gives the model enough learning capacity to be able to map the input and output. It was experimentally observed that a large kernel size as seen in Figure 4 allows for more complex structures to be detected in the input image and also feature maps, while the padding was selected in order to maintain the image’s height and width.

#### 4.2 Primary Model

*Figure 5: Convolutional autoencoder utilizing transfer learning and skip connections*

*Figure 6: SkipConv Module that facilitates a skip connection, two convolutions, and upscaling*

The chosen network architecture shown in Figure 5 is a convolutional autoencoder that encodes the RGB image using the pretrained MobileNetV2 architecture, and decodes it into a depth map using a custom module that performs convolutions, upscaling, and makes use of skip connections. As can be seen in Figure 6, the Skip Connection Module takes as input the previous layer and also an intermediate MobileNetV2 layer. The small previous layer is enlarged with interpolation upscaling to match the MobileNetV2 layer if necessary, which occurs in the first block only, after which the now equally sized layers are concatenated together. The combined image undergoes two layers of convolutions and is further upscaled before being fed into the next block. The encoder outputs 1280 feature maps from its final layer, which are gradually decreased down to one depth channel.

The way in which encoder and decoder layers connect together allows for various compression levels of the image to pass through directly to the trainable decoder. The benefit is that the decoder can learn what to do with the structural information and detail from the less altered first layers, as well as the more abstract feature maps in the final layers. Depth maps can thus include accurate depth for general areas in the image, as well as finer details that would not have persisted through the complete MobileNetV2 encoding process. This is especially important since the encoder parameters are left fixed in order to decrease GPU memory usage and training time.

### 5.0 Results

#### 5.1 Quantitative Results

| **Baseline Model** | **Final Model** |
| --- | --- |
| Lowest error (training) - 0.141  Lowest error (validation) - 0.112  Training Time - 5:04 | Lowest error (training) - 0.054  Lowest error (validation) - 0.083  Training time - 21:47 |

*Table 2: Quantitative performance comparison of Baseline Model with Final Model*

As is the standard for monocular depth estimation, we chose to measure error by finding the root mean squared error of the estimated depth map on a pixel by pixel basis. As shown above, the final model offered a significant improvement when compared to the baseline. We also decided to measure the training time of our models as we tried to make a model that would perform well under the resource constrained circumstances; in this regard, the final model’s training time was significantly lower than other state of the art models, which typically trained for upwards of 10 hours on multiple GPUs.

#### 

#### 5.2 Qualitative Results and Discussion

| RGB Image | Ground Truth Depth | Baseline Model | Final Model |
| --- | --- | --- | --- |
|  | | | |
|

*Table 3: Depth image comparison generated by Baseline Model and Final Model*

The final model, consisting of a skip connections autoencoder coupled with MobileNetV2 for transfer learning, achieved its primary goal of depth estimation extremely well. As illustrated in Table 3 above, the final depth map portrays objects at a closer distance with a darker color. While the baseline model extracts structural features in the environment better compared to the final model, it fails to accurately identify the depth within images. The final model replicates the ground truth depth image significantly better compared to the baseline model.

Unfortunately, one area where our model’s performance was lackluster was its distinction of hard edges in the generated image. Our model tended to understand the greater depth relationships, but the depths for each object tended to “bleed” over into nearby sections, which ultimately meant that areas with many tightly interwoven objects tended to show up as one large mass.

Overall, our model has exceeded our expectations, as not only is it able to accurately assess the environment and generate a good—albeit slightly blurry—depth map, it also generalizes well to new data as shown below. The depth map containing the dog and the human hand exemplifies this, as neither share many similarities with the room-based dataset the model was trained on. Skip connections also proved to be invaluable at preserving structure, and overall led to a great improvement in our model’s performance.

### 

### 6.0 Testing on New Data

The final and baseline models were tested on new data consisting of personal images taken from indoor scenes and from the NYU dataset. The indoor scene images were fed into pretrained baseline and final models to generate the output depth map. The final model performed extremely well on the input images; to demonstrate, the hands in front of the dog are the closest to the camera. This is indicated by darker pixels in the final model depth image. Additionally, the depth image correctly identifies the background objects behind the dog with lighter pixels. In contrast, the baseline model fails to accurately identify depth, as it shows the background with darker pixels while the nearer structural items consisting of the dog and hands are displayed with lighter pixels.

We conducted testing on the NYU dataset to have a ground truth image to compare the outputted depth map with. The performance of the final model is clearly demonstrated; in both input images the bed is the nearest item and is correctly displayed with a darker hue in the outputted depth image. A comparison with the ground truth depth image also demonstrates the short-comings of our model. Structural objects, for instance, the bed, can be clearly distinguished in the ground truth depth image, however, they are unclear in the depth image generated by the final model.

**Personal Images**

| **RGB Image** | **Predicted Depth Image** | |
| --- | --- | --- |
| **Baseline Model** | **Final Model** |
|  |  |  |
|  |  |  |

*Table 4: Depth image testing on Baseline Model and Final Model using personal image dataset*

**NYU Dataset**

| **RGB Image** | **Ground Truth**  **Depth Image** | **Predicted Depth Image** | |
| --- | --- | --- | --- |
| **Baseline Model** | **Final Model** |
|  |  |  |  |
|  |  |  |  |

*Table 5: Depth image testing on Baseline Model and Final Model using NYU dataset*

### 7.0 Ethical Considerations and Conclusion

It is important to note the difficulty in applying a monocular depth estimation machine learning model to a real world application. Despite our team experimenting extensively with a wide range of models including Autoencoders, General Adversarial Networks (GAN), Cyclical GANs, Conditional GANs, and skip connections, our best model occasionally fails to accurately identify the depth of certain objects in settings where there are a large number of objects in the foreground. In particular, our model struggles to identify structural objects. This may lead to potentially hazardous situations in a real world application such as obstacle detection using depth estimation in self-driving vehicles, especially if the model is not trained on the vehicle’s environment.

Additionally, in scenarios where video depth estimation is replacing more rudimentary sensing methods, there can be privacy concerns if people are unaware of the video recording capabilities of such a system. Device maintenance and operations people, as well as hackers, could possibly exploit footage recorded from the sensor for malicious purposes, especially if the device can be accessed remotely.

Colab Links:

[Final Model - Darie Skip Autoencoder](https://colab.research.google.com/drive/17VgYvFTxkaJQcQcGMDh-ztWaExbfAMVn?usp=sharing)

[Akram Transfer Learning Models](https://colab.research.google.com/drive/1_pTuB4BzKqlY7JfhXdqSaEsCXki9T_f2#scrollTo=ZCqsC6oZYeew)

[NYU Testing on Models Akram](https://colab.research.google.com/drive/1OeBlnmEhwXwXHAjIXXyoaEQQw0qj_n6o)

[Nada's GAN Work](https://colab.research.google.com/drive/1w57FA5M9k-hevYt7tQ27pdV99REZwMKj?authuser=1)

[Brandon's Image Normalization and Video Generation](https://l.facebook.com/l.php?u=https%3A%2F%2Fcolab.research.google.com%2Fdrive%2F1mTkRy19qaBnvK_QUAljdpu26QvKl6Rmq%3Ffbclid%3DIwAR1e4201uPgilZSx9E69aVl_RtTw2PrAqFBCaNAamrtGOE_Dl8D14r7HpiM%23scrollTo%3DXK0RzcBkdUzB&h=AT3XyFwHE1QeauGhy_Gm6L0DYFqFF4MnN5HRiZBFPVdTE1j-xFoJb9mXjXddGF1eukjwViUBoydf3NinI91utvXvk1bCp6jbAwwT4jVKJVxV7k42w2ZfoGlqclMW7ln54Xs6H_ebhObv0xF5klDvow)

### 8.0 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. 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].

[4] Cho and Y. Kim, "A Large RGB-D Dataset for Semi-supervised Monocular Depth Estimation", Arxiv.org, 2020. [Online]. Available: <https://arxiv.org/pdf/1904.10230.pdf>. [Accessed: 5-Jul- 2020].

[5] "DIML / CVLAB RGB+D Dataset", [Dimlrgbd.github.io](http://dimlrgbd.github.io), 2020. [Online]. Available: <https://dimlrgbd.github.io/>. [Accessed: 4- Aug- 2020].

[6] I. Alhashim and P. Wonka, "High Quality Monocular Depth Estimation via Transfer Learning", Arxiv.org, 2020. [Online]. Available: <https://arxiv.org/pdf/1812.11941.pdf>. [Accessed: 04- Aug- 2020].

#### 

#### 

#### 

#### 

#### 9.0 Contributions

### 9.1 Individual Contributions

**Nada Al Aker’s Contribution Summary:**

| **Task #** | **Description** | **Note(s):** |
| --- | --- | --- |
| 1 | Brainstorming on data retrieval, data collection, data processing with Darie (he wrote the code) | 20% contribution |
| 2 | Implemented Autoencoder Architecture and Trained the network on it |  |
| 3 | Cyclical GAN research |  |
| 4 | Conditional GAN implementation and full debugging |  |
| 5 | Wrote and designed the presentation outline and design, and edited the video |  |
| 6 | Progress report outline, and wrote the team plan (same in the proposal), and the outline for the final report |  |
| 7 | Implemented Transfer Learning with multiple networks (mentioned above) for the use of the team such as AlexNet, Resnext and ResNet. |  |
| 8 | Illustrations for Presentation and Reports |  |

**Nada Al Aker’s Incomplete Tasks:**

| **Task #** | **Description** | **Note(s):** |
| --- | --- | --- |
| 1 | I was supposed to help out with the code for data retrieval | 20% Contribution |
| 2 | Didn’t write most of the speaker notes like I expected myself to |  |

**Darie Roman’s Contribution Summary:**

| **Task #** | **Description** | **Note(s):** |
| --- | --- | --- |
| 1 | Data collection and processing, including custom Dataset class (brandon helped with processing) | 90% |
| 2 | Data augmentation |  |
| 3 | Helper functions like: data loading, validation loss, training, GAN training, pair-wise random data augmentation operations | 90% |
| 4 | Research on autoencoders, skip connections, activation functions, and conditional GANs and how to properly train them |  |
| 5 | Skip Convolution modules |  |
| 6 | Architectures: baseline, basic autoencoders, various skip connection autoencoder implementations (including final model), many conditional GANs (and transfer learning in the generator) |  |
| 7 | Data and architecture presentation slides |  |
| 8 | Presentation script | 80% |

**Darie Roman’s Incomplete Tasks:**

| **Task #** | **Description** | **Note(s):** |
| --- | --- | --- |
| 1 | Researching and implementing depth data normalization (Brandon took over) |  |
| 2 | Was not able to train a successful conditional GAN in time |  |
| 3 | The validation loss calculation method I implemented is decent, but could have been a lot better |  |

**Akram Khan’s Contribution Summary:**

| **Task #** | **Description** | **Note(s):** |
| --- | --- | --- |
| 1 | Implemented CNN architecture and conducted hyper-parameter tuning on baseline model |  |
| 2 | Conducted research and implemented Autoencoder architecture |  |
| 3 | Implemented transfer learning techniques on CNN (resNet, DenseNet 121), Autoencoder (resNet, DenseNet, MobileNet) and GAN (resNet and DenseNet, using Nada’s code) |  |
| 4 | Data processing, augmentation for implementation and training of baseline model on NYU dataset |  |
| 5 | Created testing module consisting of data from NYU dataset to implement testing of pre-trained models on new data |  |
| 6 | Researched MobileNets implementation and helped with implementation | 50% with Brandon |
| 7 | Generated depth images from input RGB images from using DIML dataset, NYU dataset and personal images for presentation and report |  |

**Akram Khan’s Incomplete Tasks:**

| **Task #** | **Description** | **Note(s):** |
| --- | --- | --- |
| 1 | Did not implement transfer learning on encoder and skip connections on decoder |  |
| 2 | Conducted research on semi-supervised learning but did not implement model |  |
| 3 | Could have tried improved interpolation technique for autoencoder with transfer learning to reduce blockiness of output images |  |

**Brandon Wu’s Contribution Summary:**

| **Task #** | **Description** | **Note(s):** |
| --- | --- | --- |
| 1 | Implemented demonstration and video processing code for the final presentation |  |
| 2 | Implemented basic autoencoder models | 25% since we each made a model |
| 3 | Did research on transfer learning with autoencoders | 33% with Akram, Darie |
| 4 | Helped with implementing first successful transfer learning model | 50% with Akram |
| 5 | Implemented checkpointing and trained the final model |  |
| 6 | Implemented image normalization functions for the dataset |  |

**Brandon Wu’s Incomplete Tasks:**

| **Task #** | **Description** | **Note(s):** |
| --- | --- | --- |
| 1 | Creating a baseline model |  |
| 2 | Implementing image augmentation functions (Darie ended up doing it) |  |

### 

### 9.2 Final Work Distribution (relative to highest):

| **Akram** | **Brandon** | **Darie** | **Nada** |
| --- | --- | --- | --- |
| 95 | 95 | 100 | 96 |