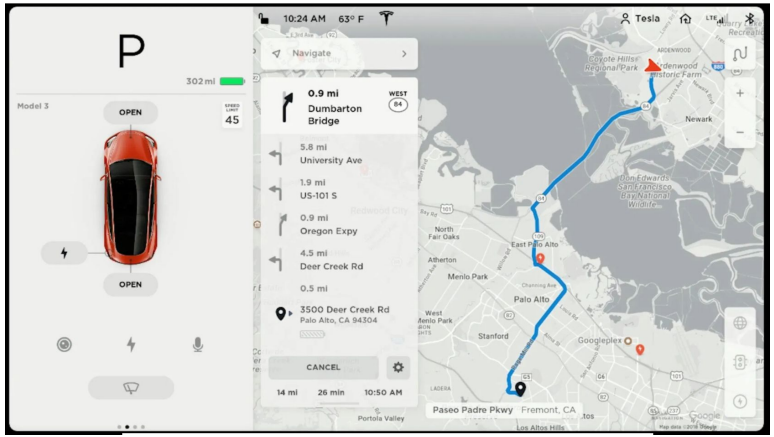
Depth Estimation of Indoor Scenes via Convolutional Neural Network

Jumi Kim
Raj Kalpesh Sanghavi
Danyang Wei

OBJECT

- This project objects to estimate the depth of images using convolutional neural networks.
- Depth estimation: a computer vision task designed to estimate depth from a 2D image.



- Why depth estimation is important?
  - It is a fundamental task in many applications including scene understanding and reconstruction
- Example - Application of depth estimation:
  - Self-driving cars
  - Grasping in robotics
  - robot-assisted surgery
  - Automatic 2D to 3D conversion in film

DATA

The project obtained **NYU-Depth V2** dataset, comprised of indoor scenes as recorded by both the RGB and Depth cameras in high resolution.

The dataset contains:

- Each object is labeled with a class and an instance number e.g., bathroom_0001, bathroom_0002
- Resolution: 640*480
- Pairs of Image (RGB) and depth images
    - E.g., living_room_0029_out 9.jpg | and .png
- Multiview of objects

OUR LIMITATIONS & MODELS

Hardware Limitation of this project

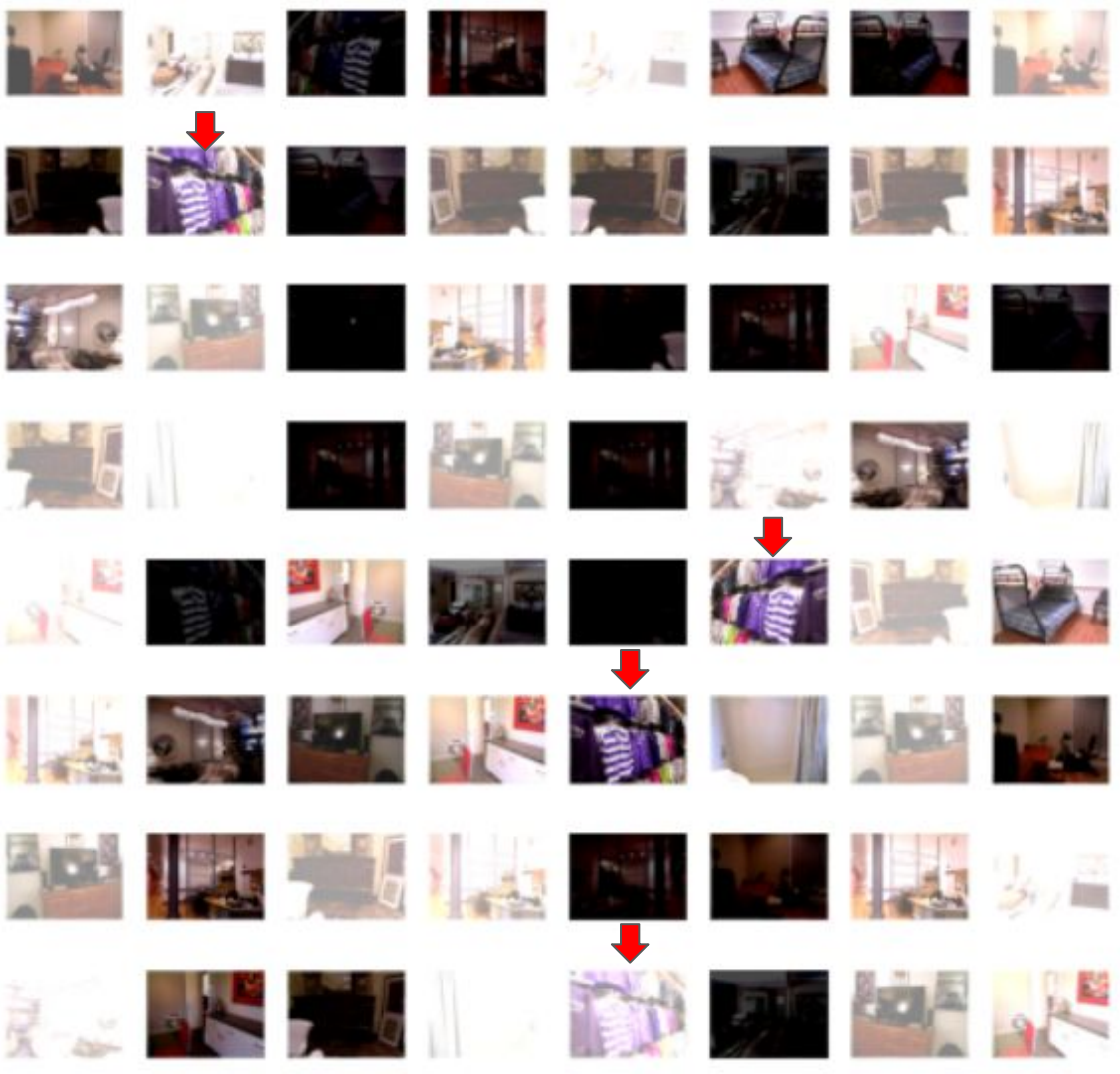
- We implemented our models using Tensorflow and trained on one 4GB GPU
- This project uses the subset of the dataset due to the hardware limitations.
- We reduced the resolution of depth images (320*240)
- It approximately took 1-2 days to train one model.
- Despite this difficulty, we trained the different models multiple times with different parameters and with/without augmentation.

Our Models:

- Hyperparameters
  - Batch size:8 | Learning rate = 0.0001 | epoch = 5
- Two models
  - CNN network
  - Transfer learning-based network (DenseNet169)

AUGMENTATION

- **Augmentation?**
  - A technique to increase the **diversity** of your **training set** by applying random but realistic transformations
- Our model also implemented augmentation to avoid over-fitting problems, which increase generalization of the model
- Our model uses:
  - Horizontal flip
  - color channel shift with range of 0.75



CNN MODEL ARCHITECTURE

Table 1.1 shows a series of convolutional and max-pooling layers followed by a series of upscaling and deconvolutional layers allow the network to extract image disparity features at the smaller scale (object edges), and generate a smooth estimate of the depth map at the larger scale (full object).

Layer	Function
INPUT	
CONV1	Convolution 7x7
POOL1	MaxPool 2x2
CONV2-A	Convolution 2x2
POOL2	MaxPool 2x2
CONV2-B	Convolution 2x2
POOL2	MaxPool 2x2
CONV3	Convolution 2x2
UP1	Upsample 2x2
CONV4	Convolution 2x2
UP2	Upsample 2x2
CONV5	Convolution 2x2
CONV6	Convolution 2x2

Table 1.1

PRETRAIN MODEL ARCHITECTURE

Fig 1.1 shows the structure of our encoder and decoder with skip connection network.

- Encoder - DenseNet-169 Layers
- Decoder - Starts with convolution layers with same output channels as that of encoder. We Then successively add upsampling layer followed by convolution layers.where the first convolutional layer of the two is applied on the concatenation of the output of the previous layer and the pooling layer from the encoder having the same spatial dimension..Each upsampling layers , except last one is followed by a Leaky Relu activation function.

Layer	Function
INPUT	
CONV1	DenseNet CONV1
POOL1	DenseNet POOL1
POOL2	DenseNet POOL2
POOL3	DenseNet POOL3
….	….
CONV2	Convoultion 1x1 of DenseNet BLOCK4
UP1	Upsample 2x2
CONCAT1	Concatenation POOL3
UP1 -CONVA	Convolution 3x3
UP1-CONVB	Convolution 3x3
UP2	Upsample 2x2
CONCAT2	Concatenation POOL2
UP1 -CONVA	Convolution 3x3
UP1-CONVB	Convolution 3x3
UP3	Upsample 2x2
CONCAT3	Concatenation POOL1
UP1 -CONVA	Convolution 3x3
UP1-CONVB	Convolution 3x3
UP3	Upsample 2x2
CONCAT3	Concatenation CONV1
UP2-CONVA	Convolution 3x3
UP2-CONVB	Convolution 3x3
CONV3	Convolution 3x3

Table 1.2

RESULTS

Pretrained Model -Input (RGB Image) / Output Results (Depth Image)

CNN Model - Input (RGB Image) / Output Results (Depth Image)

LOSS FUNCTION

- Objective: measure difference between the ground truth depth map y and the depth map $\hat{y}$ from network
- Three components (y_p is a pixel in depth image y)
  - Point-wise L1 loss: $L_{depth}(y, \hat{y}) = \frac{1}{n}\sum_n^p |y_p - \hat{y}_p|$
  - L1 loss regarding the image gradient g of the depth image: $L_{grad}(y, \hat{y}) = \frac{1}{n}\sum_n^p |g_x(y_p - \hat{y}_p)| + |g_y(y_p - \hat{y}_p)|$
  - Structural Similarity (SSIM): $L_{SSIM}(y, \hat{y}) = \frac{1-SSIM(y,\hat{y})}{2}$
- Loss Function: $L(y, \hat{y}) = \lambda L_{depth}(y, \hat{y}) + L_{grad}(y, \hat{y}) + L_{SSIM}(y, \hat{y})$

EVALUATION

- Six evaluation metrics
  - Average relative error (rel): $\frac{1}{n}\sum_p^n \frac{|y_p - \hat{y}_p|}{y}$
  - Root mean squared error (rms): $\sqrt{\frac{1}{n}\sum_p^n (y_p - \hat{y}_p)^2}$
  - Average (log_{10}) error: $\frac{1}{n}\sum_p^n |\log_{10}(y_p) - \log_{10}(\hat{y}_p)|$
  - Threshold accuracy (δ_i): $\%\ of y_p\ s.t. \max\left(\frac{y_p}{\hat{y}_p}, \frac{\hat{y}_p}{y_p}\right) = \delta < thr\ for\ thr = 1.25, 1.25^2, 1.25^3)$

COMPARISON CNN VS PRE-TRAINED MODEL

Model	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	$rel \downarrow$	$rms \downarrow$	$log_{10} \downarrow$
Simple CNN	0.242	0.463	0.756	0.513	3.563	1.023
Pre-trained	0.543	0.788	0.894	0.305	1.312	0.132

THANK YOU

HAVE A GREAT SUMMER!