1. Definition

1.1. Project Overview

There are several annotated and researched image datasets such as ImageNet, CIFAR-10, and CIFAR100. These datasets have contributed significantly to the field of computer vision (CV). CV is and can be used in several areas, one of which is helping individuals with vision impairment. In this area specifically, identifying cats, dog breeds, and airplanes is useful, however, more focus should be placed on safely navigating indoors, especially in unfamiliar environments. This is one of the encouragements for the launch of MCIndoor20000 dataset, a dataset which consists of indoor objects images. For this project, MCIndoor20000 will be used to develop an algorithm that inputs image path of indoor objects (door, sign, and stairs) and returns the label.

The MCIndoor20000 dataset contains a collection of 2D-RGB images grouped based on three classifications: door, sign, and stairs. Also, MCIndoor20000 contains five additional datasets that were generated via augmentation of the original images using the following techniques:

- 250 window sized Gaussian noise with standard deviation of 10
- 250 window sized Gaussian noise with standard deviation of 20
- Poisson noise
- Salt-pepper noise with density of 0.015
- Rotations

For this project, the original dataset along with Poisson and Gaussian (10) were used. The dataset is publicly available at https://github.com/bircatmcri/MCIndoor20000

1.2. Problem Statement

The dataset intra-class variation is major; however, all the pictures were taken from a single facility, which may lead to generalization challenges. The objective is to develop a convolutional neural network that is able to classify input images into one of three categories: door, sign, and stairs. The model needs to be generalized and able to handle wide spectrum of intra-class variations including images outside of the dataset.

The approach involves setting up the programming environment. Then data Ingestion, exploration, and preparation. Following that, model development and evaluation. Finally, wrapping up the model in a friendlier algorithm.

1.3. Metrics

Data will be split into training, validation, and testing. Training and validation data will be used for training the model, meanwhile, testing will be used to evaluate its performance by calculating the categorical accuracy. A qualitative measure will also be used to assess the model, by testing it on external images with that vary significantly from MCIndoor20000 data.

Given that the problem at hand is prediction of multiple categories, and the categories are relatively balanced and equally important, categorical accuracy will be used. Categorical accuracy is calculated by dividing the number of correct predictions over the total number of predictions as follows:

$$Acc = \frac{\sum_i^n TP_i}{\sum_i^n P_i + N_i}$$

$\boldsymbol{n}: number\ of\ classes$

$\boldsymbol{TP_i}: True\ Positives\ for\ class\ i$

$\boldsymbol{P_i}: Positives\ for\ class\ i$

$\boldsymbol{N_i}: Negatives\ for\ class\ i$

2.Analysis

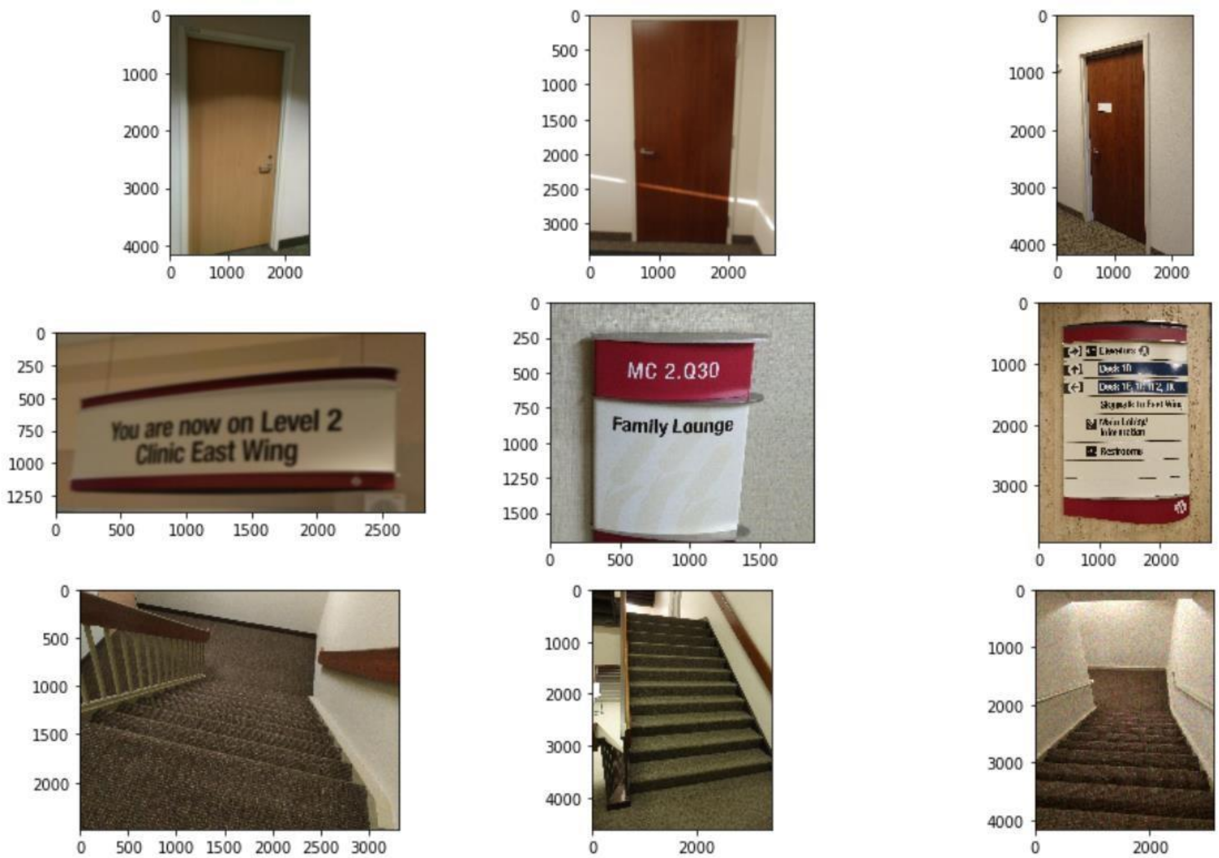
2.1. Data Exploration and Visualization

Please refer to section 1.1 of this report for information on the input data. The ingested data consists of 6166 images distributed as follows:

- Total number of Door images: 2262
- Total number of Sign images: 2106
- Total number of Stairs images: 1798
-

The images are not of constant size and shape. The sizes and shapes are mixed in the original dataset. However, they were all converted to a standard shape and size as discussed in the following sections. No abnormalities are observed in the dataset.

Sample from the dataset:



As shown in the sample, image sizes are large making it impractical to load the dataset into memory for training. Also, different images have different sizes. Another observation is the intraclass variations in terms of colors, angle, shape, and aspect ratio.

2.2. Algorithms and Techniques

From Scratch Model:

Convolutional neural network was used in this section. Three layers of conv/pooling were added for feature extraction via narrowing down width/height and increasing depth. Finally, for classification, a flatten and dense layer were included. This resulted in ~100k total number of parameters. Given the size of the dataset, overfitting is a concern.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_98 (Conv2D)           (None, 256, 256, 8)       104
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 128, 128, 8)       0
_________________________________________________________________
conv2d_99 (Conv2D)           (None, 128, 128, 16)      528
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 64, 64, 16)        0
_________________________________________________________________
conv2d_100 (Conv2D)          (None, 64, 64, 32)        2080
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 32, 32, 32)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 32768)             0
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 98307
=================================================================
Total params: 101,019
Trainable params: 101,019
Non-trainable params: 0
_________________________________________________________________
```

Transfer Learning:

Another method for model development is via transfer learning. This is achieved by utilizing the weights of a pre-trained model. For this part, InceptionV3 was used without the top layer and froze all the weights. Then a trainable global average pooling and a dense layer with softmax activation were added for classification.

2.3. Benchmark

The intended benchmark was the model from scratch. However, given that the dataset is small, the model achieved a very high categorical accuracy (99.3%). Overfitting is suspected and noticed when external images are used. However, it is difficult to benchmark on external images since a large number needs to be collected and labeled first.

3. Methodology

3.1. Data Preprocessing

Given the large size of images, they had to be reprocessed to a standard smaller size. The script for resizing them is titled `resize_images.py`. They were resized to (256,256, 3)/(299,299,3) sizes. The images were also split into training (81%), validation (9%), and testing (10%).

3.2. Implementation

- All images were resized to a standard (256,256,3) / (299,299,3) sizes
  - Used library *imageio.imread* for reading the images
  - *scipy.misc.imresize* to resize the images
  - *imageio.imsave* to save the images
  - *multiprocessing* for threading the script for speedup purposes

- Data was split into training (81%), validation (9%), and testing (10%)
  - Used *sklearn.model_selection.train_test_split* twice to get the three splits

- Target values (class) were converted to dummy variables
  - Used *keras.utils.np_utils* convert flat classification array to dummy categorical array.

- (nb, 256, 256, 3) tensors were created where nb is the number of samples
  - Read images from paths using *imageio.imread*
  - Used *numpy.expand_dims* to convert (256, 256, 3) to (1, 256, 256, 3) shape
  - Used *numpy.vstack* to stack images to form the shape (nb, 256, 256, 3)

- Tensors normalized by dividing over 256

The following steps utilized *keras* library

- Built CNN architecture (refer to section 2.2.) and compiled
- Check pointer used to save best model weights
- Developed model utilizing training and validation data
- Evaluated on test data
- Evaluated on small sample size of external images

3.3. Refinement

The initial solution was suspected to overfit the data. Qualitative assessment supported this concern. Given the relatively small size of the dataset, this was circumvented by utilizing transfer learning (refer to section 2.2)

4. Results

4.1. Model Evaluation and Validation

The final model was test on small sample size of external and correctly classified every image. As for categorical accuracy it achieved 98.3% accuracy on MCIndoor20000 testing test, and 100% on the external small testing set.

4.2. Justification

Qualitative assessment of the final model reveals it generalized better than the "from scratch" model. High accuracy is achieved (98.3%), with good generalization. With some improvements (see improvements section) the model should be ready for beta testing.

5. Conclusion

5.1. Free Form Visualization

Testing the model on external dataset

```
Detecting ...
Door detected ahead of you!
```


```
Detecting ...
Sign detected ahead of you!
```


```
Detecting ...
Sign detected ahead of you!
```


```
Detecting ...
Stairs detected ahead of you!
```

```
Detecting ...
Sign detected ahead of you!
```


```
Detecting ...
Sign detected ahead of you!
```


```
Detecting ...
Door detected ahead of you!
```

5.2. Reflection

Going through this project has helped reinforce the concepts of implementing convolutional neural networks. Also, the more challenging aspect was handling images as input into the python environment. It motivated me to learn more about image representation.

5.3. Improvements

There are few limitations and some improvements ideas that can be addressed in future iterations

- A classification for non-stair/door/sign images (perhaps enforce a softmax threshold?)
- Testing the model on larger labeled external dataset to ensure generalizability
  Developing a web interface (via flask) for accessing the model.