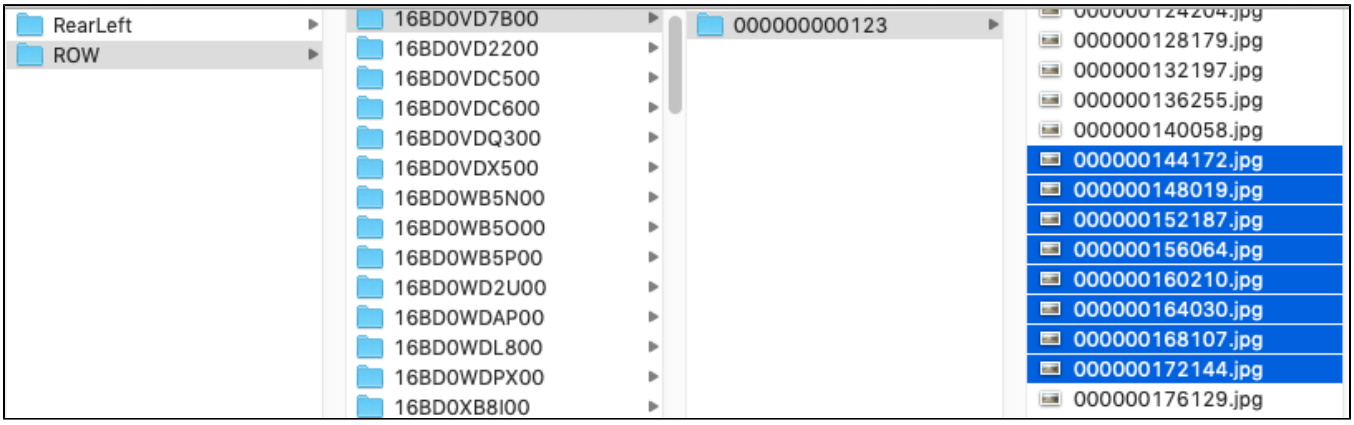
# Run a simple image recognition classifier

## 1. Dataset

- Spreadsheet: sidewalks.csv

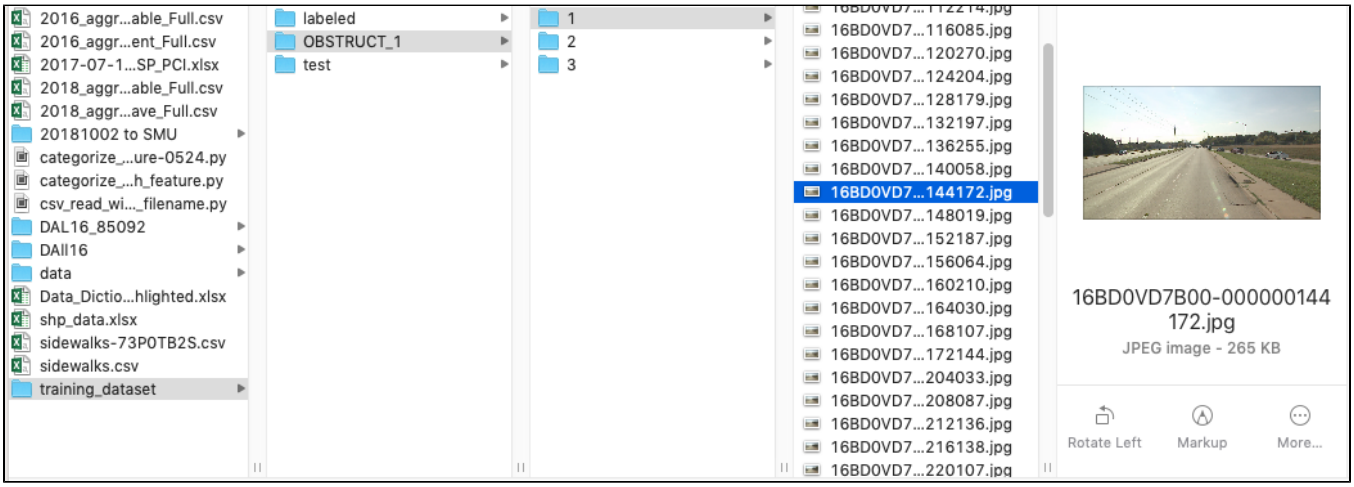
| OBJECTID | ID_SIDEWAL | STREETNAME | ... | OBSTRUCTIO | OBSTRUCT_1 | DEFICIENCY | DEFICIEN_1 | ... | BEGIN_IMAG | END_IMAGE |
|---|---|---|---|---|---|---|---|---|---|---|
| 4542 | 2836-2 | N HAMPTON RD_R | ... | None | 1 | None | 4 | ... | https://ivision.fugro.com/CityofDallas_Video/ROW/16BD0VD7B00/000000000123/000000144172.jpg | https://ivision.fugro.com/CityofDallas_Video/ROW/16BD0VD7B00/000000000123/000000172144.jpg |

- Images dir: DAll16/DAL16_85092/v12




## 2. Convert original dataset to training dataset

- Run
  - $ python categorize_with_feature.py
- Results




## 3. Create a basic CNN model with Keras

**cnn.py**

```
# Part 1 - Building the CNN

# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense

# Initialising the CNN
classifier = Sequential()

# Step 1 - Convolution
classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))

# Step 2 - Pooling
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Adding a second convolutional layer
classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Step 3 - Flattening
classifier.add(Flatten())

# Step 4 - Full connection
classifier.add(Dense(output_dim = 128, activation = 'relu'))
classifier.add(Dense(output_dim = 1, activation = 'sigmoid'))

# Compiling the CNN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])


# Part 2 - Fitting the CNN to the images

from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory('train',
                                                 target_size = (64, 64),
                                                 batch_size = 10,
                                                 class_mode = 'binary')

test_set = test_datagen.flow_from_directory('test',
                                            target_size = (64, 64),
                                            batch_size = 10,
                                            class_mode = 'binary')

classifier.fit_generator(training_set,
                         samples_per_epoch = 1500,
                         nb_epoch = 250,
                         validation_data = test_set,
                         nb_val_samples = 300)
```

## 4. Training in HAL cluster

- If your job may run more than 4 hours, you should change next two things.

- --partition=debug  --partition=solo
- --time=04:00:00  --time=72:00:00

**cnn.sb**

```
#!/bin/bash

#SBATCH --job-name="cnn-train-debug"
#SBATCH --output="cnn-train-debug%j.%N.out"
#SBATCH --error="cnn-train-debug%j.%N.err"
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --mem-per-cpu=1500
#SBATCH --ntasks-per-node=12
#SBATCH --cores-per-socket=12
#SBATCH --gres=gpu:v100:1
#SBATCH --export=ALL
#SBATCH --time=04:00:00

module load ibm/powerai
module list

python cnn.py
```

## 5. Results

- Training 3,000 images
- Test 600 images

**cnn-train-debug6075.hal02.out**

```
Found 3000 images belonging to 3 classes.
Found 600 images belonging to 3 classes.
Epoch 1/250

1/150 [..............................] - ETA: 8:45 - loss: 0.7129 - acc: 0.3000
10/150 [=>............................] - ETA: 50s - loss: 0.3547 - acc: 0.2600
11/150 [=>............................] - ETA: 47s - loss: 0.3451 - acc: 0.2636
12/150 [=>............................] - ETA: 47s - loss: 0.2712 - acc: 0.2667
13/150 [=>............................] - ETA: 48s - loss: 0.1345 - acc: 0.2538
14/150 [=>............................] - ETA: 48s - loss: 0.0803 - acc: 0.2714
15/150 [==>...........................] - ETA: 49s - loss: 0.1817 - acc: 0.2667
16/150 [==>...........................] - ETA: 48s - loss: -0.0491 - acc: 0.2625
17/150 [==>...........................] - ETA: 50s - loss: -0.1718 - acc: 0.2824
18/150 [==>...........................] - ETA: 50s - loss: -0.5505 - acc: 0.2833
19/150 [==>...........................] - ETA: 50s - loss: -0.5997 - acc: 0.2737
20/150 [===>..........................] - ETA: 50s - loss: -0.6494 - acc: 0.2750
21/150 [===>..........................] - ETA: 49s - loss: -0.5426 - acc: 0.2762
22/150 [===>..........................] - ETA: 50s - loss: -0.5179 - acc: 0.2818
23/150 [===>..........................] - ETA: 49s - loss: -0.3568 - acc: 0.2957
...
142/150 [===========================>..] - ETA: 3s - loss: 0.3929 - acc: 0.3289
143/150 [===========================>..] - ETA: 2s - loss: 0.3902 - acc: 0.3308
144/150 [===========================>..] - ETA: 2s - loss: 0.3875 - acc: 0.3313
145/150 [===========================>.] - ETA: 2s - loss: 0.3628 - acc: 0.3303
146/150 [===========================>.] - ETA: 1s - loss: 0.3385 - acc: 0.3308
147/150 [===========================>.] - ETA: 1s - loss: 0.3579 - acc: 0.3299
148/150 [===========================>.] - ETA: 0s - loss: 0.3770 - acc: 0.3291
149/150 [===========================>.] - ETA: 0s - loss: 0.3745 - acc: 0.3309
150/150 [==============================] - 151s 1s/step - loss: 0.3188 - acc: 0.3307 - val_loss: -6.6757e-09 -
val_acc: 0.3333

...
```

Reference

- https://medium.com/nybles/create-your-first-image-recognition-classifier-using-cnn-keras-and-tensorflow-backend-6eaab98d14dd