Title: Recognized the digit and the confidence level associated with each recognition

Tools & Technology:

- Python 2.7.10
- Pycharm Community Edition 2016.3.2

Deep Learning Library:

- Keras configured with Theano and Tenserflow as backend

Python Library:

- OpenCV 3.0
- CV2
- Sklearn
- Pandas
- Numpy
- Scipy
- Imutils
- Sklearn matrics

Algorithm:

1. Image Preprocessing
2. Feature extraction using image processing technique
3. Image Classification and Recognition
4. Accuracy measurement

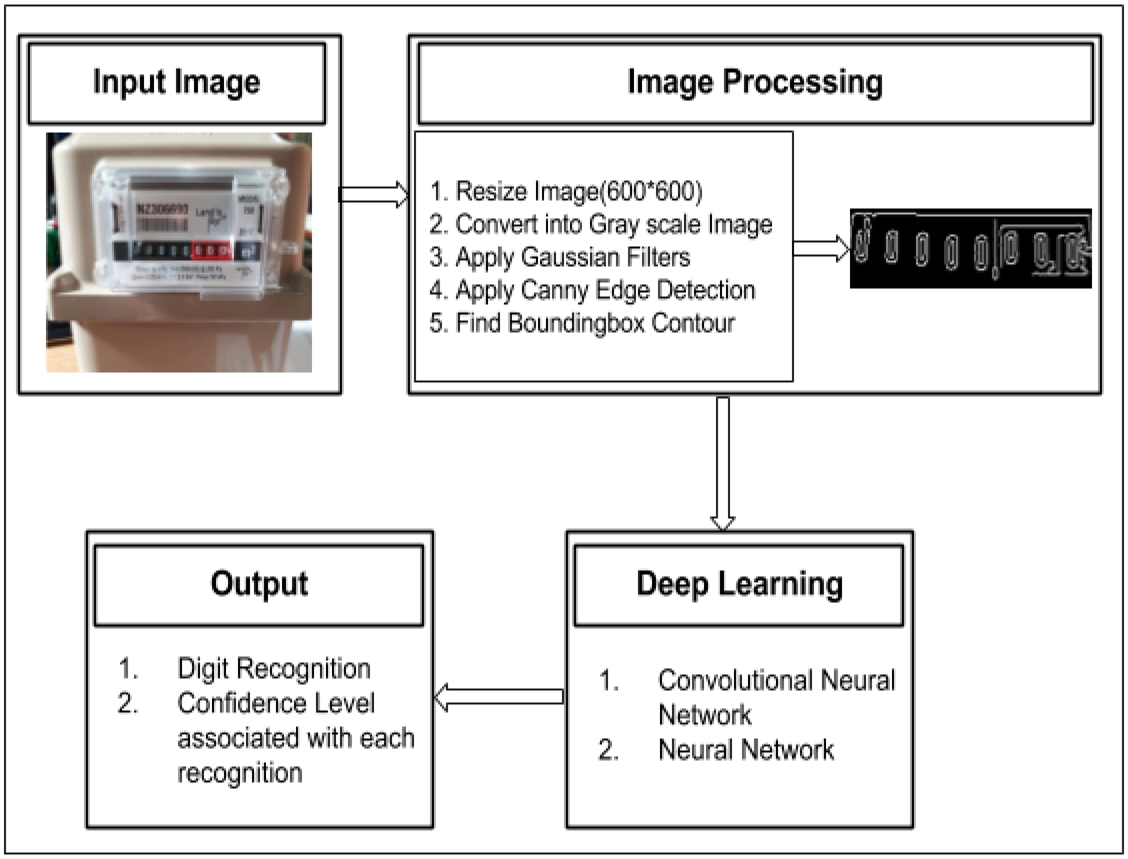
1. **Image Preprocessing**
    a. Load Image from the Directory
    b. Apply Down sampling over original image and reduce it’s size.
    c. Convert image in 600*600 scale with best quality interpolation of openCV cv2.CV_INTER_CUBIC
    d. Store image into array.

2. **Feature extraction using image processing technique:**
   It is necessary to identify required feature for each recognition from image.

Algorithm for Image processing:

a. Converting the image into grayscale
b. Applying Gaussian blurring with a *5×5* kernel to reduce high-frequency noise.
c. Computing the edge map via the Canny edge detector algorithm.
d. Find all contour and filter it out.
e. For each contour, we iterate over the boundingRect box contour, ensure the width and height are of an acceptable size to capture digit region and mark it as ROI.
f. Extract the ROI of an image and reshape it as 64 by 32 image.
g. Get 10 different ROI from one image (Trial and Error decided).
h. Find the ROI of all the images and store it into array.

Basic Architecture :


Input Image
Image Processing
1. Resize Image(600*600)
2. Convert into Gray scale Image
3. Apply Gaussian Filters
4. Apply Canny Edge Detection
5. Find Boundingbox Contour
Output
1. Digit Recognition
2. Confidence Level associated with each recognition
Deep Learning
1. Convolutional Neural Network
2. Neural Network

3. Image Classification and Recognition:

Algorithm: I developed two algorithm to solve this problem.

1) CNN with Theano (CPU only)
2) Deep Neural Network with Theano (CPU only)

Data set : Divided data set into two part

1. **Train Set and Validation Set :** Spilt size **[80,20]** from train set vs validation set
   No of sample images for train: 62
   (Each image has 10 ROI so total input 620 sample)
2. **Test Data**
   No of sample images: 10
   (Each image has 10 ROI so total input 100 sample)

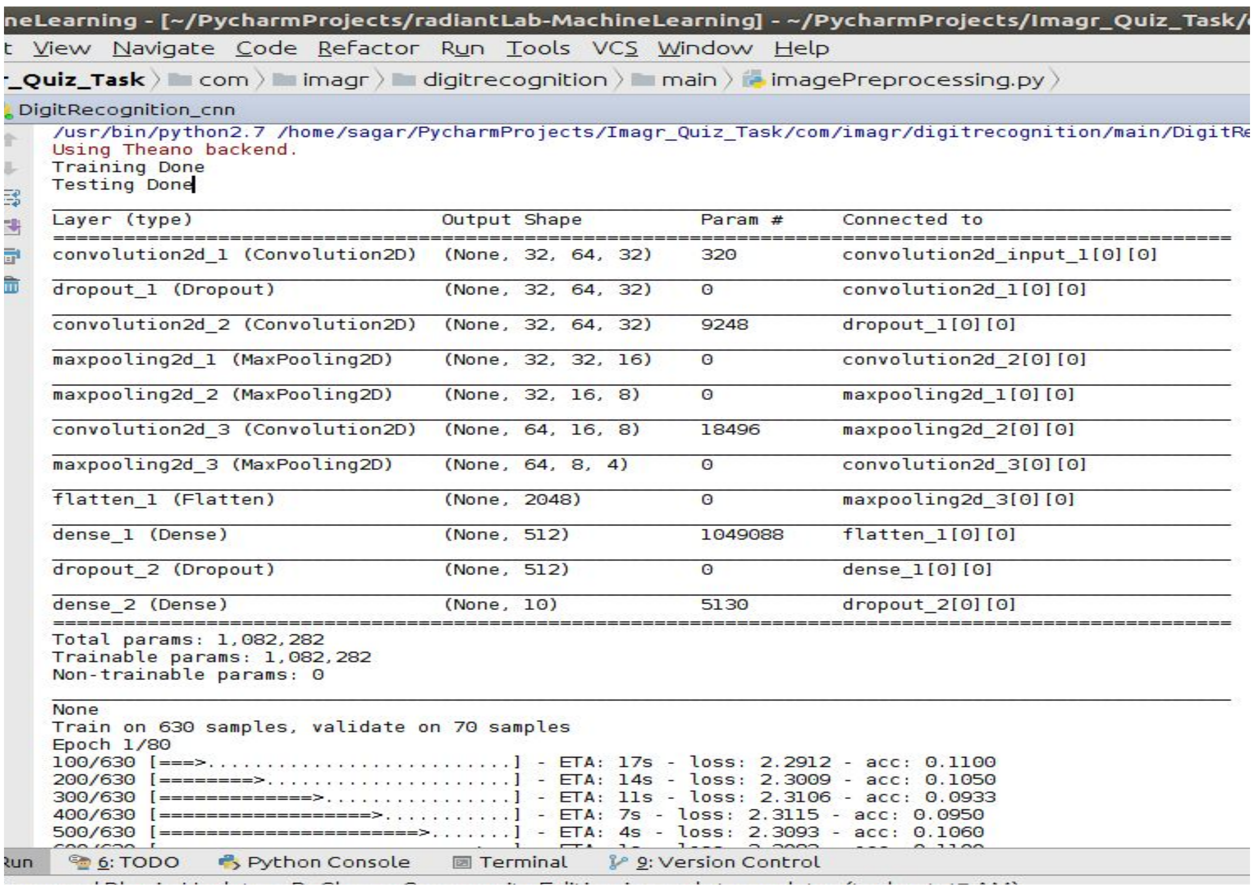
CNN Parameters:

- Input Image to CNN **:** 64X32 with shape of (1,64,32).
  (1 stands for grayscale image with total 2048 pixels)
- Learning Rate: 0.01
- Momentum:0.9
- No of batch size :500
- No of epoch : 300
- No of class Label :10

CNN layer Architecture:

- Convolutional input layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
- Dropout layer at 20%.
- Convolutional layer, 32 feature maps with a size of 3×3 and a rectifier activation function.
- Max Pool layer with size 2×2.
- Max Pool layer with size 2×2.
- Convolutional layer, 64 feature maps with a size of 3×3 and a rectifier activation function.
- Dropout layer at 20%.
- Flatten Layer.
- Fully connected layer with 512 units and a rectifier activation function.
- Dropout layer at 20%.
- Fully connected output layer with 10 units and a softmax activation function.

CNN Parameter Architecture:


neLearning - [~/PycharmProjects/radiantLab-MachineLearning] - ~/PycharmProjects/Imagr_Quiz_Task/
t View Navigate Code Refactor Run Tools VCS Window Help
_Quiz_Task 〉 com 〉 imagr 〉 digitrecognition 〉 main 〉 imagePreprocessing.py 〉
DigitRecognition_cnn
Run 6: TODO Python Console Terminal 9: Version Control

```
/usr/bin/python2.7 /home/sagar/PycharmProjects/Imagr_Quiz_Task/com/imagr/digitrecognition/main/DigitRe
Using Theano backend.
Training Done
Testing Done
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
convolution2d_1 (Convolution2D)  (None, 32, 64, 32)    320         convolution2d_input_1[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 32, 64, 32)    0           convolution2d_1[0][0]
____________________________________________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 32, 64, 32)    9248        dropout_1[0][0]
____________________________________________________________________________________________________
maxpooling2d_1 (MaxPooling2D)    (None, 32, 32, 16)    0           convolution2d_2[0][0]
____________________________________________________________________________________________________
maxpooling2d_2 (MaxPooling2D)    (None, 32, 16, 8)     0           maxpooling2d_1[0][0]
____________________________________________________________________________________________________
convolution2d_3 (Convolution2D)  (None, 64, 16, 8)     18496       maxpooling2d_2[0][0]
____________________________________________________________________________________________________
maxpooling2d_3 (MaxPooling2D)    (None, 64, 8, 4)      0           convolution2d_3[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 2048)          0           maxpooling2d_3[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 512)           1049088     flatten_1[0][0]
____________________________________________________________________________________________________
dropout_2 (Dropout)              (None, 512)           0           dense_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 10)            5130        dropout_2[0][0]
====================================================================================================
Total params: 1,082,282
Trainable params: 1,082,282
Non-trainable params: 0
____________________________________________________________________________________________________
None
Train on 630 samples, validate on 70 samples
Epoch 1/80
100/630 [===>..........................] - ETA: 17s - loss: 2.2912 - acc: 0.1100
200/630 [========>.....................] - ETA: 14s - loss: 2.3009 - acc: 0.1050
300/630 [=============>................] - ETA: 11s - loss: 2.3106 - acc: 0.0933
400/630 [==================>...........] - ETA: 7s - loss: 2.3115 - acc: 0.0950
500/630 [=======================>......] - ETA: 4s - loss: 2.3093 - acc: 0.1060
```

Accuracy:

Accuracy on **trained data :** 100 %

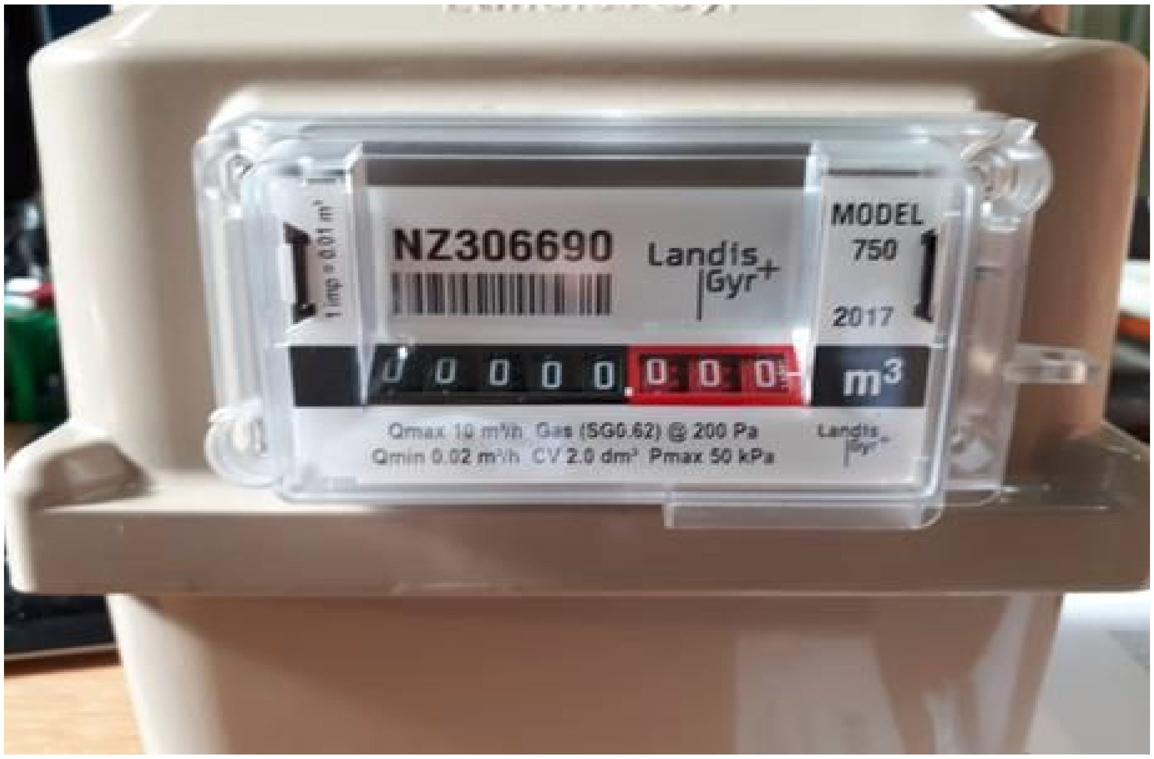
Accuracy on **test data :** 92.38 % (Currently working on improvements)

Accuracy Improvement Task:

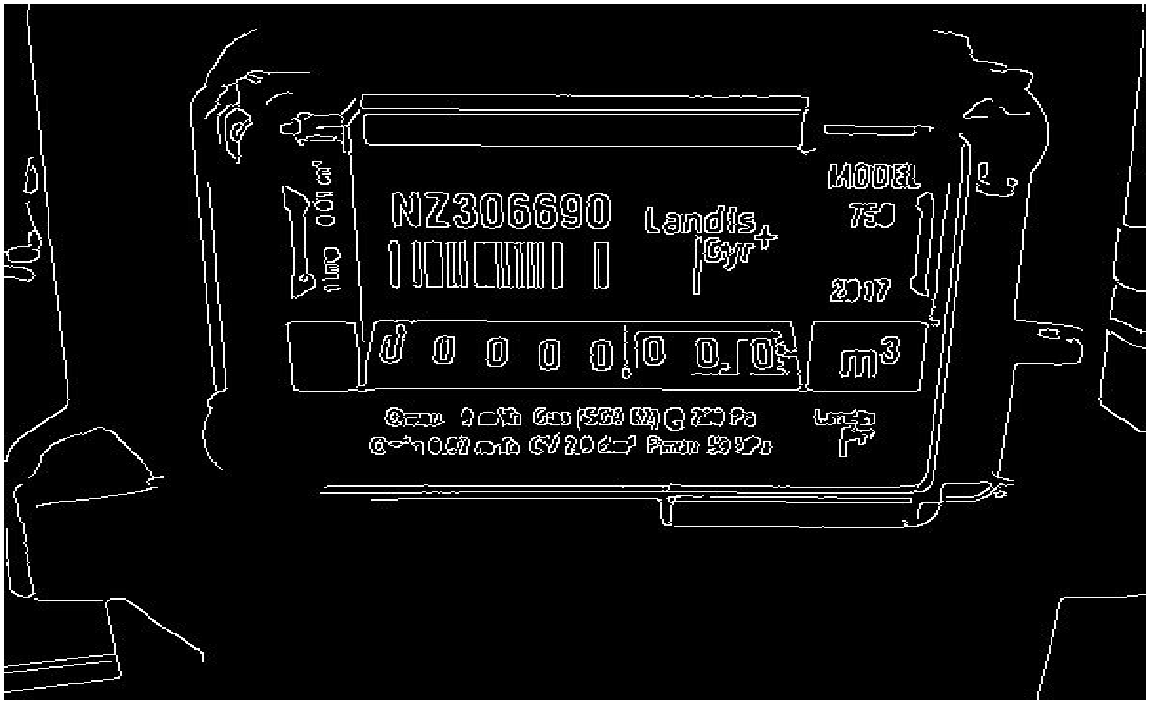
1. Use Data Augmentation to generate more training data and build model.
2. Identify more filter to find out exact ROI of digit region from image.
3. Change CNN layers Topology and its parameters like batch size , epoch,learning rate and etc.
4. Configured CNN in GPU so training will be faster to large CNN.
5. To evaluate model performance on different classifier: Developed deep neural network with theano as backend and measure its accuracy.

Screenshots:

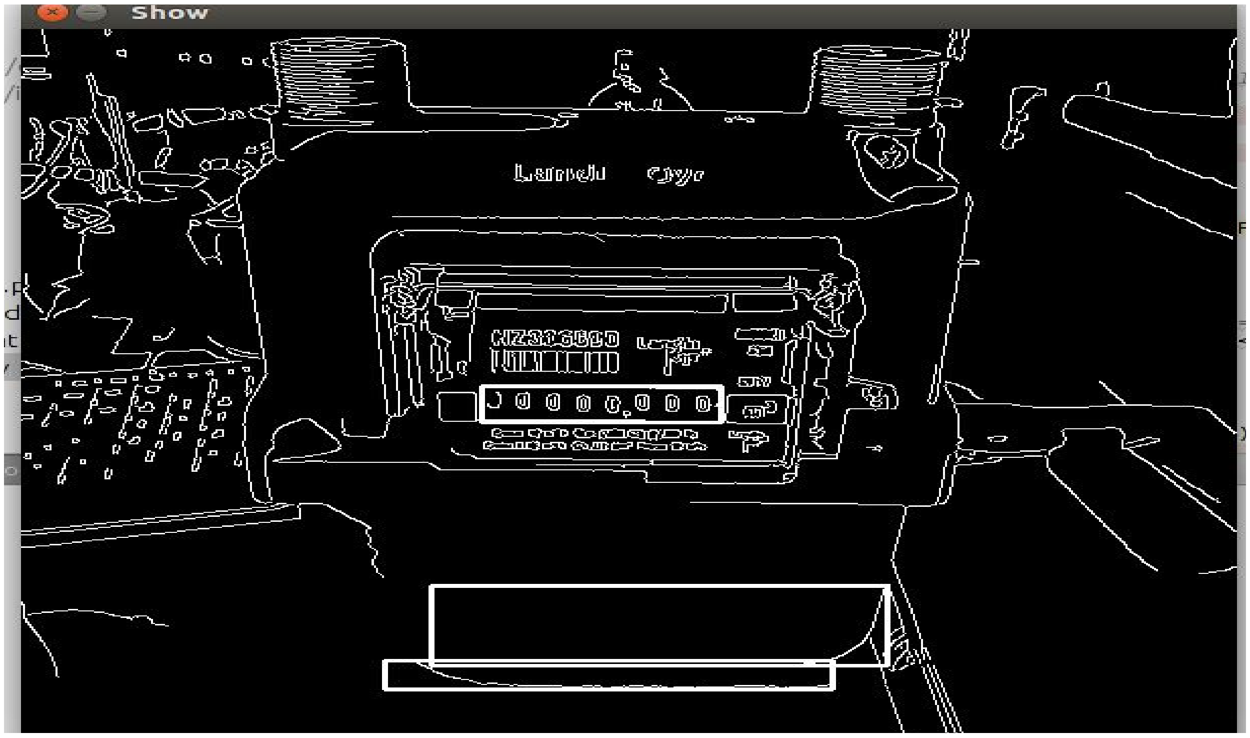
Original Image:



After Canny edge Detection Image:



Finding boundingRect Contour into an image



Find Bounding box contour Image and reshape it to (64X32) :