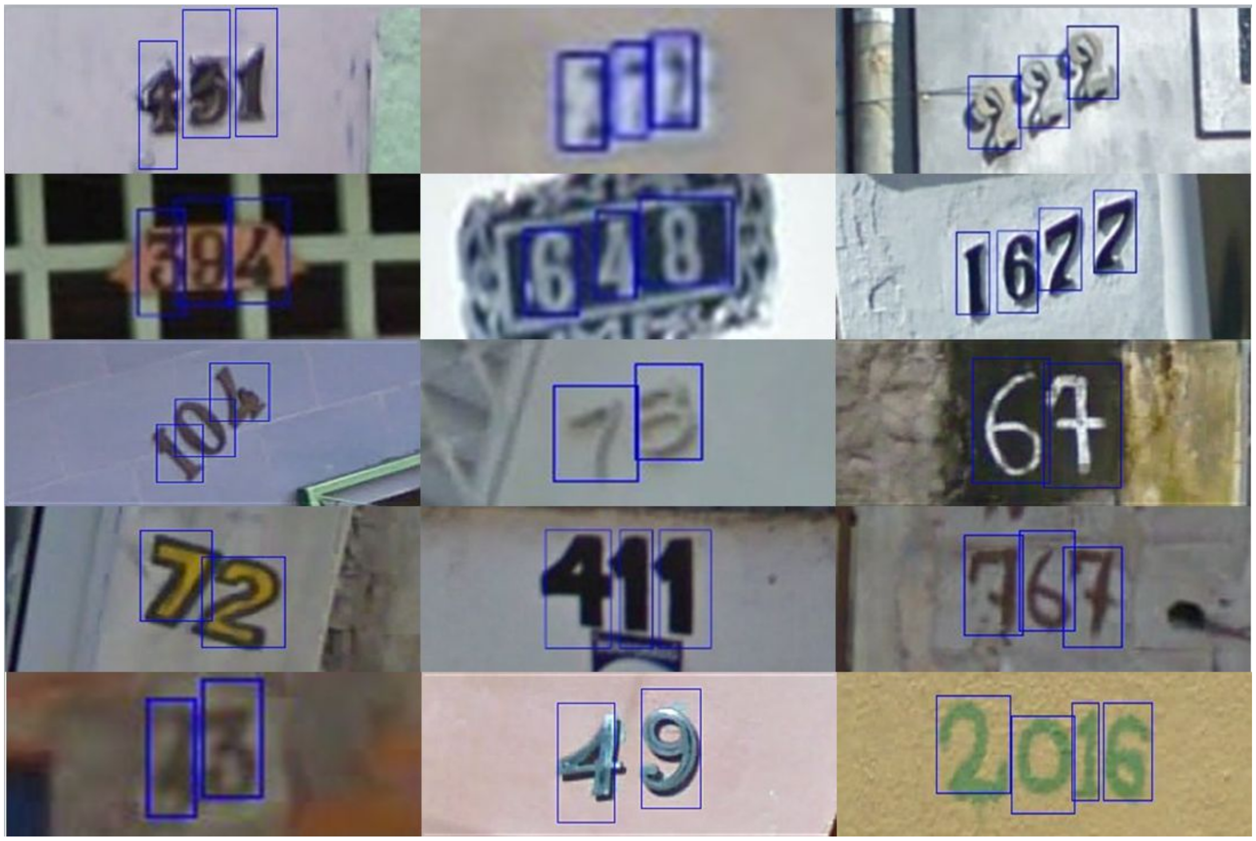
ADVANCED MACHINE LEARNING

SVHN RECOGNITION



TEAM NAME - MARAUDERS

MEMBERS

Jayanth Shankar - 140050039

Divakar Reddy - 140050044

Rishabh Chavhan - 140050061

Thanuj Raju - 140050077

Motivation

Recognizing digits in a real world can help improve navigation services and building accurate maps. It is central to a variety of emerging machine learning and computer vision applications.

A sample application of interest is the problem of identifying house numbers posted on the fronts of buildings. Recognizing characters is much more difficult than MNIST in natural scenes. We get image obtained from house numbers in Google Street View images and predict the written number.

Related Literature

1. Reading Digits in Natural Images with Unsupervised Feature Learning
2. Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks

Datasets

1. **SVHN:** 10 classes. 1 for each digit. Digit 1 has label '1', 9 has label '9' and 0 has label '10'. 73257 digits for training, 26032 digits for testing and each image is of size 32x32x3 centred around a single character.
2. **Chars74k:** 62 classes(0-9,a-z,A-Z). 7705 are natural images. 3410 are hand-drawn images. 62992 are computer generated. These are merged.
3. **CIFAR-10:** Images of real-world objects. 60000 images with 10 classes. Each label has 6000 images. 50000 training and 10000 test images. Size of each image is32x32x3

Approach

Step 1: Given an image, segment it into various objects using scipy library.

Step 2: For each of the images obtained, we need to decide whether there is a character in it.

Step 3: For each of the images containing a character, identify the digit in it.

For Step 2, we tried CIFAR-10 + 74k-char, with CIFAR-10 being label '0' and 74k-char being label '1' and we tried to train an SVM model and a CNN model and we got 100% train accuracy. We think this was due to the inherent difference between the datasets as it was too unpredictable on real-world images.

For Step 3, we used the SVHN dataset as the input to a CNN model and output labels 0-9.

Experiments

Architecture: We have 3 convolution layers of kernel dimensions of [5,5,3,10], [5,5,10,15], [5,5,15,80] with stride of [1,2,2,1]. And each of these layers is followed by a RelU Activation followed by an average pool.

CNN is trained for SVHN with minibatch stochastic gradient descent algorithm with the batch size of 128.

We ran it for 15000 steps and 10000 steps. It took 1 hour to train for 15000 steps and 40 minutes to train for 10000 steps.

Results:

For 15000, we got 85% train accuracy with 80% test accuracy.

For 10000, we got 82% train accuracy with 78% test accuracy.

We tried to make the number of steps as 25000, the train accuracy became 87% but without any significant increase in the test accuracy(81.3%).

State of the art accuracy is 98.31% using average-max pooling.

Platform:

We used a laptop with 4gb RAM, Tensorflow library in Python language. We started the code from scratch. For each model it took one hour to train. For the failed experiment where we tried to determine whether an image has a character or not, the training happened quickly.

The main model is coded in 120 lines while the data processing and image segmentation is the major part of the code volume.

Effort

Majority of time was spent on trying to find the proper algorithm for image segmentation and training the model with different parameters, while the data processing code did not take much time.

Github Link: https://github.com/divakarreddy/AML

Citations

1. http://www.cs.toronto.edu/~kriz/cifar.html
2. http://ufldl.stanford.edu/housenumbers/
3. http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/