Proposal

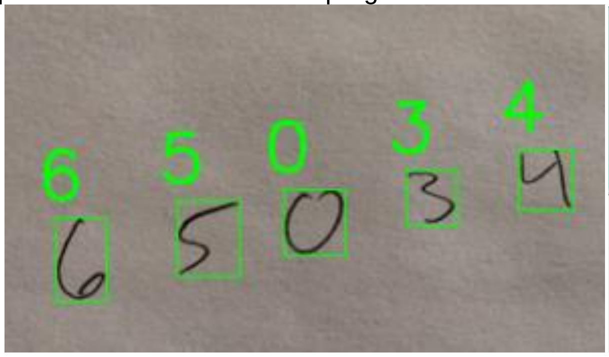
Machine Learning Nanodegree Capstone Project

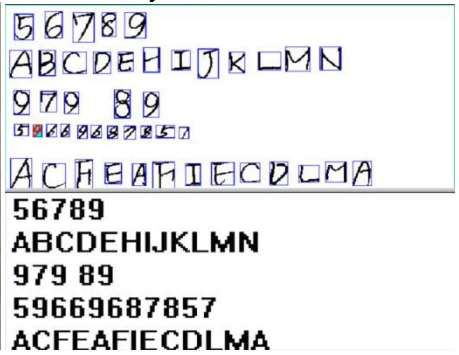
Dharmendra Kumar

March 2017

Domain Background

Over the last few years, object recognition on images has piqued interest of both the leading industry and academic professionals. While some problems like single character recognition are easy to solve, recognizing a sequence of characters still are challenging. While Machine learning has been a known science since 1980s, availability of a huge amount of open data along with the use of GPUs to solve these computationally intensive problems have accelerated progress in this domain in the last 5 years.



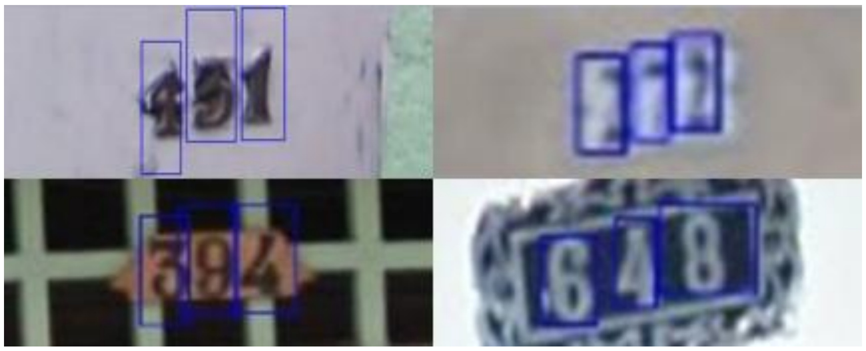


```
56789
ABCDEHIJKLMN
979 89
59669687857
ACFEAFIECDLMA
```

This project will be focused on recognizing a house number, which is basically a sequence of digits. In particular, The objective here would be to develop an application, which will recognize house number from images. The application will take as input a bunch of images and provide a house number presented on each image. One of the challenges here might be that some house numbers are displayed in some specific order. For example, some might be displayed vertically or diagonally or in some other orientation. However, the goal here would be to build an application which recognizes numbers from given images with high accuracy.

Problem Statement,

In this project, Street View House Numbers(SVHN)[1] dataset will be used to train the application model to recognize numbers on images. SVHN is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting. The dataset is obtained from house numbers in Google Street View images. Image below represents several samples from this dataset.



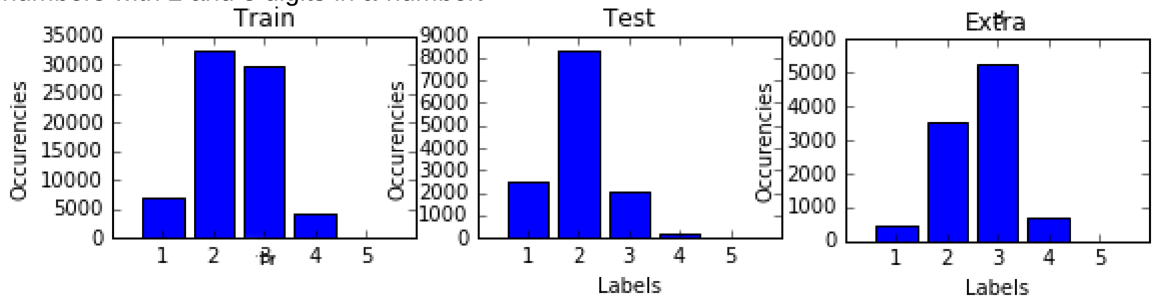
Some of the images as shown above have quite bad quality, and would be challenging for even a human to recognize them accurately. Therefore, some error can be expeced from our model as well. Here the accuracy of a house number (label) would be to identity all digits in the house number is the correct order, which is a challenge as missing even a single digit from the number or its order would result in a failure.

Recognizing numbers on images can be accomplished with human help if the amount of images to recognize is not huge. However, if the amount of images is extremely large using people to solve this problem is not efficient. And this is the place when Machine Learning comes into play!

Datasets and Inputs

SVHN dataset consists of 3 groups. Training, testing and extra data. Training set has 33402 samples, testing set has 13068 and extra set has about 230000 samples. Extra set will be used to extend our training set and perform validation during model training. Speaking about possible labels, basically there might be an infinite amount of labels, since there are no limitations about data presented. However, we are going to work with images which have up to five digits between 1 and 99.999 inclusive. Therefore, samples which have numbers with more than 5 digits will be removed from the dataset during preprocessing. Beside the images, dataset contains also a digitStruct structure which contains some meta information about each image. This structure has same length as amount of images in a dataset (one entry per image). Here each entry has two fields: name which is a string containing the filename of the corresponding image and bbox which is a struct array that contains the position, size and label of each digit bounding box in the image. For instance, digitStruct(300).bbox(2).height gives height of the 2nd digit bounding box in the 300th image. We shall use this bounding box information to build another model to guess a bounding box around the labels.

Figure below illustrates class lengths distribution across several datasets. As we can see all datasets have Gaussian distributions. The relation of 1, 4 and 5 digits classes to the whole dataset almost equal for training, test and validation datasets. About 90% of training data has 2-3 digit labels. About 80% of test data has 2 digit labels and about 10% of tests data are numbers with 1 or 3 digits each. The majority of extra data are numbers with 2 and 3 digits in a number.



Now let’s take a look on images sizes. First for, during preprocessing we are going to crop images just near the numbers to make our neural network easier to train. So original image size does not matter. Below are presented min, max and mean sizes of images height and width of train, validation and test datasets after they were cropped just near the numbers.

	Train	Test	Validation
Max Width	876	1083	668
Max Height	501	516	356
Min Width	25	31	22
Min Height	12	13	13
Mean Width	128.29	172.58	100.04
Mean Height	57.21	71.56	60.72

Since images sizes are different we are going to resize all of them to same size during preprocessing step which is described below.

Evaluation Metric:

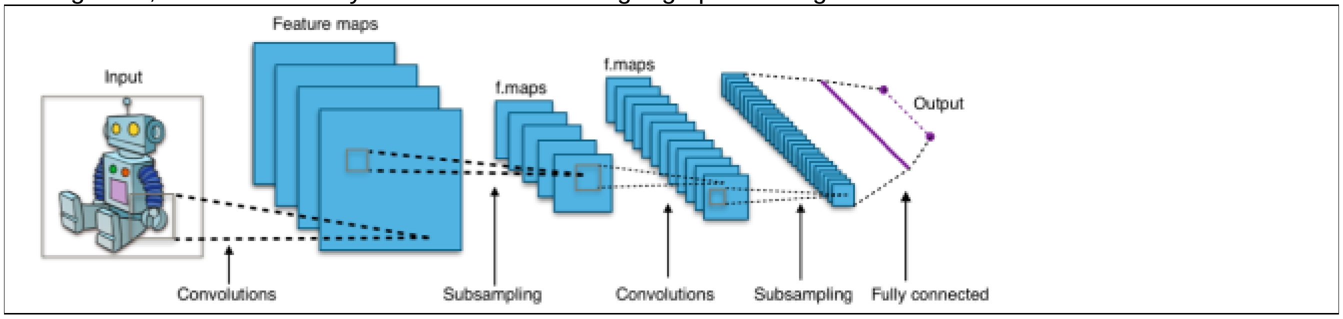
For this problem the accuracy will be calculated by counting how many numbers were correctly recognized. Hence, all digits in the house number must match in the same order for to count as a success. Then we sum up all the successfully identified house numbers and divide it by a size of a dataset. Here is the formula:

$$\Sigma(Correctly\ classified\ digits\#\ of\ digits\ in\ number)//\ \text{number of digits})/\text{number of images}$$

Solution Statement

To solve this problem, we are going to use modern deep learning techniques which are provided by the Google's TensorFlow library. In addition, a model using Keras library will be built and the best model will be further tuned to be finally used for our application. In the first part of a project we develop a multilayer Convolutional Neural Network which will be trained to predict just a single digit.

Convolutional Neural Network (CNN, or ConvNet) is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex, whose individual neurons are arranged in such a way that they respond to overlapping regions tiling the visual field. Convolutional networks were inspired by biological processes and are variations of multilayer perceptrons designed to use minimal amounts of preprocessing. They have wide applications in image and video recognition, recommender systems and natural language processing.



A ConvNet model represents a stack of various layers that transform an input to the output. The most common types of layers are:

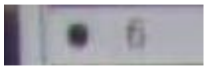
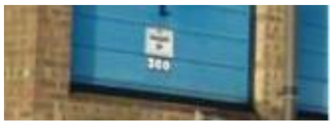
Convolutional Layer - The Conv layer is the core building block of a Convolutional Network that does most of the computational heavy lifting. The convolutional layer parameters consist of a set of learnable filters. The network will intuitively learn which filters to activate when they see some type of pattern. The input to the convolutional layer usually called input feature map and output is an output feature map.

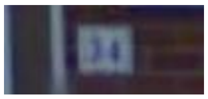
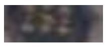
Pooling Layer - progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control over-fitting. The Pooling layer operates independently on every depth slice of the input and resizes it spatially, by using some function. In our application MAX operation will be used, as the most common one.

Fully Connected Layer - Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset. See the Neural Network section of the notes for more information.

Loss Layer - The loss layer specifies how the network training penalizes the deviation between the predicted and true labels and is normally the last layer in the network. Various loss functions appropriate for different tasks may be used there. We will use Softmax loss function for predicting a single class of K mutually exclusive classes.

After basic version will be designed, we will tune network's parameters trying to find configuration with lowest error rate. This network will be able to recognize numbers with up to five digits with low error rate. It will be able to take an image with house number presented and return the number which is presented on an image. We evaluate this approach on test dataset and achieve over 90% accuracy in recognizing complete street numbers as SVHN dataset contains many samples which is hard to recognize even for a human.

As you can see images above are very different. They have different size or quality or all together, they can contain some additional symbols nearby. Digits could be written diagonally. All such data is hard to recognize even for a human, so we expect that neural network may also face with some problems here. Moreover, due to a slow hardware, the neural network which we build is going to be quite simple and we don't use all available data, since it is hard to load all it to the RAM. Therefore, our solution would target an accuracy of 80% where the numbers are correctly identified from the images from the testing set.

Benchmark Model

There has been a few studies performed on this problem. In fact, there was a research paper written by Ian Goodfellow, Yaroslav Bulatov, Julian Ibarz,Sacha Arnoud, Vinay Shet on Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks. Their state-of-the-art model achieved 97.84% accuracy in recognizing complete street numbers, which is simply outstanding.

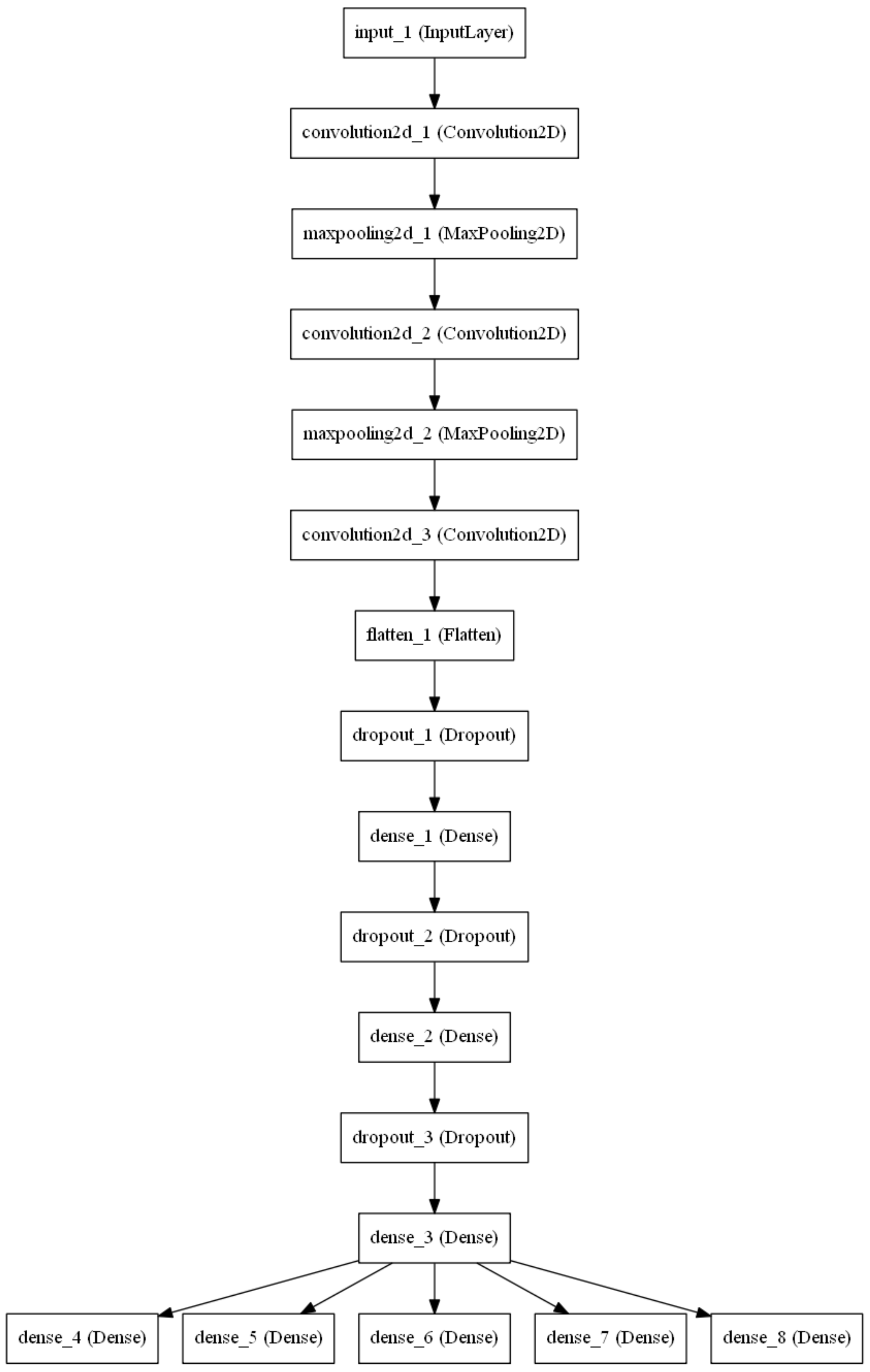
Project Design

Data Preprocessing

1. Crop each image just near the numbers to simplified effort required for model fitting.
2. Resize cropped images to 32x32 size since after cropping all images have different sizes.
3. Convert all input images into a grayscale. This is a very common technique in image preprocessing to help classifiers as digit's color has little or no significance in its identification. This is achieved by multiplying red channel by a factor of 0.2989, green by 0.5870 and a blue one by 0.1140.
4. Apply local contrast normalization on images to help our optimizer converge faster and accurately.
5. Replace '0' digit class from 10 to 0 to be consistent with other digits and classes. Instead, use class 10 to label not-existing digit resulting in a total of 11 classes for identified digit labels.
6. Split each sample class by digit. If number has size less than 5 digits, then add labels 10 at the end. It means that number 567 will be labeled as: ['5', '6', '7','10','10'].

Model:

The following diagram shows a model that will be used for this application. We will build a model using TensorFlow with independent layers for each digit, and another model using Keras with shared layers until the output layer and compare them for their performance and accuracy.


input_1 (InputLayer)
convolution2d_1 (Convolution2D)
maxpooling2d_1 (MaxPooling2D)
convolution2d_2 (Convolution2D)
maxpooling2d_2 (MaxPooling2D)
convolution2d_3 (Convolution2D)
flatten_1 (Flatten)
dropout_1 (Dropout)
dense_1 (Dense)
dropout_2 (Dropout)
dense_2 (Dense)
dropout_3 (Dropout)
dense_3 (Dense)
dense_4 (Dense)
dense_5 (Dense)
dense_6 (Dense)
dense_7 (Dense)
dense_8 (Dense)

The superior model will be tuned to improve its performance.

An independent python application will be developed leveraging the model developed with the aforementioned design and used to recognize digits along with bounding boxes around the number labels in any random image of any size and will use OpenCV for visualization. The sample usage of the application would be as follows:

Read_house_numbers image1 image2 …