MLND Capstone Project

Tahir Dar

May 19, 2017

1 Definition

1.1 Project Overview

In this work we try to recognize multi digit numbers from photographs. We work on two datasets to recognize digits. First is hand written dataset called as MNIST[1] and second is SVHN[2] dataset. Digit recognition from images is very important problem in the field of Image Processing as it is the sub problem of larger problem like word recognition, sentence recognition. Digit recognition is the first step towards applying Artificial Intelligence in image processing. This work uses Artificial Neural Networks specifically CNN technique to train the model and then we predict numbers from images.

Digit recognition has been studied earlier and the best model till date for Multi digit recognition has been given by Good Fellow at el(Ian J. Goodfellow, 2014-15).

In this work we use unsupervised feature extraction and selection using Convolutional Neural Networks(CNN).

1.2 Problem Statement

The problem statement is to correctly read a sequence of digits(Numbers) from images obtained from Street View House Numbers (SVHN) Dataset. The approach to solve such a problem is to work it out by using CNN and try to get better results using them. The challenge in this problem is to read each digit from a sequence to predict the actual sequence accurately.

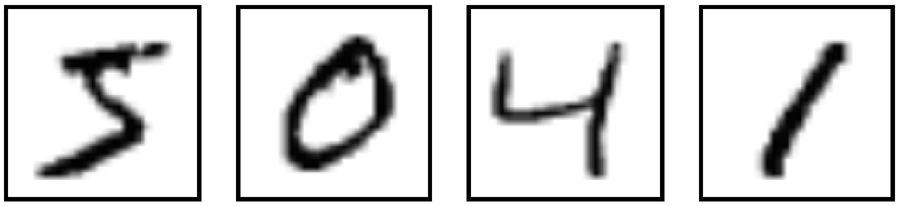
1.3 Metrics

The true positives and true negatives in case of Digit recognition both matter as we need to be as accurate while classifying as possible. We are trying to detect all the digits of the number, a single wrong number will result in wrong recognition. Accuracy metrics seems to be the prime candidate to be used as a metric in this scenario.

[1]`http://yann.lecun.com/exdb/mnist/`

[2]`http://ufldl.stanford.edu/housenumbers/`

Figure 1: Sample of MNIST dataset



Accuracy = true positives + true negatives /dataset size

2 Data Exploration

2.1 Overview of MNIST

The first dataset MNIST[3] database of handwritten digits, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

The original black and white (bilevel) images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. the images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.

The training set in MNIST contains 60000 images and testing set contains 10000 images.

2.2 Overview of SVHN

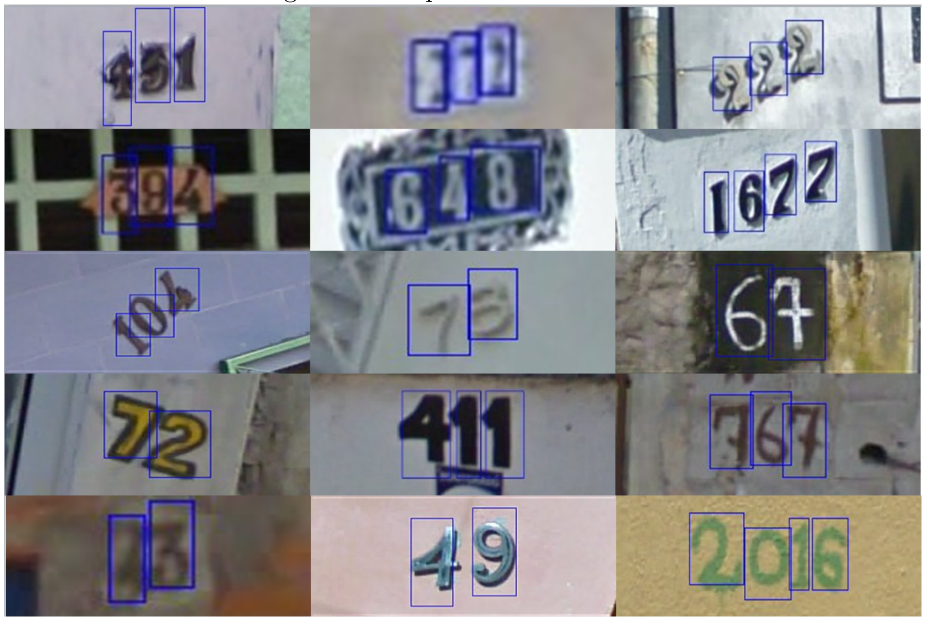
The other dataset SVHN[4] is a real-world digit dataset. SVHN can be obtained from house numbers in Google Street View images. It can be seen as similar in flavor to MNIST but incorporates an order of magnitude more labeled data (over 600,000 digit images) and comes from a significantly harder, unsolved, real world problem (recognizing digits and numbers in natural scene images). SVHN is obtained from house numbers in Google Street View images. The sample of MNIST is in figure 1.

It contains 10 classes, 1 for each digit. Digit '1' has label 1, '9' has label 9 and '0' has label 10.

It has 33402 images with multiple digits for training, 26032 digits for testing, and 531131 digits additional(extra), somewhat less difficult samples, to use as

[3]`http://yann.lecun.com/exdb/mnist/`
[4]`http://ufldl.stanford.edu/housenumbers/`

Figure 2: Sample of SVHN dataset



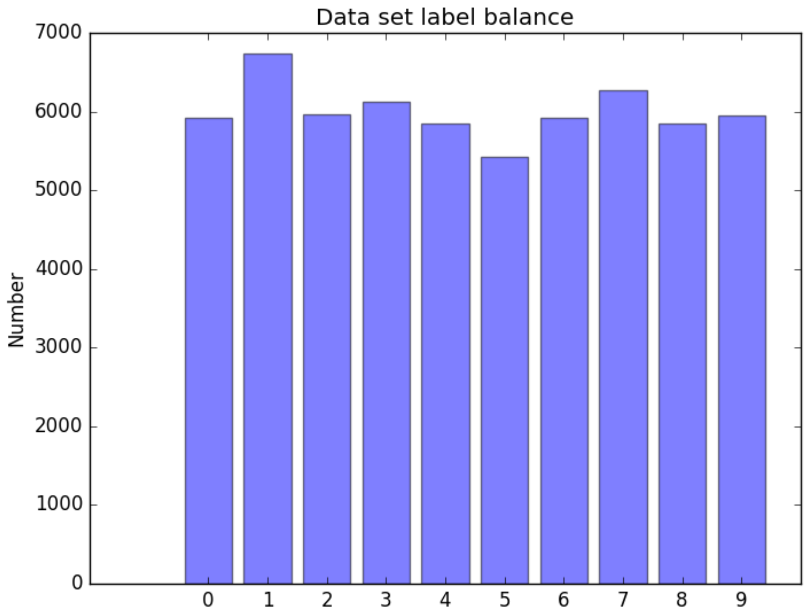
extra training data. In our work we don't use extra training data as it costs more computing time.

It comes in two formats: Original images with character level bounding boxes. MNIST-like 32-by-32 images centered around a single character (many of the images do contain some distractors at the sides).

In this work we use SVHN dataset with Original images with character level bounding boxes without extra training data.

The sample of SVHN is in figure 2. These are the original, variable-resolution, color house-number images with character level bounding boxes, as shown in the examples images above. (The blue bounding boxes here are just for illustration purposes. The bounding box information are stored in digitStruct.mat instead of drawn directly on the images in the dataset.) Each tar.gz file contains the original images in png format, together with a digitStruct.mat file, which can be loaded using Matlab. The digitStruct.mat file contains a struct called digitStruct with the same length as the number of original images. Each element in digitStruct has the following fields: name which is a string containing the filename of the corresponding image. bbox which is a struct array that contains the position, size and label of each digit bounding box in the image. Eg: digitStruct(300).bbox(2).height gives height of the 2nd digit bounding box in the 300th image.

Figure 3: Distribution of MNIST dataset



3 Exploratory Visualization

The distribution of MNIST training dataset seems Uniform which can been seen in figure 3.

SVHN dataset consists of sequences of digits. Most of the images have less than 4 digits in the training dataset. The most images have 2 digits in them as can be seen in figure 4.

4 Algorithms and Techniques

A Convolutional Neural Network(CNN) has been seen to be efficient for feature extraction for most image recognition problems. This is due to their ability to directly work on the raw pixels of image. We use the CNN architecture in shown in figure 5 for SVHN and CNN architecture for MNIST can be seen in figure 6.

Convolution Layer(CNN): In a CNN network a convolution layer is an input data filtering layer which consists of learnable filter parameters. These learnable parameters have a small receptive field but they extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map

Figure 4: Sequences of SVHN dataset

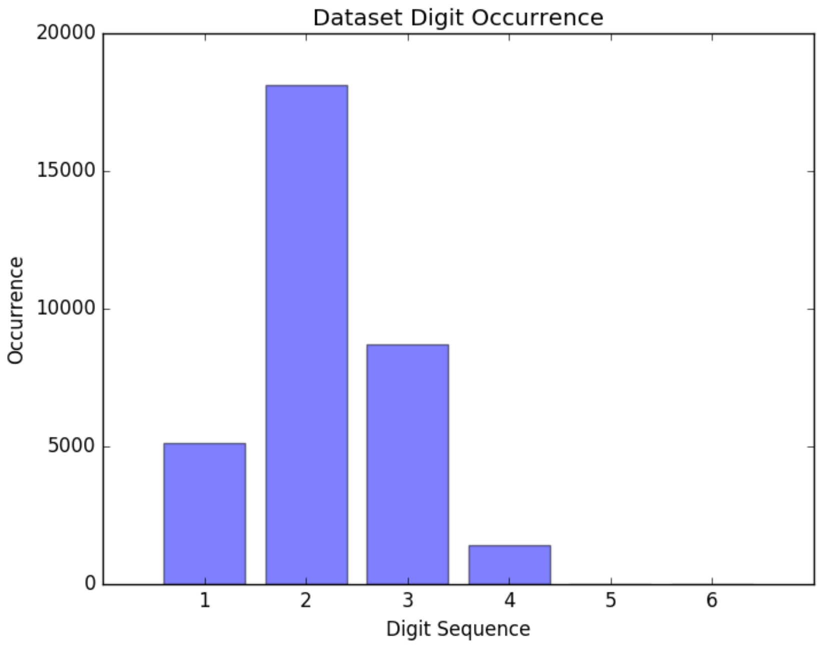


Figure 5: Architecture of CNN for SVHN

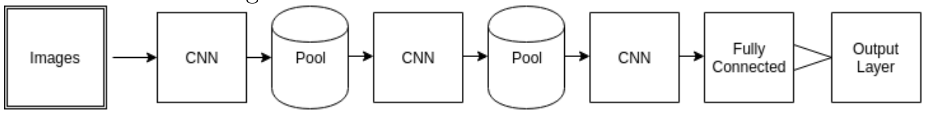
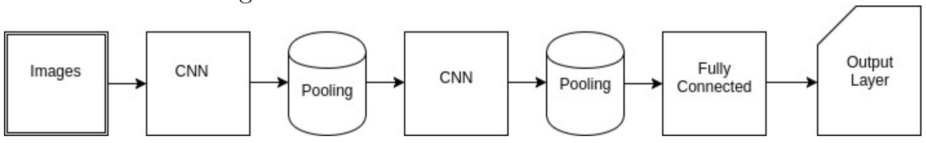


Figure 6: Architecture of CNN for MNIST



of that filter which is also called as the feature map. As a result, the network learns filters that activate when they see some specific type of feature at some spatial position in the input.

Pooling: Pooling is another important aspect in a CNN. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The intuition is that once a feature has been found, its exact location isn't as important as its rough location relative to other features. The function of the pooling layer is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer in-between successive conv layers in a CNN architecture. The pooling operation provides a form of translation invariance.

Rectified Linear Units (ReLU): These are used in the network to increase non linearity of the decision function without affecting the receptive fields of the convolution layer.

Fully Connected Layer(fully Connected): A FCN layer is used in a CNN to provide a high level reasoning to the work done by other CNV and pooling layers. This layer has connection to all the activations from previous layers.

Output Layer(output layer): Output layer is the final layer in a CNN which provides with the logits for the 5 digits detected by the network.

5 Benchmark

SVHN dataset has been studied in past and much more research has been done to read digits from images. Both MNIST and SVHN has very good accuracies in past research. MNIST has 99.79 percent accuracy rate as can be seen in this link[5]

SVHN has 96.98 percent accuracy reported by Goodfellow(Goodfellow, Bulatov, Ibarz, Arnoud, & Shet, 2014)

We decided to target an accuracy of anything above 90 percent for both SVHN and MNIST.

6 Data Preprocessing

MNIST[6] dataset can be downloaded in binary format. We extract the zip file using operating system tool before loading it using programming language. Each image is in shape of 28x28 pixels. We change it into one dimension which makes it 784 1D shape so the training set is in the shape of (60000,784) and testing set is in the shape of (10000,784). The labels are transformed into one hot encoding. so we get a shape of (60000,10) for training set and (10000,10) for testing set.

SVHN follows preprocessing as:

[5] http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html#4d4e495354

[6] http://yann.lecun.com/exdb/mnist/

1. Extract the structure of data from bounding box information provided in Digit struct file.

2. Crop the images to the region of the bounded box information that we got from Digit Struct file for images in test, train data.

3. we generate 32x32 size images from our dataset for training, testing and validation.

4. Find and delete all those data points which has more than 5 digits in the image. Only one image with id 29929 has more than 6 digits in the training set.

5. Now we have processed training set, validation set, and testing set which we save as pickle file.

We have 33402 images for training out of which we take 3000 for validation and our final training set has 30402 images, valiation set has 3000 images and testing set has 13070 images with multiple digits which count to 26032 digits.

7 Implementation

Since MNIST is the subproblem of SVHN and both the datasets similar in the approach, we will discuss implementation for SVHN only.

During implementation we do the following:

We first load the data from pickle file into training, testing and validation dataset.

We construct the following layered model for a CNN as described in Figure 5:

1) Convolutional layer with dynamic batch size (batch_size x28x28x32) and convolutional size (5x5x1x16).

2) Pooling layer with (batch_size x14x14x32)

3) Convolutional layer with (batch_size x10x 10x64 and convolution size: (5x5 x32)

4) Pooling layer with (batch_size x5x5x64)

5) Convolutional layer with (batch_size x1x1 x128) and convolution size:(5x 5x32 x64)

6) fully-connected layer with weight size: (64x16)

7) Output layer with size:(16x10)

The loss function used in this work is cross entropy. Intially we use a learning rate of 0.01 but later on switch to exponential decay learning rate with an initial learning rate of 0.05 which decays every 10000 steps with a base of 0.95. We use Adagrad Optimizer to minimize loss. We stop learning when we reach adequate accuracy level for the test dataset and we save the hyper parameters in a checkpoint file so that it can be loaded later when we need to perform detection without training the model again.

There were many complications and challanges which we got while designing this model. First we had to make the model learn, as sometimes the model doesn't seem learning from data, This is the first step in building the CNN. The next we had to figure out how many layers would be appropriate to get the CNN to extract all the features. If the model gets partial features it doesn't give good results. Since we are using Unsupervised Feature Extraction and Selection so there should be enough layers to extract the features. The other issue is to where to place Pooling layer with ReLu as sometimes we put Relu right after Convolutionl layer. The other things to figure out was what is best depth of convolutional layers and the batch size.

8 refinement

The initial model used for SVHN produced an accuracy of 85 percent with 14000 steps. The benchmark of above 90 percent was not far, then few refinements were introduced to . However, I further made some simple improvements to further increase the accuracy with few number of learning steps.

1. Added a dropout layer to the network after the third convolution layer just before fully connected layer, which randomly drops weights from the network with a keep probability of 0.95 to add more redundancy to the network. This allows the network to become more robust and prevents overfitting. 2. Introduced exponential decay to learning rate instead of keeping it constant. This helps the network to take bigger steps at first so that it learns fast but overtime as we move closer to global minimum, take smaller noisier steps.

With these changes, the model is now able to produce accuracy of 90.2 percent on test set with 15000 steps. As we are using only 73257 images for training although there are 230070 images for training when included with extra folder of dataset. It would mean that if we train for more data and give more time for training the performance would increase more however it is out of scope of this project as we have achieved the benchmark of 90 percent in test set.

9 Model Evaluation and Validation

For SVHN datset the final model is able to reach accuracy of 90.2% on test set which is slightly above the benchmark set for the project. I am saving the hyper parameters in a checkpoint file so that it can be restored later to continue training or to use it for detection over new images. Use of dropout ensures that the model is robust and does learn a redundant generalized model which can do well on most data. The model is tested over a wide range of input from the test dataset and generalizes well.

For MNIST dataset test accuracy is 0.9441% which is above the benchmark accuracy of 90%. The resaon for better performance of MNIST is because the images are more clear and visible as its a custom made dataset. The training set of MNIST is uniform which makes the accuracy to go up.

10 Justification

The accuracy of above 90% for SVHN dataset with only training set in use is a good sign and it can be improved with more data and computing power.

Figure 7: Sample result visualization

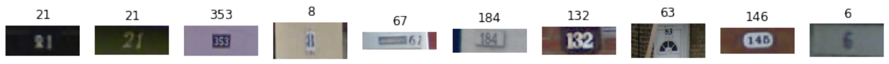
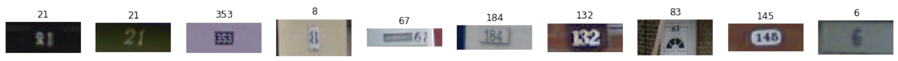


Figure 8: Sample True Labels



The results from the project show that the technique used is quite effective and may reach the high accuracy with specialized hardware and further tweaks and improvements in the architecture.

11 Free-Form Visualization

Sample results for visualization can be seen in figure 7. This model produces correct output for most images. However, the detection can fail if the image has distortion which makes it difficult to detect even for human operator and if digits are embeded in the image which look far away or at an angle making the detection difficult. In figure 7 and 8 we can see two images with true labels (83, 145) are predicted as (63, 146). This is because the 8 in 83 can sometimes feel like 6 even to the human and this image is far from the camera as we can see. and in other case 5 in 145 feels like 6 so the model comes up with 146 rather than 145. As we can see with naked eye that 5 in this image looks like 6 enough to confuse the model. These are the chellanges of machine learning and computer vision in general.

Overall if an image is clear and visible, the prediction of the model goes accurate.

12 Reflection

This project is to to recognize numbers in low resolution images captured by Google Street View cameras. Sometimes the images are not comprehensible even by the human eye. We aim to solve this problem by harnessing the power of Deep Learning and applying a Convolutional Neural Network with tuned hyperparameters to recognise numbers from such images. This Neural Network must also scalable and have the right balance between accuracy and computational time/complexity. The most interesting part of this project is finding the right hyperparameters. There are infinite possible permutations of the hyper parameters. It would be interesting to find out a combination which would predict the numbers on an image so blurry that even humans cant comprehend it. Identifying when the trade off between accuracy and computation time is the right amount is also crucial to training a CNN Model.

13 Improvements

This model can be improved by giving it more training data which is available in extra folder of SVHN dataset.

To scale this model a GPU support can be added to further speedup the process of training.

The other references which can be looked into for improvements are here(Ian J. Goodfellow, 2014-15),(Krizhevsky, Sutskever, & Hinton, 2012),(LeCun, Huang, & Bottou, 2004).

References

Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., & Shet, V. (2014). *Multi-digit number recognition from street view imagery using deep convolutional neural networks.*

Ian J. Goodfellow, J. I. S. A. V. S., Yaroslav Bulatov. (2014-15). Multi-digit number recognition from street view imagery using deep convolutional neural networks.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 25* (pp. 1097–1105). Curran Associates, Inc. Retrieved from `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`

LeCun, Y., Huang, F. J., & Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting.