Digit Recognition in Natural Images using Convolutional Neural Networks

Snigdha Kamal
Stony Brook University
snigdha.kamal@stonybrook.edu

Akshit Poddar
Stony Brook University
akshit.poddar@stonybrook.edu

Abstract

Multi-character text recognition in photographs of unconstrained natural environments is a tough computer vision problem. One of the complications in this area is recognition of house numbers on buildings in Street View Imagery. A system which successfully detects house numbers with high efficiency can be used to improve map accuracy. House number recognition also helps improve address geocoding. In most countries, geocodes of the majority of addresses are computed by interpolating the geocode of neighboring addresses. This could introduce significant errors and a poor user experience. As it is almost impossible to manually detect and transcribe billions of house numbers in images, there is a need of a digit recognition system that can automate this task. The aim of this project is to implement a system that can detect and transcribe house numbers from the SVHN data set automatically.

1. Introduction

Recognition of arbitrary multi-digits in natural images, such as Street View House numbers is a tough problem. The ability to automaticaly transcribe a house number from a street view image attached with a geo-code and then associate that number with the street address it represents can help pinpoint the exact location of an address with a high degree of accuracy.
While the problem of Optical Character Recognition on digits and documents is well researched, arbitrary digit recognition in photographs of natural environments is still highly exacting. This problem is further aggravated by factors such as the large variation in the visual appearance of the digits due to a wide range of fonts, colors, orientations and alignments as shown in 1 Environmental factors such as lighting, shadows, occlusions as well as image acquisition factors such as resolution and focus blurs further complicate the problem. Traditional approaches to this problem involve localization of numeric digits, segmenting them from the background followed by digit recognition. In this project, we attempt to combine all three steps into one using a Deep convolutional neural network framework that directly operates on the pixels in the image.

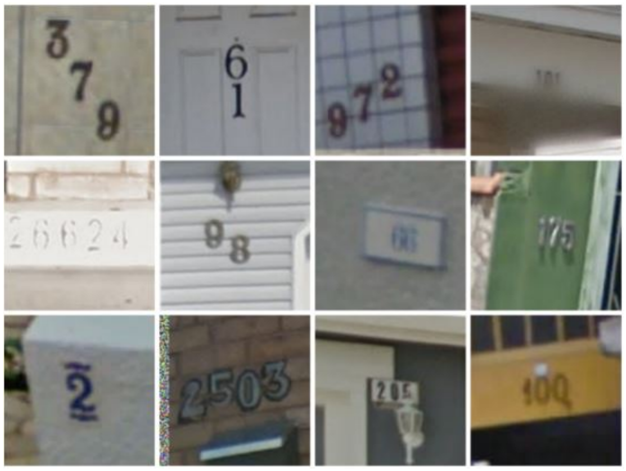

Figure 1: Multi-digit recognition in natural environment is a challenging task

2. Related Works

Convolutional neural networks (Fukushima, 1980; LeCun et al., 1998) [4] [1] are neural networks with sets of neurons made up of learnable weights and biases. Each neuron receives some inputs, performs a dot product and follows it with a non-linearity. The whole network transforms raw pixels on one end to class scores at the other. The loss function(SVM/Softmax) on the last layer helps minimize the error in prediction via back-propogation. Image based convolutional networks also use a pooling layer which summarizes the activations of adjacent filters with a single response using functions such as maximum or mean. These pooling layers render the system robust to small variations in input.
Traditional approaches to solving digit recognition separate out the localization, segmentation and the recognition steps. Netzer et.al.[5] establish that unsupervised feature learning methods are superior to hand-crafted features digit recognition in real applications. They also introduce the benchmark

Street View House Number dataset containing over 600,000 labelled digits cropped from Street View Images. Their paper draws a comparative analysis between hand-crafted features (such as HoG and Binary features) and two variants of feature-learning algorithms(K-Means and sparse auto-encoders) and reports the highest accuracy in digit classification using Stacked Sparse Auto-encoder(89.7%), establishing the fact that unsupervised feature learning methods are far superior to hand-crafted features for digit recognition in natural environments.
Goodfellow et.al.[2], in their paper propose a unified approach to multi-digit recognition in natural environment using Deep Convolutional Neural Networks which interacts directly with the image pixels. Their deep learning model achieved an accuracy of 96% in complete sequence recognition and a per-digit accuracy of 97.84% on the publicly available SVHN dataset.Their system has helped extract close to 100 million physical street numbers from Street View Imagery worldwide and increased the geocoding quality of Google Maps in several countries, particularly those that didn't already have good sources of geocoding.

3. The Street View House Number Dataset

The SVHN dataset was introduced by Netzer et. al. in [5] and obtained from a large number of Street View Images. The data-set has 10 classes, 1 class for each digit. Digit '1' has label 1 and '0' has label 10. It is divided into three subsets:

- Train: 73257 digits
- Test: 26032 digits
- Extra: 531131 additional digits (less difficult samples to use as extra training data).

The SVHN data-set comes in two varieties:

- Full Numbers: The original, variable-resolution, color house number images. Each image comes with a transcription of the detected digits and digit level bounding boxes.
- Cropped Digits: This format is similar to the MNIST format where all digits have been resized to a fixed resolution of 32x32 pixels.

Owing to GPU limitations and a long training time, we split the training dataset into two parts - 60% of the dataset (around 43954 images) was used for training and the rest 40% of the dataset was used as the test dataset for this project.

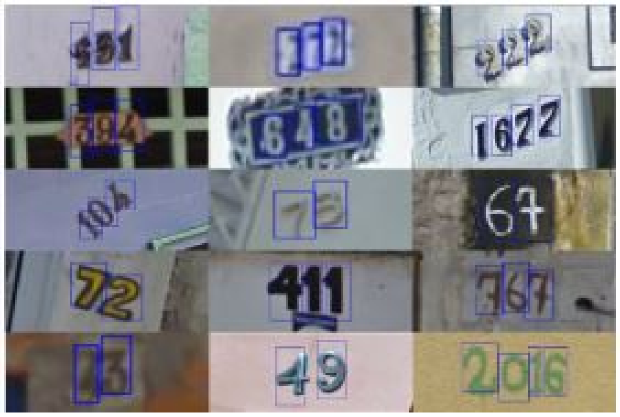
Figure 2: Samples from the SVHN Dataset

4. Problem Description

Recognition of house numbers in street view involves identification of a sequence of digits, s=s_1,s_2,..s_n. The accuracy of the system is determined by the proportion of images for which the entire house number is detected correctly. No partial credit is given to correct identification of the individual digits as a building can only be found on a map if its entire address is transcribed correctly. Another special property of house numbers that can be leveraged for the task of digit recognition is that the length of house numbers is bounded, limiting the length to at most 5 digits.
The basic approach in a deep learning framework is to train a probabilistic model given the digit images. Let X represent the input digit image and S represent the output of the network. S is defined as sequence of N random variables, $S_1 ... S_N$ representing the digit sequence and a random variable L which represents the length of the sequence (to impose the length constraint). Assuming that the probability of each digit is independent of the other, P(S|X) can be modeled as shown in Equation 1

$$P(S = s|X) = P(L = n|X) \prod_{i=1}^{n} P(S_i = s_i|X) \qquad (1)$$

If a single digit is being detected at a time, then value of L is fixed to 1, else it is fixed to 5, the maximum possible length of a street number.
We wish to learn a model P(S|X) by maximizing log P(S|X) on the training SVHN dataset using a generic method like Stochastic Gradient Descent. However, we have employed a better level optimization technique called the Adam Optimizer as it never gets stuck in a local minima and converges faster while keeping overfitting to a minimum. The softmax layer at the end uses the conventional backpropogation method for error propogation, followed by the adam optimizer for updation of weights. At test time, the network predicts S as shown in equation 2. This parameter is com-

puted for each digit independently and the log probabilities for each digit are added up to yield the output for the entire sequence of digits in an image.

$$s = (l, s_1, ...s_l) = argmax_{L,S_1,....S_L} logP(S|X) \quad (2)$$

5. Methodology

For our project we have used the deep learning library Tensorflow for the network implementation. Tensorflow is a machine learning toolkit published by Google for expressing and executing machine learning algorithms. It is a flexible system and can express a wide variety of algorithms including training and inference algorithms for deep neural networks and has been succesfully deployed in speech recognition, computer vision, robotics etc. Tensorflow comes with Tensorboard which we used to generate the graphs and visualize the internals of the ongoing computations in the network.

5.1. Experiments with the MNIST dataset

In the first step of our experiments with the Convolutional Neural Networks and Tensorflow, we implemented a simple regression model using Softmax Regression. The MNIST dataset is split into three parts, 55,000 data points of training data, 10,000 points of test data and 5,000 points of validation data. A softmax regression has two steps: first, add up the evidence of an input being in particular classes, and then convert that evidence into probability.After doing a weighted sum of the pixel intensities, the weight is negative if that pixel having a high intensity is evidence against the image being in that class, and positive if the evidence is in favor. The convolutional neural network architecture is based on LeNet. The code was run on four year old i7-Core CPU (2.5 GHz) with tensorflow in CPU-only mode and took around 50 minutes to compute the final accuracy of 91.3%

The next experiment was implementing a deep convolutional neural network with two hidden layers. Every digit is a 28x28 size image flattened into a 784 feature vector. There are 10 labels in total, one for each digit. After assigning initial weights and biases, and a softmax regression model, we implemented the stochastic gradient descent with step size of 0.5.The convolutions use a stride of one and pad of zero with max pooling over 2x2 blocks. We also included a final dropout layer in the architecture which helps reduce overfitting. We received an accuracy of 98.7% after a training period of eight hours.
After these preliminary steps, we built the neural network framework for digit recognition in the SVHN dataset. The details of the architecture are described in Section 5.2

5.2. Network Architecture

Every image in the cropped format of the SVHN dataset is of size 32x32 pixels and this is the input to the network. The images are pre-processed by subtracting the mean from the image. This was done to combat the wide variations in lightning conditions due to environmental factors. The neural network is built in Tensorflow and consists of two stages. Each stage consists of Convolutional-Relu-Local Response Normalization-Pool layers stacked together. Local response normalization is done after every stage in the network. This layer implements lateral inhibition which is necessary as ReLu neurons have unbounded activations and Local Response Normalization serves to normalize them while yielding a large response for the high frequency features. Two Fully connected layers follow the two stages and their output is then passed to a softmax function. Cross entropy function is used as the distance metric to compute the error between the true and predicted labels as shown in equation 3

$$C = \frac{-1}{n} \sum x[ylna + (1 - y)ln(1 - a)] \quad (3)$$

where x is the input to the network, y is the true output and a is the output predicted by the network.

Adam Optimzer is applied for stochastic optimization and weight updation. The size of the convolution kernels is 5x5. The size of the stride is 2. Every convolutional layer includes max pooling. The window size for max-pooling is 2x2.

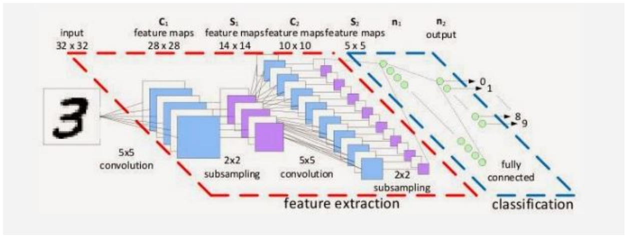

Figure 3: CNN Architecture for digit recognition in Natural Environment using SVHN Dataset

5.3. Adam Optimizer

Adaptive Moment Estimation(Adam)[3] is a convergence technique which computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients v_t, Adam also keeps an exponentially decaying average of past gradients m_t.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (4)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (5)$$

In equations 4 and 5, m_t and v_t are estimates of the first moment(the average) and the second moment (the uncentered variance) of the gradients respectively. This particular process gives the optimizer its name. As m_t and v_t are initialized to zero vectors, they are biased towards zero, particularly during the initial time steps, and also when the decay rates are small (i.e. β_1 and β_2 are close to 1). These biases are then counteracted by computing bias-corrected first and second moment estimates:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{6}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{7}$$

These are then used to update the weight parameters yielding the Adam update rule:

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t \tag{8}$$

The authors of Adam Optimzer propose default values of 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ϵ.
Adaptive Moment Estimation keeps separate learning rates for each weight as well as an exponentially decaying average of previous gradients.
Following is the algorithm for computing Adaptive Moment Estimation and subsequent updation of weights:

1. The gradient and its element-wise square using the current parameters is computed.

2. The exponential moving average of the 1^{st} order moment and the 2^{nd} order moment is updated.

3. An unbiased average of the 1^{st} order moment and 2^{nd} order moment is calculated.

4. The weight update is computed as: 1^{st}-order moment unbiased average divided by the square root of 2^{nd}-order moment unbiased average (scaled by learning rate).

5. The update is applied to the weights.

6. Results

The network was trained for around 18 hours on a i5-Core CPU (1.78 GHz) with Tensorflow in CPU-only mode.
Figure 4 shows the digits which were correctly classified by the network. The network displays robustness against variability in light, resolution and blur.

The poorly classified digits are shown in Figure 5 where the network either gave the incorrect output or got confused between two outputs.

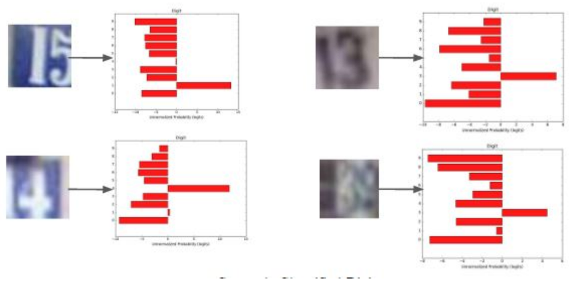
Figure 4: Correctly classified Digits

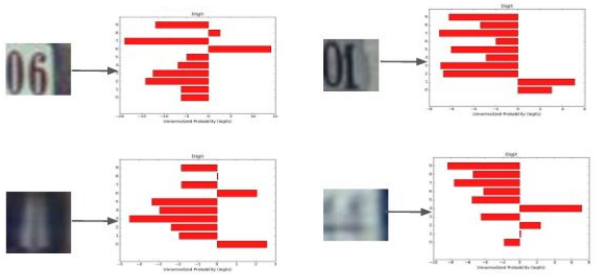
Figure 5: Poorly classified digits

After 18 hours of training, the network converged to 96.88% accuracy on the Train dataset. Figure 6 shows the plot of Train accuracy versus the number of iterations. The network was trained over 10^6 iterations. The network converged to the train accuracy of 93% within the first two hours of training and to a final accuracy of 96.88% after 18 hours

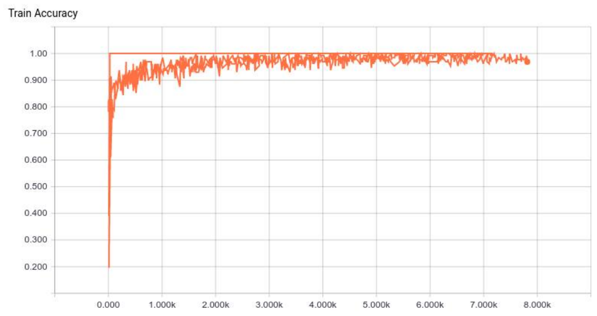

Figure 6: Train Accuracy

The graph of Test Accuracy versus the number of iterations is shown in Figure 7, where the final test accuracy is 90.87%.

Figure 8 displays the loss of the network, where the loss converges to a minimum efficiently due to the ADAM optimizer.

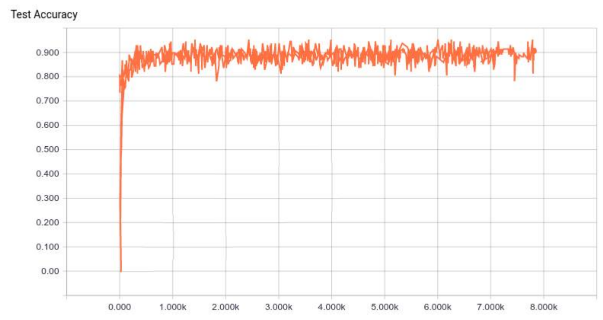

Figure 7: Test Accuracy

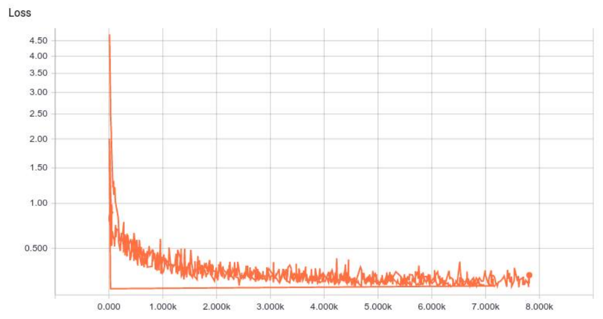

Figure 8: Loss

The visualization of neuron activation in the first convolutional layer in Figure 9 shows how the network is learning to recognize different features in the image such as corners and edges by adjusting the filter weights accordingly. These learned features will then be input to the following layers and transformed into higher level features. The filters in the last layer will begin to resemble the input image indicating that the network has effectively learned the input pattern.

7. Comparative Analysis

Table 1 demonstrates a comparative analysis of our model with the works proposed before and referenced below. Our network performs significantly better than the hand-crafted features (Histogram of Oriented Gradients and Binary features). Our system also achieved a higher accuracy in comparison to the unsupervised feature learning methods(K-means and Stacked Sparse Auto Encoders) whose highest accuracy went up to 90.6%. However, our model with an accuracy of 90.87% could not level the state-of-the-art accuracy of 97.84% on per-digit character recognition. This drop in accuracy could be attributed to the fact that the entire SVHN dataset was not used (only the train dataset was employed to both train and test the network as described above) owing to GPU limitations. Our training time was also low in comparison to the state-of-the-art method.Our model was not fine-tuned and dropout was not applied to the hidden layers either.

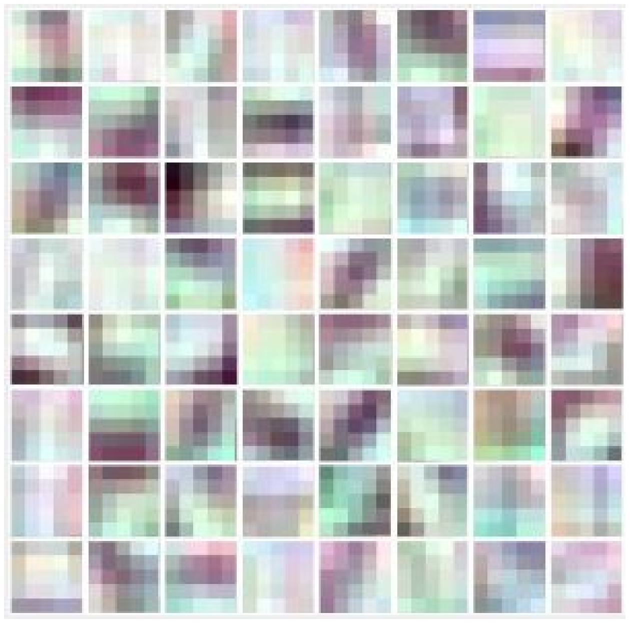

Figure 9: Visualization of neuron activation in the first convolutional layer

Method	Frobnability
HOG[5]	85.0%
Binary Features[5]	63.3%
K-means[5]	90.6%
Stacked Sparse Auto Encoders[5]	89.7%
ConvNet/MS/L4	94.85%
GoodFellow et. al.[2]	97.84%
Our Network	90.87%
Human Performance	98.0%

Table 1: Comparative Analysis

8. Conclusion

In this project, we learnt that recognition of digits in natural environments is a tough vision problem that when solved efficiently can lead to better navigational systems and address geocoding. A deep learning framework using convolutional neural networks is capable of yielding high accuracy for digit recognition in natural and noisy environments. Our convolutional neural network managed to recognize digits in the SVHN dataset, with a per-digit recognition accuracy of 90.87%. The current state of the art model which utilizes a deep learning framework yields a per-digit recognition accuracy of 97.84% which is very close to the human accuracy of 98%. This model has been scaled up and is currently being used to improve the accuracy of Google Maps, having transcribed over 100 million images of house numbers.

References

[1] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[2] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013.

[3] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[5] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.