Recognizing SVHN Numbers Using CNN Nets

(group number and name here)

Haseeb Shehzad (601483)   Naufal Khalid (546687)

January 31, 2018

Abstract

Recongnizing the charactors in a photograph is an interesting and important task. We have tried to recongnize the digits from streeet view images. The data have some natural imagery so it is challenging as a benchmark data for the research of new methods and techniques in object recognition.Recognizing the charactors from images have many useful applications like detecting a language, text from a book and identifying the house numbers for better navigations services. We used convolutional neural networks to identify the digits. We applied all the developed techniques related to CNN and tried to compare the results with state-of-the art results. We pre-processed the data, applied batch normalization, Adam optimizaton algorithm, different iterations and finally we tested our ConvNet with and withour color images. Our result are compareble to the most of the results mentioned in related work.
(1 point)

1 Introduction

We have seen a significant success in object recognition using convolutional neural network in the past decade. The image classification problem is scientifically solved problem. One of the popular and important problem is image classification of digits data either hand-written or typed. A great deal of research and effort applied to hand writing recognition and systems can now perform tasks comparable to human capabilities. We have good example of MNIST dataset that has been studied thoroughly and many successful tasks performed on it. We have another challenging problem of detecting image data of scenes that gained popularity in academia and industry. In this project, we will focus on the image data of house numbers and try to detect the digits from these images. Traditional computer vision methods of using the prior information about the structure of the images has failed when compared to the accuracy achieved by applying the state-of-the art neural networks to recognize these types of images. Our aim is to construct a convolution neural network for SVHN image data, experiment with different methods, models and algorithms and compare our findings. We tried to detect the digits using system almost to the extent of human capability to recognize digits. We would train our convolutional neural network model to recognize digits as accurately as possible.

Convolutional neural network is a special type of artificial neural network. The basic building blocks of Convolutional layer are convolution operation, rectifying non-linearity and pooling. Convolution is an operation between input and kernel function (parameters) resulting a feature space. This operation allows sparse weights across the network, enabling us to focus on significant portions of data while skipping the need to take into account all the parameters. Pooling is useful in handling minor changes in the input making them invariant to translations to small extent.

We have used the publicly available real-world data of house number digits extracted from Google Street View data. The data require minimal formatting and pre-processing. The data is similar to the MNIST dataset. However, the SVHN data is more challenging and useful because of the size of the data and other information contained in the images like color and image background of digits. The task is to identify digits from given images and successfully classify them in the future.
(2 points)

2 Related work

SVHN dataset is extensively studied, implemented and improved using the CNN and other object recognition and computer vision methods. Here we would review some of the results, achieved accuracy and improvements based on the CNN architect used [1]. Pierre et al [2], achieved 95,1% classification accuracy by using two-layer, non-linear architecture of CNN with 20 hidden neurons. They used the stochastic gradient descent as optimization method and pooling. Mark D and Tony Vladusich [3] showed the competitive results (96%) with the state-of-the art results at that time using the filters which randomly selected patches from images and least square regression to train the classifier for CNN. The classification method was three- layer, nonlinear network. Weights updated by the least square regression. Stochastic process for pooling replacing the deterministic pooling was applied as an improvement to the CNN for SVHN dataset was applied by Goodfellow et al [4]. The state-of-the art accuracy of 97,2% was achieved using this technique. An improvement to the droupout optimization technique was proposed as maxout which improved state-of-the art classification results of various benchmark datasets [4]. The accuracy achieved was 97,53%. ReNet [5] network was proposed as an alternate to the deep convolutional neural network replacing the convolution, pooling layer and the accuracy was

improved to 97,62%. An approach with unified model that integrated the localization, segmentation, and recognition steps [6]. It was also proposed that the increased performance with increased number of hidden layers and tested with eleven hidden layers, achieving the state-of-the art results with 97,84% accuracy [7]. Liao et al [8] proposed a Recurrent neural network for the recurrent connections within each convolutional layer reducing 1,77 % error. A better pooling method as a combination of max and average pooling and tree-structured pooling filters reducing error to 1,69 % [9].
(2 points)

3 Method

This project was implemented using Convolutional Neural Networks(CNNs). CNNs are special type of neural network models which are made up of different layers of neurons having learnable weights and biases. Figure 1 shows the top view diagram for a CNN. There are 4 stages in a CNN, convolution/filtering, pooling, passing through a non-linearity function and classification. In the convolution stage, an image of higher dimensions is transformed into a smaller image by extracting the most important features and taking out their dot product, known as the feature map. This process is also known as filtering as we filter the image by taking out the most important features. The process, can be repeated several times, using sliding filter in order to attain feature maps in different sizes.

The featured maps are passed through a non linear activation function such as the RELU, which performs element wise absolute value operation. This function, also detects the features with non-linearity in the data. In the pooling stage, the featured maps attained from the CNNs are reduced further. We set a sliding window (2, 2) in our case, going through all the features and taking the maximum values from them. This transforms the image in such a way that only the most dominating features are retained.

In the last stage, which is the fully connected layer is a multilayer perceptron. It computes the scores/ votes for each of the classes. This layer uses a softmax activation function to output the probabilities of class scores. This layer also flattens the feature maps into vectors.

We also use the drop out layer and the output layer for our Convolutional neural network. The dropout layer helps to reduce overfitting of data as it activates only random neurons during then training process. The output layer is the final layer which takes the probabilities, generated by the fully connected layer as input and assign them to the given classes. This layer performs the classification process. The original image is reduced to the number of output classes.

Back propagation is performed here so that the model learns from errors and can be trained to improve classification.

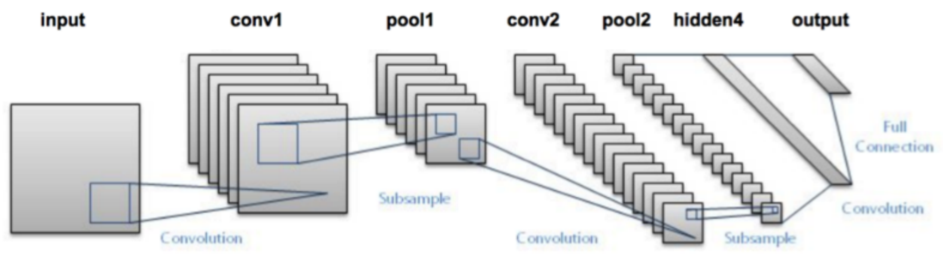
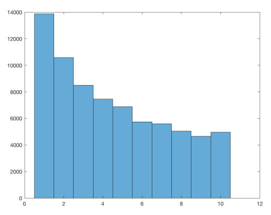
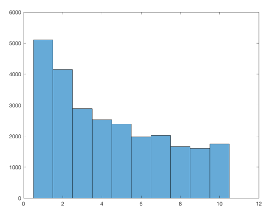


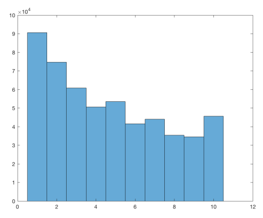
Figure 1: CNN network



(a) Training set



(b) Testing set



(c) Extra set

Figure 2: Class Distribution

(3 points)

4 Data

SVHN is one of the bench marks data sets used for the validation of new methods, applications and algorithms related to object recognition, deep learning, clustering and computer vision. Data is available from the on the following resource and can be used publicly free [10].

This dataset requires very little pre-processing. The already done processing on the dataset includes the per digit fix size of 32*32 fixed size images. The varied level of brightness and different coloured images were challenging to deal with the data. The digits are sometimes cluttered with the nearby digits also making it more challenging to identify by system. We used different techniques to deal with these challenges as mentioned below. The data contains fixed resolution images with colour channels. We have data available in training (73257), testing (26032) and extra-testing (531131) subsets. The data we used is already cropped and each image contains one digit, in (.mat) format. The data is available in vectors representing (Red-Green-Blue) values of each pixel. We have two variables in the data contains the images and contains the class labels.

Class distribution of data is shown in figures 2.

Normalization: We used the normalization to reduce the intensity of

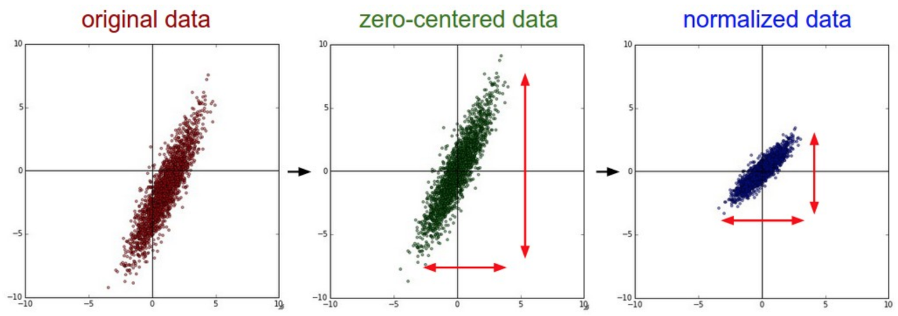


Figure 3: Effect of data pre-processing

different colors but it did not change the variety of colors. Normalization was also applied to achieve the same scale of data dimensions. We tried this in three ways as mostly proposed by different researchers.

- To reduce the range of the values of images to [1, -1] we divided the values (0-256) by 128 and subtracted the value by 1.
- Standard normalization achieved by dividing the mean from each value. We also subtracted the standard deviation from each data point.
- Batch normalization was implemented using the Tensorflow function.
- Gray scaling was also done as we were not interested in the colour but the shape of the image (contained digit). This step was essentially important for the overall results that we achieved although in some cases we had poor classification results because of gray scaling.

The data visualization after preprocessing is shown in figure 3 below taken from notes [11].
(3 points)

5 Experiments

The training data went through different rounds of experiments, before we chose the optimal one. The input data was passed through a batch normalization (an inherent function from Tensor Flow) which reduces the skewness and variability in the data. The parameters for batch normalization were normalization offset β, normalization scale γ and normalization epsilon ϵ. Experiments were performed using different values of these parameters and this resulted in an improved performance.

The normalized data was fed to the 3 layer, convolutional neural network. Each layer, had its own filtering/ convolution, pooling (We used max pooling) and linear activation function (RELU). Experiments were conducted using different filter size, the size of each layer, and the different activation functions. For our ease, we have kept the batch size constant to 50. We had to experiment the results on a limited number of iterations 20000 since, we had some hardware limitations.

Following the convolutional layers, the data was passed through the fully connected layer, having a Multilayer Perceptron structure. A softmax prediction using cross entropy was produced for each of the image.

We did some research on the optimizer to use for the experiments and ended up using The Adam Optimizer, since it caters the best optimization results. The number of hidden layers used for our experiments were 128.

Gray scaling was implemented as a part of our experiments just and various results were conducted with and without gray scaling
(5 points)

6 Results

With various rounds of experiments conducted, we were able to attain a mean accuracy of 91% of the test data. Table 6 shows the mean accuracy on the test data using different variations in the learning parameters.

Normalization(BN)	Color	Layers	Filters	layer size	iterations	Accuracy
$(\beta = 0.0, \gamma = 1.0, \epsilon = 0.001)$	YES	128	5×5	(16, 32, 64)	20000	91.2 %
None	NO	128	5×5	(16, 32, 64)	20000	20.8 %
$(\beta = 0.0, \gamma = 1.0, \epsilon = 0.001)$	YES	128	3×3	(16, 32, 64)	20000	88.8 %
None	YES	128	5×5	(16, 32, 64)	20000	83.6 %
$(\beta = 0.0, \gamma = 1.0, \epsilon = 0.001)$	YES	128	5×5	(32, 64, 128)	20000	87.8 %
$(\beta = 0.0, \gamma = 1.0, \epsilon = 0.001)$	YES	128	5×5	(8, 16, 32)	20000	84.8 %
$(\beta = 0.0, \gamma = 1.0, \epsilon = 0.001)$	YES	128	5×5	(16, 32, 64)	10000	89.8 %
$(\beta = 0.0, \gamma = 1.0, \epsilon = 0.001)$	NO	128	3×3	(16, 32, 64)	20000	87.4 %
$(\beta = 0.0, \gamma = 1.0, \epsilon = 0.001)$	NO	128	5×5	(32, 64, 128)	20000	85.6 %
$(\beta = 0.0, \gamma = 1.0, \epsilon = 0.001)$	NO	256	5×5	(32, 64, 128)	20000	90.8 %

Figure 4 shows that the CNN classifier is able to classify the SVHN data correctly.

Figure 5 shows the plot of accuracy on the training data per batch and the training loss in data per batch. The iterations are 20000 and the batch size is 5000.

Figure 6 results shows the accuracy of the CNN nets implemented showing accuracy of 90.5 %
(5 points)

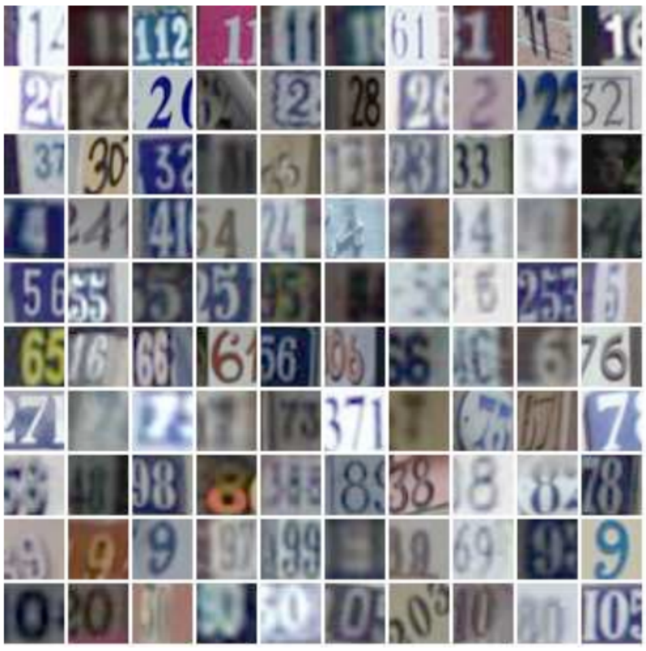


Figure 4: Classification of numbers by the CNN classifier

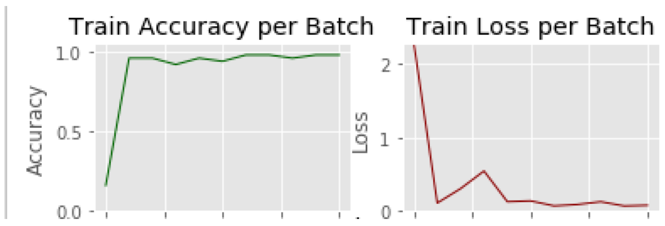
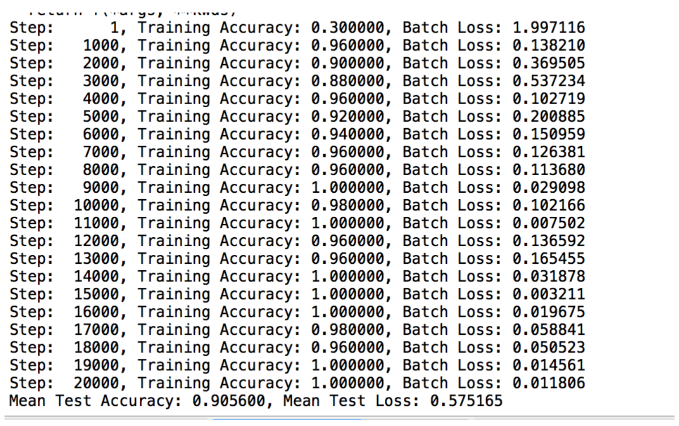


Figure 5: Data accuracy on training data



```
Step:      1, Training Accuracy: 0.300000, Batch Loss: 1.997116
Step:   1000, Training Accuracy: 0.960000, Batch Loss: 0.138210
Step:   2000, Training Accuracy: 0.900000, Batch Loss: 0.369505
Step:   3000, Training Accuracy: 0.880000, Batch Loss: 0.537234
Step:   4000, Training Accuracy: 0.960000, Batch Loss: 0.102719
Step:   5000, Training Accuracy: 0.920000, Batch Loss: 0.200885
Step:   6000, Training Accuracy: 0.940000, Batch Loss: 0.150959
Step:   7000, Training Accuracy: 0.960000, Batch Loss: 0.126381
Step:   8000, Training Accuracy: 0.960000, Batch Loss: 0.113680
Step:   9000, Training Accuracy: 1.000000, Batch Loss: 0.029098
Step:  10000, Training Accuracy: 0.980000, Batch Loss: 0.102166
Step:  11000, Training Accuracy: 1.000000, Batch Loss: 0.007502
Step:  12000, Training Accuracy: 0.960000, Batch Loss: 0.136592
Step:  13000, Training Accuracy: 0.960000, Batch Loss: 0.165455
Step:  14000, Training Accuracy: 1.000000, Batch Loss: 0.031878
Step:  15000, Training Accuracy: 1.000000, Batch Loss: 0.003211
Step:  16000, Training Accuracy: 1.000000, Batch Loss: 0.019675
Step:  17000, Training Accuracy: 0.980000, Batch Loss: 0.058841
Step:  18000, Training Accuracy: 0.960000, Batch Loss: 0.050523
Step:  19000, Training Accuracy: 1.000000, Batch Loss: 0.014561
Step:  20000, Training Accuracy: 1.000000, Batch Loss: 0.011806
Mean Test Accuracy: 0.905600, Mean Test Loss: 0.575165
```

Figure 6: Classification results from the classifier

7 Discussion

As you can see from Table 6 that batch normalization was an important part of the experiments. Without batch normalization, the accuracy dropped to 20%. Changing the image to gray scale also affected the accuracy of the classifier. You can see from the results that the accuracy dropped, when we used gray scale image instead of the color images. Changing the filter size did reduce the accuracy, however it was not a drastic drop. The highest accuracy achieved was 91.2 %. The iterations we kept constant to 20000 throughout our experiments, whereas we changed it to 10000 in one of our experiments. We noticed that the accuracy decreased to 89.8%. If we changed the layer size to (32, 64, 128), the accuracy was 87.8 % and when we changed it to (16, 32, 64) the accuracy decreased to 84.8 %. If we changed the number of hidden layers to 256, there was a minimal drop in accuracy and it was recorded at 90.8 %.
(4 points)

8 Conclusions

After experimenting with different learning parameters, we come up with the conclusion that data had to be normalized before training as there was alot of noise in the data. Using Tensor Flow helped us as its syntax was not difficult and had ample amount of documentation and tutorials for CNN network. We were able to classify data accurately with an accuracy of 91%. We implemented the system on our own laptops and since we had limited memory, we were had to limit the iterations to 20000. We were not able to use the GPU feature of Tensor flow however, it would have increased the execution speed and performance. CNN nets were able to successfully classify SVHN data even though it had alot of noise. We did not experiment with the extra dataset, however we did some research that using the extra data set can improve the accuracy. This was the reason, we can not achieve the accuracy similar to the state of the art accuracy which is 98 %. Changing the image to gray scale, reduced the accuracy of classification by almost 5%.

For further improvements on this project, we can try using the extra data set and try adding more hidden layers to see the effect of accuracy.
(3 points)

References

[1] A. Einstein, “Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies],” *Annalen der Physik*, vol. 322, no. 10, pp. 891–921, 1905.

[2] P. Sermanet, S. Chintala, and Y. Lecun, “Convolutional neural networks applied to house numbers digit classification,” 04 2012.

[3] M. D. McDonnell and T. Vladusich, “Enhanced image classification with a fast-learning shallow convolutional neural network,” *CoRR*, vol. abs/1503.04596, 2015.

[4] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, “Maxout networks,” in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML’13, pp. III–1319–III–1327, JMLR.org, 2013.

[5] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. C. Courville, and Y. Bengio, “Renet: A recurrent neural network based alternative to convolutional networks,” *CoRR*, vol. abs/1505.00393, 2015.

[6] I. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, “Multi-digit number recognition from street view imagery using deep convolutional neural networks,” in *ICLR2014*, 2014.

[7] M. Liang and X. Hu, “Recurrent convolutional neural network for object recognition,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3367–3375, June 2015.

[8] Z. Liao and G. Carneiro, “Competitive multi-scale convolution,” 11 2015.

[9] C. Y. Lee, P. Gallagher, and Z. Tu, “Generalizing pooling functions in cnns: Mixed, gated, and tree,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, pp. 1–1, 2017.

[10] “The street view house numbers (svhn) dataset.”

[11] “Cs231n convolutional neural networks for visual recognition.”

(2 points)

9 Roles of the authors

The project and report was a combined effort between us. We did pair programming on project and divided the different sections of report between us.