SVHN CNN Ensemble Classification and CNN Detection

Domenic Curro
Ryerson University MSc. Candidate
Toronto, ON M6C1M2
Domenic.Curro@gmail.com

Abstract

The classification of digits in-the-wild is a nigh solved Computer Vision problem even with the wide variety of the shapes and forms that digits can take. In this paper we attempt to obtain similar results to the state-of-the-art using a very well known and very simple Convolutional Neural Network architecture, to classify and further, to detect, house numbers from street level photos provided by the Street View House Number (SVHN) dataset. We also introduce an 11th class to the SVHN data set: background, to aid in the problem of detection.

1 Introduction

1.1 The Problem

Images of digits in the wild can suffer from motion blur, sizing issues, awkward angles, fish-eye lenses, and becoming dirty. A few data sets have been proposed containing digits of this kind, with this style of natural distortion: The Street Stanford University View House Number dataset.

1.2 The SVHN data set

The SVHN data set is a data set containing digits from the Google Street View level, and comes in two formats: the first is raw images containing a scene with a digit placed somewhere about the scene, and second is a series of 32x32 cropped and scaled wild digits. We will primarily be focusing on the second format of this data set, for the purposes of classification.

The SVHN format 2 data set consists of a Training set containing 73,257 examples, an easier Extra training set containing 531,131 example, and a Test set containing 26,032 examples.

As the number of simple examples massively outweighs the number of difficult examples, we supplement the difficult set by generate augmented seven 32x32 augmented randomly rotated (+/-15 degree) versions. A similar grayscale version of our data set is created in parallel to determine if colour helps or hinders the classification process.

1.3 Single Network

Taking inspiration from the success of LeNet on the hand written digit problem, modeled by the MNIST data set, we purport that the minimalistic nature of the LeNet architecture used to solved MNIST will also be suitable to solving our in-the-wild digit problem. The nature of the problem is similar and thus we believe that this architecture is well suited for our purposes.

1.4 Ensembles of Networks

We also hope to demonstrate that an ensemble of neural networks can be superior to one neural network.

2 Related Work

There has been much headway in the solving of the classification from as seen in [1] and continuing through [2] and finally solved with [3]. The same progress has been made with detection, as seen in [4].

3 Approach

Before we begin training our nets, we shuffle and split the data into a training set with 929,898 examples, and a Validation set with 187,189 examples.

As for the bootstrapping process, we create 20 neural networks of the same architecture as the single net, 10 of which have $1/10^{th}$ of the original data with a guaranteed coverage of 100%, and the remainder contain random samples of $1/10^{th}$ of the original data, a combined 200% data coverage.

3.1 Single LeNet MNIST Architecture Network

For our Neural Network we chose to go with the LeNet architecture which was highly successful at solving the MNIST challenge. The network was trained using the same settings as the successful network, including hyper parameters such as batch size and learning rate.

3.2 20 Ensemble of MNIST

The architecture is identical to the single net. The output of the twenty networks is pooled, and applying a hard max we recover the maximum vote.

4 Results

As we seen in Table 1, our top result is 92.77% and it's clear that there is no benefit to training with or without a gray scaled set of data. The variance in our results can be attributed to a lack of retraining our network received. Due to automation issues we were limited to a small number of training attempts.

Table 1: SVHN Test Scores Reported by [2] Including My Results (red)

Algorithm	
Binary Features (WDCH)	63.30%
Hog	85.00%
Ensemble (MSNT)	87.95%
Stacked Sparse Auto-Encoders	89.70%
K-Means	90.60%
ConvNet/MS/Average	90.75%
LeNet (MNIST)	90.85%
ConvNet/MS/L2/Smaller training	91.55%
LeNet (MNIST) / Background / Gray	91.95%
LeNet (MNIST) / Background	**92.77%**
ConvNet/SS/L2	94.28%
ConvNet/MS/L2	94.33%
ConvNet/MS/L12	94.76%
ConvNet/SS/L4	94.85%
Deeply Supervised Nets	**98.08%**
Human Performance	**98.00%**

Table 1: SVHN Test Scores by Ensemble Member

Ensemble Member		Ensemble Member	
1	85.00%	10	81.92%
2	83.08%	11	83.85%
3	84.62%	12	84.41%
4	86.15%	13	85.81%
5	86.54%	14	82.69%
6	83.77%	15	86.15%
7	82.70%	16	84.23%
8	85.00%	17	86.54%
9	84.62%	18	86.15%

Interestingly, the total Ensemble accuracy is 6.05% greater than the worst Member of the Ensemble, and 1.43% greater than the best Member of the Ensemble, but 4.82% lower than our top result. This improvement definitely speaks to the progress an ensemble can make.

The ensemble reported near perfect agreement upon 18905 examples of the test set, had a clear majority in 5705 examples of the test set, and finally had a thin majority in 1422 examples of the test set. This thin majority implies that 5.46% of the data was uncertainly classified, and thus indicates to us that we likely needed more ensembles, as there is still at least room to perform 4.82% better as demonstrated by the single network.

Alas, in the end I do believe that we may have reached the upper limit of this borrowed Network Architecture. I don't doubt that we have hit some irreducible error caused by an assumption in the architecture of our model, as it was designed for a similar but different task. As Table 1 has shown, there exist Neural Networks better suited for the complexity of this task.

4 Detection

4.1 Introduction

A natural extension to classification is detection. Classification is classily viewed as a function that takes in a set of data and outputs a class, but it can also be looked at as a function that takes in a set of data and tells you whether the input was of class. To further this distinction, consider a Neural Network trained to classify road vehicles of classes = {car, truck, bike}. If we only cared if a car was in an image, we could give the classifier a picture and check if the result was a car, treating every other result as not-a-car.

We will be using Single Network, trained on SVHN and our Single Network trained on our Background-extended SVHN data set, to do exactly detect digits in the very same vein as described in the example earlier.

4.2 Background Data Set

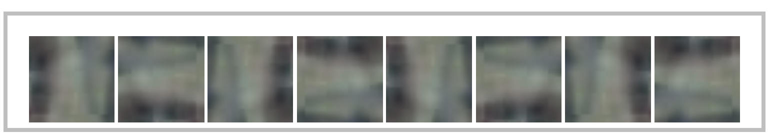

Figure 1: One Example of the Background Class Containing: Original, three rotations, four flips.

For this task, we introduce an extension to the SVHN data set, by adding an additional class for the purposes of classifying background. This 11^{th} class consists of an original 12,500 background 32x32 patches hand extracted from the SVHN format 1 train data set. These patches were then augmented with a series of flips and rotations become 100,000 samples. This flips and rotations are non-destructive as background by nature has no non-mirror-able features. See Figure 1 four for an example of this data set.

4.3 Model

We make an initial prediction that background will be classified noisily. Given the network that was not trained on background, we should expect a random result given a unique background input image. It follows that we would assume that there is indeed no need for the Background extension.

Using the Neural Network without and with the Background extension, we devise and preform the following algorithm, given an image:

1) Build prediction circles:
    a. Modify the image to add 16 padding to the border.
    b. Perform a 32x32 sliding window crop on the entire image.
    c. Classify the output of each image using the Neural Network, building a classification matrix with the prediction as the center pixel.
    d. Search through the classification matrix, and at pixel attempt find the maximum radius such that each pixel in the radius is of the same class as the center pixel.
2) Find digit locations:
    a. Determine the circle with the largest radius, and consider this and any circle with radius up to 15% smaller as digit detections.
    b. For all surviving circles, output the prediction centers and classification.

4.4 Results

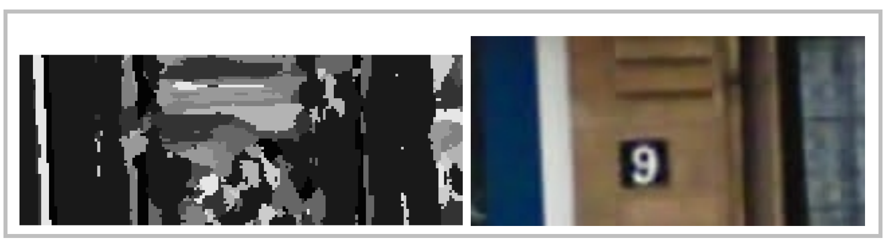

Figure 2: Classification Matrix Without Background Extension Followed by the Image Being Detected.

Our initial prediction that the Network would classify background nosily was false. Looking at Figure 2, we can see that the background was classified consistently in groupings. This is because the network is picking up on features in these patches and finding the class that has the highest output. The digit 9 was found, (the brightest patch), but is obscured by it's surrounding detections (the darkest colouring is class 0, and the lightest is class 9).

By adding an 11^{th} class we can see that the network does a fairly good job of determining background from digits, and nearly the entire image is considered background (white – class 11), and the digit is far more clear, and is the object with the greatest radius. It is obvious that an extra class for background is substantial in increasing our ability to distinguish a digit from a non-digit.

Running this algorithm on image 1-167 of the test set, we achieved an accuracy of 32.39%, and when the bounding boxes in the image were smaller than 32x32, accuracy was 54.09%. The average box center location error was 10.57 pixels. These results imply that if an ideal image was given to this algorithm, it would detect 54.09% of the digits in that image.

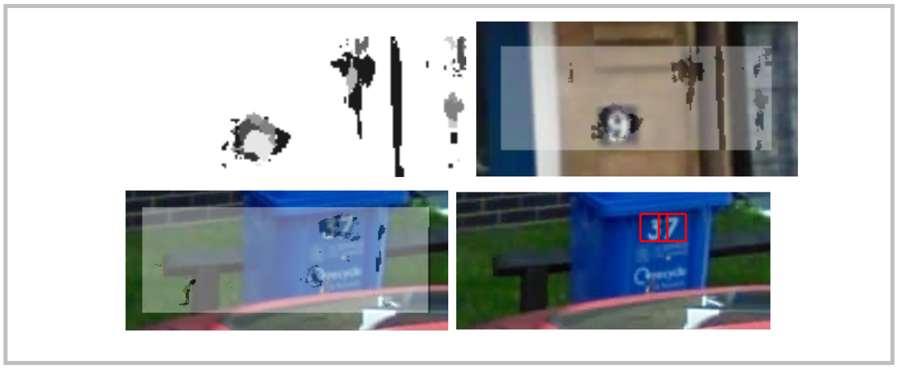

Figure 3: Two examples of Classification Matrix with Background Extension Overlaid

The low accuracy can be attributed to the lack of scaling method in our algorithm. A pyramid-scheme scaling could be applied to the input, to capture digits beyond 32x32 and to reduce the 10.57 average center pixel error. The algorithm is also not robust to tightly packed identically classed digits. Two consecutive and tightly packed 1's may be detected as a single centered 1.

A further improvement would be to take the soft approach to this problem and treat the regions as probabilities, allowing for Gaussian shapes to appear, where their mean would be the digit center.

5 Conclusion

Utilizing recent success, we were able to produce respectable results, but clearly there is much work to be done to catch up to the state of the art in classification and detection of digits in the wild.

References

[1] Netzer, Y. & Wang, T. & Coates, A. & Bissacco, A. & Wu, Bo. & Ng, A.Y. (2014) Reading Digits in Natural Images with Unsupervised Feature Learning. Computer Vision - ECCV 2014: 13th Part IV.

[2] Sermanet, P. & Chintala, S. & LeCun, Y. (2012) *Convolutional Natural Networks Applied to House Number Digit Classification*. Tsukuba: IEEE.

[3] Lee, C.Y. & Xie, S. & Gallagher, P. & Zhang, Z. & Tu, Z. (2105) Deeply-Supervised Nets. *CA*: Artificial Intelligence and Statistics Conference (AISTATS).

[4] Goodfellow, I.J. & Bulatov, Y. & Ibarz, J. & Arnoud, S. & Shet, V. (2013) Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks. arxiv.org.3