**Capstone project**

**Machine Learning Engineer Nanodegree**

**Sergey Volchkov**

**Project overview**

In recent years, digital applications such as Google Maps have to a considerable extent replaced paper maps and atlases as navigational aids. The success of these applications depends heavily on the accuracy of the maps used in them which determines the ease with which the user can find the desired street address. Google has been using cameras mounted on cars to collect street views in many cities of the world. These images can be processed automatically to add street numbers to maps and thus improve their accuracy while at the same time reducing the involvement of human operators to perform this rather tedious tasks.

The technology of automatic digit recognition has a wide potential and can be applied in other areas such as automatic license plate recognition or reading hand-written postcodes off letters and parcels.

**Problem statement**

The difficulty of detecting house numbers in street view images lies in the fact that if images are taken automatically, it is likely that only a fraction of the images will contain house numbers and, in the images that do contain them, the house numbers will not be positioned near the centre of the images. It is also likely that the house number will occupy only a small part of the image and there can be other objects in the image making detection of the relevant information more difficult. Further, a wide variety of different fonts, font sizes and colours is used in house numbers as it’s not regulated by law and they can be placed on different parts of the building, on a fence, a letterbox, etc. (which makes this task different from the task of automatic license plate recognition). Finally, images can be taken in various lighting conditions while the numbers can be made of highly reflective materials (e.g. polished metal).

House numbers often consist of short sequences of digits – one or two-digit sequences are quite common. It contrasts the task at hand with the one of detecting text in natural scenes as text strings represent repeated patterns of characters which are in most cases easier to extract from the surrounding background.

**Proposed strategy**

It is suggested to attempt solving the task of detecting house numbers in street view images by using deep convolutional neural networks (CNN). It has been found that CNNs achieve higher levels of accuracy than other machine learning methods in solving this and similar tasks[[1]](#footnote-1). Convolutional networks are a type of a neural network that use the operation of convolution in place of matrix multiplication in at least one of its layers to extract features from data that comes in the form a grid e.g. a 2D grid in the case of images[[2]](#footnote-2).

The proposed architecture consists of two CNNs to produce predictions. The first network – Localization Network - will determine the location – i.e. the bounding box – of a sequence of digits in an image. If no digits are detected in an image, it is expected to determine that as well. The second network – Classification Network - will use images obtained by cropping out the bounding boxes of the digit sequences produced by the first network to predict what those digits are. This network is only capable of identifying the length of the sequence and the digits in the range of 0 to 9. Those images that were found not to contain digits by the first network, will not be fed into this network.

**Metrics**

The success rate of the solution is to be measured based on the percentage of the correct predictions produced by the combination of the two networks. Partially correct answers – i.e. where only some of the digits have been identified correctly – are to be classified as wrong.

**Data exploration**

For the purpose of training the networks, the Street View House Numbers (SVHN) dataset will be used[[3]](#footnote-3). The dataset was produced from real-world images (Google Street View images). There are 33402 images in the training dataset and 13068 images in the test dataset. Images have different sizes and the areas occupied by the house number digit sequences within the images are quite different:



13064 13041 13035 12577

**Fig 1. Examples of unprocessed images from SVHN dataset**

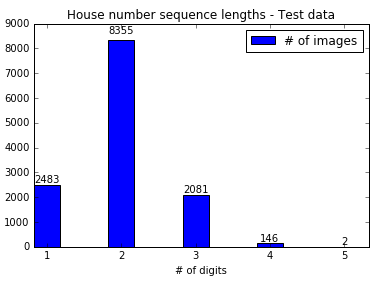
In some cases, the digit sequences occupy less than 5% of the image. We can also see that different fonts and font colours are used and that the sequences are not always placed strictly horizontally but are sometimes inclined or placed diagonally (at times also vertically). The digit sequences are 1 to 5 digits in length. Some of the images have a rather low resolution so that the digits are rather blurry and can hardly be discerned by the human eye.

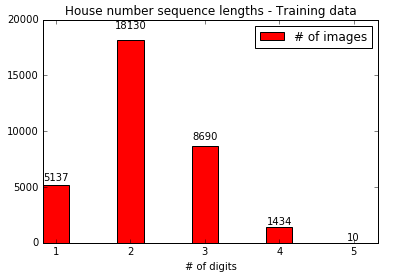
The dataset also contains labels for the images as well as the coordinates of the bounding boxes for individual digits within the sequences.

In addition to the above dataset, we will also need images not containing any digits. To produce such images, we will use the CIFAR 10 dataset[[4]](#footnote-4). It is a dataset containing 32x32 images of objects falling into ten categories none of which includes digits or text. The percentage of CIFAR 10 images used in both the train and test dataset was about 10% of the total number of images in the respective dataset.

Finally, for additional testing, we will be using real-life images taken by a mobile phone camera.

**Exploratory visualisation**





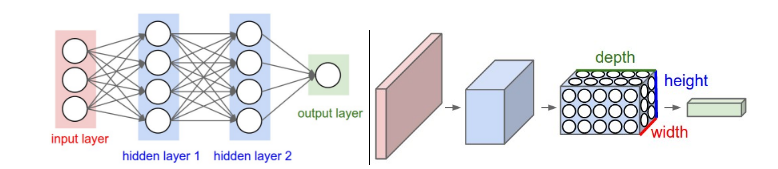
**Fig 2. Number of house numbers with different sequence lengths in the SVHN train/ test datasets**

As can be seen from the above charts, the majority of house numbers in the SVHN dataset represent two-digit sequences with less than 5% of house numbers having lengths of 4 digits or more.

**Algorithms and techniques**

The Localization and the Classification networks that the proposed solution consists of will employ slightly different algorithms to produce predictions. The Localization network will need to be able to solve a regression problem to predict a bounding box of the digit sequence, while the Classification network will, as the name implies, need to solve a classification problem to predict which of the 11 digit classes a particular digit most probably belongs to (digits 0 to 9 or no digit).

Both networks used in my solution are convolutional neural networks (CNN). CNNs are based on principles similar to normal neural networks where the network receives an input, performs a matrix operation on it (dot product) optionally followed by a non-linearity operation and produces an output. A large-scale view of CNN architecture is shown below:



**Fig 3. Convolutional neural network architecture[[5]](#footnote-5)**

A CNN will typically consist of a number of different types of layers including

* **Input layer** – holds the raw pixel values of the image
* **Convolution layer** – consists of a number of neurons each of which is connected to a small region of the preceding layer
* **Non-linear layer** – a layer that will apply a non-linear activation function such as *max(0,x)*
* **Pooling layer** - performs a pooling operation that averages the values of the neurons in the preceding layer and results in a decrease in the volume of activations
* **Fully-connected layer** – has each neuron connected to all the neurons in the preceding layer. A fully-connected layer will produce the output of the network in form of either a regression value or class probabilities for a classification task.

Layers can be stacked on top of each other to produce a sequential structure or form several branches using a graph-like architecture. To solve the problem at hand, I used multiple-output CNN’s. The Regression network outputs 4 values corresponding to the coordinates of the centroid of the house number within the image as well as its width and height, while the Classification network outputs 5 numbers corresponding to the digit class with the highest probability in each position of the house number.

To achieve better prediction results, a number of regularization techniques can be used. ***Dropout*** is one of such techniques that has been found to be effective for preventing overfitting[[6]](#footnote-6). The dropout rate defines the probability with which a particular neuron is active.

**Benchmark**

The benchmark accuracy achieved by human operators in performing this task is 98%[[7]](#footnote-7).

**Data pre-processing**

The images in the SVHN dataset have different pixel sizes and have on average 5:2 width-to-height ratio. We will firstly need to resize them so that all images have the same size as it is one of the requirements of using CNN’s that inputs need to be of equal size. We will also change the width-to-height ratio to 3:2. The reason for this change is that this is the standard width-to-height ratio for images taken by most mobile phone cameras.

I have tried using both 72x48 and 96x64 images as inputs into the first network, with 96x64 images producing better accuracy as some digits would be illegible after resizing the original image to the 72x48 format.

For the training purposes, the bounding boxes of the individual digits were combined into a single bounding box for the entire digit sequence. The area around the bounding box was then cropped out with a random number of surrounding pixels retained to the left, right, top and bottom of the bounding box.

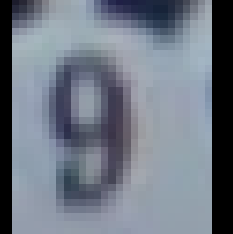


**Fig 4. Transformation applied to original SVHN images to create inputs for the Regression Network to produce two different training images**

The idea behind applying this transformation to the original images was twofold. First, the network needs to learn to detect house numbers where they are not positioned near the centre of the image. Second, it allows us to produce additional training images by simply shifting the position of the house number within the image.

Four labels were used for each image in the training and the testing phases, including the bounding box centroid x and y coordinates and the bounding box width and height. Where no digits were present in the image, the values of all the four labels were set to zero.

During the training stage, the Classification Network used 32x32 images that were produced by cropping out the bounding boxes using the coordinates provided with the SVHN dataset (the bounding boxes for individual digits were again combined to produce a single bounding box for the entire sequence. Where the bounding boxes were not square, padding was used near the borders of the images.



**Fig 5. Images used for training and validation of the Classification Network**

At the testing stage, the Classification network was using images produced by cropping out the bounding boxes predicted by the Localization Network.

The labels used for training and validation of the Classification Network were modified to produce 5 different labels corresponding to a digit in each position of the digit sequence. Where there were fewer than 5 digits in a house number, the missing digits were represented by 0’s, while the actual zeroes were replaced with the number 10. So, for example number 892 would be encoded as 8,9,2,0,0 and number 10 as 1,10,0,0,0. No separate label was used to encode the actual length of the house number sequence.

**Implementation**

I have used the baseline architecture for both the Localization and the Classification networks as shown on the right. This network architecture was suggested for an image classification task using the CIFAR 10 dataset[[8]](#footnote-8). After 400 epochs of training, the prediction accuracy of the Classification Network using pre-calculated bounding boxes was 84.32%.

**Fig. 6 Baseline network architecture**

I used the mean squared error as the loss function for the Localization Network as I felt that it was easy to implement and interpret. For example, if the total loss of the network was 100, it could be interpreted as all the four predicted values being about 5 pixels off the true values (52 \* 4)

In the cases where the width or the height of the presumed digit sequence produced by the Localization Network was below a certain threshold (I used the value of 3 in my experiments), the image was considered to not contain digits and was not passed on to the Classification Network for further processing.

**Refinement**

At the initial stages of testing, the most significant improvements came from adjustments to the data pre-processing methodology. In particular, it was found that compressing the images too aggressively with the goal of decreasing the training time had a deleterious effect on the prediction accuracy.

While testing the networks, I found that a deeper architecture including two additional convolution layers with 160 3x3 filters each allowed me to improve the prediction accuracy of the Classification Network by 3.5%, and after I added two more layers to the Localization Network, the total improvement was about 5%. I then added a 2-pixel padding to the bounding boxes produced by the Localization Network which improved the accuracy by another 3%.

Adding further convolution layers to the Network produced no improvement, and increasing the padding around the bounding boxes had a negative impact on the prediction accuracy.

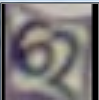
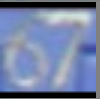
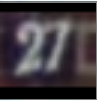
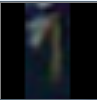
**Model evaluation and validation**

When using pre-calculated bounding boxes, the Classification Network achieves the prediction accuracy of 87.92%. Using the combination of the two networks, the total prediction accuracy was 63.99%.

**Fig. 7. Examples of images correctly predicted by the Classification Network in the standalone mode**

The examples above appear to be relatively easy as the numbers are clearly seen against solid single-colour background, are horizontally aligned and there are no extraneous objects in the images. The black border at the edges of the images does not appear to present a problem. Incorrectly interpreted images subjectively appear to be harder although in most cases still readable by the human eye:

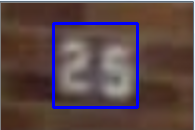
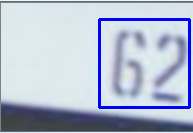


(a)271 vs 27 (b)87 vs 67 (c)67 vs 62 (d)540 vs 1840 (e)18 vs 17(???) (f) 7 vs 4

**Fig.8 Examples of incorrectly predicted images**

We can speculate that the reasons for the incorrect predictions were (a) part of the brick wall was interpreted as part of the digit sequence (b and f) reflective materials used for the numbers, low contrast between the digits and the background (c) unusual alignment and shape of the digits (d) image appears to be distorted and possibly slightly cut off on the left side (e) incorrect ground-truth data. It is possible that the network could benefit from an additional step of applying random rotations to the images at the pre-processing stage.





13 25 62 1991

**Fig.9 Examples of correctly predicted images using complete solution**

A visual inspection of the examples that were predicted correctly shows that in these images the house numbers occupy a relatively large percentage of the image area and the area within the bounding box is almost fully occupied by the house number as well. We can therefore expect that the images produced by cropping out the pixels within the bounding box that are fed into the Classification Network will be very similar to the pre-cropped images that the Classification Network was able to successfully recognise during the standalone testing phase.



D.

Complete solution: 11

Pre-cropped: 16

Correct: 16

C.

Complete solution: 17

Pre-cropped: 1

Correct: 1

B.

Complete solution: 13

Pre-cropped: 48

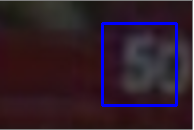
Correct: 48

A.

Complete solution: 61

Pre-cropped: 93

Correct: 96



H.

Complete solution: 36

Pre-cropped: 100

Correct: 108

G.

Complete solution: 32

Pre-cropped: 36

Correct: 36

F.

Complete solution: 5

Pre-cropped: 58

Correct: 50

E.

Complete solution: 67

Pre-cropped: 96

Correct: 96

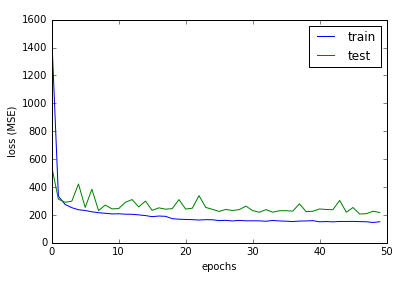
**Fig. 10. Incorrectly predicted images using the complete solution involving two networks**

The incorrectly predicted images present a somewhat mixed bag but we can see that the house number was completely missed in only one example (B), although there were quite a few partial misses (in images D-G). We can also observe that a considerable fraction of incorrectly predicted images involved cases where the house number occupied only a small area within the image (images A-C, E, G and H). On some of these images, the numbers have a clear contrast against the background and appear to present no problem when fed to the Classification Network as pre-cropped images, i.e. where the bounding box is already known (B, C, E and G). In two cases (D and F), the numbers are situated close to the image borders so it is possible that some information gets lost near the image borders which can be possibly prevented through using zero padding in the convolution layers.



**Fig. 11. Performance on “bogus” images from the CIFAR 10 dataset**

The “bogus” images from CIFAR 10 were correctly predicted to not contain any digit sequences in all but a handful of examples.



**Fig. 12. Localization Network - Training and validation loss**

One particular problem that I faced when training both networks but in particular the Localization Network – and have not been quite able to overcome – was that learning seemed to stop somewhat early. The example below shows training and validation loss results for one of the configurations of the Localization Network involving eight convolution layers with the initial two layers expanded to have 48 filters each. It can be seen from the graph that, while the training loss continues to decrease, the validation loss plateaus after about 30 epochs of training. I tried using different optimizers and learning rate decay schedules but the overall pattern remained more or less the same.

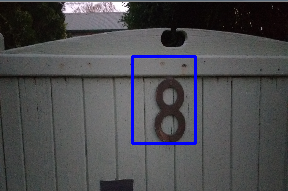
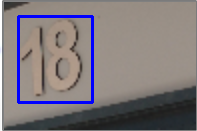
**Justification**

The final complete solution at the moment falls considerably short of the 98% benchmark achieved by human operators. At the same time, there are some promising results on some categories of images, primarily those where house numbers occupy a larger percentage of the total image area and where images have higher resolution. It should be noted that there were some practical considerations that had to be taken into account when testing different network configurations. I ran most of my experiments on my home machine with 2Gb of GPU memory so some of the more ambitious configurations were causing out-of-memory errors.

One advantage of the solution that was used for the Localization Network is that it is simple and easier to implement than the method described in some scholarly articles that involves multiple crops taken from the same image[[9]](#footnote-9).

**Free-form visualisation**

As a final test, I took a number of photos with my mobile camera in a local area to be used as test images. These photos contained digit sequences that were not necessarily house numbers – some of them contained phone numbers used on advertisements, bus stop numbers, speed limits on road signs, etc. Some photos didn’t contain any digits at all. Unfortunately, the results of this test were somewhat unsatisfactory as only slightly more than 20% of the sequences were recognised correctly. In some cases, it could probably be explained by the digit sequences occupying a very small area of the image. Since there was no pre-cropping used on these images, once they were compressed to 96x64, these sequences probably became completely unreadable. There was also a disturbingly high number of false predictions where an image did not contain any digits at all.



C. Correct

Predicted result: 18

B. Incorrect

Predicted result: 17

1. Correct

Predicted result: 8



E. Correct

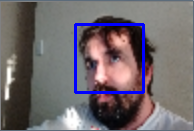
Predicted result: 58

D. Correct

Predicted result: 15

F. Correct

Predicted result: 40



I. Incorrect

Predicted result: 13

H. Correct

Predicted result: 20

G. Incorrect

Predicted result: 96

**Fig. 13. Testing using random images taken by a mobile camera**

**Reflection**

The proposed solution consists of the following steps:

* **Data collection and labelling.** The SVHN dataset was a good starting point for me, however I had to add images from another dataset as I also wanted to train the network on images that did not contain images.
* **Data pre-processing.** Various techniques can be used here e.g. images can be pre-cropped to contain fewer extraneous objects. I subtracted the mean of the dataset from each image and divided the values by the standard deviation of the dataset. It is also possible to produce additional training sample by cutting and stitching existing images.
* **Localizing digits in images and filtering out irrelevant images** (i.e. those containing no digits). I found this step to be the most challenging and am not sure that I used the best approach to solving this task. Failure to correctly locate digits in this step will produce completely wrong classification results.
* **Recognizing digit sequences.** Unlike solutions involving recognition of individual digits, only those images where all digits were recognized correctly counted as correctly classified.
* **Adjusting parameters.** Training deep networks can be a very time-consuming process unless one has got access to an extremely powerful GPU. Therefore some parameter settings and network configurations could not be tested because of time and resource constraints.
* **Testing.** We are surrounded by digit sequences in our everyday life so additional material for testing is easy to come by. Using images with different characteristics (e.g. with different resolutions, relative sizes of digit sequences, taken in varied lighting conditions etc.) can help identify weaknesses in the solution.

In addition to the challenges listed above, one additional aspect of deep networks that I found interesting and challenging at the same time is that this is a newer field compared to the rest of machine learning so ready-made solutions are harder to come by. This made me feel more of an explorer.

**Improvement**

There is vast potential for the improvement of the proposed solution.

* Some real-life house numbers contain alpha and punctuation characters such as “23A” or “14/77”. The Classification Network needs to be able to handle these as well.
* Accuracy can potentially be improved through modifying the Classification Network to handle the cases where there are no digits present in the image or the digits do not occupy nearly all area of the image. Currently it assumes that all input images it receives contain digits and those digits occupy nearly the entire image.
* Another possible way of improving the accuracy of the complete solution might be using less aggressive pooling within the Localization Network so that smaller features can be more easily picked up in cases where the digit sequence occupies a very small percentage of the image (less than 5%). The size of the input images used for this network might also need to be increased.
* The solution can be transformed into a fully-functional mobile app. It could then be used together with the geo-spatial information to add missing street addresses on the digital maps.

1. Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, Vinay Shet Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks p.2 <https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42241.pdf> [↑](#footnote-ref-1)
2. *I. Goodfellow, Y. Bengio, A. Courville* Deep Learning p. 330. Downloaded from <https://github.com/HFTrader/DeepLearningBook> on 17-12-2016. [↑](#footnote-ref-2)
3. Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011* (the dataset is available from <http://ufldl.stanford.edu/housenumbers/> ) [↑](#footnote-ref-3)
4. Alex Krizhevsky Learning Multiple Layers of Features from Tiny Images 2009. Retrieved from <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf> on 17-12-2016. [↑](#footnote-ref-4)
5. Stanford University CS231n Convolutional Neural Networks for Visual Recognition. Retrieved from <http://cs231n.github.io/convolutional-networks/> on 18-12-2016. [↑](#footnote-ref-5)
6. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Retrieved from <http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf> on 18-12-2016. [↑](#footnote-ref-6)
7. Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, Vinay Shet Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks p.5 <https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42241.pdf> [↑](#footnote-ref-7)
8. Jason Brownlee Deep Learning With Python 2016 p.156 [↑](#footnote-ref-8)
9. Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable Object Detection using Deep Neural Networks. Retrieved from <http://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Erhan_Scalable_Object_Detection_2014_CVPR_paper.pdf> on 20-12-2016. [↑](#footnote-ref-9)