University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Computer Science	Spring semester, 2015 Open
Writer: Joacim André Jakobsen	…… (Writer's signature)
Faculty supervisor: Tom Ryen External supervisor(s): Andreas Waal	
Thesis title: Optical Character Recognition on Electrical Specification Plates	
Credits (ECTS): 30	
Key words: Neural network, optical character recognition, electrical specification plates, classification, MATLAB	Pages: 19 + enclosure: 0 Stavanger, 15-6 / 2015 Date/year

Abstract

This paper looks into the feasibility of using neural networks to classify characters on electrical specification plates (ESP). This thesis is given by Verico AS (Verico). Verico performs large scale Asset Documentation, where photo documentation is one of their main tools. As such, they have large amounts of ESP imagery. Collecting data from the images is done manually, which is both time consuming and tedious work. This thesis seeks to further develop and utilize previous work done for Verico, such as background segmentation and vertical histogram analysis. The scope of the thesis is looking at the feasibility of using neural networks as a classifier for digits. MATLAB's Neural Network Toolbox is used to train and classify data. The neural network is trained on 240318 images from the Street View House Numbers (SVHN) Dataset, and then tested on two different datasets. The first is on 26032 images from the SVHN test dataset, where the neural net achieved an overall accuracy of 84.1%. Through confidence thresholding 98% accuracy is reached at 52.8% coverage. The other dataset consists of 600 images gathered from several classes of ESP. The neural net achieves 94.1% overall accuracy, and with confidence thresholding 98% accuracy is reached at 85.3% coverage.

Contents

I. Introduction **1**

A. Thesis Background 1
B. Principal Company 1
C. Motivation 1

II. Problem **2**

A. Background and Previous Work 2
B. Outline 3

III. Pre- Processing **4**

A. Thresholding 4
B. Character Segmentation 5
C. Resizing 7

IV. Neural Network **8**

A. General 8
B. Training 9
C. Datasets 9

V. Results **13**

A. Network Performance 13
B. Classification Results 15
C. Improving Accuracy 16

VI. Conclusion **18**

VII. References **19**

I. Introduction

A. Thesis Background

This thesis means to explore the viability of using neural networks to classify characters on ESP. A solution would entail a high classification accuracy with an acceptably coverage. The work performed is done during the spring of 2015. A pilot project was performed the autumn of 2014 for a 10 ECTS weighted class.

B. Principal Company

Verico is a Stavanger based company that primarily work with Asset Data Management and Software Development. Their main customers are Transmission System Operators like Statnett SF (Norway) and TenneT BV (The Netherlands). They also perform Asset Documentation, which for this paper pertains to photo documentation.

C. Motivation

The motivation for this paper comes from the attempt to automate parts of the photo documentation process. As it stands, data from photos are entered manually. This is a time consuming and tedious job, which has great potential for improvement through automation, considering the amount of photos ranges in the tens of thousands.

II. Problem

A. Background and Previous Work

Recognizing characters from photos of ESP is a hard problem. However, previous work for Verico has simplified the domain. They already have a working solution to crop and translate the ESP to a centered image. Further, they have solved cutting out the critical variables from the image. The thesis is mostly built upon a previous master's thesis [1], which simplified the problem through a solution for character segmentation. Remaining is a narrower domain, where instead of a natural photo you have the image of a single character. Fig. 1 [2] shows the full process of photo documentation; where all but the character recognition stage is solved.

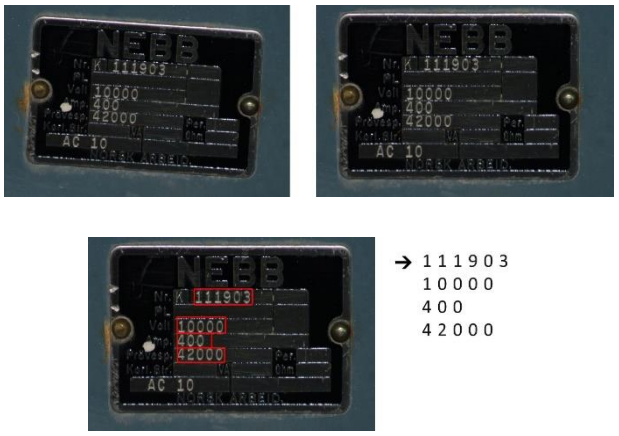

Fig. 1. The final intended full process of photo documentation.

Using neural networks for classification of digits has great potential, and good results for both handwritten characters [3], and more recently, recognizing address numbers from the SVHN Dataset. This is the context examined for a solution, especially considering Google's findings in in the 2014 paper "Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks" [4]. They showed a 98% accuracy with a coverage 89% on the SVHN dataset. Accuracy is defined as the recognition rate of digits, and coverage is the percentage of the total dataset used.

B. Outline

The outline of the process is detailed in Fig. 2. Different minutiae of the steps in the process will be discussed in detail, as well as alternative methods, both tested and untested.

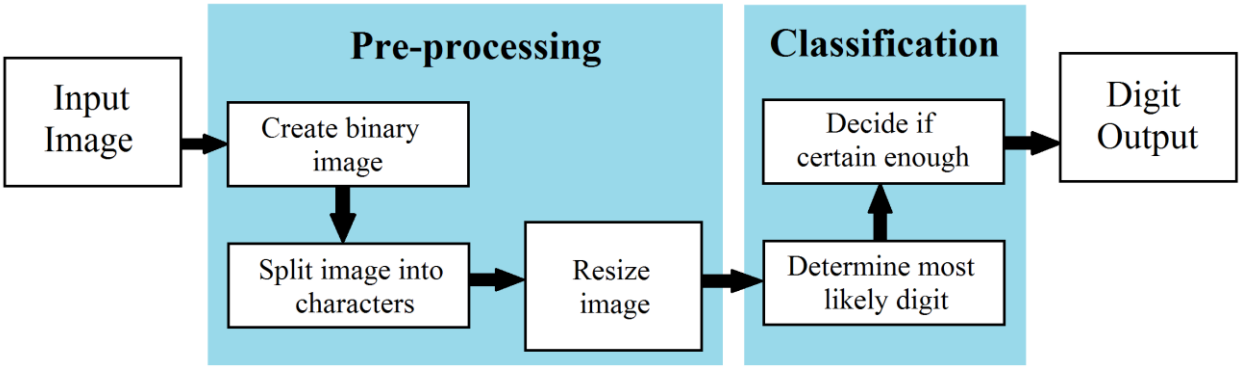

Fig. 2. Outline of OCR process.

III. Pre-processing

A. Thresholding

To create a useable input vector containing the characteristics of a digit, the specific colors of an image is irrelevant. Therefore, an image is first converted to grayscale, removing all color and hue while retaining the luminosity. MATLAB converts to grayscale values using a weighted sum of the R, G and B components as such: 0.2989 * R + 0.5870 * G + 0.1140 * B [5]. This weighting may influence the final resulting accuracy and coverage, however alternative weights was not been tested.

For the method of character segmentation used, converting to a binary image is necessary. For this process, Otsu's method [6] is used. Otsu's method searches for the threshold that minimizes the variance within the classes, i.e. black and white.

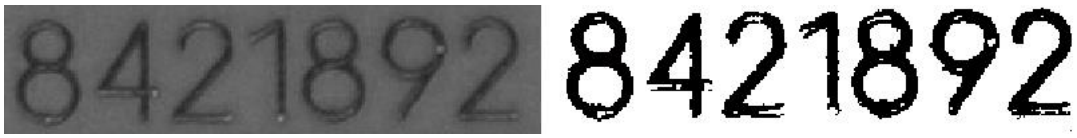

Fig. 3. Thresholding using Otsu's method.

Not all images converted to binary have as good results as shown in Fig. 3. The results of different optimizations more specific to this domain are shown further in Fig. 4, 5, 6.

B. Character Segmentation

Segmenting characters in the image is necessary to reduce the number of possible classes to 10. To segment characters several methods are available, for this paper only vertical histograms are used. Vertical histogram segmentation counts the number of white pixels in each column of the picture. This method is modified some to account for characters that may be connected, i.e. there is no actual separating pixel columns between them [1].

Another caveat to this type of segmentation is that all images need to have the same background and character color (black background and white characters). To ensure this property the ratio of white and black pixels is used. If the ratio exceeds a threshold, in this case 0.64 [1], the image is inverted.

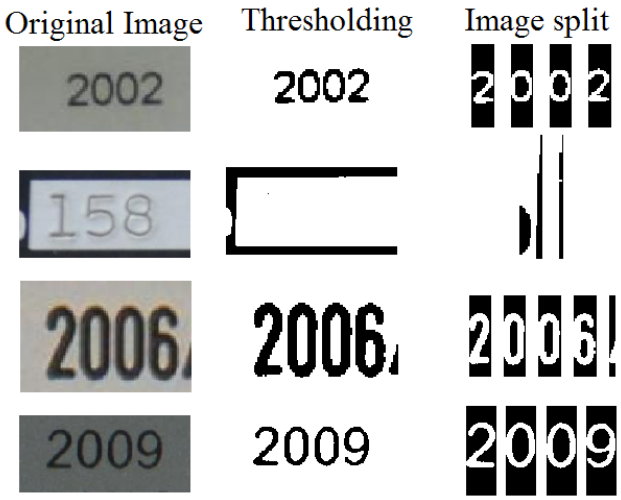

Fig. 4. Results of thresholding using Otsu's method and vertical histogram to split characters.

As seen from Fig. 4 sometimes thresholding removes all crucial information from the image. Some simple optimizations for this specific image domain were tested and the results are shown in Fig. 5 and 6.

The location intensity method is to value pixel values more as they get closer to the center of the image, where an arbitrarily sized area will give 'full score'. These new values will then be utilized to find the threshold between black and white with Otsu's method.

The other approach tested is to generate the threshold with Otsu's method using only a horizontal slice, with arbitrary height, of the image. Considering these minor improvements, this affects the choice in datasets, which is discussed further in section IV.C.

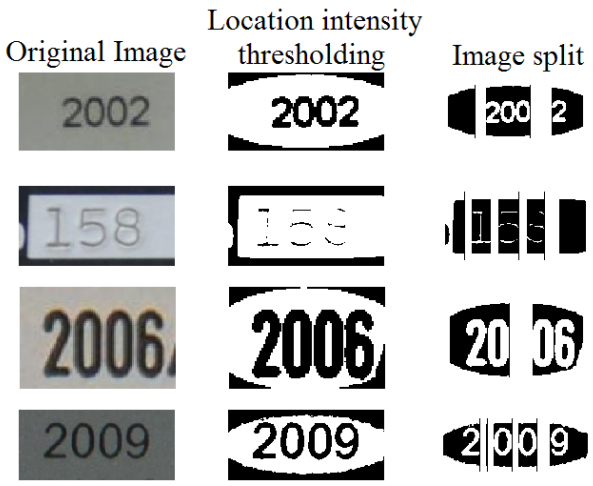

Fig. 5. Location intensity method and the resulting image splits (separated by white pixel columns).

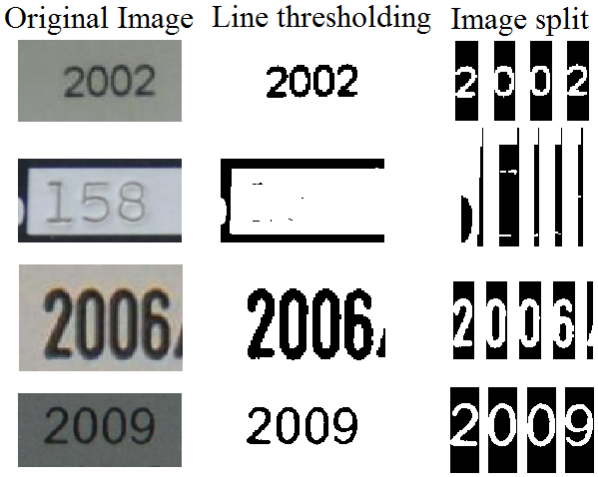

Fig. 6. Line thresholding method and the resulting image splits (separated by white pixel columns).

Another problem with image splitting is that sometimes too many splits occur. This is partially solved while generating datasets, considering a priori information of how many characters there are supposed to be. Some simple rules allows cutting away trailing splits or disqualifying the entire image after the situation. The ramifications of this is further discussed in section IV.C. The training dataset chosen does not give many usable images with thresholding, similar to some ESP class images. This results in utilizing only the coordinates of the splits to cut out the characters from the original image as seen in Fig. 7.

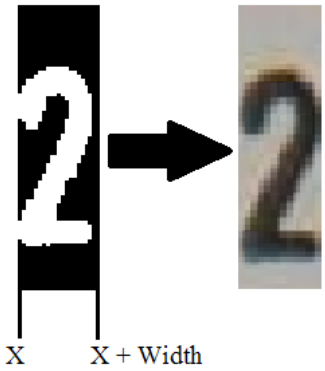

Fig. 7. Coordinates from split used to generate split from original image.

Other thresholding techniques such as Adaptive Document Image Binarization [19] were attempted, but showed no discernible improvement to Otsu's method.

C. Resizing

The length of an input vector to a neural network is required to be uniform. One way to achieve this is by resizing the image to arbitrary dimensions. For this paper, the dimensions chosen are 16 pixels wide and 22 high. This is mostly due to computational limitations.

Several different methods for resizing is possible. Those tested are nearest neighbor, bilinear, bicubic interpolation and box-shaped kernel [7]. The resulting overall accuracy of the ESP dataset was clearly best when using bilinear interpolation. The accuracy loss of the other methods compared to bilinear interpolation is detailed in Table 1.

Method	Linear	Bicubic	Box-shaped
Loss in accuracy	9.8%	3.6%	6.1%

Table 1. Accuracy loss of other scaling methods on ESP dataset.

Other methods of maintaining image 'correctness' when resizing to arbitrary dimension, e.g. adding padding of background color for outlier width to height ratios, are possible but were not tested.

IV. Neural Networks

A. General

Simply put, a neuron works by giving an input p, multiplying it with the weight w and adding the bias b, as seen in Fig. 8 [8]. This gives the output a, and depending on the transfer function f may help classify the input.

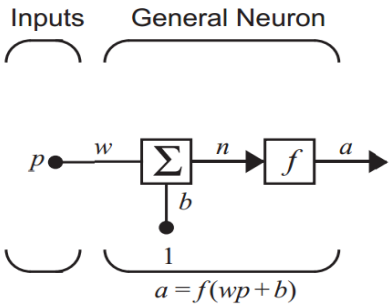

Fig. 8. Single input neuron.

Neural networks are, for ease of explanation, many such neurons connected together in layers. This way the network takes in an input vector p, multiplying it with the weight matrix W and adding the bias vector b, as seen in Fig. 9 [9].

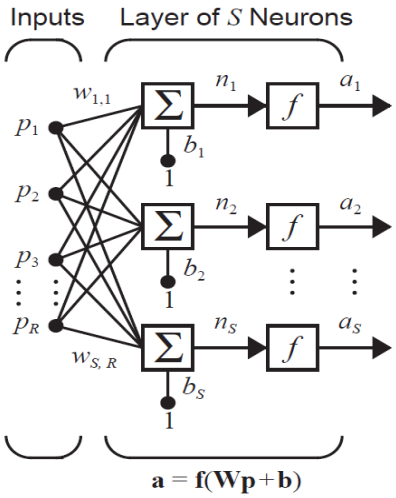

Fig. 9. Multi input neural network

The quintessential part of neural networks is that they may be trained from data. Training works by adjusting the weights and biases after, correctly or incorrectly, classifying inputs.

The neural networks trained and tested for this paper were all two layer feedforward backpropagation neural nets. Achieving the highest possible accuracy of this type of neural net is generally through four non-excluding ways: Increasing sample size, raising input quality through pre-processing, number of hidden neurons and input vector size.

Sample sizes are limited by available datasets and will be further discussed in section IV.C. Input quality has already been covered in section III. The number of hidden neurons is also limited by computational power and ties in with the input vector size. A 'rule of thumb' is to specify the number of hidden neurons between 70-90% of the input vector size [10]. However due to computational constraints, and no mentionable difference in performance for slightly larger sizes, 150 hidden neurons was chosen. This corresponds to 42% of the input vector size. Input vector size was arbitrarily set at 352 (16*22) as any larger was not possible due to lacking computational power.

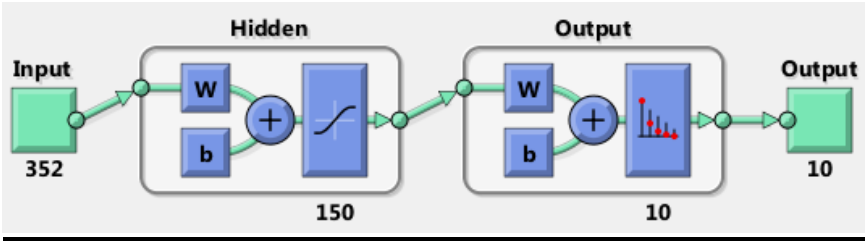

Fig. 10. Overview of the neural net used detailing input vector size, number of neurons in hidden layer and output layer.

B. Training

In MATLAB, input data is randomly divided into three sets: Training, validation and testing. When repeatedly fed the same data, a neural network will increasingly output correct classifications. However, overfitting may occur. This means the network will perform well on the training set, but not on generalized data. Therefore, when performance stops improving on the validation set for six training cycles (epochs), it will stop training. This is to ensure good generalization for new data. After that, performance is measured on the completely unused testing set.

The training algorithm used is scaled conjugate gradient backpropagation [11]. Weights are initially randomized, this along with the random division of data makes results vary. This leads to several different resulting neural nets, where generalized performance is not always indicated by training graphs. The neural network ultimately chosen performed worse during training than others, yet showed higher performance for the testing datasets. This suggests better generalization.

C. Datasets

Verico supplied over 23000 images. However, after several attempts at pruning to remove ineligible images, those too similar in background and foreground color or in unusable formats with trailing or leading letters, another sample set had to be found. This is mainly due to bad imagery 'poisoning the well'. The existence of such images was the motivation behind the efforts in section III.B. An example of such an image is found in Fig. 4, 5 and 6. Another motivation to utilize a different dataset for training is the sample size required.

A large enough sample size is important in training models with high accuracy [12]. As seen in Fig. 11 [13], the performance of differing learning classifiers improves as sample sizes increase towards ~100 000. Other untested means of increasing accuracy of a model, such as boosting, also requires greater sample sizes [14]. When deciding upon neural networks as a classifier, it is also important to note that neural networks should outperform k-means in more noisy cases [15].

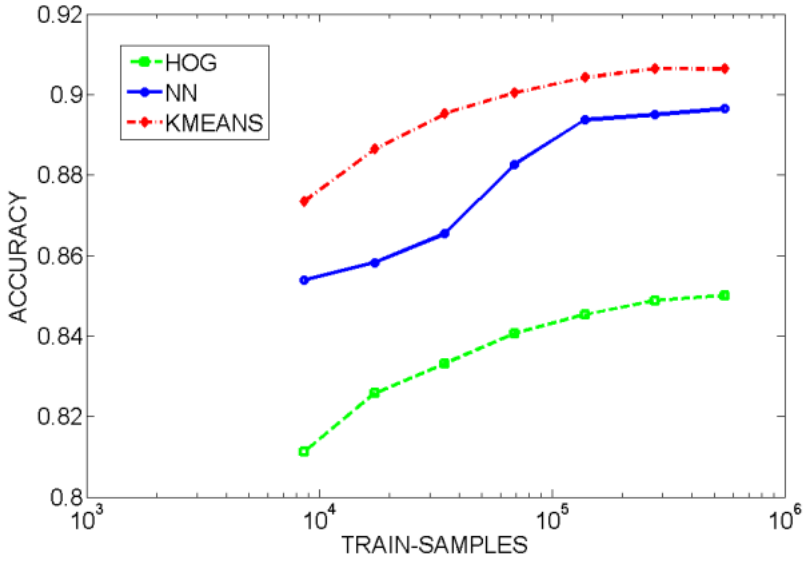

Fig. 11. Accuracy increase for different learning methods as training samples increase in magnitude.

Therefore, the choice to utilize the SVHN extra dataset was made. Pre-processing for SVHN images only consists of converting to grayscale and resizing. The training dataset consists of 240318 images. The distribution of data is shown in Table 2. This distribution is important because it will influence training, and may skew classification to some classes more than others.

Class	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Amount	40920	33570	27306	22967	24273	18947	20090	15937	15692	20616
Percentage	17.03%	13.97%	11.36%	9.56%	10.1%	7.88%	8.36%	6.63%	6.53%	8.58%

Table 2. Distribution of digits per class in the SVHN training dataset (class 10 represents the digit 0).

There are two sides to the argument about the distribution of the dataset. This is regarding the assumption of the a priori probability of classifying a digit. That if given an input image, the probability of simply guessing correctly is 10%. The other is considering if ESP imagery follow Benfords law. The law is an observation that in many naturally occurring collections that the leading digits are not uniformly distributed, but heavily skewed towards smaller digits [16]. Considering this distribution is of all trailing and leading digits, an argument may be made it follows the trend closely, as seen in Fig. 12 [17]. If this is the correct approach when considering real world data from ESP however, is up for discussion.

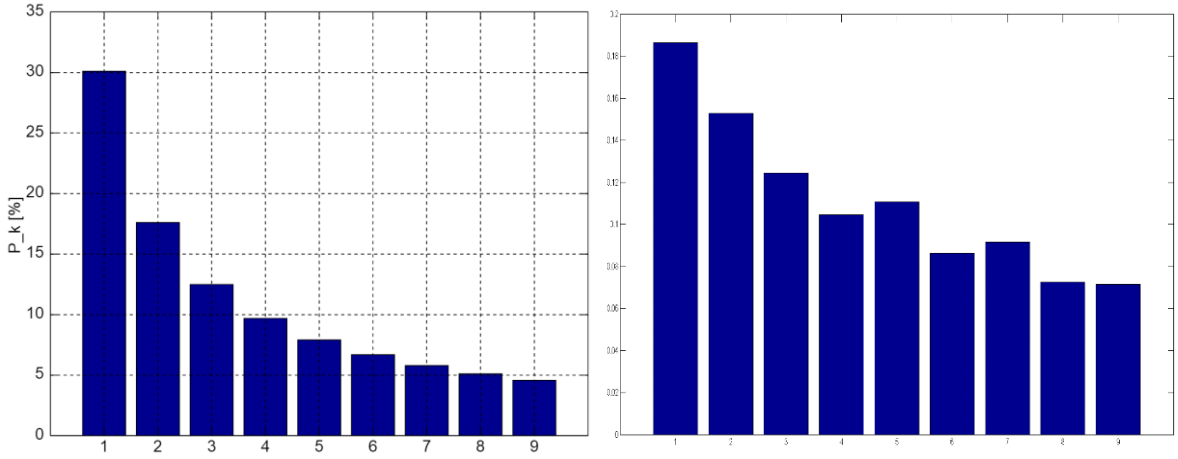

Fig. 12. Distribution of leading digits according to Benfords law (left), and SVHN training set percentage distribution of digits 1 to 9 (right).

The SVHN test dataset is used to measure generalized performance on similar data to the training set. This dataset consists of 26032 images. The distribution of this set is detailed in Fig. 13.

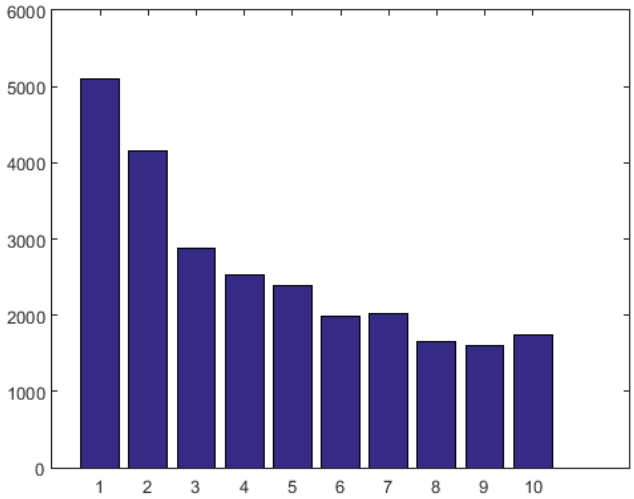

Fig. 13. Distribution of digits in SVHN test dataset (10 represents the digit 0).

This distribution follows closely to the SVHN training set. In Table 3, the distribution clearly shows its similarities compared to Table 2. Which impact this has on the results is undetermined, yet it should be a positive one.

Class	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Amount	5099	4149	2882	2523	2384	1977	2019	1660	1595	1744
Percentage	19.59%	15.94%	11.07%	9.69%	9.16%	7.59%	7.76%	6.38%	6.13%	6.70%

Table 3. Distribution of digits per class in the SVHN test dataset
(class 10 represents the digit 0).

The other test set consists of 600 images of ESP digits. This dataset is distributed uniformly with 60 images per class. These images are gathered from some of the easier cases of ESP from [1].

V. Results

A. Network Performance

The network was trained with a data split of 80% for training, 15% for validation and 5% for testing. Since other datasets were available for the evaluation of performance, the size of the original test dataset was set low. Fig. 13 shows the plot of cross entropy over epochs for each the training subsets.

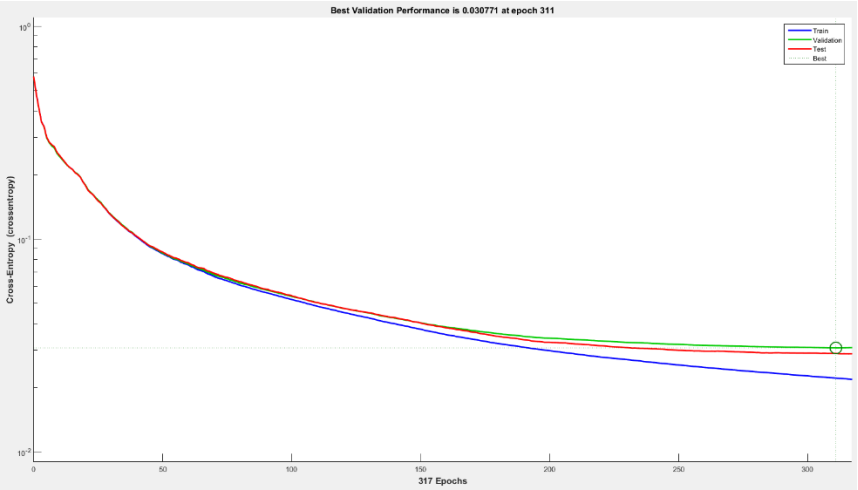

Fig. 14. Cross entropy during training of the neural network.

Cross entropy is an error measure used to monitor progress, here it determines when to stop training to reduce overfitting. The reasoning behind this early exit strategy was explained in section IV.C.

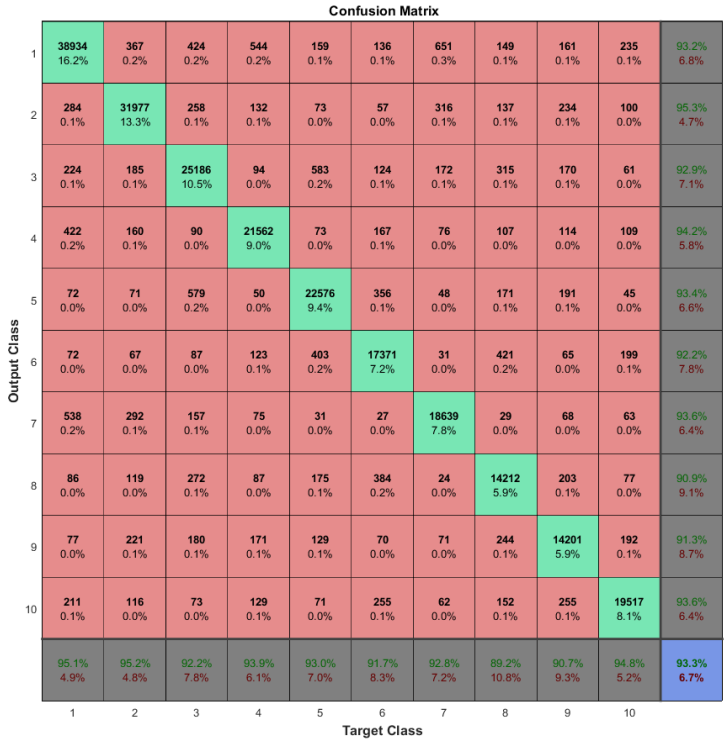

Fig. 15. Confusions matrix of training dataset after training.

Fig. 15 shows the confusion of the dataset used during training. Other neural networks tested had better performance during training than 6.7% error rate, which is on the high end of acceptable, but this neural net performed best on the other test sets. Fig. 15 clearly shows the network performing better on classes with a higher percentage of the data distribution, such as classes 1, 2 and 10 (class 10 represents the digit 0). This is partly because of the skew discussed in section IV.C. Fig. 16 shows this relationship a bit better, but it also tells that the data distribution is not the whole story.

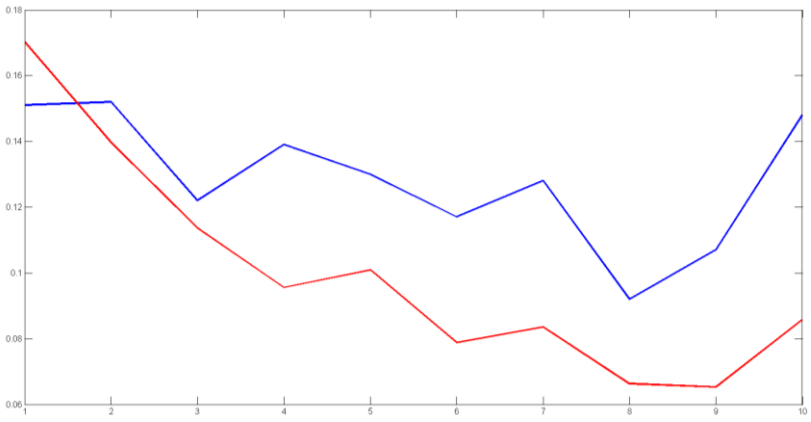

Fig. 16. The horizontal axis represents the classes and the vertical axis represents percentages. The red plot details the percentage distribution of the training dataset, also seen in Fig. 12. The blue plot is the accuracy of each class shifted by negative 0.8 (80%) in the vertical axis.

Another relationship between the classes and accuracy to consider is the similarity between the classes. Similar digits are hard to differentiate, such as 1 and 4, 1 and 7, 5 and 6 or 6 and 8. This is reflected in the error rates of for these specific classifications shown in Fig. 15. Table 4 shows the worst offenders of these relationships.

Class 1 misclassified as class 4.	Class 4 misclassified as class 1.	Class 1 misclassified as class 7.	Class 7 misclassified as class 1.	Class 5 misclassified as class 6.	Class 6 misclassified as class 5.	Class 6 misclassified as class 8.	Class 8 misclassified as class 6.
422	544	538	658	403	356	384	421

Table 4. Misclassifications between the most similar classes in the SVHN training dataset.

In all of these classes the misclassifications between those most similar to it in appearance represent the majority or errors. There may be some consideration to be had whether the distribution of data worsens this relationship, or even has a noticeable effect on the result.

B. Classification Results

The SVHN test dataset was classified with an accuracy of 84.1%. This dataset also shows similar misclassification patterns as the SVHN training dataset. Possible reasons for this is detailed in the section V.A. It is however less pronounced here because of the smaller sample size.

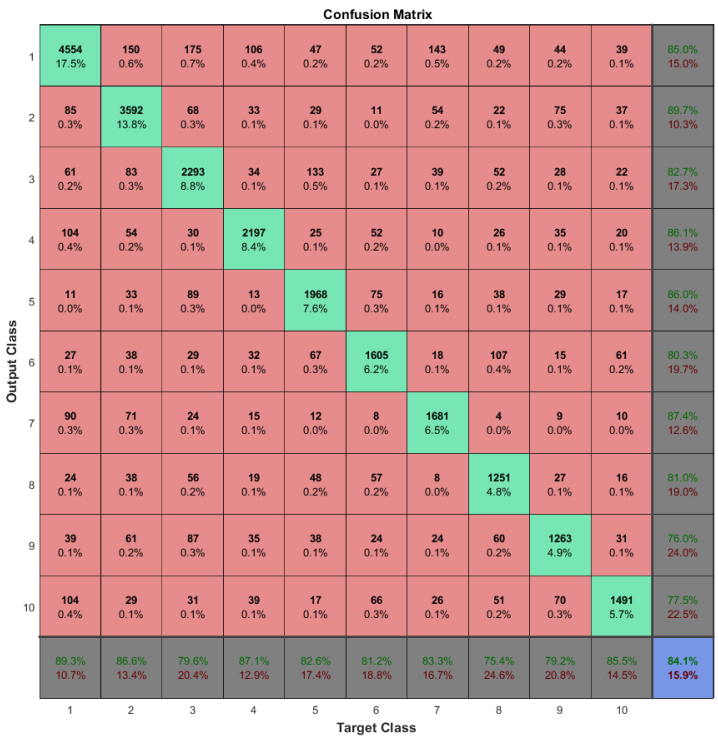

Fig. 17. Confusion matrix of SVHN test dataset.

The ESP dataset was classified with an accuracy of 94.2%. These same patterns of misclassifications may be seen here, however no conclusions regarding this may be made since the sample size is magnitudes smaller.

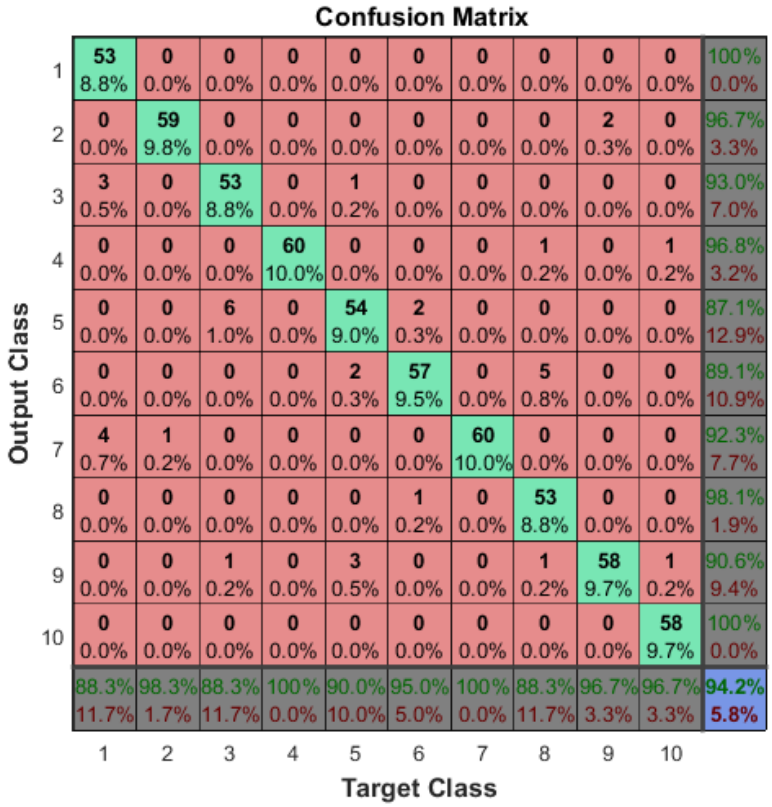

Fig. 18. Confusion matrix of ESP dataset.

C. Improving Accuracy

To improve the accuracy of a neural network it is possible to introduce a confidence threshold [18]. This is an output activator that either accepts or rejects a classification based on a threshold.

$$\text{Decide} \begin{cases} \text{If } Y > T, \text{Accept classification.} \\ \text{Else, Reject classification.} \end{cases},$$

where Y is the highest value of the output vector and T the threshold value.

This is a tool to improve the accuracy to an acceptable level. For this paper that is set at 98% accuracy, which was set as the rate of human performance in [13]. The tradeoff of this method is that you reduce the coverage, i.e. percentage of data classified.

The SVHN test dataset achieved 98% accuracy at 52.8% coverage. The confidence threshold is set at 97.87. Fig. 19 shows the plot of accuracy and coverage as the confidence threshold increases on the SVHN test dataset.

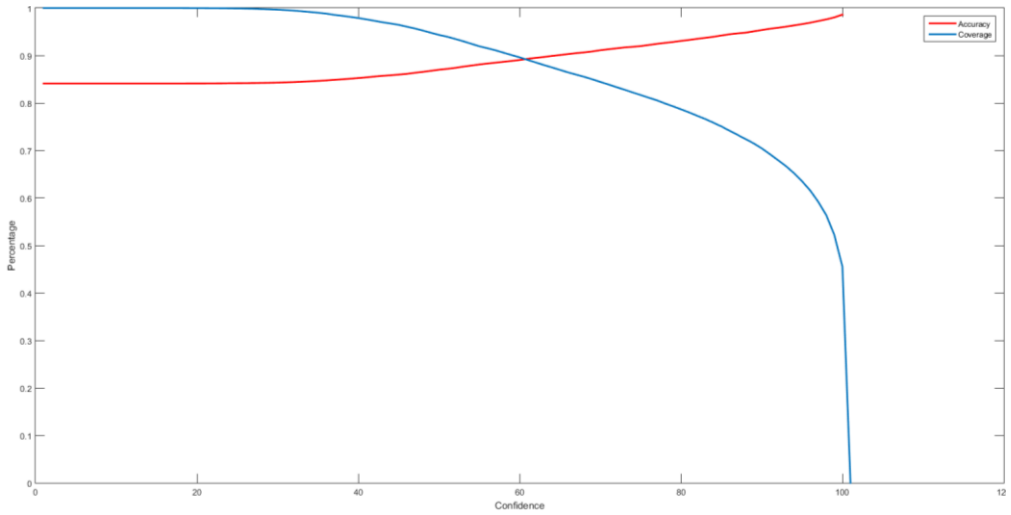

Fig. 19. Plot of accuracy and coverage as the confidence threshold increases on the SVHN test dataset.

The ESP dataset achieved 98% accuracy with 85.3% coverage. The confidence threshold is set at 80.3. Fig. 20 shows the plot of accuracy and coverage as the confidence threshold increases on the ESP dataset.

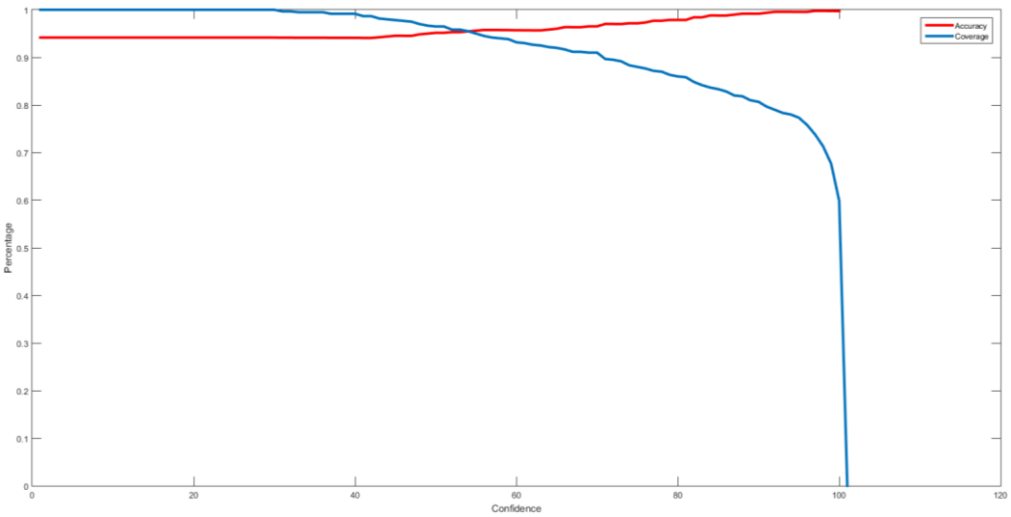

Fig. 20. Plot of accuracy and coverage as the confidence threshold increases on the ESP dataset.

VI. Conclusion

This paper has covered the process and results of classifying digits from ESP using neural networks. It is shown that OCR on ESP with high accuracy and acceptable coverage is possible. There are several limitations in its applicability to ESP. It is hard to achieve proper pre-processing of older ESP classes, however as companies switch to newer plates this will become a lesser issue. If classifications for a series of digits has to be 'correct' for all digits, this will also reduce accuracy. An estimate for the overall accuracy on the ESP dataset is 86.1%, assuming a mean digit string length of 2.5. This will also reduce the coverage to an estimated 67.2%.

Using neural networks without the limits of computational power mentioned in section III.C will most likely improve results. Several optimizations are possible: Reducing scaling, increasing hidden nodes, increasing layers and different network architectures. The most promising to look into is Deep Convolutional Neural Networks such as in [4], an architecture which closely resembles the process of biological visual mechanisms. Utilizing boosting [14] while training or increasing imagery available through translated copies are also possible avenues for improvement.

Improving the pre-processing part of the process is also possible, such as making graphs of characters, improved character segmentation or taking several translated images of the original. All these offer improvements to the process as whole or will increase accuracy.

VII. References

[1] Andreas Waal, "Optical Character Recognition (OCR) on Electrical Specification Plates", unpublished.

[2] Andreas Waal, "Optical Character Recognition (OCR) on Electrical Specification Plates", unpublished, Figure 2, pp. 8.

[3] C.Sureshkumar and Dr.T.Ravichandran, "Handwritten Tamil Character Recognition and Conversion using Neural Network", (IJCSE) International Journal on Computer Science and Engineering Vol. 2, No. 7, 2010, 2261-2267.

[4] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud and Vinay Shet, "Multi-digit Number Recgnition from Street View Imagery using Deep Convolutional Neural Networks", arXiv:1312.6082v4 [cs.CV].

[5] http://se.mathworks.com/help/matlab/ref/rgb2gray.html, accessed 10.6.15.

[6] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 1, 1979, pp. 62-66

[7] http://se.mathworks.com/help/images/ref/imresize.html, accessed 10.6.15.

[8] Martin T. Hagan, Howard B. Demuth, Mark Hudson Beale and Orlando De Jesús, Neural Network Design, 3rd ed., self-published, 1. Sep. 2014, pp. 38.

[9] Martin T. Hagan, Howard B. Demuth, Mark Hudson Beale and Orlando De Jesús, Neural Network Design, 3rd ed., self-published, 1. Sep. 2014, pp. 44.

[10] Z. Boger and H. Guterman, "Knowledge Extraction from Artificial Neural Networks Models", Invited paper, IEEE Systems, Man and Cybernetics Conference, Orlando, Oct. 1997.

[11] Martin F. Møller, Neural Networks, Vol. 6, 1993, pp. 525–533.

[12] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu and Andrew Y. Ng, "Reading Digits in Natural Images with Unsupervised Feature Learning", NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.

[13] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu and Andrew Y. Ng, "Reading Digits in Natural Images with Unsupervised Feature Learning", NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011, Figure 4, pp. 6.

[14] Harris Drucker, Robert Schapire and Patrice Simard, "Improving Performance in Neural Networks Using a Boosting Algorithm", Advances in Neural Information Processing Systems 5, [NIPS Conference], 30.11.1992.

[15] A. Khotanzad, "Classification of invariant image representations using a neural network", Acoustics, Speech and Signal Processing, IEEE Transactions Volume 38, Issue 6, 1990.

[16] Arno Berger and Theodore P. Hill, "Benford's Law Strikes Back: No Simple Explanation in Sight for Mathematical Gem", The Mathematical Intelligencer 33.1, 2011, pp. 85-91.

[17] https://en.wikipedia.org/wiki/Benford's_law#/media/File:Rozklad_benforda.svg, accessed 13.06.15.

[18] Hugo Zaragoza and Florence d'Alché-Buc, "Confidence Measures for Neural Network Classifiers", Proceedings of the Seventh Int. Conf. Information Processing and Management of Uncertainty in Knowlegde–Based System, Vol. 1, 1998, pp. 886-893.

[19] J. Savoula, T. Seppänen, S. Haapakoski and M. Pietikäinen, "Adaptive Document Image Binarization", Pattern Recognition, Vol. 33, 2000, pp. 225-236.