Handwritten Character Recognition

Project Report – CS771

11/16/2013

Guided By:
Prof. Harish Karnick
Computer Science and Engineering, IIT Kanpur

Group – 6

Deepak Pathak	10222
Abhishek Dalmia	10018
Vempati Anurag Sai	Y9227645
Ritesh Jha	13111051

1. Introduction

The problem is to recognize hand printed English characters and numerals (26 upper case, 26 lower case, 10 numerals) using Machine Learning. This is basically a classification problem spanning over the range of 62 classes.

This task is fairly non-trivial owing to its intrinsic challenges like varied writing styles, orientation, letter-size etc. It lies at the junction of computer vision, pattern recognition and machine learning [Wiki]. It has a lot of practical applications like automatic cheque reading in banks, number-plate recognition, conversion of handwritten text to e-books, information retrieval from forms and many more. Some applications like cheque reading require 100% fool-proof architecture to prevent any financial losses. This has been the very reason for the problem being extensively pursued by various researchers. A successful design will be a more convincing interface between humans and machines. It also helps us circumvent the manual labor involved in converting paper-work into electronic format. As a result the data can be stored in lesser area, easily searched through and is unlikely to be lost over the years.

About the dataset:

The dataset we are using is *Chars74K-English-Hnd Dataset* [Campos, 2009] 62 Classes and 55 sample images of each class. This hand printed dataset where character in each image is drawn on tablet PC with appropriate thickness. Size of each image provided is height x width = 900 x 1200, with digit in centre of the image in black colour over white background. These are essentially binary images.

2. Literary Survey

Though the on-line handwriting recognition has been almost perfected, the offline recognition is still an open-ended problem due to the missing “trajectory” information [Niu, 2012]. Most of the previous work has tried to address restricted vocabulary like the MNIST challenge. Scaling the same approaches to larger vocabulary is not so trivial. Systems that are capable of dealing with any word presented at the input without relying on a vocabulary have also been proposed but their accuracy is still far below those relying on limited vocabulary [Koerich et al. 2003]. Moreover, with larger vocabulary the need for more training data increases. But the availability of such diverse data is limited. Some techniques tried to employ vocabulary pruning by grouping similar data classes (Ex: Lower-case and Upper-case).

Many methods in the past have made use of pre-decided set of features like Gradient, distance and chain features. All most all of them involve pre-processing stage. Neural Networks, SVMs, k-Nearest neighbors have proven to be trustworthy. Some techniques also involved post-processing to discard unlikely hypotheses. For relevant pre-processing aprroach, we use method suggested in [Liu, 2004]. This method is discussed in detail in the pre-processing section.

The Hybrid Approach towards recognition

SVMs (Support Vector Machines) have been successful in the handwritten recognition task on many instances in the past. CNNs (Convolution Neural Networks) are deep supervised learning

architectures and are known to be a good pattern extractor. But, it's essentially a linear classifier at the last layer. SVM on the other hand tries to find an optimal hyper plane by projecting the data to a higher dimensional space. In view of these facts, the hybrid model employs best of the two worlds by using a superior pattern extractor (CNN) together with a better classifier (SVM). This hybrid model has proven very successful on the MNIST database too [Lauer, 2007].

3. Data Pre-processing

Before proceeding to any technique for pre-processing, we proceeded towards making relevant and critical observations about the data. It was observed that

1) The handwritten characters were concentrated in only some part of the image, the remaining being white space.
2) The size of image was too large to give as input to any of the classifier
3) Some images had noise, for e.g.

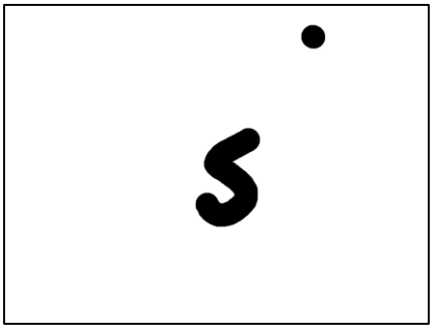


Fig 1: Noise: A dot present in the sample of 's'

4) Characters in different images were of different size and concentrated in different parts of image

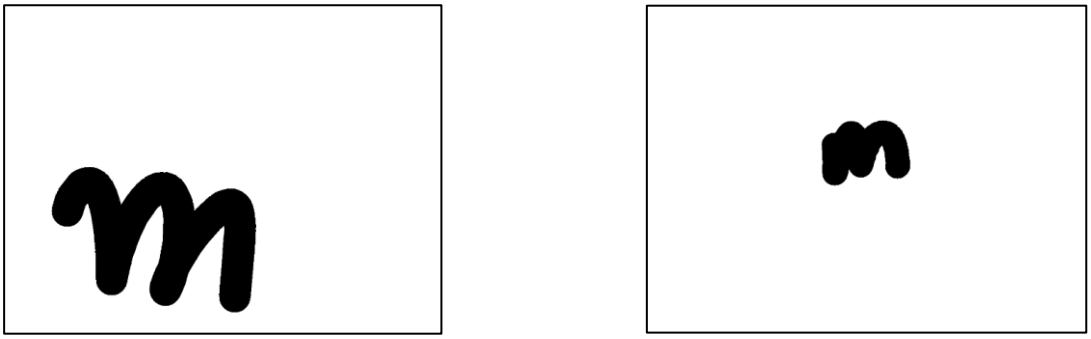


Fig 2: First sample is bigger and concentrated in bottom left part while second sample is smaller and concentrated in the centre.

The above non uniformities in the samples would make the classifiers perform badly. The approaches we have used require standard input images. Our objective in pre-processing was to normalize the samples so as to solve the above mentioned problems. We use the technique suggested in [Liu, 2004] as the best fitted one especially designed for such problems. The step-by-step procedural approach that we followed with minor changes is discussed below:

Step I: Find the centroid of the image

$(p, q)^{th}$ order geometric moment $m_{p,q}$ is defined as follows

$$m_{p,q} = \iint x^p y^q f(x,y)dxdy$$

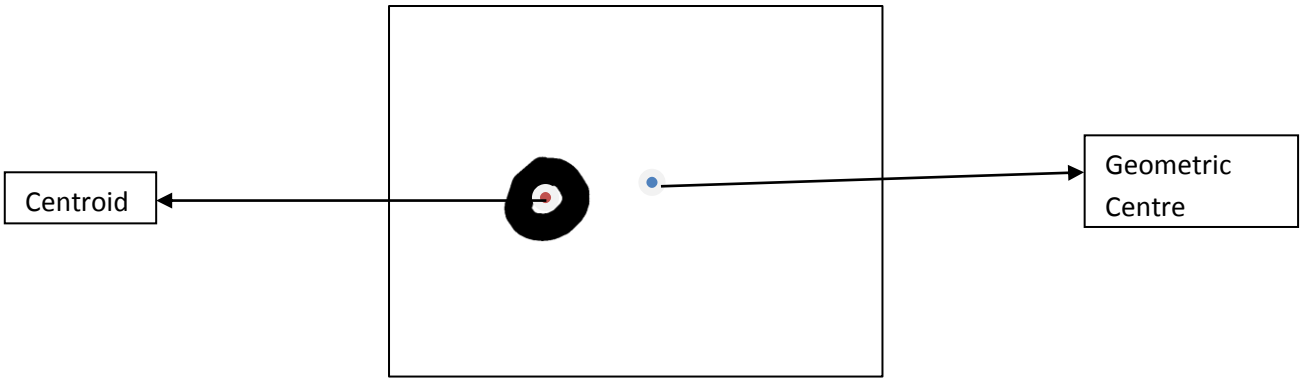
Co-ordinates of the centroid can be calculated as

$$c_x = m_{1,0} / m_{0,0}$$
$$c_y = m_{0,1} / m_{0,0}$$

Where $f(x,y)$ is the value of the pixel I(x,y) in the image. For e.g. in our case binary images f(x,y) = 1 or 0 depending on the whether the pixel is white or black. In other words –

$$c_x = \frac{\sum_x \sum_y xI(x,y)}{\sum_x \sum_y I(x,y)}$$
$$c_y = \frac{\sum_x \sum_y yI(x,y)}{\sum_x \sum_y I(x,y)}$$

Thus centroid is just the weighted average. By finding the centroid , we know in which part of the image our sample is concentrated and thus we can form a bounding box around the centroid as shown in the later steps.



Step II: Find the spread in the image by calculating the second moments

$$\tilde{\mu}_{p,q} = \iint (x - c_x)^p (y - c_y)^q f(x,y)dxdy$$

$$\mu_{2,0} = \tilde{\mu}_{2,0} / m_{0,0}$$
$$\mu_{0,2} = \tilde{\mu}_{0,2} / m_{0,0}$$

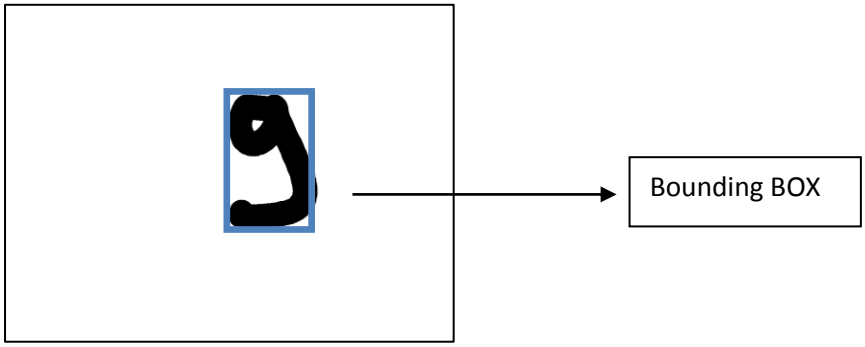
μ can be seen as the variance of distance of image from the centroid. Thus for images in which characters are bigger in size, μ would be larger while it would be small if the characters are small.

Step III: Find the bounding box

Bounding box is taken to be

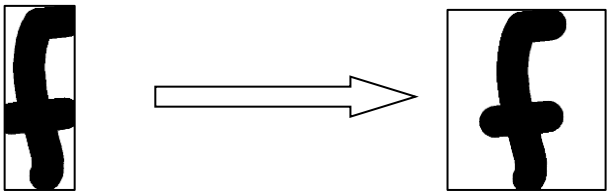
$$(c_x - 2\sqrt{\mu_{2,0}}, c_x + 2\sqrt{\mu_{2,0}}) \times (c_y - 2\sqrt{\mu_{0,2}}, c_y + 2\sqrt{\mu_{0,2}})$$

In this bounding box the characters are centered and are spread across the bounding box



Step IV: Map to the 48x48 image

- Aspect ratio preserved
- Taken the longer dimension equal to 48 and compressed the image keeping the aspect ratio same. The resulting image was centered in a 48x48 image and white padding was added along the shorter dimension.



After this pre-processing step, we proceed to training and classification techniques.

4. Approaches Implemented

In this project, we primarily tested three approaches for classification along with suggesting and implement a novel technique for combination of two strong classifiers in a fairly robust but intuitive way. Now we discuss the classifiers in detail one by one.

A. Support Vector Machines (SVM) based Classification

In this approach the SVM classifier was trained directly using the pixel values of the input images. As the handwritten character images were large to directly use in SVM, firstly the pre-processed images were further down sampled to a 24x24 pixel resolution and each pixel of down sampled image was considered as a feature for the classifier model. Thus the number of input features to SVM was reduced to 576 and the 24x24 handwritten character images were then trained and tested using the "LibSVM" multiclass classifier with RBF kernel. For multi-class classification 'one-vs-all' approach was followed where for each class a 2-class SVM classifiers is build such that k^{th} SVM has +ve label for all k^{th} class examples and –ve labels for all the rest of the examples.

B. Deep Convolution Neural Network (CNN) based classification

Convolution Neural Networks are the special kind of the multi-layer neural networks that are designed to classify the visual patterns directly using the pixel values of the images. They require a very little image pre-processing and are highly robust to image distortions and other transformations like translation, scaling, skewing and rotation. CNN's are capable of efficiently recognizing any visual pattern having very high inconsistencies like handwritten characters.

Architecture of CNN

CNN is a special kind of multi-layer neural network which slightly differs in architecture from the usual neural networks. Each layer in CNN contains set of the feature maps which can be considered as a normal image where each pixel value of image denotes a neuron. The Input layer i.e. the first layer contains only one feature map which is the Input image itself.

The Input layer of the CNN reads the pixel image directly through the input layer neurons and then feeds it through other layers of network. In CNN, each layer consists of two sub-layers called the Convolution Layer and Sub-Sampling Layer. Thus each feature map actually consists of two feature maps i.e. the convoluted feature map which is produced by convoluting the previous layer feature maps using appropriate kernel and the sub-sampled feature map which is downsampled version of convoluted feature map. Each feature map in the last layer consists of only 1 pixel each and each pixel value is considered as a feature for the linear classifier that produces the required output. Like almost every other neural networks CNN are also trained with a version of back-propagation algorithm.

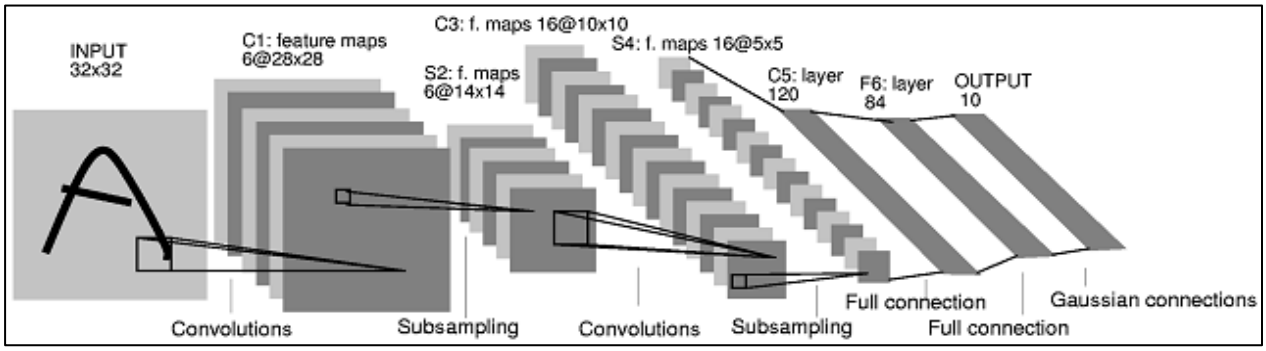


Figure: Sample Architecture of Convolutional Neural Network (CNN).
Note: This is not the one used in our implementation, this is just for demonstration. Image Credits: [LeCun, 1998]

C. Hybrid Approach: Combining SVM Classifier with CNN feature extractor

This is our approach which we expect to perform well. Overall idea is to stop the output of CNN one or two layers before the output layer and thereby use the numerical vector obtained as input for SVM. We expect that in the first phase CNN will hope fully extract essential features for the images which are then learnt in supervised manner using SVMs.

This is automatic feature extraction. We will explain this technique in details in next section.

5. Hybrid of Convolutional Neural Network with Support Vector Machines

In this section, we will discuss the hybrid model in detail. We show that the features generated by this hybrid model are not just arbitrarily reduced dimensional vectors but makes sense when back-projected to make images.

Automatically Generated Features

The structure of the convolution deep belief network (7 layered mentioned earlier) captures some feature of the given input in each of its feature map, which get refined over time. This very idea is described in [Lee, 2009] where they describe how convolution deep belief nets improve the extracted features in subsequent layers. Each kernel matrix (or weight) which is shared between the feature maps of intermediate layers, adapt to capture different features.

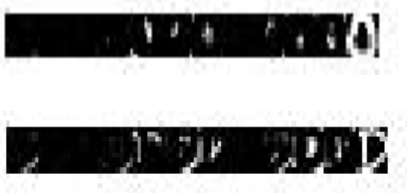


Figure: Visualization of features obtained in form of images. These are 12 feature maps arranged side by side horizontally i.e. First horizontal strip is 12 featured 9x9 images of ‘0’ arranged beside each other. Second horizontal strip is for capital ‘D’.

We exploit this very property of the convolutional deep belief networks by extracting the features from the last layer. In figure, we have shown how these features would actually look if visualized in the form of image. There are 12 feature maps in the architecture and when arranged in a row of images, they clearly show inferable aspects. From observation, it is clear that each feature map is capturing some aspect e.g. edges, loops etc. of the input digit; and more importantly this arrangement is different for different classes as it should be because weights are arranged to capture features of each class in some way.

No Loss in Classification Technique

This is easy to show that in any case by adding explicit discriminative classifier like SVM at the end of the last layer, we are not losing in anything in any case. This is so because at the last layer CNN reduces the output of previous subsampling layer into a linear array, and SVM being a generalization

of linear discriminant analysis will preserve this property. In addition to this, they will be mapped to higher dimension space to attain separability using kernel functions.

In the experiments section, we will discuss the results which we have obtained using this technique, and they are comparable to the State of the Art in this area of digit-character recognition.

6. Details, Experiments and Results

Approach 1: Using one-all multi class SVM Classifier

Input Image Size: 24 x 24 pixels
Total Number of Features input to SVM: 576

Total Examples	55 Examples x 62 Classes
Training Set	44 Examples x 62 Classes
Test Set	11 Examples x 62 Classes
Test Set Accuracy	75.1319 %
Full Cross-Validation (9 Folds)	70.3812 %

Approach 2: Using Convolution Neural Network (CNN)

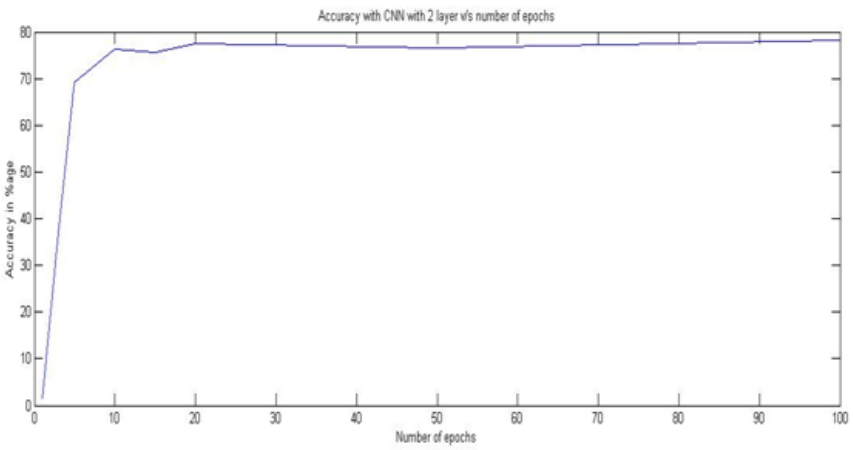
The architecture of the CNN that we finally converged to, for classification is as follows. This similar to LeNet-5 architecture used in [LeCun, 1998]

Input
Convolution(K=5,6maps)
SubSamplig(2,6maps)
Convolution(K=5,12maps)
SubSamplig(2,12maps)
Last Linear Layer
Output

We tried and experimented with **deeper architecture of one more convolutional and subsampling layer containing 10 feature maps**. But not only running time increased by large amount but accuracy was also dropped by **2%.**

Input Image Size: 48 x 48 pixels

Total Examples	55 Examples x 62 Classes
Training Set	44 Examples x 62 Classes
Test Set	11 Examples x 62 Classes
Test Set Accuracy	76.54 %
Epochs	50



Figure(a): % Accuracy v/s Number of Epochs with 2 Layer CNN

Approach 3: Using Hybrid: CNN + SVM Classifier

Input Image Size: 48 x 48 pixels
Number of features to SVM: 972

Total Examples	55 Examples x 62 Classes
Training Set	44 Examples x 62 Classes
Test Set	11 Examples x 62 Classes
Test Set Accuracy	≈81 %
Full Cross-Validation (9 Folds)	≈83 %
Epochs	20

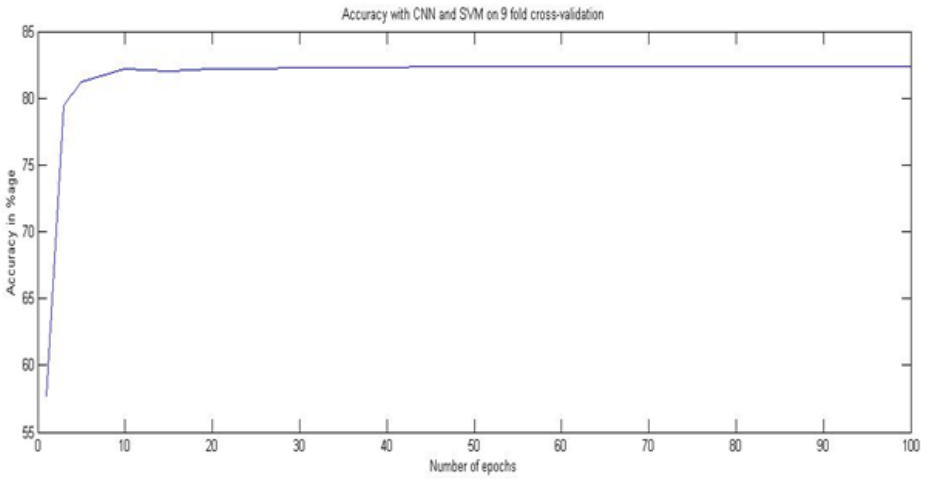


Figure (b): % Accuracy v/s Number of Epochs on 9 fold cross-validation

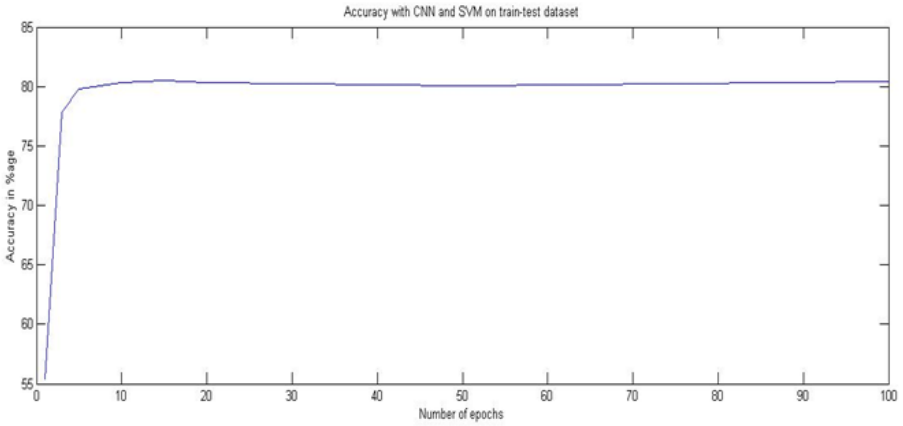
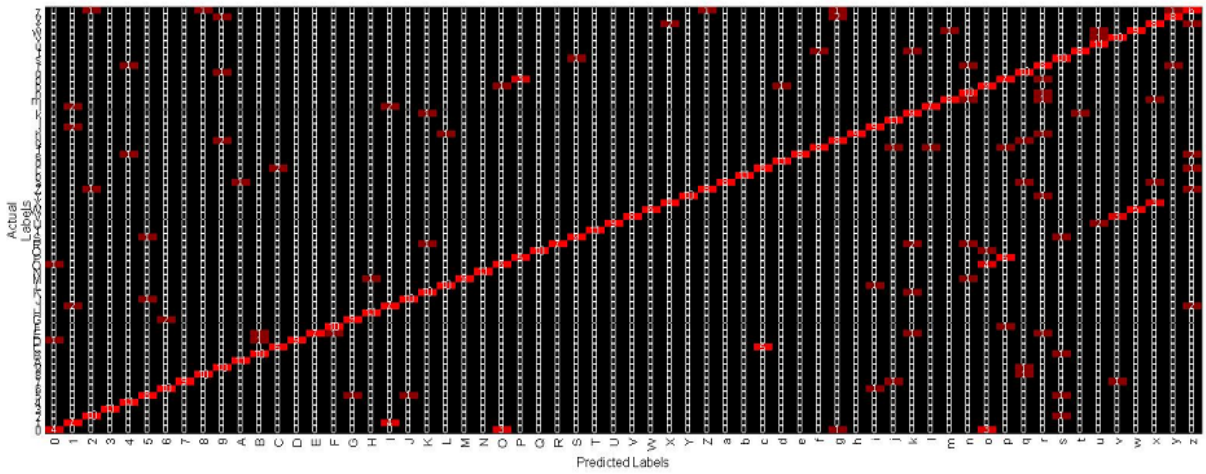


Figure (c): % Accuracy v/s Number of Epochs on test set

7. Analysis of Results

If we compare the 'accuracy v/s epoch' curve of only CNN and the hybrid model, then we come across a very important empirical result. The hybrid model is faster to train than simple CNN model. The time complexity of both the models is driven by training time of CNN architecture, as 'libsvm' implementation of SVM has made their training very efficient and fast. So in the hybrid model, we don't need to go beyond 20 epochs to achieve high accuracy, while epochs to train a simple CNN model was larger. This is evident in Figure(c), **where the learning curve lies towards the extreme left with high vertical slope.**

Heat Map of confusion matrix for Hybrid SVM and CNN classifier:



Some of the **interesting facts** seen from the confusion matrix:

1. Three '0's are misclassified as 'o' but only 1 'O' is misclassified as zero and no 'o' is misclassified as zero.
2. Some 'O' are misclassified as 'o' but only one 'o' is misclassified as 'O'

3. Only one ‘5’ is misclassified as ‘s’ but no ‘s’ is misclassified as ‘5’.
4. Only one ‘S’ is misclassified as ‘5’ but no ‘5’ is misclassified ‘S’

Various other results can easily be identified on deeper glance at the confusion matrix. **Overall we conclude that Hybrid of CNN+SVM is giving the state-of-the-art comparable results without much emphasis on feature extraction and pre-processing.**

References

[1] Teo de Campos, Bodla Rakesh Babu, and Manik Varma. "Character recognition in natural images." (2009).

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, november 1998.

[3] Maji, Subhransu, and Jitendra Malik. "Fast and accurate digit classification."*EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-159* (2009).

[4] Lee, Honglak, et al. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009.

[5] C. L. Liu, K. Nakashima, H. Sako, and H. Fujisawa, "Handwritten digit recognition: investigation of normalization and feature extraction techniques," Pattern Recognition, 37(2): 265-279, 2004.

[6] F. Lauer, C. Y. Suen, and G. Bloch, "A trainable feature extractor for handwritten digit recognition," Pattern Recognition, 40(6): 1816-1824, June 2007.

[7] Website: http://en.wikipedia.org/wiki/Optical_character_recognition, "Optical Character recognition"

[8] Niu, Xiao-Xiao, and Ching Y. Suen. "A novel hybrid CNN–SVM classifier for recognizing handwritten digits." *Pattern Recognition* 45.4 (2012): 1318-1325.