OCR for Mobile Phones

Kathryn Hymes and John Lewin

Abstract

Lighting, camera resolution and focus, skew (shear), and processing power all contribute to the difficulties in developing effective OCR applications using mobile phones. We propose a novel algorithm using PCA, divergence, and a global neural network to develop an OCR package that levers user-specific parameters with a neural network trained on an extended data set.

1 Introduction

The lack of accurate handwriting recognition (OCR) on mobile devices, using the on board digital camera, is a barrier to the development of commercial applications such as document scanners, translators, and business card readers. Specific challenges for OCR on a mobile platform include

- Low processing power
- Low quality images
- Tilt, skew, and rotations
- varied lighting and shadows

An unexplored advantage of mobile OCR is continuous generation of training data and the ability to maintain local, user specific local training parameters as well as global, large scale training parameters.

A typical solution would apply variants of a 5 layer neural network (Le Cun et. all, 1998). However, training a neural network on a mobile phone is computationally unfeasible, this approach does not lever the advantages of user-specific parameters that the mobile application offers. We present a novel three stage algorithm that leverages dimension reduction, a global neural network trained externally, and a custom local algorithm based on KL divergence as a package for mobile OCR.

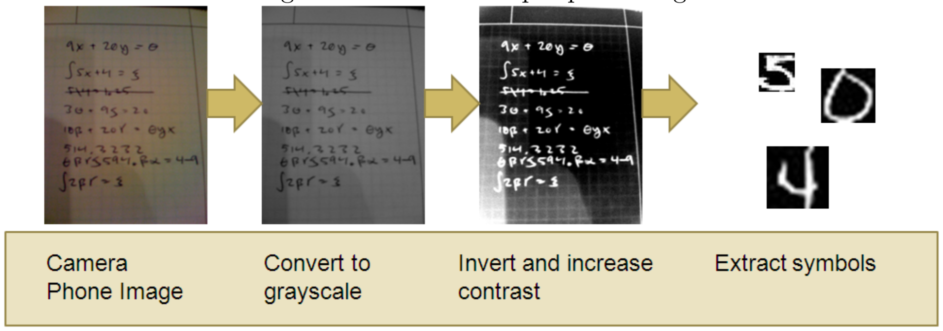
2 Data Processing

Pre-processing is a sizable challenge that we only touch on here. We require images to be taken of handwriting on a Dual Pad brand engineering pad. We have 3 working data sets:

- ZipCode 7291: A reference data set generated from 7,291 digits from zip code scans by the US postal service. The data range is 0-9.
- iPhone 170: Symbols ranging from 0-9, the integral, and a few Greek characters extracted and processed from pictures taken with an iPhone
- iPhone 500L: A data set of 500 pictures based on ten transformations each of fifty characters. The data range is 0-4.

Symbols are further normalized and adjusted for skew. The angle for adjusting for skew is the angle between the top and rightmost line of the Dual Pad. These lines are obtained by extracting the dark points in that region from the image, and fitting the line of highest variance using PCA. After processing, each symbol is represented as a grayscale 16×16 pixel matrix with values ranging from 0-225, or equivalently as a vector with 256 parameters.

Figure 1: iPhone data pre-processing



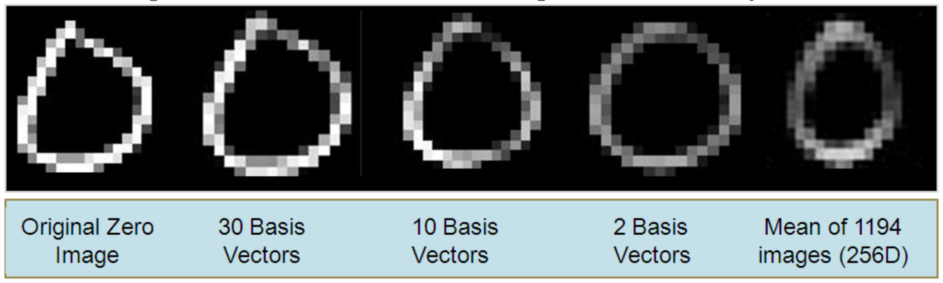
3 Dimension Reduction

We use Principal Component analysis to simplify the problem. We find the symmetry and high dimensionality of our data lend to sizable dimension reductions while maintaining image quality. A reduction to n dimensions is attained by projecting each data point onto the n greatest eigenvalues of the covariance matrix. E.G., let x_i be a data point in the original high dimensional space, M the mean matrix, and V the matrix of n greatest eigenvectors. Then each is represented in the lower dimensional space as

$$y_i = V^t(x_i - M)$$

Figure 3 illustrates a reduction of a 256 dimension image to 30, 10 and 2 dimensions, respectively along with the mean zero image. We find that the 2D projection is similar to the mean zero, with projections using 10 and 30 eigenvectors are increasingly approaching the original zero image.

Figure 2: Dimension reduction using PCA on a zero symbol



4 The KL Divergence Predictor

4.1 Building the Reference Set

Let X_i^α be the i^{th} element of a training set corresponding to symbol α (for example, X_i^5 is the i^{th} image corresponding to the number 5 in the training set). Define $\bar{X}$ as

$$\bar{X}^\alpha = \frac{1}{n}\frac{1}{\gamma}\sum_n X_n$$

So that $\bar{X}^\alpha(m,n)$ represents the estimated probability of a black pixel being found at the $(m,n)^{th}$ coordinate of a given symbol α. The parameter γ is the largest value in the image matrix (usually 255).

4.2 Using $\bar{X}$ as a predictor

Let Y be an unknown symbol. Normalize as above ($\bar{Y} = \frac{1}{\gamma}Y$). For each symbol $\bar{X}^\alpha$, find the KL divergence between $\bar{X}^\alpha$ and Y given by

$$KL(\bar{X}^\alpha, \bar{Y}) = \frac{1}{mn}\sum_{m,n} \bar{Y}(m,n) \log \frac{\bar{Y}(m,n)}{\bar{X}(m,n)}$$

Repeat the experiment for each α. Finally, predict Y to be the symbol with the least divergence. E.G.,

$$Y := \arg\min_\alpha KL(\bar{X}^\alpha, \bar{Y})$$

4.3 Testing and Validation

We compare the KL divergence predictor to a vanilla, feed forward neural network with a single hidden layer with 20 nodes, using the normalized post-office data set and the processed iPhone data set. The results are show in table 2.

Data Set	KL Error Rate	NNet Error Rate
ZipCode 7291	18.5%	33.1%
iPhone 170	48.2%	>95%
iPhone 500L	11.0%	32.9%

Table 1: KL Testing versus Vanilla Neural Network

4.4 Analysis

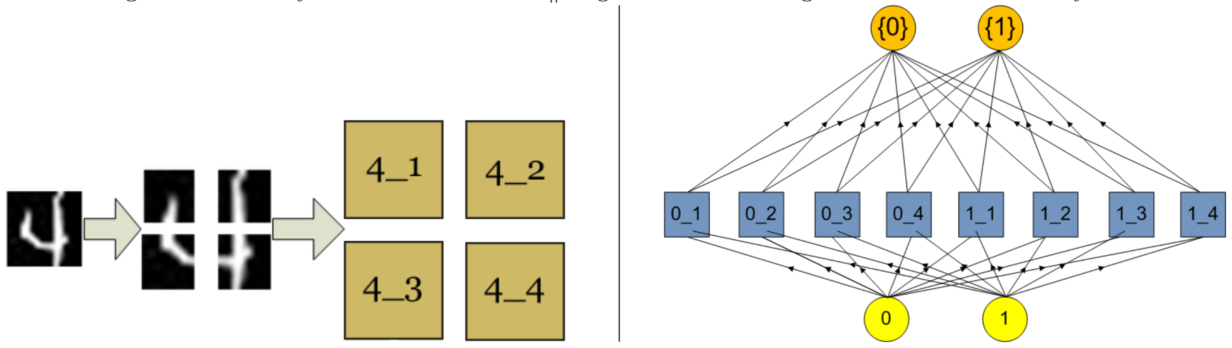
Table 2 is somewhat misleading. On larger data sets the vanilla neural network data has been shown to be $> 90\%$ effective, and multi-layer neural networks represent the state of the art in character recognition with success rates great than 99.9% (Le Cun et al.,1989). Furthermore, the iPhone 500L data set is quite limited in size and interpretation of data should be weighed appropriately. That said, the KL approach does suggest improved results for a specific persons handwriting, and in stark contrast to a multi layer neural network, the computational requirements for training are feasible on an iPhone or comprable hand held device.

5 The Whole Shebang: A global neural network with server generated parameters operating locally with a low dimension user specific KL function as the decision function

5.1 Overview

Our system incorporates all of the above techniques. First, a custom global neural network is trained away of the mobile device. Each letter is evaluated locally with a KL divergence decision function, based on locally generated parameters. In this way we lever the advantage of a neural network with use specific training attainable in a computationally feasible way locally. In practice, we would would likely use a multi-layer NNet similar to what was described in Le Cun et al. (1998), except using a KL output function as follows. Each letter is extracted as in section 2, and reduced to 10 basis vectors as in section 3. Each image is furthermore subdivided into 4 quadrants (the hidden layer) as diagrammed in fig. 5.1. The input layer is the

Figure 3: Left: Symbol Discretization || Right: Network Diagram for the case of 2 symbols



number of symbols stored, and there are 4X the number of symbols in the hidden layer as there are in the input layer corresponding to 4 quadrants per symbol. This is calculated in much the same way as a regular feed forward neural network, except the input is a matrix Y, and the output function $g_{n,k}$ corresponding to the n^{th} quadrant of the k^{th} symbol is given by

$$g_k^n = \tanh\left(\frac{KL(Y, \alpha_k^n)}{c}\right)$$

Where α_k^n is an approximate distribution of the n^{th} quadrant of the k^{th} symbol corresponding to $\bar{X}$ from section 4, Y is a distribution corresponding to the n^{th} quadrant of the unknown symbol, corresponding to $\bar{Y}$ in section 4, and tanh is the hyperbolic tangent. The constant is used to c adjust where the .5 boundary lies (equivalently, c sets the degree of similarity required, measured by divergence, such that $\bar{X}$ and $\bar{Y}$ are classified as coming from the same symbol). This model has 3 major advantages:

- The power of the neural network is leveraged on an unlimited data set from as many people as possible
- The on-phone calculations are limited to evaluation of the neural network and divergence calculations, within the computational capabilities of a multimedia phone
- The on-phone training data (the α_k^n matrices) is *user specific* and therefore customized to his or her precise writing style.

The final point is critical: we are generating a quasi-customized algorithm for the phone user without requiring him to train enormous amounts of data, or requiring heavy processing on his phone.

5.2 Results

We trained the data using the Zip Code 7291 data set and (independently) the iPhone 170 data set, leaving 10% out of the training data for testing. The weights were generated by training a single layer feed-forward neural network, using the g_k^n output function. The constant c was found before hand to be 27.47 for the Zip Code 7291 data set and 10.99 for the iPhone 170 data set, coresponding to target divergence values of .01 and .05 (.01 resulted in very few results being less than .5 on the iPhone data set). Our results can be seen in table 2.

5.3 Analysis

The results are encouraging – the network succeeds in obtaining the lowest error rate we've seen on this data on a model customized for the user. Furthermore, we have used a very simple neural network – a more

Data Set	Error Rate
ZipCode 7291	3.7%
iPhone 170	8.8%

Table 2: Final Error Rates using the hybrid NNet

robust network such as that used in Le Cun et. all will likely generate even better results (as will improved training data). However, there are two reasons to remain skeptical. First, the data set is quite small. Second, the NNet data is trained on the same data as the α's are, so it is not clear how big of an advantage the user specific g_k^n functions are. However, we believe the results are encouraging enough to warrant further exploration.

6 References

(i) Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning*, Springer 2001

(ii) Ng, A. *CS229 Lecture Notes*, Fall 2008

(iii) Suzuki et al. *INFTY - An Integrated OCR System for Mathematical Documents*, DocEng '03, Grenoble, France. 2003

(iv) Muller et al. *An Introduction to Kernel-Based Learning Algorithms*, IEEE Transactions on Neural Networks, Vol. 12, No. 2, March 2001

(v) Barber, D. *Learning from Data: Dimensionality Reduction*, 2004

(vi) Shlens, J. *A Tutorial on Principle Component Analysis*, Institute for Nonlinear Science, UCSD, 2005

(vii) Dailey, M. *Principal Component Analysis Tutorial*, Asian Institute of Technology, 2008

(viii) Le Cun, Y., Bottou, L., and Haffner, P. (1998). Gradient-based learning applied to document regognition, *Proceedings of the IEEE*,**86**(11), 2278-2324