Classifying Individual Digits in Bank Account

Zhaoyang Cheng(4620585) Kaixin Ding(4623649)
Yunhan Wei(4588401)
Email:zyclark.cheng@gmail.com

Contents

1 Introduction

Optical Characters Recognition is to automatically recognize the scanned images containing characters, which has been studied for quite a long time and applied in many fileds. In this paper we propose several feature extraction methods and compare different classifiers for recognition of digits number in bank account. Implementation is by *Matlab*, and the dataset supplied is US National Institute of Standards & Technology, NIST (`http://www.nist.gov/`).
There are 2 scenarios. One is pattern recognition system is trained once, and then applied, which means the training dataset is large(200-1000 objects per class), scenario 2 is the system is trained for each batch of cheques to be processed, which means the training dataset is much smaller(at most 10 objects per class).
First we use raw pixel to train the classifier, then features are extracted. The features extracted are:
(1):invariant moments;
(2):Loal density distribution of stroke;
(3):Histogram of Oriented Gradients;
(4):SIFT Decriptor.
Classifiers used are:
k Nearst Neighbour Classifier(knnc)
Parzen density based Classfier(parzenc)
Logister Linear Classifier(loglc)
Fisher Discriminant(fisherc)
Nearest mean classifier(nmc)
feed forward neural network classifier by backpropagation(bpxnc)
Support vector classifier(svc)
Normal densities based linear classifier(ldc)
Normal densities based quadratic (multi-class) classifier(qdc).
vote combiner of [knn([],2),parzenc,fisherc]

2 Preprocessing of Image

There are two steps in preprocessing, first the noise of image should be reduced then the slant of the digit should be kept same.
For Noise Reduction, A Laplacian Mask is used(for sharpening the abrupt change).

$$\bigtriangledown^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \tag{1}$$

$$= [f(x+1,y) + f(x-1,y) + f(x,y+1), f(x,y-1)] - 4f(x,y) \tag{2}$$

For acquiring more better result, the centeral coefficient is set as -8, so the mask is:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -8 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{3}$$

Since handwritten digit is pretty casual, so the slant of digit should be corrected such that the orientation of principle component should be parallel to the column edge of the image.two dimensional (p+q) order moments of a MxN digital image $f(x,y)$ is:

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q f(x,y) \tag{4}$$

where p,q=0,1,2...,the (p+q) order central moments is defined as

$$\mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x-\overline{x})^p (y-\overline{y}) f(x,y); \quad \overline{x} = \frac{m_{10}}{m_{00}}, \overline{y} = \frac{m_{01}}{m_{00}} \tag{5}$$

And we can calculate the orientation of the main component as follows:

$$\theta = \frac{1}{2} atan2(\frac{2u_{11}}{u_{20} * u_{02}}) \tag{6}$$

We conclude a transform matrix which sets the orientation vertivcal

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & tan(\pi - \theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

however, our matrix sometimes set the image vertical but reverse, so we find(in StackOverFlow) a transform matrix to overcome the reverse.

$$\begin{bmatrix} 1 & 0 & 0 \\ sin(0.5 * \pi - \theta) & cos(0.5 * \pi - \theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

This function works pretty fine to set the digit in image vertical.

3 Features Selection and Extraction

After preprocessing image, we come to select and extrct features from image. Raw pixel can be used as input for the classifier, we resize every image to 32x32 pixels and make each image 1x1024 feature vector.

classifier	error	TrainTime	TestTime
knnc([],2)	0.3230	0.317	121.1440
parzenc	0.9000	0.280	124.5450
loglc	0.2890	66.435	119.7950
fisherc	0.2450	0.613	119.2630
nmc	0.9090	0.0876	120.2600
bpxnc	0.863	8.7690	119.7990
svc	0.317	0.822	120.1700
ldc	0.9	1.4440	120.1820
qdc	0.9	8.4330	120.3800

Table 1: Raw Pixel Scenario 2(10 training objects per class)

Using raw pixel to train classifier is computationally expensive and possibly confronts curse of dimensionality. So some typical features should be extracted from the image.
The features extracted in this report are:
(1):Invariant moments;
(2):Loal density distribution of stroke;
(3):Histogram Oriented Gradients;
(4):SIFT Decriptor.
Dimensions of them are 14; 7; 324; 128 respectively. The meaning of them are explained in detail below.

3.1 Invariant Moments

The first feature extracted, invariant moments make use the central moments defined in (5) of a image, the normalization central moment is defined as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \qquad \text{while} : \gamma = \frac{p+q}{2} + 1 \tag{9}$$

There are 7 set of two dimensional invariant moments, which are not sensitive to translation,zoom,mirror and rotation, these formula can be derived from basic central moments [2]. I use the function `invmoments()` provided by Gonzalez[1],which returns the seven set of invariant moments, noted as $H_1 \to H_7$. After some experiments in matlab,we found three of these invariant moments have large error when used as features for different digit recognition, in Table 3, $H_5 \to H_7$ have much more error than $H_1 \to H_4$, so we just pick first 4 invariant moments, however, since feature dimension is too small, when there are 10 training objects and 100 test objects per class, the error is 0.668(here the classifier is parzenc), which is unacceptable, hence, we give up further experiments.

Digit	H_1	H_2	H_3	H_4	H_5	H_6	H_7
3_1	0.4738	1.4216	2.1146	2.4646	4.7729	3.2728	-5.2959
3_2	0.3897	1.5558	2.2299	3.2320	-6.0728	-4.0462	6.1635
4_1	0.4339	2.4843	1.8089	4.2242	7.7192	-5.6697	7.2661
4_2	0.3487	1.3337	1.7671	2.3069	4.3605	3.0287	4.9116

Table 2: 7 Invariant Moments

3.2 Local density distribution of stroke

The second feature extracted is local density distribution of stroke, the idea is to divide the image into seven row blocks by six horizonal lines, then it acts in the same way for seven column blocks with six vertical lines. 14 local density of these blocks are acquired, representing the features of strokes density distribution. The result evaluated by `eval_nist()` is 0.73 when 10 training objects and 50 test objects per class(here the classifier is parzenc). So this feature can not satisfy our need as well.

3.3 Histogram of Oriented Gradient

The third feature extracted is Histogram of Oriented Gradient. First a gradient image is returned by applied [-1,0,1] and [1,0,-1]' for x axis and y axis respectively, the horizaonal and vertical gradient for pixel (x,y) in image is

$$G_x(x, y) = H(x + 1, y) - H(x - 1, y) \tag{10}$$

$$G_y(x, y) = H(x, y+1) - H(x, y-1) \tag{11}$$

And the magnitude and direction of gradient in pixel (x,y) is

$$\begin{aligned} G(x, y) &= \sqrt{G_x(x, y)^2 + G_y(x, y)^2} \\ \alpha(x, y) &= tan^{-1}\left(\frac{G_y(x, y)}{G_x(x, y)}\right) \end{aligned} \tag{12}$$

the image is divided into cells, 8x8 pixels per cell in our experiment, and we will make a 9 bins histogram based on the gradient orientation for every cell, which means 360 degree is divided into 9 divisions, and gradient orientation will projection in corresponding bin, and the weight of the every projection is corresponding gradient magnitude, now we have 9 dimensions feature in every cell, then the cells are normalized into bigger block, which is less sensetive to illumination vatiation, the block here is 2x2 cells,here we use one step scanner, so one cell can appear on several blocks, hence we get 9 blocks,every block have 4 cells and every cell have 9 bins, so the final feature dimensions are 9x4x9=324. The error given by `nist_eval()` is low for both scenario 1 and 2, relevant data

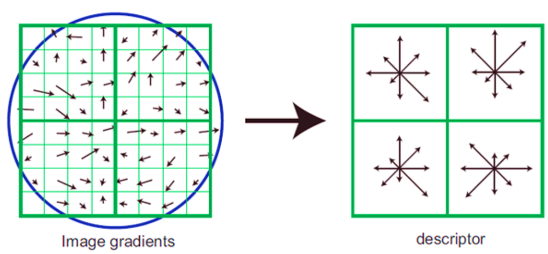


Figure 1: orientation divided(from wiki)

is given in Table 3 and Table 4;

classifier	error	TrainTime	TestTime
knnc([],2)	0.034	0.014	119.0620
parzenc	0.033	4.512	117.5470
loglc	0.045	14.5090	120.7290
fisherc	0.040	2.9770	118.4860
nmc	0.106	0.3850	123.4070
bpxnc	0.073	13.0810	117.3860
svc	:(	:(	:(
ldc	0.033	1.0020	122.2350
qdc	0.083	0.4170	124.320
votec*	0.029	7.4820	129.3600

Table 3: HOG Scenario 1 (500 training objects per class)

N.B. votec* is vote combiner of [knn([],2),parzenc,fisherc]

classifier	error	TrainTime	TestTime
knnc([],2)	0.1460	0.012	121.1394
parzenc	0.1370	0.0275	121.8485
loglc	0.1480	0.4543	122.3315
fisherc	0.1470	0.4494	122.2939
nmc	0.1660	0.0763	119.0940
bpxnc	0.883	11.640	121.0640
svc('r',2)	0.09	0.818	121.9650
ldc	0.9	0.3780	120.3670
qdc	0.9	0.5230	120.3420
votec*	0.1260	0.4510	120.3610

Table 4: HOG Scenario 2 (10 trainging objects per class)

votec* is vote combiner of [knn([],2),parzenc,fisherc], actually we try some stacked combining methods, votec does better than others and combining three classifiers does better than combing two.

3.4 SIFT Descriptor

SIFT(Scale Invariant Feature Transform) algorithm is published by David Lowe in 2004[3], In Lowe's paper, Scale Space is defined as "It has been shown by

Koenderink (1984) and Lindeberg (1994) that under a variety of reasonable assumptions the only possible scale-space kernel is the Gaussian function. Therefore,the scale space of an image is defined as a function L(x, y,sigma) that is produced from the convolution of a variable-scale Gaussian G(x,y,sigma), with an input image I(x, y)." space is controlled by parameter sigma,scale space is composed by different L(x,y,sigma).
There are 4 stages in SIFT algorithm:(1)Scale Space extrema Detection; (2)Keypoints Localization; (3)Orientation Assignment; (4)Keypoints Descriptor.

3.4.1 Scale Space extrema Detection

Keypoints are referred to as those outstanding points that will not change with the illumination condition,e.g.,the corner points,edge points...,the local extremas with orientation information are detexted in different scales. In conclusion, keypoints have three features:scale, orientation, magnitude.
Scale space is defined as a set of image which come from an original image blurred by different scale Gaussian kernel,

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{13}$$

where $*$ is the convolution operation in x and y, $G(x, y, \sigma)$ is scale-variable Gaussian function,σ is scale coordinate, lareger σ is for lower resolution(more blurred) and smaller σ is for higher resolution.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{14}$$

To efficiently detect stable keypoints location in scale space, scale space extrema in Difference of Gaussian function convolved with the image, D(x, y, σ), which can be computed from the difference of two nearby scales separated by a constant multiplicative factor k:

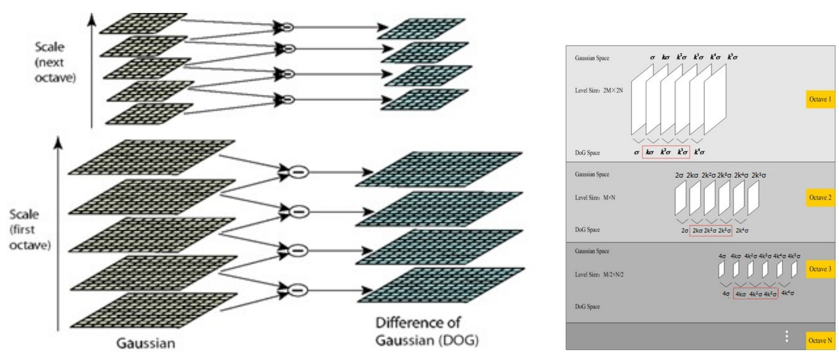
$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{15}$$

There are some reasons to choose DoG function, Details can be found in[4].
An efficient approach to construct DoG is in Figure 2, we down sample the original image to produce different octaves of scale space by a constant k, and every octave is divied into integer number of s intervals, so a scale space is

$$2^{i-1}(\sigma, k\sigma, k^2\sigma ... k^{n-1}\sigma) \tag{16}$$

where k = $2^{\frac{1}{s}}$,i is ordinal number of the octave, and s is intervals number of every octave, number of octave is determined by size of image, and s is usually $3\tilde{5}$, 0^{th}

interval of 0^{th} octave is original image, noted as ori, whose next interval in same octave is produced by convolved ori with Gaussian Kernel, and first interval of 1^{th} octave is produced by down sample the last interval of 0^{th} octave, means it decrease to half size in both width and height.For the continuity of scale space, we must produce s+3 blurred image(intervals)in each octave, a short explanation for this is: image we have s=3, in Figure3.a, there are 3 Gaussian Space and 2 DoG space, the first Octave, whose first interal and second interval is $\sigma, k\sigma$, the second octave, first interval and second interval is $2\sigma, 2k\sigma$, so we should make Gaussian Space $\sigma, k\sigma, k^2\sigma, k^3\sigma, k^4\sigma$, in which case ,the middle three terms $k^2\sigma, k^3\sigma, k^4\sigma$ can have DoG Space,meanwhile, note when s=3, k=$2^{\frac{1}{3}}$, so first interval of second octave is $2k\sigma$, namely $2^{\frac{4}{3}}\sigma$, making it continuously from $k^3\sigma$.



(a) Figure3.a

(b) figure3.b

Figure 2: The DoG pyramid

Then we make accurate Keypoints Localization and elimination edge responses,whose Details will not display in this report.

3.4.2 Orientation Assignment

By assigning a consistent orientation to each keypoint based on local image properties, the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation. The Gradient Magnitude and Orientation of the extrema are computed :

$$Mag(x,y) = \sqrt{(L(x+1,y) - L(x-1,y) + L(x,y+1) - L(x,y-1))^2} \quad (17)$$

$$\theta(x,y) = atan2((L(x+1,y) - L(x-1,y) + L(x,y+1) - L(x,y-1))) \quad (18)$$

where L is the scale of keypoints. By now ,we have obtained all three information of keypoints: location,scale octave,orientation, in our experiment, we set a fixed keypoint in the center of our 32x32 image, (x,y) is (16,16), gradient orientation is 0(horizanal), and scale is 0.

3.4.3 Representation of Descriptor

In Figure3 image gradient magnitudes and orientations are sampled around the keypoint location, using the scale of the keypoint to select the level of Gaussian blur for the image. In order to achieve orientation invariance, the coordinates of the descriptor and the gradient orientations are rotated relative to the keypoint orientation, the blue circle in left figure is a Gaussian weight assignment function(window). The samples around keypoint are then acculated into orienta-

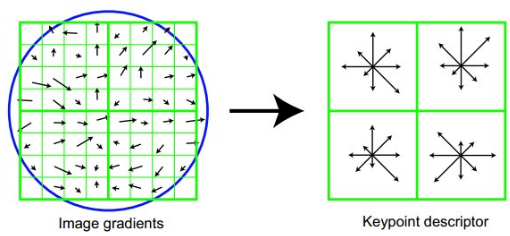


Figure 3: Generation of feature vector

tion histograms summarizing contents of 4x4 subregions, as shown in right,with length of each arror corresponding to the sum of gradient magnitudes near direction within the region, This Figure shows a 2x2 descriptor array computed from a 8x8 set of samples, whereas our ecperimrnts use 4x4 descriptors computed from a 16x16 sample array displayed in Figure4, Thus the dimension of our sift feature is 4x4x8=128.

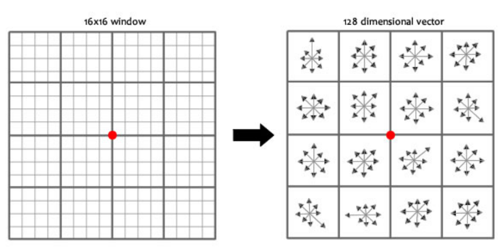


Figure 4: Generation of feature vector

classifier	error	TrainTime	TestTime
knnc([],2)	0.1930	0.0073	118.2644
parzenc	0.1590	0.0160	118.7670
loglc	0.284	0.9590	118.6814
fisherc	0.328	0.3480	119.9221
nmc	0.1870	0.0240	119.5135
bpxnc	0.901	2.358	120.4902
svc('r',2)	0.0900	0.5380	120.4102
ldc	0.4320	0.0960	121.5515
qdc	0.9	0.0960	125.9082
votec*	0.1700	0.9340	123.8860

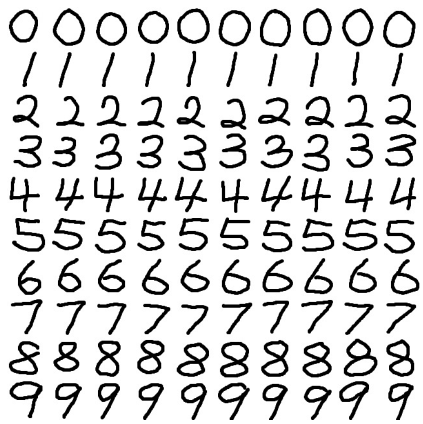
Table 5: SIFT Scenario 2(10 training objects per class)

classifier	error	TrainTime	TestTime
knnc([],2)	0.037	0.1540	118.1661
parzenc	0.032	3.4160	117.5261
loglc	0.066	4.1670	127.6324
fisherc	0.0041	0.8480	116.8364
nmc	0.126	0.1050	117.1264
bpxnc	0.034	12.7	119.7705
svc	:(	:(	:(
ldc	0.048	0.4960	119.3509
qdc	0.069	0.1670	118.1248
votec*	0.029	4.300	125.4670

Table 6: SIFT Scenario 1(500 objects per class)

4 Live Test

We write a testset consisting 100 digits in a sheet, containing 10 objects per class, then segment it into individual image by Split.m , see Figure 5. we also resize every image into 32x32 pixes, for better features extraction and recognition. In LiveTest.m we construct a system for handwritten digits. The error rate is 20% and 5%, when train set is 10 and 500 respectively and we choose the votec[knn([],2),parzenc(),fisherc()] classifier.



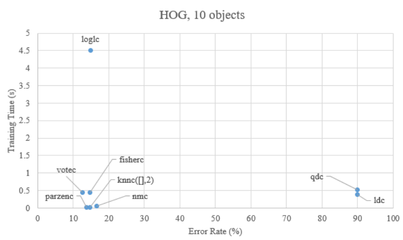
(a) Figure5.a

(b) Figure5.b

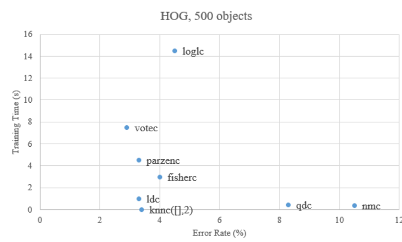
Figure 5: Live Set

5 Recommendations

There are some tips for my clients. 1):More training dataset will help, obvious difference can be found in error rate when 10, 500 training objects per class.
2):Two of four features satisfy the error rate requirement, but I do not combine them, since the scale of those features are not at same level, hence not comparable.
3):Remember, combining the classifiers will lower the the error rate, but do not count it too much because it will not help lot. 4):A Reject Option is a good choice but We do not implement it, since we have reduce noise and correct slant for every image. 5):When considering the time constraints, we would recommendate the knn([],2), This classifier is one of top accurate classifier and at low time consuming. If you want to pursue absolute accuracy or time, Fighure 6 is a reference for you.



(a) Figure6.a



(b) Figure6.b

Figure 6: Error with Time constraints

6 Future Work

It seems we did a good job in this pattern recognition assignment, however, compared to the state-of-art methods and their performance, there are quite some aspects which needs improving, Slection of classification algorithm is important, actually we should try more deep learning methods but we just use back propagation neural network with few nodes. Things more important than classification algorothm is data and feature, we should make the digit thinner and use closing or dilation operations to close small holes in images. Anyway the more we learn, the better result we will get in this OCR(Optical Characters Recognition) problem.

References

[1] W. Gonzalez and R. E. Woods. Eddins. *Digital image processing using MATLAB*, 2009.

[2] M.-K. Hu. Visual pattern recognition by moment invariants. *IRE transactions on information theory*, 8(2):179–187, 1962.

[3] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[4] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.