Digits - A Dataset for Handwritten Digit Recognition

Abstract

In this paper we describe the preprocessing steps for a contributed digit dataset, going all the way from a physical page of paper – filled out by students – past digital scanning to computerized segmentation, resizing, and blurring. Surprisingly, very little expertise can be transferred from other datasets to our new dataset for a state-of-the-art SVM classifier, although the performance for each separate dataset is acceptable. This may indicate that at least SVM, and possibly also other learners, are sensitive to small changes in preprocessing, emphasizing the need not only to create benchmark datasets for handwritten digit recognition, but also to document their preprocessing as detailed as possible and aim to replicate that as well. Our work is a small step in that direction.

1 Introduction

Handwritten digit recognition is a major research topic. E.g. [3] gives a comprehensive survey of the field including major feature sets, learning algorithms, and datasets. Contrary to OCR which focusses on the recognition of machine-printed output, where special fonts can be used and the variability between characters with the same font, size, and font attributes is reasonably small, the inter-person as well as the intra-person variability of handwritten digits is surprisingly high. This makes the recognition problem far more challenging. On the other hand, the moderate number of classes[1] makes it more amenable to be rephrased as a classification task even with a relatively small amount of training data.

[1]10 rather than more than 64 for lowercase and uppercase characters plus digits and comma, dot, colon etc..

In the process of holding a lecture for students at the Medical University in Vienna, we chose to let students contribute and analyze their own dataset for handwritten digit recognition. An input page was designed on which each student was to write 100 digits, equally distributed among zero to nine. Additionally, each student chose whether to contribute his digits to the public domain, for further experiments by the author under non-disclosure to third parties, or not contribute them at all. 44 students contributed their input pages, and of these 37 contributed them to the public domain. The latter data has been made available at `alex.seewald.at/digits`, while we continue using the larger full data for our own experiments. We are looking forward to create similarily-sized datasets each year, provided the lecture will continue to take place. Others are cordially invited to contribute as well – all our code is available freely for non-commercial purposes. Just send us a mail so we know what you intend to do with it, and see if we can help.

We report preliminary results on our and two other handwritten digit recognition datasets, and were surprised that although the task is always the same, almost no expertise can be transferred between these datsets, even though we followed the known preprocessing steps by the letter and also checked format similarity in other ways.

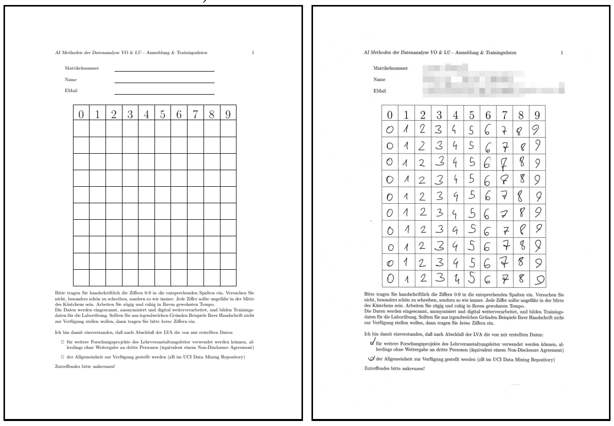
2 Preprocessing

Here, we describe the full processing of our data: from the design of the input page over digital scanning, cell and digit segmentation towards downscaling and blurring.

2.1 Design of input page

Rather than tackling the complex problem of general segmentation just to extract the sample digits, we chose an input page layout that makes segmentation

Figure 1: Left: Empty input page, Right: Filled-in input page after scanning (student data intentionally made unreadable)



AI Methoden der Datenanalyse VO & LU - Anmeldung & Trainingsdaten 1

Matrikelnummer [redacted]

Name [redacted]

EMail [redacted]

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9

Bitte tragen Sie handschriftlich die Ziffern 0-9 in die entsprechenden Spalten ein. Versuchen Sie nicht, besonders schön zu schreiben, sondern so wie immer. Jede Ziffer sollte ungefähr in der Mitte des Kästchens sein. Arbeiten Sie zügig und ruhig in Ihrem gewohnten Tempo.
Die Daten werden eingescannt, anonymisiert und digital weiterverarbeitet, und bilden Trainingsdaten für die Laborübung. Sollten Sie aus irgendwelchen Gründen Beispiele Ihrer Handschrift nicht zur Verfügung stellen wollen, dann tragen Sie *keine* Ziffern ein.

Ich bin damit einverstanden, daß nach Abschluß der LVA die von mir erstellten Daten:

- [✓] für weitere Forschungsprojekte des Lehrveranstaltungsleiter verwendet werden können, allerdings ohne Weitergabe an dritte Personen (äquivalent einem Non-Disclosure Agreement)
- [✓] der Allgemeinheit zur Verfügung gestellt werden (zB im UCI Data Mining Repository)

Zutreffendes bitte ankreuzen!

relatively easy, see the left side of Figure 1. A table consists of horizontal and vertical lines, which are recognizable in the horizontal and vertical histograms as major peaks, where adjacent peaks have the same distance – namely the width and height of each cell. See also Figure 2.

Additionally, each student wrote their student ID (fully numeric) on the first line of the page, above the table. As we know each student's id, this data could in the future be used to develop and test approaches to handwritten digit segmentation, but is currently unused as the number of samples is too small.

2.2 Scanning

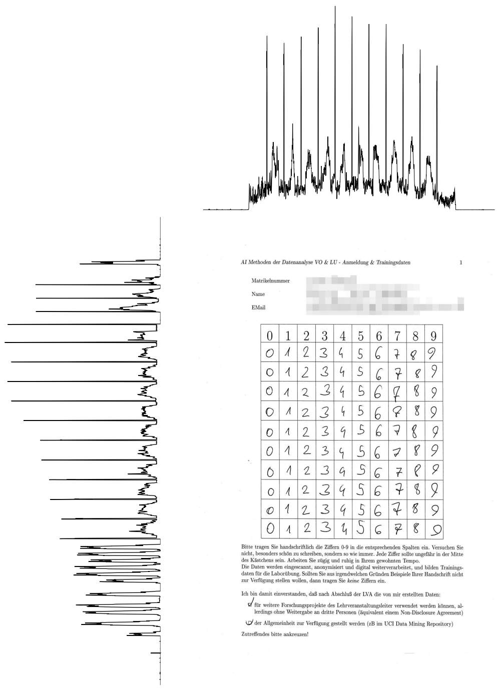
The right side of Figure 1 shows a filled-in input page after scanning with 600 dpi on a Canon IR 2200 with automatic document feeder. The output was 4950x6996 pixels in size. The Canon generally does a very good job to map grayscale values to black-and-white – one student writing with a pencil and another who wrote with a red pen were both mapped to black-and-white reasonably well, and the classification accuracy does not suffer unduly.

2.3 Cell Segmentation

Although we had full control over the input page, segmentation was still somewhat tricky. Figure 2 shows the horizontal and vertical histograms extracted from a sample scanned input page. For preprocessing we used the open-source ImageMagick C/C++ library (`www.imagemagick.org`) plus several hundred lines of C code.

The horizontal histogram was less problematic to analyze. The top 11 peaks usually corresponded to the

Figure 2: Scanned input page with horizontal and vertical histograms



eleven vertical lines of the table. However, in some cases one of the top 11 peaks included at least one set of digits which were exceptionally well aligned along the vertical axis. Since these were always near a larger peak, we chose to ignore peaks which were less than about half the cell size (in pixels) away from a larger peak. This parameter was called `MERGE_INTERVAL` and set to 150, as the cell size of the table is around 300x300 pixels at 600dpi. All parameter values are in pixels and need to be adjusted for different resolutions and/or page sizes.

For the vertical histogram, we used the same approach with the same value of `MERGE_INTERVAL`. Unfortunately there were a lot of high peaks corresponding to the base lines of the machine-printed text on the top and the bottom of the page, which were regularily confused with lines from the table. This should not come as a surprise, since yertical histograms are used to determine the position of lines of text in OCR, and there are about a dozen lines of machine-printed text on the page. Therefore, we had to ignore intervals at the top and bottom of the page (i.e. 0 to `EXCLUDE_VERTICAL_TO` (1500), and `EXCLUDE_VERTICAL_FROM` (4850) to 6996 pixels).

The initial analysis from the histograms gives us a set of coordinates that determine the positions and sizes for each cell. However, as we can see from Figure 3 (left side) the lines are not perfectly axis-parallel horizontal or vertical lines as the automatic page feeder does not align the page perfectly. Additionally, each page was aligned differently. As we already had a good approximation of line positions, we chose to refine them as follows.

For each horizontal line in the table (previously estimated by the vertical histogram), we searched for the true line along each point in the middle between two adjacent crossing vertical lines. The search was done in the vertical axis-parallel direction (± `SEARCH_ARRAYLINE_INTERVAL` (20), i.e. between 20 pixels below and above the midpoint). A line was defined as the nearest sequence of more than `IGNORE_SIZE` (2) black pixels within the searched interval, and the central pixel was used as the true line position in that case. This gives at most 10 samples (x,y) for the true line position. In reality, the number was smaller as not in all cases the line could be found[2]. To exactly define one line in 2D, two sample points would be sufficent. To get a very robust approximation of the true line position, we applied linear regression to all found position samples. The same approach was used for the vertical lines, i.e. using the midpoint between two adjacent crossing horizontal lines, and searching in horizontal axis-parallel direction.

Figure 3 shows the improvement of using just horizontal and vertical histograms versus refinement with our local search approach and linear regression. The refinement procedure outlined here captures the exact table positions almost perfectly for all our scanned input pages. However, in one of our scanned pages a scanning error in the form of a 1-point thin horizontal line somewhere within the table region led to a mistake in the positioning of a single horizontal line and five samples in the corresponding columns were made unusable. Increasing `SEARCH_ARRAYLINE_INTERVAL` solves this, but introduces more serious errors. Further work is needed to recover from these kinds of errors, which seem to appear in about 2% of all scanned pages.

2.4 Digit Segmentation

What remains to be done is to extract one sample digit from each cell that is defined by the vertical and horizontal lines of the table, which have already been reconstructed in the last step. We can no longer assume the lines to be perfectly horizontal and vertical. However, this is desirable for an efficient algorithm, so as first step we determined the largest axis-parallel rectangle which fits into the parallelogram given by the four crossing-points of adjacent horizontal and vertical lines defining each cell.

This rectangle is in an all black area, i.e. almost all of the pixels along the border of this rectangle should be black. To get into the inner area of the cell, the size of the rectangle is reduced pixel by pixel symmetrically around its center until less than one fiftieth (`THRESHOLD_PERCENT_INVERSE` (50)), i.e. 2%, of its border pixels are black. This yields the largest rectangle within the cell, i.e. on the outer border of the white area in which we expect the digit to be.

To enclose the digit, the size of the rectangle is reduced pixel by pixel, for each side separately. For each side, the size is reduced until at least one fiftieth (`THRESHOLD_PERCENT_INVERSE_2` (50)), i.e. 2%, of the border pixels from that side are black for five consecutive steps. The size is afterwards increased by the same amount so that no part of the digit is lost. This was found to be a very efficient way to handle speck noise, provided it is not connected with the digit. The final result is the smallest rectangle that contains the digit, so segmentation is complete.

Five samples could not be recovered in this segmentation step, as the cells contained parts from several digits due to overlap between the contents of different cells. All in all the segmentation was successful: 99.71% of the contributed digits were successfuly segmented by our algorithm, and half of the unusable samples came from a single scanning error which could easily be corrected by rescanning the incorrect page (see last section), or by further improvements to cell segmentation.

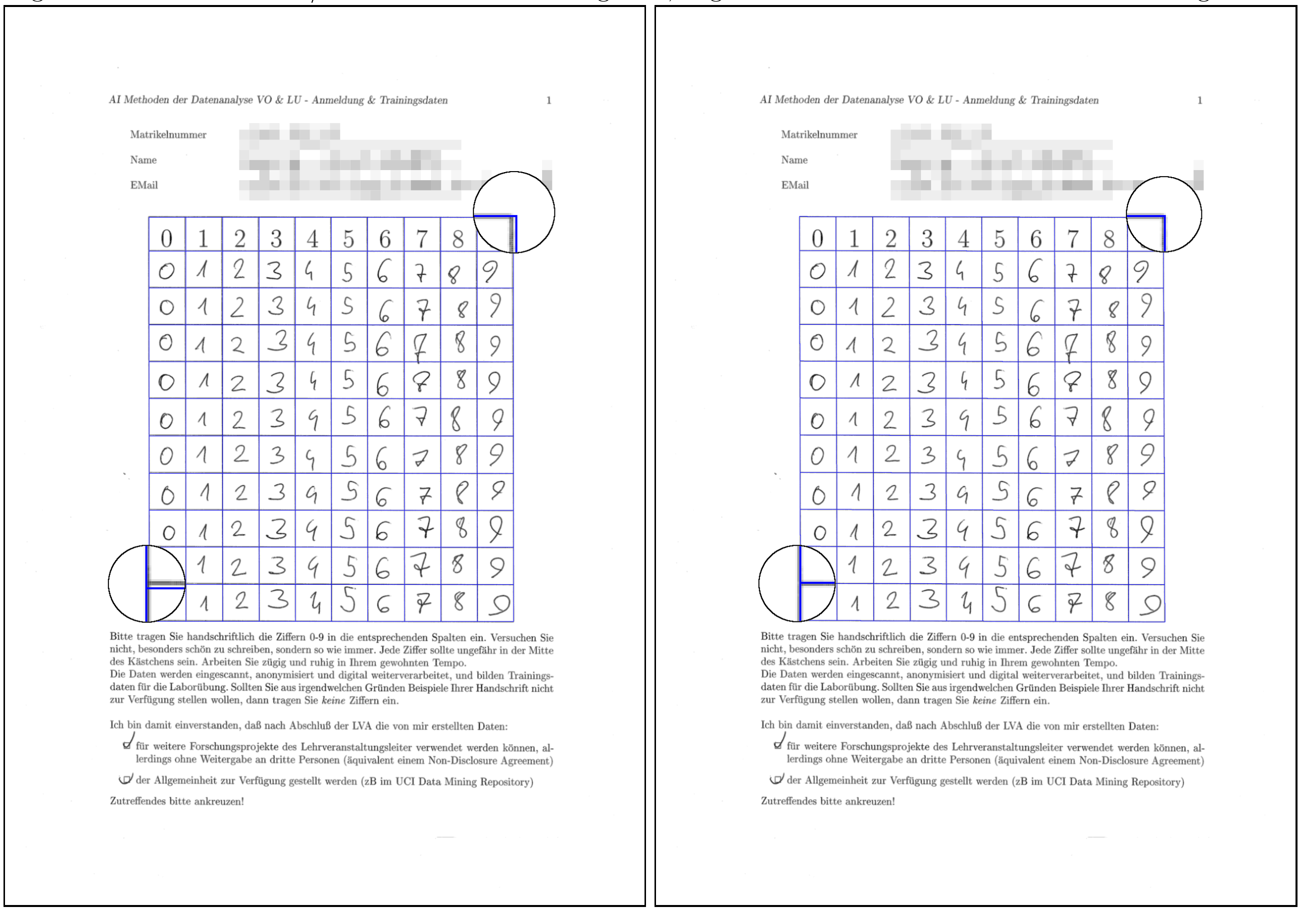
2.5 Resizing and Blurring

There are two different ways to normalize the size of a digit: One, we may resize the digit by leaving its proportions intact. We called this 1:1 scaling. While this gives a natural-looking digit, a tall digit will have many white pixels just because of its proportion, and these do not contribute much to the recognition process. Two, we may resize the digit arbitrarily to use the available space as fully as possible, and have the learning algorithms cope with the resulting distortions. We called the latter approach arbitrary scaling.

Downscaling always means a loss of information. If we were to downscale to a black-and-width bitmap, the loss of information would be quite huge. Therefore, downscaling is usually done to a grayscale bitmap where the gray values still retain some information of the distribution of black pixels within each downscaled pixel. We used the Mitchell filter from ImageMagick for downscaling, which was default for that operation. It has a parameter to control blurring. A setting of 0.5

[2]In that case, it either was connected to a digit or broken by white speck noise at the midpoint.

Figure 3: Left: Horizontal/vertical lines from histograms, Right: Refined via local search and linear regression.



AI Methoden der Datenanalyse VO & LU - Anmeldung & Trainingsdaten 1

Matrikelnummer	[redacted]
Name	[redacted]
EMail	[redacted]

0	1	2	3	4	5	6	7	8	
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
	1	2	3	4	5	6	7	8	9
	1	2	3	4	5	6	7	8	9

Bitte tragen Sie handschriftlich die Ziffern 0-9 in die entsprechenden Spalten ein. Versuchen Sie nicht, besonders schön zu schreiben, sondern so wie immer. Jede Ziffer sollte ungefähr in der Mitte des Kästchens sein. Arbeiten Sie zügig und ruhig in Ihrem gewohnten Tempo.
Die Daten werden eingescannt, anonymisiert und digital weiterverarbeitet, und bilden Trainingsdaten für die Laborübung. Sollten Sie aus irgendwelchen Gründen Beispiele Ihrer Handschrift nicht zur Verfügung stellen wollen, dann tragen Sie *keine* Ziffern ein.

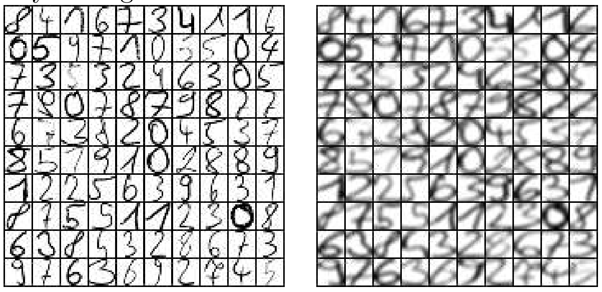
Ich bin damit einverstanden, daß nach Abschluß der LVA die von mir erstellten Daten:

[✓] für weitere Forschungsprojekte des Lehrveranstaltungsleiter verwendet werden können, allerdings ohne Weitergabe an dritte Personen (äquivalent einem Non-Disclosure Agreement)

[✓] der Allgemeinheit zur Verfügung gestellt werden (zB im UCI Data Mining Repository)

Zutreffendes bitte ankreuzen!

Figure 4: Left: b=0.5, 1:1 scaling; Right: b=2.5, arbitrary scaling



gives natural-looking digits with sharp edges. A setting of 2.5 with arbitrary scaling gives blurred-looking digits but improves classification performance. Figure 4 shows the former setting on the left, and the latter on the right.

All in all, a setting of blur=2.5 for the Mitchell filter and arbitrary scaling to 16x16 pixels were found to give the best results with several different classifiers. In fact, for arbitrary scaling to 16x16 pixels, the test set error as a function of blur was found to be approximately quadratic with a global minimum at blur=2.5. We have also prepared the digits in MNIST and USPS-compatible format for the expertise transferral experiments.

2.6 MNIST and USPS

The US Postal (USPS) handwritten digit dataset is derived from a project on recognizing handwritten digits on envelopes [1]. The digits were downscaled to 16x16 pixels and 1:1 scaled. The training set has 7291 samples, and the test set has 2007 samples. We have output our digits in a similar format by using 1:1 scaling plus Mitchell filter downsampling with blur=0.5. This yields the most similar grayscale histogram to USPS. Figure 5 shows samples from USPS and from our reformatted dataset.

The MNIST dataset, one of the most famous in digit recognition, is derived from the NIST dataset, and has been created by Yann LeCun [2]. The digits from NIST were downscaled to 20x20 pixels and centered in a 28x28 pixel bitmap by putting center-of-gravity of the black pixels in the center of the bitmap. It

Figure 5: Left: USPS, Right: digits (reformatted)

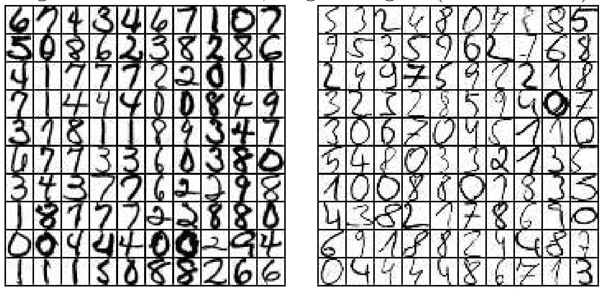
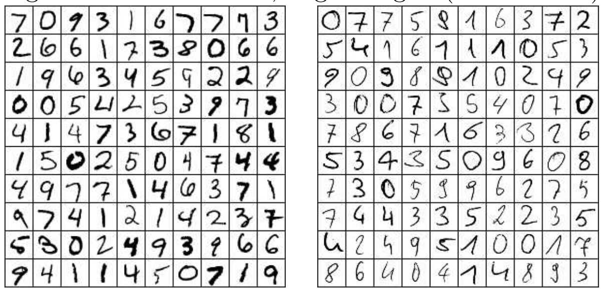


Figure 6: Left: MNIST, Right: digits (reformatted)



has 60,000 training and 10,000 test samples. We have output our digits in this format with Mitchell filter downsampling and again blur=0.5. Center-of-gravity was computed before downsampling and scaled accordingly. Figure 6 shows samples from MNIST and from our reformatted dataset. You can see that MNIST has some segmentation errors (e.g. column 4, row 4 is a badly segmented four), possibly as much as 1%[3] – for our dataset, we checked each sample manually for segmentation errors, so there should be none.

In both cases, we normalized the grayscale values to the training data – from 0 (white) to 255 (black) for MNIST, and -1 (white) to +1 (black) for USPS. For USPS, we adapted the blur parameter of the Mitchell downsampling filter to get the most similar grayscale histogram. For MNIST, we used the same value.

3 Results

In this section we note some preliminary results on our digits dataset, MNIST and USPS and discuss qualitative differences between the datasets. A support vector machine (SVM) classifier implementation from WEKA[4], SMO [4] was chosen since SVM classifiers usually perform well in pattern recognition tasks even with simple representations (e.g. each pixel as a feature as in our case). We found in earlier experiments that a polynomial kernel with exponent 5 (-E 5), lambda cost parameter 10 (-C 10) and feature space normalization (-F) performed well and therefore used this classifier for all of the following experiments.

3.1 digits

For these experiments, we chose to use all 44 user's samples. We randomly divided the users into two different groups of 22 students, and used one of them for training and the other for testing. Initial experiments had indicated that arbitrary scaling and a blur setting of 2.5 for the Mitchell downsampling filter should perform well. We chose to downsample to 16x16 pixels. The settings achieved an error rate of 6.10% on the test set. Using 1:1 scaling with blur=0.5 (similar to USPS) gave a much worse error rate of 12.20%. Gaussian blurring with a 5x5 filter improved the latter to 8.05%. We conclude that a higher amount of blurring is beneficial for this dataset, independent of the specific algorithm used.

3.2 USPS

For the US Postal dataset, we can already assume that the writers for training and test set are disjunct, so we ran the given training and test sets as is.

The test set error was 4.29% and thus a bit better than the digit dataset with similar settings. This may be explained by the higher amount of training data (7291 vs. 2192 examples). Applying a gaussian 5x5 filter on training and test data prior to training resulted in a slightly worse error of 4.63%. We conclude that a higher amount of blurring is not beneficial for this dataset.

3.3 MNIST

For the MNIST dataset, we can also assume that the writers for training and test set are disjunct, so we ran the given training and test sets as is.

Th test set error was 1.27% and thus much better than either USPS or digits. Combined with the other results, this suggests that the SVM classifier performs better with more training data, so the higher amount of training data (60,000 vs. 7291 (USPS) and 2192 (digits)) may explain this behaviour. Applying a gaussian 5x5 filter prior to training again yielded a slightly worse error of 1.28%.

These results are slightly worse than those reported in [3] for LeNet-4 (1.1%) and LeNet-5 (0.95%), but better than the previous best result for polynomial SVM also reported there. This is probably due to a combination of feature space normalization and pairwise classifica-

3.4 We have created a new handwritten digit recognition dataset from scratch, with the help of students for the AI Methods of Data Analysis class of 2005. While the collected dataset is still small, we have noted some surprising results concerning the generality of handwritten digit recognizers.