International Scholarly Research Network
ISRN Machine Vision
Volume 2012, Article ID 834127, 10 pages
doi:10.5402/2012/834127

Research Article

On the Brittleness of Handwritten Digit Recognition Models

Alexander K. Seewald

Seewald Solutions, Leitermayergasse 33, 1180 Vienna, Austria

Correspondence should be addressed to Alexander K. Seewald, alex@seewald.at

Received 19 July 2011; Accepted 7 September 2011

Academic Editor: A. Torsello

Handwritten digit recognition is an important benchmark task in computer vision. Learning algorithms and feature representations which offer excellent performance for this task have been known for some time. Here, we focus on two major practical considerations: the relationship between the the amount of training data and error rate (corresponding to the effort to collect training data to build a model with a given maximum error rate) and the transferability of models' expertise between different datasets (corresponding to the usefulness for general handwritten digit recognition). While the relationship between amount of training data and error rate is very stable and to some extent independent of the specific dataset used—only the classifier and feature representation have significant effect—it has proven to be impossible to transfer low error rates on one or two pooled datasets to similarly low error rates on another dataset. We have called this weakness *brittleness*, inspired by an old Artificial Intelligence term that means the same thing. This weakness may be a general weakness of trained image classification systems.

1. Introduction

Intelligent image analysis is an interesting research area in Artificial Intelligence and also important to a variety of current open research problems. Handwritten digits recognition is a well-researched subarea within the field, which is concerned with learning models to distinguish presegmented handwritten digits. The application of machine learning techniques over the last decade has proven successful in building systems which are competitive to human performance and which perform far better than manually written classical AI systems used in the beginnings of optical character recognition technology. However, not all aspects of such models have been previously investigated.

Here, we systematically investigate two new aspects of such systems.

(i) *Essential training set size*, that is, the relation between training set size and accuracy/error rate so as to determine the number of labeled training samples that are essential for a given performance level. Creating labeled training samples is costly and we are generally interested in algorithms which yield acceptable performance with the fewest number of labeled training samples.

(ii) *Dataset-Independence*, that is, how well models trained on one sample dataset for handwritten digit recognition perform on other sample datasets for handwritten digit recognition after comprehensive normalization between the datasets. Models *should* be robust to small changes in preprocessing and data collection, but this has not been tested before.[1]

For the first aspect, we have found that all three datasets considered here give similar performance relative to absolute training set size. This indicates that the quality of input data is similar for these three datasets. A relatively small number of high-quality samples is already sufficient for acceptable performance. Accuracy as a function of absolute training set size follows a smooth asymptotic behavior, in which low error rates (below 10%) are reached quite fast, but very low error rates are reached only after sustained effort.

For the second aspect, we were surprised to notice that none of the considered learning systems were able to transfer their expertise to other datasets. In fact, the performance on

other datasets was always significantly worse or unacceptably high.

This may point to a general weakness of present intelligent image analysis systems. We have named this weakness *brittleness* in AI terminology, which seems to us the most appropriate term.

Small differences in preprocessing methods which have not been documented in sufficient detail (except perhaps in [1]) may be responsible for this effect. Another explanation might be that idiosyncrasies of the specific dataset used for training are learned as well and hamper the generalization ability of the underlying learning algorithm. This effect is observed independently of learning algorithm or feature representation.

A more detailed documentation of preprocessing methods and classification systems in the form of Open Source code would be needed to investigate how to build more robust learning system for this domain, and possibly for intelligent image analysis systems in general.

2. Related Research

Reference [1] described the full preprocessing of dataset DIGITS and noted early results of the experiments within this paper.

Reference [2] also reported extensive experiments, noting that simpler convolutional neural networks than the one used by [3] might suffice. However, since they did not contribute code for their learning and preprocessing systems, we chose the implementation from [4] which is freely available for research purposes.

Reference [5] reported a comprehensive benchmark of handwritten digit recognition with several state of the art approaches, datasets, and feature representations. However, they analyzed neither the relationship of training set size versus accuracy/error nor the dataset-independence of the trained models, which are our two main contributions.

Reference [3] introduced convolutional neural networks into handwritten digit recognition research and demonstrated a system (LeNet-5) which can still be considered state of the art.

Reference [6] notes open research questions and also proposes contributing open source software for standard preprocessing methods. However, neither do they offer their own extensive code as open-source, nor do they note the dataset-independence issue we have noted here, although it is clearly a major issue in the practical application of such systems.

3. Experimental Setup

3.1. Datasets. We used two well-known (USPS, MNIST) and one relatively unknown (DIGITS) dataset for handwritten digit recognition. The relatively unknown dataset was created by ourselves, so we had complete control and documentation over all preprocessing steps, documented in [1].

The US Postal (USPS) handwritten digit dataset is derived from a project on recognizing handwritten digits on envelopes [7, 8]. The digits were downscaled to 16×16 pixels and scaled without distortion (i.e., retaining the aspect ratio; 1 : 1 scaling). The training set has 7291 samples, and the test set has 2007 samples. Figure 1(a) shows samples from USPS. For the experiments in Section 4.1, we used USPS as is; for Section 4.2, we used a reformatted version which mimics MNIST preprocessing by rescaling and shifting center of gravity.

The MNIST dataset, one of the most famous in digit recognition, is derived from the NIST dataseta and has been created by LeCun et al. [3]. According to this paper, the digits from NIST were downscaled to 20×20 pixels and centered in a 28×28 pixel bitmap by putting center of gravity of the black pixels in the center of the bitmap. It has 60,000 training and 10,000 test samples. Figure 1(b) shows samples from MNIST. It can be seen that MNIST has about 1% segmentation errors (e.g., column 4, row 4 is a badly segmented four)[2]. However, this cannot explain the performance differences between the systems reported in Section 4.2 as those comparisons not including MNIST fare as poorly as those that do.

The DIGITS dataset was created in 2005, based on samples from students of a lecture given by the author. Each student contributed 100 samples, equally distributed among the digits from 0 to 9. The complete preprocessing is described in [1]. Students were given the choice to withhold the samples, to allow usage of the samples by the author of this paper and to allow usage by anyone (i.e., public domain). 37 students opted for the latter option. These were randomly distributed into training and test (19 training, 17 test), yielding 1,893 training and 1,796 test samples after minor cleanup. The dataset can be freely downloaded from the authors website, http://alex.seewald.at/.

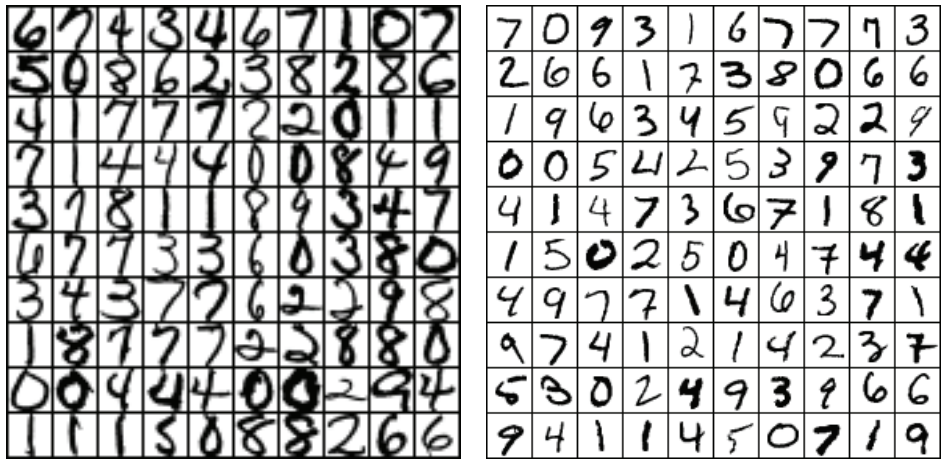
Figure 2(a) shows the digit dataset with preprocessing optimized to improve classification accuracy (viz., arbitrary aspect ratio—i.e., the digit is scaled to fill available space; blurring with blur = 2.5, no deslanting[3]), and Figure 2(b) shows the same samples reformatted in a format easier to recognize for humans (1 : 1 scaling, blur = 0.5). Mitchell filter downsampling with integrated Gaussian blurring was used in both cases.

3.2. Feature Sets. We considered two feature sets.

(i) *Pixel-based*, that is, the pixel grayscale values in eight bit precision (0 = white, 255 = black to be compatible with MNIST) in the order from top left to bottom right (784 numeric features).

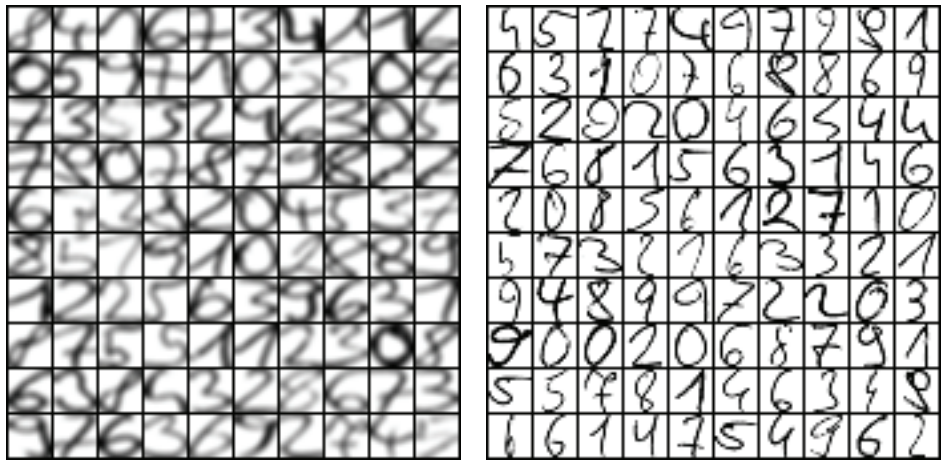
(ii) *Gradient-based*, that is, a 200-dimensional numeric feature vector encoding eight direction-specific 5×5 gradient images (same feature order as for above images). This was one of three top-performing representations in [5] and is called *e-grg* in their paper. We reimplemented *e-grg* from scratch and validated on MNIST train/test with 1-NN, yielding a test error rate of 1.29% versus the 1.35% reported in their paper with identical preprocessing.

Additional feature sets could have been considered, but we felt that these two sets would be sufficient for the purpose of this paper.



(a) (b)

Figure 1: (a) USPS, (b) MNIST.



(a) (b)

Figure 2: (a) digits ($b = 2.5$, arb.scal.), (b) digits ($b = 0.5$, 1 : 1 scaling).

3.3. Classifiers. We considered a variety of classifiers in three groups. All of the classifiers except convNN were taken from WEKA [9].

3.3.1. Instance-Based Learning. Initial experiments indicated that—probably because of the high number of classes—a simple k-NN nearest neighbor classifier with $k = 1$ performed best. For the gradient-based representation, only euclidean distance was considered. For the pixel-based representation, we additionally considered the well-known template matching method *normalized correlation coefficient*, and tangent distance, which can cope with small template distortions. We used the GPL implementation by [10] and validated on their freely available samples. Both distance measures were reimplemented in Java for use with WEKA.

3.3.2. Support Vector Machines. We also considered polynomial and RBF kernel support vector machines (SVM, see, e.g., [11]) classifiers, since these classifiers were two of the three best-performing methods according to [5].

From earlier experiments, we already knew optimized parameter settings for this classifiers on the DIGITS training set. We extended these experiments and determined similar optimized parameter settings for *e-grg* on the same dataset. These were used to train all other datasets. A slight positive bias for DIGITS and low-bias learning algorithms (such as RBF and polynomial SVM) may be present. We also considered a linear kernel SVM with default settings ($C = 1$).

3.3.3. Convolutional Networks. The de facto standard for handwritten digit recognition is the convolutional network (convNN, e.g., leNet-5) by [3]. As WEKA does not include a convolutional network learning algorithm, and there is also none available by the author of the mentioned paper, we had to resort to using the nonscriptable version of the training algorithm by [4]. A set of manual experiments was

done to validate the implementation versus MNIST, and we did extensive experiments for Section 4.2. Due to the non-scriptable version, we could not run learning curves on this system. The CPU time would in any case have been exorbitantly high at around 1,500 hours (2 CPU months) for learning curves on all three datasets.

Training was done using the following parameters, as these proved to give the best results on the original MNIST training/test sets (according to [4], validated by us),

(i) initial learning rate: 0.001,

(ii) minimum learning rate: 0.00005,

(iii) rate of decay for learning rate, applied every two epochs until minimum learning rate was reached: 0.79418335,

(iv) run with elastically deformed training inputs for at least 52 epochs,

(v) run with non-deformed training input for exactly 5 polishing epochs with a learning rate of 0.0001.

4. Results

In this section, we will show the full results from our experiments.

4.1. Essential Training Set Size. This section is concerned with analyzing the relationship between training set size and recognition accuracy, depending on dataset and learning algorithm.

4.1.1. Pixel-Based Features. Figure 3 shows the results for instance-based learning, and Figure 4 shows the results for SVM learning. Both experiments were run on pixel-based features from the three datasets. The test set was fixed while the training set was downsampled to the absolute number of training examples shown as X. The Y-axis shows the accuracy on the test set. Additionally, downsampling was randomized ten times, and the standard deviation over these ten runs is shown as error bars.

When using instance-based learning, the three datasets perform remarkably similar. Only for the tangent distance variant, DIGITS performs noticeably worse. We presume this is due to the collection of this dataset, where digits had to be written into a regular grid, which forced a very uniform orientation. As tangent distance was constructed to compensate for non-uniform orientations—wihch is not needed here—the additional degrees-of-freedom of this method may have led to overfitting on this dataset, resulting in inferior performance.

When using SVM learning, the picture is similar, albeit less clear. Only for the polynomial variant do we see very similar behavior on the three datasets. For the other two variants, some differences appear. Especially, MNIST performs very badly with the RBF kernel variant. We presume that this is due to the high number of variance in MNIST, and the higher number of parameters for the RBF kernel, such that the amount of training data is no longer sufficient for stable

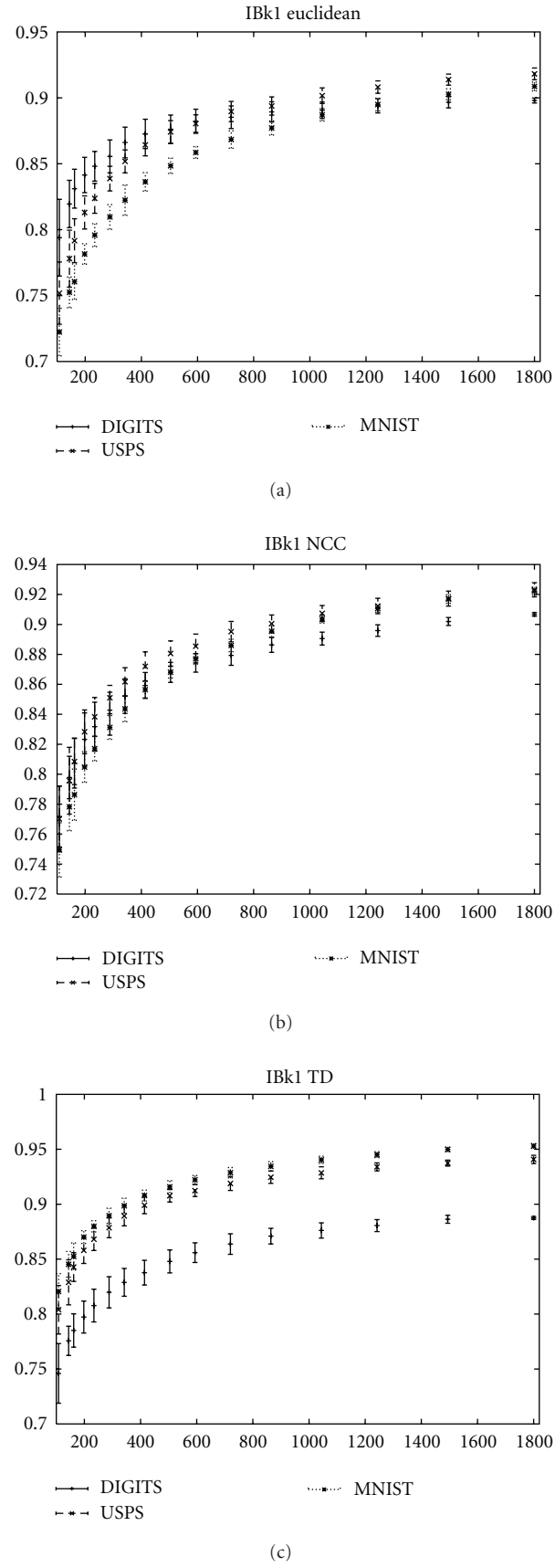
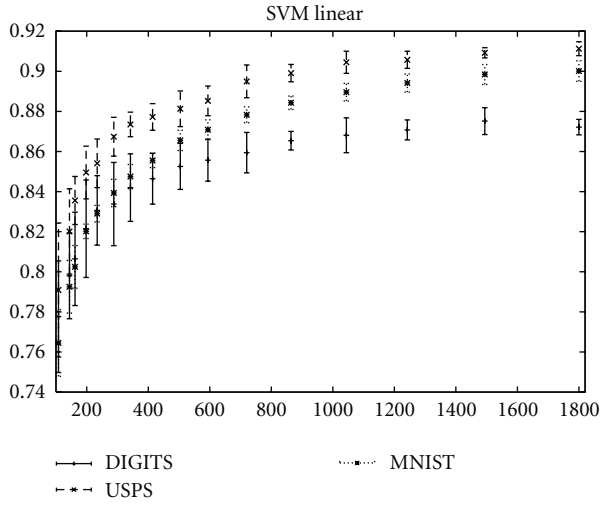
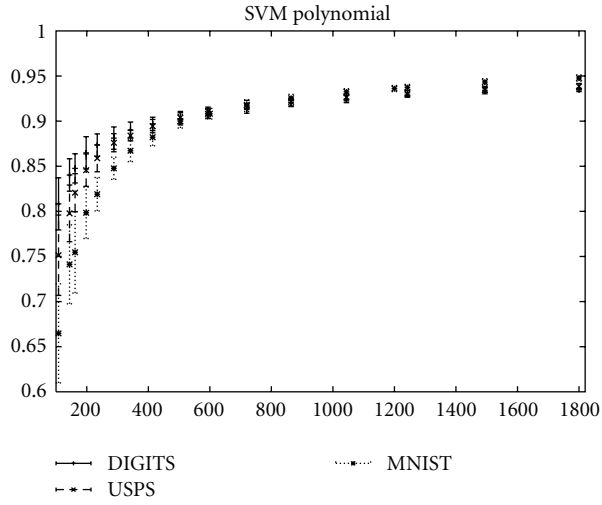


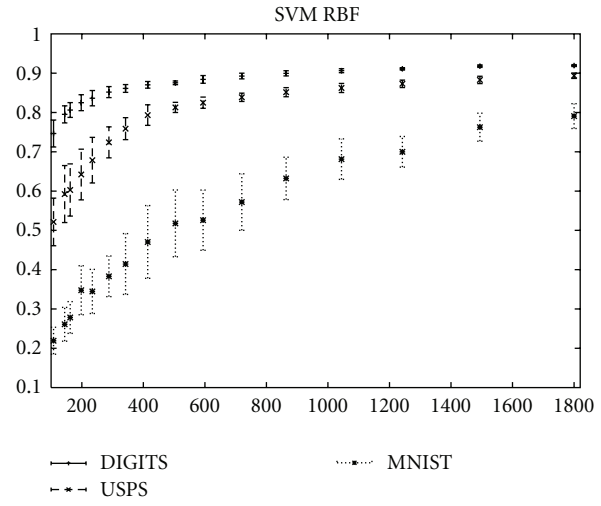
Figure 3: Abs. training set size versus test set accuracy for pixel-based features on MNIST, USPS, and DIGITS (IBk variants).



(a)



(b)



(c)

Figure 4: Abs. training set size versus test set accuracy for pixel-based features on MNIST, USPS, and DIGITS (SVM variants).

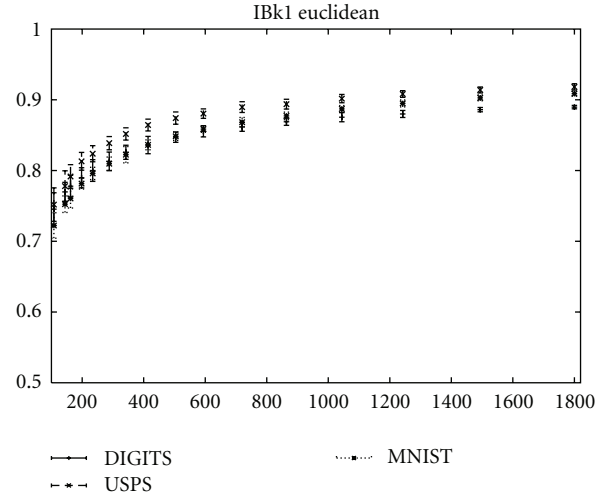


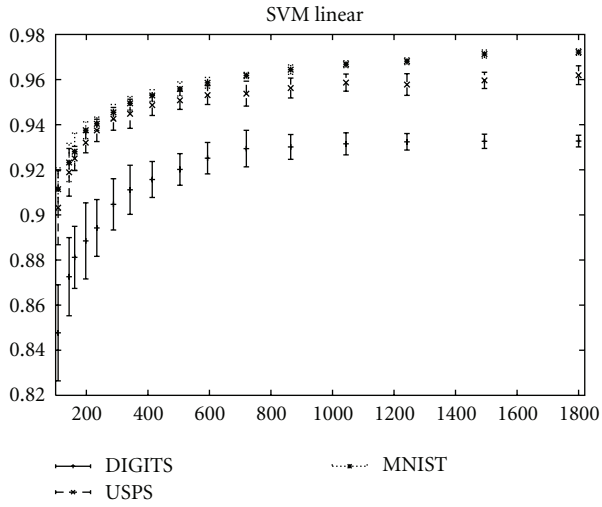
Figure 5: Abs. training set size versus test set accuracy for gradient features on MNIST, USPS, and DIGITS (IBk variants).

parameter estimation. Also, parameters were optimized for the DIGITS training set, and this may have led to some overfitting[4]. The polynomial kernel has the best accuracy here, closely followed by the linear kernel. Because of the nonscriptable version for convNN and its long training times, we could not test it here.

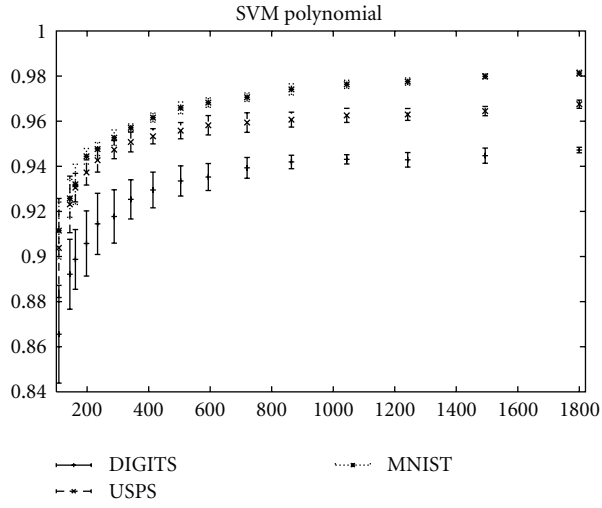
4.1.2. Gradient-Based Features. In a second step, we analyzed gradient-based features. Since pixel-based features are a very imprecise way to encode information about handwritten digits, we chose to use direction-specific feature maps which were previously found to work best (see Section 3.2).

Figure 5 shows the results with instance-based learning. Here, we only used IBk with one nearest neighbor as the other two distance measures are inappropriate for non-pixel-based data. We again observe similar behavior for all datasets, at a slightly higher level of accuracy than for the pixel-based features and the same learning algorithm. Clearly, adding relevant background knowledge in the form of tangent distance or normalized correlation distance measures is more helpful to improve IBk than this alternative feature representation.

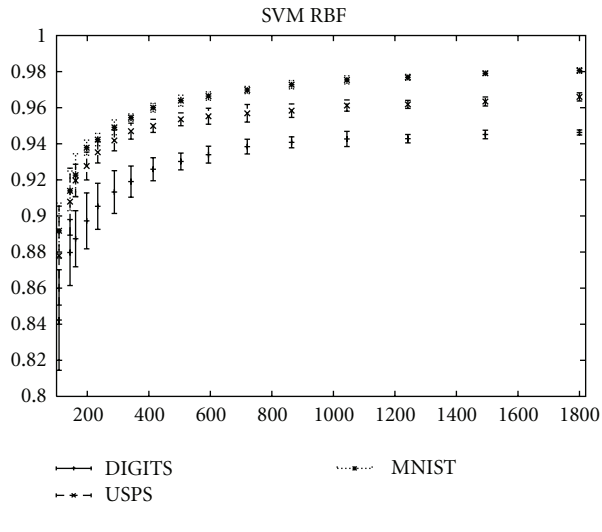
Figure 6 shows the results with SVM learning. SVM results are clearly improved throughout over all datasets. Also, we see a clear ordering for the right half of each figure: MNIST performs better than USPS, and USPS performs better than DIGITS. For SVM learning, the alternative feature set representation improves the results quite distinctly. Note also that just for 1,800 samples, we can have an error of 2% on the MNIST testset, which is quite good considering that the best published results are at around 0.5% and use orders of magnitude more training data. A linear SVM with unprecedented processing speed is only slightly worse. It might be that the higher accuracy of these systems has enabled us to see the *hardness* of the dataset—like the harder part of MNIST, DIGITS consisted of data contributed by (university) students. USPS would then be between both datasets in terms of sample complexity.



(a)



(b)



(c)

Figure 6: Abs. training set size versus test set accuracy for gradient features on MNIST, USPS, and DIGITS (SVM variants).

The shape of all learning curves is remarkably similar and might be estimated with just a few data points. They seem to depend on the learning algorithm, the feature representation, and to a lesser extent on the specific dataset in question (e.g., dataset complexity, sample distribution, or other factors).

4.2. Dataset-Independence. All previous results mean nothing if the task has not really been solved. So, as it is clear that—small differences between the datasets notwithstanding—all these datasets deal with the writer-independent recognition of handwritten digits and were created by disjunct sets of writers (which were also properly distributed between training and test set), we estimated the quality of each model by testing it on the other datasets. First, we converted both DIGITS and USPS into MNIST format by centering each digit in a 28×28 image, equalizing the histograms via nonlinear gamma correction.[5] For DIGITS, we estimated center-of-gravity from the original 300 dpi black-and-white images; for USPS, we estimated c-o-g from the grayscale images after thresholding at 50% (128). In both cases, 1 : 1 scaling was used (i.e., the aspect ratio was retained) Figure 7.

First, we trained on each training set in turn and tested on the other two sets. Note that the training and test sets are of different size, so for example, MNIST builds a model from 60,000 samples while DIGITS just builds a model from about 1,800. According to the results from the previous section, we would expect a range of about one order of magnitude (best versus worst) in error rates on the test set corresponding to the training set, with MNIST better than USPS and USPS better than DIGITS. This is exactly what we observed. Surprisingly, the performance on the other test sets is much worse.

This time, we also tested LeCun's original convolutional neural network model as reconstructed by [4]. The results from training on the complete training set with the training method described in part in [3] yielded error rates of 0.74% on MNIST-test, 1.72% on the full USPS dataset (i.e., train and test combined), and 8.74% on the full DIGITS dataset. The error rate is significantly higher for both datasets: about twice as high on USPS and more than ten times higher on DIGITS[6]. We also trained convNN with the same method and similar settings both for DIGITS and USPS.

Table 1 shows the results for pixel-based features, and Table 2 shows the results for gradient-based features. Both are showing error rates estimated on the respective test sets and the average error of the two other sets divided by the error on the test set corresponding to the training set the model was trained on. What is immediately apparent is that *no* combination of learning algorithm, feature representation and dataset to train on was able to transfer the usually good results of its own test set to the other test sets without a significant loss in accuracy. A small ratio such as less than 2.0 is only obtainable for unacceptably high error rates. The lowest error rate on any test set of 3.48% is achieved for convNN trained on MNIST and tested on USPS. This still has more than twice this error rate (8.24%) on DIGITS, and about five times the error rate on the MNIST test set (0.74%).

(a) (b)

Figure 7: (a) USPS, (b) DIGITS, both reformatted to MNIST format.

Table 1: Dataset independence for pixel-based features, each dataset separately.

Classifier	Trained on	Tested on			Avg. error versus own testset
		MNIST	DIGITS	USPS	
IBk1 euclidean	MNIST	3.09	19.21	17.49	5.94x
IBk1 euclidean	DIGITS	36.22	16.59	52.72	2.68x
IBk1 euclidean	USPS	28.41	55.01	5.33	7.83x
IBk1 NCC	MNIST	2.83	17.65	13.70	5.54x
IBk1 NCC	DIGITS	32.42	14.14	44.59	2.72x
IBk1 NCC	USPS	26.06	51.17	4.58	8.43x
IBk1 TD	MNIST	1.51	13.53	5.63	6.34x
IBk1 TD	DIGITS	25.88	10.02	37.77	3.18x
IBk1 TD	USPS	10.51	36.47	3.64	6.45x
SVM linear	MNIST	6.83	34.97	16.24	3.75x
SVM linear	DIGITS	31.57	16.09	45.54	2.40x
SVM linear	USPS	40.64	63.25	6.53	7.95x
SVM polynomial	MNIST	1.27	16.20	11.56	10.93x
SVM polynomial	DIGITS	30.05	11.47	47.68	3.39x
SVM polynomial	USPS	44.78	74.33	4.43	13.44x
SVM RBF	MNIST	4.31	53.34	20.78	8.60x
SVM RBF	DIGITS	51.50	33.74	60.09	1.65x
SVM RBF	USPS	81.05	89.98	7.37	11.60x
convNN	MNIST	0.74	8.24	3.48	7.92x
convNN	DIGITS	21.43	5.73	30.0	4.49x
convNN	USPS	4.25	27.56	3.08	5.16x

Second, we chose to also test combining two datasets and testing on the remaining dataset. We downsampled the larger training dataset to the size of the smaller training set and combined them, shuffling the results to prevent order effects. The same test sets as previously were used. This time, we computed the error of the remaining completely unseen dataset divided by the average of errors for the two seen datasets (i.e., those whose training set was part of the dataset pool). Again, convNN was trained on the same data.

Tables 3 and 4 show the results for pixel-based and gradient-based features. Again, we see that the error of the datasets who were part of the training is usually much higher than the one on the completely unseen dataset. This is not true for IBk with tangent-distance, where both for MNIST-DIGITS and USPS-DIGITS the error on DIGITS testset is higher than the one on the completely unseen dataset. This might be due to the small size of the DIGITS test set, which increases the variance of error estimates computed on this

Table 2: Dataset independence for gradient-based features, each dataset separately.

Classifier	Trained on	Tested on			Avg. error versus own testset
		MNIST	DIGITS	USPS	
IBk1 euclidean	MNIST	1.29	12.08	5.98	7.00x
IBk1 euclidean	DIGITS	21.91	7.29	37.62	4.08x
IBk1 euclidean	USPS	10.30	33.07	3.49	6.21x
SVM linear	MNIST	1.34	12.92	5.63	6.92x
SVM linear	DIGITS	19.76	5.12	30.54	4.91x
SVM linear	USPS	14.62	39.37	3.34	8.08x
SVM polynomial	MNIST	0.47	8.07	4.43	13.30x
SVM polynomial	DIGITS	17.81	3.67	24.86	5.81x
SVM polynomial	USPS	14.68	39.03	2.79	9.63x
SVM RBF	MNIST	0.57	8.30	4.28	11.04x
SVM RBF	DIGITS	17.75	4.06	25.46	5.32x
SVM RBF	USPS	14.89	40.03	2.79	9.84x

Table 3: Dataset independence for pixel-based features, two datasets combined.

Classifier	Trained on	Tested on			Error versus avg. of own testsets
		MNIST	DIGITS	USPS	
IBk1 eucl.	MNIST-DIGITS	9.23	18.32	25.56	1.86x
IBk1 eucl.	MNIST-USPS	5.97	25.89	5.28	4.60x
IBk1 eucl.	USPS-DIGITS	22.27	22.05	6.98	1.53x
IBk1 NCC	MNIST-DIGITS	7.81	13.64	21.67	2.02x
IBk1 NCC	MNIST-USPS	4.74	24.39	4.53	5.26x
IBk1 NCC	USPS-DIGITS	18.95	15.53	6.98	1.68x
IBk1 TD	MNIST-DIGITS	4.34	11.41	10.81	1.37x
IBk1 TD	MNIST-USPS	2.65	16.09	3.64	5.12x
IBk1 TD	USPS-DIGITS	9.61	14.09	4.53	1.03x
SVM linear	MNIST-DIGITS	10.62	21.10	21.92	1.38x
SVM linear	MNIST-USPS	12.27	43.10	8.27	4.20x
SVM linear	USPS-DIGITS	20.37	23.83	9.67	1.22x
SVM poly.	MNIST-DIGITS	4.96	8.85	16.54	2.40x
SVM poly.	MNIST-USPS	2.66	22.16	3.89	6.77x
SVM poly.	USPS-DIGITS	14.56	9.97	5.23	1.92x
SVM RBF	MNIST-DIGITS	12.60	34.58	31.19	1.32x
SVM RBF	MNIST-USPS	13.60	71.66	5.63	7.45x
SVM RBF	USPS-DIGITS	39.89	47.38	7.03	1.47x
convNN	MNIST-DIGITS	3.21	4.00	6.57	1.82x
convNN	MNIST-USPS	1.25	11.85	2.74	5.94x
convNN	USPS-DIGITS	7.03	5.79	4.88	1.32x

dataset. The same happens for SVM linear on USPS-DIGITS and SVM RBF on MNIST-DIGITS and USPS-DIGITS.

The better gradient-based feature representation is probably responsible for preventing such outliers in the second tables, as more stable models are learned. This time, SVM polynomial and SVM RBF give the best performance (averaged over the completely unseen test datasets' error rates), closely followed by convNN which uses pixel-based features. Still, this translates to an error of 5.94%, 10.75%, and 5.94% for MNIST, DIGITS, and USPS, which is at least an order of magnitude higher than the best results for handwritten digit recognition (reported on MNIST).

Table 4: Dataset independence for gradient-based features, two datasets combined.

Classifier	Trained on	Tested on			Error versus avg. of own testsets
		MNIST	DIGITS	USPS	
IBk1 eucl.	MNIST-DIGITS	3.58	7.41	9.87	1.80x
IBk1 eucl.	MNIST-USPS	1.98	15.59	3.34	5.86x
IBk1 eucl.	USPS-DIGITS	9.51	7.80	4.14	1.59x
SVM linear	MNIST-DIGITS	3.43	4.79	11.01	2.68x
SVM linear	MNIST-USPS	2.23	15.14	3.39	5.39x
SVM linear	USPS-DIGITS	7.50	6.24	4.38	1.41x
SVM poly.	MNIST-DIGITS	1.84	3.51	6.58	2.46x
SVM poly.	MNIST-USPS	0.91	10.75	2.54	6.23x
SVM poly.	USPS-DIGITS	5.94	4.45	2.94	1.61x
SVM RBF	MNIST-DIGITS	1.88	3.51	6.98	2.59x
SVM RBF	MNIST-USPS	1.00	11.53	2.44	6.70x
SVM RBF	USPS-DIGITS	6.22	4.57	2.84	1.68x

5. Conclusion

We have shown that relatively small amounts of training data are sufficient for state-of-the-art accuracy in handwritten digit recognition, and that the relationship between training set size and accuracy follows a simple asymptotic function.

We have also shown that *none* of the considered learning systems are able to transfer their expertise to other similar handwritten digit recognition datasets. The obtainable error rates are even in the best case far less than what has been reported on single datasets. This indicates that systems learn significant non-task-specific idiosyncrasies of specific datasets or not sufficiently well-documented preprocessing methods and do not yet offer stable dataset-independent performance. Thus present systems can be considered *brittle* in AI terminology, albeit at higher performance level than previous classical AI systems.

More work is needed to determine how to resolve this weakness. As a first step, we propose a more detailed documentation of preprocessing methods and classification systems in the form of Open Source code for further work in the field, a more comprehensive sharing of both data and methods among active research groups, and focussing specific efforts towards building more robust learning systems. An investigation into specific preprocessing choices and their effect on accuracy would be highly desirable and a major step to building systems with truly stable dataset-independent performance.

Acknowledgments

The authors gratefully acknowledge the support of the students of AI Methods of Data Analysis, class 2005. They also acknowledge Mike O'Neill, who has written and validated the non-scriptable convolutional network code, which was used for the convNN experiment (thanks, Mike, you saved us a lot of work.) Finally, special thanks to Julian A. for one important suggestion. This research has been funded by Seewald Solutions.

Endnotes

1. Unfortunately, preprocessing is in most cases not fully documented, which makes such an investigation rather hard. We already did a short analysis on this issue in [1] and were quite disappointed. This paper can be seen as a systematic extension of our previous efforts.
2. These samples actually come from the supposedly cleaner part of the test set by Census employees, SD-3, which indicates that the proportion of segmentation errors for the remaining dataset may even be higher.
3. The digits were entered in a regular grid, and visual inspection showed the slant to be minimal.
4. Note that DIGITS is by far the most accurate algorithm for the RBF kernel variant.
5. USPS already was sufficiently similar, for DIGITS we used $75.6728812921072 * v^{0.589737015486174}$, where v is the raw pixel value and the output is clamped to $[0, 255]$.
6. Although gamma correction results in digits which seem less similar to MNIST than the original set by visual inspection, this proved to reduce the error rate of the original MNIST-trained convolutional neural network by almost a third. On the other hand, although the aspect ratio was lower by 12.5% for DIGITS, additionally compensating this increased the error almost up to the original level. These anecdotes support our upcoming conclusion that performance is very sensitive to a number of factors currently not well understood.

References

[1] A. K. Seewald, "Digits—a dataset for handwritten digit recognition," Österreichisches Forschungsinstitut für Artificial Intelligence TR-2005-27, Tec. Rep., Wien, 2005.

[2] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR '03)*, 2003.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[4] M. O'Neill, *Neural Network for Recognition of Handwritten Digits*, Code Project, 2006.
[5] L. Liu, K. Nakashima, H. Sako, and H. Fujisawa, "Handwritten digit recognition: benchmarking of state-of-the-art techniques," *Pattern Recognition*, vol. 36, no. 10, pp. 2271–2285, 2003.
[6] L. Liu and H. Fujisawa, "Classification and learning for character recognition: comparison of methods and remaining problems," in *Proceedings of the International Workshop on Neural Networks and Learning in Document Analysis and Recognition*, Seoul, Korea, 2005.
[7] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*, Springer, Heidelberg, Germany, 2003.
[8] J. H. Hull, "A database for handwritten text recognition research," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 550–554, 1994.
[9] H. W. Ian and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, San Francisco, Calif, USA, 2nd edition, 2005.
[10] D. Keysers, W. Macherey, H. Ney, and J. Dahmen, "Adaptation in statistical pattern recognition using tangent vectors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 269–274, 2004.
[11] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods: Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds., MIT Press, Cambridge, Mass, USA, 1998.