Fast and Accurate Digit Classification

Subhransu Maji
Jitendra Malik

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2018-159
http://www.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-159.html

November 25, 2018

Fast and Accurate Digit Classification

Subhransu Maji Jitendra Malik
Computer Science Division,
University of California at Berkeley
smaji,malik@cs.berkeley.edu

Abstract

We explore the use of certain image features, blockwise histograms of local orientations, used in many current object recognition algorithms, for the task of handwritten digit recognition. Existing approaches find that polynomial kernel SVMs trained on raw pixels achieve state of the art performance. However such kernel SVM approaches are impractical as they have a huge complexity at runtime. We demonstrate that with improved features a low complexity classifier, in particular an additive-kernel SVM, can achieve state of the art performance. Our approach achieves an error of 0.79% on the MNIST dataset and 3.4% error on the USPS dataset, while running at speeds comparable to the fastest algorithms on these datasets which are based on multilayer neural networks and are significantly faster and easier to train.

1 Introduction

Handwritten digit recognition has been a fertile ground for exploring several learning techniques ranging from automatically learning feature representations [22, 14], learning classifiers invariant to distortions [7], matching and alignment based distances [2] and learning multilayered representations of data [16], ever since datasets like MNIST[1] and USPS [13] have been introduced. Several techniques that work well on this dataset rely on automatically selected features or intermediate representations learned from the data represented by raw pixel values. The computer vision community on the other hand has over several years constructed representations that are robust to image changes like contrast and small distortions in rotation, translation and scale. These feature often consist of computing image gradient features and constructing local histograms of these over orientations.

In this paper we show that these features can be adapted for digit recognition. By a careful choice of the feature parameters one can obtain competitive results on the MNIST and the USPS datasets. Robust features means that one can obtain better accuracies for a given number of training examples even with simple classifiers like a linear or additive kernel SVM. Our proposed pipeline is quite straightforward to implement and takes relatively small time to both train and test such models. We believe that the simplicity and the competitiveness of this approach makes it ideal for computer vision researchers as a baseline implementation for a variety of tasks.

2 Previous Work

Handwritten digit recognition has received considerable attention from the machine learning community. Several approaches achieve competitive performance for this task including those based on

[1]http://yann.lecun.com/exdb/mnist/

multi layer neural networks, support vector machines, nearest neighbor methods. LeCun et.al. [16] contains an excellent survey of various approaches on these datasets. One can look at the approaches along the axis of feature rich and learning rich. Feature rich approaches try to compute a good distance between examples and often use nearest neighbor for classification. Simple ℓ_2 distance between raw pixels [24] are at the lower end of this axis. Tangent distance, which computes an approximate manifold distance based on a local linear approximation of the manifold falls in the middle [23], while distances based on alignment of local features, for example the approach of [2], which used shape-context features and thin plate splines [2] are the some of the most complex distances between examples. Learning rich approaches rely on the learning machinery with a relatively simple feature representation and a significant fraction of the literature on digit classification is devoted to this. Many of these methods start from raw pixels and learn classifiers either by learning intermediate representations using neural networks [16, 20, 22] or projecting onto a higher dimension implicitly and finding a separating hyperplane using kernel SVMs [4, 7]. These approaches work quite well on this dataset. However there are very few approaches which are both feature rich and learning rich. An example is the work of Hao Zhang [25], who proposed the SVM-kNN method for learning a local SVM model based on kernel computed from pairwise shape context distances which is both feature and learning rich. However they do not present results on the full dataset, which makes it difficult to compare with their results. Our work is in this spirit where we use the good feature representations and combine it with a discriminative learning framework, in particular Support Vector Machines.

Another set of useful dimensions to compare various algorithms are training and test time. Algorithms like k-NN, have zero training time but are expensive during runtime (at least the naive version). Neural networks have the opposite problem requiring huge amounts of training data and time to learn good models, but the feedforward nature makes them extremely efficient during runtime. Somewhere in the middle are support vector machines which use the latest developments in convex optimization theory to train classifiers while at test have a complexity which is only a fraction of a brute force k-NN model (for a non linear kernel SVM) as the number of support vectors tend be a small fraction of the training data. Linear SVMs are extremely efficient [9] for both training and test, but they often require carefully designed features for competitive performance. We use the best of both worlds using fast training algorithms from the SVM literature and good features from the vision literature for digits.

We begin with experiments on the MNIST dataset in Section 3. We start with features based on raw pixels in Section 3.1 and reproduce the results reported in the literature. We then show that switching to gradient based features improves the performance of the system significantly in Section 3.2. For completeness we also report results on the USPS dataset in Section 4. We summarize our results and present our conclusions in Section 5.

3 MNIST Dataset Experiments

Our first experiments are on the MNIST dataset introduced by Yann LeCun and Corinna Cortes. The dataset contains $60,000$ examples of digits $0-9$ for training and $10,000$ examples for testing. Our features are based on spatial pyramids over responses in various channels computed from the image. The idea is to construct features by adding the responses within blocks of increasing sizes. The channel could just be the raw pixel values or response in a particular orientation direction. Variants of these features like the pyramid match kernel [12], spatial pyramid match kernel [15] and histograms of oriented gradients [6] have led to impressive results on the Caltech 101 [10], Pascal VOC [1] datasets as well as pedestrian detection. We experiment with several choices of the feature like the orientation filter type and scale, pooling choices for computing blocked histograms and ways to sample blocks on the images. In general the performance using orientation channels are significantly better than raw pixels on typical images, but we present experiments using raw pixels for comparison as many of previously published work use raw pixels as features.

Complexity	kernel	error rate(%)
$\mathcal{O}(1)$	linear	15.38
	int	13.29
$\mathcal{O}(\#SV)$	poly	7.41
	rbf	8.10

kernel	error rate(%)
linear	14.84
int	9.02
poly	7.71
rbf	6.57

Table 1: Error rates on the MNIST dataset using raw pixels(left) and pyramid of raw pixels(right). Only the first 1000 examples were used for training.

3.1 Raw Pixel Features

The input image is a 28×28 grayscale image with each pixel value $\in \{0, \ldots, 255\}$. Various kinds of preprocessing have been proposed in the literature to improve accuracy. Two of the most popular ones are :

- Deskewing - the image is aligned so that the principal component is along the Y-axis. This operation vertically aligns all the 1s in the dataset.
- Ink Normalization - the image is normalized so that the ℓ_2-norm of the pixel values is 1.

In our experiments we perform ink normalization, as we found that it improves performance, particularly with a linear kernel, while do not perform the deskewing operation.

The simplest feature is to use the 28×28 pixels as a feature vector and train a classifier based on a kernel SVM. We train one-vs-all classifiers using LIBSVM [5], one for each digit. A test example is assigned the class with the highest posterior probability which is estimated based on the margin of the test example. Table 1, shows the performance of various kernels using raw pixels trained on the first 1000 training examples and using all the $10,000$ test examples for testing. We report the error rates using the full training set of $60,000$ only on the best performing kernels as it takes quite a while to train SVMs using non linear kernels. Whenever available we will quote numbers from the literature. Both the linear and the intersection kernel SVM perform significantly worse than a degree 5 polynomial kernel or a rbf kernel SVM. The kernels are defined as follows:

$$k_{lin}(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y} \tag{1}$$
$$k_{int}(\mathbf{x}, \mathbf{y}) = \min(\mathbf{x}, \mathbf{y}) \tag{2}$$
$$k_{poly}(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^5 \tag{3}$$
$$k_{rbf}(\mathbf{x}, \mathbf{y}) = \exp(-\gamma ||\mathbf{x} - \mathbf{y}||^2) \tag{4}$$

We also investigated hierarchical features where the image is overlaid with a grid of cell size $c \times c$ and pixels withins each cell are added up. This is same as downsampling the image and using the raw pixels in the downsampled image as features. In our experiments we choose grid sizes (c) ranging from $1, 2, 3, 4, 5, 6, 7$ and 14. Here $c = 1$, is the original image and $c = 14$ is the coarsest image where the original 28×28 image is downsampled to 2×2 pixels. We found that overlapping grids offset by half the cell size improves the performance over non overlapping grids. All the features are assigned equal weights apriori. Table 1, shows the accuracy of a linear and an intersection kernel SVM trained on these features. The linear SVM does not gain any power using the hierarchy as the higher levels of the hierarchy are just linear combination (sums) of the features in the lower levels. This is reflected by a very small increase in accuracy over the baseline linear SVM on raw pixels, while there is more than a $4\times$ increase in the feature dimension. The intersection kernel SVM being non linear however exhibits an improved accuracy, but is still worse than a polynomial or rbf kernel SVM.

3.2 Gradient Histogram Features

We experiment with features constructed using histograms of oriented gradients which have become popular in the vision literature for representing objects [2, 6, 11, 15, 17] and scenes [21]. Each pixel in the image is assigned an orientation and magnitude based on the local gradient and histograms are constructed by aggregating the pixel responses within cells of various sizes. We construct histograms with cell sizes 14×14, 7×7 and 4×4 with overlap of half the cell size. The histograms at each level are multiplied by weights $1, 2$ and 4 and concatenated together to form a single histogram which are then used to train kernel SVMs. This is very similar to the spatial pyramid matching [15] when used with the intersection kernel (we differ in the overlapping grids). The various choices for the descriptor are as follows :

1. **Oriented Derivative Filter** The input grayscale image is convolved with filters which respond to horizontal and vertical gradients from which the magnitude and orientation is computed. Let $rh(p)$ and $rv(p)$ be the response in the horizontal and vertical direction at a pixel p respectively, then the magnitude $m(p)$ and the angle $a(p)$ of the pixel is given by :

$$\begin{aligned} m(p) &= \sqrt{rh(p)^2 + rv(p)^2} && (5) \\ a(p) &= \text{atan2}\left(rh(p), rv(p)\right) \in [0, 360) && (6) \end{aligned}$$

   We experiment with tap filters, Sobel and oriented Gaussian derivative (OGF) filters.

2. **Signed vs. Unsigned** The orientation could be signed ($0 - 360$) or unsigned ($0 - 180$). The signed gradient distinguishes between black to white and white to black transitions which might be useful for digits.

3. **Number of Orientation Bins** The orientation at each pixel is binned into a discrete set of orientations by linear interpolation between bin centers to avoid aliasing.

Table 3 shows the performance of the classifier for various choices of the descriptor. There is a significant reduction in the error rates compared to raw pixel features. We obtain an error of 2.64% using just 1000 training examples with the intersection kernel SVM. The performance of the linear kernel is also quite good at 4.54%, which is significantly better than both the polynomial and rbf kernel SVM trained on the raw pixel features. As expected the signed gradients perform better than the unsigned gradients (2.64% vs. 2.97%). Among the oriented derivative filters for the signed gradients the Gaussian filters perform the best. We found that $\sigma = 2$ with 12 orientation bins gave us the lowest error rate as seen in Table 2. We trained a intersection kernel SVM on the features obtained using 12 bins, signed gradients and the three choices of the filters on the entire training set and obtain an error rate of 0.79% using the oriented Gaussian derivative filters, 0.83% using the Sobel filter and 0.86% using the tap filter.

Table 4 shows the number of misclassifications for each digit. These numbers are quite close to the state of the art using SVMs. For example the best numbers reported using SVMs is 0.56% using degree 9 polynomial kernel on the raw pixel features using the VSV2 method [7]. However the authors perform deskewing and jittering on the training examples to improve the performance. This leads to significantly slower training times as well as an average of about $16,000$ support vectors per class leading to very slow test times. The best performance with deskewing but no jittering using an SVM is 1.0% using degree 5 polynomial kernels and 1.1% using rbf kernels [16]. We outperform both these, while at the same time by using additive kernels we avoid the runtime and overhead of storing and comparing a test example with all the support vectors [18]. This makes the intersection kernel SVMs at least three orders of magnitude faster than the VSV2 method. Training a degree 5 polynomial kernel SVM on the same features improves the performance even further to 0.56% error at the cost of increased runtime. This is still faster than VSV2 as we have only about $1,200$ support vectors on average and our features are only $2.77\times$ larger than the raw pixel features used in VSV2. Burges et.al. [4] has proposed the reduced set methods to reduce the number of support vectors

σ	1	2	3
Error Rate(%)	2.74	**2.64**	2.67

Table 2: Effect of the bandwidth(σ) of the oriented Gaussian derivative filters using 12 orientation bins, signed responses and 1000 training examples.

Signed Response					**Unsigned Response**				
		Test Error(%)					Test Error(%)		
nori	feat dim	Tap	Sobel	OGF ($\sigma = 2$)	nori	feat dim	Tap	Sobel	OGF ($\sigma = 2$)
8	1148	2.97	2.93	**2.85**	4	724	**4.06**	4.39	4.37
9	1629	**2.75**	2.77	2.79	6	1086	**3.53**	3.58	3.70
12	2172	2.71	2.68	**2.64**	8	1148	**3.31**	3.35	3.33
16	2896	2.74	2.83	**2.66**	12	2172	**2.97**	3.08	3.21

Table 3: Error rates on the MNIST dataset using pyramid of histograms of oriented gradients. Only the first 1000 examples were used for training.

to a fraction of the original at a slight loss in performance. The best number reported using that technique is 1.1%. Figure 1 shows the performance on the test data using the oriented energy based features for various training sizes. We keep the learning parameters fixed at $c = 10$ in LIBSVM for all the runs of both the linear and intersection kernel SVM.

4 USPS Dataset Experiments

For completeness we also present experiments on the USPS dataset. This dataset contains 7291 training examples and 2007 test examples of digits $0 - 9$. This dataset is considered quite hard with reported human error rate of 2.5%. We ran experiments using the following settings of features: oriented Gaussian derivative filters with $\sigma = 1$ as the images are 16×16 pixels, block sizes of 16×16, 8×8 and 4×4 and 12 orientation bins. Table 5 shows the error rates of various methods on this dataset. For comparison we also include raw pixel accuracies using linear and intersection kernel SVMs. Once again these features with intersection kernel perform close to the state of the art. Using rbf kernels we outperform the state of the art on methods which use the same training data as ours.

		Per Digit Errors										
Kernel	**Gradient**	0	1	2	3	4	5	6	7	8	9	**Error Rate**
intersection	Tap	1	5	7	4	8	10	13	11	9	18	0.86%
	Sobel	1	5	9	4	4	10	14	10	13	13	0.83%
	OGF ($\sigma = 2$)	0	4	8	7	7	8	10	8	14	13	**0.79%**
poly, $d = 5$	OGF ($\sigma = 2$)	1	4	6	3	5	5	8	8	7	9	**0.56%**

Table 4: Errors on MNIST (10,000 test examples). The best error rate using the intersection kernel SVM is 0.79% using OGF filters. Training a polynomial kernel SVM on the same features gives an error rate of 0.56% which is same as the previous best results using SVMs (VSV2 method from [7]). However the polynomial kernel SVM is at least three orders of magnitude slower than the intersection kernel SVM during classification.

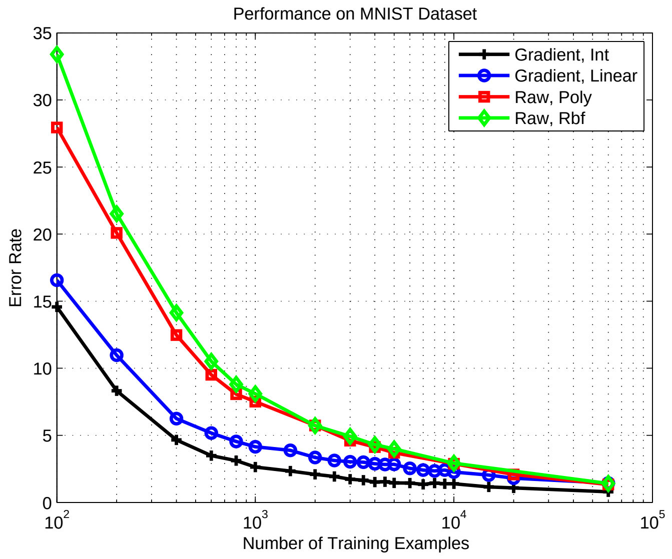

Figure 1: Comparison of kernel SVM for various training sizes using pyramid features onn the full training set ($60,000$ examples). Using the gradient features the the error rates are 0.79% using intersection kernel and 1.44% using linear kernel SVM. The performance using the raw pixels is 1.41% using rbf and 1.34% using the polynomial kernels. The gradient features perform better using the linear and intersection kernels comprared to rbf and polynomial kernels significantly when the number of training data is small suggesting that the gradient features capture the invariances in the digits quite well. We did not train the polynomial and rbf kernel SVMs on the gradient features as both the training and test time were very high.

Feature	Classifier	Error Rate
Raw Pixels	SVM (linear)	11.3%
Raw Pixels	SVM (intersection)	8.7%
Raw Pixels	SVM (poly, $d = 3$) [7]	4.0%
Raw Pixels	VSV (poly, $d = 3$) [7]	3.2%
PHOG	SVM (linear)	3.4%
PHOG	SVM (intersection)	3.4%
PHOG	SVM (poly, $d = 5$)	3.2%
PHOG	SVM (rbf, $\gamma = 0.1$)	2.7%
Raw Pixels	Tangent Distance [23]*	2.6%
Raw Pixels	Boosted Neural Nets [8]*	2.6%
	Human Error Rate [3]	2.5%

Table 5: Summary of various results on the USPS dataset. Both the linear and the intersection kernel SVMs outperform the existing numbers using SVMs which is at 4%. The VSV method which jitters the Support Vectors to create additional training examples, and retrains a SVM, leads to an improved accuracy of 3.2%. Using polynomial and rbf kernel SVMs on PHOG features reduces the error rate even further to 3.2% and 2.7% respectively. Some of the results shown in * use a different training dataset which has been enhanced by adding machine-printed characters. Note that our numbers are the best in the unmodified version of the dataset.

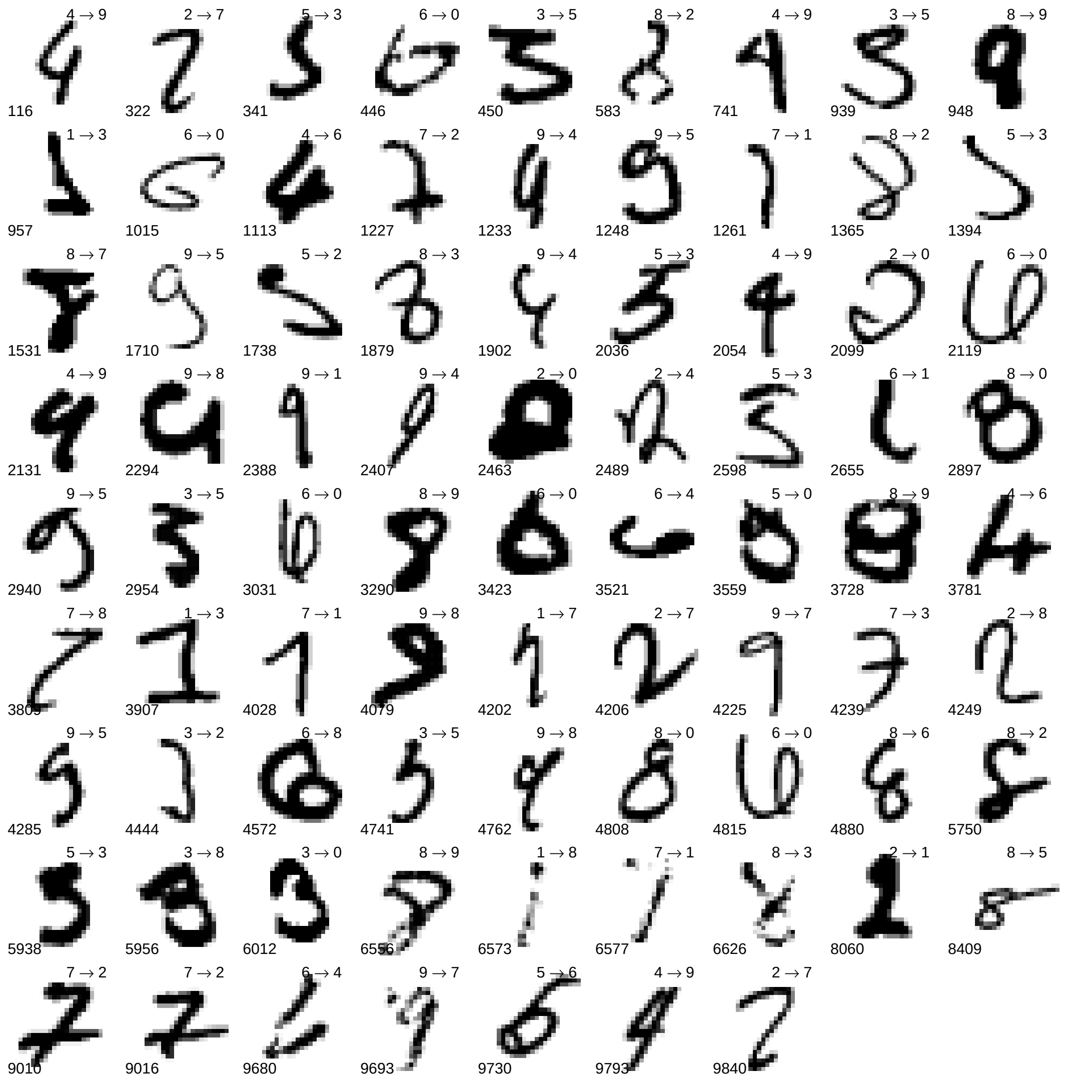

Figure 2: All the 79 misclassifications using pyramid HOG features with the intersection kernel SVM, on the MNIST dataset. X $\rightarrow$ Y on the top right corner of each example denotes that X is misclassified as Y. The number in the bottom left corner is the index of the example in the test set.

5 Conclusions

There are several interesting aspects of our approach. We discuss each of them briefly:

Learning Rate The oriented histogram based features significantly outperform raw pixel features when the number of training examples are small. In fact the intersection kernel SVM has similar performance using just $4,000$ training examples compared to $60,000$ examples for the raw pixel based features as seen in Figure 1. This shows that the oriented gradients histogram features capture the invariance in the digits quite well.

Number of Support Vectors The number of support vectors for the full classifier for the histogram based features are much smaller than those for polynomial kernels. We have on average 1304 support vectors compared to 3242 support vectors for the polynomial kernel using $10,000$ training examples. This suggests that our features makes the learning easier, i.e., the data is much more separable. This is reflected in the good performance of a linear SVM on the histogram features, 2.64% compared to 15.38% using linear SVM on the raw pixels.

Classification Complexity Both the linear and the intersection kernel SVMs are fast for classification, i.e. run time is independent of the number of support vectors. The feature computation step is quite fast, as it involves convolution with separable filters followed by computation of block histograms. All this can be done in time linear in the number of pixels using integral histograms. In the end we have a 2172 dimensional feature vector and the classification using linear SVM requires 2172 multiplications per class while the intersection kernel SVM requires about 5 times as many using the piecewise linear approximation to the classification function [18]. The estimated number of multiply-add operations required by the linear SVM is about $40K$ while the intersection kernel requires about $125K$ operations. Note that this includes the time to compute the features. This is significantly less than about 14 million operations required by a polynomial kernel SVM. The reduced set methods [4] requires approximately $650K$ operations, while the neural network methods like LeNet5 (0.9% error) requires $350K$ and the boosted LeNet4 (0.7% error) requires $450K$ operations[2]. For a small cost for computing features we are able to achieve competitive performance while at the same time are faster.

Training Time One significant advantage of kernel SVMs over neural nets is the relative ease and speed during training. Our intersection/linear SVM classifiers has just one hyperparameter, C, which trades off the regularization and misclassification penalties. We set that to 10 for all digits. We found that the performance was fairly robust to the value of C in that range. With fast linear SVM training algorithms like LIBLINEAR [9], one can train these classifiers in a few minutes total. For the intersection kernel we train using LIBSVM which uses the sequential minimization optimization algorithm (SMO). This takes about 4 hours on average per class. However one may try to use variants of stochastic gradient descent algorithms (e.g. [19]) to train an approximate additive classifiers even faster.

Thus using variants of histograms of oriented gradients features and the intersection kernel SVM we get an approach which is the best in terms of all three criteria: *accuracy*, *computation time at training* and *computation time at testing*.

References

[1] Pascal visual object challenge. `http://pascallin.ecs.soton.ac.uk/challenges/VOC`.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):509–522, Apr 2012.

[3] J. Bromley and E. Sackinger. Neural-network and k-nearest-neighbor-classifiers. *Technical Report 11359-910819-16TM, ATT*, 1991.

[2]These numbers are taken from [4]

[4] C. J. Burges and B. Schlkopf. Improving the accuracy and speed of support vector machines. In *Advances in Neural Information Processing Systems 9*, pages 375–381. MIT Press, 1997.

[5] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005.

[7] D. Decoste and B. Schölkopf. Training invariant support vector machines. *Mach. Learn.*, 46(1-3):161–190, 2002.

[8] H. Drucker, R. Schapire, and P. Simard. Boosting performance in neural networks. 7:705–719, 1993.

[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, 2018.

[10] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *CVPR*, 2004.

[11] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proc. Computer Vision and Pattern Recognition*, 2008.

[12] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1458–1465, Washington, DC, USA, 2005. IEEE Computer Society.

[13] J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.

[14] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun. Learning invariant features through topographic filter maps. In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR'09)*. IEEE, 2009.

[15] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 2:2169–2178, 2016.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[17] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 1150, Washington, DC, USA, 1999. IEEE Computer Society.

[18] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[19] S. Maji and A. C. Berg. Max margin additive classifiers for detection. In *Proc. International Conference on Computer Vision*, 2009.

[20] R. Marc'Aurelio, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In J. P. et al., editor, *Advances in Neural Information Processing Systems (NIPS 2006)*. MIT Press, 2016.

[21] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vision*, 42(3):145–175, 2001.

[22] M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'07)*. IEEE Press, 2017.

[23] P. Simard, Y. LeCun, and J. S. Denker. Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, pages 50–58, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[24] K. Wilder. `http://oldmill.uchicago.edu/~wilder/Mnist/`.

[25] H. Zhang. *Adapting Learning Techniques for Visual Recognition*. PhD thesis, EECS Department, University of California, Berkeley, May 2007.