

# THE MNIST DATABASE

## of handwritten digits

[Yann LeCun](#), Courant Institute, NYU  
[Corinna Cortes](#), Google Labs, New York

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

Four files are available on this site:

[train-images-idx3-ubyte.gz](#): training set images (9912422 bytes)  
[train-labels-idx1-ubyte.gz](#): training set labels (28881 bytes)  
[t10k-images-idx3-ubyte.gz](#): test set images (1648877 bytes)  
[t10k-labels-idx1-ubyte.gz](#): test set labels (4542 bytes)

**please note that your browser may uncompress these files without telling you.** If the files you downloaded have a larger size than the above, they have been uncompressed by your browser. Simply rename them to remove the .gz extension. Some people have asked me "my application can't open your image files". These files are not in any standard image format. You have to write your own (very simple) program to read them. The file format is described at the bottom of this page.

The original black and white (bilevel) images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. the images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.

With some classification methods (particularly template-based methods, such as SVM and K-nearest neighbors), the error rate improves when the digits are centered by bounding box rather than center of mass. If you do this kind of

pre-processing, you should report it in your publications.

The MNIST database was constructed from NIST's Special Database 3 and Special Database 1 which contain binary images of handwritten digits. NIST originally designated SD-3 as their training set and SD-1 as their test set. However, SD-3 is much cleaner and easier to recognize than SD-1. The reason for this can be found on the fact that SD-3 was collected among Census Bureau employees, while SD-1 was collected among high-school students. Drawing sensible conclusions from learning experiments requires that the result be independent of the choice of training set and test among the complete set of samples. Therefore it was necessary to build a new database by mixing NIST's datasets.

The MNIST training set is composed of 30,000 patterns from SD-3 and 30,000 patterns from SD-1. Our test set was composed of 5,000 patterns from SD-3 and 5,000 patterns from SD-1. The 60,000 pattern training set contained examples from approximately 250 writers. We made sure that the sets of writers of the training set and test set were disjoint.

SD-1 contains 58,527 digit images written by 500 different writers. In contrast to SD-3, where blocks of data from each writer appeared in sequence, the data in SD-1 is scrambled. Writer identities for SD-1 is available and we used this information to unscramble the writers. We then split SD-1 in two: characters written by the first 250 writers went into our new training set. The remaining 250 writers were placed in our test set. Thus we had two sets with nearly 30,000 examples each. The new training set was completed with enough examples from SD-3, starting at pattern # 0, to make a full set of 60,000 training patterns. Similarly, the new test set was completed with SD-3 examples starting at pattern # 35,000 to make a full set with 60,000 test patterns. Only a subset of 10,000 test images (5,000 from SD-1 and 5,000 from SD-3) is available on this site. The full 60,000 sample training set is available.

Many methods have been tested with this training set and test set. Here are a few examples. Details about the methods are given in an upcoming paper. Some of those experiments used a version of the database where the input images were deskewed (by computing the principal axis of the shape that is closest to the vertical, and shifting the lines so as to make it vertical). In some other experiments, the training set was augmented with artificially distorted versions of the original training samples. The distortions are random combinations of shifts, scaling, skewing, and compression.

CLASSIFIER	PREPROCESSING	TEST ERROR	Reference
------------	---------------	---------------	-----------

		RATE (%)	
linear classifier (1-layer NN)	none	12.0	<a href="#">LeCun et al. 1998</a>
linear classifier (1-layer NN)	deskewing	8.4	<a href="#">LeCun et al. 1998</a>
pairwise linear classifier	deskewing	7.6	<a href="#">LeCun et al. 1998</a>
K-nearest-neighbors, Euclidean (L2)	none	5.0	<a href="#">LeCun et al. 1998</a>
K-nearest-neighbors, Euclidean (L2)	none	3.09	<a href="#">Kenneth Wilder, U. Chicago</a>
K-nearest-neighbors, L3	none	2.83	<a href="#">Kenneth Wilder, U. Chicago</a>
K-nearest-neighbors, Euclidean (L2)	deskewing	2.4	<a href="#">LeCun et al. 1998</a>
K-nearest-neighbors, Euclidean (L2)	deskewing, noise removal, blurring	1.80	<a href="#">Kenneth Wilder, U. Chicago</a>
K-nearest-neighbors, L3	deskewing, noise removal, blurring	1.73	<a href="#">Kenneth Wilder, U. Chicago</a>
K-nearest-neighbors, L3	deskewing, noise removal, blurring, 1 pixel shift	1.33	<a href="#">Kenneth Wilder, U. Chicago</a>
K-nearest-neighbors, L3	deskewing, noise removal, blurring, 2 pixel shift	1.22	<a href="#">Kenneth Wilder, U. Chicago</a>
K-NN, shape context matching	shape context feature extraction	0.63	<a href="#">Belongie et al. IEEE PAMI 2002</a>

K-NN with non-linear deformation (IDM)	shiftable edges	0.54	<a href="#">Keyzers et al. IEEE PAMI 2007</a>
K-NN with non-linear deformation (P2DHMDM)	shiftable edges	0.52	<a href="#">Keyzers et al. IEEE PAMI 2007</a>
40 PCA + quadratic classifier	none	3.3	<a href="#">LeCun et al. 1998</a>
1000 RBF + linear classifier	none	3.6	<a href="#">LeCun et al. 1998</a>
K-NN, Tangent Distance	subsampling to 16x16 pixels	1.1	<a href="#">LeCun et al. 1998</a>
SVM, Gaussian Kernel	none	1.4	
SVM deg 4 polynomial	deskewing	1.1	<a href="#">LeCun et al. 1998</a>
Reduced Set SVM deg 5 polynomial	deskewing	1.0	<a href="#">LeCun et al. 1998</a>
Virtual SVM deg-9 poly [distortions]	none	0.8	<a href="#">LeCun et al. 1998</a>
Virtual SVM, deg-9 poly, 1-pixel jittered	none	0.68	DeCoste and Scholkopf, MLJ 2002
Virtual SVM, deg-9 poly, 1-pixel jittered	deskewing	0.68	DeCoste and Scholkopf, MLJ 2002
Virtual SVM, deg-9 poly, 2-pixel jittered	deskewing	0.56	DeCoste and Scholkopf, MLJ 2002
Trainable feature extractor + SVMs [no distortions]	none	0.83	<a href="#">Lauer et al., Pattern Recognition 40-6, 2007</a>
Trainable feature extractor + SVMs [elastic distortions]	none	0.56	<a href="#">Lauer et al., Pattern Recognition 40-6, 2007</a>

Trainable feature extractor + SVMs [affine distortions]	none	0.54	<a href="#">Lauer et al., Pattern Recognition 40-6, 2007</a>
2-layer NN, 300 hidden units, mean square error	none	4.7	<a href="#">LeCun et al. 1998</a>
2-layer NN, 300 HU, MSE, [distortions]	none	3.6	<a href="#">LeCun et al. 1998</a>
2-layer NN, 300 HU	deskewing	1.6	<a href="#">LeCun et al. 1998</a>
2-layer NN, 1000 hidden units	none	4.5	<a href="#">LeCun et al. 1998</a>
2-layer NN, 1000 HU, [distortions]	none	3.8	<a href="#">LeCun et al. 1998</a>
3-layer NN, 300+100 hidden units	none	3.05	<a href="#">LeCun et al. 1998</a>
3-layer NN, 300+100 HU [distortions]	none	2.5	<a href="#">LeCun et al. 1998</a>
3-layer NN, 500+150 hidden units	none	2.95	<a href="#">LeCun et al. 1998</a>
3-layer NN, 500+150 HU [distortions]	none	2.45	<a href="#">LeCun et al. 1998</a>
3-layer NN, 500+300 HU, softmax, cross entropy, weight decay	none	1.53	<a href="#">Hinton, unpublished, 2005</a>
2-layer NN, 800 HU, Cross-Entropy Loss	none	1.6	<a href="#">Simard et al., ICDAR 2003</a>
2-layer NN, 800 HU, cross-entropy [affine distortions]	none	1.1	<a href="#">Simard et al., ICDAR 2003</a>
2-layer NN, 800 HU, MSE [elastic distortions]	none	0.9	<a href="#">Simard et al., ICDAR 2003</a>
2-layer NN, 800 HU, cross-entropy [elastic distortions]	none	0.7	<a href="#">Simard et al., ICDAR 2003</a>
NN, 784-500-500-2000-30 + nearest neighbor, RBM + NCA	none	1.0	<a href="#">Salakhutdinov and Hinton,</a>

training [no distortions]			<a href="#">AI-Stats 2007</a>
6-layer NN 784-2500-2000-1500-1000-500-10 (on GPU) [elastic distortions]	none	0.35	<a href="#">Ciresan et al., arXiv 1003.0358, 2010</a>
boosted stumps	none	7.7	<a href="#">Kegl et al., ICML 2009</a>
products of boosted stumps (3 terms)	none	1.26	<a href="#">Kegl et al., ICML 2009</a>
boosted trees (17 leaves)	none	1.53	<a href="#">Kegl et al., ICML 2009</a>
stumps on Haar features	Haar features	1.02	<a href="#">Kegl et al., ICML 2009</a>
product of stumps on Haar f.	Haar features	0.87	<a href="#">Kegl et al., ICML 2009</a>
Convolutional net LeNet-1	subsampling to 16x16 pixels	1.7	<a href="#">LeCun et al. 1998</a>
Convolutional net LeNet-4	none	1.1	<a href="#">LeCun et al. 1998</a>
Convolutional net LeNet-4 with K-NN instead of last layer	none	1.1	<a href="#">LeCun et al. 1998</a>
Convolutional net LeNet-4 with local learning instead of last layer	none	1.1	<a href="#">LeCun et al. 1998</a>
Convolutional net LeNet-5, [no distortions]	none	0.95	<a href="#">LeCun et al. 1998</a>
Convolutional net LeNet-5, [huge distortions]	none	0.85	<a href="#">LeCun et al. 1998</a>
Convolutional net LeNet-5, [distortions]	none	0.8	<a href="#">LeCun et al. 1998</a>
Convolutional net Boosted LeNet-4, [distortions]	none	0.7	<a href="#">LeCun et al. 1998</a>
unsupervised sparse features + SVM, [no distortions]	none	0.59	<a href="#">Labusch et al., IEEE TNN 2008</a>

Convolutional net, cross-entropy [affine distortions]	none	0.6	<a href="#">Simard et al., ICDAR 2003</a>
Convolutional net, cross-entropy [elastic distortions]	none	0.4	<a href="#">Simard et al., ICDAR 2003</a>
large conv. net, random features [no distortions]	none	0.89	<a href="#">Ranzato et al., CVPR 2007</a>
large conv. net, unsup features [no distortions]	none	0.62	<a href="#">Ranzato et al., CVPR 2007</a>
large conv. net, unsup pretraining [no distortions]	none	0.60	<a href="#">Ranzato et al., NIPS 2006</a>
large conv. net, unsup pretraining [elastic distortions]	none	0.39	<a href="#">Ranzato et al., NIPS 2006</a>
large conv. net, unsup pretraining [no distortions]	none	0.53	<a href="#">Jarrett et al., ICCV 2009</a>

## References

[LeCun et al., 1998a]

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. [\[on-line version\]](#)

## FILE FORMATS FOR THE MNIST DATABASE

The data is stored in a very simple file format designed for storing vectors and multidimensional matrices. General info on this format is given at the end of this page, but you don't need to read that to use the data files.

All the integers in the files are stored in the MSB first (high endian) format used by most non-Intel processors. Users of Intel processors and other low-endian machines must flip the bytes of the header.

There are 4 files:

train-images-idx3-ubyte: training set images

```

train-labels-idx1-ubyte: training set labels
t10k-images-idx3-ubyte: test set images
t10k-labels-idx1-ubyte: test set labels

```

The training set contains 60000 examples, and the test set 10000 examples.

The first 5000 examples of the test set are taken from the original NIST training set. The last 5000 are taken from the original NIST test set. The first 5000 are cleaner and easier than the last 5000.

### **TRAINING SET LABEL FILE (train-labels-idx1-ubyte):**

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000801(2049)	magic number (MSB first)
0004	32 bit integer	60000	number of items
0008	unsigned byte	??	label
0009	unsigned byte	??	label
.....			
xxxx	unsigned byte	??	label

The labels values are 0 to 9.

### **TRAINING SET IMAGE FILE (train-images-idx3-ubyte):**

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
.....			
xxxx	unsigned byte	??	pixel

Pixels are organized row-wise. Pixel values are 0 to 255. 0 means background (white), 255 means foreground (black).

### **TEST SET LABEL FILE (t10k-labels-idx1-ubyte):**

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000801(2049)	magic number (MSB first)
0004	32 bit integer	10000	number of items
0008	unsigned byte	??	label
0009	unsigned byte	??	label
.....			
xxxx	unsigned byte	??	label

The labels values are 0 to 9.

### **TEST SET IMAGE FILE (t10k-images-idx3-ubyte):**

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	10000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
.....			
xxxx	unsigned byte	??	pixel

Pixels are organized row-wise. Pixel values are 0 to 255. 0 means background (white), 255 means foreground (black).

---

## THE IDX FILE FORMAT

the IDX file format is a simple format for vectors and multidimensional matrices of various numerical types.

The basic format is

```

magic number
size in dimension 0
size in dimension 1
size in dimension 2
.....
size in dimension N
data

```

The magic number is an integer (MSB first). The first 2 bytes are always 0.

The third byte codes the type of the data:

```

0x08: unsigned byte
0x09: signed byte
0x0B: short (2 bytes)
0x0C: int (4 bytes)
0x0D: float (4 bytes)
0x0E: double (8 bytes)

```

The 4-th byte codes the number of dimensions of the vector/matrix: 1 for vectors, 2 for matrices....

The sizes in each dimension are 4-byte integers (MSB first, high endian, like in most non-Intel processors).

The data is stored like in a C array, i.e. the index in the last dimension changes the fastest.

Happy hacking.

---

The digit images in the MNIST set were originally selected and experimented with by Corinna Cortes using bounding-box normalization and centering. Yann LeCun's version which is provided on this page uses centering by center of mass within in a larger window.

[Yann LeCun](#), Professor  
The Courant Institute of Mathematical Sciences  
New York University  
[yann@cs.nyu.edu](mailto:yann@cs.nyu.edu)

[Corinna Cortes](#), Research Scientist  
Google Labs, New York  
*corinna at google dot com*