Tunning Parameters of Deep Neural Network Algorithm for identifying best Cost function

Chetak Kandaswamy, Telmo Amaral

22nd April 2013

NNIG Technical Report No. 2/2013

Project "Reusable Deep Neural Networks: Applications to Biomedical Data"
(PDTC/EIA-EIA/119004/2010)

Instituto de Engenharia Biomédica (INEB)
Rua Dr. Roberto Frias, 4200-465, Porto, Portugal

Contents

1 Classification Test Results **3**
1.1 Graphical representation of Adult, Dna and Mushrooms Dataset Test Errors 3
1.2 Parameters used for Testing: Based on Mean Validation Error 3

2 Calculation of various costs **4**
2.1 Notation 4
2.2 MSE, CE and EXP 4

3 Parameter Tunning: Adult Dataset **4**
3.1 Identifying best number of hidden layers and Size (Grid Search) 5
3.2 Identifying best Pre training Learning rate 5
3.3 Identifying best Fine tuning Learning rate 5
3.4 Identifying best tau 6

4 Learning Curve for Adult Dataset **6**

5 Computational Complexity **6**
5.1 Order the datasets based on computation time 7
5.2 Data set: mnist_basic 7
5.3 Identifying best number of hidden layers and Size (Grid Search) 7

6 Parameter Tunning: DNA Dataset **8**
6.1 Identifying best number of hidden layers and Size (Grid Search) 8
6.2 Identifying best Pre training, Fine tuning Learning rate and Tau's 9

7 Parameter Tunning: Mushrooms Dataset **9**
7.1 Identifying best number of hidden layers and Size (Grid Search) 9
7.2 Identifying best Pre training, Fine tuning Learning rate and Tau's 10

8 Survey on Datasets used for Deep Learning **10**

References **12**

1 Classification Test Results

We present the test results of the Deep Neural Network algorithm on various datasets. The test were conducted on tunned parameters of the deep neural network. The following sections contain performance measurements of the sum of squared errors (SSE) or Mean square Error (MSE), cross-entropy (CE), and exponential (EXP) cost functions in order to the weights and biases of an auto-encoder. In this report, we compare the performances of MSE, CE, and EXP costs when employed in the pre-training of deep networks whose hidden layers are treated as stacks of auto-encoders. For each combination of data set, pre-training greedy module, and pre-training cost function, the test stage was repeated 30 times. Table 1 shows the mean and standard deviation* of the test errors obtained in each case. Also included for comparison are the results achieved without pre-training.

Table 1: Mean and standard deviation of test errors for each data set and pre-training cost function, using as greedy module (a) auto-encoders and (b) denoising auto-encoders.

(a) AE

data set	no pre-training	pre-training cost function		
		CE	MSE	EXP
adult	15.98±0.15%	16.73±0.33%	16.5±0.32%	**15.91±0.13%**
dna	7.30±0.51%	7.58±0.60%	**7.28±0.62%**	7.42±0.76%
mushrooms	0.29±0.05%	0.23±0.16%	0.16±0.10%	**0.15±0.15%**

(b) DAE

data set	no pre-training	pre-training cost function		
		CE	MSE	EXP
adult	15.98±0.15%	16.71±0.36%	16.42±0.26%	**15.94±0.13%**
dna	**7.30±0.51%**	7.38±0.52%	7.36±0.24%	8.17±0.17%
mushrooms	0.29±0.05%	0.22±0.15%	0.15±0.14%	**0.06±0.03%**

* mean and std are multiplied 10^2.

1.1 Graphical representation of Adult, Dna and Mushrooms Dataset Test Errors

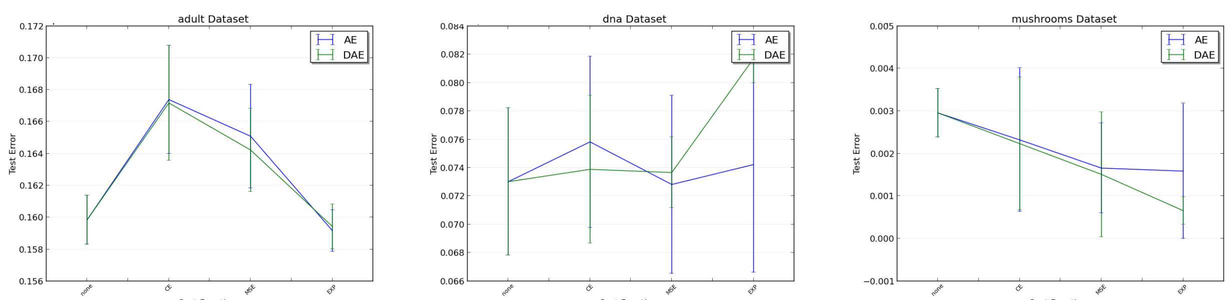

Figure 1: Comparison of Test Error of (a) Adult, (b) dna & (c) mushrooms dataset with various Cost functions (none, CE, MSE and EXP)

1.2 Parameters used for Testing: Based on Mean Validation Error

A grid search of the hyper-parameter was conducted even though it was exhaustive. It was possible for smaller and quiker datasets as shown in table 4. It seemed prohibitive, as our experiments relied on CPUs and could take very long to run (especially when the image sets *mnist-subset, mnist_ basic* and *rectangles* were involved). We were able to conduct the archtectural grid search for smaller datasets for selection procedure, then tuning each hyper-parameter individually to minimise the validation error. In most cases, averaging the error over a few repetitions helped to identify the best value for the parameter being explored. All hyper-parameters were tuned using CE pre-training costs, except for τ (tau), which affects specifically the EXP cost function. Table 2 shows all the selected values.To get better approximation and repeatability of the experiments each parameter is repeated 10 times.

Table 2: Hyper-parameter values selected for each combination of data set and greedy module.

data set	greedy module									
	AE					DAE				
	l	n_h	τ	η_{PT}	η_{FT}	l	n_h	τ	η_{PT}	η_{FT}
adult	2	[4,6]	-40	0.01	0.09	2	[4,6]	-60	0.009	0.07
dna	2	[6,9]	-40	0.009	0.09	2	[6,9]	-60	0.01	0.01
mushrooms	2	[10,15]	-10	0.01	0.09	2	[10,15]	-10	0.01	0.03

2 Calculation of various costs

2.1 Notation

The following sections contain calculations for the partial derivatives of the MSE, CE, and EXP cost functions in order to the weights and biases of an auto-encoder.The following notation is used for the auto-encoder:

n_x	number of inputs and outputs
n_h	number of hidden units
$x_j, j \in \{1, 2, ..., n_x\}$	value of jth input
$h_i, i \in \{1, 2, ..., n_h\}$	value of ith hidden unit
$\hat{x}_j, j \in \{1, 2, ..., n_x\}$	value of jth output
W_{ij}	weight connecting ith hidden unit to jth input weight connecting ith hidden unit to jth output
b_i	bias of ith hidden unit
c_j	bias of jth output
θ	any individual weight or bias

Each $\sum$ or $\frac{\partial}{\partial\theta}$ symbol applies to all multiplicative terms to its right.

2.2 MSE, CE and EXP

The MSE or (SSE) error between an $\hat{\mathbf{x}}$ vector of outputs and an $\mathbf{x}$ vector of inputs is expressed by

$$C_{SSE}(\hat{\mathbf{x}}, \mathbf{x}) = \sum_{k=1}^{n_x} (\hat{x}_k - x_k)^2 \tag{1}$$

The CE error between an $\hat{\mathbf{x}}$ vector of outputs and an $\mathbf{x}$ vector of inputs is expressed by

$$C_{CE}(\hat{\mathbf{x}}, \mathbf{x}) = -\sum_{k=1}^{n_x} \Big(x_k \ln(\hat{x}_k) + (1 - x_k) \ln(1 - \hat{x}_k) \Big) \tag{2}$$

The EXP error between an $\hat{\mathbf{x}}$ vector of outputs and an $\mathbf{x}$ vector of inputs is expressed by

$$C_{EXP}(\hat{\mathbf{x}}, \mathbf{x}) = \tau \exp\Big(\frac{1}{\tau} \sum_{k=1}^{n_x} (\hat{x}_k - x_k)^2 \Big) \tag{3}$$

3 Parameter Tunning: Adult Dataset

(for both Autoencoder and Denoising Autoencoder Greedy Module)

To test the performance of our designed cost function EXP with traditional cost function like Cross Entropy(CE) we tune the parameters with CE as the base cost. Thus tunned parameters will be used for comparative performance study of our EXP with traditional cost function like CE and MSE.

We tune parameter number of hidden layer and size of hidden layer with cost function CE. And tune tau with cost function EXP as it is a EXP parameter. The noise probability is set to 0.1 for Denoising Autoencoder.

3.1 Identifying best number of hidden layers and Size (Grid Search)

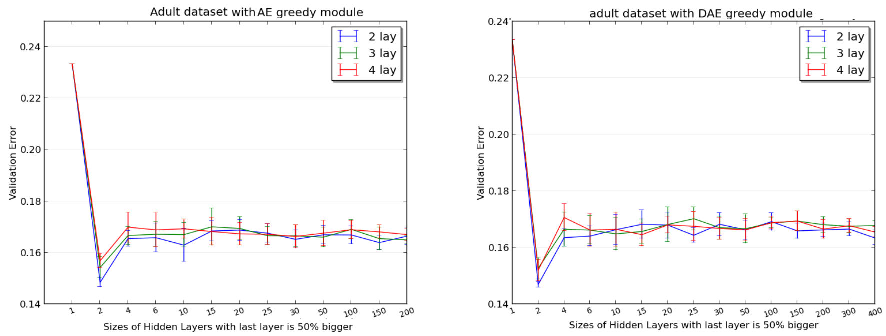

Figure 2: Sizes of hidden layers for (a) AE and (b) DAE greedy module

3.2 Identifying best Pre training Learning rate

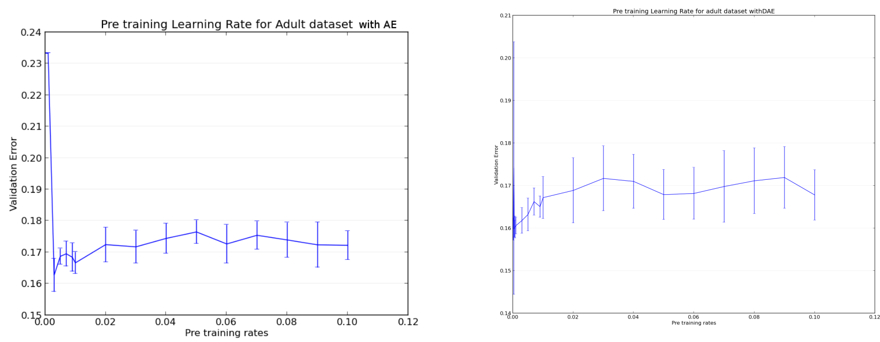

Figure 3: Pre Training Learning rate for (a) AE and (b) DAE greedy module

3.3 Identifying best Fine tuning Learning rate

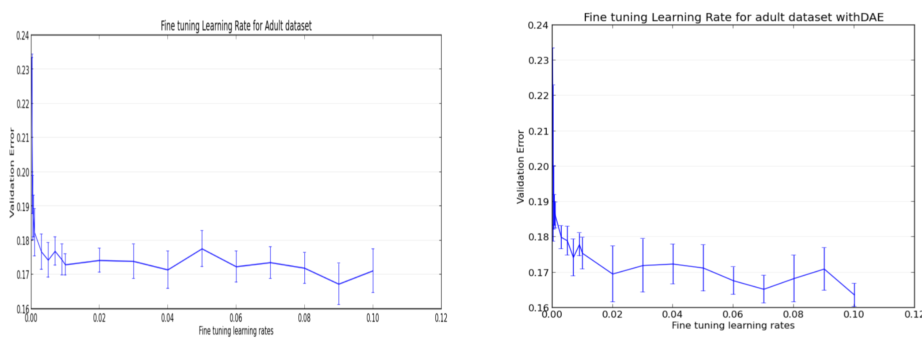

Figure 4: Fine tuning Learning rate for (a) AE and (b) DAE greedy module

3.4 Identifying best tau

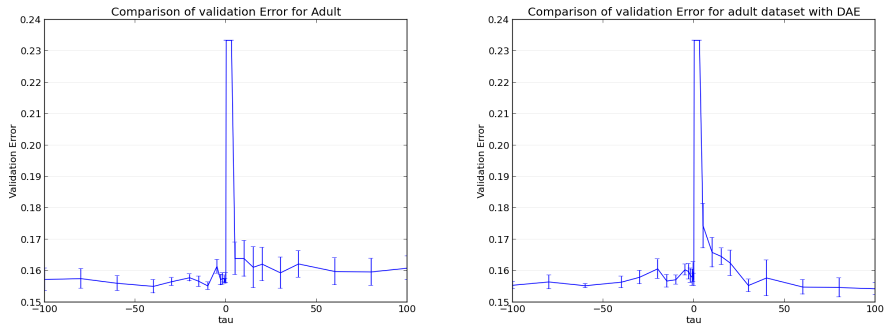

Figure 5: Various tau's for (a) AE and (b) DAE greedy module

4 Learning Curve for Adult Dataset

The Machine Learning Learning curve is conducted to study the behavior of adult dataset over the Autoencoder greedy module with EXP cost function. It can be studied that with varying the training set size we can see the the performance of the deep neural network test error. In the figure we have studied the Learning curve without parameter tunning and after parameter tunning. It can be inferred that, proper tunning converges the training error and testing error thus reducing the variance in the model. It can also be seen that at lower training sizes the model does not produce over fitting. This response initiates a series of future study into making effects of supervisory mode and also the balanced error rate for the dataset. The experiment was done with 25 repetition for getting good approximation.

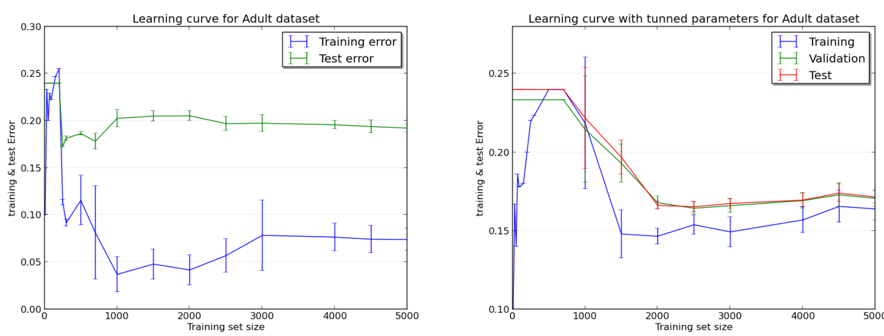

Figure 6: Learning Curve (a) without parameter tunning (b) after parameter tunning

5 Computational Complexity

We studied the impact of computation complexity on the number of neural network units used on various datasets. The datasets to be used on our designed deep neural network has a limitations as larger the datasets require higher computation capability. We presently use CPU (Central Processing Unit) based processing for computing the algorithm.

Table 3: Characteristics of the data sets used in the experiments.

data set	# features	# targets	# instances			type
			train	valid.	test	
adult	123	2	5000	1414	26147	binary
dna	180	3	1400	600	1186	binary
mnist-subset	784	10	5000	1000	1000	real-valued
mnist-basic	784	10	10000	2000	50000	real-valued
mushrooms	112	2	2000	500	5624	binary
rectangles	784	2	1000	200	50000	binary

5.1 Order the datasets based on computation time

The simulations were run on 6 datasets for identifying least computation time on the validation instances. The simulation results indicate the time taken by dna dataset for training and validation is least among the 6 datasets.

Table 4: datasets are ordered to least computation time

#	Dataset	Validation error	Time (Sec)
1	**dna**	**0.069**	**8.76**
2	mushrooms	0.0014	11.98
3	adult	0.069	32.73
4	rectangles	0.084	40.58
5	minst_subset	0.057	116.92
6	mnist_basic	0.0345	282.7

default values:

- greedy Module = DAE with noise_prob of 0.1,
- hidden layer & sizes = [200, 200]
- pt learning rate = 0.01 and ft_learning_rate = 0.1
- cost = EXP and tau = 1

5.2 Data set: mnist_basic

Table 5: Computation time required by the mnist basic dataset

#	dataset	hidden size	time	parallel	iter	val	test
1	mnist basic	[1800, 2700]	11h	18h	1	0.0265	0.03236
2	mnist basic	[1200,1800]	6h		1	0.027	0.03486
3	mnist basic	[1000,1500]	5h	10h	1	0.0265	0.03454
4	mnist basic	[800, 1200]	5h	-	1	0.0295	0.03372
5	mnist basic	[600, 900]	2h	-	1	0.0265	0.03488
6	mnist basic	[200, 200]	5min	-	1	0.0345	0.04188

5.3 Identifying best number of hidden layers and Size (Grid Search)

Single iteration for hidden Size above 400 neural network units.

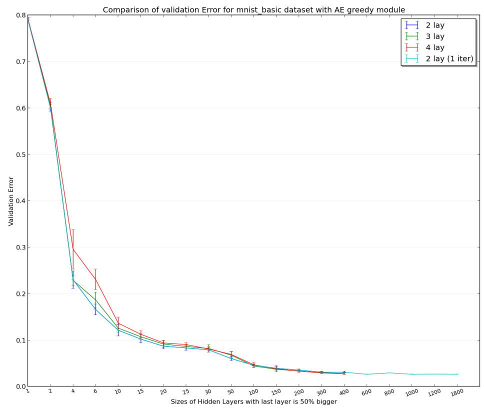

Figure 7: Sizes of hidden layers

6 Parameter Tunning: DNA Dataset

(for both Autoencoder and Denoising Autoencoder Greedy Module)

To test the performance of our designed cost function EXP with traditional cost function like Cross Entropy(CE) we tune the parameters with CE as the base cost. Thus tunned parameters will be used for comparative performance study of our EXP with traditional cost function like CE and MSE.

We tune parameter number of hidden layer and size of hidden layer with cost function CE. And tune tau with cost function EXP as it is a EXP parameter.

6.1 Identifying best number of hidden layers and Size (Grid Search)

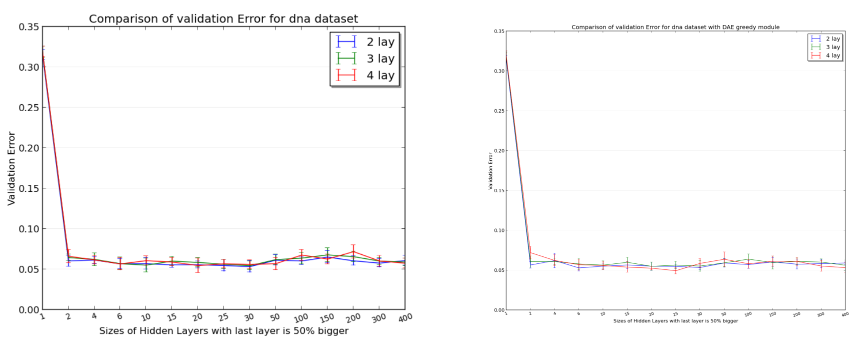

Figure 8: Sizes of hidden layers for (a) AE and (b) DAE greedy module

6.2 Identifying best Pre training, Fine tuning Learning rate and Tau's

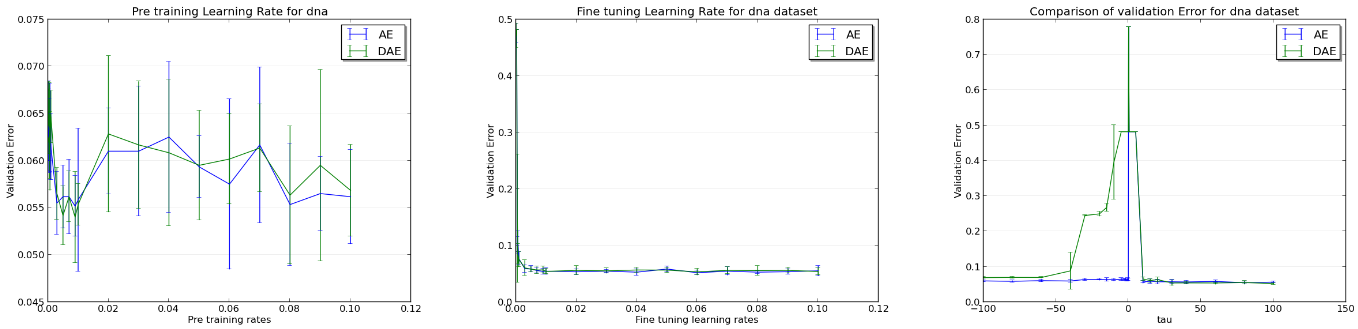

Figure 9: (a) Pre Training Learning rate, (b) Fine Tuning Learning rate & (c) Tau's

7 Parameter Tunning: Mushrooms Dataset

(for both Autoencoder and Denoising Autoencoder Greedy Module)

To test the performance of our designed cost function EXP with traditional cost function like Cross Entropy(CE) we tune the parameters with CE as the base cost. Thus tunned parameters will be used for comparative performance study of our EXP with traditional cost function like CE and MSE.

We tune parameter number of hidden layer and size of hidden layer with cost function CE. And tune tau with cost function EXP as it is a EXP parameter. The noise probability is set to 0.1 for denoising autoencoder.

7.1 Identifying best number of hidden layers and Size (Grid Search)

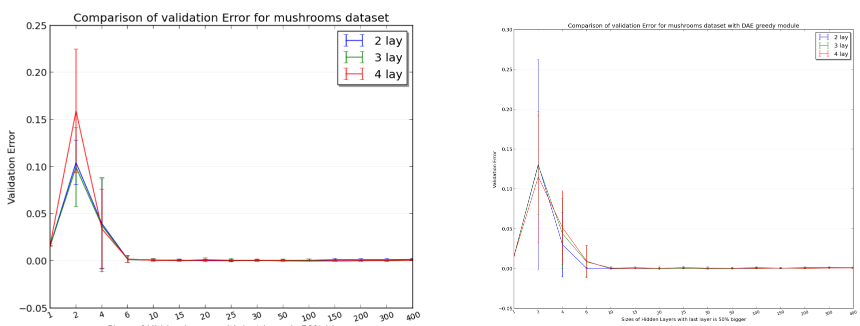

Figure 10: Sizes of hidden layers for (a) AE and (b) DAE greedy module

7.2 Identifying best Pre training, Fine tuning Learning rate and Tau's

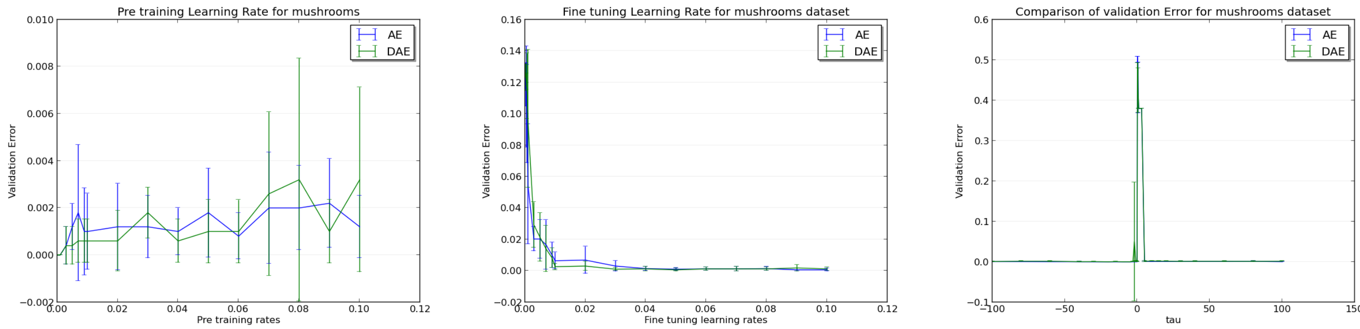

Figure 11: (a) Pre Training Learning rate, (b) Fine Tuning Learning rate & (c) Tau's

8 Survey on Datasets used for Deep Learning

A table is presented below with recent publications on Deep Learning from 2009 to 2013. The aim is to identify the trend in deep learning algorithms on the various type of database. The Survey is done from the following conference and Journals.

International Conference on Machine Learning (ICML)
Journal of Machine Learning Research (JMLR)
Advances in Neural Information Processing Systems (NIPS)
Institute of Electrical and Electronics Engineers (IEEE)
*International Conference on Learning Representations (ICLR) started in 2013
Springer Journal

#	Datasets	Cite	Reference	Publication	Type	Details
1	**MNIST**	**238**	[7][15][2][14][12][11][5][6][16]	JMLR, ICML, ICLR, Springer	Image	Handwritten digits
2	InfiniteMNIST		[7]	JMLR	Image	Handwritten digits
3	Shapeset	37	[7]	JMLR	Image	Handwritten digits
4	Cornell grasping		[8]	ICLR	Image	
5	CIFAR-10	28	[3][9]	JMLR	Image	1.6 millions of tiny images datasets
6	NORB	38	[3][15]	JMLR, ICML	Image	images of 50 different 3D toy objects
7	STL		[3]	JMLR	Image	
8	(no dataset)		[1]	Trends	Image	Uses theoretical explanation to Image
9	(no dataset)		[10]	Trends	Image/ Audio	Uses theoretical explanation to Image and audio
10	CUAVE	145	[13]	ICML	Audio	36 speakers saying digits 0 to 9
11	AVLetters		[13]	ICML	Audio	10 speakers saying the letters A to Z
12	AVLetters2		[13]	ICML	Audio	5 speakers saying the letters A to Z
13	Stanford Dataset		[13]	ICML	Audio	23 volunteers spoke the digits 0 to 9
14	**TIMIT**	**800**	[5][13]	ICML	Audio	(Speech) spoke the letters A to Z
15	MIREX		[17]	ICLR	Audio	Music Information Retrieval (MIR)
16	Aurora 4 corpus	29		ICLR	Audio	5000-word vocabulary
17	Semantic Role Labeling	33	[16]	Springer	Audio	1 million labeled trainset and 631 million unlabeled set
18	**ASTRAL**	**475**	[6]	NIPS	Domain	(Biomedical)set of protein domains
19	Colon, Leukemia, Prostate, SR- BCT and Brain		[4]	JMLR	Domain	gene expression data sets

From the table we can infer the major publications and conferences in recent years generally use Image database. But Their is a ASTRAL a protein based database has a biomedical references. TIMIT which is a speech based is generally used for Speech recognition.

References

[1] Yoshua Bengio. Learning deep architectures for ai. *To appear in Foundations and Trends in Machine Learning*.

[2] Enhong Chen. Image denoising and inpainting with deep neural networks. 2012.

[3] Adam Coates, Honglak Lee, and Andrew Y. Ng. An analysis of single-layer networks in unsupervised feature learning. *Ann Arbor*, 1001:48109, 2010.

[4] Xiaowei Zhang Delin Chu. Sparse uncorrelated linear discriminant analysis. *Journal of Machine Learning Research*.

[5] Li Deng, Dong Yu, and J. Platt. Scalable stacking and learning for building deep architectures. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2133–2136, 2012.

[6] Pietro Di Lena, Pierre Baldi, and Ken Nagata. Deep spatio-temporal architectures and learning for protein structure prediction. In *Advances in Neural Information Processing Systems 25*, 2012.

[7] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11, March 2010.

[8] Honglak Lee Ian Lenz and Ashutosh Saxena. Deep learning for detecting robotic grasps. *International Conference on Learning Representations*, 2013.

[9] Alex Krizhevsky. Convolutional deep belief networks on CIFAR-10. *Unpublished manuscript*, 2010.

[10] Jonathan Laserson. From neural networks to deep learning: zeroing in on the human brain. *XRDS: Crossroads, The ACM Magazine for Students*, 18(1), 2011.

[11] F.Q. Lauzon. An introduction to deep learning. In *2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*, pages 1438–1439, 2012.

[12] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Commun. ACM*, 54(10), October 2011.

[13] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. Multimodal deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11), ICML*, volume 11, 2011.

[14] Zineng Yuan Anthony Bonner Zhaolei Zhang Renqiang Min, Laurens van der Maaten. Deep supervised t-distributed embedding. *International Conference on Learning Representations*, 2013.

[15] Ruslan Salakhutdinov. Learning deep boltzmann machines using adaptive MCMC. In *Proceedings of the International Conference on Machine Learning*, volume 27, 2010.

[16] Jason Weston, FrÃ©dÃ©ric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*. Springer, 2012.

[17] Changshui Zhang Zhen Hu, Kun Fu. Audio classical composer identification by deep neural network. *International Conference on Learning Representations*, Mar, 2013.