Deep Learning in MIR: Demystifying the Dark

Part II:

The State-of-the-Art

by Philippe Hamel (hamelphi@google.com)
National Institute of Advanced Industrial Science and Technology (AIST), Japan
Google Inc., Mountain View, USA

ISMIR 2013
November 4
Curitiba, Brazil

Overview

The state-of-the-art in deep learning

- Pre-training and initialization
- Dropouts
- Activation functions
- Structure (convolution, recurrent)
- Parallelization

Deep learning in MIR

Pre-training and initialization

Pre-training

What is pre-training?

- Greedy layerwise unsupervised training.
- Typically, stacked RBMs or DAEs.

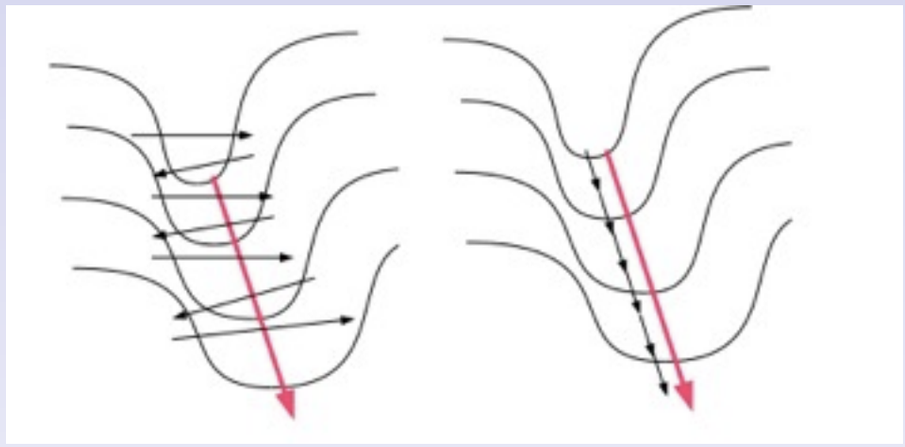
What does it do?

- Learns features from input data distribution.
- Initializes a network with good parameters.
- Helps prevent underfitting and overfitting. [Ehran et al. (2010)]

Deep learning without pretraining

Second-order method: Hessian-Free Optimization

- Martens et al. (2010)



First-order method: Stochastic gradient descent (SGD)

- Glorot & Bengio (2010), Mohamed et al. (2012), Krizhevsky et al. (2012)

Parameter initialization

Instead of pretraining, initialize parameters at a “sensible” scale.

What is “sensible”?

It’s complicated...

- Glorot & Bengio(2010)

$$W \sim U\Big[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\Big]$$

- Sutskever et al. (2012)

Should I use pretraining?

It depends...

Do you have a lot of labeled data ?

- Probably NO

Do you have a lot of unlabeled data and few labeled examples?

- Probably YES

Dropout

Combining Models

Principle: Train many models independently and combine them at test time.

Successful applications of model combination:

- Netflix prize winners
- Random forests (Kaggle competions)

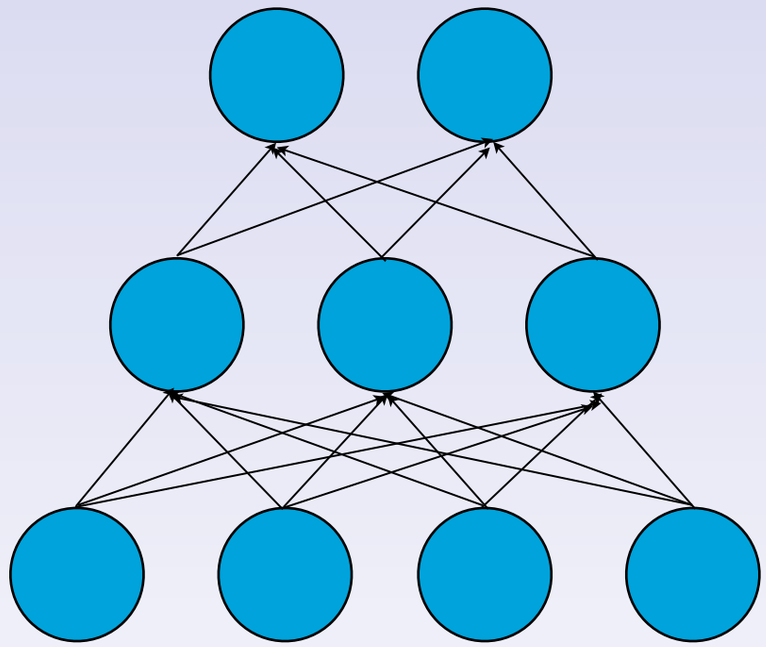
Neural Network Ensemble Learning

Problem: Combining neural networks is costly in terms of training time

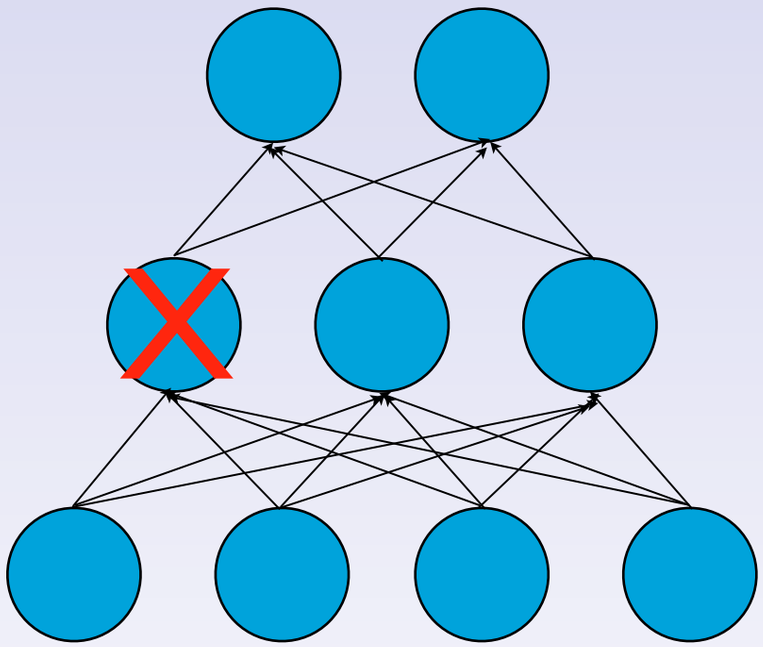
Solution: sharing weights between models

In practice: turn off units with probability 0.5

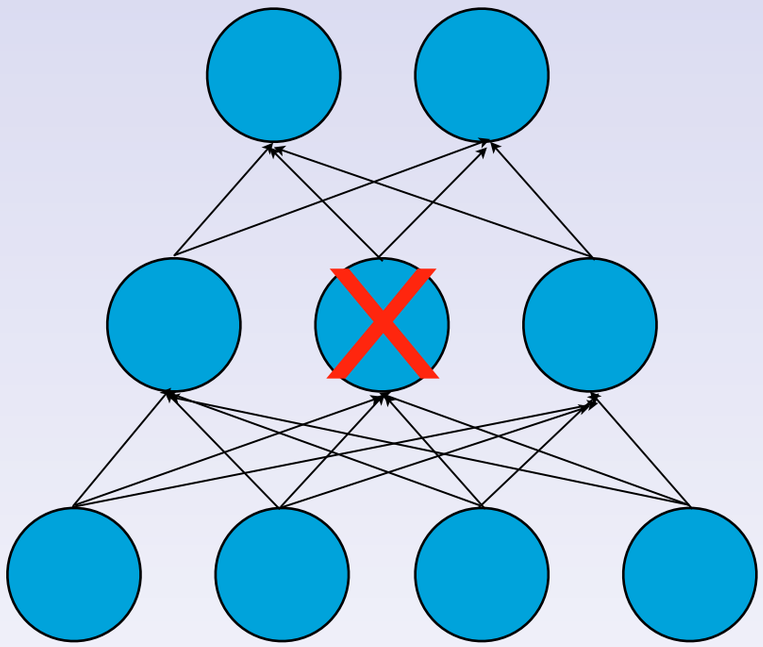
Dropout



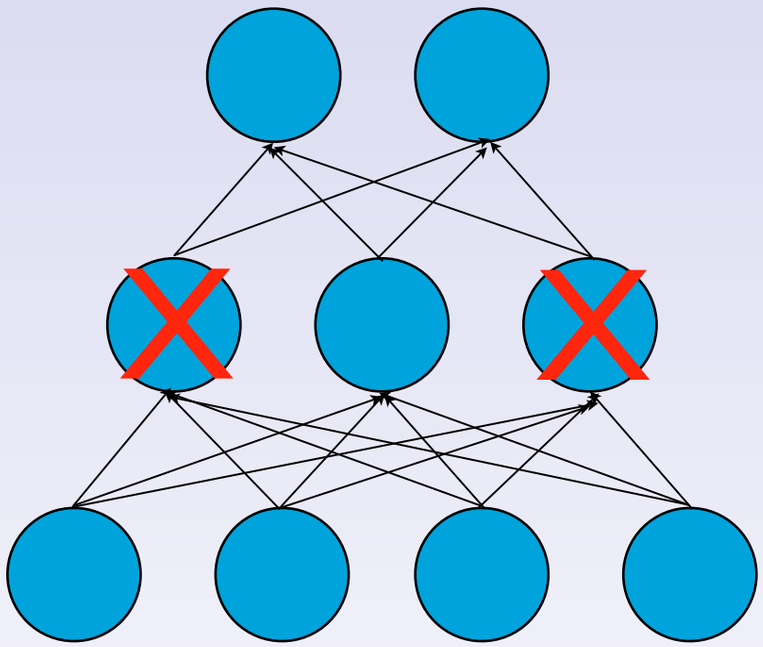
Dropout



Dropout



Dropout



Dropout

At test time: divide all weights by 2.

For a single layer network:

- Equivalent of taking the geometric mean of the output of 2^N networks.

For a deep network:

- Close enough approximation. Works very well in practice.

Dropout

- Better regularization than L2
- Prevents co-adaptation of feature detectors
- Allow to train larger networks
- Better performance

Dropout results

MNIST

Method	Unit Type	Error %
2 layer NN [19]	Logistic	1.60
SVM gaussian kernel	-	1.4
Dropout	ReLU	1.25
Dropout + weight norm constraint	ReLU	1.05
DBN + finetuning	Logistic	1.18
DBN + dropout finetuning	Logistic	0.92
DBM + finetuning	Logistic	0.96
DBM + dropout finetuning	Logistic	**0.79**

Dropout results

Street view house numbers

Method	Error %
Binary Features (WDCH) [14]	36.7
HOG [14]	15.0
Stacked Sparse Autoencoders [14]	10.3
KMeans [14]	9.4
Multi-stage Conv Net with average pooling [18]	9.06
Multi-stage Conv Net + L2 pooling [18]	5.36
Multi-stage Conv Net + L4 pooling + padding [18]	4.90
Conv Net + max-pooling	3.95
Conv Net + max pooling + dropout in fully connected layers	3.02
Conv Net + max pooling + dropout in all layers	2.78
Conv Net + max pooling + dropout in all layers + input translations	**2.68**
Human Performance	2.0

Dropout results

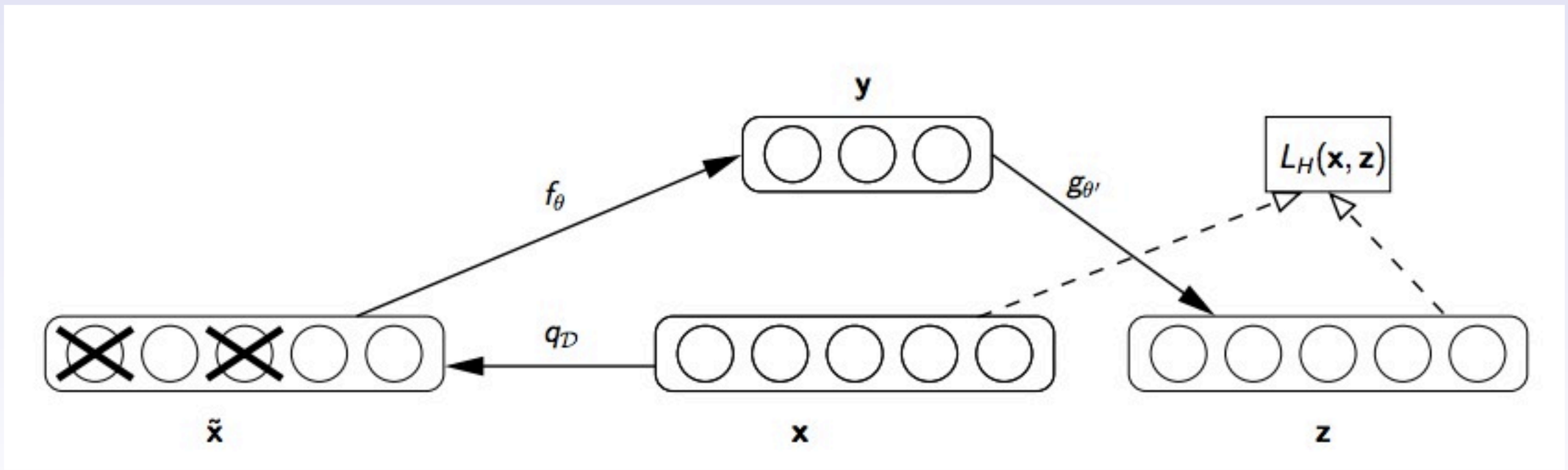
TIMIT

Method	Phone Error Rate%
Neural Net (6 layers) [12]	23.4
Dropout Neural Net (6 layers)	21.8
DBN-pretrained Neural Net (4 layers)	22.7
DBN-pretrained Neural Net (6 layers) [12]	22.4
DBN-pretrained Neural Net (8 layers) [12]	20.7
mcRBM-DBN-pretrained Neural Net (5 layers) [2]	20.5
DBN-pretrained Neural Net (4 layers) + dropout	**19.7**
DBN-pretrained Neural Net (8 layers) + dropout	**19.7**

Dropping out inputs

Denoising Auto-Encoders (DAE)

[Vincent et al. (2008)]



Activation functions

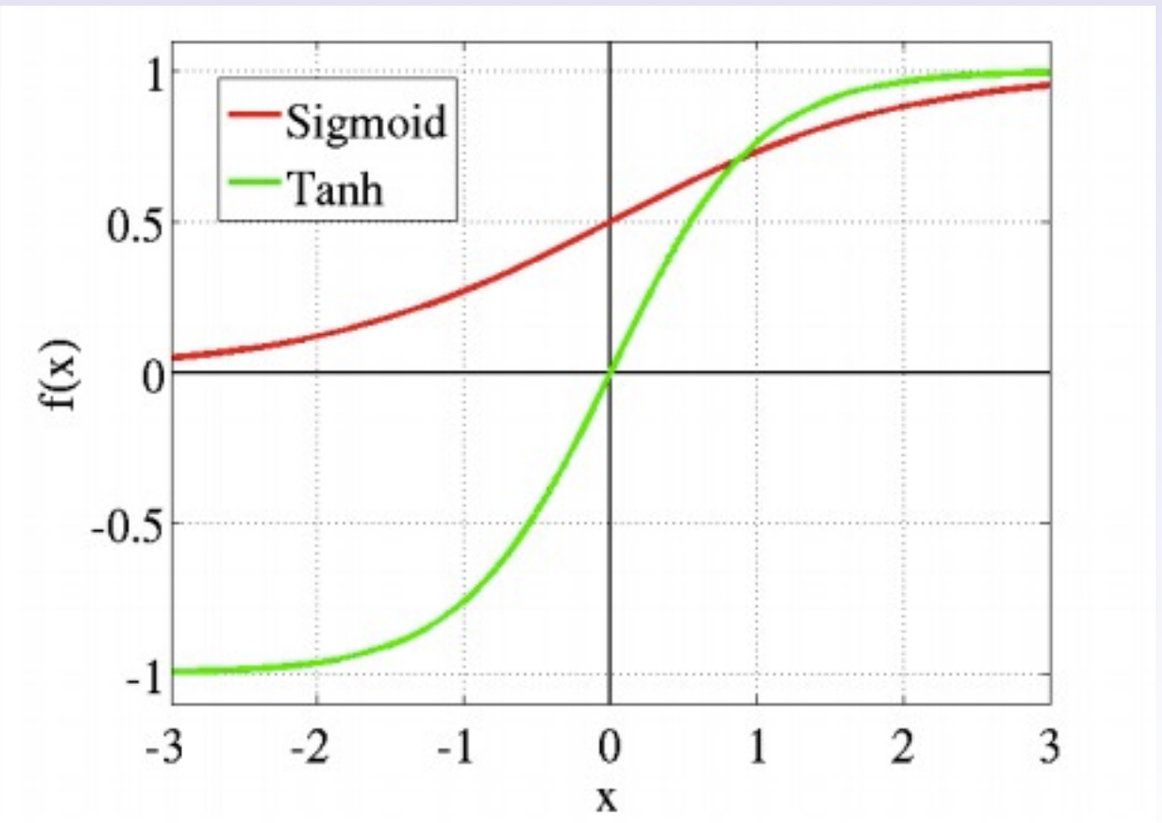
Activation functions

NNet with non-linear activation function: universal approximator for continuous functions. [Cybenko 1989, Hornik 1991]

$$a(x) = f(Wx + b)$$

Sigmoid:

$$S(t) = \frac{1}{1 + e^{-}t}$$

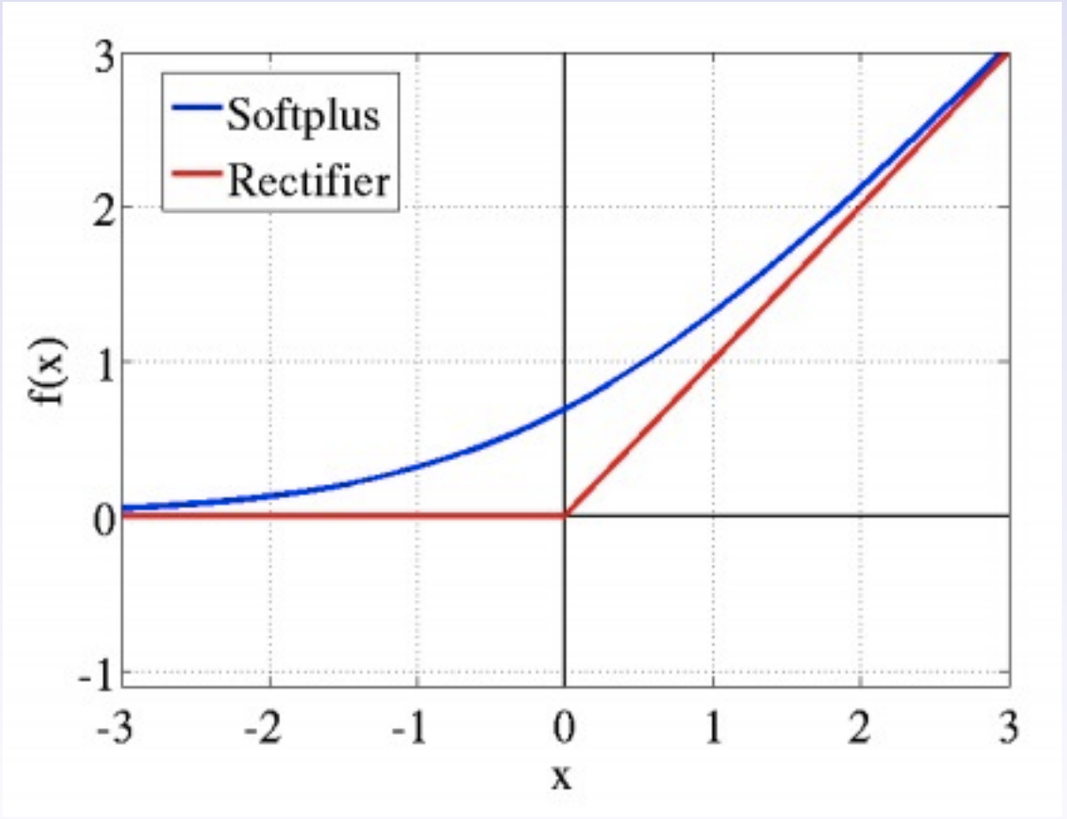


Rectified linear units (ReLUs)

- Learns faster than sigmoid or tanh function.
- Outputs are sparse.

$$rectifier(x) = max(0, x)$$

$$softplus(x) = log(1 + e^x)$$



ReLUs results

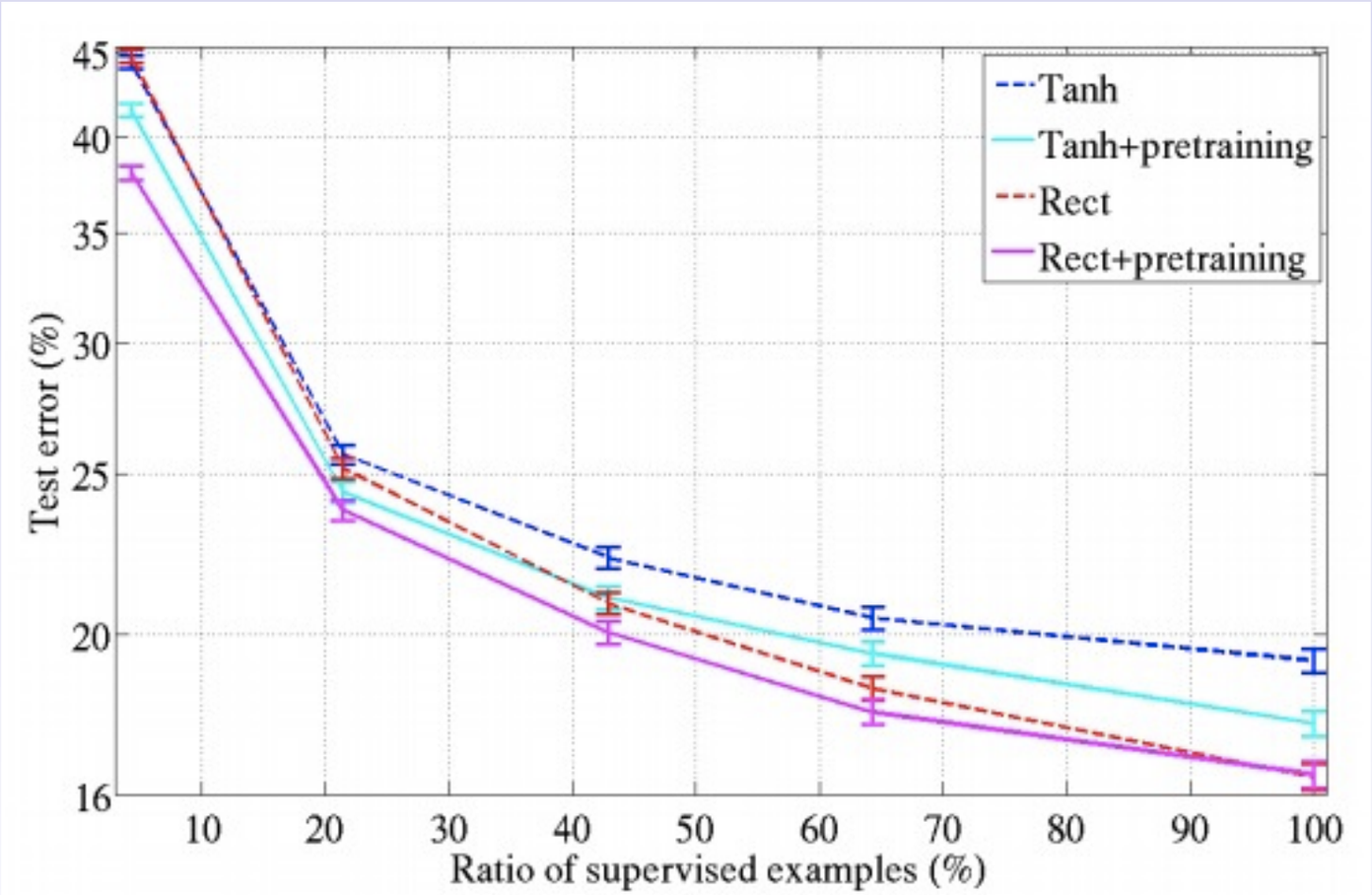
[Glorot et al. (2011)]

Table 1: **Test error on networks of depth 3.** Bold results represent statistical equivalence between similar experiments, with and without pre-training, under the null hypothesis of the pairwise test with $p = 0.05$.

Neuron	MNIST	CIFAR10	NISTP	NORB
***With* unsupervised pre-training**				
Rectifier	**1.20%**	**49.96%**	**32.86%**	**16.46%**
Tanh	**1.16%**	**50.79%**	35.89%	17.66%
Softplus	**1.17%**	**49.52%**	**33.27%**	19.19%
***Without* unsupervised pre-training**				
Rectifier	**1.43%**	**50.86%**	**32.64%**	**16.40%**
Tanh	1.57%	52.62%	36.46%	19.29%
Softplus	1.77%	53.20%	35.48%	17.68%

ReLUs results

[Glorot et al. (2011)]



Norb dataset

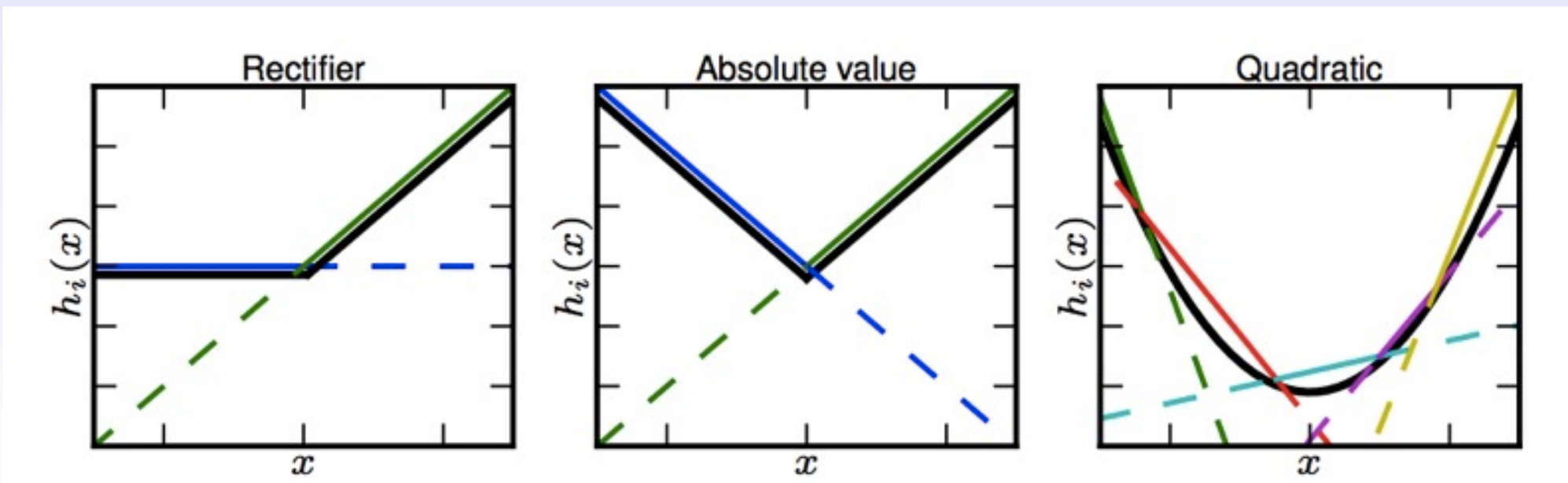
Maxout:
Piecewise linear units

Combination of linear functions.

$$h_i(x) = \max_{j \in [1,k]} z_{ij}$$

Can approximate an abitrary convex function.

$$z_{ij} = x^T W_{\ldots ij} + b_{ij}$$



[Goodfellow (2013)]

Maxout

Street view house numbers

Method	Test error
(Sermanet et al., 2012a)	4.90%
Stochastic pooling (Zeiler & Fergus, 2013)	2.80 %
Rectifiers + dropout (Srivastava, 2013)	2.78 %
Rectifiers + dropout + synthetic translation (Srivastava, 2013)	2.68 %
Conv. maxout + dropout	**2.47 %**

Structure

Structure

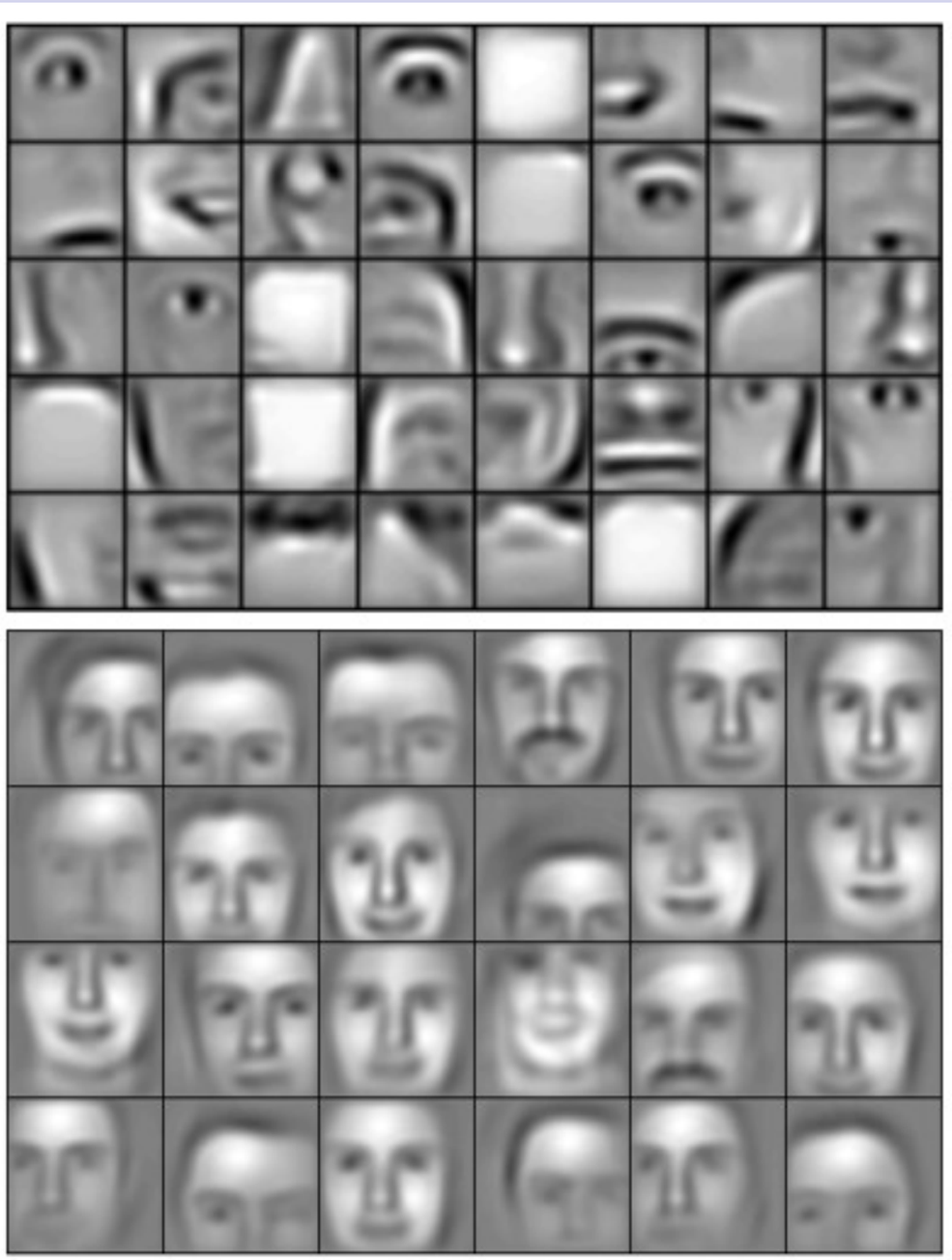
Spatial structure

- Convolutional neural networks (CNN)
- Local receptive fields

Sequential structure (e.g. in time)

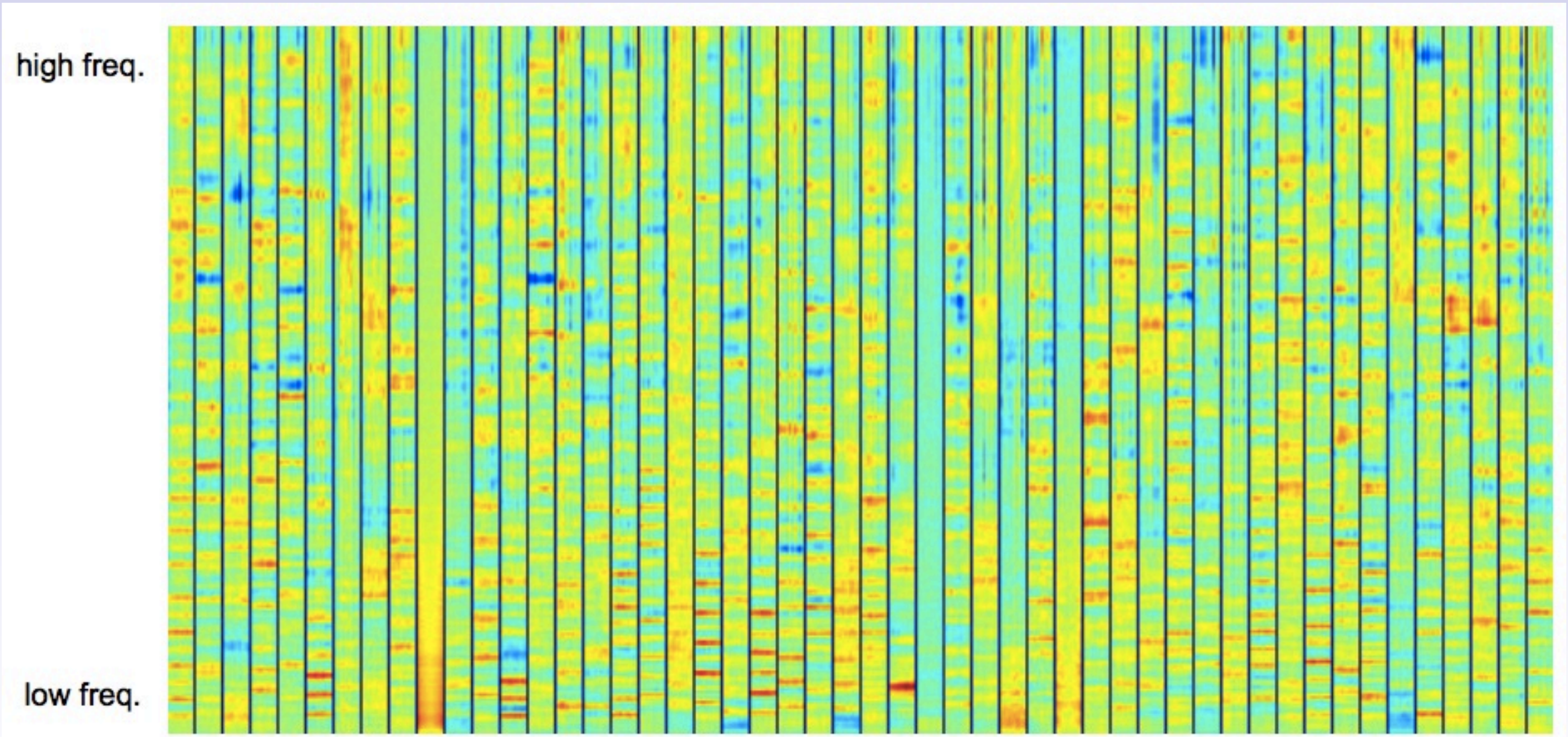
- Recurrent neural networks (RNN)
- Long short-time memory (LSTM)

Convolutional Deep Belief Network



[Lee (2009)]

Convolutional Deep Belief Network



[Lee (2009)]

Convolutional Deep Belief Network

Table 5: Test accuracy for 5-way music genre classification

Train examples	RAW	MFCC	CDBN L1	CDBN L2	CDBN L1+L2
1	51.6%	54.0%	**66.1%**	62.5%	64.3%
2	57.0%	62.1%	**69.7%**	67.9%	69.5%
3	59.7%	65.3%	**70.0%**	66.7%	69.5%
5	65.8%	68.3%	**73.1%**	69.2%	72.7%

[Lee (2009)]

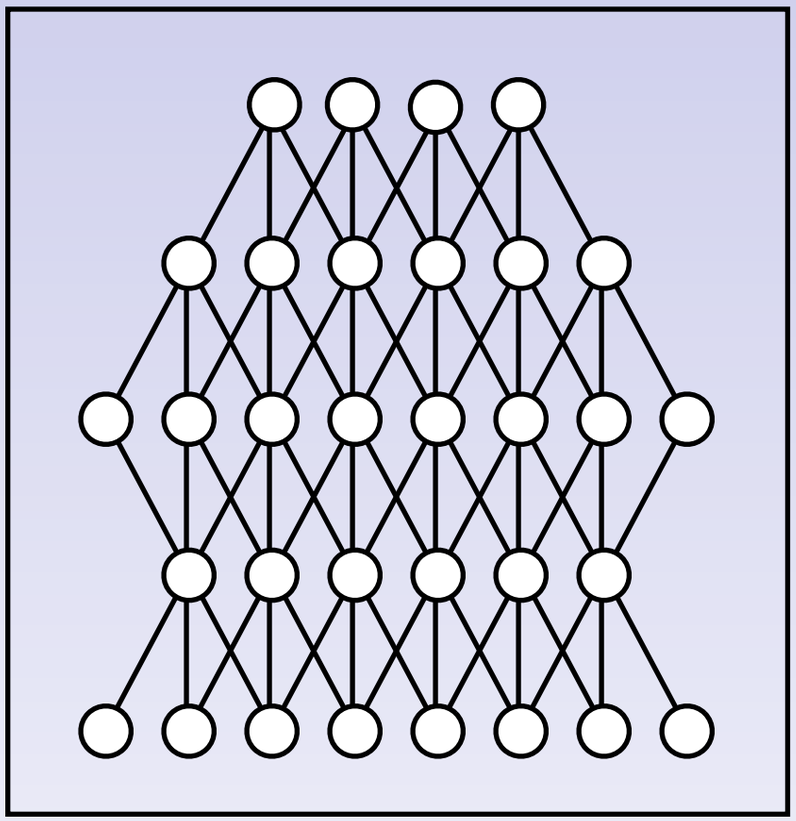
Recurrent Networks

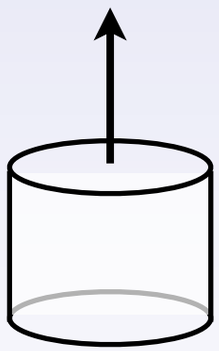
- Vanishing gradient problem [Bengio (1997)]
- HF optimisation [Martens & Sutskever (2011)]
- Deep RNN (Bidirectionnal LSTM) [Graves et al. (2013)]

Parallelization

DistBelief (Google)
[Dean (2012)]

Model

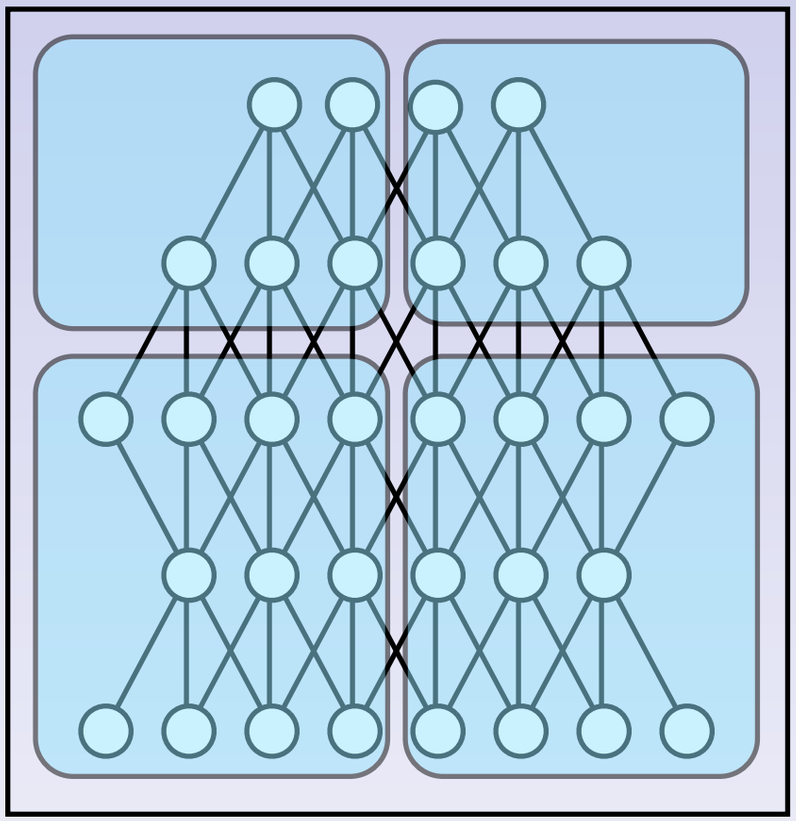


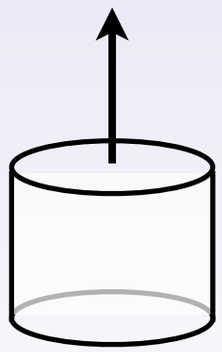


Training Data

Consider your favorite
deep learning model,
layered graphical model, or
classical neural network

Model





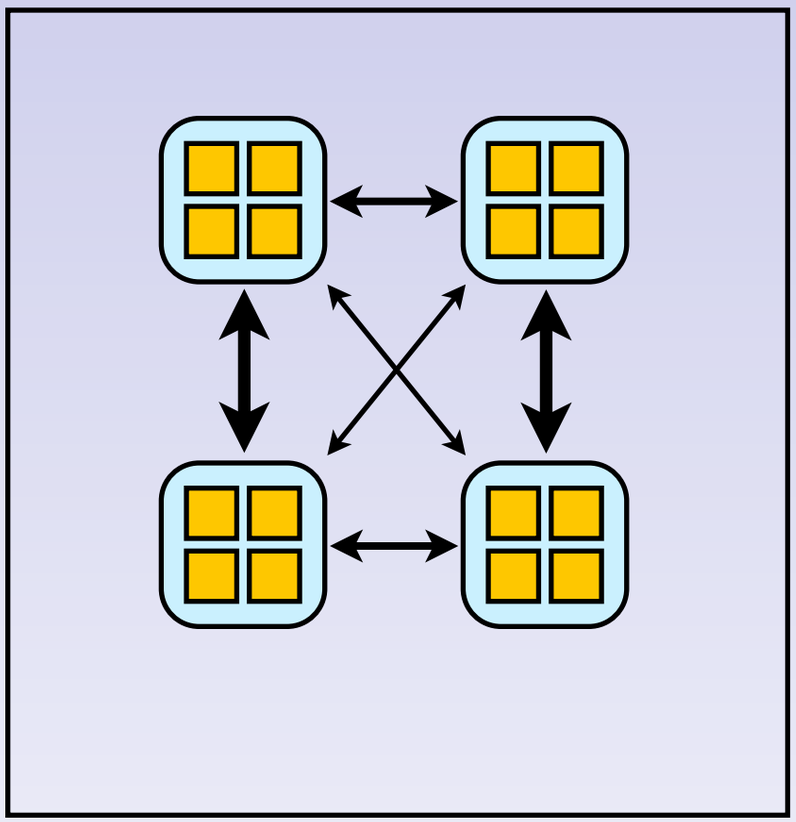
Training Data

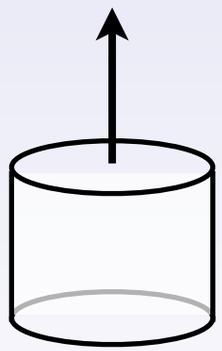
Consider your favorite deep learning model, layered graphical model, or classical neural network

DistBelief allows you to easily distribute computation within your model across multiple machines and multiple cores.

Machine (Model Partition)

Model





Training Data

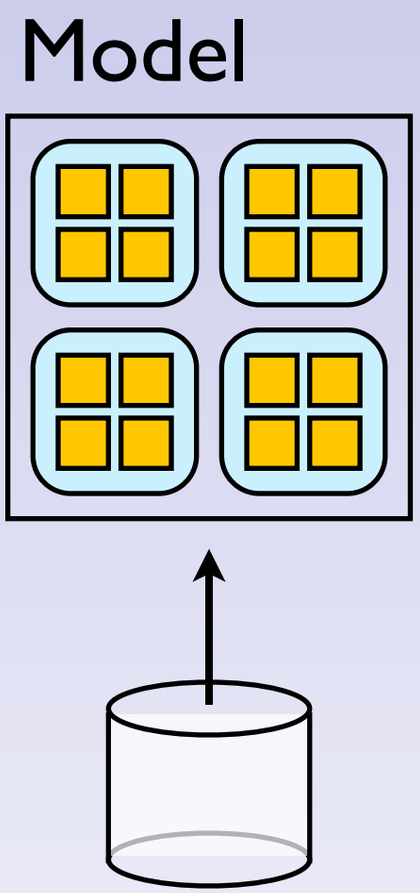
Consider your favorite deep learning model, layered graphical model, or classical neural network

DistBelief allows you to easily distribute computation within your model across multiple machines and multiple cores.

Machine (Model Partition)

Core

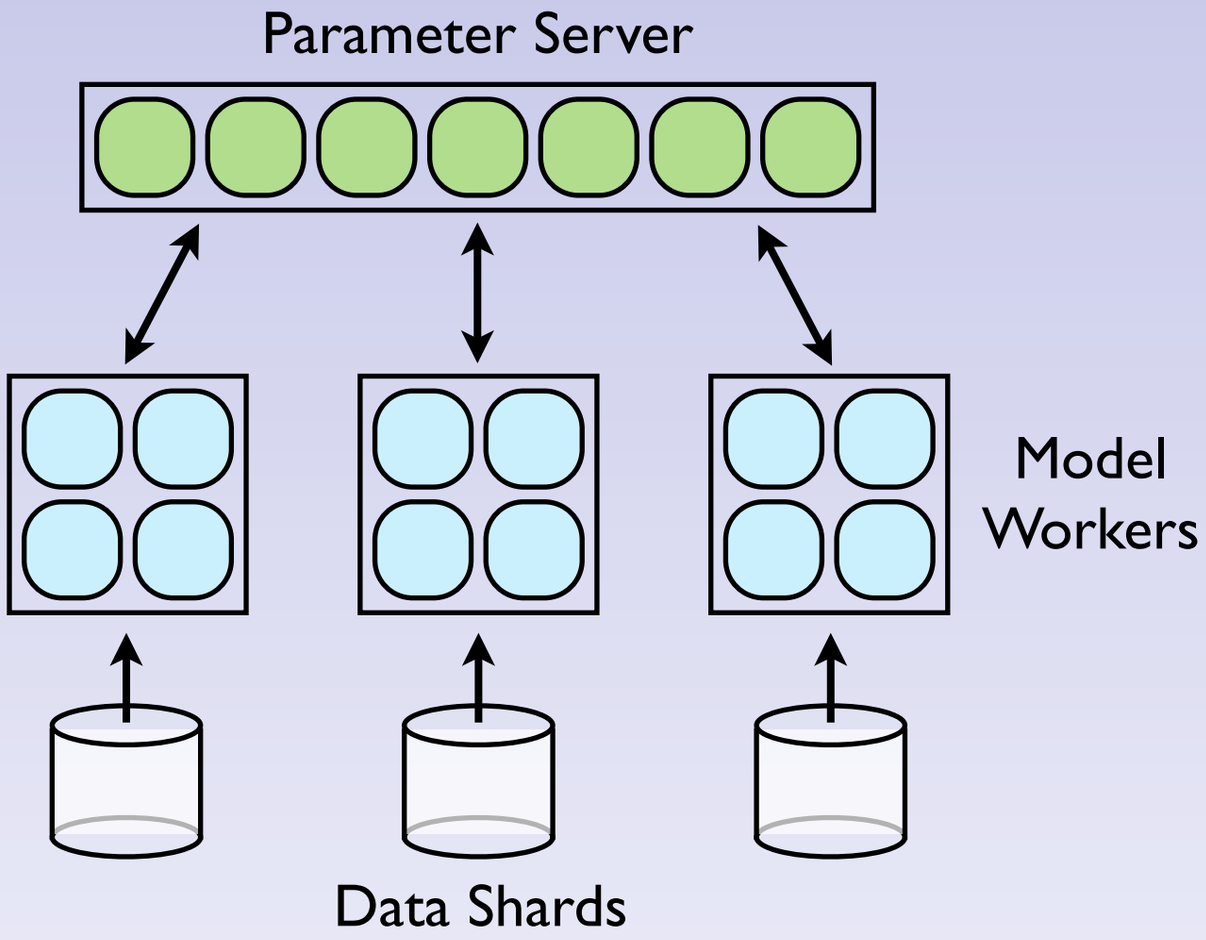
Basic DistBelief Model Training



- Unsupervised or Supervised Objective
- Mini-batch Stochastic Gradient Descent (SGD)
- Model parameters sharded by partition
- 10s, 100s, or 1000s of cores per model

Larger model + *more data* = better quality at test time.

Asynchronous SGD with AdaGrad



Significant gains by adding more models

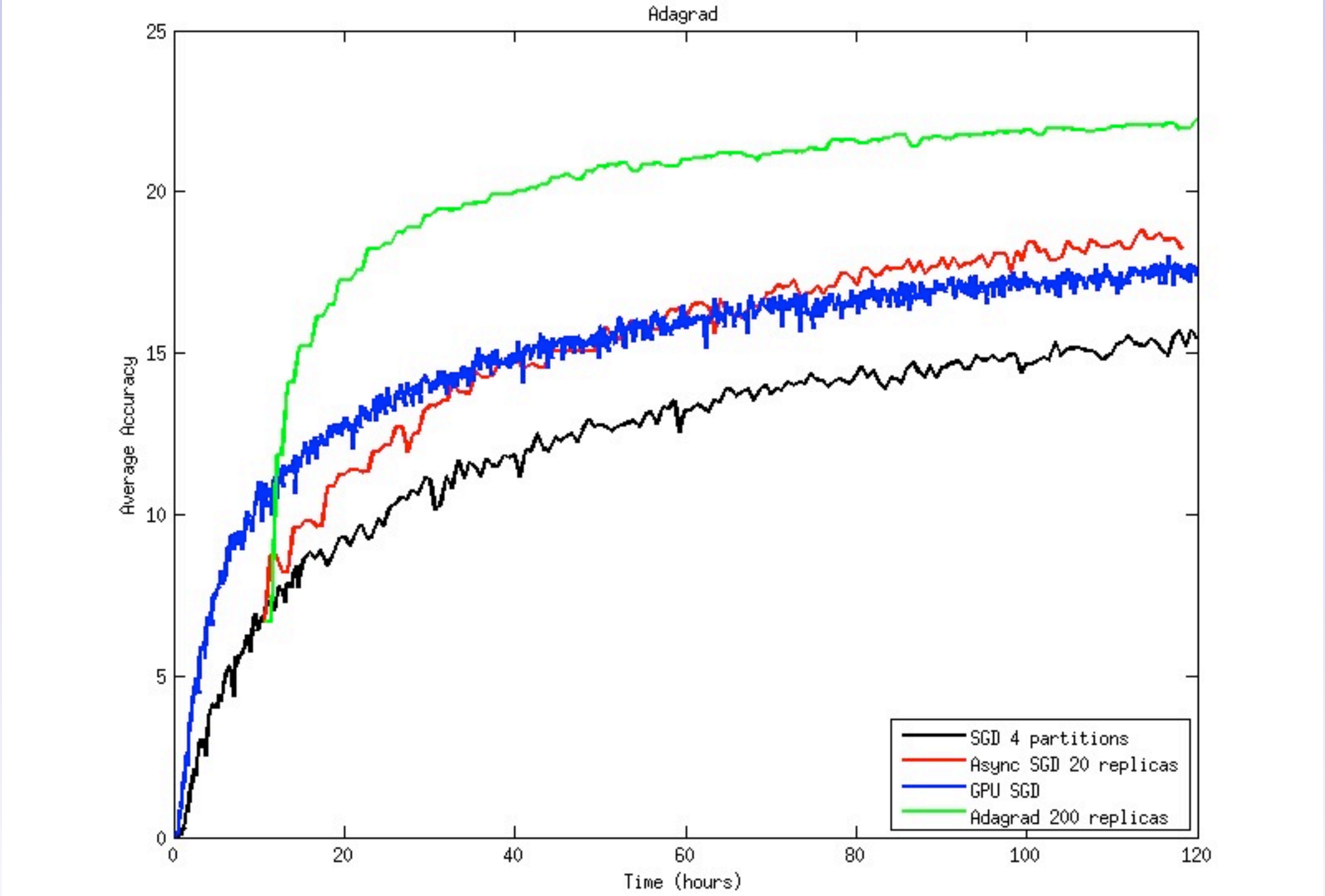
Number of models limited by optimization stability

AdaGrad: Adaptive Subgradient Methods for Online Learning and Stochastic Optimization
COLT 2010 Duchi, Hazan and Singer

Diagonal Approximation to the Hessian

Allows having 100 or more models

Asynchronous SGD with AdaGrad



To reach the same model quality DistBelief reached in 4 days took 55 days using a GPU

Learning on Youtube images

10M 200x200 images

1 billion parameters

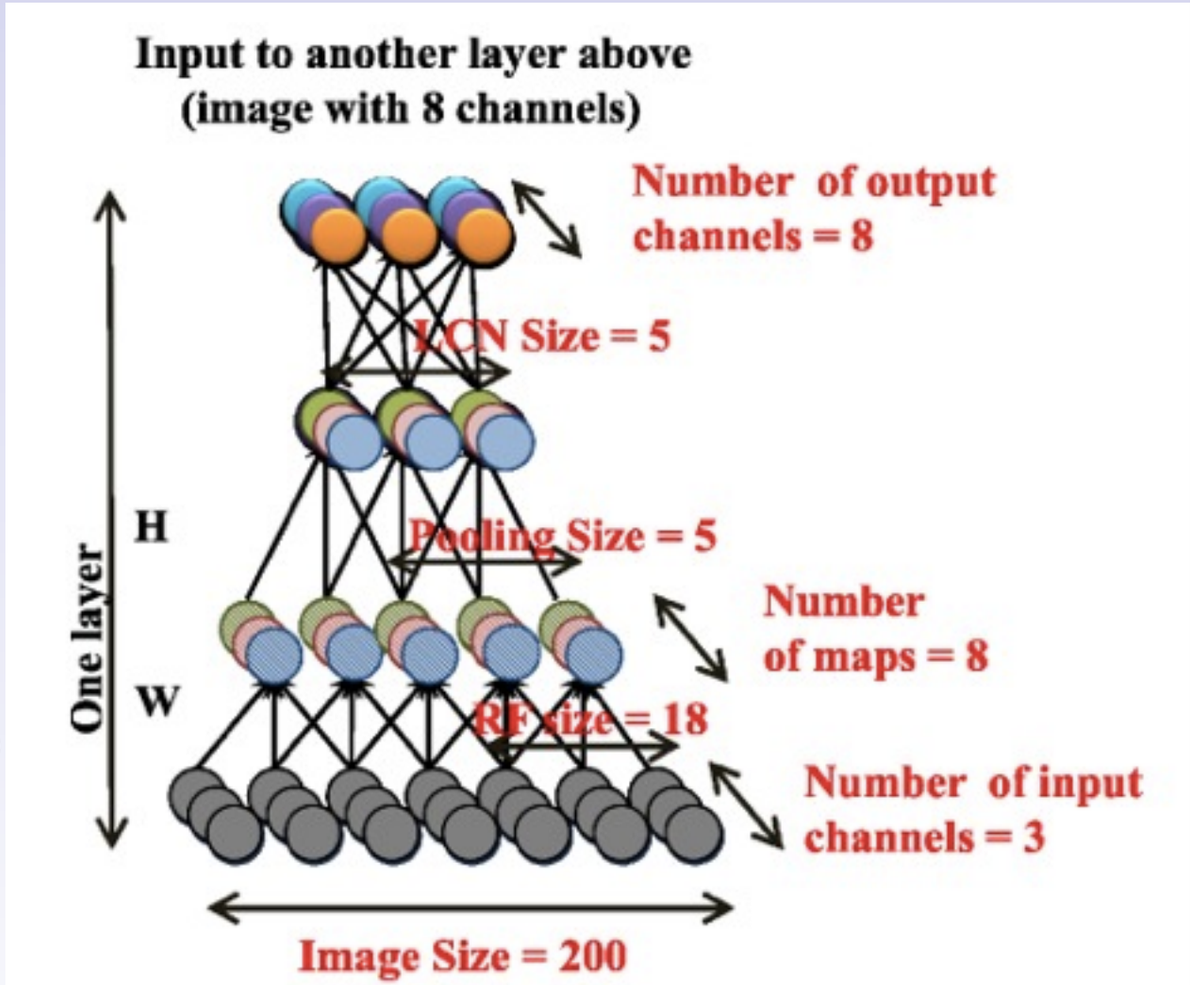
1000 machines (16,000 cores)

3 types of layer:

- Filtering (local receptive fields)
- Pooling
- Local Contrast Normalization (LCN)

3x3 = 9 layers

Unsupervised training



[Le et al. (2012)]

Learning on Youtube images

Figure 3. Top: Top 48 stimuli of the best neuron from the test set. Bottom: The optimal stimulus according to numerical constraint optimization.

[Le et al. (2012)]

Cats!

[Le et al. (2012)]

Learning on Youtube images

ImageNet

Dataset version	2009 (~9M images, ~10K categories)	2011 (~14M images, ~22K categories)
State-of-the-art	16.7% (Sanchez & Perronnin, 2011)	9.3% (Weston et al., 2011)
Our method	16.1% (without unsupervised pretraining) **19.2%** (with unsupervised pretraining)	13.6% (without unsupervised pretraining) **15.8%** (with unsupervised pretraining)

[Le et al. (2012)]

Hinton's recipe for training deep nets

- Use pretraining if few labeled data, else initialize W to sensible values
- Use LOTS of neurons
- Use rectified units
- Use dropouts
- If spatial structure, use convolutional front-end

Deep Learning and MIR

Adapting ML to MIR

- Occultation vs. interference
- Monophony vs. polyphony
- Scaling
- Long time dependencies

Instrument Recognition with DBNs

	SVM	MLP	DBN	%
Bass	0.88	0.88	0.88	13.85%
Brass	0.87	0.88	0.91	22.37%
Guitar	0.0	0.0	**0.21**	2.13%
Organ	0.96	0.89	0.96	7.46%
Piano	0.45	0.43	**0.57**	6.39%
Strings	0.94	0.95	0.97	9.59%
Woodwind	0.82	0.85	0.89	29.83%
Global	0.84	0.84	0.88	

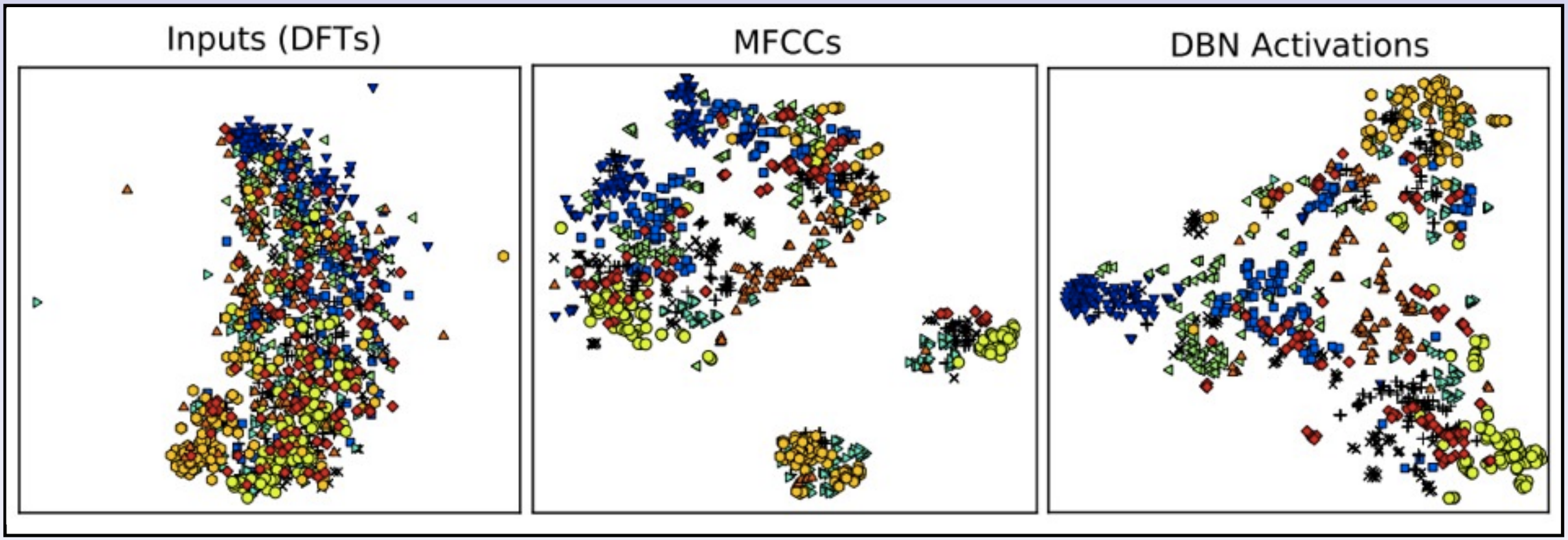
Solo instruments

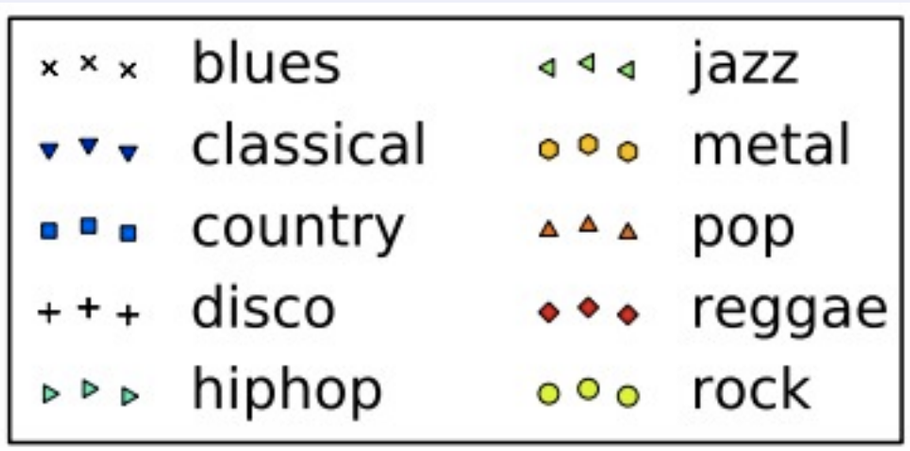
	SVM	MLP	DBN	%
Bass	0.86	0.83	0.85	50.00%
Brass	0.38	0.45	**0.63**	25.90%
Guitar	0.05	0.15	**0.28**	11.94%
Organ	0.84	0.84	0.85	62.99%
Piano	0.83	0.80	0.83	64.44%
Strings	0.37	0.37	0.36	18.82%
Woodwind	0.31	0.41	**0.52**	31.81%
Global	0.72	0.72	0.74	

Poly instruments

[Hamel et al. (2009)]

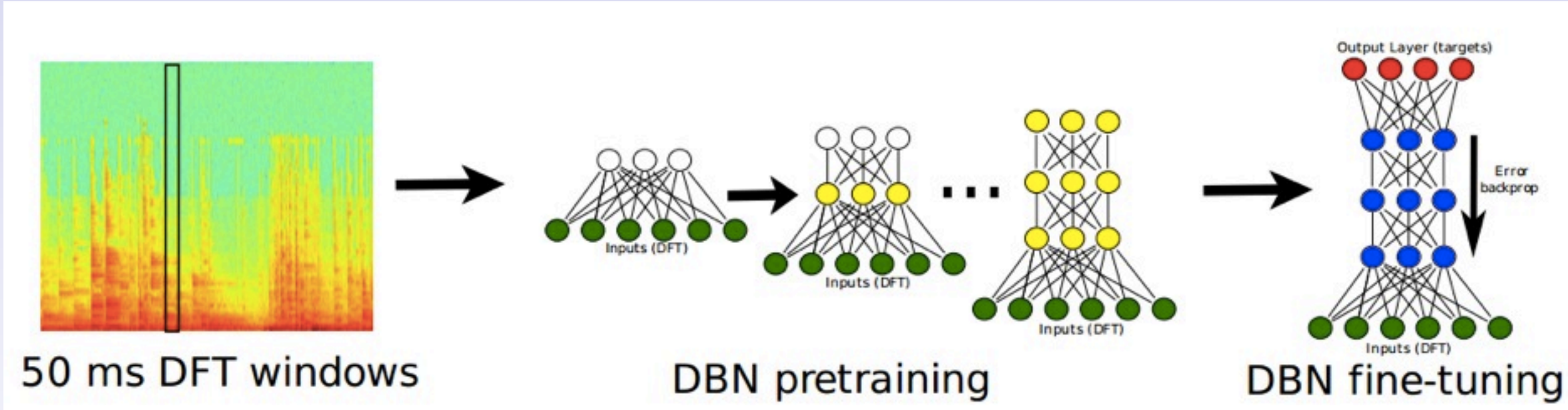
Learning audio features with DBNs





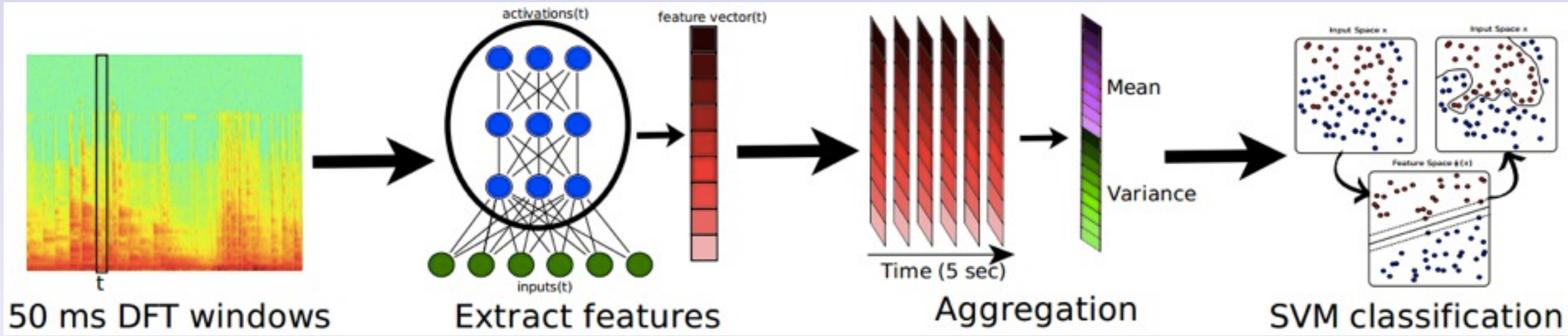
[Hamel et al. (2010)]

Learning audio features with DBNs



[Hamel et al. (2010)]

Learning audio features with DBNs



[Hamel et al. (2010)]

Learning audio features with DBNs

Genre recognition
(GTZAN)

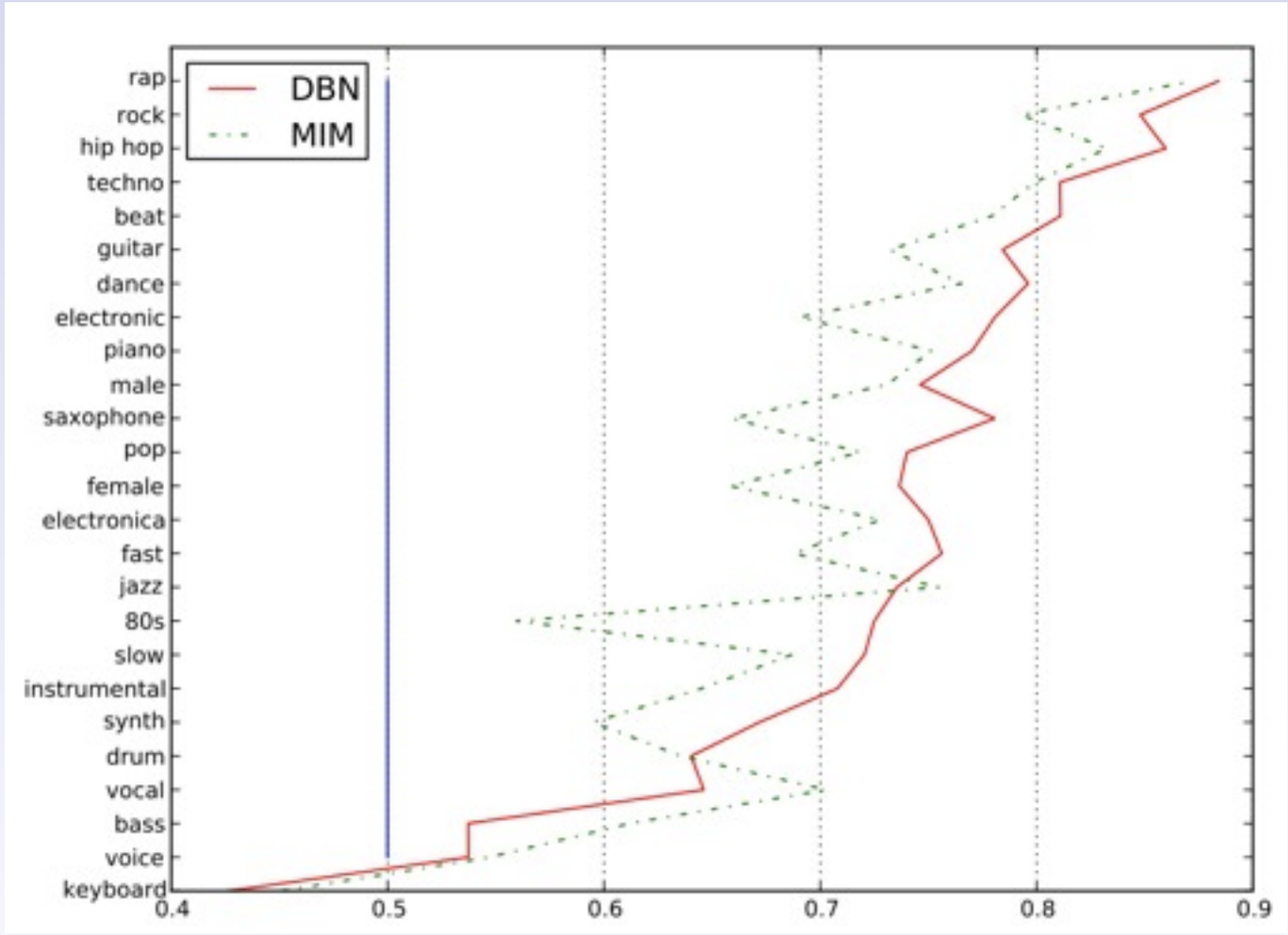
	Accuracy
MFCCs	0.790
Layer 1	0.800
Layer 2	0.837
Layer 3	0.830
All Layers	**0.843**

[Hamel et al. (2010)]

Learning audio features with DBNs

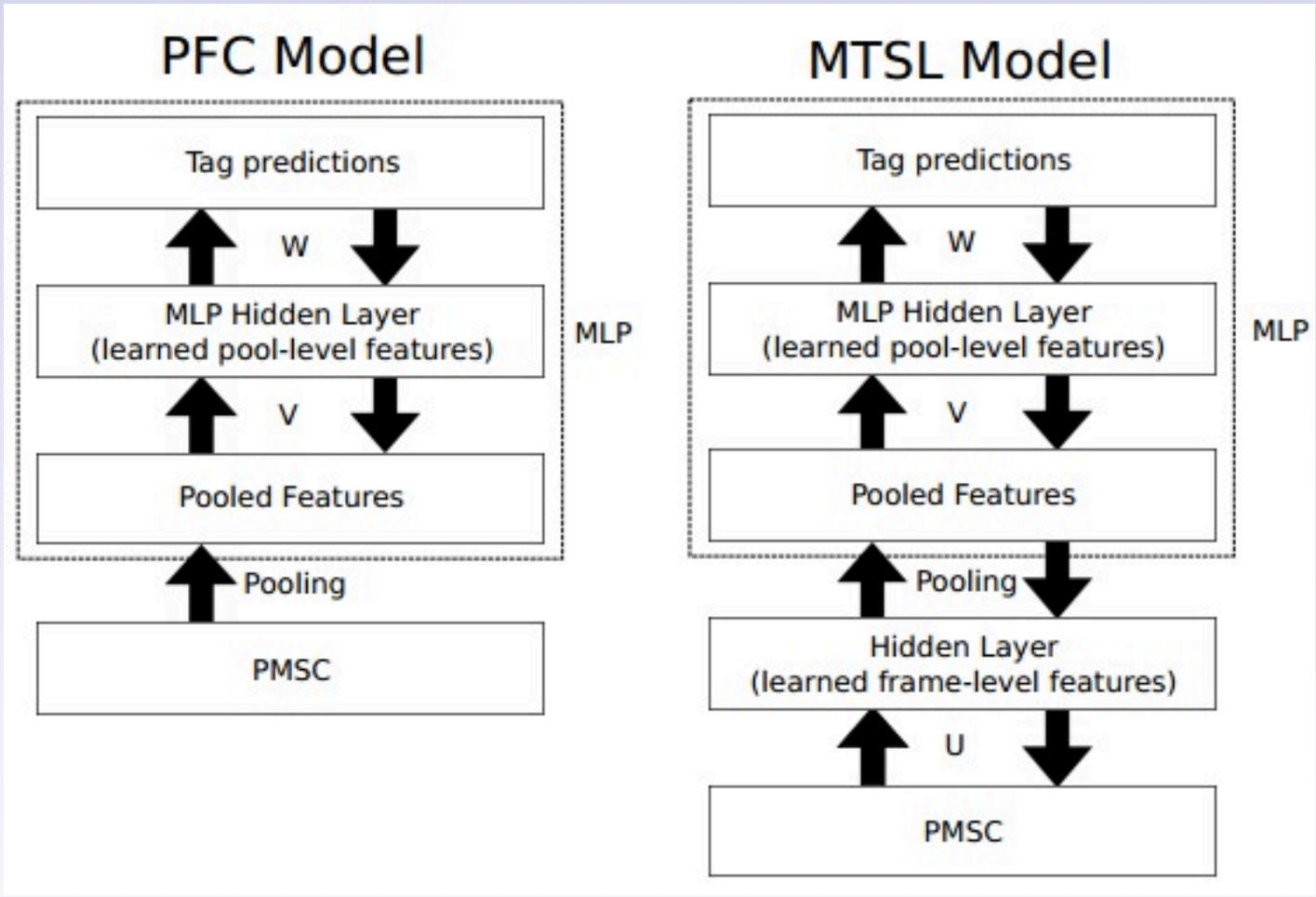
Auto-tagging (majorminer)

	Mean Accuracy	Standard Error
DBN	0.73	0.02
MIM	0.70	0.02



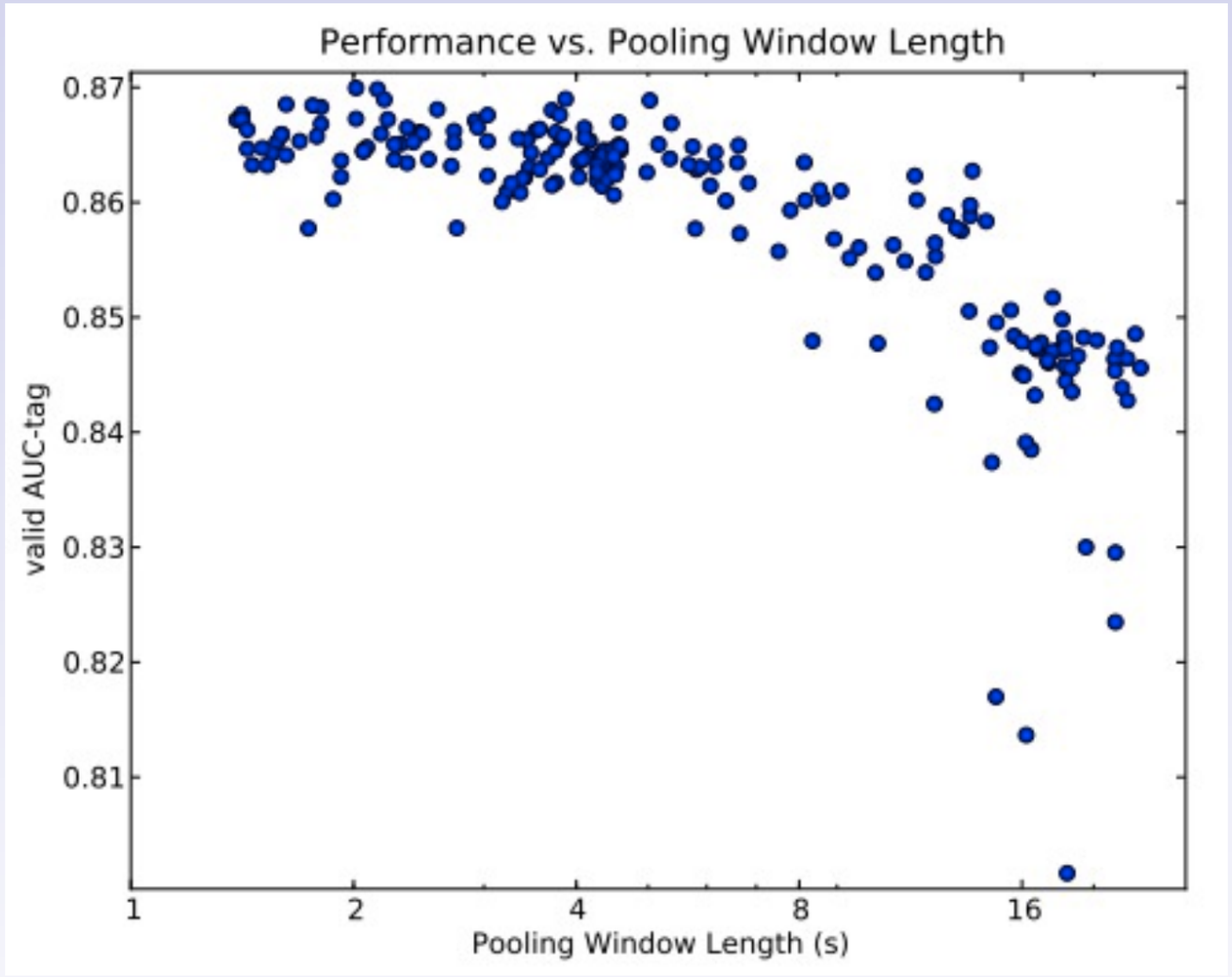
[Hamel et al. (2010)]

Multi-timescale learning



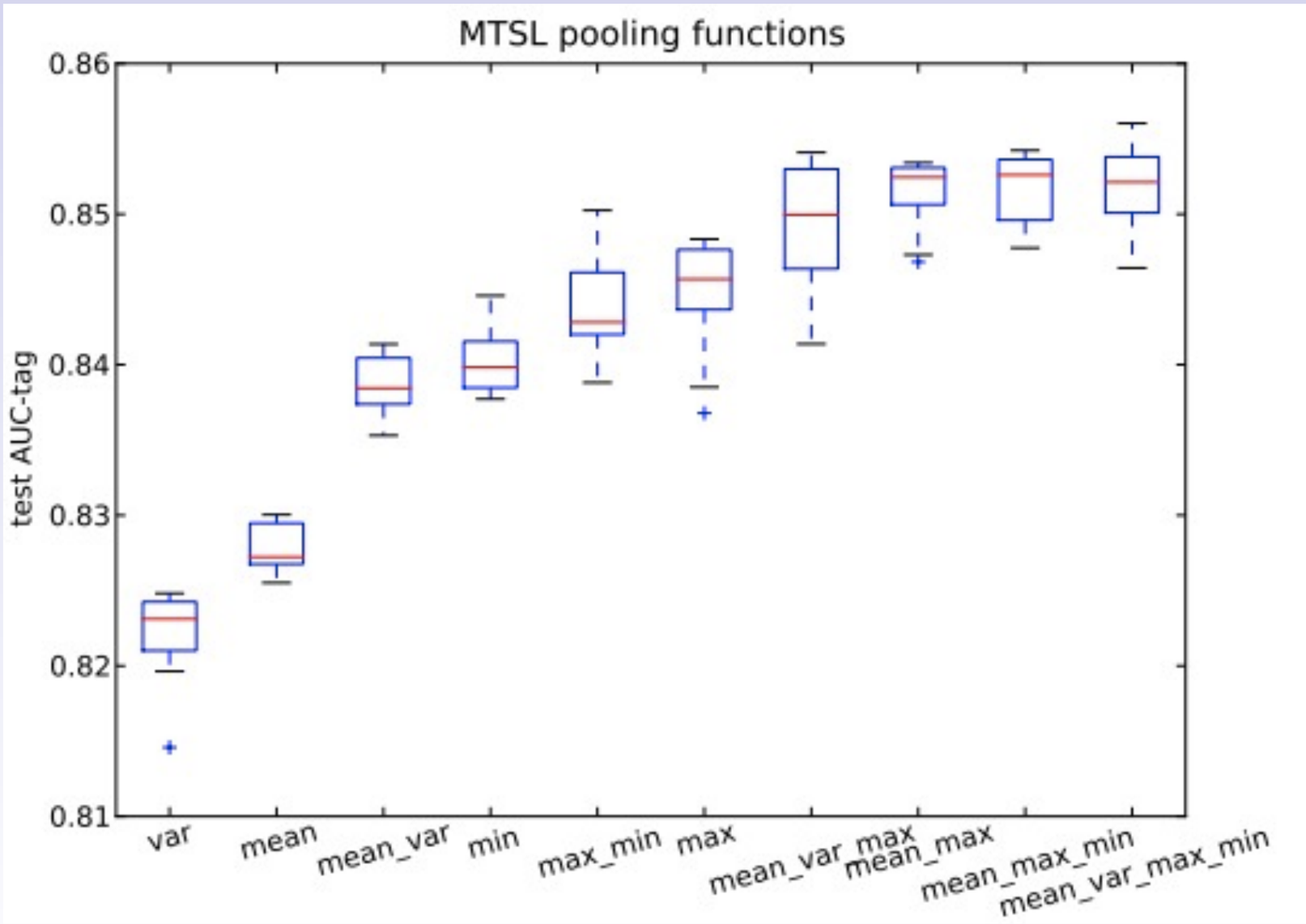
[Hamel et al. (2011)]

Multi-timescale learning



[Hamel et al. (2011)]

Multi-timescale learning



[Hamel et al. (2011)]

Multi-timescale learning

MagnaTagatune (compared to MIREX 2009)

Measure	Manzagol	Zhi	Mandel	Marsyas	Mel-spec+PFC	PMSC+PFC	PSMC+MTSL
Average AUC-Tag	0.750	0.673	0.821	0.831	0.820	0.845	**0.861**
Average AUC-Clip	0.810	0.748	0.886	0.933	0.930	0.938	**0.943**
Precision at 3	0.255	0.224	0.323	0.440	0.430	0.449	**0.467**
Precision at 6	0.194	0.192	0.245	0.314	0.305	0.320	**0.327**
Precision at 9	0.159	0.168	0.197	0.244	0.240	0.249	**0.255**
Precision at 12	0.136	0.146	0.167	0.201	0.198	0.205	**0.211**
Precision at 15	0.119	0.127	0.145	0.172	0.170	0.175	**0.181**

[Hamel et al. (2011)]

Deep learning in MIR

- Music Transcription
- Chord recognition
- Music recommendation
- Onset Detection
- Feature Learning
- Classification (genre, instrument, etc.)
- Autotagging

Deep learning in MIR

Audio chord recognition with RNNs [Boulanger-Lewandowsky et al. 2013]

Deep content-based music recommendation [van den Oord et al. (2013)]

Musical Onset Detection with CNN [Schlüter & Boch (2013)]

Learning Rhythm And Melody Features With DBNs [Schmidt & Kim (2013)]

Moving Beyond Feature Design: Deep Architectures and Automatic Feature Learning in Music Informatics [Humphrey et al. (2012)]

Onset detection with RNNs [Böck et al. (2012)]

Polyphonic piano transcription with DBNs [Nam et al. (2011)]

Audio-based music classification with a pretrained CNN [Dieleman et al. (2011)]