

# Parallelization of Neural Networks Training



**Dr. Volodymyr O. Turchenko** **short name (Vlad)**

**Ph.D. in Computer Engineering**

Associate Professor, FP7 Marie Curie Research Fellow

Research Institute of Intelligent Computer Systems

Ternopil National Economic University

3 Peremoga Square, Ternopil, 46004, UKRAINE

E-mail: [vtu@tneu.edu.ua](mailto:vtu@tneu.edu.ua)

Web: <http://uweb.deis.unical.it/turchenko/>

**FIELD: NEURAL  
NETWORKS and  
PARALLEL  
COMPUTING**

<b>Aug 2011 – Sep 2012</b>	FP7 Marie Curie Research Fellow, Head of Neural Network and Parallel Computing Research Group, Research Institute of Intelligent Computer Systems, Ternopil National Economic University (TNEU), Ukraine
<b>Apr 2009 – Jun 2011</b>	FP7 Marie Curie Research Fellow, Department of Electronics, Informatics and Systems, University of Calabria, Italy
<b>Mar 2002 – Apr 2009</b>	Assistant, then Associate Professor of the Information Computing System and Control Department, Ternopil National Economic University, Ukraine
<b>November 2001</b>	Ph.D. in Computer Engineering, National University “Lviv Polytechnics”, Lviv, Ukraine
<b>June 1995</b>	Dipl. Eng. (honors) of System Engineering, Brest Polytechnic Institute, Brest, Belarus

# Fulbright Program



- Started in 1946 to “promotion of international good will through the exchange of students in the fields of education, culture and science”, approximately 300,000 people in 155 countries have participated, today, Fulbright is the most widely recognized and prestigious international exchange program in the world
- While each grantee has a specific teaching, research or professional project to pursue, it is important to recognize that the ultimate goal of the Fulbrighter is to promote mutual understanding and respect between the United States and other nations
- Fulbrighters are “cultural ambassadors” of their home countries, their goals are (i) to meet as many people as possible by speaking and writing about their home countries to interested groups and (ii) to learn about the history and culture of the host country
- The Fulbright Program offers grants to study, teach and conduct research for U.S. citizens to go abroad and non-U.S. citizens to come to the United States
- **Fulbright U.S. Scholar Program**  
<http://fulbright.state.gov/grants/scholar-program/us-citizen.html>
- Deadline is **Oct 30, 2012** to send an application for the **2013/14** educational year
- Duration: **12 months**

# Ukraine - <http://en.wikipedia.org/wiki/Ukraine>

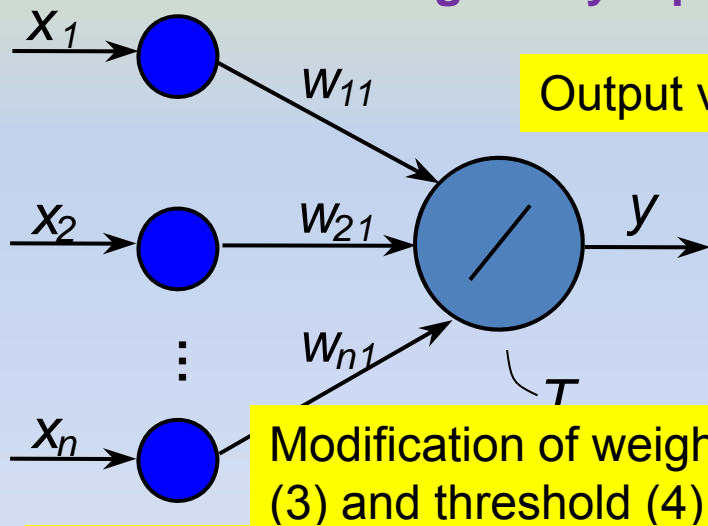


- An area of 603,628 km<sup>2</sup>, making it the second largest contiguous country on the European continent (after European part of Russia)
- Population is 45 million people, major cities are: Kyiv (capital), Kharkiv, Odessa, Dnipropetrovsk, Donetsk, Zaporizhzhia, Lviv
- Ukraine just co-hosted [UEFA Euro 2012](#) soccer championship alongside [Poland](#) in June 2012
- Rich history, georghaphy, culture and traditions
- What to see: The [Seven Wonders of Ukraine](#) are the seven historical and cultural monuments of Ukraine, Carpathian Mountains, Black and Azov Sea, Dnipro river, Ukrainian cuisine has a long history and offers a wide variety of original dishes

# Neural Network Description - 1

- excellent **abilities to model** difficult nonlinear systems
- very good alternative to traditional methods for solving complex problems in many fields, including **image processing, pattern recognition, robotics**, etc
- solvers of intelligent tasks - **prediction, classification, recognition, optimization**, etc
- good **generalized properties, self-adaptation** allow considering NNs as universal tools

## Single - layer perceptron: Formal Neuron



Output value (1)

$$y = \sum_{i=1}^n w_{i1} x_i - T$$

SSE value (2)

$$E = \frac{1}{2} (y^k - d^k)^2$$

Goal of training:

$$E(t) \rightarrow \min$$

Modification of weights (3) and threshold (4)

$$w_{i1}(t+1) = w_{i1}(t) - \alpha(t) \frac{\partial E(t)}{\partial w_{i1}(t)}$$

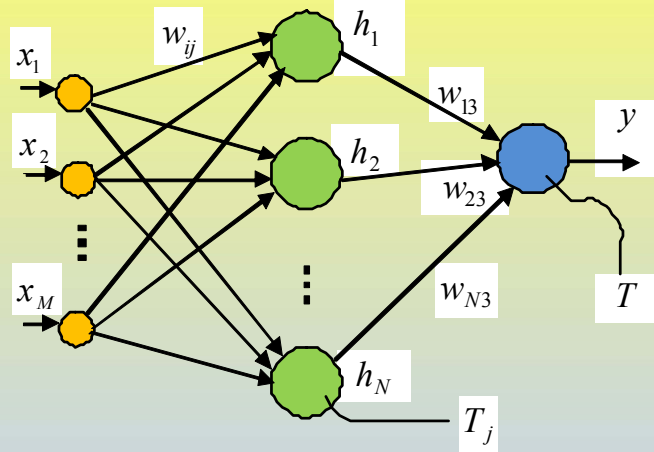
$$T(t+1) = T(t) + \alpha(t) \frac{\partial E(t)}{\partial T(t)}$$

### A simple Widrow-Hoff training algorithm

- 1) to define the value of  $\min \text{SSE}_{\text{required}}$ ;
- 2) to initialize randomly the weights and threshold;
- 3) to calculate neuron output according to (1);
- 4) to calculate SSE according to (2);
- 5) to modify weights and threshold using (3-4);
- 6) to make steps 3-5 until  $\text{SSE} > \text{SSE}_{\text{required}}$ .

# Neural Network Description - 2

## Multi-layer perceptron N-M-1



Output value of the perceptron

$$y = F_3 \left( \sum_{j=1}^N w_{j3} \left( F_2 \left( \sum_{i=1}^M w_{ij} x_i - T_j \right) \right) - T \right)$$

Sigmoid activation function of hidden layer neurons

$$F(x) = \frac{1}{1 + e^{-x}}$$

Goal of training:  $E(t) \rightarrow \min$

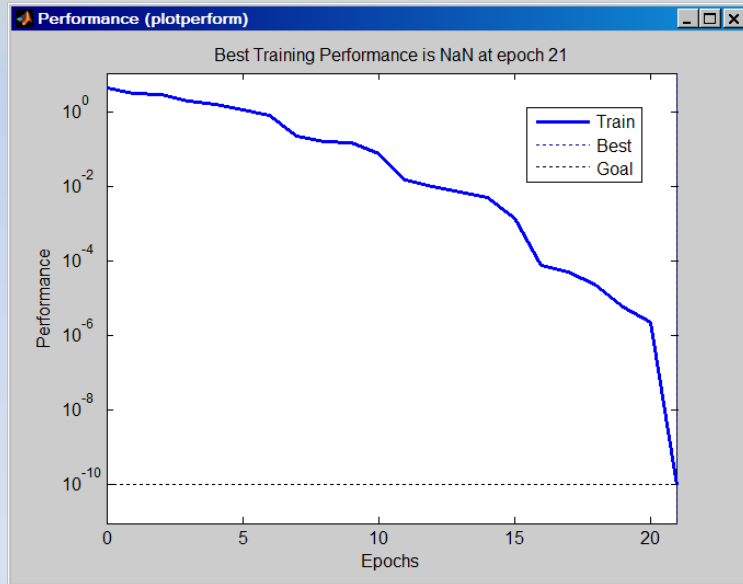
Modification of weights and thresholds

$$\Delta w_{ij}(t) = -\alpha \frac{\partial E^P(t)}{\partial w_{ij}(t)}$$

$$\Delta T_j(t) = -\alpha \frac{\partial E^P(t)}{\partial T_j(t)}$$

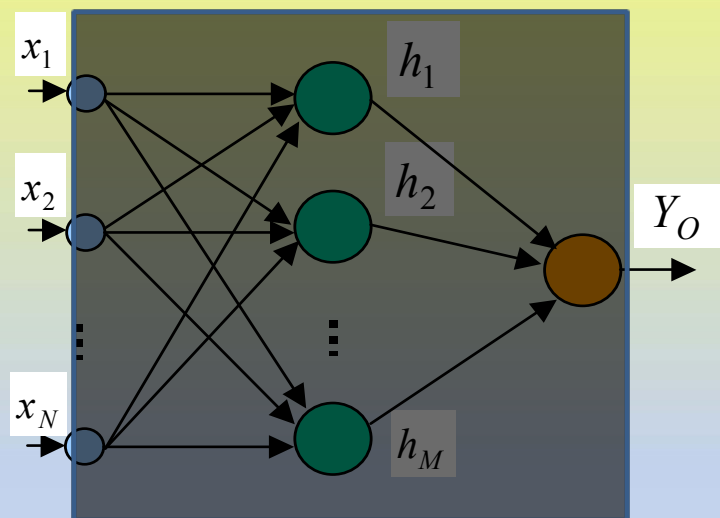
There are many gradient-based training algorithms, which provides **local minimum** near starting point: Back-propagation (Rumelhart et al. 1986), conjugate gradient, Newton's method, the DFP algorithm (Davidon 1959; Fletcher and Powell 1963), the Levenberg-Marquardt algorithm (Levenberg 1944; Marquardt 1963), and the BFGS algorithm

Amato et al. (1991) and Owen and Abunawass (1993) applied simulated annealing (SA) technology (**global minimization**) for complex learning tasks



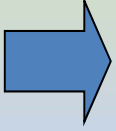
# Neural Network Description – 3: how to use

5.81	5.84	6.27	6.50	6.80	6.95	7.20	7.28	7.40	7.80	8.34
------	------	------	------	------	------	------	------	------	------	------

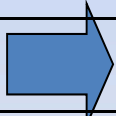


Multi-layer perceptron N-M-1

Training set: 3-(M=3)-1

k	$x_1^k$	$x_2^k$	$x_3^k$		$d^k$
1	5.81	5.84	6.27		6.50
2	5.84	6.27	6.50		6.80
3	6.27	6.50	6.80		6.95
4	6.50	6.80	6.95		7.20
5	6.80	6.95	7.20		7.28
6	6.95	7.20	7.28		7.40
7	7.20	7.28	7.40		7.80
8	7.28	7.40	7.80		8.34

Prediction set:

1	7.40	7.80	8.34		???
---	------	------	------	--	-----

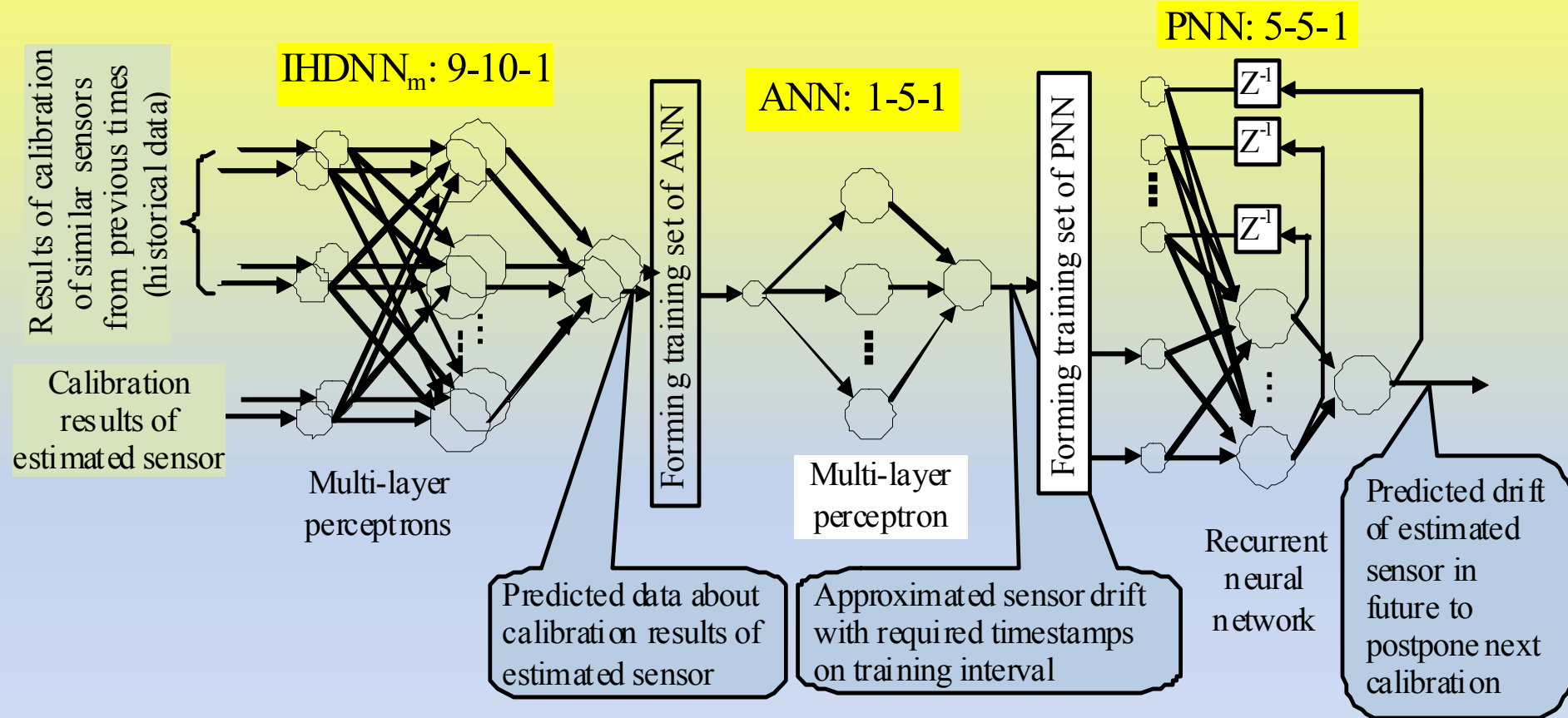
➤ NNs can require a high computational load in the training phase

➤ High computational load and large training time make worse wide NN application for scientific research and real-time systems

➤ Therefore NN parallelization is one of the technologies, which could outperform this disadvantage



# Applications: Sensor Drift Prediction



Ensemble of Integrating Historical Data Neural Network (IHDNN), Approximating Neural Network (ANN) and Predicting Neural Network (PNN) for sensor drift prediction

The application of neural networks allows improving the accuracy of the sensor data gathering in several times (**3-5 times**) on increased duration of inter-calibration interval (**6-12 times**) using small number of the input data of sensor drift as a training set (**1-5 values of sensor drift**)

# Applications: Vehicle license plate recognition



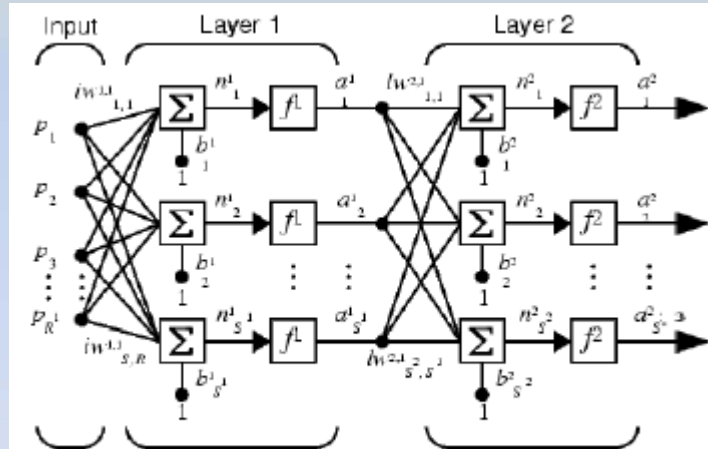
Original image



Deblurred image



Thresholded image



Neural Network Structure

Real value	Noised inputs	Classification Results
4		4
7		7
6		6
5		5
T		T
E		E
P		P

366-50-46

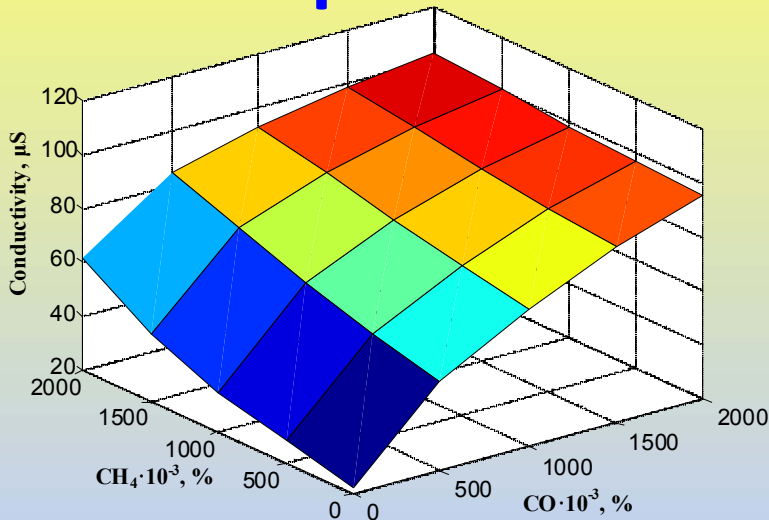
NN can correctly recognize the characters on license plate with probability of 95% in presence of noise with 50% density



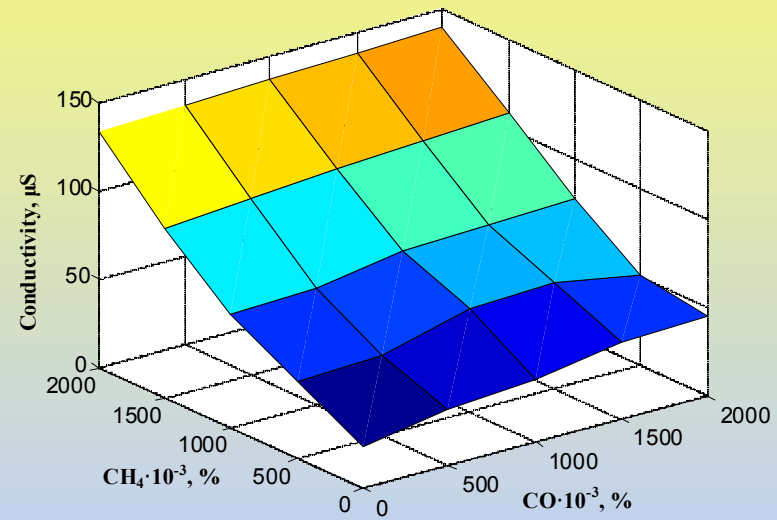
# Applications: Recognition of output signal of multi-parameter sensor



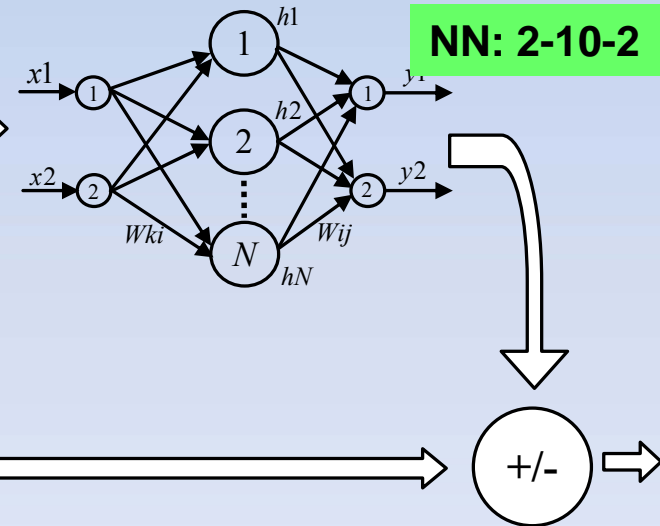
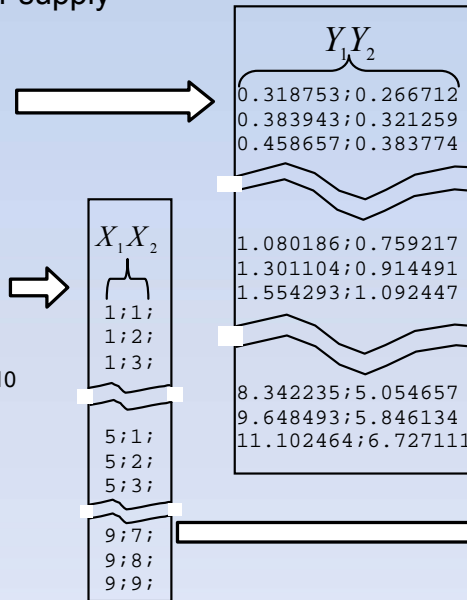
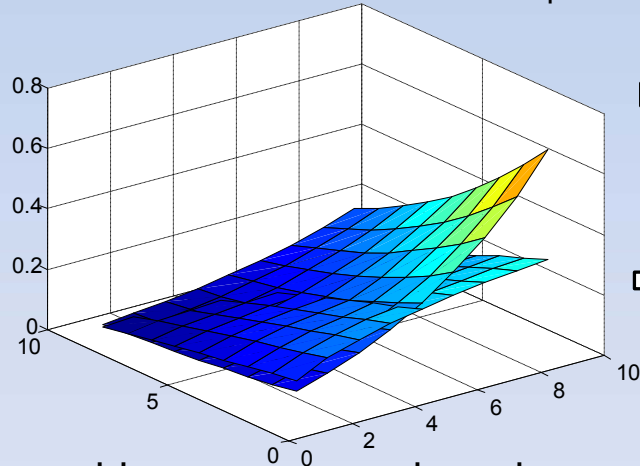
Sensor  
TGS-813  
Figaro, Inc  
(USA)



1st mode – 4V power supply

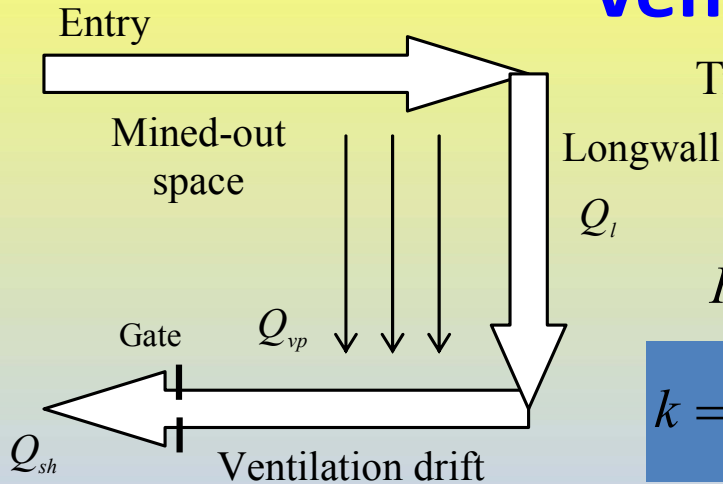


2nd mode – 5V power supply



NN provides average and maximum recognition errors not more than 1% and 2% respectively at  $SSE=10^{-5}$

# Applications: Forming of control influences for mine ventilation network



Transient aerodynamic process of airflow forming in the entry

$$\frac{dQ}{dt} = \frac{1}{k} (H - R \cdot Q \cdot |Q| - RR \cdot Q \cdot |Q|)$$

$H$  is a depression,  $R$  and  $RR$  are aerodynamic resistances

$$k = \frac{\rho \cdot l}{S} \quad \text{is inertia coefficient}$$

$$\frac{dC_{vp}}{dt} = \frac{1}{V_{vp}} (Q_m - (Q_m + Q_{vp}) \cdot C_{vp})$$

The debit of air-methane mixture  $Q_{vp}$

A model of aerogas environment of longwall is

$$\frac{dC_l}{dt} = \frac{1}{V_l} (Q_{ml} - (Q_l + Q_{ml}) \cdot C_l)$$

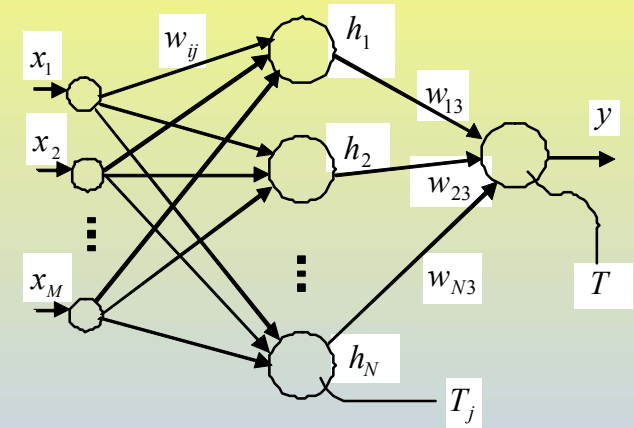
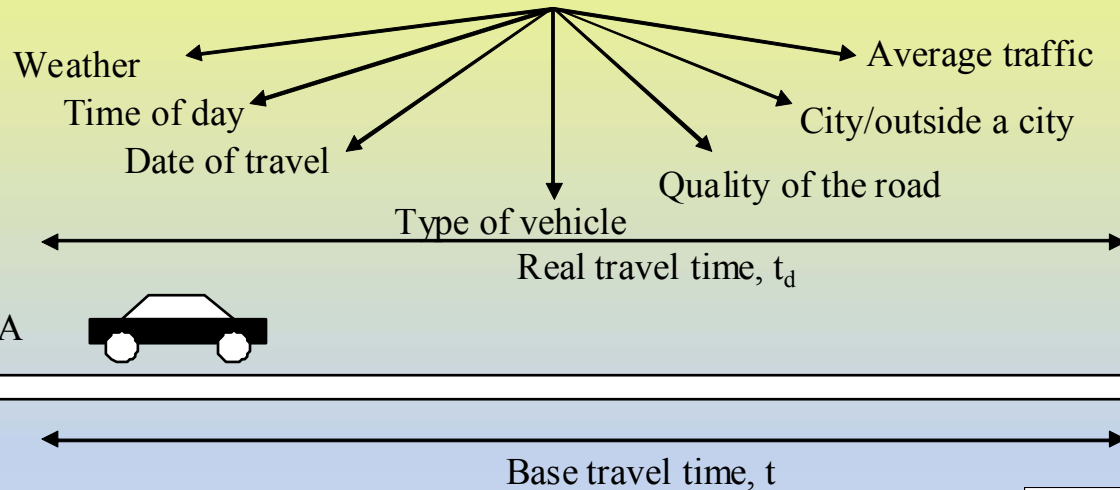
The resulting flow of air-methane mixture  $Q_{sh}$  in the ventilation drift with methane concentration  $C_{sh}$

$$\frac{dC_{sh}}{dt} = \frac{1}{V_{sh}} (Q_{md} + Q_{mld} - (Q_{sh} + Q_{mld} + Q_{md}) \cdot C_{sh})$$

**Neural - based Exponential Control Influence** – growing methane concentration to **1%** in mined-out space and **0.3%** in ventilation drift during transient aero-gas-dynamic process

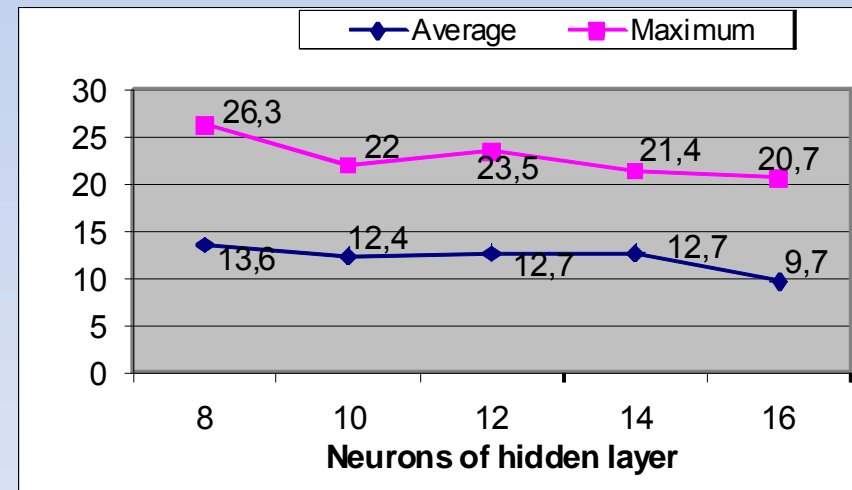
# Applications: Prediction of vehicle arrival time for Vehicle Routing Problem

## Influence factors on vehicle travel time



Multilayer perceptron **8-5-1**

N°	Base travel time	Weather	Time	Date	Quality	Traffic	City/Outside	Type of vehicle	Estimated real time
1	15,00	1	1	1	1	1	1	1	15,00
2	15,00	1	3	1	1	1	1	1	18,50
3	15,00	1	5	1	1	1	1	1	25,00
4	15,00	1	3	1	4	1	1	1	30,00
5	15,00	3	1	1	1	1	1	1	20,00
6	15,00	3	3	1	4	1	1	1	40,00
7	15,00	5	1	1	1	1	1	1	35,00
8	15,00	1	1	3	1	1	1	1	25,00
9	15,00	1	1	5	1	1	1	1	25,00
10	15,00	3	2	3	2	1	1	1	30,00
11	15,00	1	3	3	4	1	1	1	20,00
12	15,00	1	3	3	4	2	1	1	22,00
13	15,00	1	3	3	4	2	2	1	25,00
14	15,00	1	3	3	4	3	1	1	25,00



Dependence of maximum and average relative prediction errors from the number of hidden layer neurons

# Applications: Prediction of arrival time of interplanetary shocks

**Interplanetary shocks** are the regions created by supersonic gas flow with sharp differences of gas density, pressure, temperature, ionization and other its parameters

The **energetic storm particle** (ESP) events are associated with interplanetary shocks

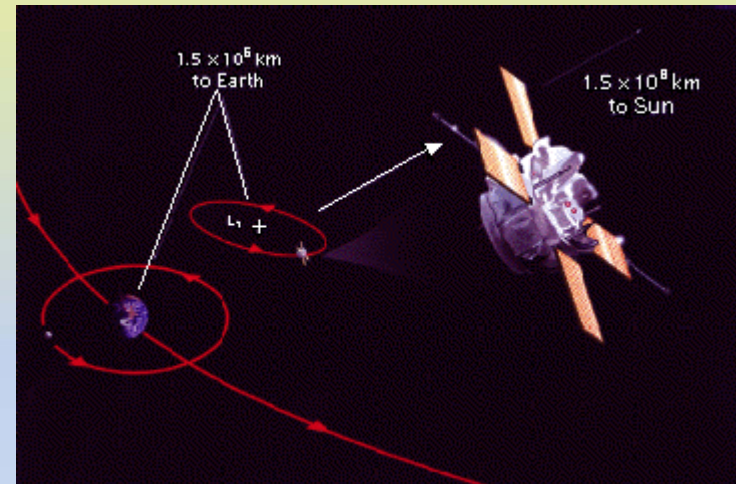
They negatively influence on:

- spacecrafts and satellites on a low-orbital Earth's orbit
- terrestrial high-frequency radio devices
- electrical grids and electrical power systems
- people's health

Satellites **problems** caused by IP and ESP events:

- two Canadian satellites ANIK E-1 and E-2 were disabled, January 20-21, 1994
- AT&T experienced a massive power failure in its Telstar 401 satellite, January 11, 1997
- ASKA Japanese communication satellite was lost, July 2000

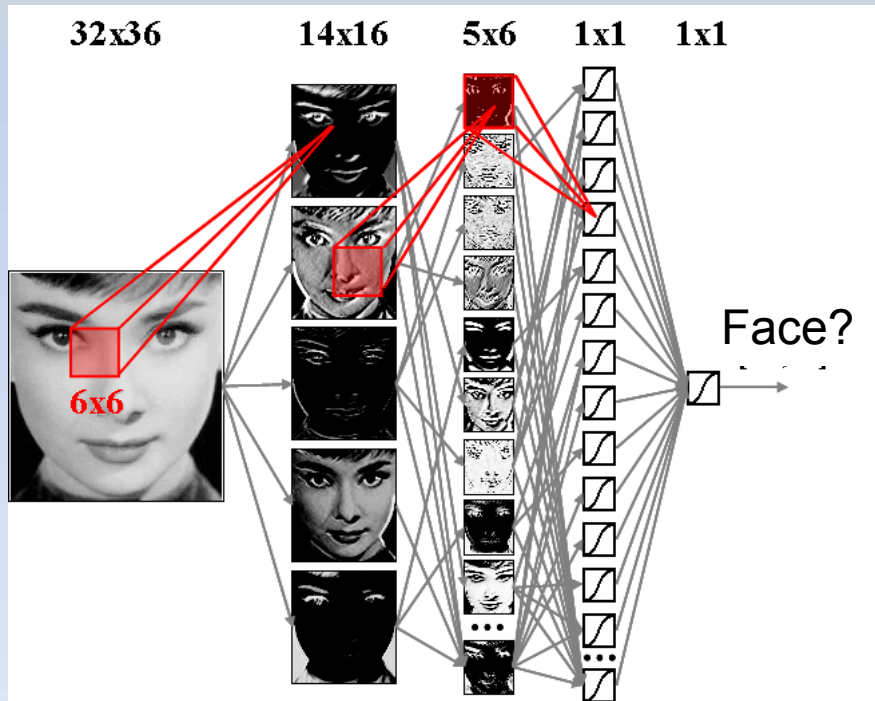
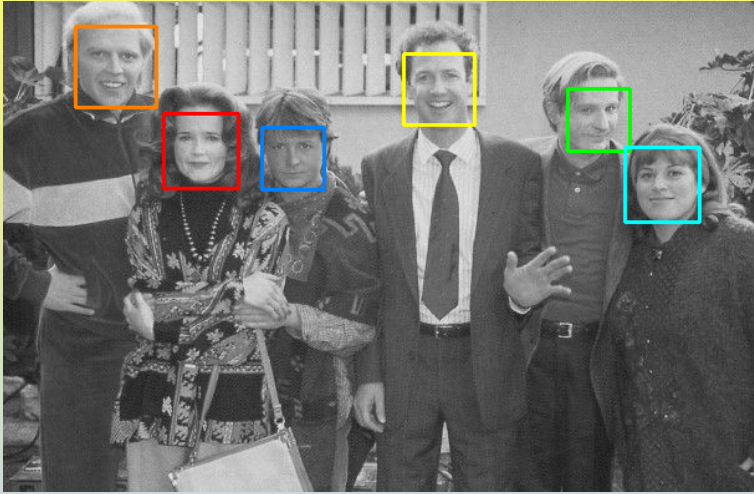
ACE (Advanced Composition Explorer) NASA spacecraft is stationed in Lagrange point L1 at the distance about 1.5 million km from the Earth



- ❑ Electron, Proton, and Alpha Monitor (EPAM) instrument of ACE spacecraft provides the data characterize the dynamic behavior of electrons and ions with  $\sim 0.03$  to  $\sim 5$  MeV
- ❑ The data are automatically generated by the spacecraft and published in real-time at [http://www.srl.caltech.edu/ACE/ASC/level2/lvl2DATA\\_EPAM.html](http://www.srl.caltech.edu/ACE/ASC/level2/lvl2DATA_EPAM.html)

NN provides high prediction accuracy (5.84% of average relative prediction error) on the 11 historical events of 2000

# Applications: Face detection and recognition

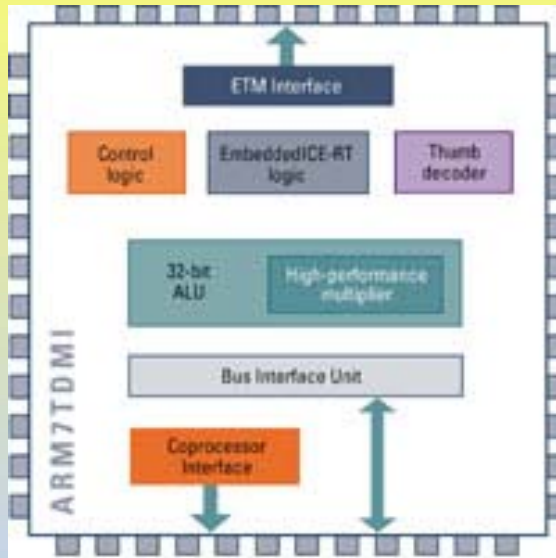


Face detection method	Detection level, %	Number of false detections
Viola P., Jones M.	76.1	10
Rowley H. et al	76.9	8
Nilsson M. et al	81.0	10
Li S. Z., Zhang Z. Q.	83.6	10
Kudriashov P.	83.0	7
Feraud R.	86.0	8
<b>Combined Cascade of Classifiers, Paliy I.</b>	<b>88.4</b>	<b>8</b>
Garcia C., Delakis M.	90.5	10

The developed combined cascade of classifiers is inferior on 2% of accuracy only to Garcia C., and Delakis M. method, but it overperforms them by 8 time faster processing speed



# Applications: Prediction of instruction's energy of embedded mChips

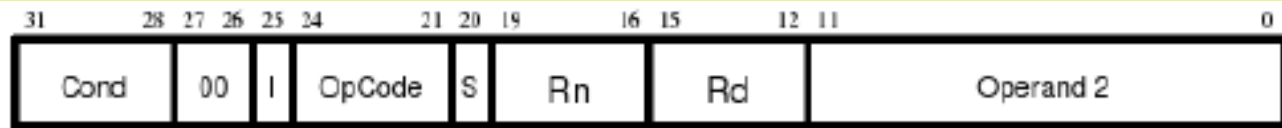


Applications: (i) Personal audio (MP3, WMA, AAC players), (ii) Entry level wireless handset, (iii) Pager, (iv) Ink-jet/bubble-jet printer, (v) Digital still camera

The experiments shown that:

- ❖ we should apply some sorting methodology to pre-process input data and divide them onto the groups and prepare them to NN
- ❖ the prediction should be done within each data instruction group separately

Prediction of energy allows developing the routines with low energy usage



All instructions of ARM7TDMI will be divided on the following groups according to the scheme of energy using:

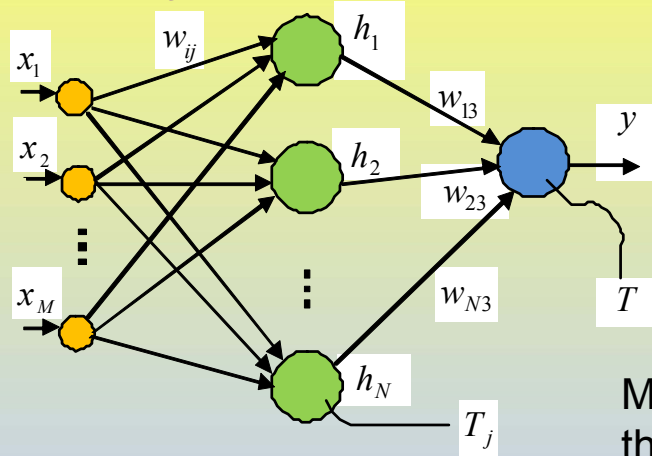
1. ADC, ADD, MVN
2. BIC, RSB
3. CMN, MOV
4. CMP, ORR, SBC, SUB
5. AND, EOR, RSC, TEQ, TST

NN structure: 11-10-1	Number of the experiment		
	1	2	3
Vectors for training	30	20	10
Vectors for predicting	10	20	30
Mean square error	$0,9 \cdot 10^{-5}$	$0,4 \cdot 10^{-5}$	$0,2 \cdot 10^{-5}$
Epochs amount	60794	21994	24198
Average training error	0,4%	0,3%	0,3%
Maximum training error	1,4%	0,8%	1,1%
Average predicting error	1,2%	1,3%	3,4%
Maximum predicting error	2,9%	5,5%	11,2%



# Neural networks and their parallelization

## Multi-layer perceptron N-M-1



Output value of the perceptron

$$y = F_3 \left( \sum_{j=1}^N w_{j3} \left( F_2 \left( \sum_{i=1}^M w_{ij} x_i - T_j \right) \right) - T \right)$$

Sigmoid activation function of hidden layer neurons

$$F(x) = \frac{1}{1 + e^{-x}}$$

Goal of training:

$$E(t) \rightarrow \min$$

Modification of weights and thresholds

$$\Delta w_{ij}(t) = -\alpha \frac{\partial E^P(t)}{\partial w_{ij}(t)}$$

$$\Delta T_j(t) = -\alpha \frac{\partial E^P(t)}{\partial T_j(t)}$$

**connection** parallelism (parallel execution on sets of weights)

## Levels of parallelism

**node** parallelism (parallel execution of operations on sets of neurons)

**pattern** (exemplar) parallelism (parallel execution of operations on set of NN's training patterns)

example (**modular**) parallelism (parallel execution of examples on replicated networks)

- ❖ first three levels are **fine-grain** parallelism & the fourth level is **coarse-grain** parallelism
- ❖ **Fine-grain** parallel algorithms require a lot of low-level communications, for example to combine the results of parallel calculation of weights of each neuron
- ❖ **Coarse-grain** algorithms carry out main computation operations on processors of parallel machine. The communications are rarely executed taking into account functional sense of the algorithm

# Development of batch pattern “fine-grain” level of parallelism



## Procedia Computer Science

Volume 1, Issue 1, May 2010, Pages 525–533

ICCS 2010



International Conference on Computational Science, ICCS 2010

## Improvement of parallelization efficiency of batch pattern BP training algorithm using Open MPI

Volodymyr Turchenko<sup>a</sup>,  , Lucio Grandinetti<sup>a</sup>, George Bosilca<sup>b</sup>, Jack J. Dongarra<sup>b</sup>

<sup>a</sup> Department of Electronics, Informatics and Systems, University of Calabria, via P. Bucci, 41C, 87036, ITALY

<sup>b</sup> Innovative Computing Laboratory, The University of Tennessee, 1122 Volunteer Blvd. Knoxville, TN 37996, USA

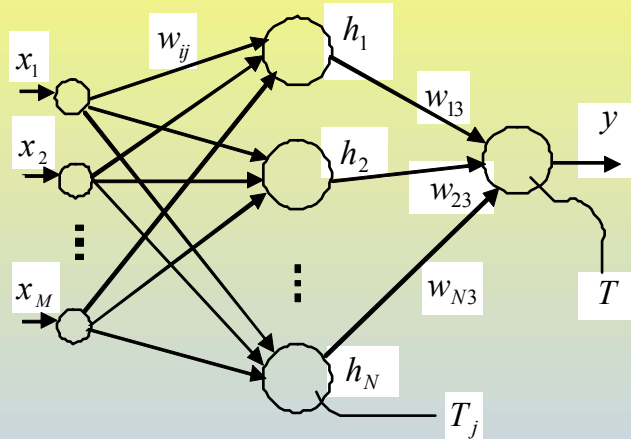
<http://dx.doi.org/10.1016/j.procs.2010.04.056>, How to Cite or Link Using DOI

 Permissions & Reprints

# Recent solutions

- **Vin et al (2004)** and **Kramer et al (2005)** described the development of grid-based frameworks for NNs parallelization, however they do not deal with parallelization efficiency issues
- **De Llano** and **Bosque (2010)** investigate parallel training of **multi-layer perceptron** (MLP) on SMP computer, cluster and computational grid using MPI parallelization. However their implementation of relatively small MLP architecture 16-10-10-1 (270 connections) does not provide positive parallelization speedup
- **Cernansky (2009)** developed a parallel training algorithm of Elman's simple **recurrent neural network** (RNN) based on Extended Kalman Filter on multicore processor and GPU. The reduction of the RNN training time using a GPU solution was showed (4 times), but it is impossible to assess the parallelization efficiency since the number of GPU threads was not clear
- **Lotric** and **Dobnikar (2009)** developed a parallel training algorithm for fully connected RNN based on linear reward penalty correction scheme. However their results show very limited speedup on the computational cluster and could not be normally assessed on GPU

# Sequential batch pattern training algorithm of MLP



**Fig. 1.** The structure of a three-layer perceptron

$$y = F_3 \left( \sum_{j=1}^N w_{j3} \left( F_2 \left( \sum_{i=1}^M w_{ij} x_i - T_j \right) \right) - T \right)$$

We calculated the delta weight and thresholds for **all training patterns** sequentially according to 3.4 and then update **all weights by one operation** in point 5 - this is a **precondition** for parallelizing of this algorithm

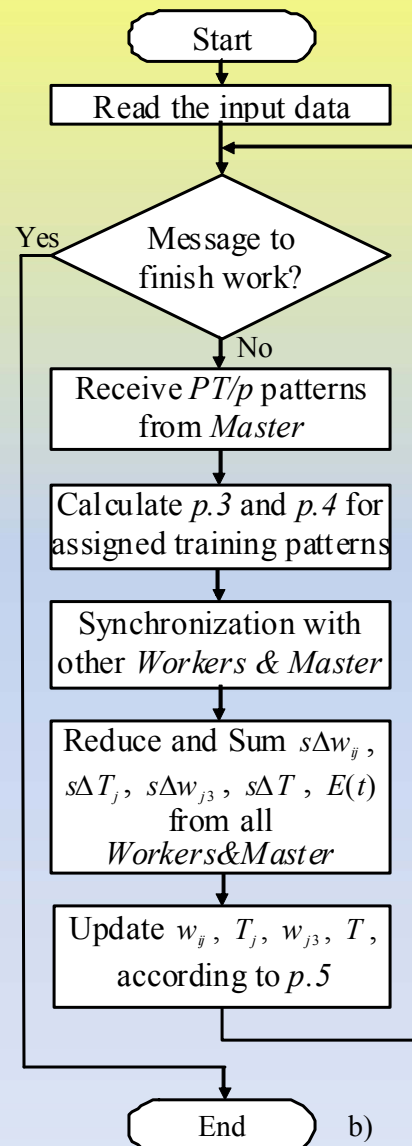
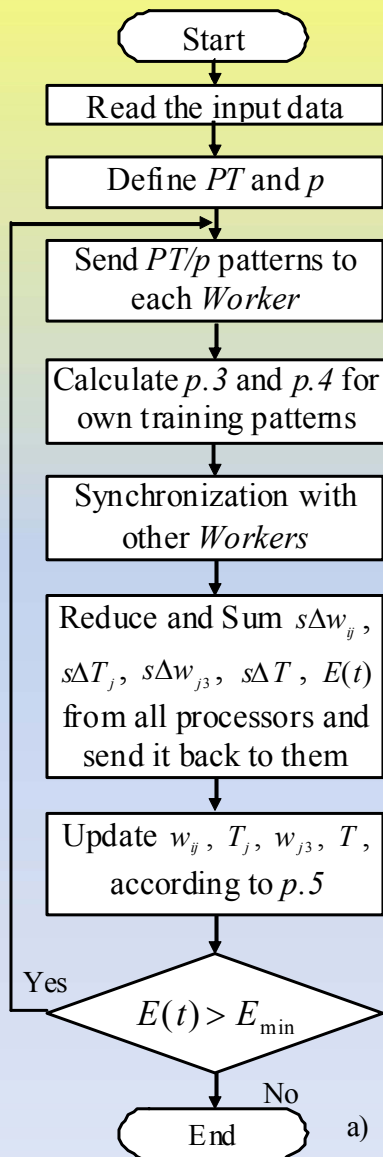
The batch pattern BP training algorithm consists of the following steps:

1. Set  $SSE = E_{\min}$  and training iterations  $t$ ;
2. Initialize the weights and thresholds;
3. Calculate for the training pattern  $pt$ :
  - 3.1. The output value  $y^{pt}(t)$ ;
  - 3.2. The error of the output neuron  $\gamma_3^{pt}(t) = y^{pt}(t) - d^{pt}(t)$ ;
  - 3.3. The error of hidden layer neuron  $\gamma_j^{pt}(t) = \gamma_3^{pt}(t) \cdot w_{j3}(t) \cdot y^{pt}(t) \cdot (1 - y^{pt}(t))$ ;
  - 3.4. The delta weights and delta thresholds of all perceptron's neurons and add the result to the value of the previous pattern
 
$$s\Delta w_{j3} = s\Delta w_{j3} + \gamma_3^{pt}(t) \cdot F'_3(S_j^{pt}(t)) \cdot h_j^{pt}(t), \quad s\Delta T = s\Delta T + \gamma_3^{pt}(t) \cdot F'_3(S_j^{pt}(t)),$$

$$s\Delta w_{ij} = s\Delta w_{ij} + \gamma_j^{pt}(t) \cdot F'_2(S_j^{pt}(t)) \cdot x_i^{pt}(t), \quad s\Delta T_j = s\Delta T_j + \gamma_j^{pt}(t) \cdot F'_2(S_j^{pt}(t)),$$

$$F'_3(S_j^{pt}(t)) = y^{pt}(t) \cdot (1 - y^{pt}(t)), \quad F'_2(S_j^{pt}(t)) = y_j^{pt}(t) \cdot (1 - y_j^{pt}(t));$$
  - 3.5. The SSE using  $E^{pt}(t) = \frac{1}{2} (y^{pt}(t) - d^{pt}(t))^2$ ;
4. Repeat step 3 for each  $pt$ ,  $pt \in \{1, \dots, PT\}$ ,  $PT$  is the size of the training set;
5. Update the weights and thresholds  $w_{ij}(PT) = w_{ij}(0) - \alpha(t) \cdot s\Delta w_{ij}$ ,  $T_j(P T) = T_j(0) + \alpha(t) \cdot s\Delta T_j$ ;
6. Calculate the total SSE  $E(t)$  on the training iteration  $t$  using  $E(t) = \sum_{pt=1}^{PT} E^{pt}(t)$ ;
7. If  $E(t)$  is greater than the desired error  $E_{\min}$  then increase the number of training iteration to  $t + 1$  and go to step 3, otherwise stop the training process.

# Parallel batch pattern training algorithm of MLP



- The sum operations of delta weights and thresholds are independent of each other
- The *Master* divides all patterns in equal parts corresponding to number of the *Slaves*
- Then the *Master* sends to the *Slaves* the numbers of the appropriate patterns to train
- Each Slave calculates the partials of delta weights, thresholds and SSE only for its assigned number of training patterns
- The global operations of reduction and summation should be executed just after the synchronization point. Then the summarized values of the delta weight and thresholds should be sent to all CPUs working in parallel. We implement this by `MPI_Allreduce()`
  - The `MPI_Allreduce()` function includes an internal synchronization point – we have removed the function `MPI_Barrier()` from our code
  - Reduce operation was implemented by only one call of `MPI_Allreduce()` with pre-encoding of all data into one message before sending and after-decoding the data to appropriate matrixes after receiving

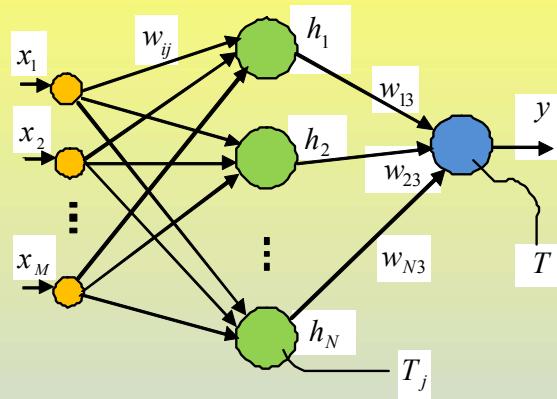
# Experimental environment

## The SMP parallel supercomputer

**Pelikan** (TYAN Transport VX50) - two identical blocks VX50\_1 and VX50\_2. Each block consists of four 64-bit dual-core processors AMD Opteron 8220 (2800 MHz), 16 GB of local RAM (667 MHz, registered DDR2), AMD-8131 Hyper Transport PCI-X tunnel interface. Due to some technical problem we have used only 8 cores located inside one block VX50\_1

## The computational cluster

**Battlecat**, consists of one head node and 7 working nodes. The head node consists of two Dual Xeon 1.6 GHz processors with 4 MB cache and 2 GB of local RAM. Each working node has one dual core CPU Intel Core 2 Duo 2.13 GHz with 2 MB cache and 2 GB of local RAM. The nodes are connected by 1 Gigabit link.



## Typical MLP architectures

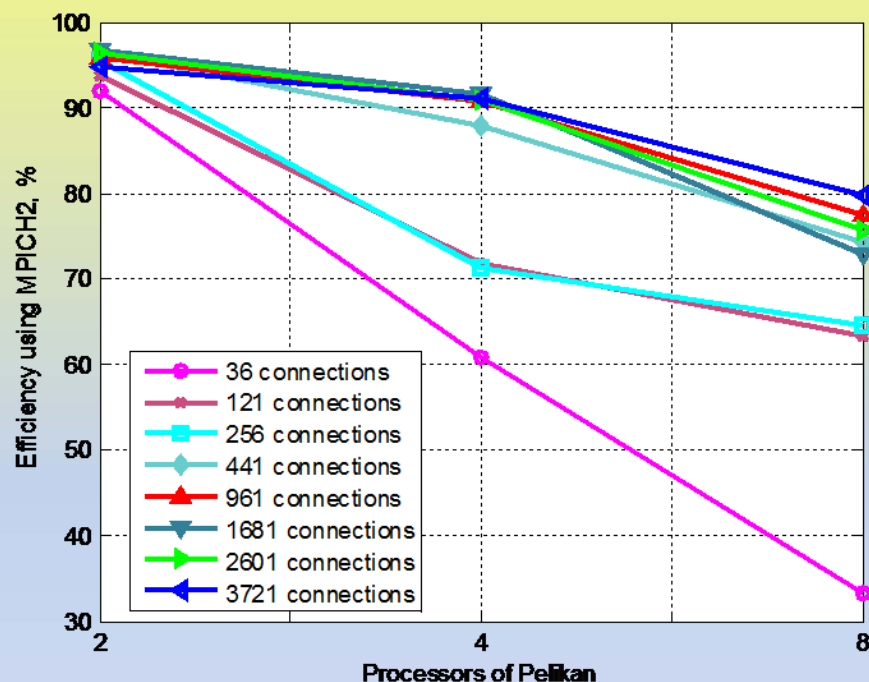
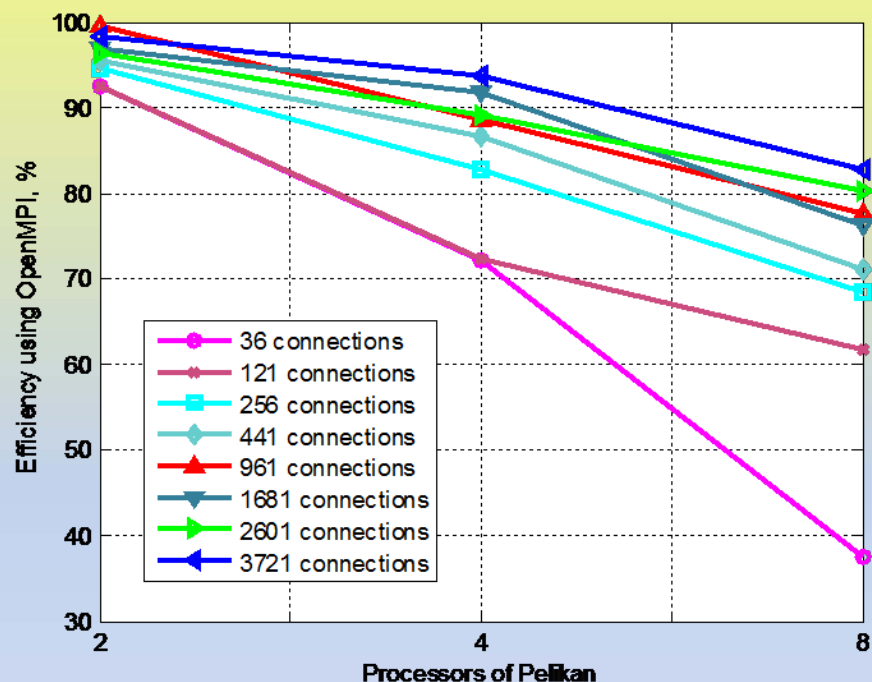
5-5-1 (36 connections)  
10-10-1 (121 connections)  
15-15-1 (256 connections)  
20-20-1 (441 connections) and so on

- New tuned collective's module of Open MPI
- Trigger value *coll\_tuned\_use\_dynamic\_rules* to 1
- file *mca-params.conf* located in home folder *~/.openmpi* of a user
- *\_info* with the arguments *--mca coll\_tuned\_use\_dynamic\_rules 1 --param coll all*.  
(1 basic linear, 2 nonoverlapping, 3 recursive doubling, 4 ring and 5 segmented ring)

```
host% mpirun -np 4 --mca  
coll_tuned_use_dymanic_rules 1 --mca  
coll_tuned_allreduce_algorithm 5  
myroutine.bin
```

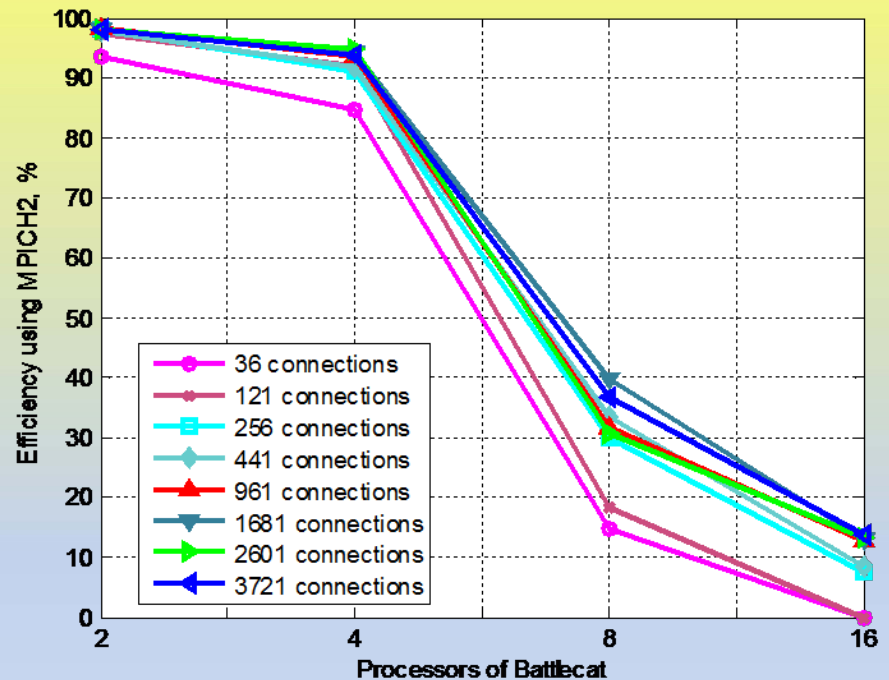
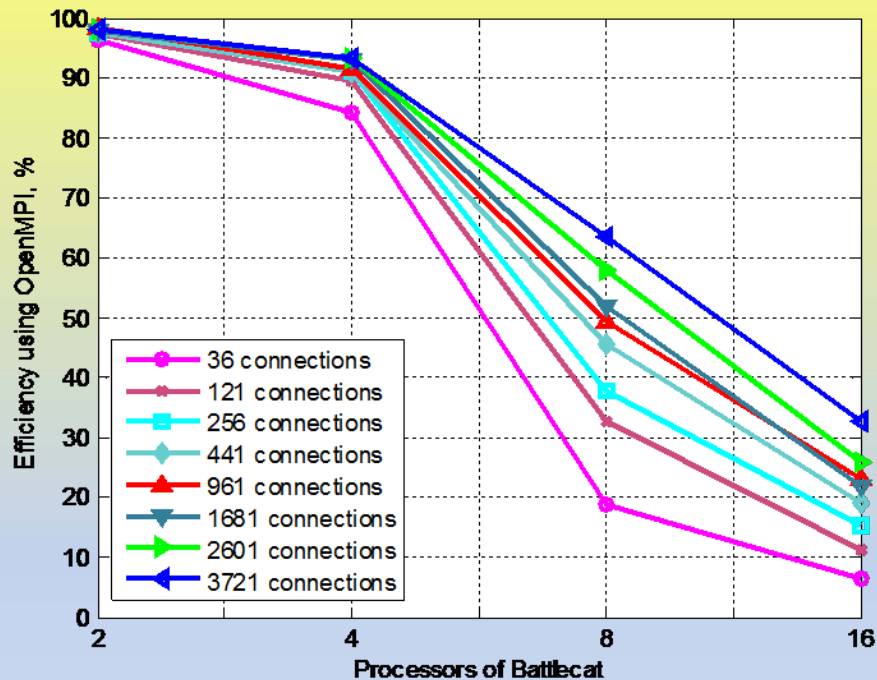


# Efficiency using Open MPI vs MPICH2 on SMP machine



- We have used Open MPI 1.4 and MPICH2-1.2.1 releases in our comparison
- Time measurement `MPI_Wtime()`, 200 training patterns,  $10^{-4}$  training epochs
- Total number of experiments is 24: The *default* algorithm provided better results for 11 cases of total 24, the *5-segmented ring* for 10 cases of 24 and *4-ring* for 3 cases of 24 on the SMP computer

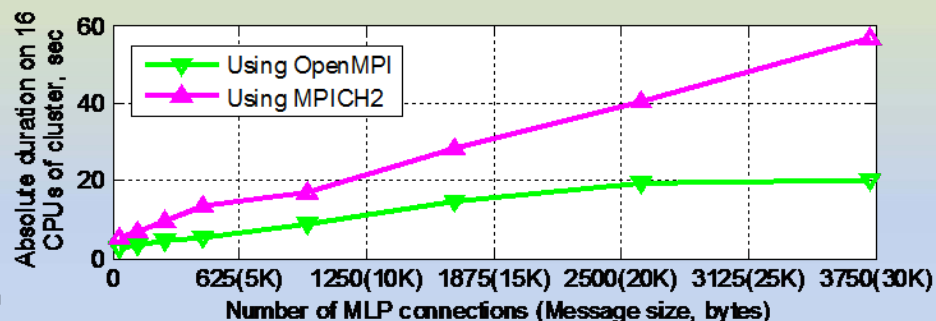
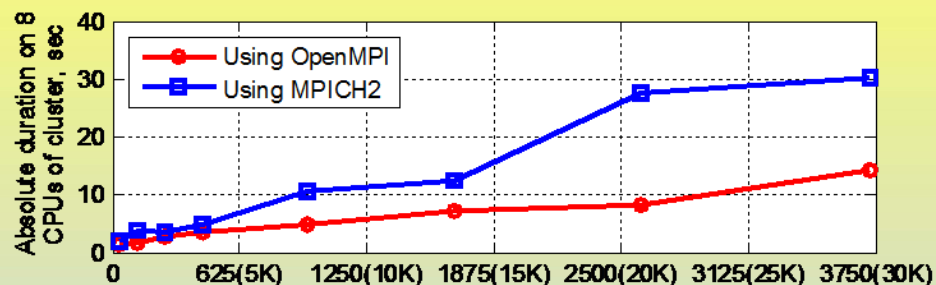
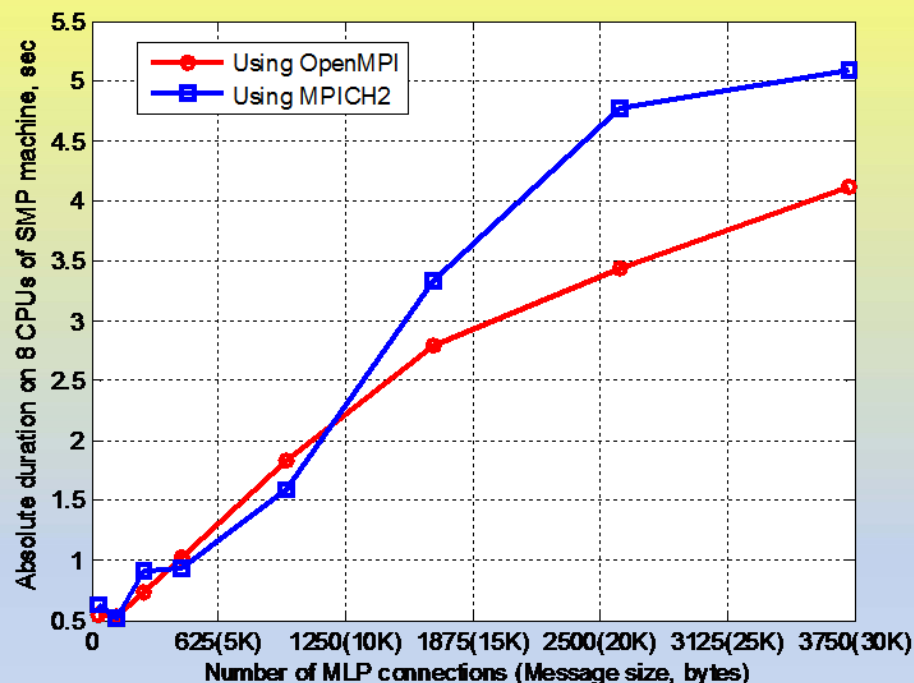
# Efficiency using Open MPI vs MPICH2 on cluster



- Total number of experiments is 32: The *5-segmented* ring algorithm provided better results for 12 cases of total 32, the *default* for 10 cases of 32 and *4-ring* for 10 cases of 32 on the computational cluster
- We have simply calculated the differences of parallelization efficiencies corresponding to the same scenarios in the following table

**Acknowledgments.** These results were obtained within my Marie Curie International Incoming Fellowship grant Ref. Num. 221524-908524 within the 7<sup>th</sup> European Community Framework Programme.

# Analysis of communication time



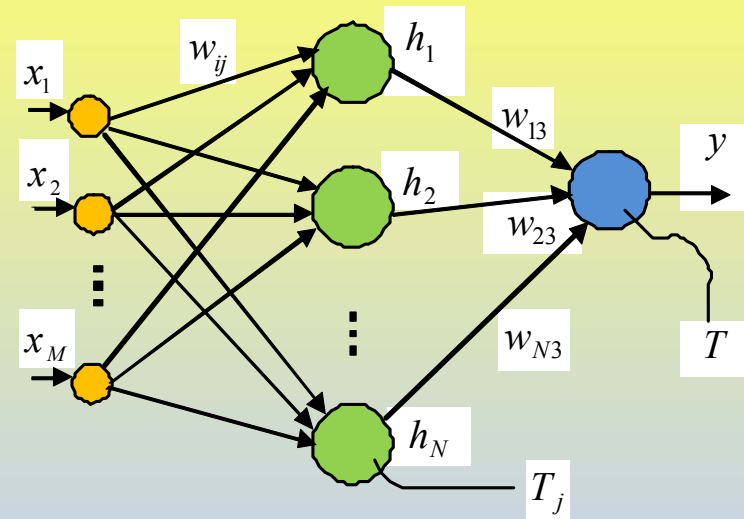
- The analysis of the absolute durations of communication overhead (Fig. 5) clearly shows that Open MPI tuned collective's module outperforms MPICH2 implementation both on SMP computer (minor degree) and computational cluster (strong degree)
- The time values in seconds on ordinate axes are for  $10^4$  messages because the training process of MLP is fulfilled by  $10^4$  training epochs and each epoch finishes by the communication of one message

# Experimental environment - 2

**The computational cluster *Reef*.** We have used up to 32 cores, located inside of 4 nodes. Each node contains two quad-core Intel Xeon E5345 CPUs, 2.33 GHz. Interconnections are Gigabit Ethernet, Fast Ethernet and Infiniband. We have switched network to use only Infiniband. We have used the Open MPI 1.2.8 library. We call this system “**Cluster Infiniband**”

**The computational cluster *Battlecat*** has one head node and 7 working nodes. The head node has two Dual Xeon 1.6 GHz CPUs, 4 MB cache, 2 GB local RAM. Each working node has one Intel Core 2 Duo 2.13 GHz, 2 MB cache and 2 GB local RAM. Inter-connection is Gigabit Ethernet. We have used the Open MPI 1.4.1 library and 16 cores. We call this system “**Cluster Gigabit Ethernet**” or “**CGE**”

**The parallel supercomputer *Crati*** (NEC TX7) has 28 one-core 64-bit CPUs Intel Itanium 2 (1 GHz) and 64 GB of total RAM. Each CPU has a second level cache of 3 Mb. Crati has ccNuma architecture of interconnection network. The NEC MPI/EX 1.5 is used. We call this system “**Computer ccNuma**” or “**CccNuma**”

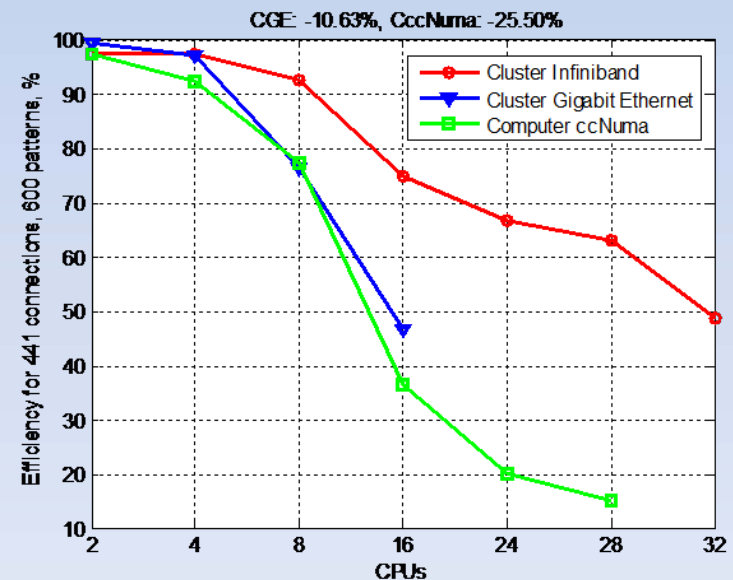
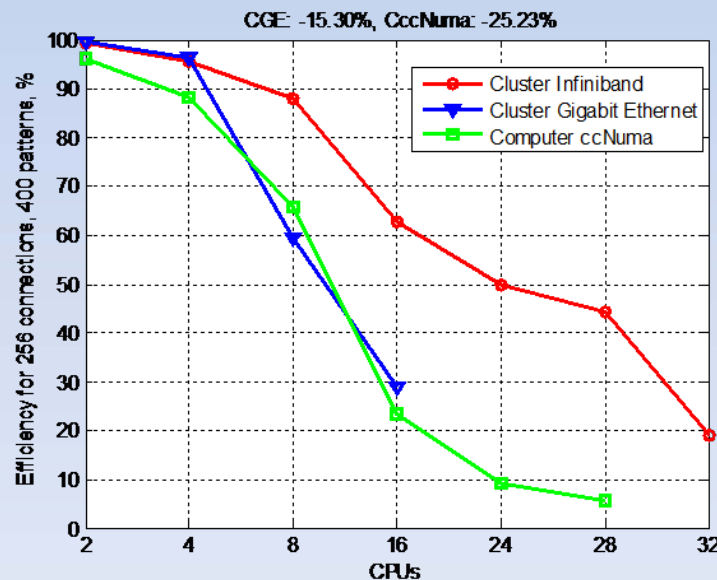
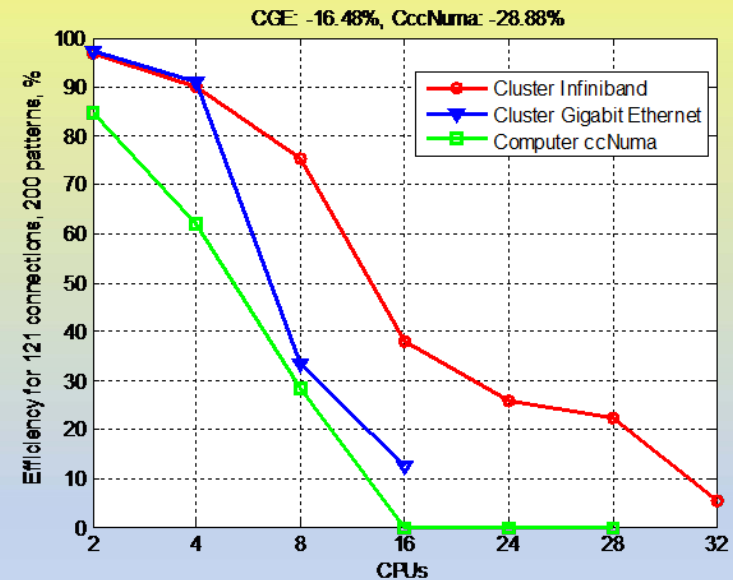
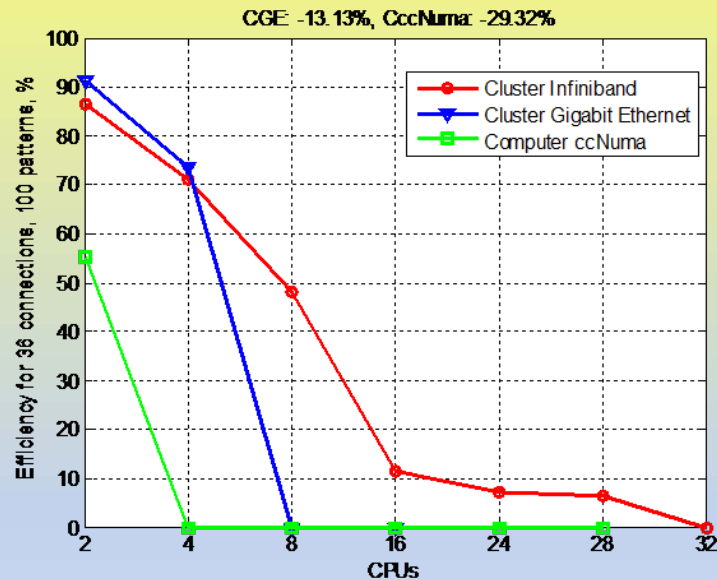


## Researched MLP architectures

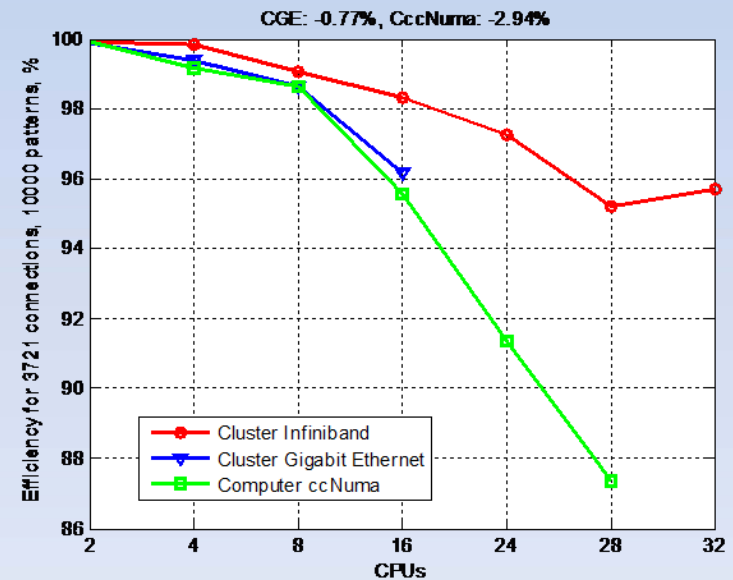
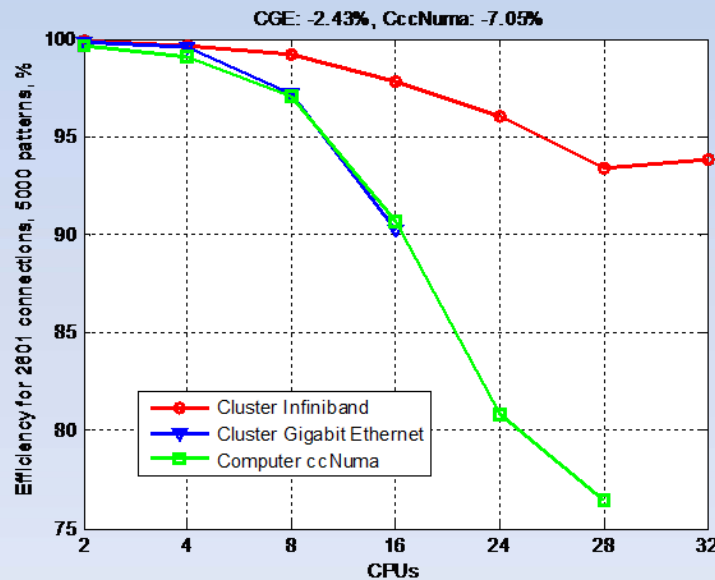
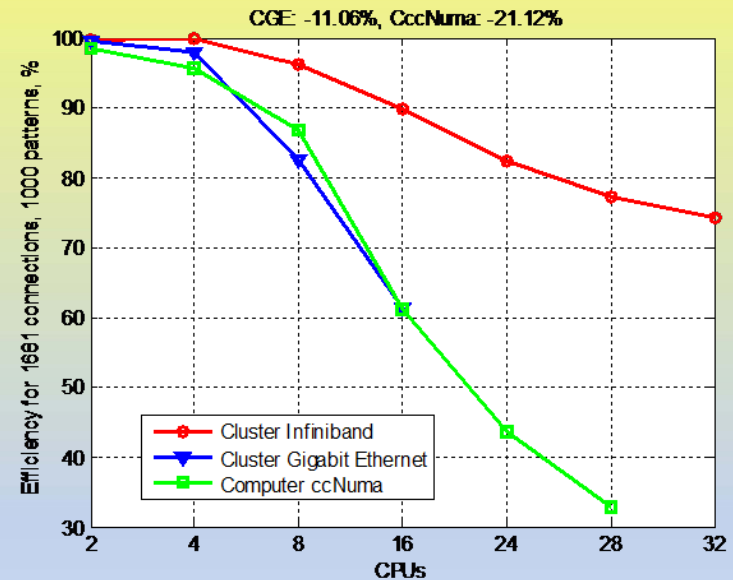
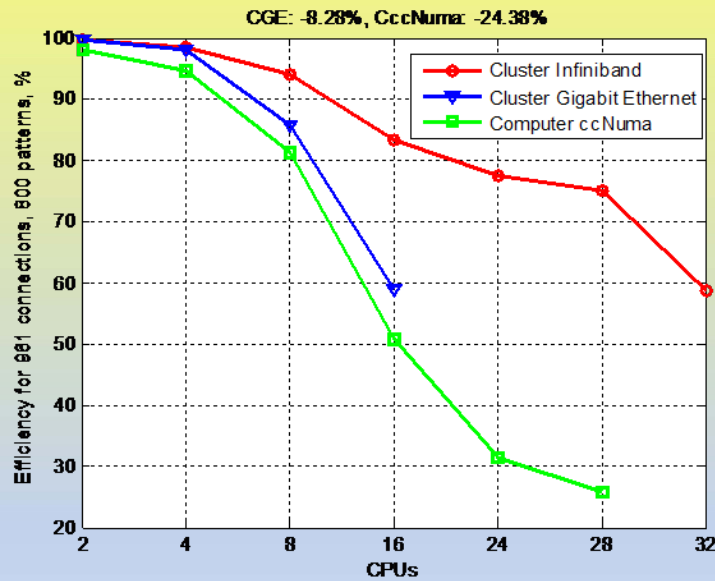
5-5-1 (36 connections), 10-10-1 (121 connections), 15-15-1 (256 connections), 20-20-1 (441 connections), 30-30-1 (961 connections), 40-40-1 (1681 connections), 50-50-1 (2601 connections) and 60-60-1 (3721 connections) and 70-70-1 (5041 connections). The number of training patterns is changed as 100, 200, 400, 600, 800, 1000, 5000 and 10000.

(5-5-1 = 25 weights between the input and the hidden layer + 5 weights between the hidden and the output layer + 5 thresholds of the hidden neurons and 1 threshold of the output neuron = 36 connections)

# Parallelization efficiencies on different parallel platforms



# Parallelization efficiencies on different parallel platforms





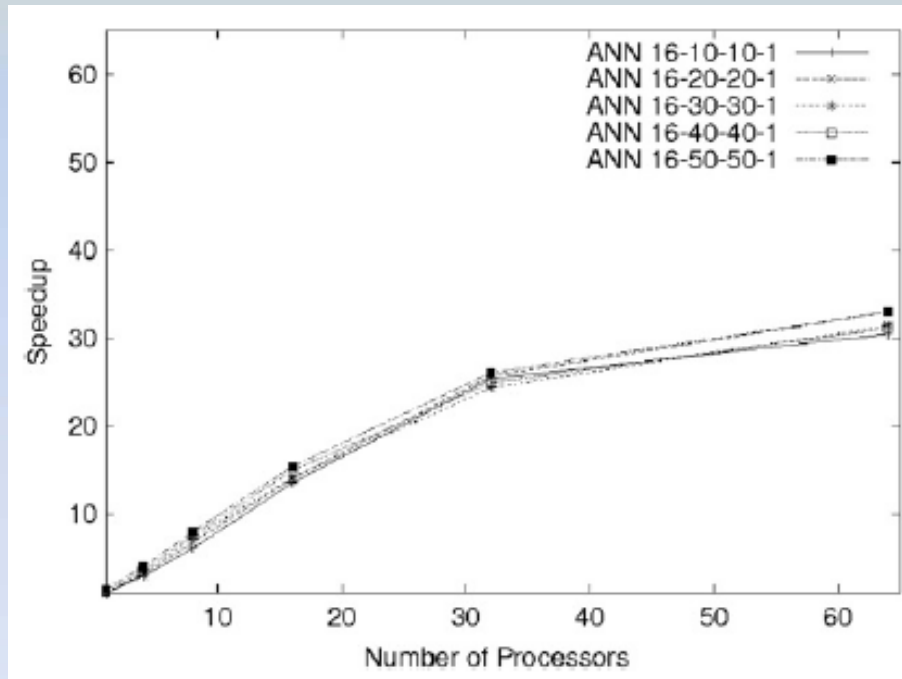
# Comparison with the existing solutions

R.M. de Llano, J.L. Bosque, Study of Neural Net Training Methods in Parallel and Distributed Architectures, Future Generation Computing Systems, Vol. 26, Issue 2, 2010, 267-275.

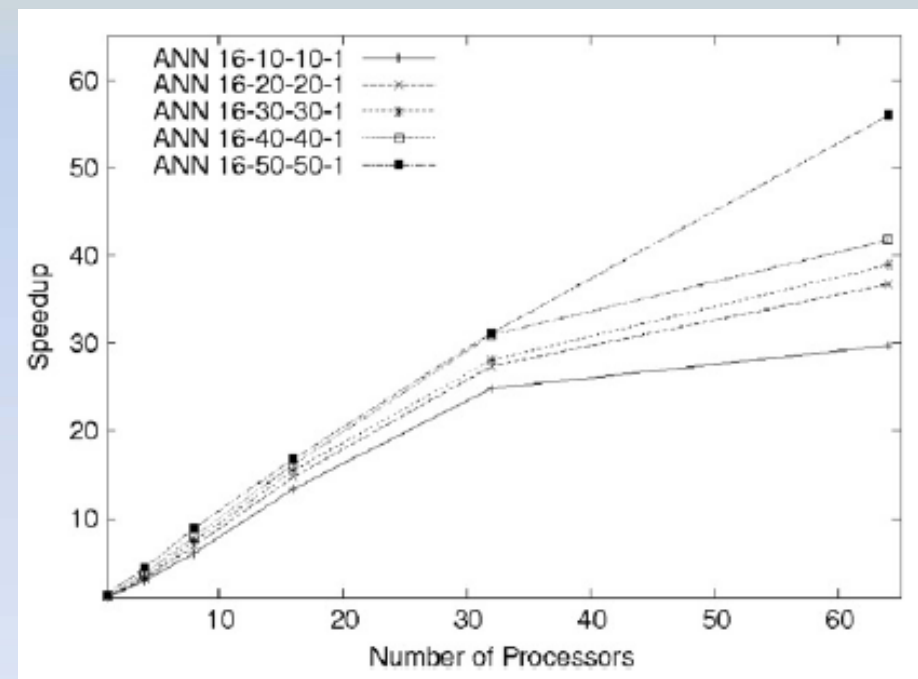
- 20000 training patterns
- MPI-G2, PBLAS, ScaLAPACK
- 3 messages, MPI\_Scatter, MPI\_Send/Receive  
MPI\_Gather
- 270 connections – speedup is less than 1

Configurations of ANN and number of weights associated with each configuration.

ANN configuration	Number of weights
16-10-10-1	270
16-20-20-1	740
16-30-30-1	1410
16-40-40-1	2280
16-50-50-1	3350



SMP



Cluster

# My Project

## EFFICIENT PARALLEL BATCH AND SINGLE PATTERN NEURAL NETWORK TRAINING ALGORITHMS USING OPEN MPI AND GPU-COMPUTING

### Research Plan:

**Objective 1:** test enhanced batch pattern parallel algorithm for NN training by changing the parameters of the internal algorithms of MPI collective functions on different parallel architectures

**Objective 2:** develop GPU-based versions of the parallel batch and single pattern algorithms for neural network training

**Objective 3:** test experimentally the efficiency improvement of the GPU-based version of the algorithms in comparison with their Open MPI implementations

**Duration of my stay at the UT: 9 months, Sep 1, 2012 – May 31, 2013**

**Thank you for your attention!**

**Questions & Answers?**