NEW LEARNING APPROACH BY GENETIC ALGORITHM IN A CONVOLUTIONAL NEURAL NETWORK FOR PATTERN RECOGNITION

Majid Mohammadi
Department of Computer Science
Shahid Bahonar University
Kerman, Iran
mohammadi@uk.ac.ir

Mohammad Ali Mehrolhassani
Department of Computer Science
Shahid Bahonar University
Kerman, Iran
alimehrolhassani@yahoo.com

ABSTRACT

Almost all of the presented articles in the CNN[1] are based on the error backpropagation algorithm and calculation of derivations of error, our innovative proposal refers to engaging TICA[2] filters and NSGA-II[3] genetic algorithms to train the LeNet-5 CNN network. Consequently, genetic algorithm updates the weights of LeNet-5 CNN network similar to chromosome update. In our approach the weights of LeNet-5 are obtained in two stages. The first is pre-training and the second is fine-tuning. As a result, our approach impacts in learning task.

1 Background

The CNN network presented by Fukushima (Fukushima, 1975; Fukushima, 1980; Fukushima, 1986; Fukushima, 1989; Imagawa, 1993) in 1975 to solve handwritten digit recognition problems, called it Neocognitron. Originally the CNN inspired by the works of Hubel and Wiesel (Wiese, 1962) on the neurons of the cat's visual cortex. The LeCun in (Y. LeCun, 1990) has implemented the first CNN, trained by online Backpropagation, with an extreme accurate recognition process (high accuracy percentage).

Briefly, in the paper, using the TICA filters and NSGA-II algorithms caused to be capable of training a type of CNNs called LeNet-5 by NSGA-II algorithm in two stages: pre-training and fine-tuning on the tiny pack of handwritten digits[4] (a distinct pack encompasses 50 samples from MINST dataset).

[1] Convolutional Neural Network
[2] Topographic independent component analysis
[3] Non dominated Sorting Genetic Algorithm II
[4] The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. It contents a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

1.1 The LeNet-5 Model

Figure 1 illustrates the principal architecture of LeNet-5 model which had been trained with LeCun`s Online BP[1] Algorithm.

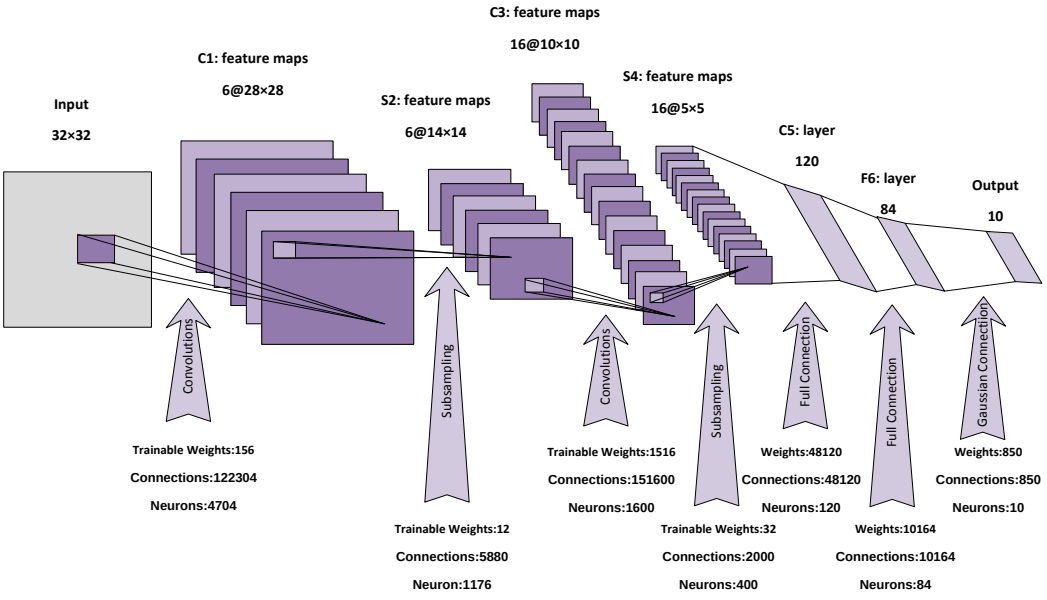


Figure 1: The architecture of LeNet-5

The CNN extracts the features in hierarchy and creates a spectrum of input pattern upon the output layer; the obtain spectrums help to categorize the input facts. The learning method of the CNN concentrates upon extracting features of the pattern automatically.

The LeNet-5 consists of seven layers exclude the input layer (Duffner, 2007). The size of input dimension set to 32×32 pixels. The first five layers, C1, S2, C3, S4, and C5 determine convolution and sub-sampling layers.

The small window that called receptive filed refers to a dimension of 5×5 window for convolution layers and a sub-sampling ratio factor of 2. In figure 2 shows two preliminary layers associated with the input domain called retina. In general, at the end of each convolution layer the subsample layer is attached.

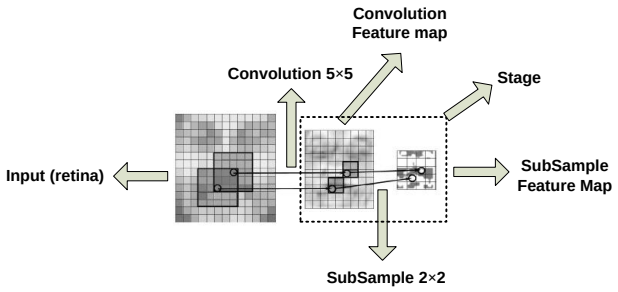


Figure 2: The convolution map and subsample of LeNet-5 (Duffner, 2007).

[1] Online back propagation

1.2 TICA Filters

Topography independent component analysis is a 2D map component that has been organized with adjacent component to extract similar features (figure 4). The component extracts visual features from tiny patches of natural images. In fact, the model has minimized the correlations of energies with a pertinent object function applied. The model can be a CNN (K. Kavukcuoglu, 2009) or can be an optimization algorithm (Koray, 2008). Figure 4 shows the TICA filters with the size of 16×16 that after 5000 iterations obtains them.

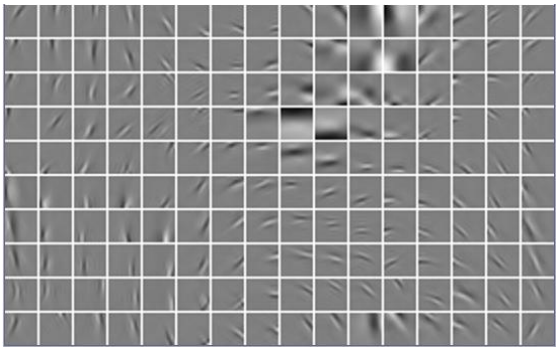


Figure 4: The TICA Filters with dimensions 16×16

1.3 NSGA-II Algorithm

One way for obtaining the answers of equivalent to minimize or maximize exploits the evolutionary algorithms; thus, the evolutionary algorithms to be applied instead of derivative calculations. In the implemented algorithms, the answers instead of gene in chromosome and the chromosome called individual and finally the individual formed a population.

The population based on natural selection competes with others for searching. The algorithm performs on population some operation etc. crossover, mutation, selection; the population size, mutation rate, crossover rate and selection strategy conducts the outcome.

The times of running the algorithm called iteration. In the iteration evaluated the population by a function so-called fitness function. Several policies exist to stopping the algorithm: finding the appropriate answer or answers, specified iteration, converge the evaluating of individuals.

In fact, the population to be generated randomly in initializing section and evolved in iterations by randomly operators. Some individuals to be chose for mutation of genes and some of them to be chose to mixture with others. At the end of iteration selects to new population in next iteration with a specific policy from some of individuals. Researcher has implemented numerous evolutionary algorithms as the NSGA-II algorithm.

Srinivas and Deb presented NSGA method in 1993 (Deb, 1995). The NSGA algorithm implements for MOP target purposes with a high performance. The regenerated version, NSGA-II presented in 2002 (Amrit Pratap, 2002) which use the number of domination for each individual by other individual. Constantly, for pertinent performance, the mentioned techniques help to acquire Pareto-Optimal[1] points with reduced complex computing by exerting non-dominated

[1] When in multi-objective optimization follows the set of optimum solutions which have no superiority with themselves, the set of answers called Pareto1-Optimal which is the name of economist nominated Vilferedo Pareto.

individuals and calculating of crowding point's distance individuals. Figure 5 illustrates the NSGA-II algorithm.

Totally, the sorting functional NSGA-II algorithm has been improved by storing the frequency of each dominated individual with the others.

The NSGA-II has three crucial issues: quick sorting for non-dominated individuals; computation of crowded distance in entire Pareto-Front; method of individual selection.

For quick sorting, the entire individuals must be compared with themselves and pulled the non-dominated out of them with assigning the rank 1. The task is repeated until the entire population allocates a rank from 1 to N to sort the individuals. Figure 6 illustrates the Pareto-Front issue in three parts. The part (a) shows the situation C member with the two lines` perpendicular paralleled with axes on C, as the adjacent of the specified individual C has posed over the district of top left, bottom left, top right and bottom right. The bottom left members dominated by C, If would be assumed the objective functions f1 and f2 maximized, the top right area dominated C member. The district of bottom right and top right have been ignored to determine for dominating. The matlab code declares in mathworks web site[1].

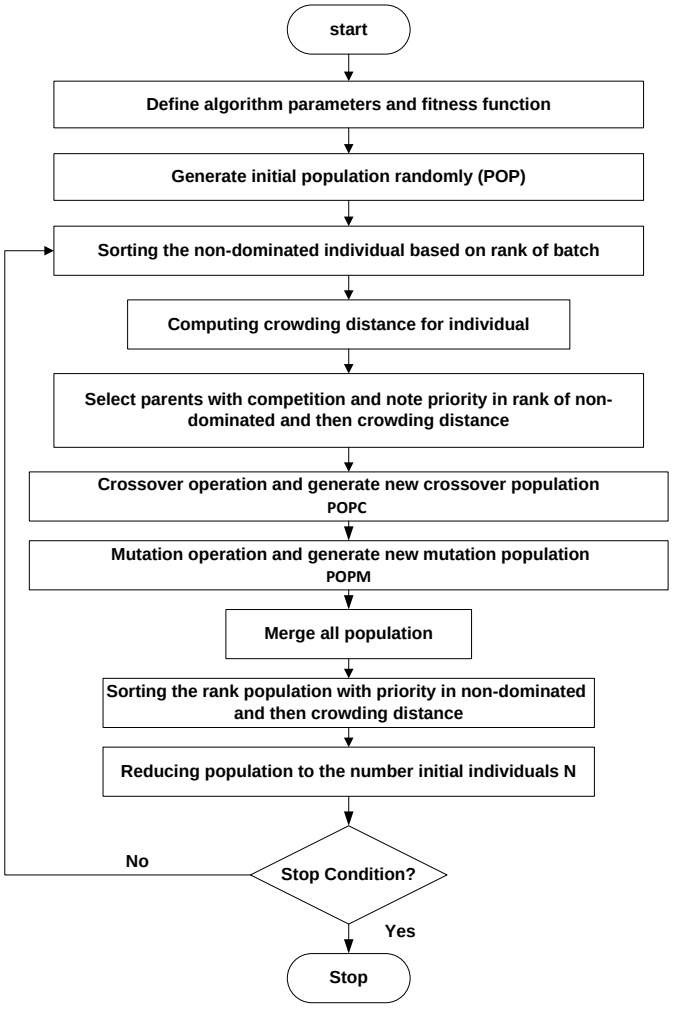


Figure 5: Flow chart of NSGA-II algorithm

[1] https://www.mathworks.com programmed by S. Mostapha Kalami Heris

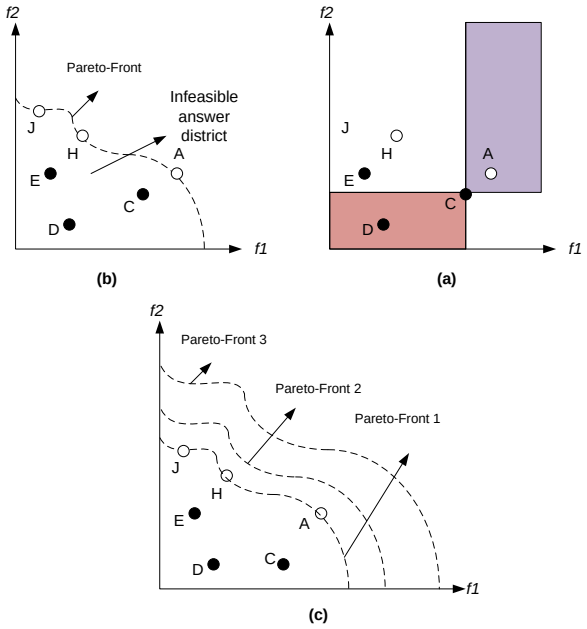


Figure 6: Pareto-Front in NSGA-II algorithm: the (a) shows the districts of C member; the (b) shows the set optimal answers Pareto-Front in on border; the (c) shows the Pareto-Front in next iterations for maximization problems.

The Pareto-Front refers to compute additional measure inside of population ranking called crowded distance. Therefore, the distance of chromosomes is computed and stored with respective adjacent in Pareto-Front by using Euclidean distance for entire members. Totally, the population rank and the crowded distance prepare the fitness value for entire members whilst the member with the low rank could be assigned to the low fitness value.

The method of individual selection determines the computation of ranking of non-dominated and crowded distance of individuals. Thus, two values acquired for ranking and computing crowded distance for comparing two members of population. The algorithm selects the member with lower priority. Competition between members of the population with identical rank caused to win the one which has the higher crowded distance. The later generation is induced by merging the winner parents and offspring in competition and after that the population`s size sets the primary length.

2 The proposal

The proposal focused on genetic or heuristic algorithms to determine the weights of LeNet-5; without using the Backpropagation and the derivation. For better following our proposal, the steps listed below:

1) The F6 layer has been eliminated in LeNet-5 model; consequently, the length of chromosome in GA reduces to 51046 variables.
2) The train and test validation function have merged and 50 samples in 10 classes (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9) selected from MINST dataset randomly. The average of errors for the entire 50 samples leads the LeNet-5 to learning.
3) All the biases in LeNet-5 set to zero. The number of individual population or particles vector size sets to 100.
4) We have two stages with specified respective parameters (table 1) and techniques in the NSGA-II algorithm usage (figure 7). The first stage calls Pre-training and the second

calls Fine-tuning; the mentioned parameters have been extracted from our experiments and a tuned LeNet-5 with online Backpropagation.

5) The RMSE[1] and MCR[2] employed to minimize the MOP[3] in the NSGA-II or the other experimental algorithms which compute them with Eq 1 and Eq 2.
6) In the NSGA-II algorithm`s fitness function applied ax^2 and by^2 (x denote RMSE and y denotes MCR)
7) The output of LeNet-5`s model for digit 1 and desired digit 1 as depicted the table of digits labels for using in supervised learning (figure 8).

Table 1: The parameters in the NSGA-II at Pre-training and Fine-tuning stages

Variable	**Pre-training stage**	**Fine-tuning Stage**
Iteration(s)	1000	1000
Number of population	100	27
Chromosome Lengths	51046	51046
Variable Minimum	-0.9000000000000000	-4.0000000000000004
Variable Maximum	0.9000000000000000	6.0000000000000001
Population Crossover	0.2	0.1
Population Mutation	0.3	0.4
Ratio Mutation Genes	0.5	0.00025

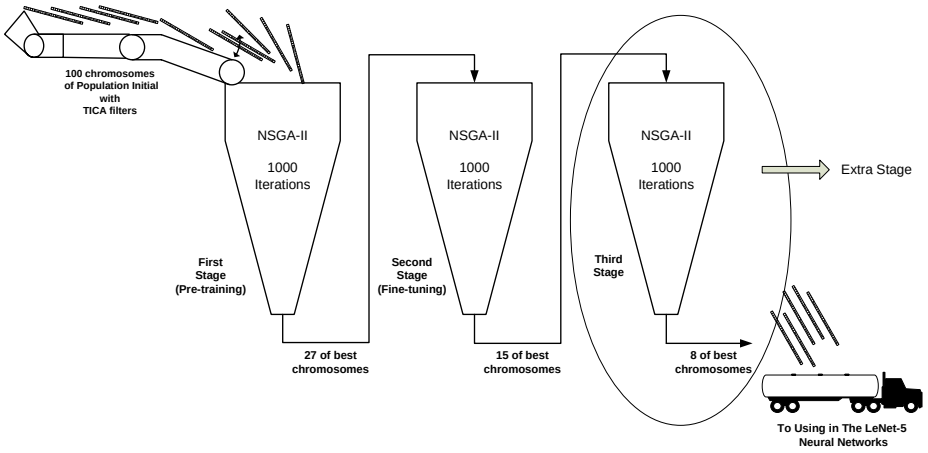


Figure 7: Stages of the proposal.

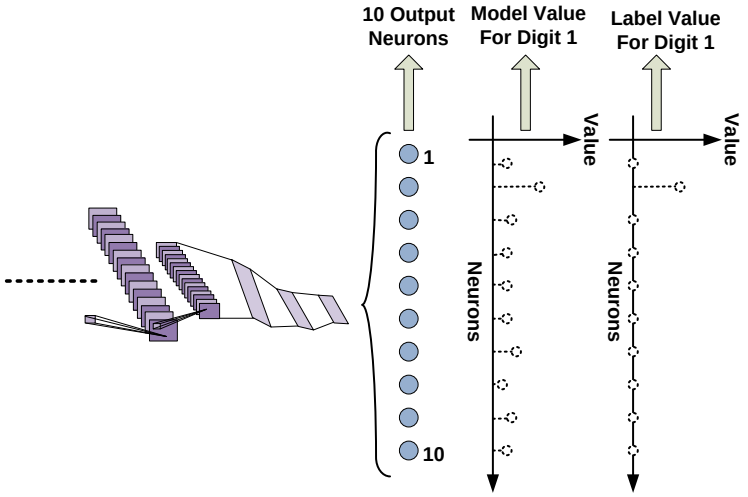
$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(X_{obs,i} - X_{model,i}\right)^2}{n}} \qquad (1)$$

Where $X_{obs,i}$ denotes value of observation in model, $X_{model,i}$ denotes value of output desire in i location.

$$MCR = \left(1 - \frac{M}{N}\right) \times 100 \qquad (2)$$

Where M denotes sum of number of templates become correct, N denotes number of all the templates.

[1] Root means square error
[2] Misclassification rate
[3] Multi-objective optimization



Outputs Of Neurons	Digits									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0
8	0	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	1	0
10	0	0	0	0	0	0	0	0	0	1

Figure 8: Up: Digit 1 in output of LeNet-5 model and digit 1 in label. Down: Table of digit labels.

2.1 Pre-training stage

The parts of chromosome that assigns to the layer C1 and C3 in the LeNet-5 should not to be modified at the first stage; at the beginning of the NSGA-II algorithm, C1 and C3 initialized by 160 TICA filters with dimension of 16×16 in section 1.2 randomly. The filters should be resized to dimension of 5×5 to fit the LeNet-5`s weights matrix in respective layers (figure 9). Figure 10 illustrates the pre-training.

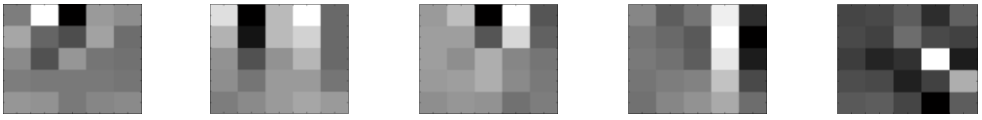


Figure 9: The five TICA filters form left to right of the TICA filters in section 1.2 that resize into Dimension of 5×5

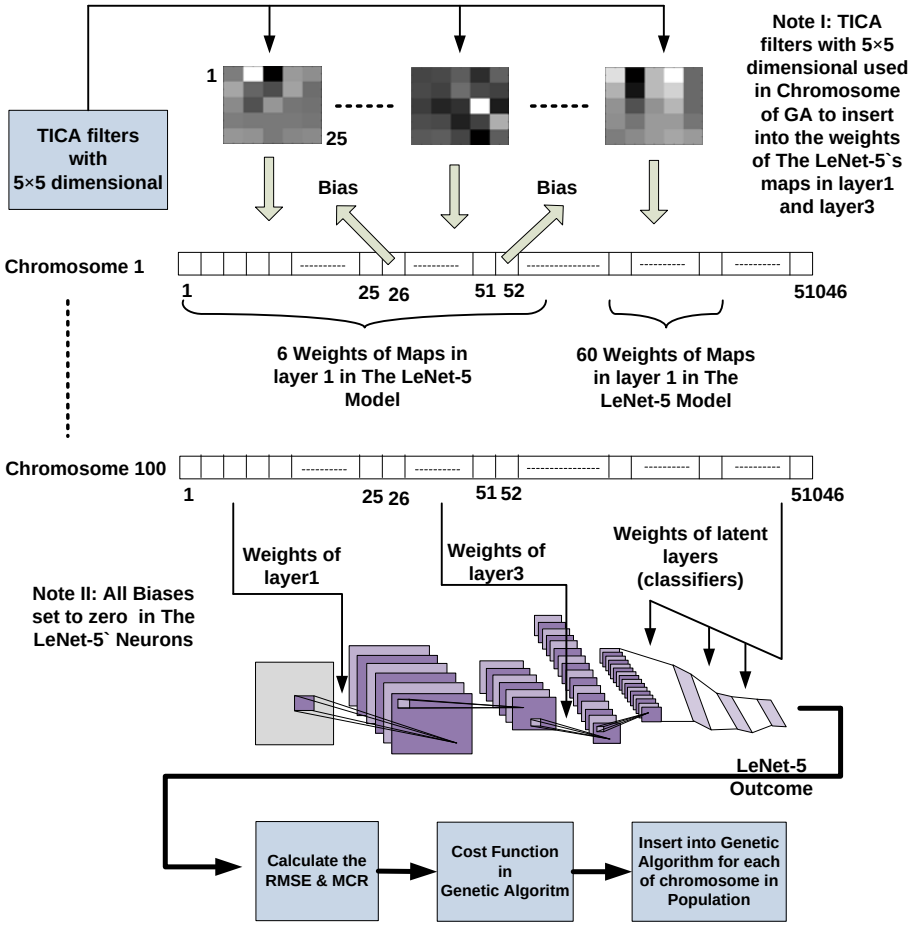


Figure 10: Showed relevance between chromosome of the NSGA- II algorithm and weights of layers in LeNet-5 Neural network at Pre-training stage of initializing.

The pre-training algorithm declares in algorithm1 with pseudocode in matlab.

Algorithm 1 Pre-training

1: Define the parameters of algorithm matched with table 1 and the fitness function equal to x2 and y2 in Cost function
2: Generate initial population`s chromosomes randomly (POP) from the 160 TICA filter dimension of 5×5
3: [TestData TargetData]=SelectSamples(); % Select 50 samples from MINST dataset
3: for i:=1:npop
4: chromosomtoCNN(pop(i)); % Replace the vector of the chromosome into the CNN neural network
5: [RMSE MCR]=PerformanceCNN(TestData,TargetData);
6: pop(i).Cost=Cost(RMSE MCR); % Calculate the cost function for chromosome
7: end % Initializing population
8: for iteration=1:nIteration
9: Create popC for Crossover population
10: for iCrossover=1: nCrossover
11: i1=BinaryTournamentSelection(pop); % Select an individual from population
12: i2=BinaryTournamentSelection(pop); % Select an individual from population
13:[popC(iCrossover,1).Position

popC(iCrossover,2).Position]=Crossover(pop(i1).Position,pop(i2).Position,VarRange);
14: popC(iCrossover,1).Position(1:1684)=pop(i1).PrimaryPosition(1:1684); % Replace the Choromosome TICA filter
15: chromosomtoCNN(popC(iCrossover));
16: [RMSE MCR]=PerformanceCNN(TestData,TargetData);
17: popC(iCrossover,1).Cost=Cost(RMSE MCR); % Calculate the cost function for chromosome
18: popC(iCrossover,2).Position(1:1684)=pop(i2).PrimaryPosition(1:1684); % Replace the Choromosome TICA filter
19: chromosomtoCNN(pop(iCrossover));
20: [RMSE MCR]=PerformanceCNN(TestData,TargetData);
21: popC(iCrossover,2).Cost=Cost(RMSE MCR); % Calculate the cost function for chromosome
22:end % Crossover
23: Create popM for Mutation population
24: for iMutation=1: nMutation
25: i=BinaryTournamentSelection(pop);
26: popM(iMutation).Position=Mutate(pop(i).Position,mu,VarRange);
27: popM(iMutation).Position(1:1684)=pop(i).PrimaryPosition(1:1684); % Replace the Choromosome TICA filter
28: chromosomtoCNN(pop(iMutation));
29: [RMSE MCR]=PerformanceCNN(TestData,TargetData);
30: popM(iMutation).Cost=Cost(RMSE MCR); % Calculate the cost function for chromosome
31: end % Mutation
33: popC=popC(:); % Turn to on dimension
34: pop=[pop popC popM] % Merge all population
35: [pop F]=NonDominatedSorting(pop); % Sorting the non-dominated individual based on rank of batch for population
36: pop=CalcCrowdingDistance(pop,F); % Computing crowding distance for population
37: pop=SortPopulation(pop); % Sorting the rank population with priority in non-dominated and the crowding distance
38: pop=pop(1:nPop); % Reducing population to the number initial individuals N
39:end % Iteration

2.2 Fine-training stage

The 27 pertinent individuals have been selected from the preceding stage for initializing preliminary population. The entire layers` weights have been authorized to be modify in the LeNet-5; caused the TICA filters to be improved to extract pertinent features in the layer C1 and C3 of the LeNet-5 scheme by the NSGA-II algorithm. In figure 11 and 12 present graphs for RMSE and MCR errors.

The code of the fine-tuning algorithm declares in Algorithm 1 with remove the lines of replaced the TICA filter in the chromosome (lines 14, 18, 27) and with initialize the parameters at beginning from table 1.

2.3 The results

Our practices (methods in table 3 and 4 and respective figures) in the article are listed at table 2. (Note: each of the practices has been rendered five times and the average of best results (minimum in RMSE and MCR) has been chosen and rounded in the specified iterations at table 3 and 4 and respective figures). It should be noted that the RMSE and MCR not have related with together; in whole results some time they are opposed (Figure 13).

The '*' sign marks our proposal`s results in table 2. The two stages` results (pre-training and fine-tuning) have been typed in column 1 and 2 of table 3 and 4 for RMSE and MCR; learning achieved with tracing the results.

Table 2: Experimental List

No	Experiments	DataSet	Number of sample
1	*NSGA-II initialized with TICA Filters in two stages (second stage in our proposal)	MNIST	50
2	*NSGA-II initialized with TICA Filters in two stages (first stage in our proposal)	MNIST	50
3	NSGA-II initialized with TICA Filters	MNIST	50
4	Standard GA	MNIST	50
5	Standard PSO	MNIST	50
6	Standard PSO initialized with TICA Filters	MNIST	50
7	NSGA-II	MNIST	50

Table 3: The selected RMSE (in percentage) values in experimental for 50 samples

Iteration(s)	Experiments (Refer to table 2)						
	1	2	3	4	5	6	7
100	1.85	4.45	5.8	6.8	3.8	6.8	7.8
300	1.4	3.4	5.1	5.9	3.8	5.8	6.5
500	1.37	2.87	4.9	3.72	3.8	3.77	5.8
700	1.46	2.46	4.8	3.72	3.8	3.77	5.8
950	1.39	2.19	3.59	3.72	3.8	3.77	4.4
1000	1.34	1.94	3.51	3.72	3.8	3.77	4.22

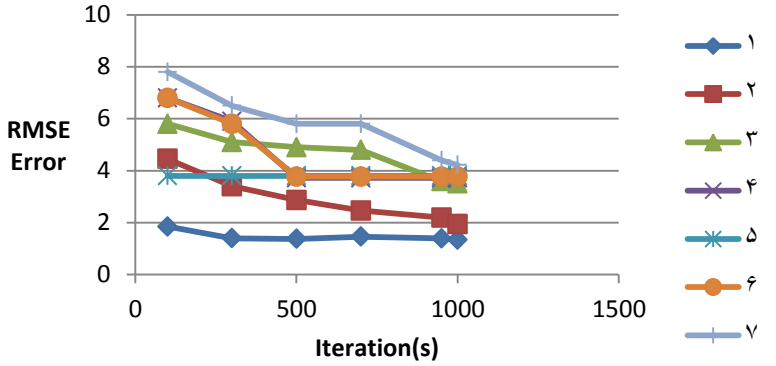


Figure 11: Compare the selected RMSE (in percentage) values from table 3 in experiments of iterations

In figure 12, circumstances of the results for the practices 3 up to 7 imply plenty of errors in the MCR error. Figure 13 depicted the outcomes in the proposal in second stage from original values.

Note: Cost function or fitness function equivalent in MOP in the entire experimental algorithms has applied the ax^2+by^2 equivalent (x denotes RMSE and y denotes MCR; a=b equal to one) a=b equal to one). In SOP[1] separately applied power equivalent (x^2).

Finally, in the study, the table 5 shows the comparative results with the trained LeNet-5 and a derivation trained model has been practiced on the 50 samples comparing of the ability of generalization. Hence, for test phase, our assessment was carried out on 10,000 samples which

[1] Single-objective problem

were selected from the MINST dataset randomly. Our model tested with 35 percent errors and the other tested with 55 percent in MCR measure.

Table 4: The selected MCR (in percentage) values in experiments for 50 samples

Iteration(s)	**Experiments (Refer to table 2)**						
	1	**2**	**3**	**4**	**5**	**6**	**7**
100	12	45	74	88	60	90	90
300	18	35	72	82	60	82	86
500	6	28	64	62	60	68	82
700	4	24	62	62	60	68	80
950	4	24	62	62	60	68	78
1000	4	22	60	62	60	68	70

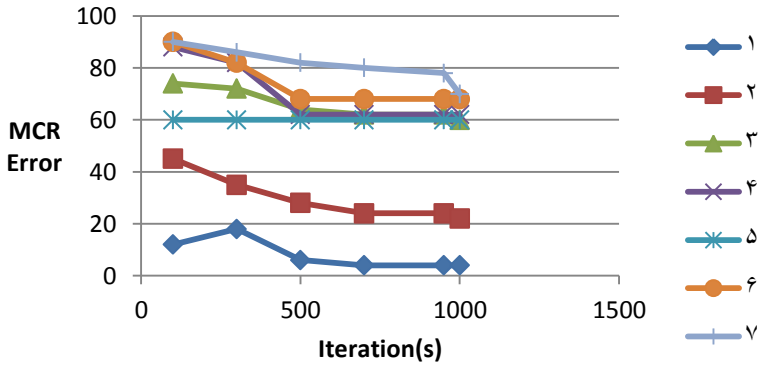


Figure 12: Compare the selected MCR (in percentage) values from table 4 in experimental in iterations

Table 5: The selected MCR (in percentage) values in experiments for 10,000 samples

No	**Experiments**	**MCR Error**	**DataSet**	**Number of sample used for training**	**Number of sample used for final test**
1	The LeNet-5 in our approach	35	MNIST	50	10,000
2	LeNet-5 with backpropagation algorithm	55	MNIST	50	10,000

Finally, the novelties in the proposal were listed below:

1) The TICA filters dimension of 16×16 resized to dimension of 5×5 without being required of learning them for LeNet-5 scheme. It means researcher can apply TICA filters in their convolutional neural network with resizing them in desired condition directly; as dimension of 16×16 to 5×5.
2) Supervised learning has been employed at the beginning of our proposal with TICA filters. TICA filters must be learnt by neural network by unsupervised training from natural patch images and afterward the original dataset must be taught to CNN model by supervised training.
3) The weights of layer C1 and C3 were not modified to the end of the first stage simultaneously and afterwards allowed them to be modified in the second stage.

4) Appling the GA algorithm, as NSGA-II trains the paramount neurons in neural networks as the LeNet-5. Subsequently, the plethora of computation is not required in derivation and Backpropagation algorithm; therefore, the approach is advantageous to parallel processing.
5) The generalization test is well as the obtain models in the article trained with a few samples.

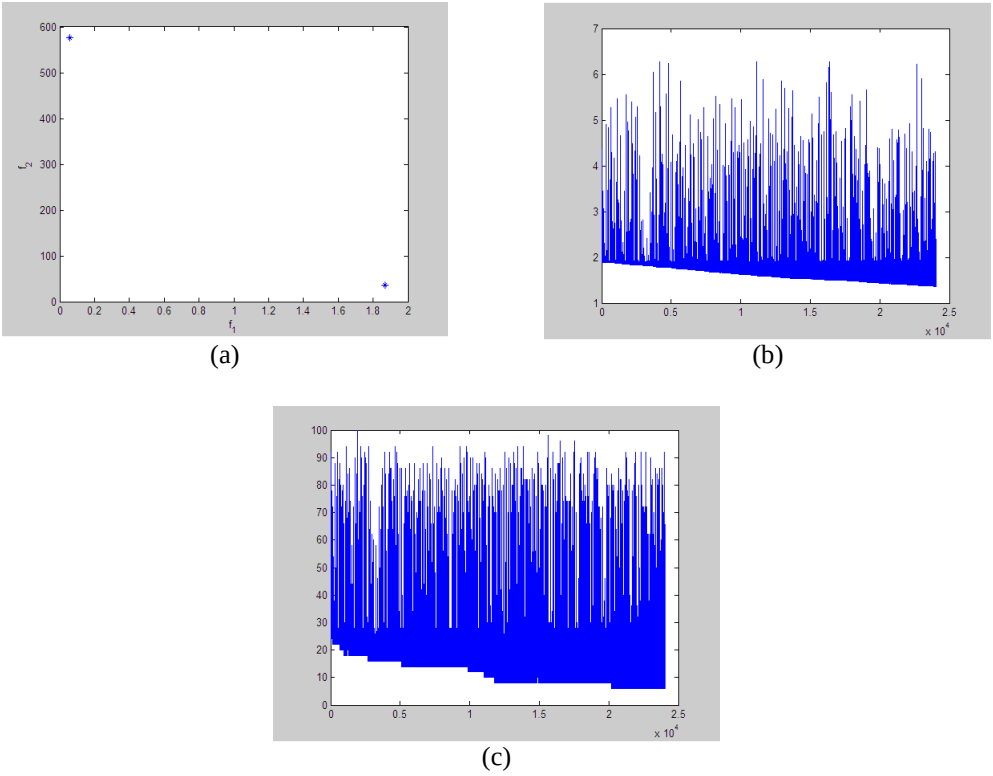


Figure 13: The second stage in our proposal from original values without select the best them: the nominate points in fine-tuning stage shows in (a) in the last iteration for multi-objective optimization in NSGA-II algorithm (the f1 axis determines the RSME and f2 axis determines the MCR); the RSME in 1000 iterations in fine-tuning stage shows in (b); the (c) shows the MCR (in percentage) in 1000 iterations in fine-tuning stage.

3 Conclusion

In the study presented the procedure by applying the TICA filters and the NSGA-II genetic algorithms with RMSE and MCR objective functions for training of the LeNet-5 convolutional neural network in two stages. The first stage called pre-training and the second stage called fine-tuning based on optimization solution. In fact, the TICA filters and NSGA-II algorithm with simple and useful methods in computations helped us to learn the input patterns.

Furthermore, the proposal is capable of rendering in parallel processing on GPU and cloud computing (future proposal).

REFERENCES

Amrit Pratap Sameer Agarwal, and T. Meyarivan A fast and elitist multiobjective genetic algorithm: NSGA-II . 2002.

Deb N. Srinivas and K. Multiobjective function optimization using nondominated sorting genetic algorithms . 1995.

Duffner Stefan Face Image Analysis With Convolutional Neural Networks . [s.l.] : Dissertation, 2007.

Fukushima K. A neural-network model for selective attention in visual pattern recognition . 1986.

Fukushima K. Analysis of the process of visual pattern recognition by the neocognitron . 1989.

Fukushima K. Cognitron:A self-organizing multi layered neural network . 1975.

Fukushima K. Neocognitron: A self-organizing neural-network model for a mechanism of pattern recognition unaffected by shift in position . 1980.

Imagawa K. Fukushima and T. Recognition and segmentation of connected characters with selective attention . 1993.

Wiese D. Hubel and T. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex . 1962.

K. Kavukcuoglu M.A. Ranzato, R. Fergus, and Y. LeCun Learning invariant features through topographic filter maps . 2009.

Koray Kavukcuoglu, Marc'Aurelio, Ranzato, Yann LeCun Fast Inference in Sparse Coding Algorithms with Applications to Object Recognition . New York : [s.n.], 2008.