ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 111-116

FlexNet
A Flexible Neural Network Construction Algorithm

Karim Mohraz & Peter Protzel
Bavarian Research Center for Knowledge-Based Systems (FORWISS)
Am Weichselgarten 7, 91058 Erlangen, Germany
mohraz@forwiss.uni-erlangen.de

Abstract. Dynamic neural network algorithms are used for automatic network design in order to avoid time consuming search for finding an appropriate network topology with trial & error methods. The new FlexNet algorithm, unlike other network construction algorithms, does not underlie any constraints regarding the number of hidden layers and hidden units. In addition different connection strategies are available, together with candidate pool training and the option of freezing weights. Test results on 3 different benchmarks showed higher generalization rates for FlexNet compared to Cascade-Correlation and optimized static MLP networks.
Keywords: network construction, generalization, Cascade-Correlation.

1. Introduction

For neural networks, it is equally important to find an optimal network topology as it is to determine an optimal set of weights. The use of dynamic neural network algorithms considerably speeds up the process of finding an appropriate network topology for a given problem. Dynamic network algorithms can be divided into 3 categories: network construction, pruning, and genetic algorithms for network design. The FlexNet algorithm presented here is a network construction algorithm. A number of network construction algorithms have been developed so far: Upstart algorithm [1], Add&Remove [2], Cascade-Correlation [3], to name a few. However, the drawback with these procedures is that they either have been designed for the use of binary neurons [1] or underlie constraints such as a limited number of layers and hidden units [2], [3]. FlexNet, on the other hand, creates networks with as many layers and as many hidden units as are needed to solve a given problem. In addition, the user is able to choose between different connection strategies and has the option of freezing weights.

2. The FlexNet Algorithm

The philosophy of FlexNet is based on strategies used in Cascade-Correlation (CasCor): it starts with only the input and output layers and incrementally builds up a complex network architecture by first training candidate neurons and then installing the best ones. However, as the name implies, FlexNet is a highly flexible and powerful network construction algorithm, which is not limited by constraints such as CasCor's one-neuron layers and deeply cascaded structure.
The main aspects of FlexNet can be summarized as follows:

- variable number of hidden layers and units in these layers,
- variable cross-cut connections,
- variable candidate pool training,
- possibility of freezing weights.

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 111-116

2.1 Dynamic network construction

Similar to the CasCor procedure, FlexNet consists of two training phases: a main and a candidate training phase. In the main training phase, the current network is trained until a satisfactory performance is obtained or error stagnation is observed. In the latter case, the algorithm switches to the candidate training: Candidate units are trained separately at different positions in the hidden layers. The best candidate, i.e. the one that contributes to the highest error reduction rate, is permanently installed in the network. CasCor does not consider the benefits of installing new units in existing hidden layers. Instead, after each candidate training phase, it creates a new one-neuron hidden layer resulting in deeply cascaded networks with poor generalization ability. FlexNet, on the other hand, allows multiple units per hidden layer. Furthermore FlexNet does not necessarily install candidates in the newly created layer, but also checks candidates' performances in existing hidden layers.

In addition, FlexNet does not only train individual candidates (as CasCor does) but is able to train and install sets of several candidates, which has positive effects on both convergence speed and generalization. By training not only one set of candidates, but a pool of several sets of candidates, the chances to install weak candidate units decrease and weight space is searched more effectively [3].

An example of network construction through FlexNet is given in Figure 1 a) - c):

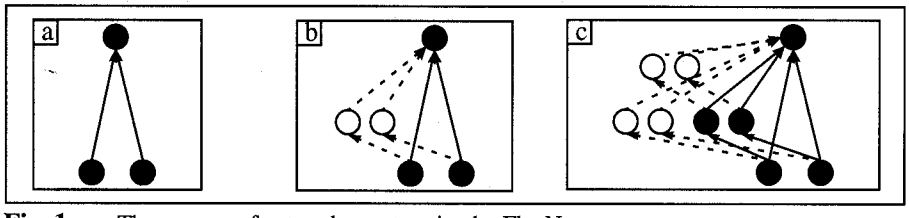

Fig. 1. The process of network construction by FlexNet.

FlexNet starts out with the input layer connected to the output layer (a). This linear network is trained until error stagnation is observed. Given that the candidate pool contains 3 different sets of candidates (with weights initialized at random), and each set consists of 2 units, these 3 candidate pairs are successively connected to the network and trained (b). The best candidate set is then permanently installed. The newly developed network is trained again until error stagnation is observed. Now there are 2 possible insertion points for the candidate sets (c): either the existing first hidden layer is expanded or a new hidden layer between output layer and the existing hidden layers is created. For every insertion point, a new pool of candidate sets is provided and trained (at this stage this means: 3 candidate sets in the first hidden layer and 3 candidate sets in the second hidden layer). Again the best candidate set is installed in the network and the procedure is repeated until a satisfactory error level is reached.

2.2 Connection strategies

Another new and valuable aspect of FlexNet is its modifiable connection structure. The user can choose how new candidates should be connected to the input, output, and al-

ready installed hidden units. There are various connection strategies ranging from the standard MLP (without shortcut connections) to the limited fan-in random wired method [4].

In this version of FlexNet, 3 connection strategies are implemented:

- *Adjacent Layers*: Like in standard MLPs only adjacent layers are connected. (Due to FlexNet's incremental network construction a limited number of shortcut artefacts to the output layer will remain.)
- *Full*: Units in fully connected networks have direct weight connections to all other neurons in the network, except for the neurons in the same layer.
- *Medium*: Every new unit is connected to the output units as well as to all units in the lower layers. This strategy is a compromise between *Full* and *Adjacent Layers*.

Weight freezing is also possible in FlexNet. To tackle the moving target problem, input weights of hidden units are frozen (after they are installed in the network) to prevent the unlearning of previously learned features [3]. The freezing option is used only with the *Medium* connection strategy.

3. Benchmarks

FlexNet is tested on 3 benchmark problems. The results are compared to static MLPs and CasCor. For a fair comparison to the dynamic networks, an appropriate MLP network, with its architecture and parameters hand-tuned [8], had to be found for each benchmark. Note that the number of runs needed to obtain these optimized MLPs is not reported here. Resilient Propagation (RP) [6] and Quickprop (QP) [5] were used as learning paradigms. The neural network simulator used was FAST [7], which contained the MLP and CasCor algorithms. 10 runs with both Rprop and Quickprop were made for each of the MLPs, CasCor, and the FlexNet flavors (named after their connection strategy): FlexAdj, FlexFull, FlexMed, and FlexFreeze. The number of candidate sets in the pool was generally set to 3 sets, but the number of candidates per set was varied. The main criterion to evaluate network performance was the classification error. In addition, training time (epochs) and an estimate of the network size (average number of hidden units) are provided.

3.1 2-Spirals

The network's task in this benchmark is to distinguish between two intertwined spirals, which coil around each other three times [9]. Each spiral is represented by 96 training points. The test data has the same size as the training set, with each test point lying between two training points on the same spiral. Table 1 shows the results for each network category. FlexAdj was not able to solve this task possibly because of missing shortcut connections. Note that the 2-5-5-5-1 MLP was fully connected, containing all shortcut connections. FlexFull, FlexMed, and FlexFreeze had up to 30% less classification error than the static MLP and CasCor. FlexFull and FlexMed had to install ≈ 45 hidden units, but needed fewer training epochs than the optimized static MLP. Freezing weights with

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 111-116

FlexNet did not affect generalization but needed about twice as many epochs to converge.
An interesting result can be seen when the receptive fields [7] of the trained networks are plotted as shown in Fig. 2. While the MLP and CasCor did not extrapolate the shape of the spirals beyond the training data, all FlexNet networks produced outer spiral rings even in those areas, where no training data had been presented.

	Paradigm	Classification error	ø Epochs	ø Hidden units
MLP 2-5-5-5-1	RP	8.5%	7440	15
CasCor	RP	9.3%	2808	14
FlexFull	QP	6.1%	4551	43
FlexMed	QP	6.2%	4590	46
FlexFreeze	QP	6.1%	6907	50

Tab. 1. Network performances on the 2-spirals benchmark

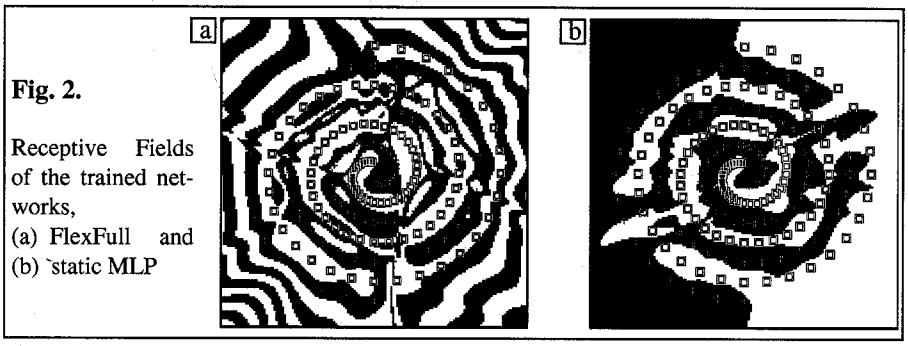

Fig. 2.

Receptive Fields of the trained networks,
(a) FlexFull and
(b) static MLP

3.2 Mexican Hat

The function $z = -(4(x^2+y^2)-2)e^{\frac{-(x^2+y^2)}{0.72}}$, to be approximated by the networks, is shaped like a Mexican Hat (Fig. 3). The networks were trained until the sum squared approximation error fell below 0.1. The training data were taken from a 40 x 40 grid of the Mexican Hat function within the interval -2.5 < x, y < 2.5.

	Paradigm	ø Epochs	ø Hidden units
MLP 2-14-10-1	QP	875	24
CasCor (mixed activation fcts.)	QP	15000	55
FlexAdj	RP	5251	49
FlexFull	RP	2872	33
FlexMed	RP	3796	23
FlexFreeze	RP	5246	57

Tab. 2. Network performances on the Mexican Hat benchmark

The static MLP required 875 epochs for a satisfactory approximation of the Mexican Hat function. However, it took a considerable amount of time in order to find the optimal architecture.
FlexAdj, FlexFull, and FlexMed required between 3000 - 5000 epochs and installed

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 111-116

more than 24 hidden neurons except for FlexMed, which reached about the same dimension as the static MLP, but used crosscut connections. Weight freezing once again slowed down FlexNet and an average of 57 units had to be installed. CasCor could not converge in 15000 epochs. Only with mixed activation functions [3] (candidate units were assigned at random Bessel, Cosine, Sine, Gaussian, and Sigmoid activation functions) could CasCor converge in 8000 epoch with an average of 70 hidden units in 70 cascaded layers! The generalization result is very poor as can be seen in Fig. 3b, especially when compared to the smooth approximation of the other network types.

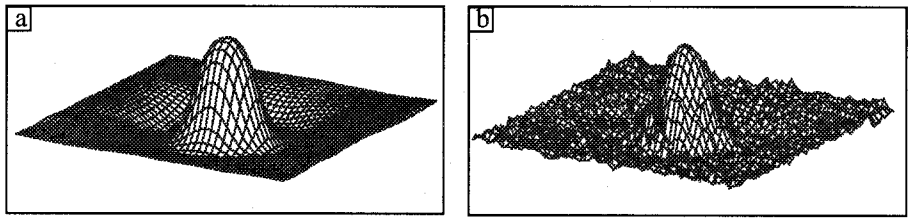

Fig. 3. Approximation of the Mexican Hat Function, by (a) FlexMed and (b) CasCor

3.3 Breast cancer classification

The breast cancer database was initially obtained from the University of Wisconsin [12] and arranged as a neural network benchmark in the Proben1 collection [11]. A tumor is to be classified as benign or malignant depending on input attributes such as clump thickness, uniformity of cell size and shape, the amount of marginal adhesion, and the frequency of bare nuclei. The data set consists of 9 input and 2 output attributes. 65.5% of the examples are benign. The training set contains 350 data sets and the test set 349 data sets.

	Paradigm	Classification error	ø Epochs	ø Hidden units
MLP 9-8-4-2	RP	6 %	2568	12
CasCor	QP	5.4 %	2448	13
FlexAdj	RP	5 %	3100	33
FlexFull	RP	5.4 %	3212	37
FlexMed	RP	4.5 %	3015	38
FlexFreeze	RP	4.7 %	5442	49

Tab. 3. Network performances on the breast cancer benchmark

FlexMed obtained the lowest percentage of classification error with an average of 3015 epochs and 38 hidden units, followed by FlexFreeze, which required, due to its weight freezing technique, an additional 2500 epochs as well as 10 hidden units. Classification error of FlexAdj was 1 percentage point lower than the error of the static MLP, using an average of 33 hidden units. MLP and CasCor required the least number of epochs and hidden units, at the same time however, produced the highest classification error.

4. Conclusion

The design of neural networks is a difficult and time-consuming job. The new algorithm presented here significantly reduces the amount of time and effort needed in order to

find an optimized network topology. Unlike other network construction methods, no constraints are imposed on the FlexNet procedure regarding the number of hidden layers and hidden units in these layers. Features such as different connection strategies, candidate pool training, and the option of freezing weights enables FlexNet to build optimal networks for a given problem. In the tested benchmarks, networks created by FlexNet achieved higher generalization rates than static MLP and CasCor networks, although a considerable amount of time was spent on fine-tuning the MLPs. FlexNet, using *Medium* connection strategy, consistently obtained a high generalization accuracy with acceptable convergence speed as well as number of required hidden units. This shows, that high order feature detectors and fully connected networks containing all shortcut connections are not always needed in order to solve a problem. CasCor's generalization ability was impaired by its one-neuron hidden layers. FlexFreeze, which also used the weight freezing technique but allowed more hidden units per layer, did not suffer from generalization problems. However, convergence took somewhat longer and a greater number of hidden units were installed compared to FlexMed.
The objective of future work is to incorporate crossvalidation training into FlexNet in order to increase generalization. The performance of candidate sets will then be evaluated on an additional validation set instead of the training set.

References

[1] Frean, M.: *"The Upstart Algorithm, A Method for Constructing and Training Feedforward Neural Networks"*, Neural Computation 2, 1990.
[2] Hirose, Y., Yamashita, K., Hijiya, S.: *"Back-Propagation Algorithm which varies the Number of hidden Units"*, Neural Networks, Vol. 4, pp 61-66, 1991.
[3] Fahlman, S., Lebiere, C.: *"The Cascade-Correlation Learning Architecture"*, Technical Report CMU-CS-90-100, Carnegie Mellon University, Pittsburgh, PA, August 1991.
[4] Klagges, H., Soegtrop, M.: *"Limited Fan-in Random Wired Cascade-Correlation"*, IBM Research Division, Physics Group, Munich.
[5] Fahlman, S.: *"An Empirical Study of Learning Speed in Back-Propagation Networks"*, CMU-CS-88-162, Carnegie Mellon University, Pittsburgh, PA, September 1988.
[6] Riedmiller, M.: *"Rprop - Description and Implementation Details"*, Technical Report, Institut für Logik, Komplexität und Deduktionssyteme, Universität Karlsruhe, 1994.
[7] Mohraz, K., Arras, M.: *"Forwiss Artificial Neural Network Simulation Toolbox"*, Bavarian Research Center for Knowledge-Based Systems (FORWISS), Internal Report, Erlangen, 1994.
[8] Arras, M., Protzel, P.: *"Assessing Generalization by 2-D Receptive Field Visualization"*, Bavarian Research Center for Knowledge-Based Systems (FORWISS), Internal Report, Erlangen, 1993.
[9] Mohraz, K.: *"Neuronale Netze mit dynamischer Architektur"*, Thesis, Research Center for Knowledge-Based Systems (FORWISS), Erlangen, 1994.
[10] Lang, K., Witbrock, M.,: *"Learning to tell two spirals apart"*, Proceedings of the 1988 Connectionist Models Summer School, M. Kaufmann Publ., 1988.
[11] Prechelt, L.: *"PROBEN1 - A set of Neural network Benchmark Problems and Benchmarking Rules"*, Technical Report 21/94, Universität Karlsruhe, 1994.
[12] Mangasarian, O., Wolberg, W.: *"Cancer diagnosis via linear programming"*, SIAM News, Volume 23, Number 5, 1990.