Generalisation Performance vs. Architecture Variations in Constructive Cascade Networks

Suisin Khoo and Tom Gedeon

School of Computer Science
College of Engineering and Computer Science
Australian National University
ACT 0200 Australia
{suisin.khoo,tom}@cs.anu.edu.au

Abstract. Constructive cascade algorithms are powerful methods for training feedforward neural networks with automation of the task of specifying the size and topology of network to use. A series of empirical studies were performed to examine the effect of imposing constraints on constructive cascade neural network architectures. Building *a priori* knowledge of the task into the network gives better generalisation performance. We introduce our Local Feature Constructive Cascade (LoCC) and Symmetry Local Feature Constructive Cascade (SymLoCC) algorithms, and show them to have good generalisation and network construction properties on face recognition tasks.

1 Introduction

The functionality and complexity of a backpropagation trained neural network (NN) are greatly influenced by the network architecture, number of neurons, and neuron connectivity. The greater number of connections and the larger the possible magnitude of the weights, the better the NN is able to model complex functions.

A constructive cascade neural network is a feedforward neural network in which the network architecture is built during the learning process, in order to obtain a good match between network complexity and the complexity of the problem to solve [1].

In some highly regular tasks such as image recognition, a priori knowledge about the task can be built into the network for better generalisation performance. Classical works in visual pattern recognition have shown the advantages of using local features and combining them to form higher order features. Extracting local features can be viewed as a way of reducing the space of possible functions that can be generated without overly reducing the network's computational power [2].

In this paper, we propose two constructive cascade algorithms that incorporate a priori knowledge about the problem to be solved, i.e. face recognition, into the design of the architecture and explore the relationship between the generalisation performance and variations on the architecture. These are the Local Feature Constructive Cascade (LoCC) and Symmetry Local Feature Constructive Cascade (SymLoCC) algorithms. They have lesser number of free parameters but preserve the good generalisation properties of constructive cascade algorithms. Unlike other face recognition technique such as in [3] our approach does not require normalisation and pre-processing of data sets or additional feature extraction procedures.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 236–243, 2009.

2 Method

2.1 Neural Network Topology

Constructive algorithms such as CasPer and Cascade Correlation (CasCor) start with the minimal size NN architecture often with no hidden neurons [4] as in Fig. 1. Initially, all input neurons are fully-connected to the output neurons. The number of connections in the initial network would be very large for face recognition, hence LoCC and SymLoCC start with an initial network with one hidden layer as shown on the left in Fig. 2.

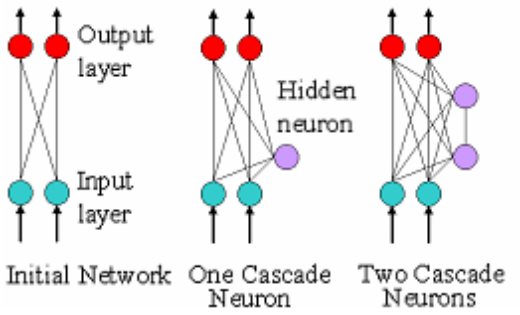

Fig. 1. Cascade Correlation

The initial architecture of our algorithm is a slight modification of the two layer 8x8 architecture proposed in [5] which was shown to give good generalisation performance on handwritten digit recognition and face recognition. Instead of a two-dimensional 16 by 16 hidden layer with a total of 256 hidden neurons, our algorithm uses a reduced number of hidden neurons, which is 8 by 8 hidden layer, to achieve smaller network size with similar good generalisation performance. Each hidden neuron functions as a local feature extractor that receives inputs from the corresponding receptive field in the input layer. The hidden neurons are thus partially connected to the input neurons but are fully-connected to the ten output neurons.

The middle diagram in Fig. 2 shows the network structure after one cascade layer is added. Instead of adding hidden neurons to the network one at a time during the learning process as in typical constructive cascade algorithms such as CasCor [6], our algorithm adds a number of cascade chunks (cascade layers), each of a fixed size set prior to training. Treadgold and Gedeon used a similar approach to build cascade towers for their algorithm [4]. In order to limit the functionality of the cascade layer and to reduce the overall number of hidden neurons, the size of each cascade layer is smaller than the hidden layer. Each cascade layer consists of 4 by 4 neurons, each extracts local features from both the input and the hidden layer in the initial network. Preserving the characteristic of constructive cascade algorithms, the cascade layers receive input from all preceding layers and are fully-connected to the output layer.

There are three variants of the two architectures presented in this paper, with the same overall structure shown in Fig. 2. They differ in the size and structure of the receptive fields. In the first variant, each neuron in the hidden layer takes its inputs from 25 neurons on the input layer situated in a 5 by 5 square neighbourhood. For neurons in the hidden layer that are one unit apart, their receptive fields in the input layer are four units apart. Hence, their receptive fields overlap by one row or column of units.

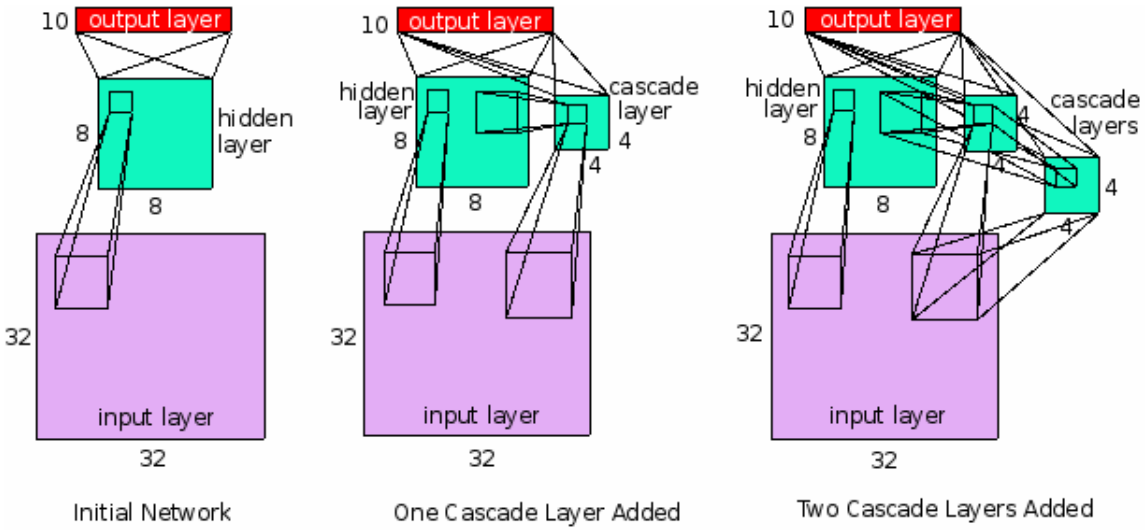

Fig. 2. Local Feature Constructive Cascade Neural Networks

In the first variant, each neuron in a cascade layer takes its inputs from the input layer situated in a 9 by 9 neighbourhood and from the hidden layer situated in a 3 by 3 neighbourhood. For each neuron that is adjacent, the input patches from the input layer are eight units apart and are two units apart on the hidden layer. In other words, 9 by 9 patches on the input plane have one row or column of unit overlapped and the 3 by 3 patch on the hidden layer has one row or column of unit overlapped. Each cascade layer has one-to-one connection to all preceding cascade layers.

In the second variant, each hidden neuron receives 36 inputs from a 6 by 6 square neighbourhood in the input layer with 2 rows or columns of units overlapped. In the cascade layer, each neuron receives inputs from a 10 by 10 patch with 2 rows or columns of neurons overlapped. Neurons in the cascade layer also receive inputs from 16 hidden neurons in the hidden layer, arranged in 4 by 4 receptive fields with two rows or columns of neurons overlapped. As in the first variant, all neurons in each cascade layer have one-to-one connections with all preceding cascade layers.

Finally, the third variant employs larger receptive fields than both previous architectures. Hidden neurons in the initial network each takes a 7 by 7 receptive field in the input layer. Neurons in cascade layers each takes a 11 by 11 receptive field in the input layer and from the 5 by 5 receptive field in the hidden layer. As before, each cascade layer has one-to-one connections with all preceding cascade layers.

2.2 Data Set

The two data sets used were originally proposed by Georghiades, Belhumeur, and Kriegman [7] and is available from the Yale Face Database B. The original data set contains a total of 5,850 single light source images of 10 subjects each seen under 576 viewing conditions (9 poses of 64 illumination conditions and 1 with ambient illumination). The images are 8-bit grey scale with size 480 height x 640 width. Due to its massive size, not all of the data set in the Yale Face Database B was employed in the first data set. Thirteen images of each subject under nine different poses were randomly selected from the original data set. Of these images, 630 of them are designated as the training set, and 360 as test set. Each face image was resized using

nearest-neighbour interpolation [8] into a 24 x 32 image, with added 0s to form a 32 x 32 images. The second consists of 650 frontal view images in total, split into 450 training data and 200 testing data, randomly.

2.3 Training Methodology

Human faces possess some symmetric characteristics. SymLoCC implements this knowledge by adjusting all weights leaving the input layer so that the weights for neurons in the right-half of the square plane mirror the value of the weights in the left-half of the plane (i.e. the weights are shared). Hence, the number of free parameters from input layer to hidden and cascade layers are 50% less than in the non-symmetric algorithm LoCC.

Before training our neural networks, all weights and biases of the network are initialised using the Nguyen-Widrow method [9]. The activation function used in all networks is the hyperbolic tangent function. Hyperbolic tangent functions are symmetric functions, which are believed to be able to yield faster convergence than non-symmetric functions. Resilient propagation (RPROP) [10] was use for all networks learning. RPROP is an adaptive learning algorithm which performs a direct adaptation of the weight step size based on local gradient information. Instead of taking into account the magnitude of the error gradient as seen by a particular weight, RPROP uses only the sign of the gradient. This allows the algorithm to adapt the step size without the size of the gradient interfering with the adaptation process. Also, a weight biasing technique [11] is employed to bias the search direction of the RPROP algorithm in favour of the weights of the newly added neurons, setting different initial update values in the RPROP algorithm.

For each architecture variant, the network starts from the initial architecture and the learning process continues for a maximum of 100 epochs. A new cascade layer is added to the network when either the maximum epoch is reached or the MSE is less than or equal to 0.03. The input patterns were presented in a consistent order, batch trained. Each experiment was performed 5 times with different initial conditions. The performance function is the standard mean squared error (MSE), the output layer was composed of 10 units, one per class, and we used a winner-takes-all method to classify the networks' output error on the test data set.

3 Experimental Results

We have evaluated our method using two subsets of Yale Database B as described in Section 2.2. The problem to be solved is face recognition of ten subjects. Information of each face image is input into the network through 32 x 32 that is 1,024 input neurons. The 10 output neurons each represent one of the subjects. Table 1 shows the average of the best performance of the five repetitions of each experiment.

Addition of each cascade layer increases the number of weights by some 1,500 to 2,000. These increases lead usually to some increase in performance for each architecture-variant combination. The performance of most networks is quite good, ranging from a few results in the vicinity of 80%, but with more than half of the architecture-variants with results over 95%.The average of the best performance of the five

Table 1. Average of best performance on both data sets

Architecture			Number of weights	Generalisation Performance % (all poses)	Generalisation Performance % (frontal pose)
LoCC	551-991-331	initial network	2,314	94.2	95.7
		cascade layers: 1	3,930	95.2	96.5
		2	5,562	97.7	99.5
		3	7,210	98.6	99.0
	662-10102-442	initial network	3,018	94.5	93.5
		cascade layers: 1	5,050	94.8	95.0
		2	7,098	98.2	99.2
		3	9,162	97.7	99.1
	773-11113-553	initial network	3,850	92.2	92.9
		cascade layers: 1	6,362	94.7	95.8
		2	8,890	98.6	98.8
		3	11,434	97.6	98.6
SymLoCC	551-991-331	initial network	1,514	83.3	92.0
		cascade layers: 1	2,482	90.8	94.1
		2	3,466	97.9	99.1
		3	4,466	97.6	98.8
	662-10102-442	initial network	1,866	82.6	88.5
		cascade layers: 1	3,098	91.9	95.0
		2	4,346	97.8	98.4
		3	5,610	97.7	98.5
	773-11113-553	initial network	2,282	86.7	81.8
		cascade layers: 1	3,826	93.6	94.1
		2	5,386	97.8	97.6
		3	6,962	98.3	98.7

repetitions of each experiment performed using the first data set is plotted in Fig. 3, and similarly for the second data set in Fig. 4. The connecting lines in the graph show the growth of complexity of each architecture from the initial network to the final network structure with three cascade layers.

In Fig. 3 we can see a number of patterns. Most obviously, as the number of weights increases, there is generally an increase in the generalisation performance of the networks. The best results are those located in the top-left corner, hence the best result is either *LoCC-551-991-331 initial network*, or *SymLoCC-551-991-331 with two cascade layers*. The choice between them would be context dependent. I.e., is the 50% increase in number of weights worthwhile for 3.7% increase in performance? Alternatively stated, is this worthwhile for a $^{2}/_{3}$ decrease in the remaining error (from 5.8% to 2.1%)? In subsequent discussion, we will focus on differential effects of our various architectural variations, and the effect of sequential addition of cascade layers.

For the first of our architectures, the local feature constructive cascade (LoCC) network, shown in Fig. 3 by the solid lines, we can see that addition of a cascade layer increases the weights appreciably, while only slightly increasing the generalisation ability. This pattern holds for the 2nd and 3rd variants, as they also show little improvement in generalisation. The addition of a second cascade layer has a beneficial effect, in that there is a more significant increase in generalisation ability. The addition of a third cascade layer is slightly detrimental in two cases and mildly beneficial in one case, so this is not useful as we add weights without improving generalisation ability much or at all.

Fig. 3. LoCC and SymLoCC results using the first data set (all poses)

For the second architecture, using symmetry (SymLoCC), shown in Fig. 3 by dashed lines, the performance starts at much lower values. The addition of a cascade layer improves generalisation for all three variants. At this point the second architecture variants with a cascade layer have very much the same number of weights as the first architecture and no cascade, but with lower generalisation. On the addition of a second cascade layer, again we find an increase in performance for all three variants, and now we achieve a situation of lower weights but higher performance than the next step of the first architecture. A third cascade layer produces little effect.

We can note a few subtle observations. For the second architecture, as we move from the 1st variant to the 3rd, adding the first two cascade layers, the slope of improvement decreases from 1st variant to 3rd variant on the addition of the first cascade layer, and further decreases in the same way on the addition of the second cascade layer. A possible explanation is that the patch sizes are overall better in the 1st variant and worsen to the 3rd variant.

In Fig. 4, we have slightly better data, in that the frontal poses are by definition more symmetrical and might benefit more from the symmetry incorporated in our second architecture. No pre-processing has been done to align faces in the images.

Our first architecture (solid lines) has similar effects as in Fig. 3, in that the addition of the first cascade layer has some effect, which is increased by the second cascade layer. At this point the overall best result is reached, the 1st variant with two cascade layers has 99.5% accuracy on the test set. A third cascade layer is not beneficial.

Fig. 4. LoCC and SymLoCC results using the second data set (frontal pose only)

Our second architecture (dashed lines) also has similar results to Fig. 3, in that results start lower than the first architecture, the addition of two cascade layers produces results which are substantially improved, and better than the first architecture.

For Fig. 4, we can make a different illustration of the tradeoff if we assume that absolute performance is most important. In that case the best networks have 3,466 weights 99.1%, and 5,562 weights and 99.5% generalisation performance, respectively. Depending on the measurement error, the percentage results may not be reliably distinguishable so clearly the version with significantly lower weights is best.

4 Conclusion and Future Work

We have introduced our algorithm for constructing cascade networks for face recognition using our notion of cascade layers, by constraining the cascade process by adding chunks of 4 x 4 neurons. We have examined a number of variants of this model, focusing on the sizes of patches taken by each layer from the preceding layer. We have extended this model by introducing a symmetry component. From our testing we have shown that our models work well on a standard face image database. We have demonstrated that restricting the data to just the frontal pose improves all our results. An alternative expression of this statement demonstrates the strength of our approach: if we take the frontal pose as the baseline, then using multiple different poses cost only a 0.9% drop in maximum performance.

Our future work will be to further examine the architecture variations in our model. The three variations discussed here increased all the patch sizes from 1st variant to 2nd, and from 2nd variant to 3rd variant, however the patch sizes could be varied independently. We will also investigate the effects of aligning the frontal pose images on the images.

References

1. Kwok, T.-Y., Yeung, D.-Y.: Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems. IEEE Trans. on Neural Networks 8, 630–645 (1997)
2. LeCun, Y.: Generalization and Network Design Strategies. Technical report, Dept. of Computer Science, University of Toronto (1989)
3. Grudin, M.A.: On Internal Representations in Face Recognition Systems. Pattern Recognition 33, 1161–1177 (2000)
4. Treadgold, N.K., Gedeon, T.D.: Exploring Architecture Variations in Constructive Cascade Networks. In: IEEE Int. Jt. Conf. on Neural Networks, pp. 343–348 (1998)
5. Khoo, S.: Application of Shared Weight Neural Networks in Image Classification. Honours Thesis, Dept. Computer Science, Australian National University (2008)
6. Fahlman, S., Liebiere, C.: The Cascade-Correlation Learning Architecture. Technical report, School of Computer Science, Carnegie Mellon University (1990)
7. Georghiades, A.S., Belhumeur, P.N., Kriegman, D.J.: From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose. IEEE Trans. on Pattern Analysis and Machine Intelligence 23, 643–660 (2001)
8. Cover, T.M., Hart, P.E.: Nearest Neighbor Pattern Classification. IEEE Trans. on Information Theory IT-13, 21–27 (1967)
9. Nguyen, D., Widrow, B.: Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights. In: IEEE Int. Jt. Conf. on Neural Networks, pp. 21–26 (1990)
10. Riedmiller, M., Braun, H.: A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: IEEE Int. Conf. on Neural Networks, pp. 586–591 (1993)
11. Treadgold, N.K., Gedeon, T.D.: Increased Generalization through Selective Decay in a Constructive Cascade Network. In: Proc. 1998 IEEE Int. Conf. on Systems, Man, and Cybernetics, pp. 4465–4469 (1998)