PROGRESSIVE IMAGE COMPRESSION

T.D. Gedeon[1] & **D. Harris**[2]

[1] School of Computer Science and Engineering,
The University of New South Wales, Australia
[2] Centre for Neural Networks,
King's College London, U.K.

ABSTRACT

Most applications of neural networks for image compression have emphasised the degree of compression. In many applications the main consideration is the decompressed image quality. We can guarantee a consistent level of functionality of units in the compression layer based on their distinctiveness, and can progressively reduce the size of the compression layer for the desired level of image quality.

1. INTRODUCTION

In this paper we will generally assume a feed-forward network of three layers of processing units. All connections are from units in one level to the subsequent one, with no lateral, backward or multilayer connections. Each unit has a simple weighted connection from each unit in the layer above. The hidden layer layer consists of fewer units than the input layer, thus compressing the image. The output layer is the same size as the input layer, and is used to recover the compressed image. The network is trained using a training set of patterns with desired outputs being the same as the inputs, using back-propagation of error measures [1]. By back-propagation we mean the general concept of developing the error gradient with respect to the weights, and not restricted to the standard gradient descent method. In the examples we cite here we have used the the basic logistic activation function $y=(1-e^{-x})^{-1}$, again this is not germane to the generality of our method. Training by back-propagation is popular because of its simplicity theoretically, and the ease of use and production of such networks. The method has been used successfully in a number of disparate areas ranging from pattern synthesis [2], to image compression [3, 4].

2. DISADVANTAGES OF BACK-PROPAGATION

The major disadvantages of the back-propagation method are that it can be slow to train networks, and that the architecture required for a solution to a problem is not currently determinable *a priori*. In practice, it is deciding the number of hidden units which is most difficult. Many workers have remarked that to train networks successfully or at an effective rate, more hidden units are required than the minimal number. If the new units end up duplicating the functionality of existing units, we have gained nothing, but have decreased the speed of the network by increasing its size. Furthermore, we have lost significant time if we have restarted the training process from scratch.

In the case of image compression, the degree of compression desired could be used to determine a hidden layer size. Given these observations, however, we should use a somewhat larger size initially. Nevertheless, for the measure of the degree of compression to be meaningful, any units with redundant functionality after training must be removed. Otherwise, results are not comparable between different training regimes, as well as inconsistent using the same regime, given that with different initial random weights, a different number of functionally useless units may result.

Brute force methods to find minimal size networks by eliminating randomly chosen units from trained networks have been used; recently some approaches have emerged delineating properties to choose which units should be eliminated.

3. PROPERTIES

The seminal work on pruning trained networks [5] uses the outputs of units in a two stage pruning process, which operates by inspection. Such pruning by inspection is difficult even on small examples, some automatable process would be ideal. Properties such as *relevance* [6, 7], *contribution* [9], *sensitivity* [10], *badness* [11], and *distinctiveness* [12] have been described in detail elsewhere. We will briefly describe *distinctiveness* here.

The *distinctiveness* of hidden units is determined from the unit output activation vector over the pattern presentation set [12]. That is, for each hidden unit we construct a vector of the same dimensionality as the number of patterns in the training set, each component of the vector corresponding to the output activation of the unit. This vector represents the functionality of the hidden unit in (input) pattern space. In this model, vectors for clone units would be identical irrespective of the relative magnitudes of their outputs and be recognised. Units with short activation vectors in pattern space are recognised as insignificant and can be removed.

Pattern	1	2	3	4	5	6
p.000	1.000	1.000	1.000	1.000	1.000	0.000
p.001	0.000	1.000	1.000	0.000	0.000	0.000
p.002	0.000	0.000	0.000	1.000	0.000	1.000
p.003	0.000	0.000	0.000	1.000	1.000	0.000
p.004	0.000	0.000	1.000	1.000	0.000	0.000
p.005	1.000	0.439	1.000	1.000	0.999	0.706
p.006	0.000	1.000	0.000	1.000	1.000	0.000
p.007	1.000	0.000	0.000	1.000	0.000	0.000
p.008	1.000	0.000	0.000	1.000	0.000	0.000
p.009	0.000	0.000	0.000	0.000	0.000	0.000
p.010	0.000	0.000	1.000	0.000	0.167	0.000
p.011	0.000	0.000	1.000	0.000	0.000	0.000
p.012	0.000	0.000	0.000	0.000	0.000	0.000
p.013	1.000	0.000	0.000	0.989	1.000	1.000
p.014	0.000	0.000	0.000	0.000	0.000	0.000
p.015	0.000	0.000	1.000	1.000	1.000	0.000

Table 1. Six hidden unit activations by pattern.

The recognition of similarity of pairs of vectors is done by the calculation of the angle between them in pattern space. Since all activations are constrained to the range 0 to 1, the vector angle calculations are normalised to 0.5, 0.5 to use the angular range of 0-180° rather than 0-90°. Angular separations of up to about 15° are considered too similar and one of them is removed. The weight vector of the the unit which is removed is added to the weight vector of the unit which remains. With low angular separations as above, the averaging effect is insignificant and the mapping from weights to pattern space remains adequate in that the error measure is no worse subsequently.

This produces a network with one fewer unit which requires no further training. Similarly, units which have an angular separation over about 165° are complementary, and both can be removed.

It would be possible to discover pairs of similar units by inspection of the above table, but would be difficult for much larger numbers of patterns or units. The vector angles for the six hidden units are shown in Table 2.

Pair of units	Vector angle
1 2	71.7
1 3	90.0
1 4	68.0
1 5	68.9
1 6	61.6
2 3	71.7
2 4	94.1
2 5	64.5
2 6	70.9
3 4	97.1
3 5	81.5
3 6	92.2
4 5	60.9
4 6	84.9
5 6	70.6

Table 2. Vector angles for pairs of the hidden units.

Clearly, none of the units are very similar to any other.

A further category of undesirable units is also discovered and included in the distinctiveness analysis. Groups of three or more units which together have no effect, or two or more units with a constant effect can be recognised. That is, in the pattern space, the sum of their vectors is zero or constant. The discovery of such groups is done by a sorted Gaussian vector pivot on the cumulative rectangular matrix of pattern space vectors. This produces the reduced row echelon matrix. For the case of groups of units with jointly no effect, the entire group can be removed. If the joint effect is constant, a bias can replace the entire group. Because of the inaccuracies incurred by banding and subsequent use of this information, it may be necessary to retrain the network. Fortunately, such groups are not common in our experience, therefore retraining is not often required.

4. IMAGE COMPRESSION

Distinctiveness analysis has been used to analyse the functionality of units in pattern space to determine whether there are any redundant units [12], used to speed up learning by ensuring distinct unit functionality [13], to determine when units should be added to a network [14], and to improve a networks's damage resistance and robustness [15].

A 64 by 64 image with 4 bits per pixel was chosen for testing our method. The image was broken into 16 non-overlapping 16 by 16 images, as shown in Figure 1. Each 16 by 16 image is a separate training pattern.

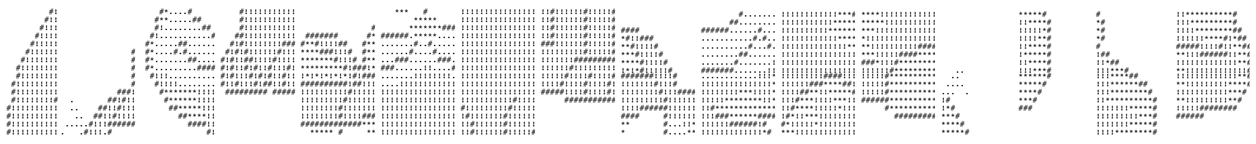

Figure 1. Broken up image

The image was broken into non-overlapping pieces so as to allow for generalisation to have clearly occurred – since there is little obvious similarity between the pieces, and particularly the large areas of white space around the edges of the image could be expected to interfere. This allowed a single image to be used, thus simplifying the discussion.

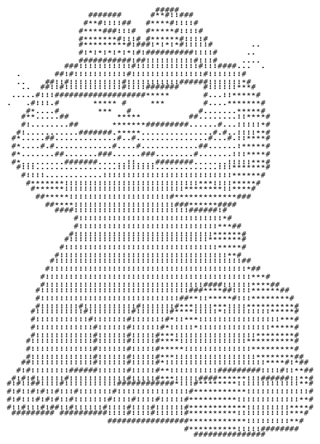

Figure 2. Complete Image

Note the tall profile of the pixels, due to using a subset of ASCII varying in image density to represent gray levels. The image was captured by one of the authors off the electronic net in the U.K. – the picture originated in the U.S. and had been distributed worldwide. As can be seen, when the pieces are correctly aligned they form a representation of a cartoon character ©Jim Davis.

While such network traffic was never the intention of any of the providers of this service, without draconian measures the best alternative is to make such traffic less expensive by transmitting compressed images. Certainly these techniques are also useful in other areas of image compression.

The network architecture used was a 256 input, *x* unit hidden layer, and 256 output network, each of the pixels in the 16 by 16 parts of the image being a single input, and output. The number of bits per pixel were mapped onto the range from 0 to 1 as input, and the output values remapped to the simulated gray scale.

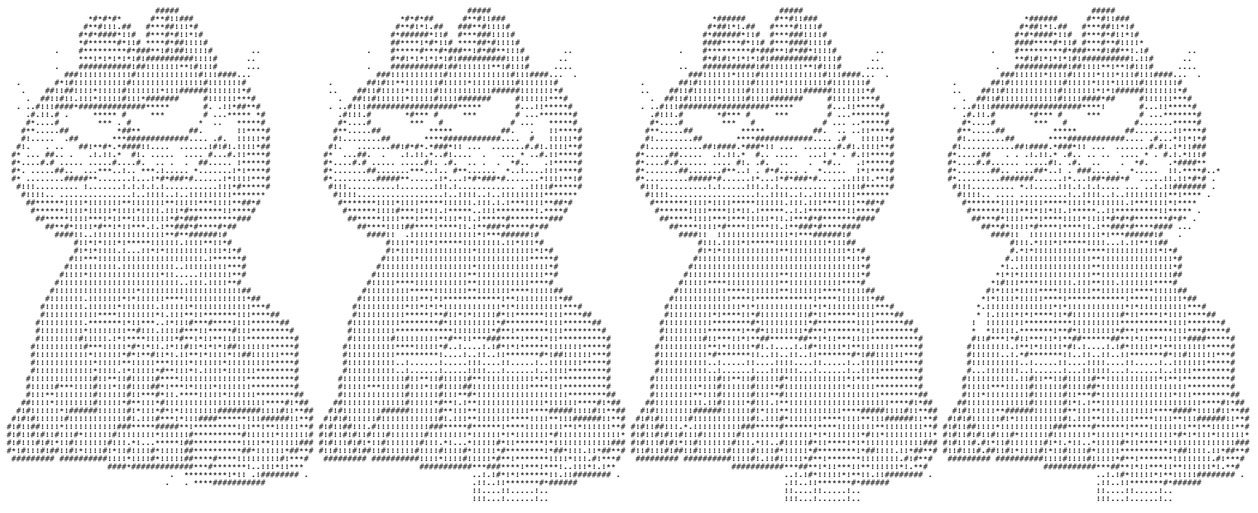

Figure 3. 16 hidden units compression 16 | 14 hidden units compression 18 | 12 hidden units compression 21 | 10 hidden units compression 26

The initial value of *x* was 16. This gives a minimum compression ratio of 16 to 1, which is reasonable given the (deliberately) generalised network architecture used. Figures 3 and 4 contain the images produced for a number of compression ratios.

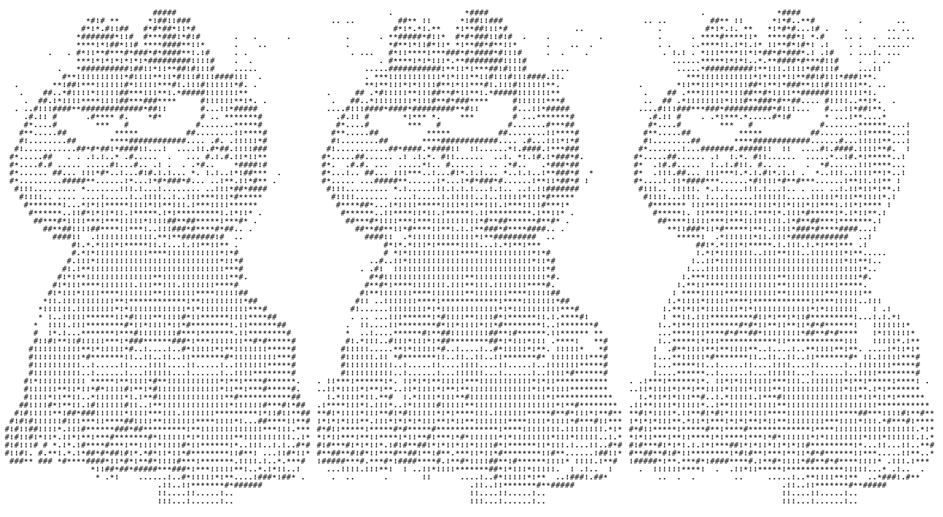

Figure 4. 8 hidden units compression 32 7 hidden units compression 37 6 hidden units compression 43

The cartoon character is clearly recognisable in the 7 unit case at a compression ratio of 37 to 1, and marginally in the 6 unit case at a ratio of 43 to 1. The use of overlapping patches, or training on a number of similar entire images [17] would boost the compression ratio of both stages of our process. The network was trained for 200 presentations of all 16 patterns.

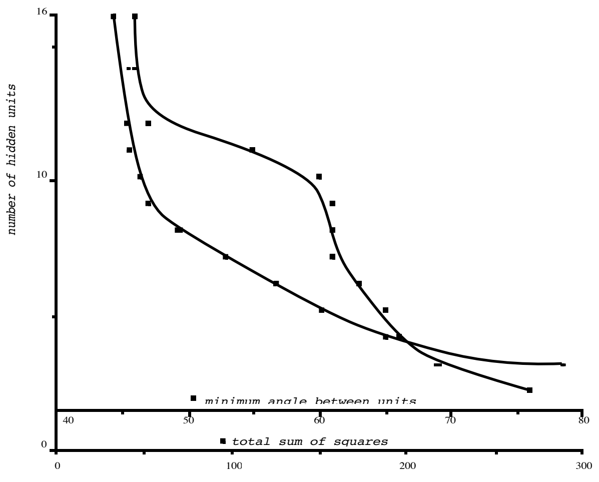

Figure 5. Units vs. image quality, unit significance

The method from [13] has been used to ensure the distinctiveness of the functionality of the hidden units. Briefly, this is done by regularly checking their angular separations, and if any units are too similar to other units, they are forced to diverge by the use of noise of appropriate magnitude applied to their weight vector. This has the effect of speeding up the learning process. At the same time, as we progressively reduce the size of the hidden layer the units we remove are significantly distinct – we are trading image quality for degree of compression. Figure 5 shows the relationship between the number of hidden units and the total sum of squares error measure which can be taken as a rough indicator of the image quality. Initially there is little drop in quality as units are removed. Beyond a certain point each further unit removed produces a significant degradation in quality. Figure 5 also shows the relationship between the number of hidden units and the smallest angle between hidden units. These values are all much higher than the standard 15° we use. It is interesting to note that the removal of the first few units in the vicinity of 60° does not cause a marked reduction in the error measure. These units are the equivalent of the secondary backup units we introduced for increasing network damage resistance in [16]. Since we are removing significant units at each stage, the network will require retraining. After the removal of a unit, the network is trained for 200 epochs.

5. CONCLUSION

We have demonstrated that using the distinctiveness of units, we can ensure maximal functionality of compression layer units to improve image compression for a given image quality.

By widening the degree of distinctiveness used by our analysis and removing significant units further compression can be achieved at the expense of image quality. We have demonstrated this using a feed-forward neural network with a simple non-specialised architecture, using a simplistic method of subdividing the image (non-overlapping patches). This approach is therefore of general utility, and could be used to improve the performance of more complex compression schemes based on the back-propagation neural network .

REFERENCES

1. Rumelhart, DE, Hinton, GE, Williams, RJ, “Learning internal representations by error propagation,” in Rumelhart, DE, McClelland, Parallel distributed processing, Vol. 1, MIT Press, 1986.

2. Lewis, J, Probing the Critic: Approaches to Connectionist Pattern Synthesis, IJCNN, vol. 1, pp. 85-89, Seattle, 1991.

3. Cottrell, G, Munro, P, & Zipser, D, “Learning internal representations of gray scale images,” Proc. 9th Ann. Cog. Sci. Soc. Conf., Seattle, pp. x-y, 1987.

4. Namphol, A, Arozullah, M, & Chin, S, “Higher Order Data Compression with Neural Networks, IJCNN, vol. 1, pp. 55-59, Seattle, 1991.

5. Sietsma, J, & Dow, RF, “Neural net pruning - why and how,” IJCNN, vol. 1, pp. 325-333, 1988.

6. Mozer, MC, Smolenski, P, “Using relevance to reduce network size automatically,”, Connection Science, vol. 1, pp. 3-16, 1989.

7. Segee, BE, & Carter, MJ, “Fault Tolerance of Pruned Multilayer Networks,” IJCNN, vol. 2, pp. 447-452, Seattle, 1991.

8. Rumelhart, DE, “Learning and Generalisation,” Proc. IEEE Int. Conf. on Neural Networks, San Diego, 1988.

9. Sanger, D, “Contribution analysis: a technique for assigning responsibilities to hidden units in connectionist networks”, Connection Science, vol. 1, pp. 115-138, 1989.

10. Karnin, ED, “A simple procedure for pruning back-propagation trained neural networks,” IEEE Transactions on Neural Networks, vol. 1, pp. 239-242, 1990.

11. Hagiwara, M, “Novel back propagation algorithm for reduction of hidden units and acceleration of convergence using artificial selection,” IJCNN, vol. 1, pp. 625-630, 1990.

12. Gedeon, TD, Harris, D, “Network Reduction Techniques,” Proc. Int. Conf. on Neural Networks Methodologies and Applications, AMSE, San Diego, vol. 2, pp. 25-34, 1991.

13. Gedeon, TD, Harris, D, “Network Reduction Techniques,” Int. Conf. on Neural Networks Methodologies and Applications, AMSE, San Diego, 1991.

14. Harris, D, Gedeon, TD, “A meta-learning strategy to improve the performance of training algorithms,” EXPERSYS-91, Paris, 1991.

15. Harris, D, Gedeon, TD, “Adaptive insertion of units in feed-forward neural networks,” Neuro-Nîmes, 4th Int. Conf. on Neural Networks and their Applications, 1991.

16. Gedeon, TD, Harris, D, “Creating Robust Networks,” IJCNN, Singapore, 1991.

17. Fleming, MK & Cottrell, W, “Categorisation of Faces Using Unsupervised Feature Extraction,” IJCNN, vol. 2, pp. 65-70, 1990.