Theoretical Results for Applying Neural Networks to Lossless Image Compression

Steve G. Romaniuk

Department of Information Systems & Computer Science

National University of Singapore

10 Kent Ridge Crescent

Singapore 0511

e-mail: stever@iscs.nus.sg

Abstract

The ability to employ neural networks to the task of image compression has been pointed out in recent research. The pre-dominant approach to image compression is centered around the backpropagation algorithm to train on overlapping frames of the original picture. Several deficiencies can be identified with this approach: First, no potential time bounds are provided for compressing images. Second, utilizing backpropagation is difficult due to its computational complexity. To overcome these shortcomings we propose a different approach by concentrating on a general class of 3-layer neural networks of 2(N+1) hidden units. It will be shown that the class $\mathcal{N}^*$ can uniquely represent a large number of images, in fact, growth of this class is larger than exponential. Instead of training a network, it is automatically constructed. The construction process can be accomplished in $\mathcal{O}_{Worst}(n) = n^4 - n^2$ time, where n is the image size. Obtainable compression rates (lossless) exceed 97% for square images of size 256.

1 Introduction

Within the last few years a surge in methods to perform image compression has occurred [2, 3, 12, 17, 18, 13]. Their motivation stems both from obtaining savings in storage and communication costs. In this paper we concentrate on lossless image compression, where the exact original image is reconstructed from its compressed counterpart. To date image compression has incorporated known standards derived from general data compression algorithms like [7, 10, 19]. An international still image standard is ISO-JPEG [18] a combination of traditional lossless and lossy image compression methods. A major impact of lossless methods is in the areas of satellite photography and medical imaging [8]. Here the reliability of reproducing exact images is extremely important.

Due to the similarity of associative networks - represent an efficient way of storing information and allowing for its recall, - with the task of image compression, it only seems reasonable to employ neural networks to this task. Besides this apparent surface similarity between the two approaches, one theoretical finding in the area of neural networks stands out and promises significant improvements in image compression compared to traditional, classical methods - Kolmogorov's mapping neural network existence theorem [9]. Plainly speaking, this theorem states that any continuous function $f : [0,1]^n \rightarrow \mathcal{R}^m$ can be implemented by a 2-layer feedforward neural network of 2(n+1) hidden units. Since the theorem holds for real valued functions f on the unit interval, it also holds for all binary functions. Now, it is a rather simple task to convert an image $I \in \mathcal{I}^n$ of the class of all square images of size n into a set of boolean functions. If C is the maximum pixel value for a given image, then $\lceil lgC \rceil$ boolean functions $f_i \in \mathcal{B}^{2\lceil lgn \rceil}, i \in [1, \lceil lgC \rceil]$ completely describe said image. Here, $\mathcal{B}^{2\lceil lgn \rceil}$ is the class of boolean functions of dimension $2\lceil lgn \rceil$. Applying the results from the existence theorem implies that there exists a 3-layer network of $2\lceil lgn \rceil$ inputs, $4\lceil lgn \rceil + 2$ hidden units and $\lceil lgC \rceil$ output units which can represent any image $I \in \mathcal{I}^n$. Assuming the network to be fully connected a total of $8\lceil lg^2 n \rceil + (1 + \lceil lgC \rceil) 4lgn + 2\lceil lgC \rceil$ connections need to be stored. A small number indeed compared to the original size of the image I which is $\lceil lgC \rceil n^2$ bits. Furthermore, as the image size n is increased total storage grows as a polynomial of degree 2, whereas the number of connections that need to be stored only grows at a rate of $\lceil lg^2 n \rceil$.

A word of caution is in place. What we have neglected to address so far is the precision required for the connection weights and the complexity of the hidden layer units transfer function. Kolmogorov's theorem provides no answer to this question and even though empirical results appear to indicate that the precision of the weights can be handled by 32-64 bits, it is no guarantee. The complexity of the transfer functions is even less clear.

Since we are dealing with binary mappings (actually these are functions) simple threshold units may suffice, but so far we have no evidence to substantiate this claim.

2 Other Approaches to Image Compression

Unfortunately, the existence theorem provides no clues to how such a network is obtained, nor in case of an equivalent theorem based on the backpropagation algorithm [16, 9], how long it would take to find such a network.

Nonetheless, in recent years, several approaches have been proposed to utilize backpropagation trained neural networks for the task of image compression. In [1] an autoassociation network is employed. An image I is split into N subpictures and each subpicture is represented by an n-dimensional vector. Compression and decompression proceeds a vector at a time. This method is both time consuming and wasteful in terms of the theoretical bound suggested by Kolmogorov's Theorem. Time consumption is dependent on the size of N and n. Therefore, appropriate subpicture sizes have to be found in order to curb training time. Obviously, as the subpicture size is reduced the number of vectors each network needs to be trained on is also reduced. This can lead to quicker solutions, but the major drawback is an increase in the number of networks that need to be stored (connection weights and other unit information). Another problem that results from the use of a training algorithm such as backpropagation is, that the final weight configuration for a subpicture is dependent on the initial weight values of the network.

In [11] a slightly different approach is proposed. Instead of dividing an image into disjoint subpictures these are allowed to overlap. Also in [11] evidence is provided that suggests to use small frame sizes (8 by 8) instead of large ones. Finally, by constraining the connection weights a decrease in training time is observed compared to the earlier method.

3 Lossless Image Compression by Network Construction

Shortcomings of previous neural networks methods applied to image compression [1, 4, 11] suggest that in order to curb compression time automatic network construction is required. Furthermore, by limiting the compressed image to a few networks instead of large sets of networks (Mougeot et al. use 5329 overlapping frames for images of size 512 x 512.), both time as well as space can be saved. To further curb storage requirements for the network, it is essential to limit as many network parameters as possible. By utilizing default weights, simplifying transfer functions and allowing for default representations, high degrees of savings can be enjoyed.

3.1 A General Class of Networks

The purpose of this section is to introduce a general class of neural networks that exhibit the properties stated above. We start by introducing the class of neural networks $\mathcal{N}^*$.

Definition 1 *The class of 3-layer neural networks $\mathcal{N}^*$ consists of N input units and has 2 hidden layers of (N+1) units each, accounting for a total of 2(N+1) hidden units. There is one output unit. The input layer is fully connected to the 1 hidden unit layer and all connection weights are equal to 1. The second layer hidden units have N input connections coming from the input layer and all weights are binary. Furthermore, every 2-layer unit is connected to exactly one 1-layer hidden unit with a connection weight of N. Finally, all 2-layer units are connected to the single output unit via connections of weight 1. All network units except 2-layer units have a bias of 0. The bias of 2-layer units is equal to -N.*

Figure 1 displays the general network structure for the class of networks $\mathcal{N}^*$.

In the following theorem we relate a general class of images $\mathcal{C}^*$ with their corresponding network counterparts in $\mathcal{N}^*$. Also, we provide a bound for the cardinality of the class.

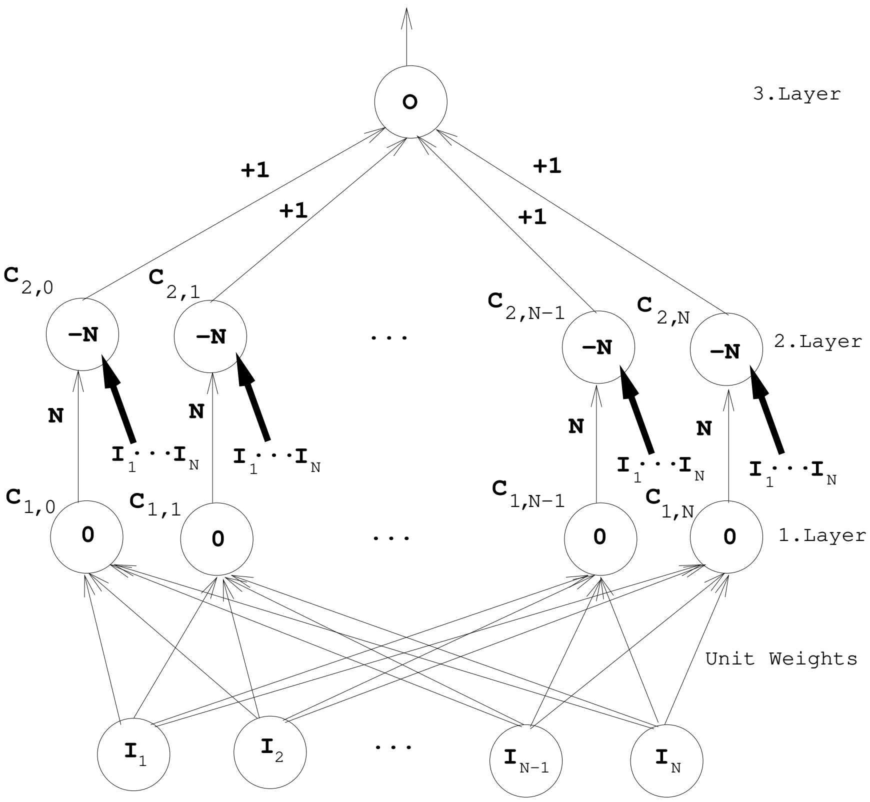

Figure 1: General Structure of Networks in $\mathcal{N}^*$

Theorem 1 *Let $C^* \subset \mathcal{B}^N$ be a subset of the class of all boolean functions of fixed input dimension N, $\mathcal{B}^N$. The class $\mathcal{C}^*$ contains those boolean functions whose training examples can be separated either by the sum of all their active inputs, or by some appropriate subset thereof. The following results hold,*

1. $\forall f \in \mathcal{C}^* \, \exists N_I \in \mathcal{N}^*$.
2. $|\mathcal{C}^*| \sim 2^{[3-\sqrt{\frac{2}{N\pi}}(3+\frac{4N}{N+2})]2^{N-1}}$.

Proof Theorem 1 *To show that there always exists a network $N_I \in \mathcal{N}^*$ for every function $f \in \mathcal{C}^*$ we observe: Given N inputs $I_1, \cdots, I_N$, we can subdivide the resulting 2^N possible patterns that describe f into $N+1$ groups $G^{'}_k, k \in [0, N]$, where k indicates the number of active inputs. By representing $N+1$ units on the first hidden layer and assigning to each hidden unit the following transfer function $T_{1,i}, i \in [0, N]$,*

$$T_{1,i}(x) = \begin{bmatrix} 1 & x = i \\ 0 & x \neq i \end{bmatrix} \tag{1}$$

we allow the patterns describing boolean function f to be recognized by exactly one of the $N+1$ hidden units based solely on the number of active inputs in said patterns.

Now, let us further refine recognition of patterns within each of these $N+1$ groups $G^{'}_k$. By dropping (removing) between 0 to $N-1$ inputs from the total of N original inputs, we can create up to N new groups. The original patterns in $G^{'}_k$ are subdivided into at most N additional subgroups $G^{'}_{k,i}$. A subgroup $G^{'}_{k,i}$ contains patterns of $G^{'}_k$ for which exactly i inputs (of the original k active inputs) are active. We consider a subdivision successful, if for all resulting subgroups the patterns contained in each subgroup have the same output (either 0 or 1).

If the above condition holds for every group $G^{'}_k$ a second layer of $(N+1)$ hidden units is created such that each unit correctly performs the above described subdivision of patterns in $G^{'}_k$. The choice of transfer function for 2-layer hidden units is then given by,

$$T_{2,i}(x) = \left[\begin{array}{ll} 1 & x > 0 \wedge x \in A_i \\ 0 & otherwise \end{array} \right. \tag{2}$$

where $A_i \subset [0, i]$ is a collection of indexes of subsets for which outputs are 1. By assigning a bias of $-N$ to each hidden unit, it is guaranteed that the unit is in an active state only when exactly i inputs are active for any presented pattern. In other words, a pattern has to belong to some group G'_k. Finally, an output unit needs to be created which receives the 2-layer hidden units as its inputs. The transfer function for the output unit is,

$$T_3(x) = \left[\begin{array}{ll} 1 & x > 0 \\ 0 & otherwise \end{array} \right. \tag{3}$$

The resulting network is displayed in Figure 1. Note, that for any pattern presented at the inputs only one of the cells labeled $C_{1,i}$ will be active. The choice of $C_{1,i}$ is determined by the number of active inputs $I_1, \cdots, I_N$. By assigning an appropriate weight vector (all weights are binary) to the N inputs of any of the 2-layer hidden units one particular subdivision (as described earlier) is implemented. If we assume that our original input pattern is recognized by unit labeled $C_{1,k}$ then the units output activation will be 1. For all other units the output will be 0. Now, since the corresponding inputs to the 2-layer units have weights of N only one of the $N + 1$ hidden units on the 2 layer will be activated. Of course the active unit will be the one corresponding to $C_{1,k}$, that is $C_{2,k}$. Depending on the selected subdivision, the input pattern will be either classified as 0 or 1 by appropriately setting the output of the 3-layer unit.

We have thus shown that every function $f \in \mathcal{C}^*$ can be uniquely represented by some network $N_I \in \mathcal{N}^*$. We next prove the second part of Theorem 1.

In general, we can state the number of functions [1] that can be generated from patterns in $G'_k, k \in [0, N]$ as,

[1] Every pattern in G'_k can be assigned an output of either 0 or 1. This assignment generates functions which belong to set G_k.

$$
|G_k| \sim \left[\begin{array}{ccccc} 2 & k & = & 0, & N \\ 2N & k & = & 1, & N-1 \\ \sum_{i=1}^{k-1}\binom{N}{i} & 1 & < & k & \leq \frac{N}{2} \\ \sum_{i=1}^{N-k-1}\binom{N}{i} & \frac{N}{2} & < & k & < N-1 \end{array} \right. \tag{4}
$$

The value for k=0 and N follows from the simple observation that only a single pattern exists for which all inputs are either 0 (k=0) or 1 (k=N). Therefore, only two possible assignments can be made to the output of each of the two patterns. Hence, $|G_0| = |G_N| = 2$.

For the case of G_1 *there exist N possible patterns. After dropping exactly 1 input from the original N, the sum of active inputs is either 0 or 1. We can drop 1 of the N inputs in* $\binom{N}{1} = N$ *ways, yielding a total of* $2N$ *possible functions. The case for* G_{N-1} *is analogous. Simply reverse the role of 1s and 0s and the case is identical to* G_1. *For the remaining values of k we observe the same similarity between* G_k *and* G_{N-k}.

For the third part of Equation (4) we observe, that if we chose to drop (or ignore) $i \in [0, k-1]$ *of the N inputs, we can form k groups from the patterns in* G'_k. *To correctly separate patterns in* G'_k *by the number of their active inputs, requires these patterns to share the same output value (either 1 or 0). Of course there exist* $\binom{N}{i}$ *ways in which i inputs can be dropped from the original N inputs. We can use this result as an estimate for the number of functions contained in* $G_k, k \in [2, \frac{N}{2}]$. *For* $k \in [\frac{N}{2}+1, N-2]$ *the results are identical to the previous ones, except that the role of 0s and 1s is reversed.*

In order to derive an estimate for the sum of the $|G_k|'s, k \in [2, N-2]$ *we observe (assuming N is even),*

$$
\sum_{k=2}^{N-2} |G_k| \sim 2 \sum_{k=2}^{\frac{N}{2}-1} \sum_{i=1}^{k-1} \binom{N}{i} + \sum_{i=1}^{\frac{N}{2}-1} \binom{N}{i}. \tag{5}
$$

Now,

$$
\sum_{i=0}^{N} \binom{N}{i} = 2 \sum_{i=0}^{\frac{N}{2}-1} \binom{N}{i} + \binom{N}{\frac{N}{2}} = 2^N. \tag{6}
$$

Therefore if follows,

$$\sum_{i=0}^{\frac{N}{2}-1} \binom{N}{i} = 2^{N-1} - \frac{\binom{N}{\frac{N}{2}}}{2}. \tag{7}$$

Let us consider the first sum,

$$\begin{aligned}
\sum_{k=2}^{\frac{N}{2}-1} \sum_{i=1}^{k-1} \binom{N}{i} &= [\binom{N}{1}] + [\binom{N}{1} + \binom{N}{2}] + \cdots + [\binom{N}{1} + \binom{N}{2} + \cdots + \binom{N}{\frac{N}{2}-2}] \\
&= (\frac{N}{2} - 2)\binom{N}{1} + (\frac{N}{2} - 3)\binom{N}{2} + \cdots + \binom{N}{\frac{N}{2}-2} \\
&= \sum_{i=1}^{\frac{N}{2}-2} (\frac{N}{2} - i - 1)\binom{N}{i} \\
&= (\frac{N}{2} - 1) \sum_{i=1}^{\frac{N}{2}-2} \binom{N}{i} - \sum_{i=1}^{\frac{N}{2}-2} i\binom{N}{i}. \qquad (8)
\end{aligned}$$

Using the previous result we can write for the first sum,

$$(\frac{N}{2} - 1) \sum_{i=1}^{\frac{N}{2}-2} \binom{N}{i} = (\frac{N}{2} - 1)[2^{N-1} - \frac{\binom{N}{\frac{N}{2}}}{2} - \binom{N}{\frac{N}{2}-1} - 1]. \tag{9}$$

We can estimate the second sum,

$$\sum_{i=1}^{\frac{N}{2}-2} i\binom{N}{i} \leq (\frac{N}{2} - 2) \sum_{i=1}^{\frac{N}{2}-2} \binom{N}{i} = (\frac{N}{2} - 2)[2^{N-1} - \frac{\binom{N}{\frac{N}{2}}}{2} - \binom{N}{\frac{N}{2}-1} - 1]. \tag{10}$$

Thus,

$$\begin{aligned}
\sum_{k=2}^{\frac{N}{2}-1} \sum_{i=1}^{k-1} \binom{N}{i} &\geq (\frac{N}{2} - 1)[2^{N-1} - \frac{\binom{N}{\frac{N}{2}}}{2} - \binom{N}{\frac{N}{2}-1} - 1] \\
&- (\frac{N}{2} - 2)[2^{N-1} - \frac{\binom{N}{\frac{N}{2}}}{2} - \binom{N}{\frac{N}{2}-1} - 1] \\
&= 2^{N-1} - \frac{\binom{N}{\frac{N}{2}}}{2} - \binom{N}{\frac{N}{2}-1} - 1. \qquad (11)
\end{aligned}$$

We can further estimate the sum over all $G_k, k \in [2, N-2]$ as,

$$\sum_{k=2}^{N-2} |G_k| \sim 2[2^{N-1} - \frac{\binom{N}{\frac{N}{2}}}{2} - \binom{N}{\frac{N}{2}-1} - 1] + [2^{N-1} - \frac{\binom{N}{\frac{N}{2}}}{2} - 1]$$

$$= \quad 3[2^{N-1} - \frac{\binom{N}{\frac{N}{2}}}{2} - 1] - 2\binom{N}{\frac{N}{2}-1}. \tag{12}$$

By using the following relationships,

$$\binom{N}{\frac{N}{2}} = \frac{n!}{\frac{N}{2}!^2} \sim \frac{(\frac{N}{e})^N\sqrt{2N\pi}}{[(\frac{N}{2e})^{\frac{N}{2}}\sqrt{N\pi}]^2} = \sqrt{\frac{2}{N\pi}}2^N. \tag{13}$$

and,

$$\binom{N}{\frac{N}{2}-1} = \frac{N}{N+2}\binom{N}{\frac{N}{2}}. \tag{14}$$

It follows,

$$\begin{aligned}
\sum_{k=2}^{N-2}|G_k| &\sim 3\,2^{N-1} - 3\sqrt{\frac{2}{N\pi}}2^{N-1} - 3 - \frac{2N}{N+2}\sqrt{\frac{2}{N\pi}}2^N \\
&= 3\,2^{N-1} - [3\sqrt{\frac{2}{N\pi}} + \frac{4N}{N+2}\sqrt{\frac{2}{N\pi}}]2^{N-1} - 3 \\
&= [3 - \sqrt{\frac{2}{N\pi}}[3 + \frac{4N}{N+2}])2^{N-1} - 3
\end{aligned} \tag{15}$$

Finally,

$$|\mathcal{C}^*| \sim 2^{[3-\sqrt{\frac{2}{N\pi}}(3+\frac{4N}{N+2})]2^{N-1}}. \tag{16}$$

q.e.d.

In the next section we provide a small but illustrative example regarding the construction of subdivisions and transfer functions.

3.2 An Illustrative Example

For this example consider $N = 4$. Table 1 lists the possible training patterns for group G_2', $|G_2'| = \binom{4}{2} = 6$, that is patterns with 2 active inputs.

By droping k of the original 4 inputs we can form up to 4 possible types of subdivisions of the original inputs:

Table 1: Examples in G_2'

Pattern	I_1	I_2	I_3	I_4
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	1	0
5	0	1	0	1
6	1	0	0	1

1. **k=0:** $\{(I_1, I_2, I_3, I_4)\}$.

2. **k=1:** $\{(I_2, I_3, I_4), (I_1, I_3, I_4), (I_1, I_2, I_4), (I_1, I_2, I_3)\}$.

3. **k=2:** $\{(I_1, I_2), (I_1, I_3), (I_1, I_4), (I_2, I_3), (I_2, I_4), (I_3, I_4)\}$.

4. **k=3:** $\{(I_1), (I_2), (I_3), (I_4)\}$.

Each of the 15 subdivisions (listed above for $k = 0, \cdots, 3$) implements a specific subnetwork $N_{i,k}^l, k \in [0,3] \wedge l \in [1, \binom{4}{k}]$ (l indexes subsets in natural order). Figure 2 displays 2 potential subnetworks, $N_{2,1}^1$ (I_2, I_3, I_4) and $N_{2,1}^3$ (I_1, I_2, I_4) for the case $k = 1$. Note, that all connection weights are 1 unless labeled otherwise. Table 2 contains 3 possible functions F_1, F_2, F_3 which can be formed from the patterns originally listed in Table 1.

Function F_1 is correctly recognized by subnetwork $N_{2,1}^1$ in Figure 2, but not by subnetwork $N_{2,1}^3$. Therefore, the transfer function for 1-layer hidden unit labeled $C_{1,2}$ in Figure 2, is given as:

$$T_{1,2}(x) = \left[\begin{array}{ll} 1 & x = 2 \\ 0 & x \neq 2 \end{array} \right. \tag{17}$$

The transfer function for the corresponding unit on the second hidden layer labeled $C_{2,2}$ is,

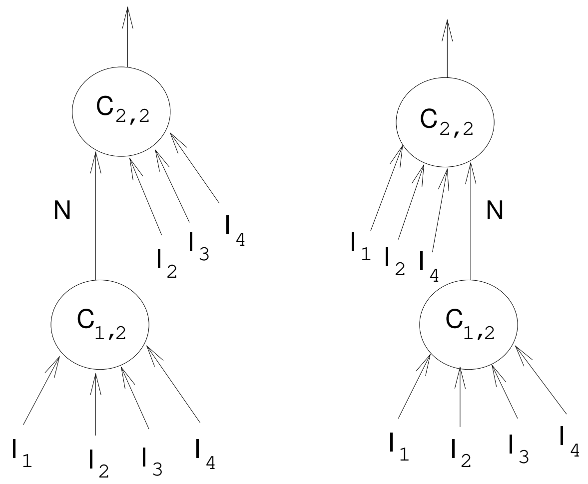

Figure 2: Subnetworks $N^1_{2,1}$ and $N^3_{2,1}$ (all weights are unit weights unless labeled otherwise).

Table 2: Example functions formed by patterns in G'_2

Pattern	I_1	I_2	I_3	I_4	F_1	F_2	F_3
1	1	1	0	0	1	0	1
2	0	1	1	0	0	1	1
3	0	0	1	1	0	1	0
4	1	0	1	0	1	1	1
5	0	1	0	1	0	0	0
6	1	0	0	1	1	0	1

Table 3: Possible subgroups generated for various values of k

k	Subgroups	# Subgroups
0	$\{2\}$	1
1	$\{1,2\}$	2
2	$\{0,1,2\}$	3
3	$\{0,1\}$	2

$$T_{2,2}(x) = \begin{bmatrix} 1 & x > 0 \ \wedge \ x \in A_2 \\ 0 & otherwise \end{bmatrix} \tag{18}$$

where $A_2 = \{1\}$.

Function F_2 can not be correctly recognized by subnetwork $N_{2,1}^1$. Consider patterns (1) and (4) of Table 2. Both patterns have 1 active input (after ignoring input I_1), but their outputs are 0 and 1, respectively. On the other hand function F_2 is correctly recognized by network $N_{2,1}^3$. Finally, function F_3 can not be identified by either network $N_{2,1}^1$ or $N_{2,1}^3$.

By dropping (or ignoring) k out of the 4 original inputs $I_1, \cdots, I_4$, we obtain the subgroups listed in Table 3.

For example, if $k = 1$ (we drop one of the inputs, $I_1, \cdots, I_4$), then either 1 or 2 inputs are always active for the patterns displayed in Table 1. This indicates that we can form 4 functions that exploit the above result using the earlier given patterns.

In Table 4 input I_1 has been dropped. The resulting 4 functions that can be created and for which subnetwork $N_{2,1}^1$ exists are also listed. This concludes the small but demonstrative example.

3.3 Network Construction Time Complexity

After having introduced the class $\mathcal{C}^*$ of boolean functions which can be represented by some 3-layer network $N_I \in \mathcal{N}^*$ we next outline an algorithm that allows us to construct said network. Figure 3 displays high level pseudo-code for algorithm NC. As a first step

Table 4: Possible functions formed for $k = 1$ from patterns in G_2'

Pattern	I_1	I_2	I_3	I_4	F_1	F_2	F_3	F_4
1	-	1	0	0	0	0	1	1
2	-	1	1	0	0	1	0	1
3	-	0	1	1	0	1	0	1
4	-	0	1	0	0	0	1	1
5	-	1	0	1	0	1	0	1
6	-	0	0	1	0	0	1	1

every image $I \in \mathcal{I}^n$ is converted into a set of binary functions $F = \{f_1, f_2, \cdots, f_{\lg C}\}$. Note, that C represents the maximum number of possible pixel values for any given image I. For simplicity, we assume this number to be a power of 2.

Binary conversion proceeds by encoding the (x, y) coordinates of every pixel in image I. Since the images under consideration are square images of dimension n, $N = 2lgn$ bits are required to convert the (x, y) coordinate into binary. These N bits are considered the inputs to network N_I. Finally, for every function $f_i, i \in [1, lgC]$ a separate network is constructed using algorithm NC.

In step (2) of the algorithm, the binary patterns representing function f_i (elements of training set T) are partitioned into $N + 1$ groups G_i'. In step (3) each group G_i' is further subdivided. Each of these subdivisions is realized by some network $N_{i,k}^l, l \in [1, \binom{N}{k}]$. The construction of such network is said to have succeeded, if all patterns contained in group G_i' are correctly recognized by some $N_{i,k}^l$. If the construction of a network for a single group G_i' fails then algorithm NC also fails to find a network representation for image I.

In Theorem 2 we provide a time analysis of algorithm NC in terms of image size n.

(1) Convert $I \in \mathcal{I}^n$ to binary

(2) For all patterns $t_j \in T$ do

(2.1) If $S(t_j) = i$ then

(2.1.1) $G'_i = G'_i \cup \{t_j\}$

(3) For all groups $G'_i, i \in [0, N]$ do

(3.1) For k=0 to N-1 do

(3.1.1) Create all networks $N^l_{i,k}, l \in [1, \binom{N}{k}]$

(3.1.2) Success = TRUE

(3.1.3) For all networks $N^l_{i,k}, l \in [1, \binom{N}{k}]$ do

(3.1.3.1) For all $t_j \in G'_i, j \in [1, \binom{N}{i}]$ do

(3.1.3.1.1) If $N^l_{i,k}(t_j) \neq OUTPUT(t_j)$ then

(3.1.3.1.1.1) Success = FALSE

(3.1.3.1.1.2) Break

(3.1.3.2) If Success then

(3.1.3.2.1) Save network $N^l_{i,k}$

(3.1.3.2.2) Break

(3.1.3.3) Else

(3.1.3.3.1) Halt;Failure

(4) Halt;Success

Figure 3: Network Construction Algorithm, NC

Theorem 2 *The class of images $\mathcal{I}^* \subset \mathcal{I}^n$ which is represented by some network $N \in \mathcal{N}^*$ can be constructed by algorithm NC under worst case conditions in,*

$$\mathcal{O}_{Worst}(n) = n^4 - n^2. \tag{19}$$

time.

Proof Theorem 2 *To execute step (1) - convert image I into its binary equivalent - requires,*

$$\mathcal{O}(n) = |T| lg|T| = 2^N lg 2^N = 2^{2lgn} lg 2^{2lgn} = n^2 lgn^2. \tag{20}$$

time. Note, that by definition $N = 2lgn$ and $|T| = 2^N$.

Calculating step (3) can be accomplished under worst case conditions in,

$$\mathcal{O}_{Worst}(n) = \sum_{i=0}^{N} \sum_{k=0}^{N-1} \binom{N}{k} \binom{N}{i} \tag{21}$$

time. Here, the first sum represents the number of possible groups G_i' that need to be considered and the second sum refers to the total number of networks that need to be tested in step (3.1.3).

We can rewrite the above equation as,

$$\begin{aligned}
\mathcal{O}_{Worst}(n) &= \sum_{i=0}^{N} \sum_{k=0}^{N-1} \binom{N}{k} \binom{N}{i} \\
&= \sum_{i=0}^{N} \binom{N}{i} \sum_{k=0}^{N-1} \binom{N}{k} \\
&= (2^N - 1) \sum_{i=0}^{N} \binom{N}{i} \\
&= 2^N (2^N - 1) \\
&= 2^{4lgn} - 2^{2lgn} \\
&= n^4 - n^2
\end{aligned}$$

q.e.d.

It is an important finding that algorithm NC can construct a network $N_I \in \mathcal{N}^*$ in polynomial time. As mentioned earlier, for previous image compression algorithms based on backpropagation, no such bounds could be provided.

3.4 Expected Degree of Compression

In the last Theorem we state the degree of compression that we anticipate for representing a given image $I \in \mathcal{I}^*$. Again we assume a worst case scenario.

Theorem 3 *The class of images $\mathcal{I}^* \subset \mathcal{I}^n$ which can be represented by some network $N_I \in \mathcal{N}^*$ requires under worst case conditions,*

$$\mathcal{O}_{Storage}(n) = 4 \ln C \ln^2 n \ln(\ln n + 1). \tag{22}$$

bits of storage.

Proof Theorem 3 *In order to find the storage requirement of image $I \in \mathcal{I}^*$ we need to determine the maximum storage for representing corresponding network $N_I \in \mathcal{N}^*$.*

To represent the weights from the input layer to the first layer does not require any storage space, since all weights are unit weights (1). We can represent this by default.

Representing the transfer functions for the 1-layer hidden units $T_{1,i}(x), i \in [0, N]$ again does not require any storage. By default, every 1-layer unit $C_{1,i}$'s transfer function $T_{1,i}(x)$ is assigned the value i.

The weights from the 1-layer hidden units to the 2-layer units can be stored using $\lceil lgN \rceil + N(N+1)$ bits. Since all units have an initial connection with fixed weight of N, $\lceil lgN \rceil$ bits are sufficient to represent this information. Finally, a total of $N+1$ 2-layer hidden units are given and each receives connections from the N input units. These weights are binary, thus yielding another $N(N+1)$ bits of storage.

To represent the 2-layer hidden units transfer functions $T_{2,i}$ requires under worst case conditions,

$$2\sum_{i=1}^{\frac{N}{2}-1}(i+1)\lceil lg(i+1)\rceil + (\frac{N}{2}+1)\lceil lg(\frac{N}{2}+1)\rceil + 2. \tag{23}$$

bits of storage. The first sum is due to the fact that for 2-layer hidden units $C_{2,i}, i \in [1, \frac{N}{2}-1]$ the set of indexes A_i contains at most $i+1$ indexes and the largest possible index is given by i. To binary encode these $i+1$ indexes requires $(i+1)\lceil lg(i+1)\rceil$ bits. For the hidden cells $C_{2,i}, i \in [\frac{N}{2}+1, N-1]$ the same number of bits are needed (the role of 0s and 1s is simply reversed). Therefore, the first sum is multiplied by a factor of 2. The second sum indicates the number of bits of storage needed for hidden unit $C_{2,\frac{N}{2}}$' transfer function $T_{2,\frac{N}{2}}(x)$ and is derived like the earlier case. Note, that we are considering even Ns in our derivation. Finally, to encode the transfer functions of hidden units C_0 and C_N 2 bits are sufficient.

No storage is needed to represent the weights from the 2-layer hidden units to the single output unit, since all weights are 1 by default. Finally, again no storage is required to represent the transfer function for the ouput unit due to its fixed behavior.

Thus, an upper bound for the total storage requirement is given by,

$$S_{NN} = 2\sum_{i=1}^{\frac{N}{2}-1}(i+1)\lceil lg(i+1)\rceil + (\frac{N}{2}+1)\lceil lg(\frac{N}{2}+1)\rceil + 2 + N(N+1) + \lceil lgN\rceil. \tag{24}$$

We can further state,

$$\begin{aligned}\sum_{i=1}^{\frac{N}{2}-1}(i+1)(\lceil lg(i+1)\rceil &\leq \sum_{i=1}^{\frac{N}{2}-1}(i+1)(lg(i+1)+1)\\ &= \sum_{i=1}^{\frac{N}{2}-1} ilg(i+1) + \sum_{i=1}^{\frac{N}{2}-1} i + \sum_{i=1}^{\frac{N}{2}-1} lg(i+1)\\ &+ \sum_{i=1}^{\frac{N}{2}-1} 1.\end{aligned} \tag{25}$$

It follows,

$$\sum_{i=1}^{\frac{N}{2}-1} i = \frac{1}{8}(N^2 - 2N). \tag{26}$$

and,

$$\sum_{i=1}^{\frac{N}{2}-1} 1 = \frac{N}{2} - 1. \tag{27}$$

For the third sum we can write,

$$\begin{aligned}\sum_{i=1}^{\frac{N}{2}-1} lg(i+1) &\leq lge \int_2^{\frac{N}{2}+1} lnxdx \\ &= \lg e[x\ln x - x]_2^{\frac{N}{2}+1} \\ &= lge[(\frac{N}{2}+1)(\ln(\frac{N}{2}+1) - \frac{N}{2} - 1) - 2(\ln 2 - 1)]. \end{aligned} \tag{28}$$

The first sum can be rewritten as,

$$\sum_{i=1}^{\frac{N}{2}-1} ilg(i+1) = \sum_{i=2}^{\frac{N}{2}} ilgi - \sum_{i=2}^{\frac{N}{2}} lgi. \tag{29}$$

Now,

$$\sum_{i=2}^{\frac{N}{2}} lgi \leq lge[(\frac{N}{2}+1)(\ln(\frac{N}{2}+1) - \frac{N}{2} - 1) - 2(\ln 2 - 1)]. \tag{30}$$

For the first sum we can write,

$$\begin{aligned}\sum_{i=2}^{\frac{N}{2}} ilgi &\leq lge \int_2^{\frac{N}{2}+1} x\ln x dx \\ &= lge[x^2(\frac{\ln x}{2} - \frac{1}{4})]_2^{\frac{N}{2}+1} \\ &= \frac{1}{4}lge[(\frac{N}{2}+1)^2(2\ln(\frac{N}{2}+1) - 1) - 4(2\ln 2 - 1)]. \end{aligned} \tag{31}$$

Combining the previous results we can state,

$$
\begin{aligned}
\sum_{i=1}^{\frac{N}{2}-1}(i+1)\lceil lg(i+1)\rceil &\leq \frac{1}{4}lge[(\frac{N}{2}+1)^2(2\ln(\frac{N}{2}+1)-1)-4(2\ln 2-1)] \\
&+ lge[(\frac{N}{2}+1)(\ln(\frac{N}{2}+1)-\frac{N}{2}-1)-2(\ln 2-1)] \\
&+ \frac{1}{8}(N^2+2N-8) \\
&= lge[\ln(\frac{N}{2}+1)(\frac{1}{2}(\frac{N}{2}+1)^2+(\frac{N}{2}+1)) \\
&- \frac{5}{4}(\frac{N}{2}+1)^2-10\ln 2+6]+\frac{1}{8}(N^2+2N-8). \qquad (32)
\end{aligned}
$$

Thus the storage requirement for the 2-layer hidden units is bounded by,

$$
\begin{aligned}
S_{NN} &\leq 2lge[\ln(\frac{N}{2}+1)(\frac{1}{2}(\frac{N}{2}+1)^2+(\frac{N}{2}+1))-\frac{5}{4}(\frac{N}{2}+1)^2-10\ln 2+6] \\
&+ \frac{1}{4}(N^2+2N-8)+lge(\frac{N}{2}+1)[\ln(\frac{N}{2}+1)+1]+2+N^2+N+lge(\ln N+1) \\
&= lge[\ln(\frac{N}{2}+1)((\frac{N}{2}+1)^2+3(\frac{N}{2}+1))-\frac{5}{2}(\frac{N}{2}+1)^2 \\
&- 20\ln 2+12+(\frac{N}{2}+1)+\ln N+1]+\frac{1}{4}(N^2+2N-8)+N^2+N+2 \\
&= lge[\ln(\frac{N}{2}+1)((\frac{N}{2}+1)^2+3(\frac{N}{2}+1))-\frac{5}{2}(\frac{N}{2}+1)^2+\frac{N}{2}+\ln N-20\ln 2+13] \\
&+ \frac{5}{4}N^2+\frac{3}{2}N. \qquad (33)
\end{aligned}
$$

Finally, we can bound the growth rate for storage by,

$$
\mathcal{O}_{Storage}(n)=\ln CN^2\ln(\frac{N}{2}+1)=4\ln C\ln^2 n\ln(\ln n+1). \qquad (34)
$$

The factor $\ln C$ is due to the number of functions that need to be stored.

q.e.d.

Table 5 displays the expected compression rates for various image dimensions n by employing Equation (24). As the results indicate the compression rates for the class $\mathcal{I}^*$ can be very high for commonly used image sizes. Note, S_{NN} is expressed in terms of bits of storage.

Table 5: Image Sizes and Expected Compression Rates

n	128	256	512	1024
S_{NN}	1221	1641	2129	2689
CR	92.5%	97.5%	99.2%	99.7%

4 Extending the Model

So far we have only looked at finding an efficient compression method for a general class of images, $\mathcal{I}^*$. Even though, - as pointed out in Theorem 1 - this class is very large, still a substantial number of images can not be encoded by any network in $\mathcal{N}^*$. One possible application of the results derived thus far is to accept this limitation and view the network construction method as a limited but extremely successful form for obtaining very high degrees of compression. The algorithm could be used as a potential pre-processor for many conventional image compression methods. In the event that it fails to derive a network structure for a given image, one can fall back on some other method.

The second alternative for dealing with this problem, is to extend the model to allow compression of any image in $\mathcal{I}^n$. We can identify 2 approaches: First, consider finding a network $N_I \in \mathcal{N}^*$ such that the maximum number of patterns in T can be correctly encoded. For the remaining patterns, - we refer to these as *exceptions* - one can apply a direct method of compression. Since these patterns are individual pixel points within the original image, any 1-dimensional compression scheme, such as Lev-Zimple [19] will suffice to compress this set of exceptions. Decompression now involves a two-step process: Lookup of a pixel value involves first searching the *table of exceptions*. If lookup fails the pattern is presented to the network for recognition. The effectiveness of this method is of course dependent on the number of exceptions that need to be stored in the table. The second approach requires that subdivisions of patterns which contain exceptions are further trained on by a network construction algorithm that automatically determines an appropriate network architecture. In recent years several approaches have been suggested, capable of deriving near minimal network structures [5, 6, 14, 15]. A disadvantage of the second approach is of course, that there would be no bound on the time requirement for constructing the network.

5 Summary

The ability to construct neural networks rather then train them has profound implications when applied to the area of lossless image compression. For one, it allows to identify worst case time bounds. By restricting the class of networks and thereby the class of images that can be represented by such a network, it is possible to achieve very high compression rates (in excess of 97%). This impressive feat is due to network parameter reduction. By limiting connection weight assignments, simplifying transfer functions, and allowing for default representations, it is possible to drastically cut storage requirements. In this paper, we introduced a class of 3-layer neural networks, $\mathcal{N}^*$, capable of representing a large class of images very effectively. It was shown that the class of images $\mathcal{I}^*$ grows faster than exponentially, and a network construction algorithm with a worst case bound of $\mathcal{O}_{Worst}(n) = n^4 - n^2$ was introduced. It is anticipated that the findings presented in this paper will spur further research into the study of applying constructed networks to the task of image compression.

References

[1] Bourlard, H. and Kamp, Y. Autoassociation by multilayer perceptrons and singular values decomposition. *Biological Cybernetics*, **59**, (1988), 291-294.

[2] Chen, W., Smith, C. Adaptive coding of monochrome and color images. *IEEE Transactions on Information Theory*, **11**, 11, (1977), 1285-1291.

[3] Clarke, R. *Transform coding of images.* Academic Press, New York, 1985.

[4] Cottrell, G., Munro, P. and Zipser, D. Image compression by back propagation: An example of extensional programming. In N.E. Sharkey (Ed). *Advances in Cognitive Science*, Vol.3, Norwood, NJ: Ablex, 1987.

[5] Fahlman, S.E. and Lebiere, C. The Cascade-Correlation Learning Architecture, In D. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2*. San Mateo, CA.: Morgan Kaufmann, 1987, pp. 524-532.

[6] Frean, M. The Upstart Algorithm: A Method for Constructing and Training FeedForward Neural Networks, *Neural Computation*, 2, 1991, pp. 198-209.

[7] Golomb, S. Run-length encodings. *IEEE Transactions on Information Theory*, 12, 6 (1966), 399-401.

[8] Green, J.L. Space data management at the NSSDC: Applications for data compression. *Proceedings Scientific Data Compression Workshop*, Snowbird, Utah, 1988.

[9] Hecht-Nielsen, R. *Neurocomputing*, Addison-Wesley Pub, 1990.

[10] Huffman, D. A method for the construction of minimum redundancy codes. Proceedings of IRE, 40, (1952), 1098-1101.

[11] Mougeot, M., Azencott, R., Angeniol, B. Image Compression With Back Propagation: Improvement of the Visual Restoration Using Different Cost Functions, *Neural Networks*, 4, (1991), 467-476.

[12] Ramamurthy, B., Gersho, A. Classified vector quantization of images. *IEEE Transactions on Communications*. 34, (1986), 1105-1115.

[13] Romaniuk, S.G., Namuduri, K.R., and Ranganathan, N. A Feature-Based Heuristic Algorithm for Lossless Image Compression. Data Compression Conference, Snowbird, Utah, USA. Also available as, Tech. Rep. TRC1/93, DISCS, National University of Singapore, January, 1993.

[14] Romaniuk, S.G., Hall, L.O. Divide and Conquer Networks. *Neural Networks*, Vol. 6, pp. 1105-1116, (1993).

[15] Romaniuk, S.G. Evolutionary Growth Perceptrons. In S. Forrest (Ed.) *Genetic Algorithms: Proceedings of the 5th International Conference*, Morgan Kaufman, 1993. (1993).

[16] Rumelhart, D.E., Hinton, G.E., Williams, R.J. *Learning Internal Representations by Error Propagation.* In D.E. Rumelhart and J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition (pp. 318-362). Cambridge, MA: MIT Press, 1986.

[17] Vaisey, J. and Gersho, A. Image compression with variable block size segmentation. *IEEE Transactions on Signal Processing*, **40**, 8 (1992), 2040-2060.

[18] Wallace, G.K. The JPEG still picture compression standard, *CACM*, **34**, 4 (1991), 30-44.

[19] Ziv, J., Lemple, A. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory.* **23**, 5 (1977), 337-343.