Feedforward Neural Network Design with Application to Image Subsampling

by

Adriana Dumitraş

M.A.Sc., "POLITEHNICA" University of Bucharest, Romania, 1988

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES

(Department of Electrical and Computer Engineering)
We accept this thesis as conforming
to the required standard

The University of British Columbia

October 1999

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

AFTER THE RESTRICTION TIME (OCT 18, 1999 - OCT 18, 2000) PASSED.

(Signature)

Department of ELECTRICAL & COMPUTER ENG.

The University of British Columbia
Vancouver, Canada

Date OCT. 21, 1999

Abstract

Feedforward artificial neural networks (FANNs), which have been successfully applied to various image processing tasks, are particularly suitable for image subsampling due to their high processing speed. However, the performance of FANNs in image subsampling, which depends on both the FANN topology and the FANN training algorithm, has not been acceptable so far. High performance image subsampling is important in many systems, such as subband decomposition systems, and scalable image and video processing systems.

This thesis addresses the design of FANNs with application to image subsampling. More specifically, we focus on both the topological design of FANNs and the training algorithm, so that efficient FANN structures, yielding good performance in image subsampling, are obtained. That is, we aim at obtaining compact FANNs that yield good subsampled versions of the original images, such that if reconstructed, they are as close as possible to the original images. Moreover, we aim at obtaining better performance–speed tradeoffs than those of the traditional lowpass filtering and subsampling methods.

First, we propose a design method for FANNs, which leads to compact tridiagonally symmetrical feedforward neural networks (TS–FANNs). Next, in order to

address the problem of artifacts that generally appear in the reconstructed images after FANN–based subsampling, we propose a training method for FANNs. When applied to first–order (FOS) and multi–stage first–order (MFOS) image subsampling, the FANNs trained using our method outperform the traditional lowpass filtering and subsampling (LPFS) method, without requiring pre– or post–processing stages. Motivated by our observation that the computational demands of the MFOS process increase approximately linearly with the image size, we then combine the proposed methods and evaluate the performance–complexity tradeoffs of the resulting TS–FANNs in FOS and MFOS. We show that our TS–FANNs–based subsampling has important advantages over subsampling methods based on fully connected FANNs (FC–FANNs) and LPFS, such as significantly reduced computational demands, and the same, or better, quality of the resulting images.

The main contributions of this thesis consist of a method for FANN design with tridiagonal symmetry constraints, a training algorithm for FANNs applied to image subsampling, the design and evaluation of the performance–speed tradeoffs of FC–FANNs in image subsampling, and the design and evaluation of the performance–speed tradeoffs of TS–FANNs in image subsampling. The FANN performance in image subsampling is evaluated objectively (using the peak signal–to–noise ratios), subjectively (by visual examination of the subsampled and of the reconstructed images), and in the context of a video coding application. The speed and memory demands of the designed FANN structures are evaluated in terms of the subsampling time and the number of FANN parameters, respectively.

Contents

List of Tables

List of Figures

List of Acronyms

ANN	Artificial Neural Network
FABS	FANN–Based Subsampling algorithm
FANN	Feedforward Artificial Neural Network
FANNS	FANN–based image subsampling
FC–FANN	Fully connected FANN
FANNS+int.	FANN–based image subsampling followed by interpolation
FOS	First–order subsampling
GFABS	Generalized FANN–Based Subsampling algorithm
HOS	High–order subsampling
int.	Interpolation (cubic interpolation, if not specified otherwise)
LPF	Lowpass filtering
LPF1, LPF2, LPF3	Lowpass filtering using three different filters
LPFS	Lowpass filtering followed by subsampling
LPFS+int.	Lowpass filtering, followed by subsampling and interpolation
LPF1S, LPF2S, LPF3S	Lowpass filtering using LPF1, LPF2, LPF3, respectively, followed by subsampling

MEDS	Median filtering–based image subsampling
MFOS	Multi–stage first–order subsampling
med	Median operator
OBD	Optimal Brain Damage algorithm
PSNR	Peak signal–to–noise ratio
TOBD	Tridiagonal Optimal Brain Damage algorithm
TS–FANN	Tridiagonally symmetrical FANN

List of Symbols

$C(\mathbf{w})$	Cost function
f	The node's activation function
$\mathbf{D}$	Subsampling matrix
η_k	Learning rate at step k
H	Number of FANN hidden nodes
$\mathbf{H}$	Hessian matrix
Π	Total number of FANN parameters
M	Number of FANN input nodes
N	Number of FANN output nodes
$\mathbf{J}$	Jacobian matrix
P	Total number of training patterns
s_q	Saliency
$\mathbf{w} = [w_1, w_2, \ldots, w_M]^T$	Weight vector
$\mathbf{W}$	Weight matrix
$\widetilde{\mathbf{W}}$	The weight matrix $\mathbf{W}$ padded with zeros until it becomes a square matrix

$\mathbf{x} = [x_1, x_2, \ldots, x_M]^T$	FANN input vector
$x_c(t)$	Continuous time signal
$x[n]$	Discrete sequence of samples
ξ	Index of the FANN training patterns
$\mathbf{y} = [y_1, y_2, \ldots, y_N]^T$	FANN output vector

To my parents with all my love.

Acknowledgements

I have understood for a long time that we are, essentially, solitary creatures and that most of our learning experiences are lonely journeys. And yet, during the learning journey leading to the present thesis, I was fortunate to meet very special people, who have helped me shape the way and touched my heart.

First and foremost, I would like to thank my supervisor Professor Faouzi Kossentini for his generous advice and professional help during the past three years. I would like to thank him for enthusiastically exposing me to different, challenging and professionally rewarding experiences, such as the research projects in collaboration with the National Sciences and Engineering Research Council of Canada (NSERC) and the National Research Council of Canada (NRC), and the projects related to our involvement in the MPEG–7 standardization activity. I am grateful for his professional and financial support that allowed me to participate and present our work at IEEE conferences and standardization meetings all over the world, as well as for the financial support provided during all my Ph.D. research activity. Finally, I am grateful for his advice not only as my supervisor, but also as a friend.

I would like to thank the members of my departmental thesis committee, Professors Peter Lawrence, Ian Cumming and Rabab Ward, the members of my univer-

sity thesis committee, Professors Takahide Niimura, Jose Marti, Michael Davies and James Little, as well as the external examiner, Professor Anastasios Venetsanopoulos from University of Toronto, for their useful comments and suggestions on the thesis.

I would like to thank Associate Professor Yu Hen Hu from the University of Wisconsin–Madison, Assistant Professor Brian Evans from the University of Texas at Austin and Associate Professor Hussein Alnuweiri from the University of British Columbia for their useful comments related to Chapter 3 of the thesis.

I am grateful for receiving continuous and friendly encouragements over many years from Associate Professor Yu Hen Hu from the University of Wisconsin–Madison, Professor Vasile Lăzărescu and Associate Professor Nicolae Tomescu from the "Politehnica" University of Bucharest, Romania.

I thank my long time friends Eugenia Rădulescu, Cristian Munteanu and Jasmina Sabolović for deleting the thousands of miles between us with their warm emails and letters. I would like to thank my friend Shahram Shirani for reading the first chapters of the thesis, for our challenging discussions, for sharing his wisdom and good heart. I would like to thank all of my colleagues in the Signal Processing and Multimedia Group at the University of British Columbia, especially Guy Côté and Alen Docef, for their friendship.

Last but not least, I would like to thank my parents for their unconditional love, help and understanding. I owe you all that is good in me.

ADRIANA DUMITRAŞ

The University of British Columbia

October 1999

Chapter 1

Introduction

Effective and efficient processing of large amounts of digital data has become more important in recent years due to an increasing number of multimedia applications. Parallel and distributed processing models, such as the artificial neural network (ANN) models, have been proposed in order to address the performance and speed requirements of such applications. The initial interest for the ANN models was mainly due to the belief that these models can mimic the basic functionalities of the human brain. However, the existing artificial neural models are only simplified models of the biological neural structures, with functionalities which are still far from those of the human cognitive structures [1, 2]. Despite these limitations, the ANN models in general, and the feedforward artificial neural network (FANN) models in particular, have been successfully applied over the last decades to various digital signal processing tasks [3]. In particular, FANNs can be applied to image subsampling.

The importance of image subsampling is manifold. First, it provides efficient representation of images, by simple lossy compression [4]. The resulting low resolution

images may be processed using less computations and memory, while certain areas of interest in the image may be retrieved and processed later at higher resolutions [5, 6], [7]–[9]. Second, subband decomposition systems [10]–[12], as well as systems that build image pyramids [13]–[15], involve subsampling. The former approach uses analysis filter banks to produce the subbands, each of which is further filtered and downsampled as many times as desired. The latter builds a representation using a set of lower resolution copies of the image, obtained by iterative filtering with a generating kernel and decimation [15]. Third, several applications in digital television require sampling structure conversions of the video signal [16, 17]. Such sampling conversion usually involves upsampling, lowpass filtering and then downsampling of the signal. Fourth, scalable image and video processing systems address the various bandwidth constraints by providing several spatial and temporal resolutions of the images/video. These are obtained by successively downsampling and upsampling the image/video and by encoding the resulting pictures. The access to the lower resolution images does not require the decoding of the higher resolution images [5, 8, 9, 18]. Finally, hierarchical search methods [8], which are particularly popular in motion estimation and compensation, involve subsampling of the present video frame and the previous (reference) video frame successively in the spatial domain. The search process starts with the lowest resolution frame, and the motion vector estimated at such resolution level is used as the starting point for motion vector estimation at the next resolution level. Clearly, the accuracy of subsampling, in this case, has a strong impact on motion vector estimation.

In most of these image subsampling applications, the subsampled images are

obtained by lowpass filtering and downsampling. However, when lowpass filtering is being applied to the input image, most of the high frequency information is permanently lost. Moreover, due to most of the existing subsampling methods being based on pixel neighborhood operations [19], the images reconstructed using the subsampled versions may often contain significant distortion, usually expressed in terms of visible blockiness in continuous features of the image [19, 6]. Of course, several good post–processing techniques for eliminating blocking artifacts have been proposed [20], but the associated processing cost is often quite high. Last but not least, the number of computations required by the lowpass filtering stage is often too high for applications where high processing speed is demanded.

A solution to simultaneously reduce both information loss and blockiness, and increase processing speed, is to apply FANN models to image subsampling. The FANNs are especially suitable for image subsampling due to the following reasons: (a) they inherently subsample the input images, for given dimensions of the neural structure, and (b) they can perform high speed parallel processing. The efficiency and performance of FANN models in image subsampling depend on their sizes, connectivities, and associated training algorithms. The aim of most FANN topological design methods is the optimization of the size and connectivity of the neural structure. Many FANN design algorithms, that are based on empirical, statistical, growing, pruning or hybrid methods, have been developed. The outcome of the growing algorithms is generally a large, fully connected and symmetrical structure. The result of the pruning algorithms is generally a simple, partially connected, and non–symmetrical structure. For efficient processing, partially connected FANNs are desirable. For

hardware and software implementations, in applications which are very demanding in terms of computations (e.g., high–order subsampling of images having large sizes), symmetrical neural structures are especially desired.

The effectiveness of any FANN structure designed using topological methods depends significantly on the selected FANN training algorithm. Several supervised, semi–supervised and unsupervised training algorithms have been proposed, each of which employs a training data set in order to determine the optimal FANN parameters. Supervised training algorithms also require a set of desired output values, which is used as a reference during FANN training. In image processing applications, the FANN desired outputs generally consist of the gray–level values corresponding to pixels having fixed positions within each of the local processing windows. These pixel values and their corresponding positions are usually selected prior to the training process. Unfortunately, the pixel gray–level values do not provide any geometrical information to the FANN during training. Moreover, the fixed positions of the desired output pixel values do not allow the FANN to extract the geometrical information during training. Therefore, the image subsampling performance of FANNs trained using such standard methods has not been acceptable so far, as the images reconstructed after FANN subsampling often exhibit blocking and/or ringing artifacts. In order to improve the quality of the FANN subsampled and reconstructed images, the local geometrical information is required during FANN training. For high efficiency, adaptive methods which obtain the local geometrical information during FANN training are desired.

1.1 Thesis objective

The main objective of this thesis is the design of efficient FANNs with high image subsampling performance. More specifically, our goal is to obtain compact FANN structures that yield a good subsampled version of the original image, such that if reconstructed, it is as close as possible to the original. To achieve our objective, we focus on the FANN topological design and the FANN training algorithm. We apply our designed FANNs to first–order (FOS) and multi–stage first–order (MFOS) image subsampling, showing that they achieve better performance–speed tradeoffs than the traditional lowpass filtering and subsampling methods.

In the first part of the thesis, we propose an algorithm for the design of FANN structures with tridiagonal symmetry constraints. The algorithm employs a Householder transformation of the FANN weight matrix. In the second part of the thesis, we propose a training algorithm for FANNs. Our method is based on a pattern matching approach to select the FANN desired output values during the supervised training stage. In the third part of the thesis, we combine the proposed methods in order to design tridiagonally symmetrical FANNs (TS–FANNs) which are fast and effective when applied to FOS and MFOS.

The performance of the FANNs in image subsampling is evaluated objectively, subjectively and through a video coding application. Our objective performance evaluation of the interpolated images is based on the peak signal–to–noise ratio values. Our subjective performance evaluation is based on the visual examination of the subsampled images [21] and on the visual examination of the reconstructed (bilinear or cubic interpolated) images. We also evaluate the performance of the FANNs in

chrominance subsampling within a video coding application. The efficiency of our designed FANNs is evaluated using the subsampling time and the number of FANN parameters.

1.2 Thesis structure

In Chapter 2, we review the main concepts that will serve as background material throughout the thesis. In particular, we discuss the characteristics of the FANNs, the most popular design methods for FANN topology, as well as review basic image subsampling concepts. Chapter 3 introduces a new algorithm for FANN design with tridiagonal symmetry constraints. In the same chapter, we evaluate the performance and complexity of this algorithm using one–dimensional signals. In Chapter 4, we introduce a new training algorithm for subsampling using FANNs, and we then apply our trained FANN structures to FOS. We also evaluate the performance of our trained FANNs in chrominance subsampling within a low bit rate video coding system. In Chapter 5, we employ our design algorithm in order to reduce the connectivity of the FANN structure. We also modify our training algorithm by including an adaptive threshold, and we then show that this leads to better image quality when applied to MFOS. In the same chapter, we evaluate the performance and complexity of the resulting TS−FANNs in MFOS. Finally, in Chapter 6, we highlight the contributions of the thesis and suggest future research directions.

Chapter 2

Background

In this chapter, the fundamental concepts related to feedforward artificial neural networks (FANNs), the existing FANN design methods, and the main concepts related to image subsampling are reviewed. In Section 2.1, the characteristics of feedforward artificial neural networks are presented. In Section 2.2, FANN learning, generalization, and optimal selection are addressed. A review of the existing methods for FANN design is included in Section 2.3. We discuss image subsampling in Section 2.4. A summary of the chapter is included in Section 2.5.

2.1 Characteristics of Feedforward Artificial Neural Network Models

Due to the various sources of inspiration (biological, physiological, psychological) and the independent development of the artificial neural network (ANN) models in various research communities, such as neurobiological, mathematical, computer science,

etc., a synonymic ANN terminology[1] and various ANN definitions exist [1]–[26]. In this thesis, we will employ the terms "artificial neural networks" and "neural networks", both referring to the artificial neural network models. We will also employ the ANN definition proposed in [2]: "Artificial neural models are parallel and distributed processing structures, consisting of processing elements, interconnected by signal channels, known as connections. Each processing element has a single output connection. Processing is performed locally by each processing element".

A feedforward neural network model is defined by the characteristics of the processing node, the network topology, the data model, the cost function, and the training algorithm. These are discussed next.

2.1.1 The Processing Node

A general model of a simple perceptron (processing node) is illustrated in Figure 2.1. This model has basic characteristics that are similar to those of the biological neuron [27]. More specifically, it models the intensity of the biological synapses via the "weight" values and it provides the output signal value $y(t)$ by performing an arithmetic operation on the input signals. If this arithmetic operation consists of a summation of the input signals, followed by a comparison of the result with a selected threshold, then the McCulloch–Pitts model of the simple perceptron, which is illustrated in Figure 2.2, is obtained. The notations $\mathbf{x} = [x_1, x_2, \ldots, x_M]^T$, $\mathbf{w} = [w_1, w_2, \ldots, w_M]^T$, f, and y stand for the input vector, the weight vector, the node's activation function and the output signal, respectively. The output value y

[1]For instance, connectionist models, parallel and distributed processing models, neuromorphic models, are all equivalent names for artificial neural networks [22].

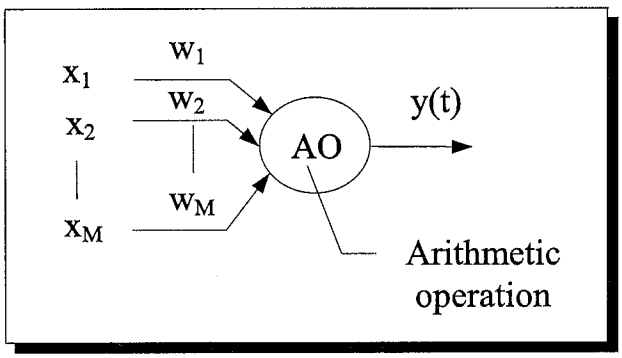


Figure 2.1: A general model of a simple perceptron (processing node).

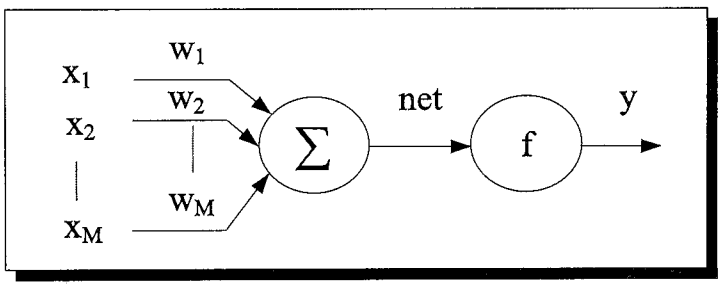


Figure 2.2: The McCulloch–Pitts simple perceptron.

of the McCulloch–Pitts model is given by $y = f\left(\sum_{i=1}^{M} w_i x_i\right)$. We note that, if the simple perceptron presented above is a component having the position j in a multi–node structure, then its output value is given by $y_j = f_j\left(\sum_{i=1}^{M} w_{ij} x_i\right)$.

To summarize, in the McCulloch – Pitts simple perceptron with the position j, the weighted sum of the input values is first computed and passed as an argument to the activation function f_j. Next, the value of the activation function is evaluated. The activation function is the Heaviside step function, having a value equal to 1 for an argument greater than b_j and 0 otherwise, where b_j is a threshold [27]. The output

value of the McCulloch – Pitts node after the comparison with the threshold has been performed is given by $y_j = f_j\left(\sum_{i=0}^{M} w_{ij}x_i\right)$.

Other perceptron models can be obtained from the general model illustrated in Figure 2.1, simply by choosing various activation functions. This is discussed in the next section. We also note that other models of the simple perceptron can be obtained by including local feedback in the structure [2, 28].

2.1.2 The Activation Function

As mentioned above, the activation function of the simple McCulloch – Pitts perceptron is the step function[2]. Therefore, the output value of the simple perceptron can only be equal to $0/+1$ (for a binary step function) or $-1/+1$ (for a bipolar step function). Other activation functions have also been proposed. Generally, an activation function can be linear or nonlinear, and monotonic or non–monotonic. An activation function can have a step, ramp, triangular, trapezoidal, sigmoidal, polynomial shape, or a shape of a radial basis function such as Gaussian, Mexican hat, spline, sigma–pi, etc. Finally, the output value of an activation function can be continuous or discrete, and binary or bipolar or other. Examples of activation functions often employed in practical applications are the unipolar and bipolar sigmoidal functions defined by

$$f(net) = \frac{1}{1+\exp(-s\,net)}, \text{ and} \tag{2.1}$$

[2]We note that, due to the selection of the step activation function, the McCulloch – Pitts node is also known as the "conventional" or "Socratic" node. As stated in [29], "World is invaded by Socratic thinking, which is based on dichotomies and polarities (plus/minus, true/false, all/none)".

$$f(net) = \frac{1 - \exp(-s\,net)}{1 + \exp(-s\,net)}, \quad \text{or}$$

$$f(net) = \tanh(s\,net) = \frac{1 - \exp(-2\,s\,net)}{1 + \exp(-2\,s\,net)}, \tag{2.2}$$

where s denotes the slope of the activation function f, and net is the weighted sum of the input values, given by $net = \mathbf{w}^T\mathbf{x} = \sum_{i=0}^{M} w_i x_i$ [27, 30].

2.1.3 The Topology

From the structural point of view, FANN micro–structures, mezo–structures and macro–structures can be defined. A micro–structure consists of a simple neural processing node. A mezo–structure consists of interconnections of simple nodes according to a selected topology. A macro–structure consists of interconnections of mezo–structures according to a selected topology [31]. The network topology is defined by its geometry and interconnection scheme [32].

A general classification of ANN mezo–structures is illustrated in Figure 2.3. The feedforward neural mezo–structure is illustrated in Figure 2.4, where the notations $\mathbf{x} = [x_1, x_2, \ldots, x_M]^T$ and $\mathbf{y} = [y_1, y_2, \ldots, y_N]^T$ stand for the input and output vectors (respectively), $\mathbf{w}_j$ is the weight vector between the input nodes and the output node j, w_{ij} is the weight of the connection between the nodes i and j, $\mathbf{W}$ is the weight matrix having as columns the vectors $\mathbf{w}_j$, and F is a nonlinear function. The output value of the neural structure is given by $\mathbf{y}(k) = F\left[\mathbf{W}^T\mathbf{x}(k)\right]$, where k is the discrete[3]

[3]We here assume a discrete system. For a continuous system, this relationship becomes $\mathbf{y}(t) =$

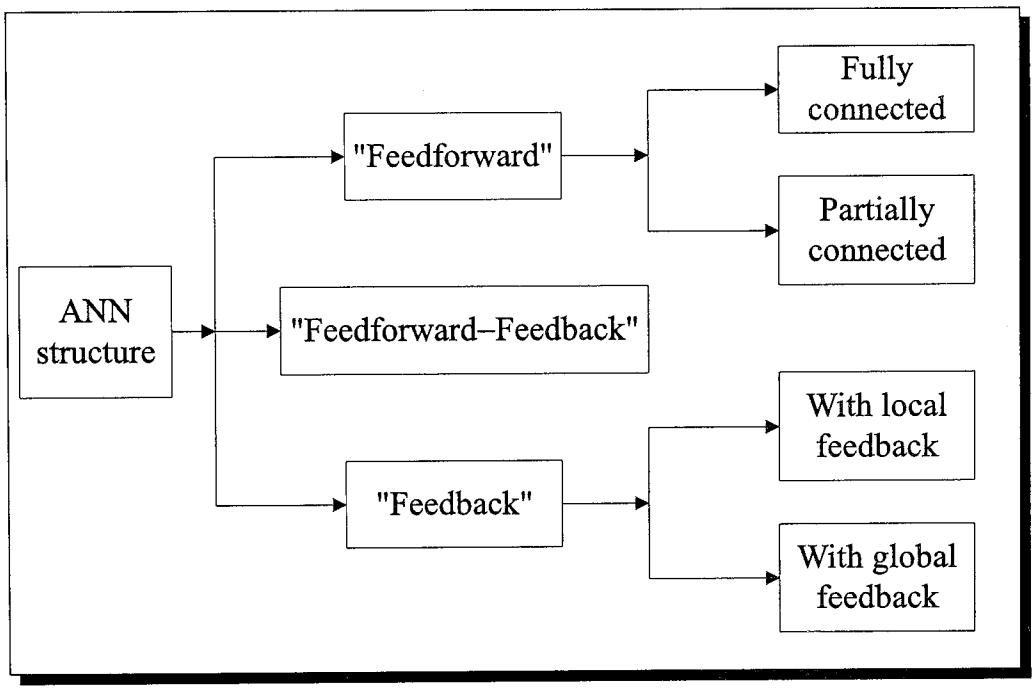


Figure 2.3: A classification of ANN mezo–structures.

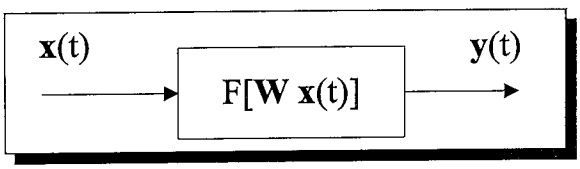


Figure 2.4: Block diagram of a feedforward neural mezo–structure.

time index [25]. By adding a feedback connection between the output and the input of the mezo–structure illustrated in Figure 2.4, the feedback neural mezo–structure is obtained. The characteristics of feedback mezo–structures and related issues are discussed in [1, 2, 25].

As mentioned earlier, a neural FANN macro–structure is the interconnection of several mezo–structures. The topology of a macro–structure and related issues are discussed in [1, 2].

2.1.4 The Data Model and the Cost Function

The data model and the cost function describe the application that is being solved using the FANN model. More specifically, the cost function represents a hypothesis on the distribution of the input data. The cost function for FANNs is usually denoted by $C(\mathbf{w})$, and it is defined on the parameter (weight) space. The goal is to determine the weight vector $\mathbf{w} \in \Re^{\Pi}$ for which $C(\mathbf{w})$ is minimized, that is,

$$\text{Minimize the scalar cost function } C(\mathbf{w}) \text{ subject to } \mathbf{w} \in \Re^{\Pi}, \tag{2.3}$$

where the Π–dimensional weight vector may also include other network parameters, such as the thresholds b_j that have already been mentioned in Section 2.1.1. The cost function must satisfy the following theorem [24]:

Theorem 1 *Let $\nabla^2 C(\mathbf{w})$ be a Hessian matrix, nonsingular in $\mathbf{w}^*$. If the Jacobian matrix $\nabla C(\mathbf{w}^*) = 0$ and if the Hessian matrix $\nabla^2 C(\mathbf{w}^*)$ is symmetrical and positively*

$F[\mathbf{W}^T \mathbf{x}(t)]$.

defined, then $C(\mathbf{w}^*) < C(\mathbf{w})$ *for each* $\mathbf{w}$ *which satisfies the condition* $0 < \|\mathbf{w} - \mathbf{w}^*\| < \epsilon$, $\epsilon > 0$.

2.1.5 The Training Algorithm

In biological systems, learning is generally defined as a change in behavior [25, 31]. In artificial systems, learning is generally defined as a set of parameter modifications due to an adaptation process [25]. The goal of these changes performed during learning is the minimization of a cost function. The result of the learning process is a set of parameter values.

The effectiveness of FANN learning depends on the careful selection of the simple processing nodes (discussed earlier) and on the training algorithms. These algorithms usually update the weights locally, at the simple node level, according to a training rule. The training rule may be supervised, semi–supervised or unsupervised. Supervised training rules, also known as "learning with a teacher", are illustrated in Figure 2.5 (a). Supervised training rules employ a training data set $\{\mathbf{x}(\xi), \mathbf{d}(\xi)\}$, where $\{\mathbf{x}(\xi)\}$ are the input values, $\{\mathbf{d}(\xi)\}$ are the corresponding desired output values, $1 \leq \xi \leq P$ is the index and P is the number of the training patterns, respectively. The aim is to minimize the distance between the actual output values $\mathbf{y}(\xi)$ and the desired output values $\mathbf{d}(\xi)$, using the selected cost criterion [33]. Semi–supervised training rules, also known as "learning with a critic", are similar to the supervised training, except that the desired output values $\mathbf{d}(\xi)$ are not defined. Instead, the network receives a mark which quantifies how well it has learned the previous training patterns [22]. In other words, the learning process in the FANN is reinforced or pe-

nalized. In Figure 2.5 (b), r denotes the reinforcement/penalty signal. Unsupervised training rules, illustrated in Figure 2.5 (c), update the network parameters using only the input data. No desired output values are provided in this case.

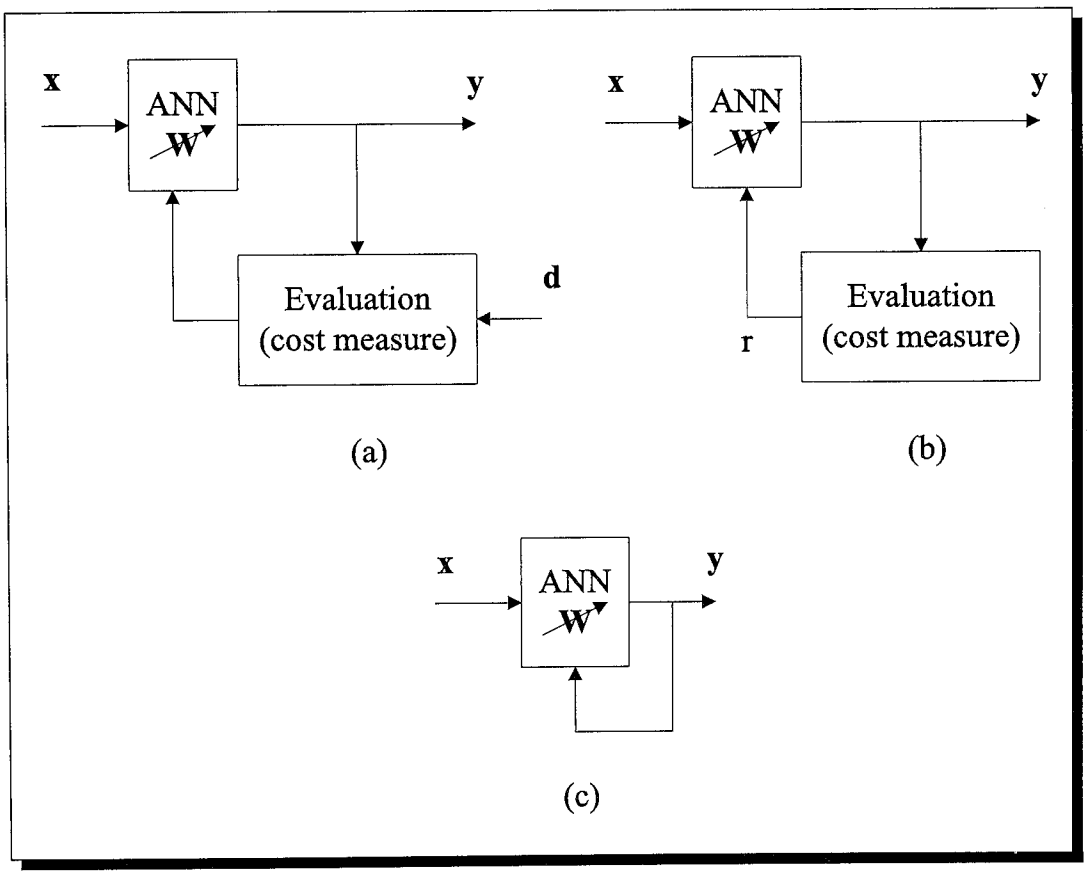


Figure 2.5: (a) Supervised, (b) semi–supervised, and (c) unsupervised learning.

2.2 FANN Learning, Generalization, Evaluation

In this section, we first discuss the FANN learning as an approximation / optimization process. Next, we address the FANN generalization ability. Finally, we comment on the optimal selection of a FANN structure. We will restrict our discussion to the supervised learning in a multilayer perceptron FANN structure that is illustrated in Figure 2.6.

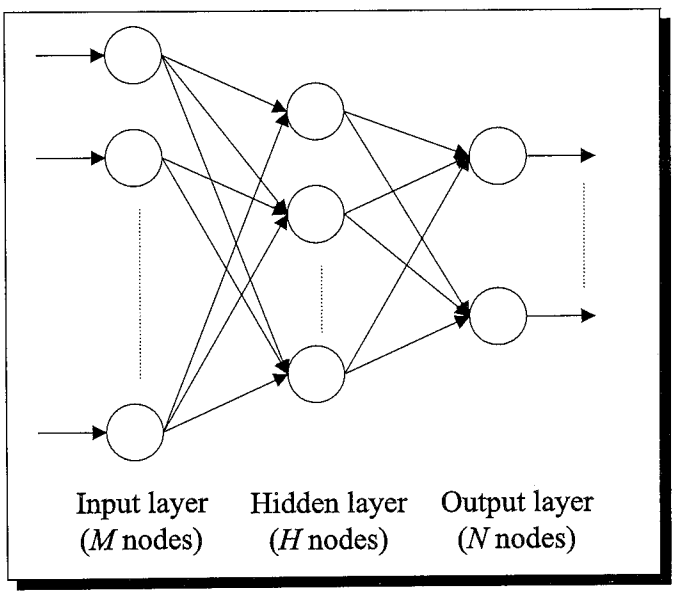


Figure 2.6: An M–H–N multilayer perceptron FANN.

2.2.1 FANN Learning and Generalization

FANN Learning as an Approximation Process

Supervised FANN learning of a data set $\{\mathbf{x}(\xi), \mathbf{d}(\xi)\}$, with $\mathbf{d}(\xi) = F(\mathbf{x}(\xi))$ and $1 \leq \xi \leq P$, can be viewed as the approximation of the function $F(\mathbf{x})$ by the function

$F(\mathbf{w};\mathbf{x})$ implemented by the FANN model. The goal is to determine the set of parameters $\mathbf{w}^*$ that leads to the best approximation $F(\mathbf{w}^*;\mathbf{x})$ of the function $F(\mathbf{x})$, given the input data set.

Important issues need to be addressed here, such as the classes of functions that can be effectively approximated by $F(\mathbf{w};\mathbf{x})$ and FANN overtraining. The former is a representation problem. From this perspective, FANNs are nonlinear models that are able to perform "universal approximation", that is, they can approximate any continuous input–output relationship. This powerful property, which also motivates our selection of FANNs as neural models in this work, is based on Kolmogorov's [34] theorem of existence. This theorem states that two hidden layers are sufficient for designing a FANN universal approximator. However, the theorem does not provide the details that are necessary to build the neural model.

FANN overtraining is a phenomenon which consists of a decrease of the training[4] error simultaneously with an increase of the testing error. In other words, the FANN approximates well the relationships in the training data set, but generalizes poorly when using the testing data set. A small size of the training data set and a large number of FANN parameters as compared to the number of training patterns, are some of the reasons that can lead to overtraining. Hence, the estimation of the testing error during training (so that FANN training is terminated when the value of the testing error increases) and the limitation of the number of FANN parameters are possible solutions to avoid overtraining.

[4]The training/testing errors are averages of the cost function values computed for all of the training/testing patterns, respectively.

FANN Learning as an Optimization Process

When viewed as an optimization process, the goal of FANN training is to obtain the set $\mathbf{w}^*$ of optimal parameters which solves the minimization problem (2.3) stated in Section 2.1.4. The weights $\mathbf{w}^*$ are determined by searching in the space of all possible network configurations. If the cost function $C(\mathbf{w})$ is additive, then the minimization problem (2.3) becomes

$$\text{Minimize the cost function } C(\mathbf{w}) = \sum_{\xi=1}^{P} \sum_{j=1}^{N} C_j(\mathbf{w}; \mathbf{x}(\xi)) = \sum_{\xi=1}^{P} C(\mathbf{w}; \mathbf{x}(\xi))$$

$$\text{subject to } \mathbf{w} \in \Re^{\Pi},$$

where $1 \leq \xi \leq P$ are the FANN training patterns. The cost function $C(\mathbf{w})$ must satisfy Theorem 1 in Section 2.1.4. Moreover, the cost function $C(\mathbf{w}; \mathbf{x})$ must be convex, that is $C[(1-\gamma)a + \gamma b] \geq (1-\gamma)\, C(a) + \gamma\, C(b)$, with $\gamma \in [0, 1]$ [24].

Supervised FANN Learning Algorithms

In most algorithms that are used to solve the optimization problem stated above, the cost function and its derivatives are evaluated. Next, a minimum of the cost function is obtained and the search in the weight space is further refined around this minimum. Let us assume that the search for the minimum of the cost function $C(\mathbf{w})$ in the parameter space is linear, with $[w_1, w_2, \ldots, w_{\Pi}]^T$ the vector of FANN parameters, and that it is performed in discrete time. Moreover, let us assume that the search starts from the initial point $\mathbf{w}_0$ in the direction[5] $\vec{\mathbf{d}}$, and that the search

[5]The notation for the direction must not be confused with that used for the vector of the desired output values $\mathbf{d}$ in the FANN.

follows the straight lines defined at each step k by $\mathbf{w}_{k+1} = \mathbf{w}_k + \eta_k \vec{\mathbf{d}}_k$, where $\mathbf{w}_k$, $\mathbf{w}_{k+1}$ and η_k are the weight vectors at the steps k and $k+1$, and the learning rate at step k, respectively. The search direction $\vec{\mathbf{d}}_\mathbf{k}$ can be determined by applying deterministic (e.g., based on the computation of the first or second order derivatives of the cost function) or stochastic methods (e.g., "simulated annealing").

First order deterministic methods compute the search direction based on the value of the local gradient. Therefore, these methods are also called gradient descent or steepest gradient methods. A well–known example is that of the backpropagation algorithm [1]. Second order deterministic methods compute the search direction based on the second order derivatives of the cost function $C(\mathbf{w})$. The conjugate gradient method, Newton's method, quasi– and pseudo–Newton methods are examples of second order derivative based search methods. In particular, all versions of Newton's method are based on the idea that the cost function can be approximated locally by a quadratic function. Moreover, this quadratic function can be minimized exactly. The rule for updating the FANN parameters in the Newton method is given by $\mathbf{w}_{k+1} = \mathbf{w}_k - \mathbf{H}^{-1}(\mathbf{w}_k)\,\mathbf{J}(\mathbf{w}_k)$, where $\mathbf{J}$ and $\mathbf{H}$ denote the Jacobian and Hessian matrices of the cost function, respectively. The rule for updating the FANN parameters in quasi–Newton methods is given by $\mathbf{w}_{k+1} = \mathbf{w}_k - \eta_k \hat{\mathbf{H}}_k^{-1} \nabla C(\mathbf{w}_k)$, where $\hat{\mathbf{H}}^{-1}$ is an approximation[6] of the inverse Hessian matrix $\mathbf{H}^{-1}$ (computed recursively) and η_k is the learning rate at each step k, given by [24, 35]

$$\eta_k = \arg\min_{\eta \geq 0} C\left(\mathbf{w}_k - \eta \hat{\mathbf{H}}_k^{-1} \nabla C(\mathbf{w}_k)\right). \tag{2.4}$$

[6]Quasi–Newton methods employ approximations of the inverse Hessian matrices, thus attempting to address the problem of high cost of Newton method. More specifically, the Netwon method converges theoretically in Π^3 steps, with Π the number of FANN parameters. At each step k, the inverse matrices $\mathbf{H}$ of size $\Pi \times \Pi$ must be computed [24].

Finally, pseudo–Newton methods are equivalent to applying the Newton rule separately, for each weight. In pseudo–Newton methods, the elements outside the main diagonal of the Hessian matrix $\mathbf{H}$ are ignored.

FANN Generalization

During the testing stage, the FANN generalization ability is evaluated by computing the error on the testing data set. Generalization can be defined as "the ability to estimate quantitatively the characteristics of a phenomenon that was not met previously, based on its similarities with other known phenomena" [22, 36, 37]. FANN generalization can be improved by avoiding the overtraining process that has been described in Section 2.2.1. As stated earlier, one solution is the selection of a small number of FANN parameters. Assuming that the FANN's input and output layers have fixed sizes, the selection of the number of FANN parameters becomes the selection of the number H of hidden nodes so that overtraining is avoided. This is illustrated in Figure 2.7.

A solution to reduce the number of FANN parameters is regularization, which consists of imposing constraints on the input–output function implemented by the FANN. Assume, for instance, that the cost function is given by $C(\mathbf{w}) = \lambda C_{training}(\mathbf{w}) + \gamma C_{complexity}(\mathbf{w})$, where $C_{training}$, $C_{complexity}$, $\mathbf{w}$, $\mathbf{x}$, λ, γ are the standard error term, the regularization term, the FANN parameters, the input vector, and the regularization parameters, respectively [26, 34]. If the first component of the cost function is, for instance, the L_2 norm of the error, and if the numbers of input and output values are equal to $M > 1$ and $N = 1$ (respectively), the training error may be computed by

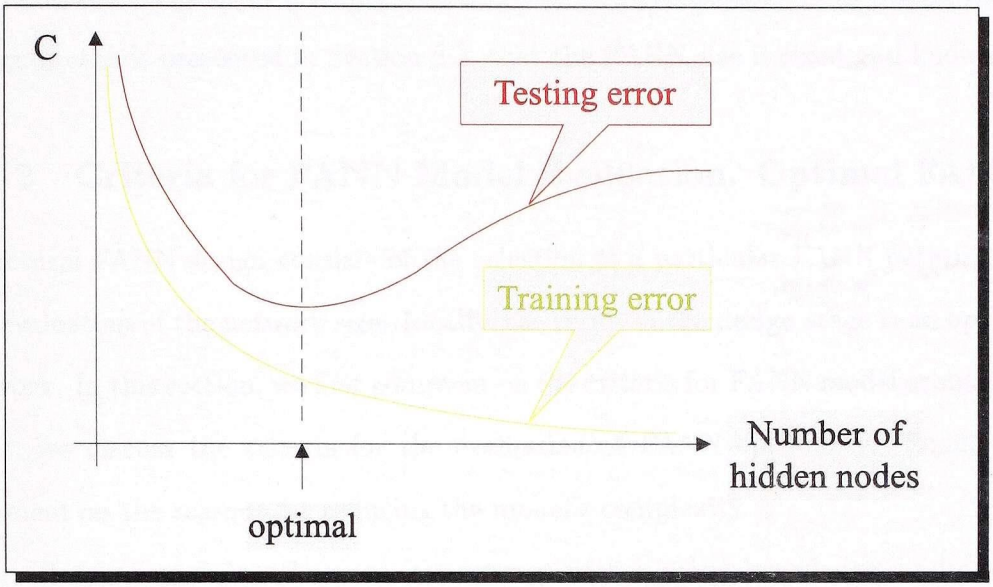


Figure 2.7: Training error and testing error as functions of the number of hidden nodes

$$C_{training}(\mathbf{w}) = \frac{1}{2}\sum_{\xi=1}^{P}[d(\xi) - y(\xi)]^2 = \frac{1}{2}\sum_{\xi=1}^{P}[d(\xi) - F(\mathbf{w};\mathbf{x}(\xi))]^2, \qquad (2.5)$$

Various forms of the regularization (penalty) term $C_{complexity}(\mathbf{w})$ are discussed in [38]. If the penalty term causes some of the FANN weight values to decrease, the FANN structure is reduced. This is known as a weight decay regularization [39]. In addition to regularization, other solutions for the selection of the number of FANN parameters are provided by FANN design methods, which are discussed in Section 2.3.

Finally, the selection of the number of FANN parameters is directly related to the selection of a sufficient number of training patterns. In other words, sufficient data must be available in order to impose constraints on the FANN parameters during the training stage. The relationship between the number of FANN parameters and

the necessary number of training patterns can be determined by assuming, in the design methods presented in Section 2.3, that the FANN size is fixed and known.

2.2.2 Criteria for FANN Model Evaluation. Optimal FANNs

Structural FANN design consists of the selection of a particular FANN network and the evaluation of the network size. Ideally, the result of the design stage is an optimal network. In this section, we first comment on the criteria for FANN model evaluation. Next, we discuss the criteria for the evaluation of FANN optimality. Finally, we comment on the reasons for reducing the model's complexity.

Parsimony, data coherence, consistency with a priori knowledge, and dimensionality, which are criteria for general model evaluation [40], can also be applied for FANN model evaluation. The "parsimony principle", also known as "Ockham's razor principle", has been formulated by the medieval philosopher William of Ockham (1300–1349), and it states that "there must not exist more entities than necessary". This idea expresses a fundamental principle of modern science, which is the necessity to simplify the scientific theories. In particular, this idea applies to FANNs as follows. Given two FANNs and a common training set, the network having the smallest number of parameters generalizes better [41]. The data coherence criterion requires that the neural model be able to learn the given data set according to a selected cost criterion. Consistency with a priori knowledge is required if a priori knowledge has been employed in order to build the neural model. The dimensionality criterion places an upper bound on the size of the dataset that is necessary to obtain a good estimate of the model parameters (weights, thresholds).

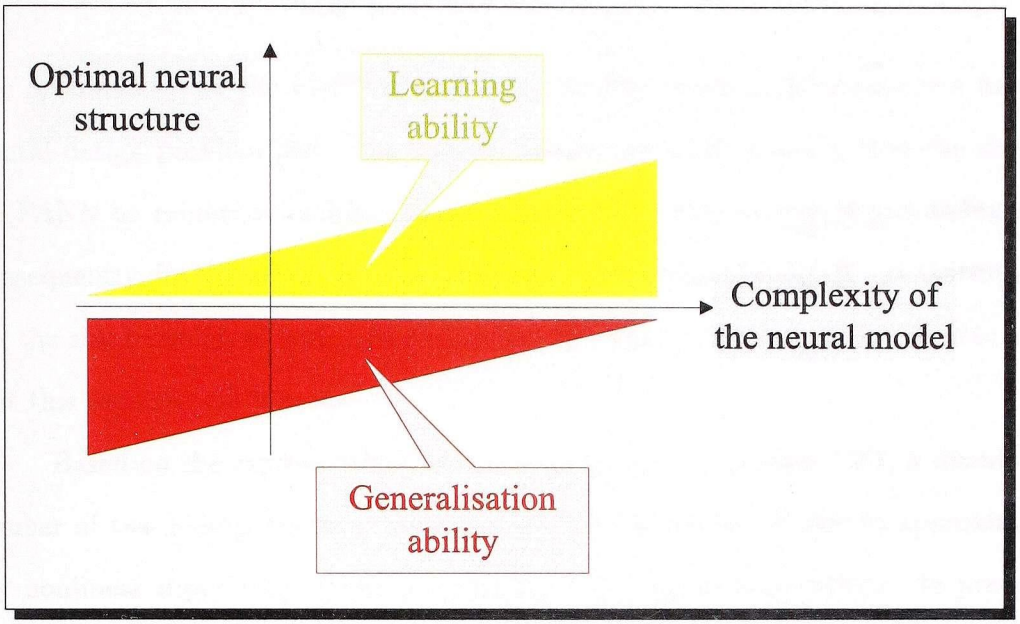


Figure 2.8: The relationship between learning, generalization and model complexity.

A FANN model that is designed so that the above mentioned criteria are met is, ideally, an optimal FANN structure. The optimality of the obtained FANN can be evaluated by using the average generalization error, the size of the neural structure (evaluated by the number of connections/nodes), or other criteria. As Figure 2.8 illustrates, these optimality criteria are not independent [42]. For instance, better generalization is clearly obtained when reducing the complexity of the FANN model. Additionally, this implies faster testing, and lower implementation costs.

2.3 FANN Design Methods

The optimization of the FANN neural structure for better performance is a fundamental design problem [39]. The general design rule (2.3) requires that the size of the FANN be minimized subject to the FANN being able to learn the training set. Consequently, FANN design is accomplished by specifying the number of hidden layers, the size (number of nodes) in each layer, and the connectivity of the structure so that this requirement is met.

Based on the representation theorem mentioned in Section 2.2.1, a maximum number of two hidden layers is necessary so that the FANN is able to approximate any nonlinear input–output function employed in current applications. In practice FANNs with more than two hidden layers may be selected, in order to avoid a large number of hidden nodes in each of the hidden layers.

Determining the size of the hidden layer, i.e. number of hidden nodes, is a difficult problem. If the number of the hidden nodes is too large, then the FANN has too many parameters and generalizes poorly [43]. However, if the number of hidden nodes is too small, then the number of weights that can be modified during the training stage is not sufficient. Consequently, the FANN model is unable to learn. The sizes of the input and output layer can usually be determined quite easily. More specifically, the size (i.e., number of nodes) of the FANN input layer is normally set to the size of the training patterns. The size of the output layer depends on the application problem. For instance, in a classification problem, the size of the output layer is equal to the number of classes. Finally, the FANN may be fully connected, if each simple node i in layer L is connected with each node in layer $L + 1$, and partially

connected otherwise.

A simple solution for the design problem stated earlier is to train several FANNs using the same training set, until the cost criterion is met. Then, the structure having the minimum size is selected. Although simple, this method is time consuming. Instead, several design methods have been proposed, all of which evaluate the size of the hidden layer(s). These methods can be empirical, statistical, or onthogenic. Empirical methods generally depend on the application problem [1, 2, 31, 35, 44, 45]. Statistical FANN design methods aim at obtaining a tradeoff between the complexity and the generalization performance of the FANN model. The model complexity is evaluated by the number of independent parameters. The generalization performance is generally evaluated by the average generalization error [46, 47]. The statistical criteria proposed for FANN selection in [39], [46]–[52] can be expressed by the general form $C(\mathbf{w}) = C_{training}(\mathbf{w}) + \gamma\, C_{complexity}(\mathbf{w})$, where $C(\mathbf{w})$, $C_{training}$, and $C_{complexity}$ are the cost terms that denote the overall complexity of the model, the training error, and the penalty for the model complexity, respectively. The term $C_{complexity}$ is different for various statistical criteria.

Onthogenic methods are based on modifications of the FANN topology by growing, pruning or hybrid methods. The topology can be changed by adding/deleting layers, nodes, or connections. When the FANN topology is modified, the size of the FANN structure is usually modified as well. Growing methods start from an initial small structure and successively add nodes and/or layers, until a desired cost criterion is met. On the other hand, pruning methods delete nodes and/or layers from a network with a reasonable size, until the cost constraint is violated. Hybrid methods

are based on combinations of the growing and pruning methods, usually improving performance. The onthogenic design methods, particularly pruning methods, are discussed next.

2.3.1 Growing Methods

As stated earlier, the growing methods for FANN design are onthogenic methods that start from an initial small structure and successively add nodes and/or layers, until a desired cost criterion is met. The resulting FANN is generally a large, fully connected and symmetrical neural structure.

FANN growing can be formulated as a search in the functional space. The goal of this search is to determine the size of the FANN which is able to approximate the desired input–output function based on the training data set. The initial and the final FANN structures, the type of elements (connections/nodes/layers) that are added in the neural structure, the number of the added elements, the connectivity of the inserted elements, and the stop criterion for the searching process in the functional space [43, 54] are specific to each growing design method. Many growing methods start from an initial structure with no hidden nodes or no direct input–output connections. The final structure, which can implement satisfactorily the desired input–output function, is not unique. Usually, the final structure that implements "well enough" the desired function according to a selected cost criterion, is selected. The elements that are added in the neural structure during the growing process can be connections, nodes, or layers. These elements are usually added one by one. The added elements can be fully or partially interconnected with the existing elements in the neural structure.

The growing process is terminated if a selected cost criterion is met. We note here that, the stop condition for a growing process can be expressed based on the statistical criteria mentioned earlier. The growing methods for FANN design can be classified, based on the growing strategy, as illustrated in Table 2.1.

Ad hoc methods are based on the idea that a new hidden node must be added in the FANN structure when the error stops decreasing for a selected period of time during the training proces. After a new hidden node is added, the weights are updated and training is resumed. Input space partitioning methods are based on the idea that each hidden node performs a partition of the input data space. If a new training vector, which is presented to the FANN input, is incorrectly classified in one of the existing partitions, then a new hidden node, representing a new partition, is added; otherwise, the network structure is not changed. Error correcting methods are based on the idea that the output error is generally expected to decrease when a new hidden node is added in the FANN structure. The methods that generate trees build FANN networks by adding nodes so that the resulting FANN structures are trees (pyramids) with the network output nodes on top. The methods that build the network starting from a tree are based on re–ordering the decision trees in multilayer perceptron–like structures. Finally, modularization methods are based on the idea that, by dividing a network into several sub–networks, each of which is able to solve a part of a large application problem, the FANN design by applying growing methods is simplified.

The advantages of the onthogenic growing methods for FANN design are small sizes of the initial neural structures and good scalability with the size of the dataset. The main drawbacks of the onthogenic growing methods are (a) performance degra-

Table 2.1: Onthogenic growing methods for FANN design. The acronyms N/L and FC/PC denote the element added in the network (Node/Layer) and the connectivity of the added element (Fully Connected/Partially Connected).

Method		Author(s)	Ref.	Remarks	N/L	FC/PC
Ad hoc	DNC	Ash	[55]	Dynamic Node Creation	1 N	FC
		Chang	[43]		1 N	FC
		Honavar & Uhr	[53]	Generation	1 N	N/A
		Mezard & Nadal	[22]	Tiling	N, L	PC
Input space partitioning	RCE	Reilly	[55]	Restricted Coulomb Energy	1 N	FC
	GAL	Alpaydin	[56]	Grow and Learn	1 N	PC
	OSA	Masciolli & Martinelli	[57]	OilSpot Algorithm	N/A	FC
Error correcting	CASCOR	Fahlman	[30]	CaSCade CORrelation	1 L of 1 N	FC
	CL	Refenes	[43]	Constructive Learning	1 N	N/A
	LPPA	Masciolli & Martinelli	[57]	Linear Programming Perceptron Algorithm	1 N	N/A
	GMDH	Ivakhnenko	[54]	Group Method of Data Handling	1 N	PC
Methods that generate trees	UPSTART	Frean	[58]		Groups of 2 N	N/A
		Nahban & Zomaya	[55]		N, L	N/A
Methods that start from a tree	EN	Sethi	[59]	Entropic Networks	N/A	N/A
		Brent	[60]		N/A	N/A
Modularization	TACOMA	Mirghafori & Morgan	[61]		N/A	N/A
		Lange & Voigt	[62]	TAsk Decomposition by COrrelation MeAsures	1 L	N/A

dation due to the overtraining which may be present when the new hidden nodes are fully connected with the existing nodes, (b) significant time requirements for the methods that generate trees and those that build the FANNs starting from a tree, (c) no theoretical proof of convergence for some of the growing methods, and (d) sub–optimality. Comparisons of some of the onthogenic growing methods have been performed by Fiesler [53], and Mascioli, Martinelli, and others [57].

2.3.2 Pruning Methods

As stated earlier, pruning methods delete connections, nodes, and/or layers from a FANN, until a cost constraint is violated. The initial structure is usually large enough to allow learning. The pruned structure is generally a simple, partially connected, and non–symmetrical neural structure [63]. The goal of FANN pruning methods is to obtain a neural structure with lower complexity than that of the initial structure, without significant performance degradation.

Let the FANN be a M–H–N multilayer perceptron. Without loss of generality, let us assume that $N = 1$. Moreover, let $\mathbf{w} = [w_1, ..., w_r, ..., w_\Pi]^T$ be the Π dimensional parameter vector of the network, where w_r is the r^{th} component of the vector and $1 \leq r \leq \Pi$. The training set consists of $\{\mathbf{x}(\xi), d(\xi)\}$, where $1 \leq \xi \leq P$. The notations $\mathbf{x}$, d, ξ, P denote the input vector, the desired output value, the index of the input pattern, and the total number of patterns, respectively. Finally, let us assume that the cost function to be minimized consists of one term, given by (2.5), and that, by applying a pruning method, a weight w_q is deleted, with $1 \leq q \leq \Pi$. Then, the hypotheses $w_q = 0$ and $w_q \neq 0$ must be compared by testing the model $\left(\mathbf{w}_{(q)}\right)$ with

respect to the model $(\mathbf{w})$. The parameter vectors $\mathbf{w}_{(q)}$ and $\mathbf{w}$ are identical, with the exception of the weight w_q, which is zero in the model $\left(\mathbf{w}_{(q)}\right)$ and nonzero in the model $(\mathbf{w})$. Successive deletions of the weights w_{q1}, w_{q2}, etc., are equivalent to testing the models $\left(\mathbf{w}_{(q1)}\right)$, $\left(\mathbf{w}_{(q2)}\right)$, and so on. If the weight $w_{q1} = 0$, then $w_{q1} = w_{q2} = 0$, and so on.

Several issues need to be addressed, such as the testing procedure for the model $\left(\mathbf{w}_{(q)}\right)$ with respect to the model $(\mathbf{w})$, the optimal sequence of connections to be deleted $q_1, q_2, ...q_L$, and the selection of the stop criterion. The testing of the model $\left(\mathbf{w}_{(q)}\right)$ with respect to the model $(\mathbf{w})$ is generally performed by selecting a cost criterion. The value of the cost function must be lower than a threshold in order to permanently delete a particular connection. The optimal sequence of connections to be deleted is difficult to select. A solution is the computation of sensitivity parameters associated to each weight. Next, the connection having the minimum sensitivity is removed. The pruning process is terminated if a selected stop criterion is met. This criterion can require, for instance, that the percentage of the deleted connections be below a threshold. Other stop criteria can be expressed based on the statistical criteria mentioned earlier.

The previously developed pruning methods for FANN design can be classified based on the pruning strategy as illustrated in Table 2.2.

Pruning methods that eliminate weights

Brute–force pruning methods, such as that proposed in [64], assign a zero value to one weight at a time. If the output error increases significantly, then the weight is restored to the initial value, otherwise it is permanently deleted. Weight clustering methods

Table 2.2: Onthogenic pruning methods for FANN design. The acronyms C/N denote the element pruned by the method (Connection/Node).

Method		Author(s)	Ref.	Remarks	C/N
Brute–force		Thodberg	[64]		C
		Reed	[43]		C
Weight clustering	LB	Kruschke	[65]	Local bottleneck	C
	DB	Kruschke	[66]	Distributed bottleneck	C
	SWS	Nowlan & Hinton	[43]	Soft Weight Sharing	C
Sensitivity estimation	OBD	Denker, LeCun & Solla	[67]	Optimal Brain Damage	C
	OCD	Cibias, Soulie & Gallinari	[68]	Optimal Cell Damage	C
	OBS	Hassibi & Stork	[69]	Optimal Brain Surgeon	C
	SSM	Cottrell & Girard	[50]	Statistical Stepwise	C
		Karnin	[70]		C
Regularization based		Ishikawa	[38]		C
		Hinton	[34]		C
		Krogh & Hertz	[22]		C
		Weigend	[34]		C
		Hansen	[71]	OBD with Weight Decay	C
		Hansen & Pedersen	[72]	OBD with Weight Decay	C
		Chauvin	[73]		C
		Ji	[74]		C
		Nowlan & Hinton	[75]		C
		MacKay	[76]		C
	CSDF	Yasui, Malinowski & Zurada	[77]	Convergence Suppression and Divergence Facilitation	C
Brute–force		Siestma & Dow	[43]		N
Weight clustering		Mozer	[78]	Skeletonization	N
	CSDF	Yasui, Malinowski & Zurada	[77]	Convergence Suppression and Divergence Facilitation	N
Sensitivity estimation		Mozer & Smolensky	[78]	Skeletonization	N
		Kruschke	[66]		N
Other		Xue & Hu	[79]	Correlation	N
	FARM	Kung & Hu	[80]		N

are based on the idea that, by grouping the weight vectors which enter the hidden nodes, the dimension of the space scanned by these vectors is reduced. The weight clustering (grouping) can be performed by a competition of the hidden nodes, as in the case of the "local bottleneck" (LB) and "distributed bottleneck" (DB) methods [65], or by other methods [43]. Sensitivity estimation pruning methods are based on the idea that the elimination of any weight in a FANN has an impact on the value of the cost function. In addition to the "sensitivity", various other indices such as the "saliency", "evidence", and "relevance" have been employed in this class of pruning methods.

In the Optimal Brain Damage (OBD) algorithm, proposed by Denker, Le Cun, Solla and others [67], the estimate of the error increase when some of the connections are deleted is expressed in terms of the saliency of the cost function[7]. The weights that have minimum saliency are permanently deleted.

Let the symbols M–H–N denote a FANN with M input, H hidden and N output nodes. Also, let P be the number of training patterns, $\mathbf{x}(\xi)$, $\mathbf{y}(\xi)$, $\mathbf{d}(\xi)$ and $\mathbf{z}(\xi)$ be the input, the actual output, the desired output and the hidden layer output vectors (respectively), for an input pattern ξ, w_{ih} be the weight connection between an input node i and a hidden node h, v_{hj} be the weight connection between a hidden node h and an output node j. Finally, let's assume that the FANN parameters are the components of a Π–dimensional vector $\mathbf{w} = [w_1, \ldots, w_r, \ldots, w_\Pi]^T$, where Π is the total number of parameters, and that $N = 1$. When a component of $\mathbf{w}$ is deleted, the value of the cost function $C(\mathbf{w})$ changes by $\delta C(\mathbf{w})$, which is given by

[7]To be consistent with other references, we shall use the shorter expression "the weight saliency".

$$\delta C(\mathbf{w}) = C(\mathbf{w} + \delta\mathbf{w}) = \sum_{r=1}^{\Pi} \frac{\partial C}{\partial w_r} dw_r + \frac{1}{2} \sum_{r=1}^{\Pi} \frac{\partial^2 C}{\partial w_r^2} dw_r^2 + \tag{2.6}$$

$$+ \frac{1}{2} \sum_{r \neq q}^{\Pi} \frac{\partial^2 C}{\partial w_r \, \partial w_q} dw_r \, dw_q + O\left(\|\delta \mathbf{w}\|^3\right).$$

Assume that the cost function reaches a local minimum value, i.e., $\nabla C = 0$. Moreover, assume that terms of order higher or equal to two in Equation (2.6) are neglected. Finally, assume that only one weight at a time is deleted, which means that $\delta w_r = 0$ for all r, except for the case when $r = q$. Therefore, the terms $\nabla^2 C / \nabla w_r \nabla w_q$ outside the main diagonal may be dropped [67], [71]. With these assumptions and using the Levenberg–Marquardt approximation for the Hessian matrix, Equation (2.6) becomes

$$\delta C \approx \frac{1}{2} \sum_q \frac{\partial^2 C}{\partial w_q^2} (dw_q)^2 .$$

Since $\delta w_q = -w_q$ for $r = q$, this equation is equivalent to

$$\delta C \approx \frac{1}{2} \sum_q \frac{\partial^2 C}{\partial w_q^2} w_q^2.$$

The increase in the cost function value may also be expressed as

$$\delta C = \sum_{q \in \mathcal{D}} s_q, \tag{2.7}$$

where $\mathcal{D}$ is the set of the weights $\{w_q \mid q \in \mathcal{D}\}$ to be deleted. Then, the saliency s_q of w_q can be expressed by

$$s_q \cong \frac{1}{2} \frac{\partial^2 C}{\partial w_q^2} w_q^2.$$

If the cost function is additive, i.e. $C = \frac{1}{P} \sum_{\xi=1}^{P} C(\xi)$, and if the output value of the single hidden layered FANN is $y(\xi) = F(\mathbf{w}; x(\xi))$, then the saliency can also be expressed by

$$s_q = w_q^2 \frac{1}{2P} \sum_{\xi=1}^{p} \frac{\partial^2 C(\xi)}{\partial (y(\xi))^2} \left(\frac{\partial F(\mathbf{w}; \mathbf{x}(\xi))}{\partial w_q} \right)^2 .$$

Expressing the FANN input–output mapping by [71]

$$F(\mathbf{w}; \mathbf{x}(\xi)) = \sum_{h=1}^{H} v_h \tanh \left(\sum_{i=0}^{M} w_{ih} x_i(\xi) \right) + v_0,$$

the input–hidden and hidden–output saliencies can then be given by

$$s_h = v_h^2 \frac{1}{2P} \sum_{\xi=1}^{P} \tanh^2 \left(\sum_{i=0}^{M} w_{ih} x_i(\xi) \right) \text{ and} \tag{2.8}$$

$$s_{ih} = v_h^2 w_{ih}^2 \frac{1}{2P} \sum_{\xi=1}^{P} \left[1 - \tanh^2 \left(\sum_{m=1}^{M} w_{mh} x_m(\xi) \right) \right]^2 , \text{ where} \tag{2.9}$$

$v_h = v_{hj} |_{j=1}$ and $s_h = s_{hj} |_{j=1}$. The saliencies are computed as follows:

REPEAT

1. Train the network until the error $C < \epsilon_1$, with ϵ_1 given.
2. Compute the weight saliencies and arrange them in a decreasing order.
3. Delete the connection with minimum saliency. Go to step 1.

UNTIL

The stop condition is satisfied (e.g., a selected percentage of the weights has been deleted).

The "Optimal Cell Damage" (OCD) method, introduced by Cibias, Soulie, Gallinari, and others [68], is a variation of the OBD method. However, in the OCD method, some of the input variables are also deleted. The "Optimal Brain Surgeon" (OBS) method proposed by Hassibi and Stork [69] employs the entire Hessian matrix for the estimation of the weight saliencies. Thus, the OBS method is a generalization of the OBD method. The performance of the OBS method is better than that of the OBD, however its complexity is also higher. The entire Hessian matrix is also employed in the "Statistical Stepwise" method proposed by Cottrel, Girard, and others [50].

Regularization–based pruning methods employ the minimization of a cost function which contains a penalty term. This term penalizes the neural structures having high complexity. As stated in Section 2.2.1, weight decay regularization occurs when, due to the selected penalty term, some of the FANN weight values become zero. Thus, the corresponding connections are deleted from the neural structure. Various penalty terms for weight decay regularization–based pruning have been proposed in [34, 38, 71]. A variation of the OBS method with weight decay has been proposed by Hansen and Pedersen [72]. Various cost functions for pruning methods, which implicitly perform regularization, have been introduced by works such as [73]–[77]. An adaptive regularization method for weight pruning has been proposed by [76], that eliminates the empirical selection of the regularization parameters which is required in most of the above mentioned methods.

Pruning methods that eliminate hidden nodes

"Brute–force" methods for pruning the FANN nodes are based on the evaluation of redundancy criteria. For instance, if the output of a hidden node is constant for all of the training patterns, then the node can be deleted. Alternatively, if two hidden nodes have the same output value with the same sign or with opposite signs for all of the training patterns, then one of these nodes can be deleted. Siestma and Dow [43] have employed this idea in their interactive method for pruning of the FANN, which allows the user to decide by inspection which nodes can be eliminated. Sensitivity estimation methods generally delete the hidden nodes that do not lead to major changes in the value of the cost function. Mozer and Smolensky [78] have proposed a skeletonization method based on the computation of a "relevance" index. The "relevance" index is computed for each node as the difference between the output error obtained without the node in the structure, and the output error obtained with the node in the structure. A low relevance is associated with a low importance of the node, which can then be deleted. Kruschke [66] has proposed a sensitivity–based pruning method which uses the competition of the hidden nodes combined with lateral inhibition. The nodes that are completely inhibited can be deleted from the FANN structure. Finally, other methods based on the evaluation of the correlation between the hidden nodes [79], as well as recursive least squares approximation of the best set of hidden nodes [80], have also been proposed.

The advantages of the pruning methods for FANN design are the small size of the resulting neural structures, and the good generalization due to the small number of parameters. Moreover, overtraining is avoided. The main drawbacks of the pruning

methods are related to (a) the selection of the initial FANN size, (b) time requirements due to the FANNs having large sizes during many of the pruning stages, and (c) sub–optimality. Comparisons of some of the onthogenic methods have been performed by Fiesler [53]. However, a quantitative comparison is difficult due to their employing various benchmark data sets.

2.3.3 Onthogenic Hybrid Methods

The hybrid methods for FANN design combine the growing and pruning methods in order to improve performance. The hybrid methods that have been proposed can be generally classified into sequential and non–sequential methods. Sequential hybrid methods consist of a growing stage followed by a pruning stage. In these methods, the layers are added one by one and the nodes are pruned one by one [55], [81]–[83], or only the nodes are added/pruned [84, 85]. Non–sequential hybrid methods cannot be decomposed into distinct growing/pruning stages, although growing/pruning operations are employed at various design stages. An example is the "Controlled Growth of CASCOR" method [72], which is a modified version of the basic CASCOR method. More specifically, a hidden node is first pruned before being added in the structure. Another solution has been proposed in [36], where the hidden nodes are added during training if an information theory criterion is met.

The advantage of the onthogenic hybrid methods for FANN design is better performance than that of the growing and pruning methods. The main drawback of the onthogenic hybrid methods is their high complexity.

2.4 Image Subsampling

Image subsampling is equivalent to intelligently discarding data [19, 5], and it can be informally defined as the process of representing an input image on a new sampling grid, with a lower sampling density than the original grid [6, 5]. Subsampling is also known as downsampling, scaling down, shrinking or decimation, and belongs to the class of geometric scaling procedures, which also include the reverse operation, i.e. upsampling or upscaling, achieved either by pixel replication or by interpolation.

Several approaches to subsampling are possible and they differ according to the subsampling domain, the geometry and the type of the subsampling grid. The standard references for sampling issues in digital signal processing are [86, 87], but important details are also included in other publications (e.g., [10, 11]). We shall next briefly refer to some of them, based on the above classification.

2.4.1 The Subsampling Domain

For still images, the subsampling domain is either the spatial or the frequency domain [8]. In the case of video sequences, spatial (intra–frame), temporal (inter–frame) [16, 88] or spatio–temporal subsampling [5] is performed. These are discussed next.

Subsampling of 1–D Signals in the Spatial Domain

Let us denote by $x_c(t)$ and $x[n]$ a continuous–time signal and a discrete sequence of samples obtained from the continuous signal by periodic sampling, respectively. In other words, $x[n] = x_c(nT)$, where $-\infty < n < +\infty$ [86]. Let us also denote by $X_c(j\Omega)$ the Fourier transform of the original signal $x_c(t)$. The Fourier transform of

the sampled sequence consists of periodically repeated copies of $X_c(j\,\Omega)$, which are separated by integer multiples of the sampling frequency. If the sampling frequency Ω_s is $\Omega_s > 2\,\Omega_N$, that is, if the sampling frequency is higher than twice the highest frequency component of the continuous signal, then these replicas of $X_c(j\,\Omega)$ do not overlap. Consequently $x_c(t)$ can be recovered from the sampled sequence with an ideal lowpass filter. If the sampling frequency does not meet the above condition, the reconstructed signal is distorted due to aliasing. The frequency Ω_N is commonly referred as the Nyquist frequency. The frequency $2\,\Omega_N$ that must be exceeded by the sampling frequency is called the Nyquist rate. The above discussion is expressed by the Nyquist theorem, stated as follows [86]:

Theorem 2 *(Nyquist) Let $x_c(t)$ be a bandlimited signal with $X_c(j\,\Omega) = 0$ for $|\,\Omega\,| > \Omega_N$. Then, $x_c(t)$ is uniquely determined by its samples $x[n] = x_c(n\,T)$, where $n = 0, \pm 1, \pm 2, \ldots$, if $\Omega_s = (2\pi/T) > 2\,\Omega_N$.*

Subsampling of 2–D Signals in the Spatial Domain

Let us denote by $x_c(t_1, t_2)$ a continuous–time two–dimensional signal. A discrete sequence (array) $x[n_1, n_2]$ of samples from the continuous signal $x_c(t_1, t_2)$ can be obtained by periodic sampling; that is, $x[n_1, n_2] = x_c(n_1\,T_1, n_2\,T_2)$, where $-\infty < n_1, n_2 < +\infty$ and T_1, T_2 are positive real constants known as the horizontal and vertical sampling intervals (periods). Let us also denote by $X_c(j\,\Omega_1, j\,\Omega_2)$ the Fourier transform of the original signal $x_c(t_1, t_2)$. The Fourier transform of the sampled sequence consists of periodically repeated copies of $X_c(j\,\Omega_1, j\,\Omega_2)$. If the condition

$$X_c(\Omega_1, \Omega_2) = 0 \quad \text{for } |\,\Omega_1\,| \geq \frac{\pi}{T_1}, |\,\Omega_2\,| \geq \frac{\pi}{T_2}$$

is satisfied, then the continuous bandlimited signal $x_c(t_1, t_2)$ can be recovered from its samples $x[n_1, n_2] = x_c(n_1 T_1, n_2 T_2)$ [87]. This is basis of the sampling theorem in the two–dimensional case.

Although images may be generally modeled as bandlimited signals, image conditioning is usually performed before subsampling in order to meet the above conditions. Image conditioning commonly involves lowpass filtering (LPF), which can eliminate or minimize the overlapping of spectral components [86]–[89]. Of course, much of high frequency information would consequently be lost.

Subsampling of 2–D Signals in the Frequency Domain

When the input image/video frames have been transformed, it is often desirable to perform downsampling in the frequency domain, thus avoiding the inverse transform, downsampling and forward transform of the images. A simple solution to downsample in the frequency domain (e.g., DCT domain) is to discard some of the coefficients of the transformed image. The remaining coefficients are then used to reconstruct an image having a lower resolution. Although simple, this method leads to severe artifacts in the reconstructed image. One of the alternatives is to compute new DCT coefficients for the lower resolution image by using the original DCT coefficients [90].

Subsampling Domains for Video Sequences

Spatial subsampling of video sequences is similar to that of still images. Temporal subsampling is based on applying a subsampling grid along the temporal dimension of the three–dimensional video signal. By the separate spatial and temporal subsampling of video, one implicitely assumes that the spatial and temporal components of

the video are independent. However, as this is not necessarily true, spatio–temporal subsampling must be also addressed. It has been shown in [88] that three–dimensional non–separable sampling leads to better perceptual quality as compared to other methods. A thorough discussion of spatial, temporal and spatio–temporal subsampling of video sequences can be found in [16].

2.4.2 The Subsampling Grid

As discussed in Section 2.4.1, downsampling a 1–D sequence by a factor of M in the spatial domain is accomplished by simply retaining every M^{th} sample and discarding the rest of the samples. Downsampling a two–dimensional sequence in the spatial domain is accomplished by retaining some of the samples in the plane (n_1, n_2). The sampling locations in this plane are organized as a lattice (grid). The downsampling factor is replaced in the two–dimensional case by a downsampling matrix $\mathbf{D}$, which is nonsingular and has integer values.

Definition 1 *Let* $\mathbf{d}_1$, $\mathbf{d}_2$ *be two linearly independent real vectors in two–dimensional real Euclidian space* $\Re^2$. *A lattice* Λ *in* $\Re^2$ *is the set of all linear combinations of* $\mathbf{d}_1$, $\mathbf{d}_2$ *with integer coefficients [91]*

$$\Lambda = \{\lambda_1 \mathbf{d}_1 + \lambda_2 \mathbf{d}_2, \lambda_1, \lambda_2 \in \mathcal{Q}\}$$

Definition 2 *If every point of a lattice* Λ *is also a point of a lattice* $\mathcal{M}$, *then* Λ *is a sublattice of* $\mathcal{M}$ *[91].*

Let us now restrict $\mathbf{d}_1$, $\mathbf{d}_2$ to be integer vectors and let us assume that they are the columns of a subsampling matrix $\mathbf{D}$. A two–dimensional downsampler retains only

samples at points on a sublattice generated by the matrix $\mathbf{D}$, that is, points $\mathbf{m}$ of the form $\mathbf{m} = \mathbf{D}\,\mathbf{n}$, or in a matrix form,

$$\begin{pmatrix} m_1 \\ m_2 \end{pmatrix} = \mathbf{D} \begin{pmatrix} n_1 \\ n_2 \end{pmatrix}, \text{ where } \mathbf{D} = \begin{pmatrix} d_{00} & d_{01} \\ d_{10} & d_{11} \end{pmatrix},$$

and $\mathbf{n}$ is an arbitrary integer vector. One out of every $\mid det\,(\mathbf{D}) \mid$ samples of the sequence is retained, where $det\,(\mathbf{D})$ is the determinant of the matrix $\mathbf{D}$ [12, 87, 92].

It can be shown that $\mathbf{D}$ can always be expressed in a simpler form given by

$$\mathbf{D} = \begin{pmatrix} d_{00} & d_{01} \\ 0 & d_{11} \end{pmatrix}.$$

It can be also shown that rectangular sampling corresponds to a diagonal sampling matrix [12, 87, 91]. Examples of these cases are illustrated in Figure 2.9 (adapted from [91]). Clearly, the specific geometry of the subsampling sublattice is determined by the values of the subsampling matrix coefficients.

In addition to its geometry, the subsampling lattice (grid) is defined by its type. The type of the subsampling grid may be fixed or adaptive. In fixed grid subsampling, the two–dimensional signal (image) is subdivided into equal regions, and the same grid is used for each of the regions. Although computationally simple, this method does not take into account the amount of detail present in certain regions of the input image, and thus, visible artifacts are usually present. Spatially adaptive subsampling methods use a dense sampling grid for each active (i.e., high–detail) region in the image, and one that contains only a few pixels for regions with little

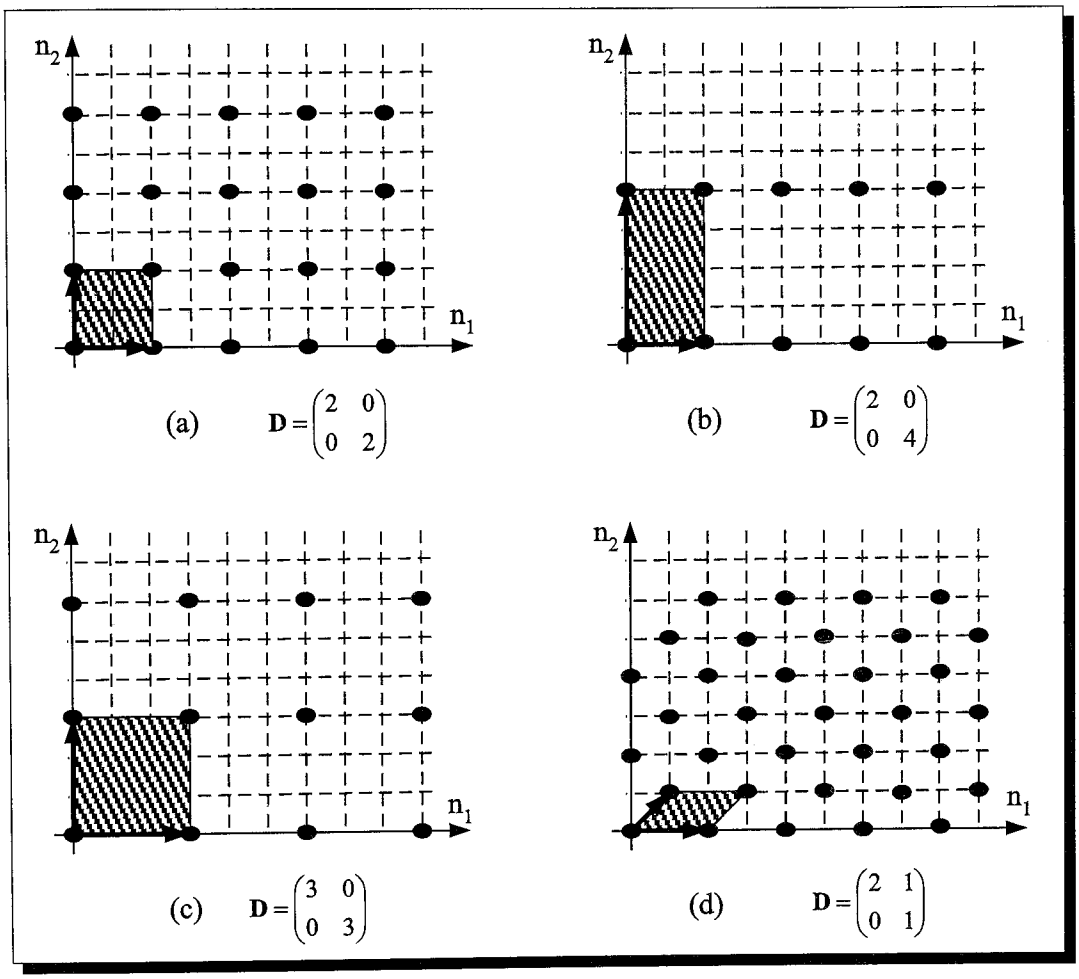


Figure 2.9: Subsampling lattices, sublattices, and corresponding subsampling matrices: (*a*) separable subsampling by 2 in each direction, (*b*) nonseparable subsampling by 2 and by 4, respectively, (*c*) separable subsampling by 3 in each direction, and (*d*) quincunx subsampling.

detail [5, 4, 14, 89]. Adaptive subsampling methods outperform the fixed grid ones, but their associated interpolation steps are also significantly more complex [93]–[98].

2.4.3 The Subsampling Order

Most existing subsampling methods are based on pixel neighborhood operations [19]. A simple way to downsample a two–dimensional signal (image) is to select one pixel within a local neighborhood to be representative of its surrounding pixels, as illustrated in Figure 2.10 (*a*). Another way is to compute a statistical measure of the local intensity values, such as the mean, which will represent in the downsampled image the entire input block, as illustrated in Figure 2.10 (*b*). In each of these methods, first–order subsampling (FOS) or high–order subsampling (HOS) can be performed, depending on the size of the input and that of the subsampled block.

In first–order subsampling, which is illustrated in Figures 2.11 (*a*) and (*b*), subsampling by 2 in each direction is being performed. A total subsampling rate of $4 : 1$ has been obtained in each of the examples illustrated in Figures 2.11 (*a*) and (*b*), although the size of the input and output blocks is different in each case.

In high–order subsampling, which is illustrated in Figures 2.11 (*c*) and (*d*), subsampling by more than 2 in each direction is performed. In both examples, the subsampling rate is equal to $16 : 1$. In the single–stage high–order subsampling case, which is illustrated in Figure 2.11 (*c*), the selection of the output pixel value is difficult. A solution to address this problem is to perform multi–stage first–order subsampling, which is illustrated in Figure 2.11 (*d*). In this case, a HOS stage is being decomposed into a sequence of first–order subsampling stages.

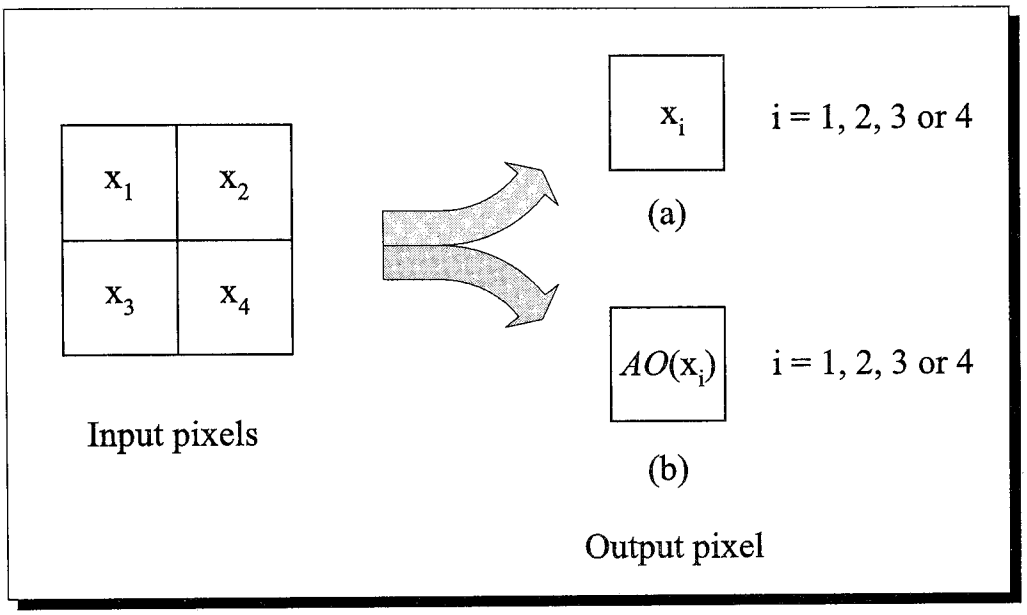


Figure 2.10: A simple example of first–order subsampling, where (a) one pixel is selected as the representative of all pixels in the input block, and (b) the output pixel is obtained by applying an arithmetic operation (AO) on the input pixels.

2.5 Summary

In this chapter, we have briefly reviewed some fundamental concepts that will serve as background material throughout the thesis. These concepts are related to feedforward neural networks and image subsampling. We have presented the characteristics of FANNs, i.e., the processing node, the activation function, the topology, the cost function and the training algorithm. Next, we have addressed FANN learning as an approximation/optimization process. We have also summarized the most popular supervised FANN learning algorithms and the criteria for FANN model evaluation. A review of the FANN design methods has also been included, focusing on the on-

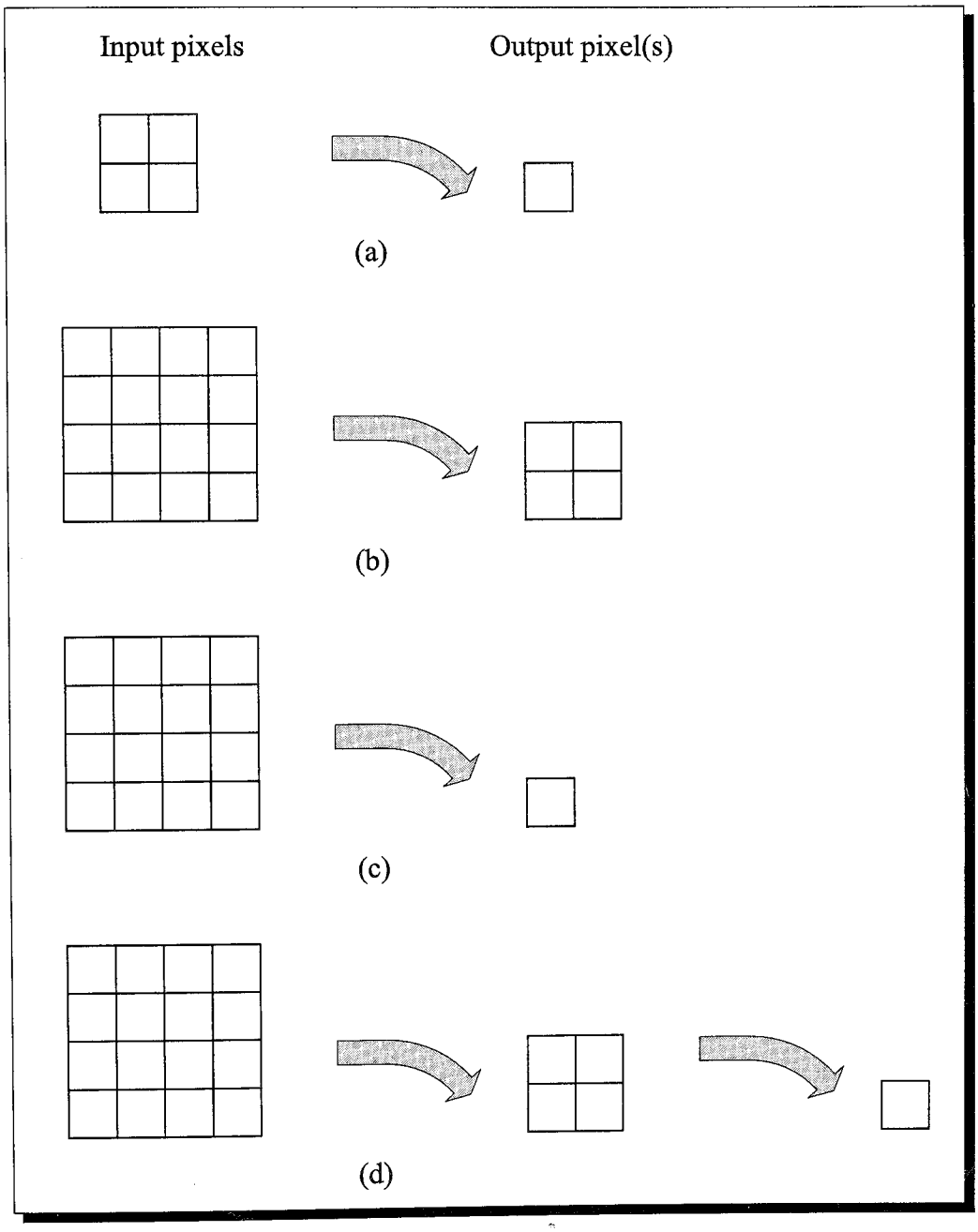


Figure 2.11: (*a*) First–order subsampling of a 2×2 block, (*b*) first order subsampling of a larger input block, (*c*) single–stage high–order subsampling, and (*d*) multi–stage first–order subsampling. In each case, a rectangular subsampling grid is employed.

thogenic (growing, pruning and hybrid) methods.

In the second part of the chapter, we discussed the main concepts related to image subsampling. More specifically, the subsampling domain, the subsampling grid and the subsampling order have been addressed. We have illustrated the concept of 2–D spatial subsampling using various subsampling grids, and we have presented solutions for first–order and high–order image subsampling.

Chapter 3

Symmetrical Pruning for FANN Design

In this chapter we address symmetrical pruning for feedforward neural network design. In Section 3.1, we provide a motivation for introducing tridiagonal symmetry constraints in the FANN design. In Section 3.2, we propose a tridiagonally symmetrical pruning algorithm. In Section 3.3, we illustrate, via a simulation example, that the designed FANN structures obtained by applying the proposed algorithm are compact and tridiagonally symmetrical. In the same section, we also compare the results of our proposed algorithm with those of the Optimal Brain Damage algorithm, already presented in Section 2.3.2. A summary of the chapter is included in Section 3.4.

3.1 Motivation

Symmetry can be defined as "the repetition of exactly similar parts facing each other or a center" [99]. When they are introduced in the FANN design, the symmetry

constraints may refer to the data set, to the partial derivative equations in the training algorithm or to the network topology [100]. Among the few studies on FANN design with symmetry constraints, we mention Shawe–Taylor's [101] work. He obtained a network with identical weight values for the symmetrical connections and an output invariant for a set of transforms performed on the input data. Yang, Yin, Gabbouj, and others [102] studied several ways of introducing symmetry in filter structures. They required that specific details of the input signal be preserved. The weights could have had different values, but the corresponding connections had to be symmetrically positioned in the structure. In this chapter, we address the FANN design with topological symmetry constraints. More specifically, tridiagonal symmetry constraints will be placed on the position, not on the weight values, of the weight connections in the structure.

Tridiagonally symmetrical neural structures are desirable for efficient hardware and software implementations. First, the memory requirements for storing the FANN weights can be generally reduced for a tridiagonal structure. This is important in many hardware and software implementations, especially in embedded applications. For example, for the non–symmetrical FANN illustrated in Figure 3.1 (a), the entire 5×4 input–hidden weight matrix (i.e., 20 values) must be stored. Alternatively, 10 non–zero weight values and their corresponding 10 indices can be stored. For the tridiagonally symmetrical FANN (TS–FANN) illustrated in Figure 3.1 (b), having the same number of weight connections as the FANN in Figure 3.1 (a), only the 10 non–zero weight values and at most half of the corresponding indices[1]

[1]Further memory savings may be obtained by applying, for example, a generic scanning rule for the diagonal weight values.

must be stored. Second, accessing the weight values in a tridiagonal FANN is more efficient than in a non–symmetrical FANN. For instance, assume that a simple zig–zag scanning rule (shown by arrows in Figure 3.1 (b)) is employed in order to read the weight values. Then, it is sufficient to store the weight values of the tridiagonal FANN illustrated in Figure 3.1 (b) as the sequence of numbers corresponding to [0 w_{23} 0 0 0 w_{44} w_{33} w_{22} w_{11} w_{21} w_{32} w_{43} w_{54}] in order to obtain a complete description of the FANN input–hidden layer connectivity. Last and most important, both the mapping of a tridiagonal FANN structure onto parallel VLSI or programmable digital signal processors, and the optimization of the corresponding hardware/software realizations, are easier as compared to the non–symmetrical case [103]–[106].

If the tridiagonal symmetry constraints are taken into account during training, the obtained TS−FANN structure may lead to several minima of the cost function, periodicities or almost flat zones [22]. Multiple minima are due to possible permutations of the nodes in a layer or to the equivalent structures obtained by changing the sign of all the weights entering in and exiting from a hidden node [22, 107]. Alternatively, if the tridiagonal symmetry constraints are introduced after training, the above problems can be avoided. Therefore, we propose that initial training be performed without any symmetry constraints and be followed by pruning with tridiagonal symmetry constraints. Our goals are (a) to reduce the structure until the weight matrix becomes tridiagonal, (b) to prove that, based on useful approximations, not only tridiagonally symmetrical pruning is simple and fast, but the final structure is also compact, and (c) to illustrate the good performance of the algorithm even when the application problem does not contain obvious symmetries.

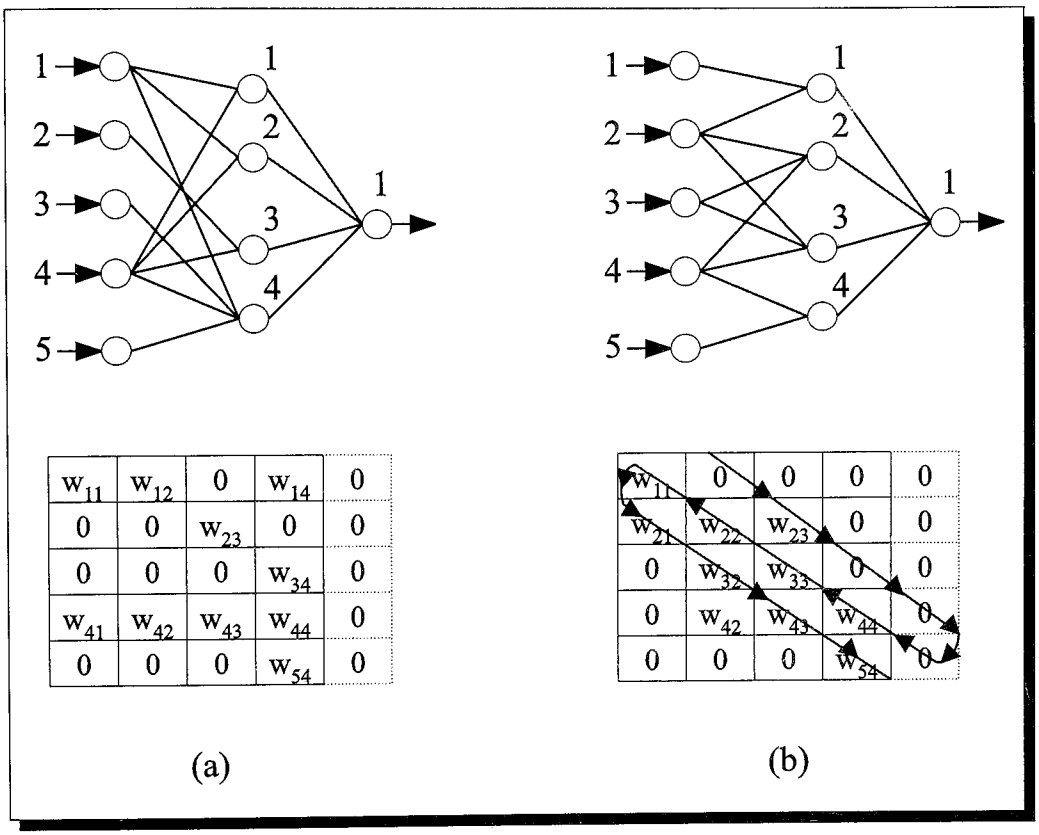


Figure 3.1: An example of a (a) non–symmetrical, and (b) tridiagonally symmetrical FANN. The corresponding input–hidden weight matrix is shown below each FANN structure. The dotted lines indicate where the weight matrix is padded with zeros so that it becomes a square matrix. Note that the number of weights in each case is equal to 10. The arrows illustrate a simple zig–zag scanning rule that can be employed in order to read the non–zero weight values.

3.2 Proposed Algorithm

With tridiagonal symmetry constraints, the optimization problem (2.3) becomes the minimization of the neural network size subject to the network being able to learn the dataset, the connections between the nodes being tridiagonally symmetrical in the final structure, and the weight values being not necessarily equal. As stated above, this design problem may be approached by introducing tridiagonal symmetry constraints in the pruning procedure. The algorithm that we propose in this section makes use of approximations in order to both satisfy the symmetry constraints and to reduce the design time. Next, we present the proposed Tridiagonal Optimal Brain Damage (TOBD) algorithm. This is followed by a discussion on important issues associated with both the training and testing steps.

3.2.1 Tridiagonal Optimal Brain Damage (TOBD) Algorithm

In what follows, we briefly define sparse matrices in general and tridiagonal matrices in particular. An $m \times n$ matrix is sparse if it has a small number τ of nonzero elements, $\tau \ll mn$. In other words, for increasing m and n, $\tau < \min(m, n)^{(1+c)}$, $0 \leq c \leq 1$. A sparse matrix $\mathbf{M}$ with bandwidth B has elements that are equal to zero if $\mid i - j \mid \geq B$, and elements that are equal to M_{ij} otherwise. If all the matrix elements are zero for $i \neq j$ and $B = 2$, then $\mathbf{M}$ is tridiagonal. Any matrix can be reduced to a tridiagonal form in a finite number of plane rotations, based on the Givens method, or reflections, based on the Householder method [108].

The $m \times m$ matrix $\mathbf{M}$ and $\Im \mathbf{M} \Im^{-1}$ have the same eigenvalues for any nonsingular matrix $\Im$. The transform $\Im \mathbf{M} \Im^{-1}$ is known as a similarity transform. $\Im$ is a

reflection if [108]

$$\Im(\mathbf{q}) = \mathbf{I} - 2\,\mathbf{q}\,\mathbf{q}^T, \quad \text{with } \mathbf{q}^T\,\mathbf{q} = 1. \tag{3.1}$$

Moreover, the matrix $\Im(\mathbf{q})$ is orthogonal, i.e., it preserves the Euclidian norm, and it is also symmetrical as

$$\left(\mathbf{I} - 2\,\mathbf{q}\,\mathbf{q}^T\right)\left(\mathbf{I} - 2\,\mathbf{q}\,\mathbf{q}^T\right) = \mathbf{I} - 4\,\mathbf{q}\,\mathbf{q}^T + 4\,\mathbf{q}\,\mathbf{q}^T\,\mathbf{q}\,\mathbf{q}^T = \mathbf{I},$$

and $\Im^{-1}(\mathbf{q}) = \Im^T(\mathbf{q}) = \Im(\mathbf{q})$, where $\mathbf{I}$ is the unitary matrix. We assume now that the matrix $\mathbf{M}$ is the $M \times H$ input–hidden weight matrix, denoted by $\mathbf{W}$. Without loss of generality, we also assume that $M > H$ and that the columns of $\mathbf{W}$ are the weight vectors entering each hidden node (e.g., the first column contains the weight values from all the input nodes into the first hidden node). Our aim is to find the transform matrix $\Im$, such that the reflection $\Im\,\widetilde{\mathbf{W}}\,\Im^{-1}$ performed on the matrix $\widetilde{\mathbf{W}}$ yields a tridiagonal weight matrix. The matrix $\widetilde{\mathbf{W}}$ is obtained by padding $\mathbf{W}$ with zeros, until it becomes an $M \times M$ square matrix. Its elements are

$$\tilde{w}_{ih} = \begin{cases} w_{ih} & \text{for} \quad 1 \le i \le M,\ 1 \le h \le H, \text{ and} \\ 0 & \text{for} \quad 1 \le i \le M,\ H+1 \le h \le M. \end{cases}$$

First, let us assume that the minimum OBD saliency value corresponds to an input–hidden weight $\tilde{w}_{i^*h^*}$ that is a diagonal element of $\widetilde{\mathbf{W}}$, i.e., $i^* = h^*$. We also require that symmetry constraints be satisfied. More specifically, we need to satisfy *the neighboring condition*: the minimum saliency connection is deleted ($\tilde{w}_{i^*h^*} = 0$), but the neighboring connections in the network with the weights $\tilde{w}_{h^*-1,h^*}$, $\tilde{w}_{h^*+1,h^*}$ (neighbors on the same column) and $\tilde{w}_{h^*,h^*-1}$, $\tilde{w}_{h^*,h^*+1}$ (neighbors on the same line)

are preserved, and all the other components of the vector $\tilde{\mathbf{w}}_{h^*}$ are deleted. We also need to satisfy *the unitary condition* given by $\mathbf{q}^T\,\mathbf{q} = 1$. If $\widetilde{\mathbf{W}}$ is multiplied to the left by $\Im(\mathbf{q})$, one obtains $\widetilde{\mathbf{W}}^{(left)} = \Im(\mathbf{q})\,\widetilde{\mathbf{W}}$. Then, the column $\tilde{\mathbf{w}}_{h^*}$ of $\widetilde{\mathbf{W}}$, $1 \leq h^* \leq M$, is independently transformed into the column $\tilde{\mathbf{w}}_{h^*}^{(left)}$ of $\widetilde{\mathbf{W}}^{(left)}$, given by

$$\tilde{\mathbf{w}}_{h^*}^{(left)} = \left(\mathbf{I} - 2\,\mathbf{q}\,\mathbf{q}^T\right)\,\tilde{\mathbf{w}}_{h^*} = \tilde{\mathbf{w}}_{h^*} - 2\,\left(\mathbf{q}^T\,\tilde{\mathbf{w}}_{h^*}\right)\,\mathbf{q}.$$

Using the neighboring condition, we require that all the weights in the same column be preserved, and thus the elements of $\mathbf{q} = [q_1,\, q_2,\, \ldots\, q_M]^T$ become

$$q_i = \begin{cases} 0 & \text{for } \; i = h^* - 1 \quad \text{or} \quad i = h^* + 1, \text{ and} \\ \\ \dfrac{\tilde{w}_{ih^*}}{2\sum_{m=1}^{M} q_m \tilde{w}_{mh^*}} & \text{for } \; 1 \leq i < h^* - 1 \\ & \text{or} \quad i = h^* \quad \text{or} \quad h^* + 1 < i \leq M. \end{cases}$$

Together with satisfying the unitary condition, we obtain

$$q_i = \begin{cases} 0 & \text{for } \; i = h^* - 1 \quad \text{or} \quad i = h^* + 1, \text{ and} \\ \\ \dfrac{\tilde{w}_{ih^*}}{\sqrt{\sum_{m=1, m\neq h^*-1, m\neq h^*+1}^{M} \tilde{w}_{mh^*}^2}} & \text{for } \; 1 \leq i < h^* - 1 \quad \text{or} \quad i = h^* \\ & \text{or} \quad h^* + 1 < i \leq M. \end{cases} \tag{3.2}$$

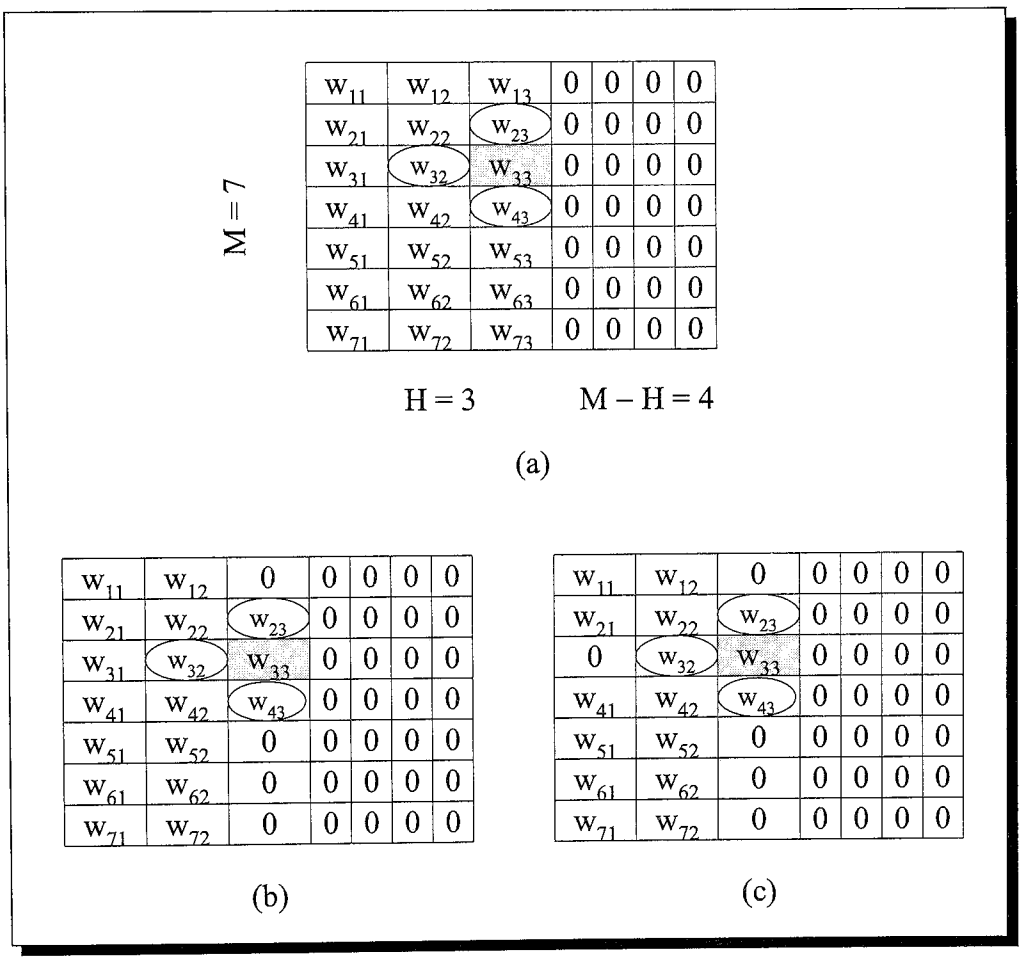


Figure 3.2: An example of applying the transform on the padded weight matrix in the TOBD algorithm. We assume that the minimum saliency weight is w_{33}. (a) is the weight matrix before applying the transform, (b) is the weight matrix after the multiplication to the left with the transform matrix, and (c) is the weight matrix after the result of (b) is multiplied to the right by the inverse of the transform matrix.

Similarly, the multiplication of $\widetilde{\mathbf{W}}$ to the right by $\Im(\mathbf{q})^T$ independently transforms each line of the matrix into a line of $\widetilde{\mathbf{W}}^{(right)}$. All the elements in the line h^* are pruned, except the neighbors of $\tilde{w}_{i^*h^*}$ on the same line, i.e., $\tilde{w}_{h^*,h^*-1}$, and $\tilde{w}_{h^*,h^*+1}$. Thus, the transform applies to a neighborhood around the minimum saliency weight.

Let us consider the example illustrated in Figure 3.2. The matrix $\mathbf{W}$ has a size of 7×3. Then, the matrix $\widetilde{\mathbf{W}}$, which is shown in Figure 3.2 (a), has a size of 7×7. Suppose the selected weight is w_{33}. Then, its circled neighbors belong to the tridiagonal matrix. After the multiplication to the left by the transform matrix, the weight matrix shown in Figure 3.2 (b) is obtained. Note that all the connections in the same column with w_{33}, with the exception of the neighbors w_{23} and w_{43}, are deleted. Figure 3.2 (c) illustrates the weight matrix after the result of (b) has been multiplied to the right by the inverse of the transform matrix. We here note that the connections in the same line with w_{33} are eliminated, with the exception of the neighbor w_{32}.

Now, let's assume that the minimum saliency value is obtained for an input–hidden weight outside the diagonal of the weight matrix, (i.e., $i^* \neq h^*$). Then, the connection is pruned according to the standard OBD algorithm followed by the deletion of all the weights in that column, except those of the neighbors of the diagonal element.

Finally, if the minimum OBD saliency value is obtained for a hidden–output connection and the number of output nodes is $N > 1$, the same method as that described above for the input–hidden weight matrix is applied. If the number of output nodes is equal to 1, then both the connection having minimum saliency and

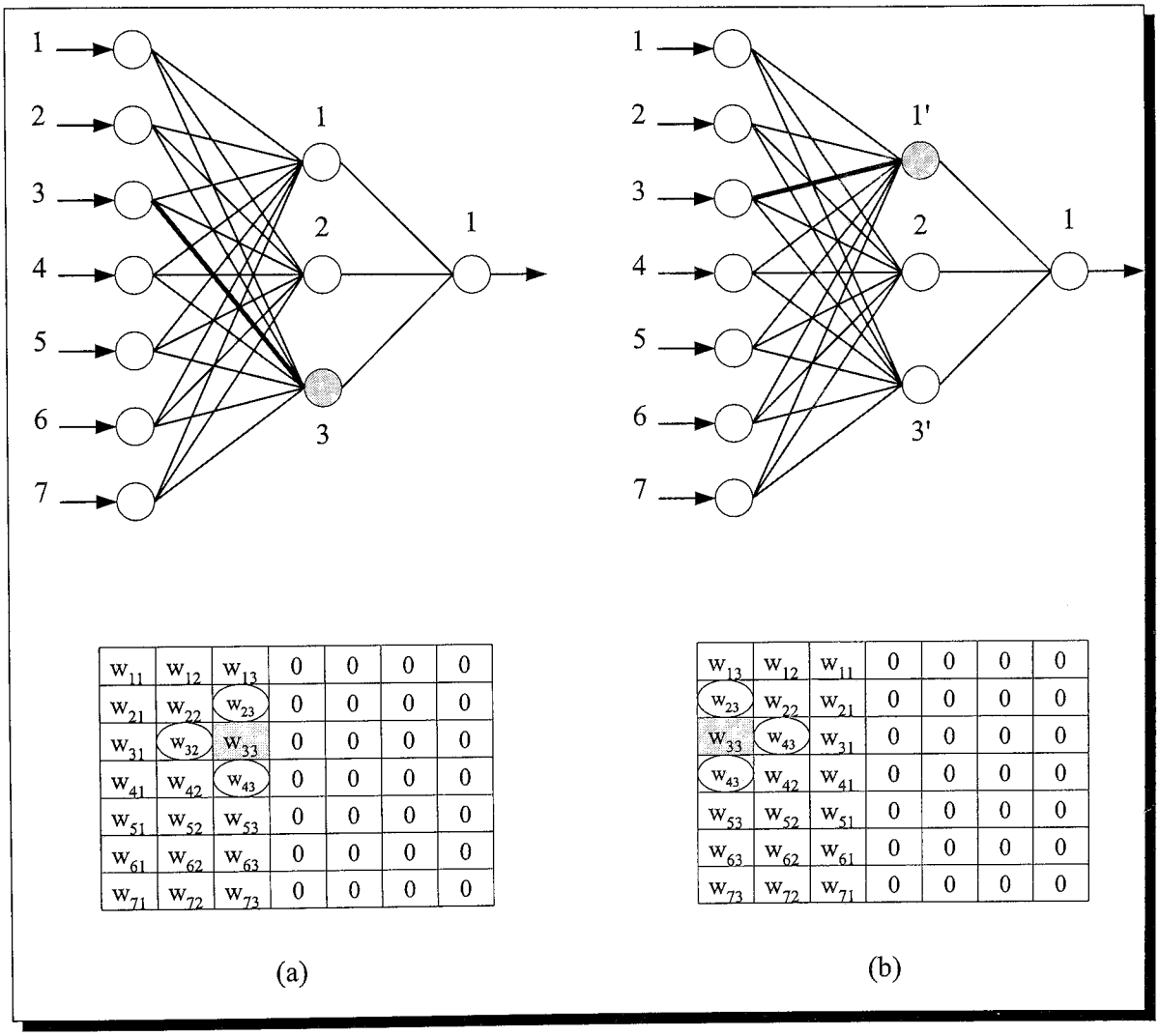
its counterpart having a symmetrical position in the network are deleted. More specifically, if the hidden–output weight v_{hj}, with $1 \leq h \leq H$ and $1 \leq j \leq N$, has the minimum saliency, then we prune v_{hj} and $v_{H-h+1,j}$ if $h = \left[\frac{H}{2}\right]$ and H is even, or v_{hj} and $v_{H-h,j}$ if $h = \left[\frac{H}{2}\right]$ and H is odd, or v_{hj} if $h = \left[\frac{H}{2}\right] + 1$ for all H. This step also preserves the symmetry of the weight matrix. A summary of the TOBD algorithm is provided in Appendix A.

3.2.2 Algorithm Discussion

We note that, several weights are deleted at the same time in the proposed TOBD algorithm. By comparison, only a single weight[2] at a time is deleted in the Optimal Brain Damage algorithm (already described in Section 2.3). The increase in the cost function δC is then expressed in the OBD algorithm as the sum of the weight saliencies given by Equation (2.7). The right term of Equation (2.7) becomes actually an approximation to the multi–weight saliency in the case of the TOBD algorithm. Next, the error due to TOBD pruning may be approximated by the sum of the saliencies corresponding to the deleted weights.

Since the numbering of the hidden nodes in a FANN is arbitrary, equivalent FANN structures are obtained if a permutation of the hidden nodes is performed. However, the tridiagonal symmetry of the structures resulting from the application of our algorithm is still preserved. Let us consider a simple example. Assume that, in the 7–3–1 FANN structure shown in Figure 3.3 (a) with its corresponding input–hidden weight matrix, the weight having the minimum saliency is w_{33}. This weight

[2]The deleted weight corresponds to the minimum of the cost function



w_{11}	w_{12}	w_{13}	0	0	0	0
w_{21}	w_{22}	w_{23}	0	0	0	0
w_{31}	w_{32}	w_{33}	0	0	0	0
w_{41}	w_{42}	w_{43}	0	0	0	0
w_{51}	w_{52}	w_{53}	0	0	0	0
w_{61}	w_{62}	w_{63}	0	0	0	0
w_{71}	w_{72}	w_{73}	0	0	0	0

w_{13}	w_{12}	w_{11}	0	0	0	0
w_{23}	w_{22}	w_{21}	0	0	0	0
w_{33}	w_{43}	w_{31}	0	0	0	0
w_{43}	w_{42}	w_{41}	0	0	0	0
w_{53}	w_{52}	w_{51}	0	0	0	0
w_{63}	w_{62}	w_{61}	0	0	0	0
w_{73}	w_{72}	w_{71}	0	0	0	0

Figure 3.3: A simple example that shows (a) the initial FANN structure and its input–hidden weight matrix below, and (b) the FANN structure after the permutation of the hidden nodes, so that node 3 becomes node $1'$, and its input–hidden weight matrix below. The shadowed boxes indicate the weight having the minimum saliency. The circled weight values indicate the neighbors of the minimum saliency weight.

corresponds to the connection indicated with a thick line. Since the weight is placed on the main diagonal, that is $i^* = h^*$, all the weights in the same line and in the same column will be deleted, with the exception of the circled neighbors w_{31}, w_{23}, and w_{43}. Assume that, by performing a simple permutation, the shadowed node 3 becomes node $1'$. The minimum saliency weight does not have a diagonal position since $i^* \neq h^*$. In this case, according to our proposed algorithm, only the minimum saliency weight w_{33} is deleted. Thus, its neighbors $w_{21'} = w_{23}$, w_{32} and $w_{41'} = w_{43}$ are still preserved in the structure. Therefore, the tridiagonal symmetry at this stage is also preserved.

3.2.3 Complexity Issues

The speed of the designed FANNs depends largely on the number of "multiply and add" operations performed in the test phase, which depends in turn on the total number Π of parameters in the neural structure. For a fully connected FANN (FC−FANN), Π is the sum of all weights and biases, and is given by $\Pi = (M + 1)\, H + (H + 1)\, N$. For the partially connected designed FANN, this expression becomes

$$\Pi = [(M + 1)\, H + (H + 1)\, N]\, (1 - \epsilon)\ , \text{where} \tag{3.3}$$

ϵ is the percentage of pruned connections. More specifically, the number of parameters in the partially connected FANN is obtained by subtracting the number of parameters corresponding to the pruned connections from the number of parameters of the FC−FANN. Although the test time is clearly shorter for FANN structures with fewer parameters, the test time as a stand–alone index is still useful for comparing different implementations of FANN structures that have the same numbers of parameters. The

test time can be evaluated based on [109]. More specifically, the necessary time in the forward pass of data through the network in the test phase can be expressed as

$$t^*_{test} = 2\,[M_{eff}\,t_t + (M_{eff} + H_{eff})\,t_0]\,P. \tag{3.4}$$

where t_t, t_0, and P are the time required by a single add or multiply operation, the transfer time between two nodes of data represented using b bits, and the number of patterns, respectively. If there is at least one hidden node connected to all the inputs, then $M_{eff} = M$. If there is at least one output node connected to all the hidden layer outputs, then $H_{eff} = H$. Otherwise, the effective values are given by the maximum number of connections entering the hidden nodes and the output nodes, respectively, where the maximum is evaluated for all the nodes in the subject layer.

The complexity of the re–training stage may be evaluated similarly, based on the observation that re–training requires both a forward and a backward pass through the entire dataset. For each of the TOBD and OBD algorithms, re–training requires $O(\Pi^3)$ operations per weight update, where Π is the number of parameters. However, the number of parameters during re–training is smaller for the TOBD than the OBD case, since several weights are eliminated during each TOBD pruning step.

3.3 Simulation Example

In what follows, we illustrate that, even when the application problem does not contain any symmetries, our TOBD algorithm reduces significantly the number of FANN parameters, so that compact, tridiagonally symmetrical structures are obtained without a significant loss in terms of performance. For this purpose, we select the nonlinear

regression problem described by $d(\xi) = F(\mathbf{w}; x(\xi)) + e(\mathbf{w}; \xi)$. Without loss of generality, we assume that the FANN multilayer perceptron used to solve the problem has one output node, i.e., $N = 1$. The function $e(\mathbf{w}; \xi)$ is the output error when the input pattern is ξ. The training data set is $\Delta = \{(x(\xi), d(\xi)), 1 \leq \xi \leq P\}$. We assume a data model described by the additive cost function

$$C(\mathbf{w}) = \frac{1}{P}\sum_{\xi=1}^{P} C(\mathbf{w};\xi) = \frac{1}{P}\sum_{\xi=1}^{P}\sum_{j=1}^{N} C(\mathbf{w};\, e_j(\xi)) \overset{N=1}{=} \frac{1}{P}\sum_{\xi=1}^{P} e^2(\mathbf{w};\, \xi) =$$

$$= \frac{1}{P}\sum_{\xi=1}^{P}\frac{1}{2}\{d(\xi) - F[\mathbf{w};\, x(\xi)]\}^2 = \frac{1}{P}\sum_{\xi=1}^{P}\frac{1}{2}\{d(\xi) - y[\mathbf{w};\, x(\xi)]\}^2. \quad (3.5)$$

3.3.1 Simulation Details

Before evaluating the results, there are important issues that need to be addressed: the data set, the neural structure, the activation function, the training algorithm and the performance evaluation criteria. The data set consists of 5760 samples of the "chirp"[3] signal and it is divided into two equal subsets, one for training and another one for testing. The training and test signals are illustrated in Figure 3.4. Each of the subsets is read through a 7–sample window, sliding at each epoch one sample to the right. The desired output value is the sample value following the window (e.g., for the first window, the desired output value is the eighth sample value). Two 7×2880 input matrices result, as well as a 1×2880 vector of desired output values. The neural structure is of the type 7–H–1, where the input and the output layers have the same size as the input and the output patterns. Any empirical relationship may be used in

[3]The chirp signal is available as a benchmark data set in Matlab. This signal is obtained by recording a real bird chirp, which has similar characteristics to a speech signal.

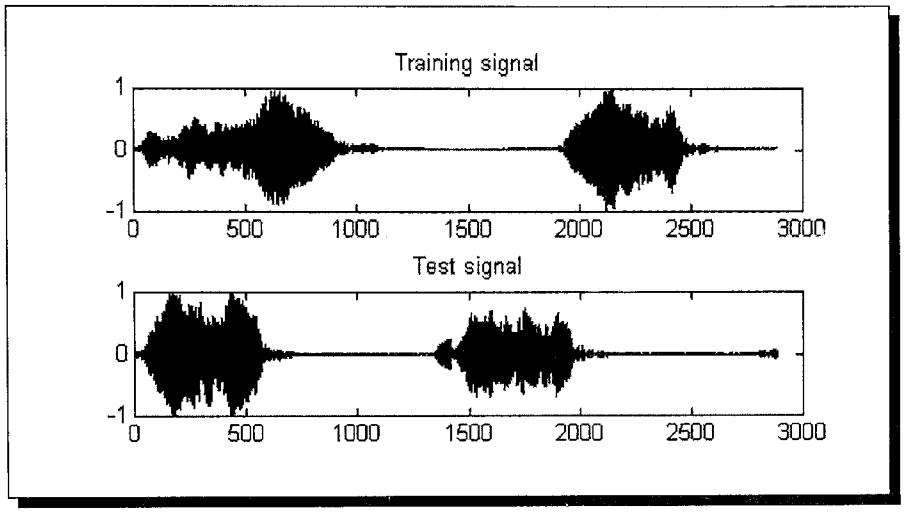


Figure 3.4: Training and test chirp signals.

order to estimate the initial size H of the hidden layer. With $P = 2880$, $M = 7$, and $N = 1$, we have selected $H = 3$ [34, 43]. The activation function f of each simple perceptron in the hidden and output layers was selected to be the sigmoidal tangent given by expression (2.2). Backpropagation with momentum [1] was selected as the training algorithm for the original fully connected FANN. The weights are adjusted based on the expression $w(k+1) = w(k) + \eta \nabla C + \alpha [w(k) - w(k-1)]$, in order to find the optimum weight vector minimizing the cost function in Equation (3.5) [22, 1, 24]. The multilayer perceptron was trained for 2000 epochs with the learning rate $\eta = 0.01$ and the momentum $\alpha = 0.0001$. The normalized training mean square error given by (3.5) was equal to 0.004456.

We evaluate the performance of the designed FANNs using the test mean square error normalized by the number of patterns (NMSE) given by (3.5), and the peak signal–to–noise ratio (PSNR) in decibels. We also evaluate the complexity of the

designed FANNs using the total number Π of parameters in the final neural structure, and the test time.

3.3.2 Simulation Results

In this section, we report on simulation results obtained using the trained FANN structure (a) without pruning, (b) after pruning using the OBD algorithm, and (c) after pruning with the TOBD algorithm, respectively. The evaluated neural structures in these cases are (a) fully connected, (b) non–symmetrical partially connected, and (c) tridiagonally symmetrical and partially connected, respectively. The NMSE and PSNR values are averaged over fifteen independent runs (trials) with identical training parameter values. The test NMSE for the fully connected and non–symmetrical structure after training was equal to 1.04526. The average number of the FANN parameters was 28 (24 weights and 4 biases).

We applied the OBD algorithm to the trained FANN. The test NMSE, the PSNR and the number of parameters in the final partially connected and non–symmetrical structure are given in Table 3.1. The final non–symmetrical structure after pruning 6 (5 input–hidden and one hidden–output) weights is presented in Figure 3.5.

We applied the TOBD algorithm to the fully connected and trained FANN. The number of parameters has been reduced until the re–training error is approximately equal to that obtained for the OBD algorithm. The test NMSEs and the number of parameters after applying the TOBD algorithm are included in Tables 3.1 and 3.2. Figure 3.6 presents a final partially connected and tridiagonally symmetrical neural

structure, where dotted lines indicate the pruned connections / weights. Note that in this case, since some of the weight values are the same, only 9 (6 weights and 3 biases) out of the 15 network parameters must be stored.

Finally, the experimental and the theoretical test times for the proposed method are given in Table 3.2. The evaluation was performed on an IBM–PC 486/66 MHz computer. The theoretical test times have been computed using Equation 3.4. The symbol t_0 denotes a simple add or multiply operation and t_t is the transfer time for a data represented with $b = 16$ bits.

Table 3.1: Training and test normalized mean square errors, and average test PSNR values before and after TOBD and OBD.

Results	Before pruning	After OBD (12 steps)	After TOBD (5 steps)
Training error	0.00445 (0.003898–0.005096)	0.00339 (0.00289–0.00390)	0.00308 (0.00251–0.00365)
Test error	1.04526 (1.04379 – 1.04673)	0.98223 (0.94473–1.01973)	1.07343 (1.04443–1.10243)
Average test PSNR [dB]	67.9976	68.2971	65.7976

3.3.3 Comparisons

The aim of the proposed design algorithm is to obtain a good tradeoff between the size of the FANN structure, the generalization performance, and the computational efficiency. Introducing constraints in the design process is generally expected to increase the test error. In our simulations, the test error increases by an average of 0.028 for the neural structure resulting from the application of the TOBD algorithm. However, this happens for an almost three–fold reduction in the number of FANN

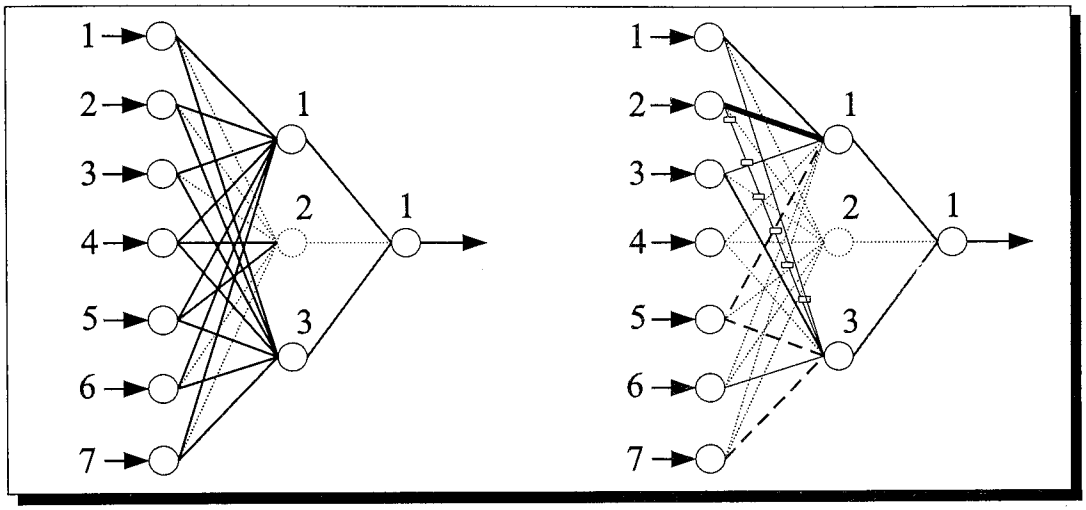


Figure 3.5: A neural structure given by OBD. Dotted lines indicate deleted connections/nodes.

Figure 3.6: A neural structure given by TOBD. Dotted lines indicate the deleted connections/nodes. Connections with the same weights have been drawn with the same lines.

parameters (weights and biases). We note that the error after more than 10,000 re–training epochs is still lower than that of the fully connected trained network. On the other hand, the test error for the OBD algorithm slightly decreases after pruning, which confirms its better generalization ability. However, only a 1.5–fold reduction in the number of FANN parameters has been obtained.

Finally, with respect to the test time, if the TOBD is selected as the reference, then the OBD network and the initial fully connected FANN require $5760\,(2t_t + 2t_0)$, and $5760\,(2t_t + 3t_0)$ more time units, respectively, based on the results in Table 3.2.

Table 3.2: Number of parameters (weights and biases) and test times for neural structures given by TOBD and OBD. The test time corresponds to the FANNs having the average number of parameters.

FANN given by	No. of FANN parameters	Average test time [sec]	
		Theoretical	Experim.
TOBD	11 (8 – 15)	5760 $(5t_t + 7t_0)$	15
OBD	22 (20 – 24)	5760 $(7t_t + 9t_0)$	25
Fully connected	28 (28 – 28)	5760 $(7t_t + 10t_0)$	42

3.4 Summary

In this chapter, we have addressed FANN design with topological symmetry constraints. More specifically, we have proposed a tridiagonally symmetrical pruning algorithm for feedforward neural network design. Our algorithm is based on a Householder transform of the input–hidden weight matrix. We have shown, via a simple simulation example, that this results in compact tridiagonal FANNs, which are suitable for efficient hardware and software implementations. Moreover, the number of FANN parameters is reduced substantially without a significant loss in performance.

Chapter 4

Application of FANNs to First–Order Image Subsampling (FOS)

In this chapter, we apply feedforward neural networks to first–order image subsampling (FOS). By addressing several problems in the case of FANN–based first–order subsampling, the material included in this chapter paves the way to applying our designed FANN models to high–order image subsampling, which is discussed in the next chapter.

In Section 4.1, we first state the main limitations of most image subsampling methods, thus providing a motivation for FANN–based image subsampling. Second, we state the problems that arise when performing FANN–based image subsampling using standard FANN training algorithms. This motivates our first–order FANN–based subsampling (training) algorithm (FABS), which is described in Section 4.2.1. We next comment on the relationship of the proposed method with other methods, as well as discuss complexity issues. In Section 4.3, we focus on the application of our

algorithm when larger input blocks are employed in the subsampling process. In Section 4.4, we present experimental results using still images and video which illustrate the good performance of FABS and generalized FABS algorithms. In Section 4.5, we present the application of the FABS algorithm to subsampling of chrominance video frames and compare the performance of our system with that employing traditional lowpass filtering and subsampling. In Sections 4.7.1 and 4.7.2, we comment on the FANN generalization ability, and speed and memory requirements, respectively. A summary of the chapter is included in Section 4.8.

4.1 Motivation

Our goal is to obtain a good subsampled version of the original image, such that if reconstructed, it is as close as possible to the original. To achieve our goal, one can use standard lowpass filtering followed by (e.g., first–order) subsampling, as illustrated in Figure 4.1. However, as stated in Section 1, when lowpass filtering is being applied to the input image, most of the high frequency information is permanently lost. Moreover, due to most of the existing subsampling methods being based on pixel neighborhood operations [19], as already stated in Chapter 2, the reduced images may often contain significant distortion, usually expressed in terms of visible blockiness in continuous features of the image [19, 6]. We aim at reducing both the information loss introduced by lowpass filtering, as well as blockiness, without applying post–processing methods.

A solution to address the above problem is to apply FANN models to image subsampling, as illustrated in Figure 4.2. This is mainly motivated by the FANN's

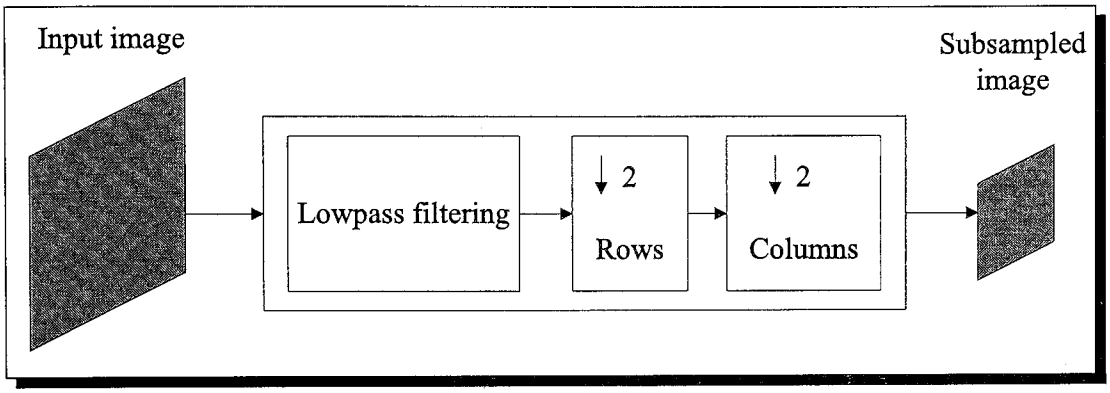


Figure 4.1: Block diagram of a conventional first–order image subsampling system.

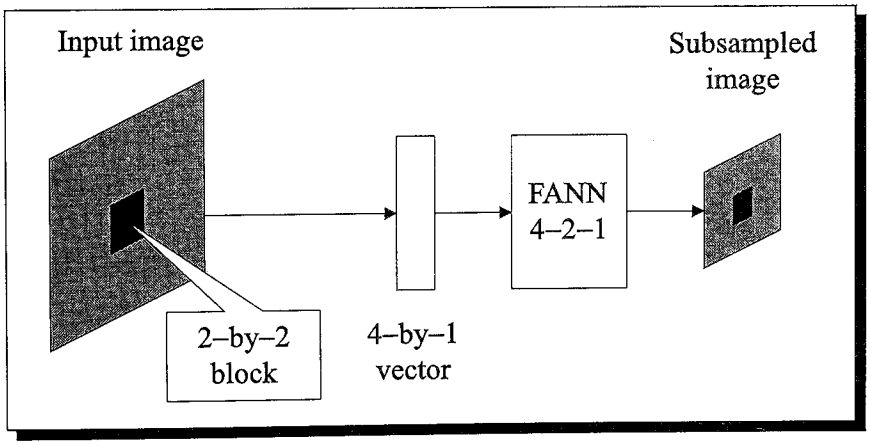


Figure 4.2: Block diagram of a feedforward neural network–based first–order image subsampling system.

ability to inherently subsample input images for certain sizes of the input, hidden and output layers. This is also motivated by FANN's ability to perform high speed parallel processing. Unfortunately, the performance of the FANNs in image subsampling has not been acceptable so far, as the reproduced images exhibit blocking and/or ringing artifacts when the standard FANN training algorithms are being used. In standard supervised algorithms that solve the optimization problem (2.3) stated in Chapter 2, the desired output value $d(\xi)$ is selected, for each ξ, prior to the training process. For instance, one can choose it as the pixel value in the center of the input window. However, this approach does not take into account the local characteristics of the pixel neighborhood. Thus, the FANN receives during training only information regarding the pixel gray values and therefore, cannot learn geometrical structures in the current input window. The edges and other continuous features in the image, obtained during the testing step, are therefore significantly affected. In what follows, in order to improve the quality of the FANN subsampled and reconstructed image, and mainly to reduce blockiness, while still maintaining relatively low complexity, we propose a supervised strategy for solving the training/optimization problem. The proposed FANN–based subsampling (FABS) algorithm is based on pattern matching and takes into account the local characteristics of the input window. Thus, the image artifacts that typically result from image subsampling can be minimized or eliminated.

4.2 FOS Using 2×2 Input Blocks

For simplicity, let us focus on first–order spatial subsampling using a fixed rectangular subsampling grid. Let the two–dimensional image be stored as a matrix, each matrix

element $x_{m,n}$ representing the gray level of the pixel in row m and column n. Moreover, let the FANN model be an M–H–N model with M input, H hidden and N output nodes, as illustrated in Figure 2.6. During the training step, the FANN receives input data through a $W \times W$ sliding window, unwraps the resulting matrix as illustrated in Figure 4.2 to become an $M \times 1$ input vector $\mathbf{x}$ with components x_i. Without loss of generality, let the size of the sliding window be 2×2 and let N be equal to 1 (one output node). This allows us to address first–order image subsampling as illustrated in Figure 2.11 (a). The size of the hidden layer may be quickly evaluated, once the number of patterns and the sizes of the input and output layers are determined.

4.2.1 Proposed FANN–Based Subsampling (FABS) Algorithm

The proposed FANN–based subsampling (training) algorithm can be divided into six steps. First, we compute the actual output value $y(\xi)$ given by

$$y(\xi) = f\left(\sum_{h=1}^{H} w_{hj}\, f\left(\sum_{i=1}^{M} w_{ih}\, x_i(\xi)\right)\right), \tag{4.1}$$

for each input window (pattern) pattern ξ, where $1 \leq \xi \leq P$ and P is the maximum number of patterns. The notations w_{ih} and w_{hj} denote the input–hidden and hidden–output weights, respectively, and $j = 1$. The initial weight values are randomly selected, f is the unipolar sigmoidal activation function given by expression (2.1).

Second, we compute the median of all possible three–pixel combinations, as illustrated in Figure 4.3, and compare it to the value of the fourth pixel in the window, yielding the values,

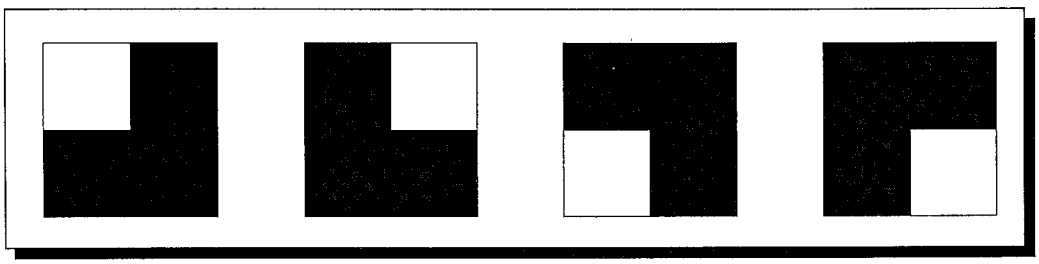


Figure 4.3: Shapes taken into account in the pattern–matching algorithm.

$$q^{(1)}(\xi) = \{x_{m,n}(\xi) - med\,[x_{m,n-1}(\xi),\ x_{m+1,n-1}(\xi),\ x_{m+1,n}(\xi)]\}^2,$$

$$q^{(2)}(\xi) = \{x_{m,n-1}(\xi) - med\,[x_{m,n}(\xi),\ x_{m+1,n-1}(\xi),\ x_{m+1,n}(\xi)]\}^2, \quad (4.2)$$

$$q^{(3)}(\xi) = \{x_{m+1,n-1}(\xi) - med\,[x_{m,n-1}(\xi),\ x_{m,n}(\xi),\ x_{m+1,n}(\xi)]\}^2, \text{ and}$$

$$q^{(4)}(\xi) = \{x_{m+1,n}(\xi) - med\,[x_{m,n-1}(\xi),\ x_{m,n}(\xi),\ x_{m+1,n-1}(\xi)]\}^2, \text{ where}$$

med stands for the median operator. Third, the minimum $q^*(\xi)$ of the above $q^{(r)}(\xi)$'s, $1 \leq r \leq 4$, for the current input window is obtained, i.e.,

$$q^*(\xi) = \min_{1 \leq l \leq 4} \left\{q^{(l)}(\xi)\right\}. \quad (4.3)$$

Fourth, the desired output value is set to $d(\xi) = x^{(l^*)}(\xi)$, where l^* is the value of l for which the minimum in (4.3) is reached. The function $x^{(l^*)}(\xi)$ is equal to $x_{m,n}(\xi)$, $x_{m,n-1}(\xi)$, $x_{m+1,n-1}(\xi)$ or $x_{m+1,n}(\xi)$ if $l^* = 1, 2, 3$ or 4, respectively. Fifth, the global error $C(\mathbf{w})$ at the end of one epoch[1] is computed by adding the

[1] An epoch is defined as one pass through the P–dimensional set of training patterns.

squared errors $e(\xi)$ for all the input patterns (unwrapped input windows), that is,

$$C(\mathbf{w}) = \frac{1}{P}\sum_{\xi=1}^{P} C(\mathbf{w};\xi) = \frac{1}{2P}\sum_{\xi=1}^{P}[d(\xi) - y(\xi)]^2 .$$

Finally, the weights are adjusted according to a quasi–Newton rule given by,

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \eta_k \mathbf{d}_k = \mathbf{w}_k - \eta_k \hat{\mathbf{H}}_k^{-1} \nabla C(\mathbf{w}_k),$$

where $\mathbf{w}_k$ and $\mathbf{w}_{k+1}$ are the weight vectors at epochs k and $k+1$ (respectively), $\mathbf{d}_k$ is the search direction of the minimum in the parameter space; $\hat{\mathbf{H}}_k^{-1}$ is the approximation of the inverse Hessian matrix $\mathbf{H}^{-1}$ and $\mathbf{H}(\mathbf{w}_k) = \nabla^2 C(\mathbf{w}_k)$. We have used the Levenberg–Marquardt approximation of $\hat{\mathbf{H}}^{-1}$ and a learning rate η_k given by expression (2.4) [22, 24].

One can easily notice that the optimization problem (2.3) becomes in our approach the minimization of

$$C(\mathbf{w}) = \frac{1}{2}\sum_{\xi=1}^{P}[d(\xi) - y(\xi)]^2 \text{ subject to} \tag{4.4}$$

$$q(\xi) = \min_{1 \le l \le 4} \left\{ q^{(l)}(\xi) \right\} \text{ and } \mathbf{w} \in \Re^{\Pi}.$$

The above algorithm steps are repeated until our goal is reached, i.e. $C \le C_{desired}$ or until a predefined number of epochs is exceeded. The network parameters obtained at the end of the training process are saved and used during the testing phase. The proposed FABS algorithm is summarized in Appendix B.

4.2.2 Examples

The main idea of the first four steps is that, one pixel in the current neighborhood (out of four candidates) may become the desired output value, thus being the representative of the entire window, if and only if its value is close enough to the median of the other three pixels. Basically, the three–pixel combinations shown in Figure 4.3 make available information regarding the presence of local edges. Consequently, a better FANN behavior is expected, realized in terms of fewer blocking artifacts and a generally better image reproduction quality, as compared to conventional FANN algorithms.

We note that, since the FANN output is a single pixel value, the actual edge orientation cannot be preserved. Instead, this edge information is encoded in the gray level of the FANN desired output selected by our algorithm. The gray level of the FANN output pixel is perceived as being subjectively close to that of the original input block when viewed from a distance. Thus, blockiness is masked due to the integration operation performed by the human eye. We can illustrate this by using two simple examples. In the first example, we employ 2×2 blocks having vertical, horizontal and diagonal edges, pixel discontinuities, uniform gray levels and corners, respectively. These blocks are shown in the left column of Figure 4.4. There are 6 two–pixel combinations representing straight lines and diagonal edge structures, 4 one–pixel combinations representing image discontinuities, 4 three–pixel combinations representing the corners and one four–pixel combination representing a smooth block. In Figure 4.4, we have shown only 10 out of the total 15 possible combinations. For each of the blocks shown in the left column, we have represented the mean, the

median and the FANN desired output selected by our algorithm by using gray values. When the figure is viewed from a distance, the original block is perceived as a single dot. Moreover, the perceived dot is closer to the FANN desired output than to the mean or the median values of the block, respectively. In most of the cases shown in Figure 4.4, the FANN desired output value is different than the mean and the median values of the original block. The only exceptions occur when pixel discontinuities are present (one pixel per block), or when the input block is smooth. In these cases, no geometric details are present in the input block. Thus, the FANN desired output is very close to the mean and the median values[2]. Note here that the one–pixel and two–pixel combinations are subclasses of the three–pixel combinations used in our algorithm. As the FANN desired output values are perceived similarly, we did not need to include the two–pixel combinations during the FANN training process. The one–pixel combinations represent discontinuities and are eliminated by subsampling. Therefore, they have not been included in the training set as well. Finally, the four–pixel combination, which represents a smooth block, was also not included in the training set. The FANN would not benefit by learning a smooth block, as the latter does not contain specific geometric details.

To further demonstrate the effectiveness of our algorithm, a second example is provided in Figure 4.5. The original block is a 16×16 block whose center coincides with the geometric center of the 256×256 image Lena. The mean and median values

[2]In this particular example, the mean and the median values for each block are equal. More specifically, their values are set to 105 (blocks 1–4 in the left column), 93.75 (block 5), 102.25 (block 6) and 81.25 (blocks 7-10), respectively. Due to the odd size of the original block, the median is computed as the average of the two central values of the ordered values in the block. Therefore, an additional lowpass filtering stage is accidentally introduced when computing the median. Thus, the performance of the median filter is not better than the simple average filter.

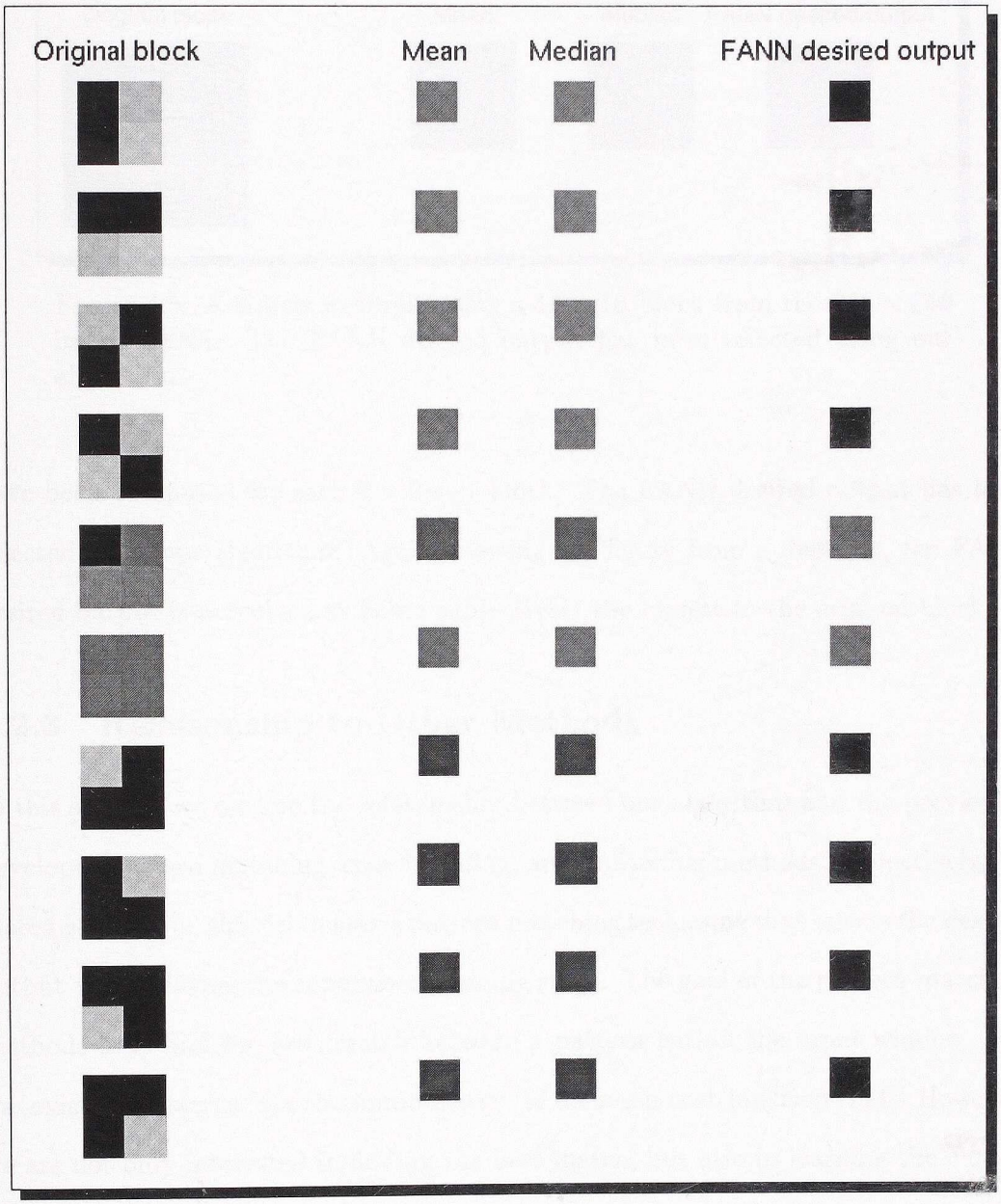


Figure 4.4: A simple example using 2×2 blocks. The FANN desired output values have been selected using our algorithm.

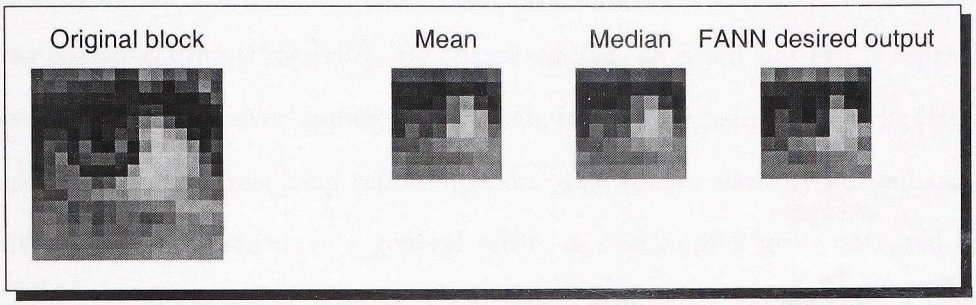


Figure 4.5: A simple example using a 16×16 block from the 256×256 image LENA. The FANN desired output has been selected using our algorithm.

have been computed for each 2×2 sub–block. The FANN desired output has been selected using our algorithm. Again, viewing the figure from a distance, the FANN desired output is perceived as being subjectively the closest to the original block.

4.2.3 Relationship to Other Methods

In this section, we outline the relationship between our algorithm and the previously developed pattern matching, coarse–coding, and halftoning methods, respectively. As stated earlier, our algorithm uses a pattern matching technique that selects the desired output values during the supervised training stage. The goal of the pattern matching methods is to find the best match between a pattern within the input window, and the existing patterns, via the minimization of a chosen cost function [124]. However, we are not only interested in finding the best match, but also in learning the output gray level corresponding to the best match.

In our first–order FANN–based subsampling algorithm, each input window is represented by a single pixel value in the subsampled image. The idea of processing

an input image by using local windows has been addressed by other methods as well, such as coarse–coding [125]–[127]. By coarse–coding, an input pattern is represented by several coarser and overlapping grids. Then, high–order neural networks (HONNs) are used to learn the resulting representation. The shapes used by our subsampling algorithm can be regarded, in a general sense, as overlapping grids obtained by the decomposition of the input window. They would resemble a coarse–coding representation with the same coarseness level as the original image. However, it is not clear how the coarse–coding method, previously applied to black and white edge images [125]–[128], could be generalized for gray level images, such as those used in our subsampling experiments in Section 4.4. Note that a given gray level input window can be decomposed into coarse grids depending not only on the shape of the objects, but also on the gray levels of the pixels. Moreover, the HONN models, due to their large number of connections, require significant time and memory resources, as compared to the FANN model employed in our method.

Finally, the idea of representing a local neighborhood in the input image by a gray–level pixel value in the final image may resemble, in an abstract sense, work in the area of halftoning [129, 130]. However, the number of the image gray levels and the sizes of the images resulted by applying halftoning methods and our subsampling method, respectively, are different. Moreover, although ANNs have been applied extensively in image halftoning, generally recurrent models, such as cellular neural networks (CNNs) [131], Hopfield networks [132, 133] and Q'trons (generalizations of the Hopfield networks) [134] have been employed. For these models, image halftoning has been formulated as an optimization problem, and a neuron has been associated

with each pixel of the halftoned image [129, 134, 135]. Applying such a method for image subsampling would be highly impractical when using large images.

4.2.4 Implementation Issues

Before the FABS algorithm can be implemented, several issues must be addressed. For example, the type of the input window, the intensities of the input gray levels, the number of training epochs and the objective/subjective performance measures are all parameters that have to be evaluated. The input images are read through a $W \times W$ sliding window of size 2×2, moving one pixel to the right. Both non–overlapping and overlapping square windows have been used during experiments. As the latter did not significantly improve performance relative to the former, we have decided to use non–overlapping windows.

The intensities of the input gray levels, in the range $[0, 255]$, are here mapped by the FANN sigmoidal activation function to the interval $(0, 1)$. One can either scale the sigmoidal function to take values in the range $[0 \ldots 255]$, or restrict the desired output values to be less than one by dividing each desired output pixel value in the original image by 255. We have chosen the latter solution. Moreover, since the activation function reaches only asymptotically the values 0 and 1 given a finite input, and since the argument of the sigmoidal function is a finite sum of finite values, the pixel values at the extremes of the brightness range will never be learned by the network satisfactorily [13]. This problem will be addressed in more detail in Section 4.4.

Another implementation issue is the selection of the number of training epochs. The number of necessary training epochs to achieve good performance levels on test images may be estimated. We have set the number of epochs to 1000 based on the results discussed in [22].

Finally, various objective evaluation criteria may be used to assess the performance of the proposed algorithm. Of course, what really matters for the end user is the subjective quality [123]. Unfortunately, the latter is not always related to measures such as the popular peak signal–to–noise ratio (PSNR). Therefore, we have chosen to evaluate the results by: (a) visual examination of the subsampled images [21], (b) visual examination of the reconstructed (bilinear or cubic interpolated) images from the subsampled versions, and (c) objective evaluation of the interpolated images from the subsampled versions.

4.3 FOS Using Larger Input Blocks

4.3.1 Generalized FABS Algorithm: Fixed Threshold

Let us address in what follows first–order image subsampling using larger input blocks, which is illustrated in Figure 2.11 (b). Without loss of generality, let the size of the input block be 4×4. The FANN output consists of four pixels. As in the case of 2×2 blocks, the subsampling rate is $4 : 1$. We have divided each 4×4 window into four 2×2 blocks. For each 2×2 block, we then evaluated the standard deviation. If the standard deviation is higher than a threshold β, the presence of an edge is detected in the input block. Then, the 4–2–1 FANN, trained using our FABS algorithm described

earlier, is applied in order to obtain the output pixel value. If the standard deviation of the block is smaller than the threshold β, a smooth block is detected. The output pixel value is then set to the average of the four pixels in the 2×2 block.

The value of the threshold has been set to 43. This selection is based on our experimental observation that an edge in the 2×2 input block is likely to be perceived if at least two pixel differences are higher than 75 on a 0–255 scale. For example, for an input block with the pixel values set to $x_1 = 75$, $x_2 = 0$, $x_3 = 75$, and $x_4 = 0$, the standard deviation of the vector obtained by unwrapping the input block is equal to approximately 43. Using this method, we are able to determine whether an edge of any orientation[3] is present in the input block.

4.4 Experimental Results: FABS Algorithm

In this section we present the values of the parameters, as well as the training and the testing sets that have been used in our experiments. The FANN multilayer perceptron is trained using the quasi–Newton algorithm with the Levenberg–Marquardt approximation [22, 24]. In the FABS algorithm, the neural structure has the structure 4–2–1 ($M = 4, H = 2, N = 1$). The nodes of the neural structure have sigmoidal activation functions with slope equal to 0.5. For the still image experiments, the training set consists of the 256×256 image LENA, and the test set includes the 8–bit 256×256 image TEETH, 512×512 images MANDRILL, BOAT and FIGHTER, and 1200×1524 JPEG–2000 test image TOOLS (industrial objects), and a 456×532 section of the stan-

[3]By changing the orientation of the edge, i.e., after the permutation of the pixels inside the input block, the value of the standard deviation does not change.

dard JBIG bi–level test image F08–200 (black capital letters on white background). For experiments using video frames, the training set consists of the standard QCIF[4] video sequences CLAIRE, GRANDMA, SALESMAN, MISS AMERICA, SUZIE and the test set includes MOTHER–AND–DAUGHTER, TREVOR and CARPHONE. In both the still image and video cases, the test set has mostly different characteristics than the training set, allowing an accurate evaluation of the FANN generalization capabilities. The test images consist of mainly texture (TEETH), sharp edges in all orientations (BOAT, FIGHTER, F08–200, and TOOLS) and a busy image (MANDRILL). The video sequences used for testing have also different characteristics, although they all consist of head–and–shoulders frames. In the FABS algorithm, each input image or video frame is read through a 2×2 non–overlapping window, sliding over the entire image.

We will compare the experimental results obtained by FANN first–order subsampling with those obtained by lowpass filtering followed by subsampling, using three different 2–D filters: LPF1, which was designed via frequency sampling, LPF2, which is a separable finite impulse response (FIR) filter designed using separable 2–D windows, and LPF3, which is a nonseparable FIR filter designed using 2–D windows. All of the filters have an order of 11 and a cutoff frequency equal to $0.5\ \pi$. The notations LPF1S, LPF2S and LPF3S will refer to lowpass filtering using the filters LPF1, LPF2 and LPF3, respectively, followed by subsampling. The notations LPF1S+int., LPF2S+int. and LPF3S+int. will refer to lowpass filtering using these filters followed by subsampling and interpolation. The latter is cubic interpolation, if not specified otherwise.

[4]In the standard QCIF (Quarter Common Intermediate Format) video format, all luminance frames Y are of size 144×176 and all chrominance U and V frames are typically of size 72×88.

4.4.1 Subsampling of Still Images

The FANN was trained using the FABS algorithm and the image LENA. The PSNR values (Tables 4.1 and 4.2) using all the test images are better, by an average of 1.61 dB, for the FANN subsampled (FANNS) and interpolated (bilinear in Table 4.1, cubic in Table 4.2) images than for the best LPF subsampled (LPFS) and interpolated images. As expected, cubic interpolation yields better results than those of bilinear interpolation. The average improvement with respect to LPF1 subsampling followed by bilinear interpolation (see Table 4.1) is equal to 0.5025 dB. The average improvement for each image with respect to the three filters is equal to 1.5464 dB. The average improvement with respect to LPF1 subsampling and cubic interpolation in Table 4.2 is equal to 0.3527 dB. The average improvement with respect to the three filters is equal to 1.621 dB. The average improvement with respect to LPF1 subsampling and interpolation for Tables 4.1 and 4.2 is equal to 0.4276 dB. Finally, the average improvement with respect to the three filters for both Tables 4.1 and 4.2 is equal to 1.5837 dB.

Not only do the objective results show better FANN performance, but also a good visual quality can be observed, as illustrated in Figure 4.6. To better interpret the quality of the results, 16×16 blocks within the original and interpolated images LENA and BOAT, and a 512×512 block of the image TOOLS have been magnified. The results are illustrated in Figures 4.7, 4.8, and 4.9, respectively. As expected, all of the FANNS images have a sharper appearance as compared to the LPFS images. Assuming that the initial image has an equally distributed histogram, this can be explained by an increase of the histogram density at the tails and a decrease in the

Table 4.1: Test PSNR [dB] using still images, when the FANN was trained to subsample the 256 × 256 image LENA. The acronym *int.* denotes bilinear interpolation.

Image	FANNS+int.	LPF1S+int.	LPF2S+int.	LPF3S+int.
Teeth (256 × 256)	29.148	28.288	25.373	24.196
Mandrill (512 × 512)	22.873	22.547	21.677	21.207
Boat (512 × 512)	23.463	23.069	22.834	22.662
Fighter (512 × 512)	25.793	25.363	24.798	23.260

Table 4.2: Test PSNR [dB] using still images, when the FANN was trained to subsample the 256 × 256 image LENA. The acronym *int.* denotes cubic interpolation.

Image	FANNS+int.	LPF1S+int.	LPF2S+int.	LPF3S+int.
Teeth (256 × 256)	30.570	29.927	26.400	25.043
Mandrill (512 × 512)	23.246	23.003	22.113	21.607
Boat (512 × 512)	24.838	24.462	23.424	23.158
Fighter (512 × 512)	26.279	26.130	25.446	23.733

middle, due to using the sigmoidal activation function. The histograms of the original, LPFS, FANNS and cubic interpolated images BOAT (see Figure 4.6), are illustrated in Figure 4.10. The histogram changes are more visible in the rightmost peak and in the middle of the histogram corresponding to the FANN subsampled and interpolated image. The histogram of the FANN subsampled image also shows that, due to the saturation of the sigmoidal function, the FANN does not learn the gray values close to 0 or 1.

In order to determine whether the sharper appearance is due to the preservation of edges, we have also computed the histogram of the difference images, between the original image and the FANN/LPF subsampled and cubic interpolated images, respectively. These histograms are shown in the rightmost column of Figure 4.10. The

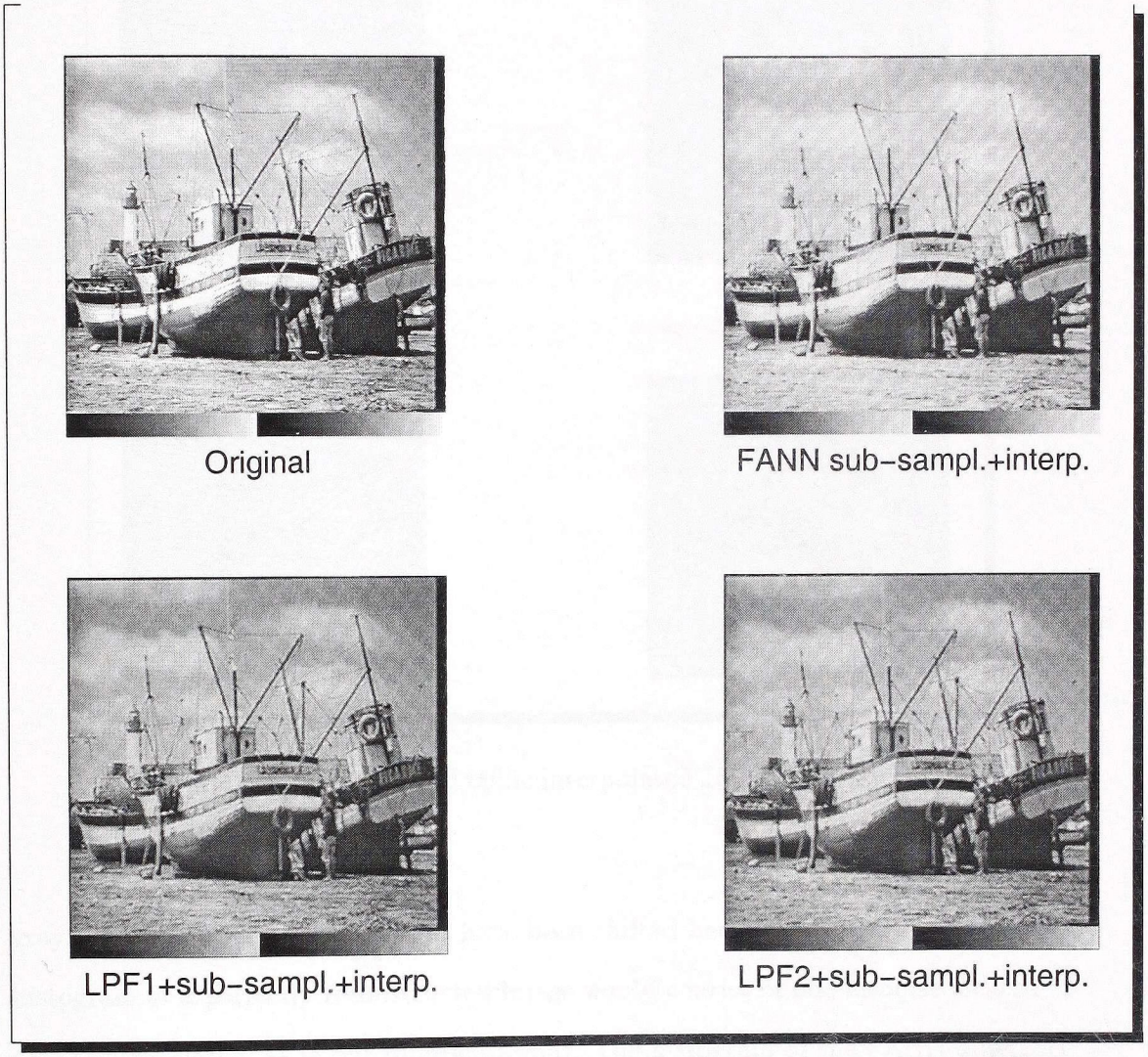


Figure 4.6: Subsampled and cubic interpolated 512×512 image BOAT. The FANN was trained on the 256×256 image LENA using a 2×2 non–overlapping window.

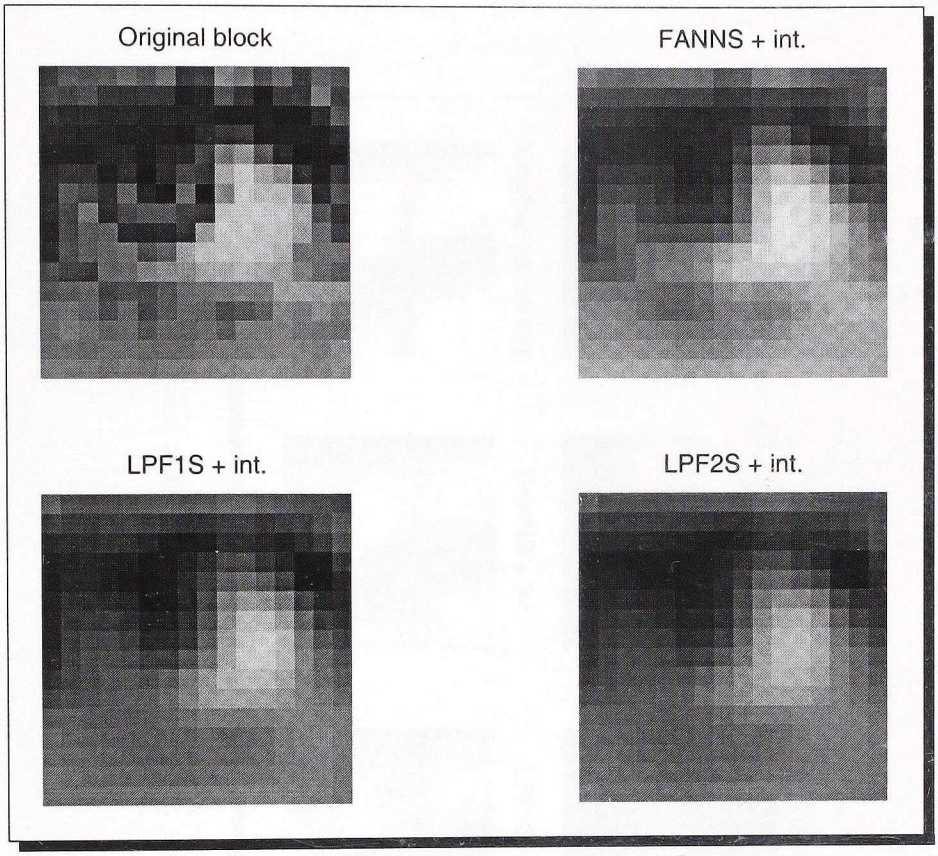


Figure 4.7: Subsampled and cubic interpolated 16×16 block from the 256×256 image LENA.

gray levels in the difference images have been shifted back to the $[0, 255]$ range. The histogram of a perfectly reconstructed image would consist of one impulse located at zero (or translated, as in our representation). The histogram of the FANN difference image is clearly the closest to the ideal histogram. Moreover, Figure 4.11 illustrates the histogram modification as a result of FANN subsampling and cubic interpolation, in a 64×64 block in the 512×512 BOAT image, confirming that the histogram has

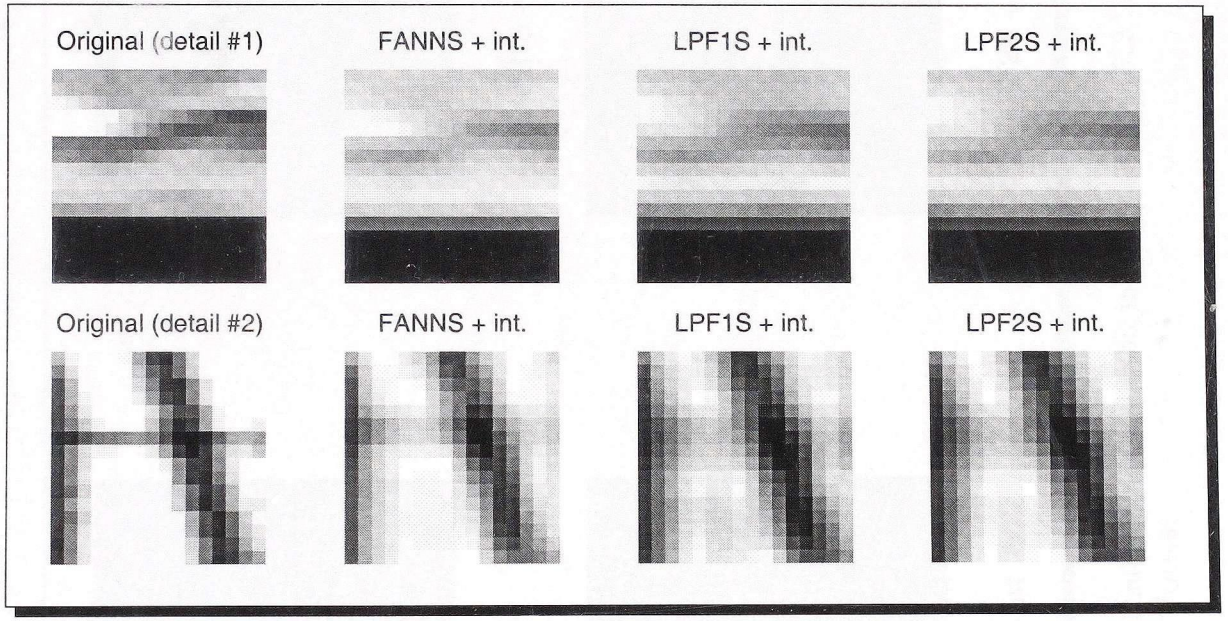


Figure 4.8: Subsampled and cubic interpolated 16×16 block from the 512×512 image BOAT.

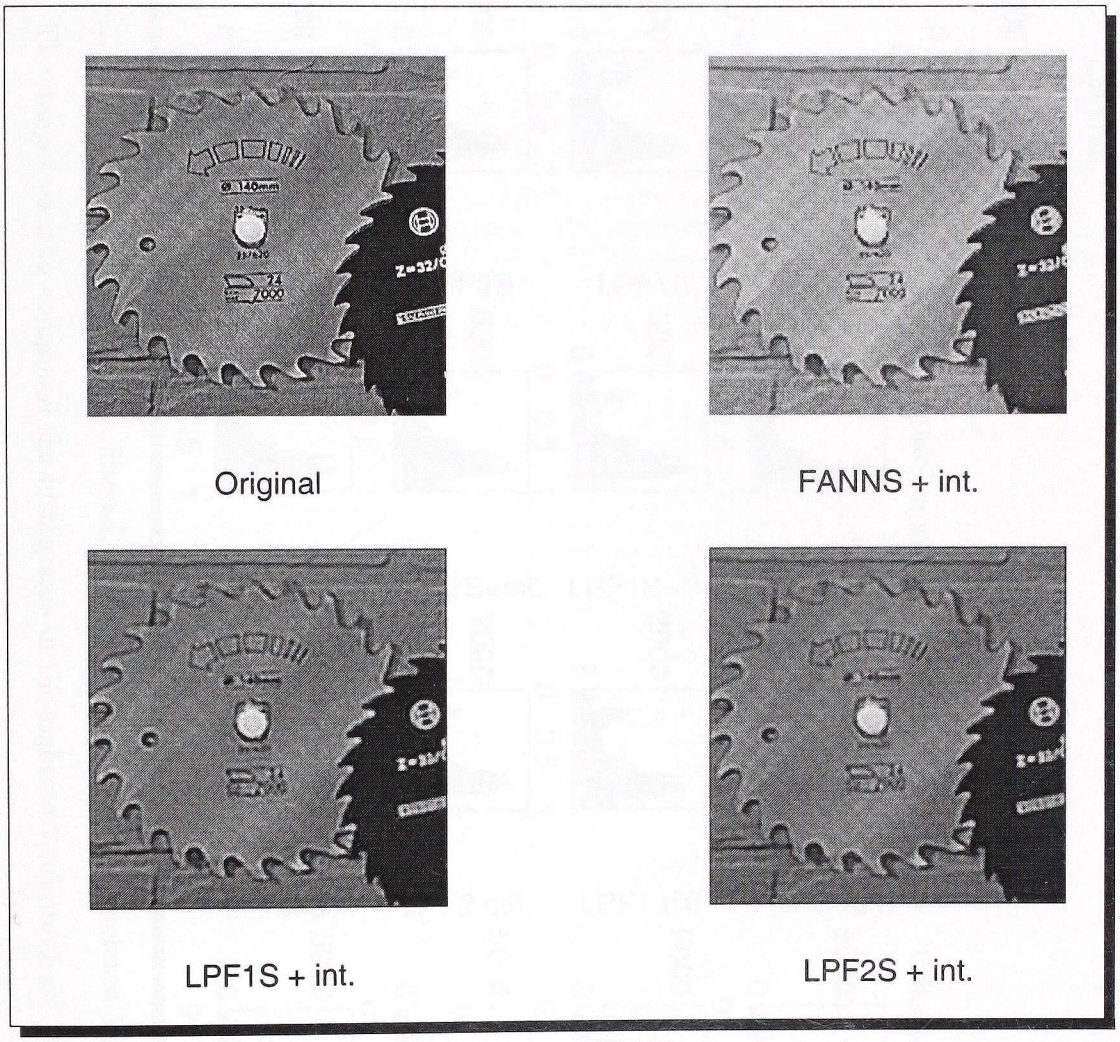


Figure 4.9: Original, subsampled and cubic interpolated 512×512 block of the 1200×1524 image TOOLS.

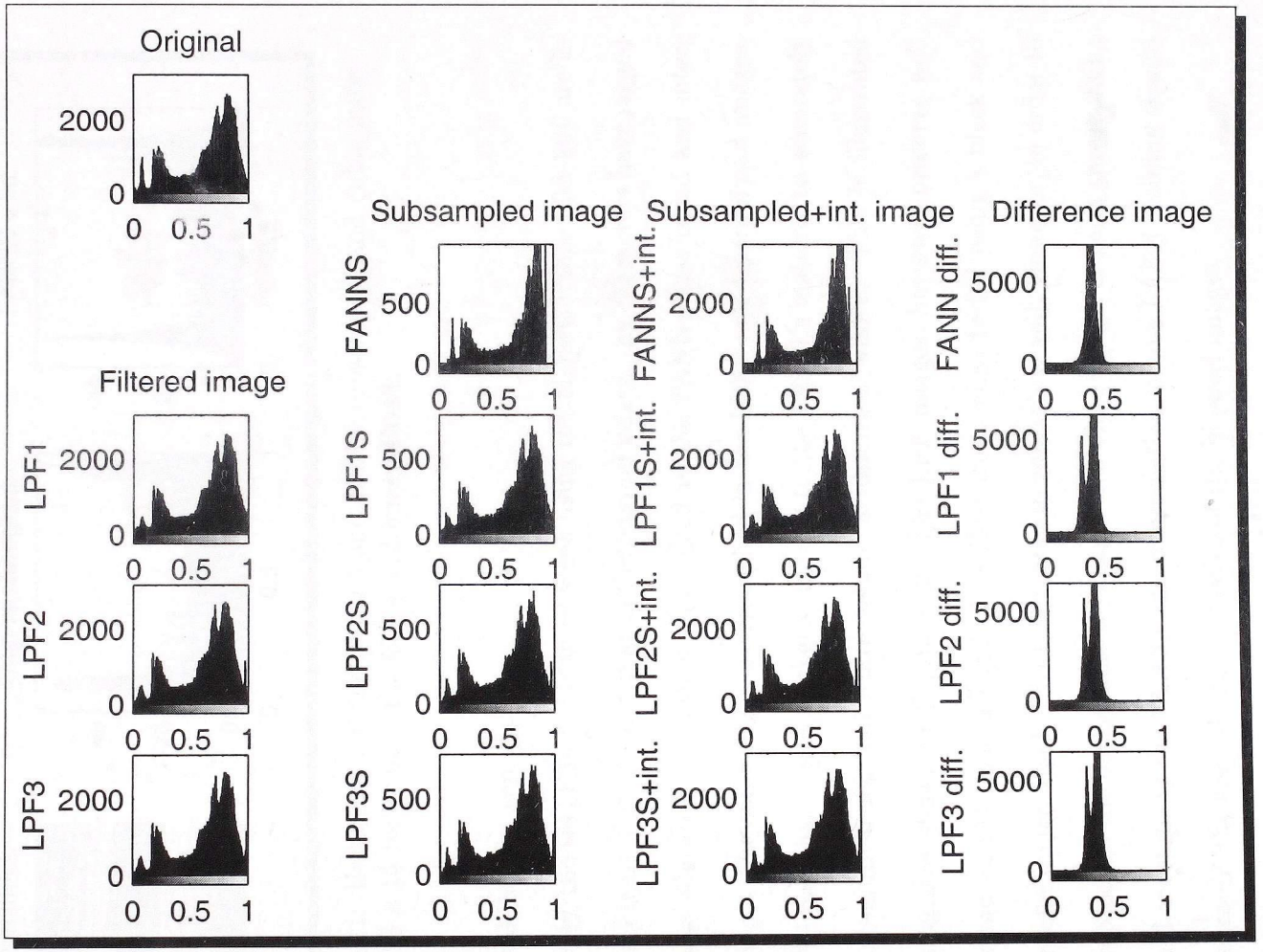


Figure 4.10: Histograms of the original, filtered, subsampled and cubic interpolated 512×512 image BOAT. The images used to computed the histograms in the rightmost column have been computed as the difference between the original image and the subsampled and cubic interpolated images.

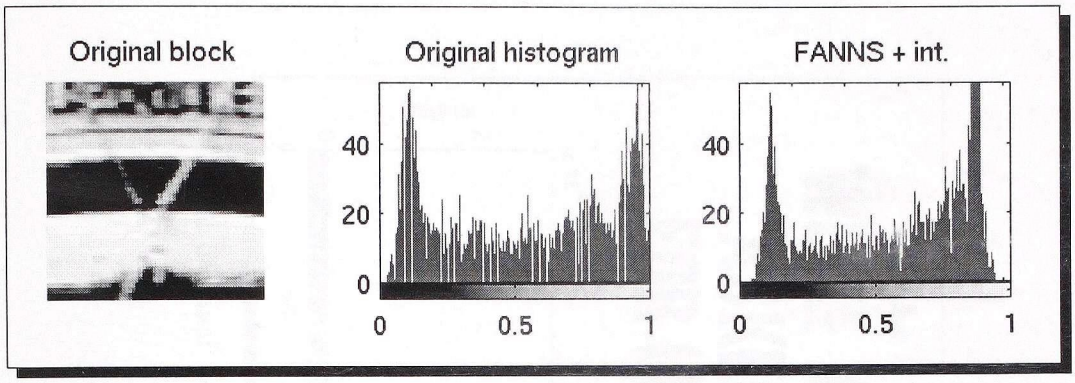


Figure 4.11: Histograms of the original and FANN subsampled and cubic interpolated 16 × 16 block from the 512 × 512 image BOAT.

changed both locally and globally.

Finally, the test results obtained when using the bi–level image F08-200 are illustrated in Figure 4.12. Notice that in the original image, the characters have jagged edges. These edges are accurately reproduced in the FANN subsampled and interpolated image. The same edges are smoothed in the LPFS and interpolated images. The FANN image has a gray background, due to the FANN's inability to accurately learn the absolute black and white values, as discussed above. This is illustrated by the histograms shown in Figure 4.12. The LPF images, however, preserve well the foreground and the background. It is clear that, when tested using a black and white character image, the FANN is still able to generalize well. However, in order to correctly reproduce the background of a black and white image, either a thresholding stage should accompany the FANNS and interpolation, or the FANN training should take into account the special characteristics of the bi–level images.

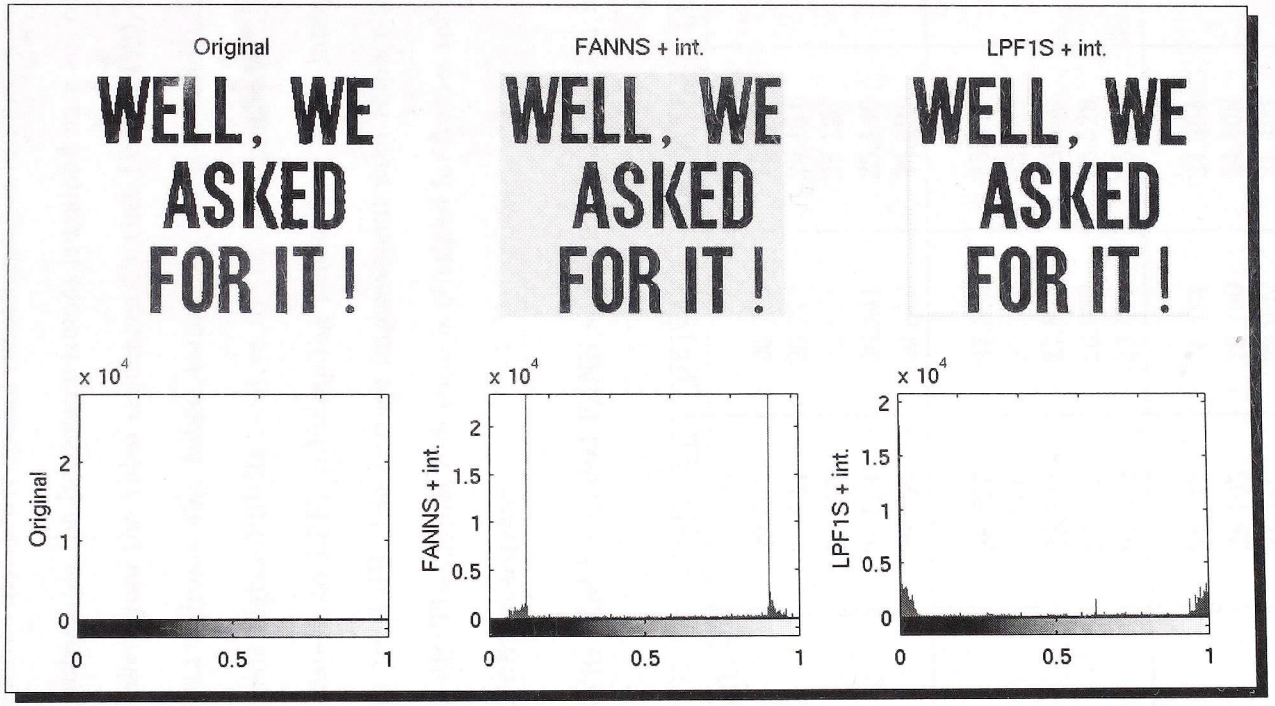


Figure 4.12: Original, subsampled and cubic interpolated 512×512 block of the 1200×1524 image F08–200.

4.4.2 Subsampling of Luminance Video Frames

In this section, we address first–order spatial subsampling of luminance video frames. The FANN used to subsample the luminance frames is trained on a set consisting of different Y frames taken from the video sequences CLAIRE (frame 490), GRANDMA (frame 490), SALESMAN (frame 49), MISS AMERICA (frame 49) and SUZIE (frame 49). Our FANN yields higher PSNRs relative to the LPF methods. The average improvement with respect to LPF1 subsampling followed by cubic interpolation in Table 4.3 is equal to 1.893 dB. The average improvement with respect to the three filters is equal to 3.3 dB. The highest difference is obtained for frames taken from the MOTHER–AND–DAUGHTER sequence.

Table 4.3: Test PSNRs [dB] for spatial FANN subsampling of the 144 × 176 video frames.

Frames	FANNS+int.	LPF1S+int.	LPF2S+int.	LPF3S+int.
Mother–and–daughter				
275	29.645	26.769	25.125	24.300
276	29.635	26.772	25.131	24.307
491	29.575	26.814	25.124	24.272
875	29.504	26.841	25.136	24.287
876	29.459	26.810	25.121	24.278
Carphone				
50	28.327	27.389	25.555	24.668
75	28.766	27.819	25.945	25.038
76	28.738	27.833	25.956	25.046
275	27.488	26.989	25.529	24.802
276	27.412	27.052	25.598	24.872
Trevor				
16	27.927	26.481	24.554	23.630
50	28.703	26.700	24.808	23.903
75	31.785	29.002	27.498	26.770
76	31.933	29.124	27.577	26.832

4.4.3 Subsampling of Noisy Images

Real image recording systems are not ideal. Therefore, all image samples are usually contaminated with noise in various levels, due to sensors (quantum noise) or to circuits (thermal noise) [4, 16]. The sources of noise are usually modeled by impulsive, salt–and–pepper or other types of noise, which can be removed by filtering in a pre–processing stage. The cost, however, is the time delay associated with the additional filtering step.

In this section, we address FANN subsampling of video frames contaminated with noise. The FANN structure used here was trained using the FABS algorithm and the same video frames as before, but corrupted with salt–and–pepper noise with density of 0.25. Test results using MOTHER–AND–DAUGHTER noisy frames, with noise density between 0.05 and 0.5, show clearly better FANN PSNRs as compared to the LPFS ones. This is illustrated in Figure 4.13. Moreover, our FANN achieves a graceful performance degradation in terms of PSNR values with increasing noise density. It is clear that the FANN can effectively attenuate the noise present in the frames, while in the standard approach, lowpass filtering accentuates the salt–and–pepper noise [136].

Since the FABS algorithm employs median operations, it is not a surprise that its results when using the frames corrupted by salt–and–pepper noise are far superior to the lowpass filtering followed by subsampling. Also, it is important to compare the subsampling techniques using images corrupted by Gaussian–distributed noise. Results for a test frame from the MOTHER–AND–DAUGHTER sequence, with Gaussian noise (variance 100), subsampled by the FANN and then cubic interpolated, as well

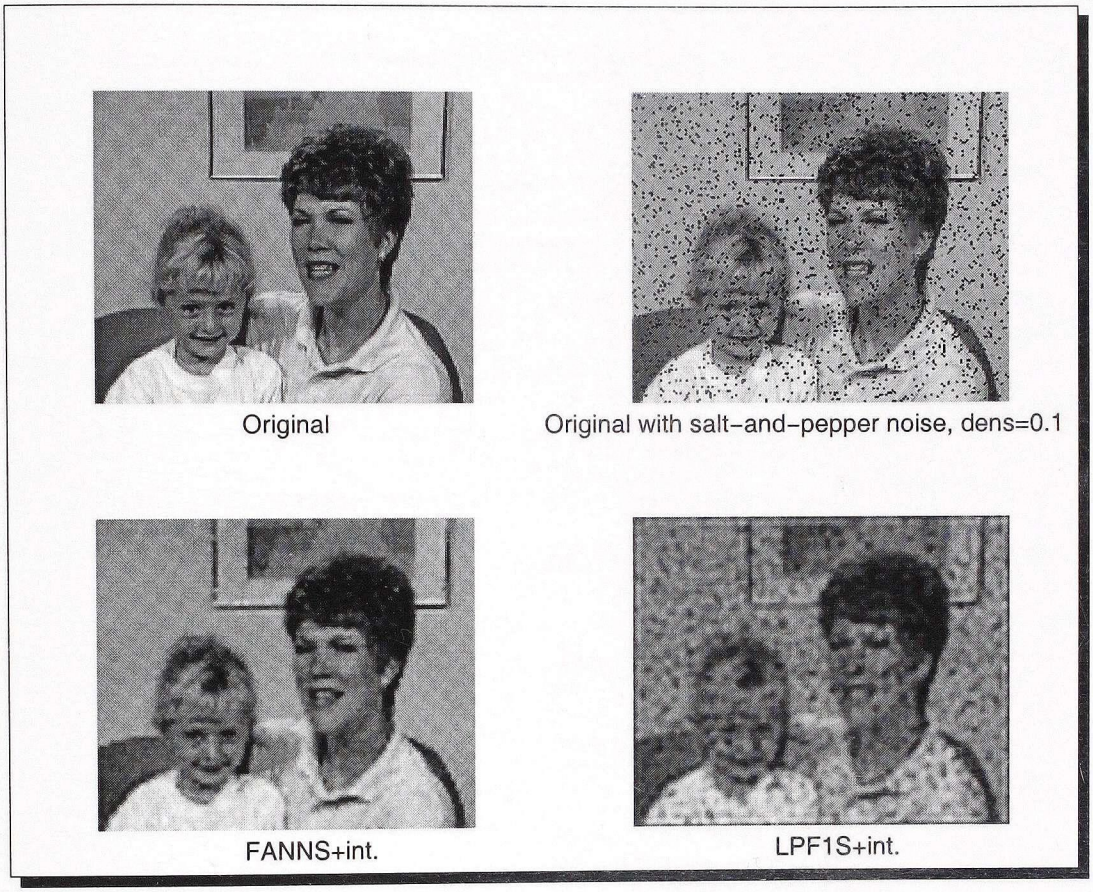


Figure 4.13: Subsampled and cubic interpolated MOTHER–AND–DAUGHTER with salt–and–pepper noise.

as subsampled and interpolated following the standard approach, are given in Figure 4.14. As in the case of salt–and–pepper noise, the FANN performance degrades gracefully as the variance of the Gaussian noise increases, while the LPF results drop abruptly.

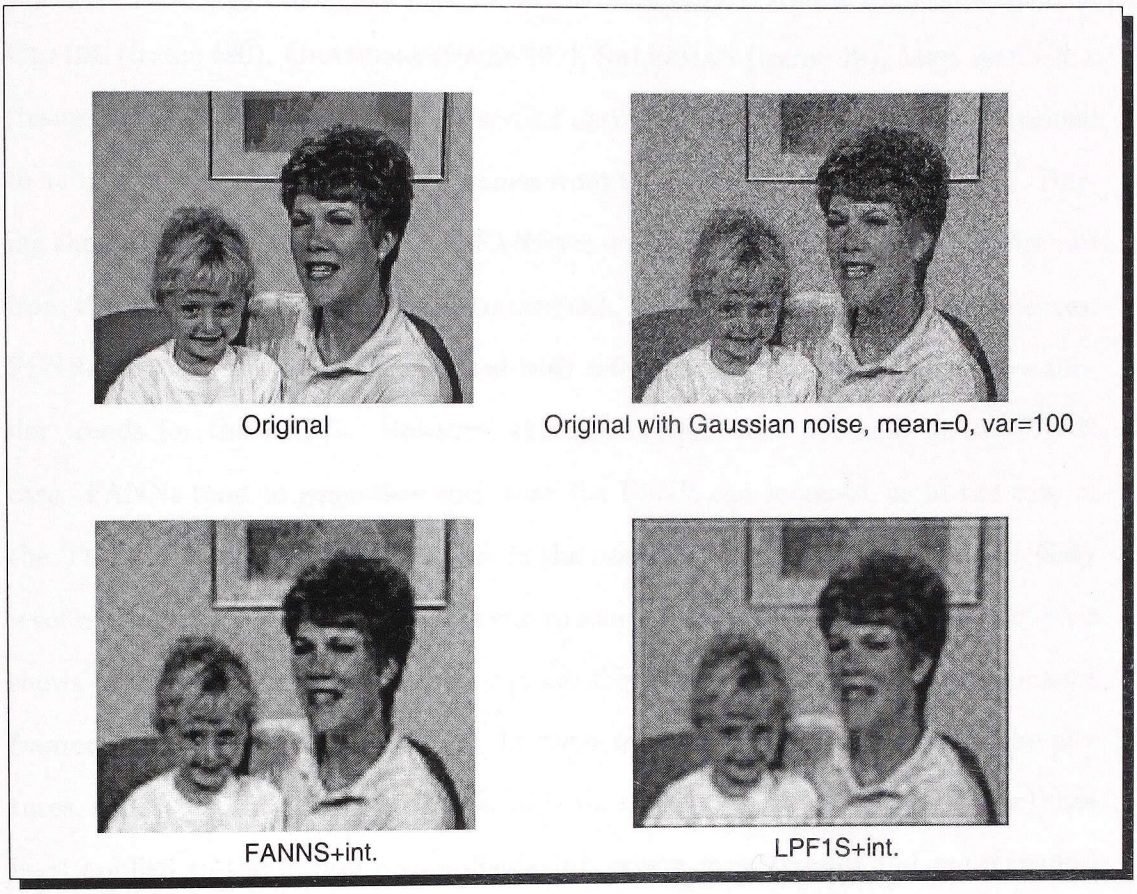


Figure 4.14: Subsampled and cubic interpolated MOTHER–AND–DAUGHTER with Gaussian noise.

4.5 Application: Video Coding

In color video coding, it is common to subsample the chrominance frames, while keeping the luminance component at the same resolution level. This is possible due to the lower sensitivity of the human visual system to color information, as compared to its sensitivity to the luminance information [19, 8, 123, 137]. The block diagram, illustrating the operation of the chrominance subsampling system using FANNs, is shown

in Figure 4.15. One FANN has been trained to subsample U frames from the sequences CLAIRE (frame 490), GRANDMA (frame 490), SALESMAN (frame 49), MISS AMERICA (frame 49) and SUZIE (frame 49). A second network of the same size has been trained to subsample the corresponding V frames from the above mentioned sequences. During the testing step, each of the two FANNs is used to subsample chrominance frames from the sequences MOTHER–AND–DAUGHTER, CARPHONE and TREVOR sequences. PSNRs for U and V frames, computed with reference to the original ones, have similar trends for the LPFSs. However, this observation does not hold in the FANN case. FANNs tend to generalize such that the PSNR can increase, as in the case of the TREVOR sequence, or decrease as in the case of the CARPHONE sequence. Gray level reproduction of the subsampled chrominance frame 75 in the TREVOR sequence shows that the FANN processed frames are the closest to the original chrominance frames, as illustrated in Figure 4.16. In order to allow the interpretation of the pictures, the transformation $(frame - min(frame))/(max(frame) - min(frame))$ has been applied to the frames before displaying, where $min(frame)$ and $max(frame)$ denote the minimum and the maximum values of the pixels in the original frame, respectively.

In a low bit rate video coding experiment, we applied the FANN to chrominance subsampling of U and V frames, previously cubic interpolated[5]. Then, we encoded the QCIF test sequence MOTHER–AND–DAUGHTER (150 frames) using Telenor's H.263 video coder [138]. Both values of the PSNR, given by

[5]All the available video sequences have the chrominance frames already subsampled and we need therefore to upsample them before testing our FANN subsampler. We have employed cubic interpolation in order to upsample the chrominance frames.

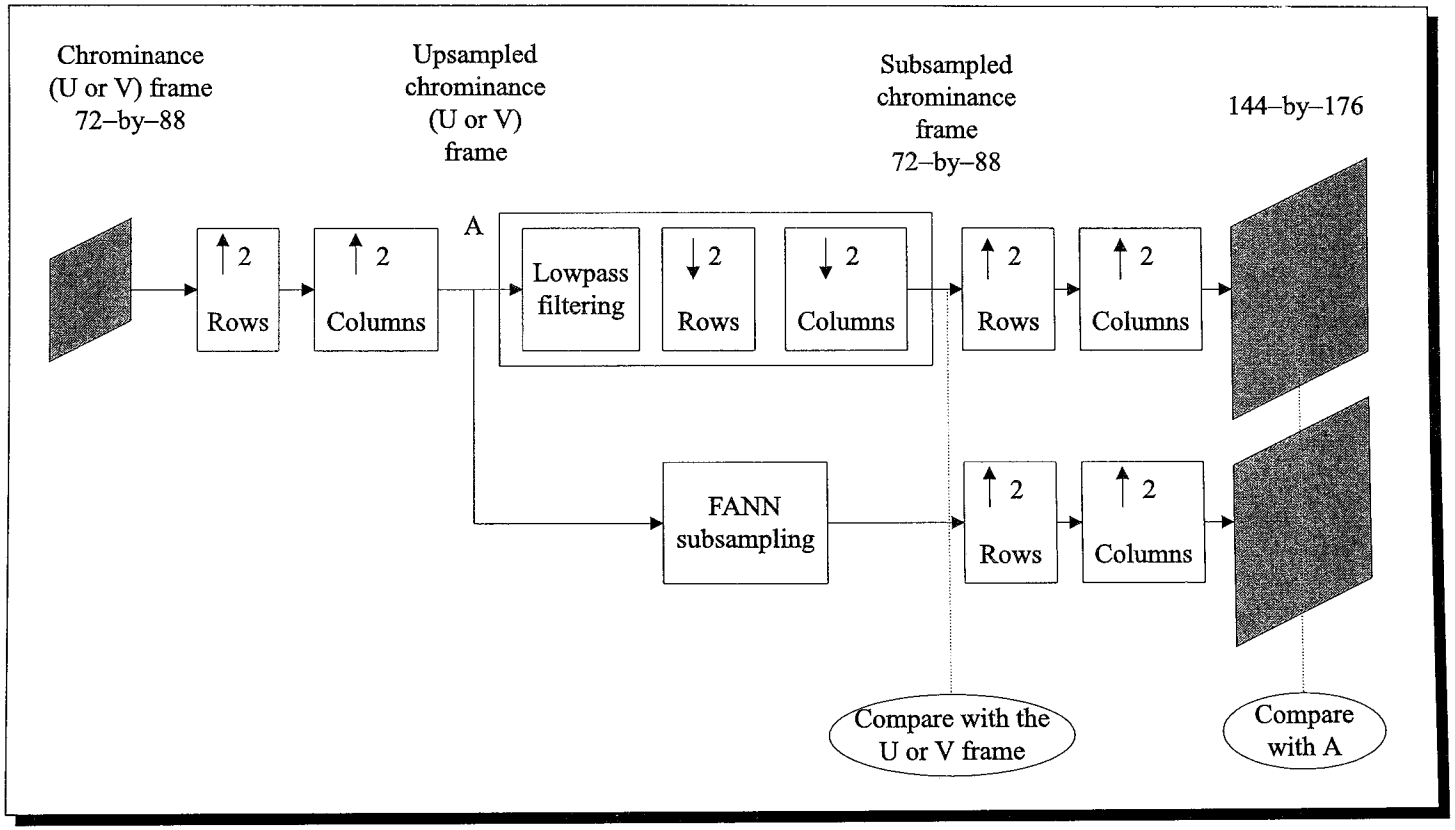


Figure 4.15: Block diagram of the chrominance subsampling system. All the available video sequences have the chrominance frames already subsampled and we need therefore to upsample them before testing our FANN subsampler. Upsampling has been performed by cubic interpolation.

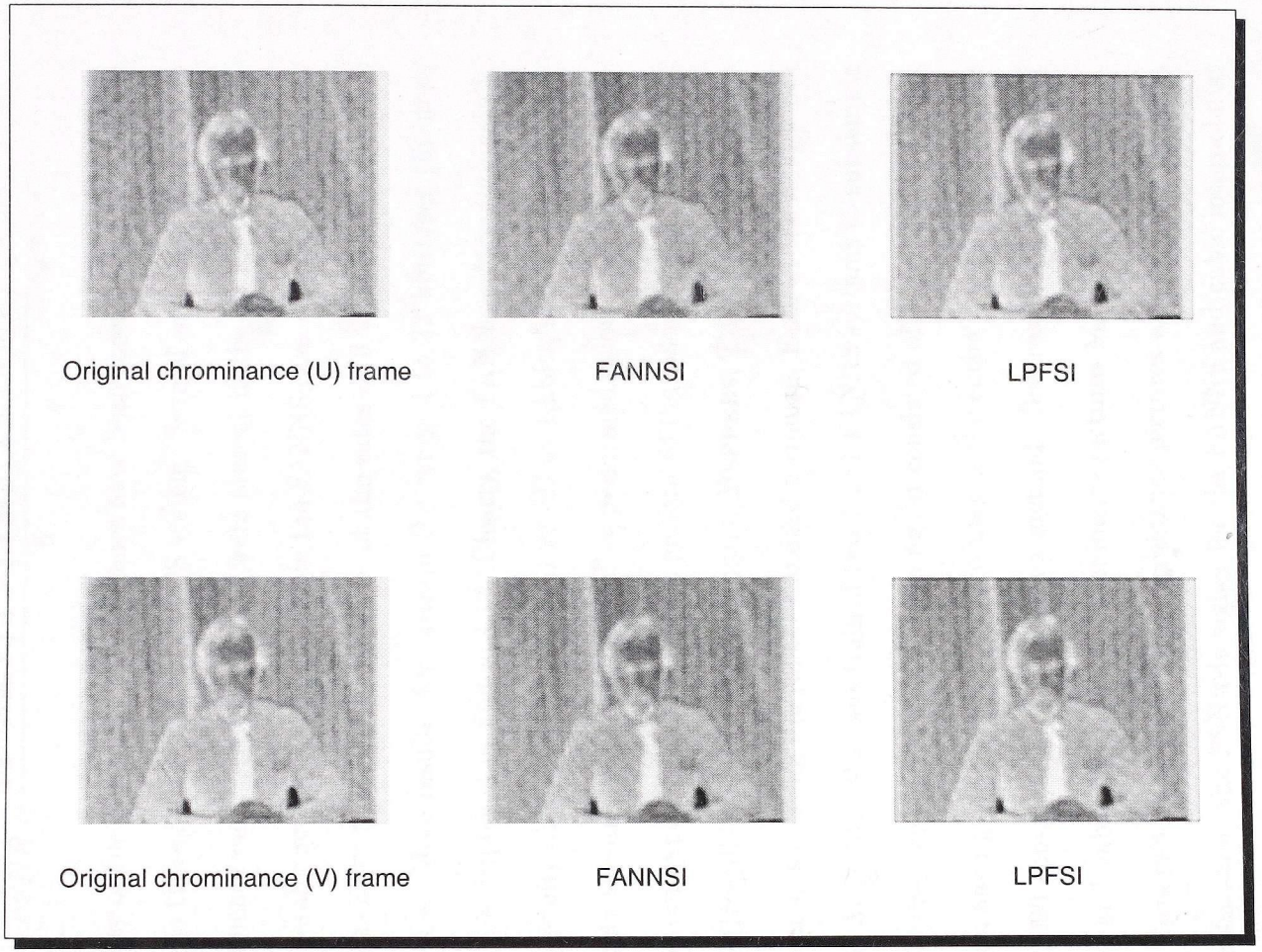


Figure 4.16: Gray level representation of the chrominance frames. In order to allow the interpretation of the pictures, the transform $(frame - min(frame))\ /\ (max(frame) - min(frame))$ has been applied to the frames before displaying, where $min(frame)$ and $max(frame)$ denote the minimum and the maximum values of the pixels in the original frame, respectively.

$$PSNR = \frac{4\,PSNR(Y) + PSNR(U) + PSNR(V)}{6},$$

and visual quality of the obtained video sequence have been compared to the original ones, as well as to those obtained by the LPFS system. Several subjective evaluations of the video sequence indicate that artifacts were present in the LPFS case, as compared to the generally good quality observed in the FANNS case. Quantitative results for the MOTHER–AND–DAUGHTER sequence, at the rates of 8 and 24 kbits/sec, are given in Table 4.4. More results, for rates in the range 4 to 32 kbits/sec (in steps of 2 kbits/sec) are displayed in Figure 4.17. Clearly, the FANN performance gain is substantial, especially at low bit rates (e.g., 1.92 dB at 4 kbits/sec).

To evaluate the impact of the upsampling performed prior to the chrominance subsampling, on our experimental results, we performed the following experiment: we converted the RGB still images LENA, MANDRILL, PEPPERS, TIFFANY and FIGHTER to the YUV color space, using the NTSC standard formulae. Each of the images has the size equal to 512×512. We have trained two 4–2–1 FANNS systems to subsample the U and V pictures, respectively. The training set consisted of a 256×256 block in the LENA image, having the same geometric center as the entire image. Training was performed for 1000 epochs using our proposed method. Test results after comparing the subsampled and cubic interpolated chrominance pictures MANDRILL, PEPPERS, TIFFANY and FIGHTER to the original chrominance pictures are included in Table 4.5. As the results show, the PSNR is higher for the FANNS and cubic interpolated images as compared to the LPFS and cubic interpolated images. The quality of the reconstructed images is also subjectively good.

Table 4.4: PSNRs [dB] for different coding rates (8 kbits/sec and 24 kbits/sec), when using Telenor's H.263 low bit rate encoder. Notation Y stands for the luminance frames, U and V stand for the chrominance frames.

Method	8 kbits/sec			24 kbits/sec		
	Y	U	V	Y	U	V
FANNS+int.	30.878	37.214	37.381	34.295	39.768	39.610
LPF1S+int.	30.434	32.838	33.045	33.938	38.031	38.168

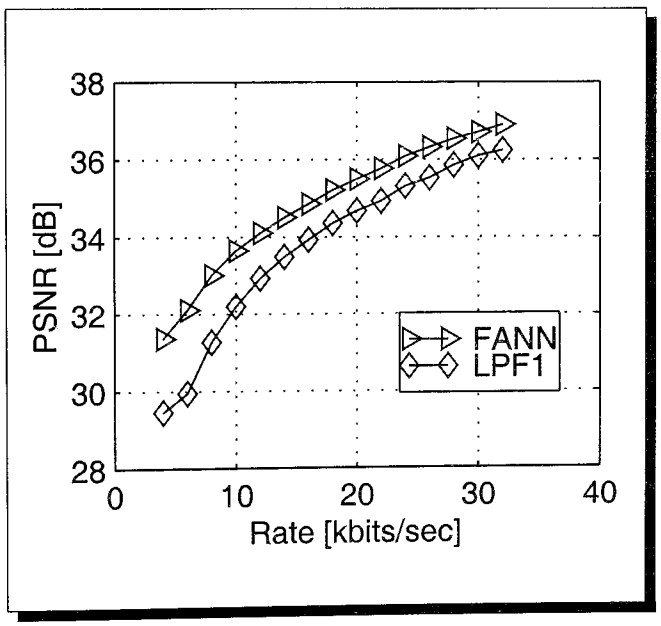


Figure 4.17: Peak signal–to–noise ratio [dB] with respect to rate in low bit rate experiments using Telenor's H.263 video encoder.

Table 4.5: Test PSNR [dB] on chrominance frames, generated by direct RGB to YUV conversion. Each of the two FANNs was trained to subsample a 256×256 block of the corresponding 512×512 LENA chrominance frame (U or V, respectively).

Image		FANNS+int.	LPF1S+int.	LPF2S+int.	LPF3S+int.
U	Mandrill	31.954	31.408	31.139	30.992
	Peppers	35.974	35.743	34.755	34.187
	Tiffany	35.964	34.949	34.255	33.854
	Fighter	39.851	39.432	38.856	38.590
V	Mandrill	32.882	32.501	32.044	31.777
	Peppers	36.313	35.603	34.254	33.512
	Tiffany	36.828	36.468	35.943	35.657
	Fighter	37.984	37.318	37.003	36.875

4.6 Experimental Results: GFABS Algorithm with Fixed Threshold

4.6.1 Subsampling of Still Images

We have applied the extended FABS algorithm with a fixed threshold β, described in Section 4.3.1, to 4×4 blocks. The FANN output consists of four pixels. As in the case of 2×2 blocks, the subsampling rate is $4 : 1$. As expected, our experimental results indicate that using 4×4 input blocks as compared to using 2×2 input blocks yields significant performance gains. More specifically, we subsampled 512×512 test images MANDRILL, BOAT and FIGHTER by applying the FANN. The PSNR values of the subsampled and then cubic interpolated images are 23.431, 27.263 and 30.896 dB, respectively. These values are higher than the values included in Table 4.2. The subjective quality of the images is better as well.

4.7 Discussion

4.7.1 FANN Generalization

An important issue is the generalization ability of the designed FANN. Both still image and video subsampling results obtained by applying the basic FABS algorithm favor clearly the FANN method as compared to the LPF methods. However, the results are more convincing for the video frames, indicating a better FANN generalization ability for the latter. The FANN generalization ability depends on the size of the training set and the number of FANN parameters. The sizes of the training sets used for our still image and video experiments have been evaluated using the theoretical results in [22]. However, for the experiments using still images, we have employed only the image LENA, as compared to the video experiments where we have employed several frames in the FANN training. As expected, by selecting various still images in order to build the training set, the performance slightly increases. However, due to the reduced number of parameters in our FANNs, the performance does not increase significantly. A solution to further improve the performance of the FANN is to increase the size of the input windows, as shown in Section 4.6.1. The size of the window is obviously an important tradeoff parameter [15], since small windows make the algorithm sensitive to high frequency image variations and large windows do not allow the algorithm to adapt to local characteristics of the image. Moreover, larger windows increase the subsampling time during the testing stage.

We believe that, in addition to the size of the training set and the size of the input block, the difference in FANN performance between still image and video experimental results is also due to the characteristics of the test set and to the range

of the pixel values in the input images. Although the video sequences employed during the testing stage have different characteristics than those employed during the training stage, they all consist of head–and–shoulders frames. Thus, the FANN can generalize well. By comparison, the still images included in our test set are diverse. Moreover, the range of the pixel values in the chrominance frames is rather small, as compared to that of the still images in the test set. Thus, the FANN inability to learn correctly the 0 and 1 values does not have any impact in the chrominance subsampling case, the FANN being able to accurately provide the correct output pixel values even at the upper and lower bounds of the pixel value range.

4.7.2 Speed and Memory Comparisons

The speed of our FANN image subsampler depends on the number of required "multiply and add" operations, which is directly related to the number of connections in the designed structure. A fully connected M–H–N FANN has $[(M+1)\ H\ +\ (H+1)\ N]$ parameters (13 for our network). Thus, the total number of "multiply and add" operations per input window (i.e., per $2 \times 2 = 4$ pixels) is exactly

$$[(M\ +\ N)\ H\ -\ (M\ -\ N\ -\ H\ +\ 2)]\,,$$

or 7 for our FANN. Our experimental results, shown in Table 4.6, confirm that the FANN requires a significantly lower number of "multiply and add" operations as compared to the LPFS systems taken into account. The test times on an UltraSparc 2 computer indicate that the FANNS is an average 5.5 times faster than the LPFS system during the testing stage.

The CPU time required by 1000 training epochs using the 256×256 image

Table 4.6: CPU times on an UltraSparc 2 computer, number of floating point operations and memory requirements.

Complexity indices	Image size	FANNS	LPF1S	LPF2S	LPF3S
CPU time [sec]	144×176	0.08	0.34	2.12	2.56
	256×256	0.19	1.16	6.99	8.47
	512×512	0.77	4.81	29.15	35.50
Floating point operations	144×176	234,451	6,154,212	37,902,550	46,035,951
	256×256	606,227	15,880,676	97,386,710	118,381,551
	512×512	2,424,851	63,459,812	388,366,550	472,275,951
Memory [bytes]	512×512	1,048,744	1,049,544	1,056,768	1,056,768

LENA is equal to 3.218 hours. However, we note that, for both still images and video, reducing the testing time is what really matters for the end user. For a large still image, such as the 1200×1524 TOOLS image of the JPEG–2000 test set, the testing (subsampling) times are equal to 3.65 and 24.73 seconds, when the FANN and LPF1 methods are employed, respectively. Thus, in this case the FANN is 6.77 times faster than the LPF1. Finally, as Table 4.6 indicates, the required memory for our FANN method is comparable to that of the LPFS's, which makes our FANN trained with the FABS algorihtm well–suited for practical applications.

When the generalized FABS algorithm with a fixed threshold is applied, the subsampling time during the testing stage increases 148.06 times, as compared to the basic FABS algorithm described in Section 4.2.1. Therefore, methods to reduce the testing time should be considered. This problem will be addressed in Section 5.4.

4.8 Summary

We have applied FANN models to first–order image subsampling by proposing a new FANN training method which is based on pattern matching. Besides its high speed and low memory requirements, as compared to traditional LPFS methods, our method has the advantage of better image reproduction quality. Therefore, it can be used in a variety of applications, ranging from pyramidal coding [9] to user–defined decimation steps in various still image and video coding systems. In particular, we have shown the superior performance of our algorithm in chrominance subsampling within a low bit rate video coding system. We have also shown that our algorithm can be easily generalized for various sizes of the input blocks, with good performance results.

Chapter 5

Application of FANNs to High–Order Image Subsampling (HOS)

In this chapter, we apply FANNs to high–order image subsampling (HOS). In Section 5.1, we comment on the direct application of our FABS algorithm to HOS. In Section 5.2, we evaluate the performance of our GFABS algorithm. In Section 5.3, we show that, by selecting an adaptive threshold, the performance of our GFABS algorithm in HOS is significantly improved. In the same section, we next show that the computational demands of our GFABS algorithms increase with the size of the input image and the size of the input window. In order to address this problem, in Section 5.4, we reduce the connectivity of the trained FANN by using our TOBD algorithm. The performance and computational demands of the resulting tridiagonally symmetrical FANNs in HOS are presented and discussed in the same section. A summary of the chapter is included in Section 5.6.

As stated in Chapter 2 and illustrated in Figure 2.11, high–order image subsampling (HOS) can generally be performed as a single–stage or a multi–stage process. In single–stage HOS using large input blocks, the value of the output pixel is difficult to select [17]. Moreover, in FANN–based single–stage HOS, training is especially difficult. Both of the above mentioned problems can be addressed by decomposing the high–order subsampling process into several first–order subsampling (FOS) stages. In addition to simplifying the subsampling process, this multi–stage FOS (MFOS) solution provides several images with lower resolution than that of the original image.

In what follows, our goal is to evaluate the performance and complexity of FANN–based multi–stage FOS. Without loss of generality, let us consider a $16:1$ subsampling process. Let us also decompose this process into two $4:1$ first–order subsampling stages. In other words, image subsampling by 2 in each direction is performed during each of the FOS stages.

5.1 FABS Algorithm

We have applied twice the 4–2–1 FANN trained using the FABS algorithm of Chapter 4 to subsample the input images. In each of the FOS stages, subsampling by 2 in each direction has been performed. The sizes of the original images are equal to 512×512. We have obtained downsampled images having the sizes equal to 256×256 and 128×128, respectively. Then, we upsampled the 128×128 images by cubic interpolation: (a) by 2 in each direction, and (b) by 4 in each direction.

All of the downsampled images have good subjective quality. However, the upsampled images obtained using the method (b) show some feature distortion. By

evaluating the histograms of the upsampled images obtained using both the methods (a) and (b), we note that the second downsampling stage was applied to images having already the histogram modified by the FANN. Thus, it is more difficult to reconstruct the original image after successive FANN downsampling stages. We have employed histogram equalization after the first FANN downsampling stage, but such did not improve the results. Therefore, we conclude that, when using multi–stage downsampling by FANNs, the histogram should be preserved during each stage as much as possible.

5.2 GFABS Algorithm: Fixed Threshold

The GFABS algorithm with fixed threshold (described in Section 4.3.1) avoids major histogram changes during each downsampling stage. We have applied it twice to subsample the images listed in Table 5.1. The size of each of the input windows is equal to 4×4. The sizes of the downsampled images after each of the successive downsampling stages are, again, 256×256 and 128×128. The PSNR values that have been obtained after reconstructing the downsampled images by cubic interpolation are included in Table 5.1. These values have been compared with those of the LPFS performed also in two successive stages, followed by cubic interpolation. We note that the PSNR values obtained using the FANN method are slightly higher for the images MANDRILL and FIGHTER. The PSNR values obtained when using the LPF method are higher for the image BOAT. However, the subjective quality of the images obtained using the FANN method is significantly better than that of the images obtained using the LPF method.

Table 5.1: Test PSNR [dB] for MFOS using still images. The FANN was trained using the GFABS algorithm with fixed threshold. The training set consists of the 256 × 256 image LENA.

Image	FANNS+int.	LPF1S+int.	LPF2S+int.	LPF3S+int.
Mandrill	20.760	20.656	19.093	18.257
Boat	22.183	22.684	19.339	17.940
Fighter	25.347	24.823	19.864	18.164

5.3 GFABS Algorithm: Adaptive Threshold

The selection of a fixed threshold value in our FABS algorithm has the advantages of simplicity and generally good performance. Moreover, the algorithm can also be easily applied to 8 × 8 and 16 × 16 windows, by selecting a different threshold value. However, the value of the threshold, which changes with the window size, must be a priori selected. Also, using a global value, (i.e., the same value for the entire image), the threshold value cannot capture the local characteristics of each of the image blocks. A solution to address the above problems is to select adaptively a different threshold value for each image block during the subsampling process.

The GFABS algorithm with adaptive threshold β is summarized in Appendix C. The main idea of the algorithm is to use more local information at the image block level than that already employed in the FABS algorithm in order to select the FANN desired output value. More specifically, the standard deviation σ_k of an input image k is first computed. Next, the image k is divided into 4×4 windows. Each window is then unwrapped into a 16×1 vector (pattern). This vector is presented as input to the FANN and the output is computed. The FANN desired output is next selected

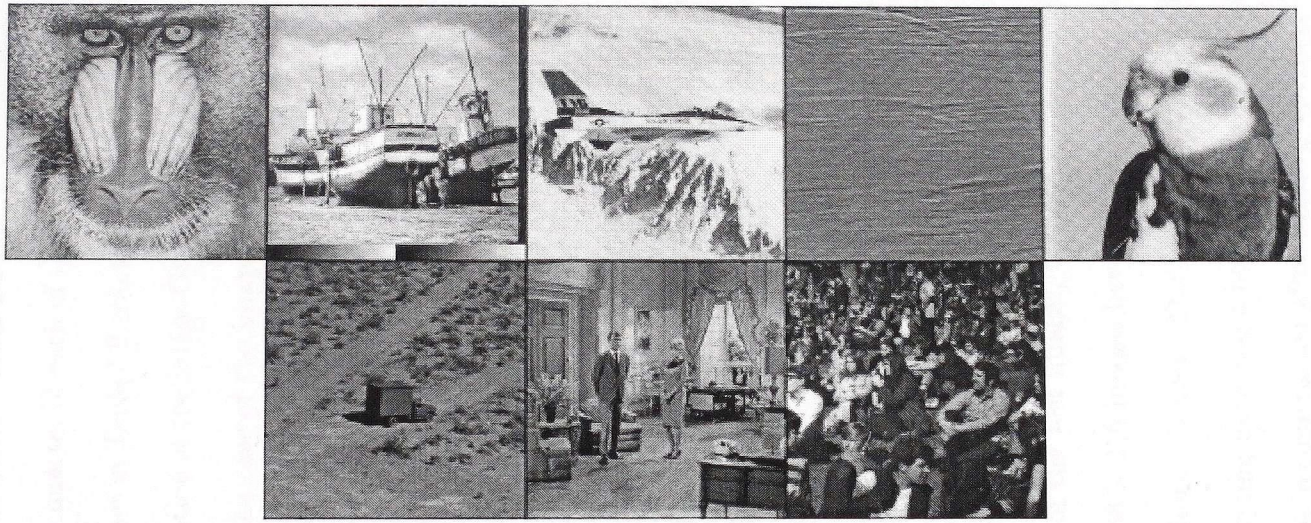
as follows. If the standard deviation of the current block is higher than the standard deviation of the corresponding input image, then an edge is declared present in the input block. Consequently, the FANN desired output value is selected according to the FABS algorithm of Section 4.2.1. Otherwise, the block is declared smooth. The FANN desired output value in this case is set to the average of the four pixels in the block. Using the FANN output value and the desired output value already computed, the value of the error for the current window is obtained. By repeating the above steps for all windows in the input image and for all input images, the global error value is obtained. Next, the weights are adjusted using the global error value. The above steps are repeated until the error value decreases below a selected threshold.

5.3.1 Experimental results

As already mentioned in Section 4.7.2, the subsampling time increases when using the GFABS algorithm with fixed threshold. However, we note here that this is mainly due to the processing stages that are necessary to evaluate the smoothness of each block and not to the FANN structure. In order to address the problem of the increasing testing time, we now train the FANN using the GFABS algorithm with an adaptive threshold. We also select a larger training set, which consists of 256×256 regions from the 512×512 images LENA, TARGET, GOLD, WOMAN1 and MAN. These images are illustrated in Figure 5.1 (a). The size of each of the images, with the exception of the image GOLD, is equal to 512×512 pixels. The image GOLD consists of 576×720 pixels.

Let the block size be equal to 4×4. Also, let the FANN be of the type 16–8–4.

(a)



(b)

Figure 5.1: (a) Training and (b) test images. The 16–8–4 FANN was trained using 256×256 regions of the images (a). The test images (b) have the same size as the images (a).

We train the FANN for 1000 epochs with the same parameters as those employed in Section 4.4. The PSNR values for first–order subsampling are included in Table 5.2. Since only one quarter of each of these images has been employed in FANN training, the PSNR values in Table 5.2 are, to some extent, test results. For all of these images, with the exception of the image GOLD, FANNS leads to better results than the LPFS methods. In the case of the image GOLD, we believe that the slightly lower FANN performance is due to the different characteristics of the large flat areas in the bottom half and right side of the image, which have not been included in the training set.

The test PSNR values for various test images, which are illustrated in Figure 5.1 (*b*), are included in Table 5.3. For all of the images in Table 5.3, the FANN results are very good. Particularly, the test results for the images MANDRILL, BOAT and FIGHTER are significantly better than those in Table 4.2, corresponding to the 4–2–1 FANN trained using the FABS algorithm. Moreover, in all cases, the FANN results outperform the results obtained by applying the LPFS methods. The FANN results outperform, on average, by 1.758 dB the best LPFS method employed in our comparisons. The FANN results also outperform, on average, by 0.158 dB those of subsampling via median filtering (MEDS) method, employed here in order to compare the FANN performance to that of conventional nonlinear filtering.

For all of the test images, the subjective quality is good as well. The FANN subsampled 256×256 images have a sharper appearance than the LPF1S and than the median filtered images. The images reconstructed by cubic interpolation after FANN subsampling have also a better quality than those reconstructed after LPF1S and than those reconstructed after MEDS, especially in the high detail and in the

Table 5.2: Training PSNR [dB] for FOS of still images. The 16–8–4 FANN was trained using 256 × 256 regions of these images and the GFABS algorithm with adaptive threshold. The size of each image is equal to 512 × 512 pixels, with the exception of the image GOLD which has the size equal to 576 × 720 pixels.

Image	FANNS+int.	Conventional methods			
		LPF1S+int.	LPF2S+int.	LPF3S+int.	MEDS+int.
Lena	33.118	31.184	27.559	26.066	33.157
Target	15.921	15.656	15.505	15.379	15.889
Gold	31.288	31.504	28.466	27.102	31.106
Woman1	29.060	28.864	26.168	24.900	29.020
Man	30.255	29.225	26.855	25.723	30.226

Table 5.3: Test PSNR [dB] for FOS using 512 × 512 still images. The 16–8–4 FANN was trained using 256 × 256 regions of the images shown in Table 5.2 and the GFABS algorithm with adaptive threshold.

Image	FANNS+int.	Conventional methods			
		LPF1S+int.	LPF2S+int.	LPF3S+int.	MEDS+int.
Mandrill	23.424	23.003	22.113	21.607	23.338
Boat	26.831	24.462	23.424	23.158	26.2404
Fighter	30.852	26.130	25.446	23.733	30.906
Seismic	37.389	34.105	28.832	27.034	36.535
Bird	36.496	33.601	29.019	27.293	36.424
Truck	32.650	31.335	27.472	27.171	32.618
Couple	28.885	27.947	25.824	24.798	28.952
Crowd	31.166	31.044	28.542	27.417	31.412

smooth areas, respectively.

Next, we applied the trained FANN to subsampling of large images in the JPEG–2000 image set. As Table 5.4 shows, for most of these images, the FANN PSNR values are slightly higher as compared to those of the LPFS methods. More specifically, the FANN results outperform, on average, by 0.35 dB those of the best LPFS method employed in our comparisons. Moreover, the FANN results outperform, on average, by 0.498 dB those of the MEDS method. Finally, note that the FANN achieves better performance although the resolution of the JPEG–2000 images is very different than that of the training images.

We next apply the 16–8–4 FANN trained earlier using the GFABS algorithm with adaptive threshold to high–order subsampling. In other words, we apply twice the FANN to the images included in Tables 5.5 and 5.6, respectively. The FANN performance for the images in Table 5.5 is close to that of the LPF1S. However, FANN generalizes well, as shown by the results included in Table 5.6. For all the test images, with the exception of the image BOAT, the test PSNR values for FANN subsampling are higher than those of the LPF followed by subsampling. The FANN results also outperform, on average, by 0.7 dB those of the best LPFS method employed in our comparisons. Finally, the FANN results are slightly higher than those of the MEDS method.

For all of the FANN subsampled and interpolated images, the subjective quality is good as well. Let us illustrate this by using the example of the 512×512 test image BIRD, shown in Figure 5.2. The FANN subsampled 128×128 image BIRD, which is illustrated in Figure 5.3 (a), has again a sharper appearance than the LPF1S image

Table 5.4: Test PSNR [dB] for FOS using large JPEG–2000 still images. The 16–8–4 FANN was trained using 256 × 256 regions of the images shown in Table 5.2 and the GFABS algorithm with adaptive threshold.

Image	Image size	FANNS+int.	Conventional methods			
			LPF1S+int.	LPF2S+int.	LPF3S+int.	MEDS+int.
Txtur1	1024 × 1024	20.991	20.977	20.355	20.003	20.982
Mat	1146 × 1528	32.664	32.630	28.269	26.623	33.391
Tools	1200 × 1524	22.060	22.075	21.110	20.653	22.402
Water	1999 × 1465	41.024	38.119	30.145	27.986	41.161
Aerial2	2048 × 2048	28.980	29.780	26.315	24.843	29.349
Cafe	2560 × 2048	22.085	22.011	21.240	20.849	22.334
Woman	2560 × 2048	27.637	27.404	25.304	24.293	22.334

Table 5.5: Training PSNR [dB] for MFOS. The 16–8–4 FANN was trained using 256×256 regions of these images and the GFABS algorithm with adaptive threshold. The size of each image is equal to 512×512 pixels, with the exception of the image GOLD, which has the size equal to 576×720 pixels.

Image	FANNS+int.	Conventional methods			
		LPF1S+int.	LPF2S+int.	LPF3S+int.	MEDS+int.
Lena	27.443	26.648	22.352	20.766	27.490
Target	14.288	14.342	13.918	13.627	14.256
Gold	27.002	27.833	23.704	22.120	27.271
Woman1	25.548	25.691	21.675	20.123	25.544
Man	25.685	25.331	22.175	20.844	25.683

Table 5.6: Test PSNR [dB] for MFOS. The 16–8–4 FANN was trained using 256×256 regions of the images shown in Table 5.5 and the GFABS algorithm with adaptive threshold.

Image	FANNS+int.	Conventional methods			
		LPF1S+int.	LPF2S+int.	LPF3S+int.	MEDS+int.
Mandrill	20.749	20.656	19.093	18.257	20.657
Boat	22.025	22.684	19.339	17.940	22.364
Fighter	25.147	24.823	19.864	18.164	25.051
Seismic	29.602	28.081	23.060	21.347	29.535
Bird	32.604	30.263	23.952	22.061	32.529
Truck	28.570	27.729	23.764	22.240	28.522
Couple	24.555	24.005	21.226	20.005	24.562
Crowd	24.956	24.415	22.970	21.869	24.973

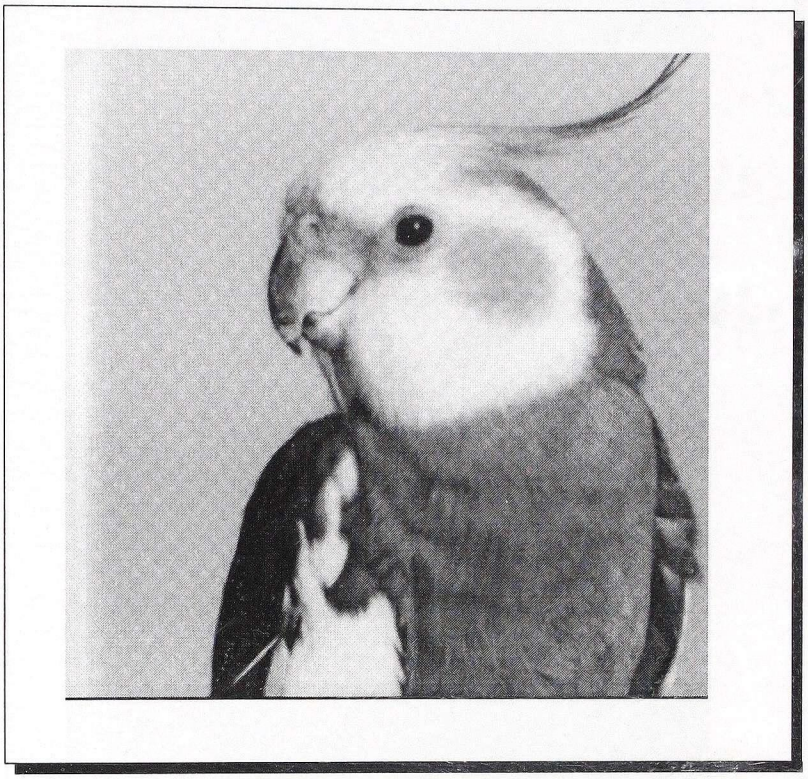


Figure 5.2: Original 512×512 image BIRD.

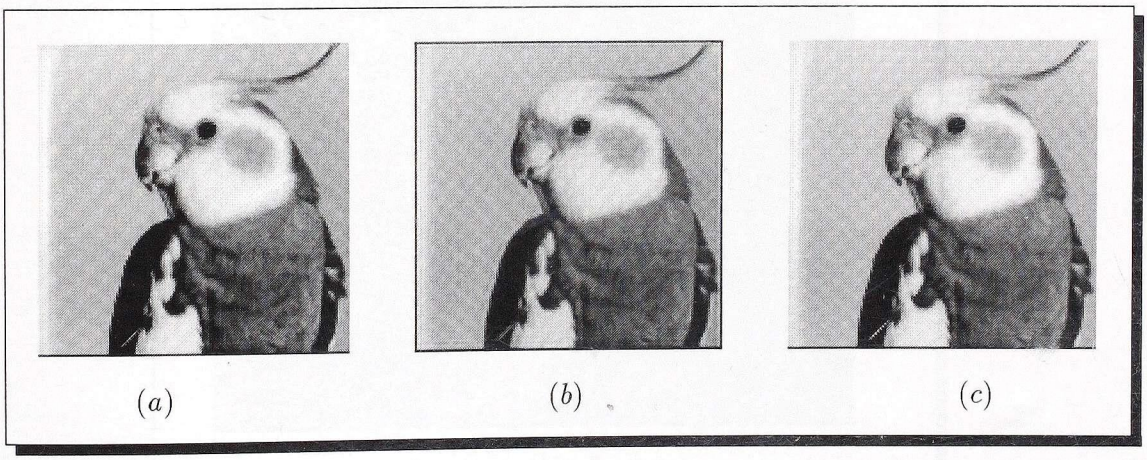
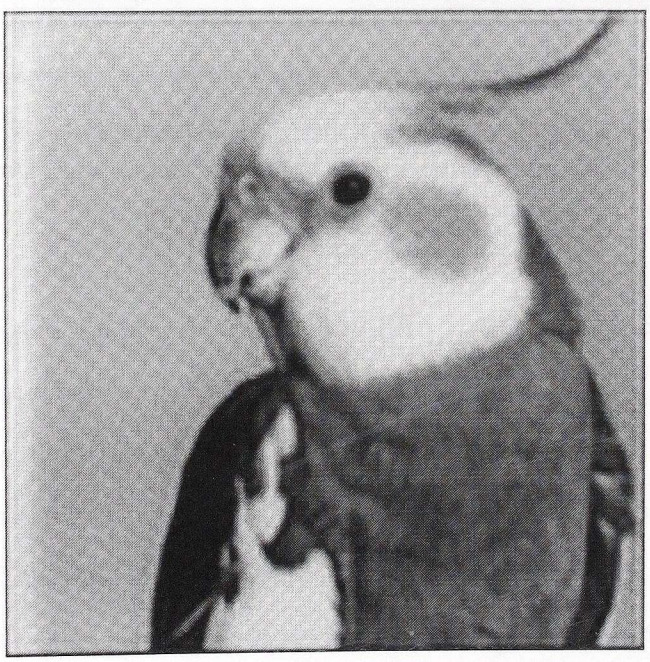


Figure 5.3: Subsampled image BIRD by (*a*) FANNS, (*b*) LPF1S, and (*c*) MEDS. The size of each subsampled image is 128×128.

(*a*)



(*b*)

Figure 5.4: Reconstructed 512×512 image BIRD by cubic interpolation after (*a*) the 16–8–4 FANN and (*b*) LPF1S have been applied twice to achieve HOS.

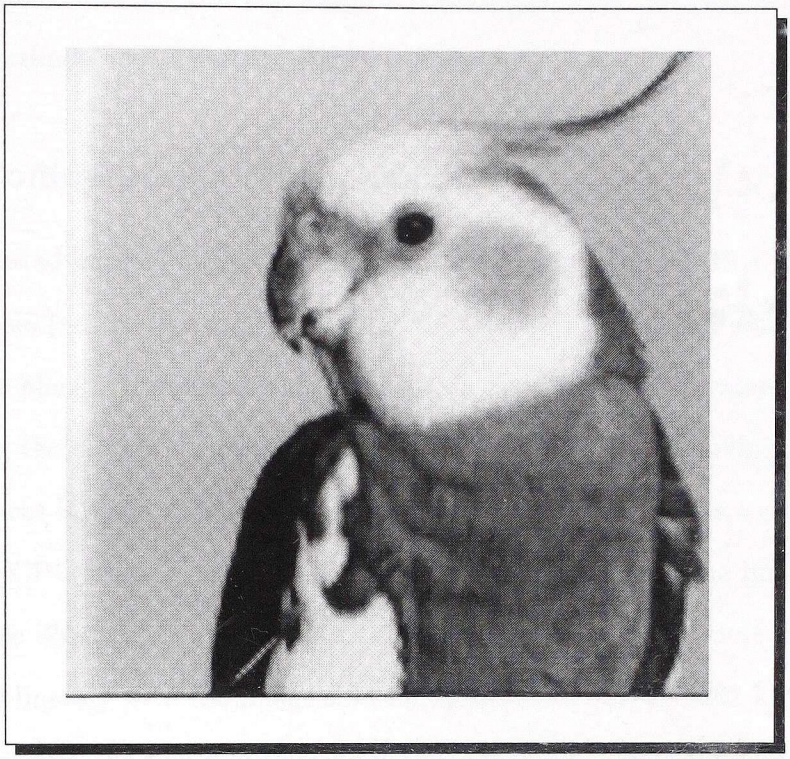


Figure 5.5: Reconstructed 512×512 image BIRD by cubic interpolation after the median subsampler has been applied twice to achieve HOS.

illustrated in Figure 5.3 (*b*). Moreover, the FANN subsampled image illustrated in Figure 5.3 (*a*) has a better quality than the median filtered image illustrated in Figure 5.3 (*c*), which contains some background artifacts. Figures 5.4 (*a*) and (*b*) show that the image reconstructed by cubic interpolation after FANN subsampling has also a better quality than that of the image reconstructed after LPF1S. The latter image is smoother, indicating that some of the details are permanently lost. Moreover, as Figure 5.4 (*a*) and Figure 5.5 show, the image reconstructed after FANNS has also

a better quality than that of the image reconstructed after MEDS, which contains significant artifacts in the smooth image regions.

5.3.2 Computational Demands

As stated earlier, higher testing times when using the FABS algorithm with fixed threshold (see Section 4.7.2) are due to the steps required to evaluate the smoothness of each block. To address this problem, in Section 5.3, we trained the 16–8–4 FANN using the GFABS algorithm with adaptive threshold. The training CPU time for 1000 epochs is equal to 58.263 hours, or 2.4276 days, on an Ultrasparc 2 computer. The testing CPU times in the case of first–order subsampling of the images listed in Table 5.4 are illustrated in Figure 5.6. Clearly, the CPU testing time increases approximately linearly with the image size, for image sizes between 262 kbytes and 6.3 Mbytes. However, the slopes of the LPFS curves have significantly larger values than those corresponding to the FANNS curves. A detail of Figure 5.6, which illustrated in Figure 5.7, shows that the FANNS gain with respect to the fastest of the LPFSs becomes higher as the image size increases. The CPU time required by FANNS is slightly higher than that required by MEDS, as Figure 5.7 also shows. It is also useful to evaluate the number of floating point operations (FLOPS) in the testing stage. As Figure 5.8 illustrates, the number of FLOPS for FANNS, LPF1S and LPF2S, when testing images having sizes between 262 kbytes and 5.2 Mbytes, increases almost linearly for FANNS and LPF1S and nonlinearly for the other filters.

The graphical representation of the testing times for high–order subsampling is illustrated in Figure 5.9. For comparison purposes, the numerical values of the FOS

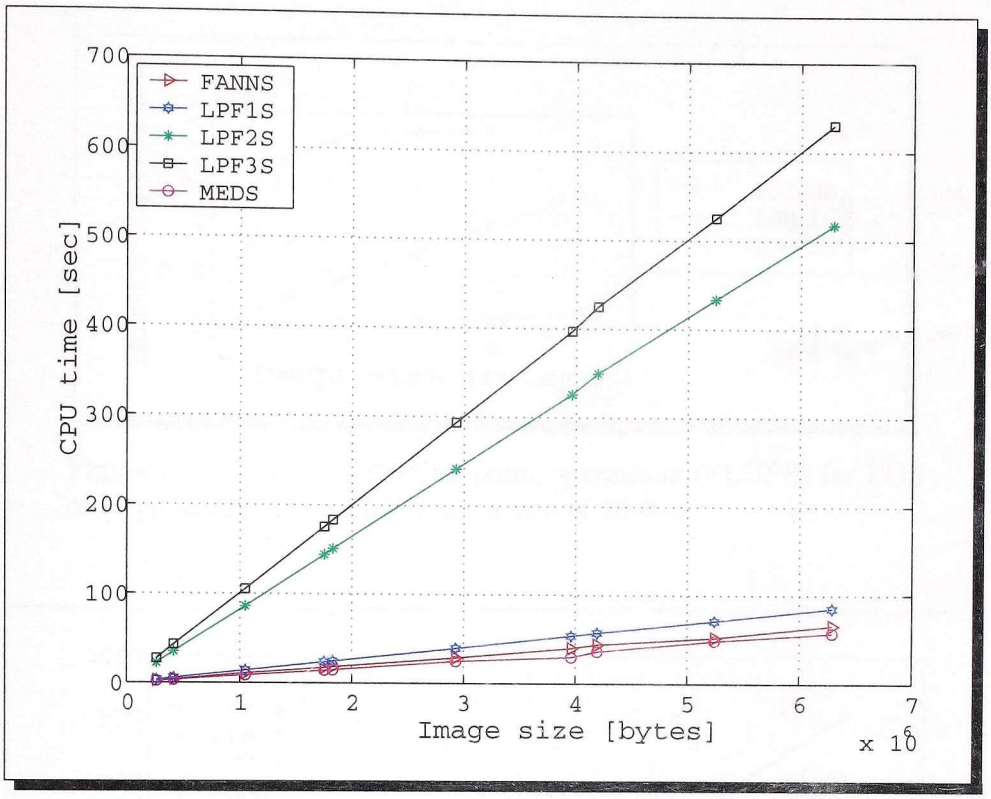


Figure 5.6: Test time [sec] for FOS of large images. The FANN has a size of 16–8–4.

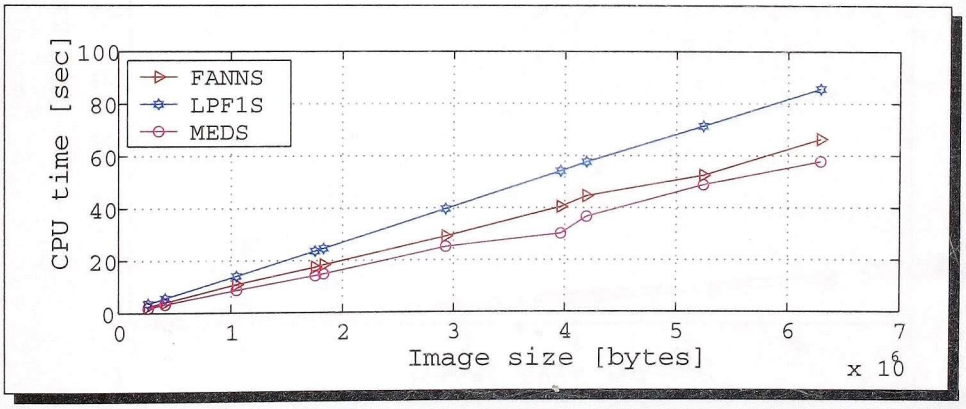


Figure 5.7: A section of Figure 5.6, illustrating the FANNS, LPF1S and MEDS test times for large images.

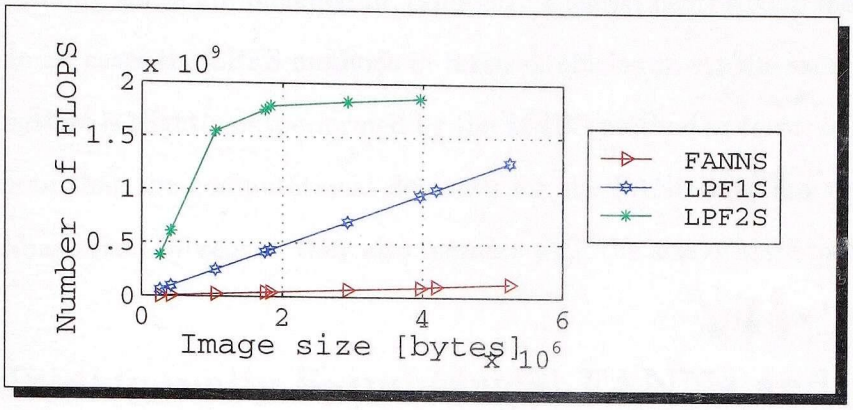


Figure 5.8: Number of floating point operations (FLOPS) for FOS of large images. The FANN has a size of 16–8–4.

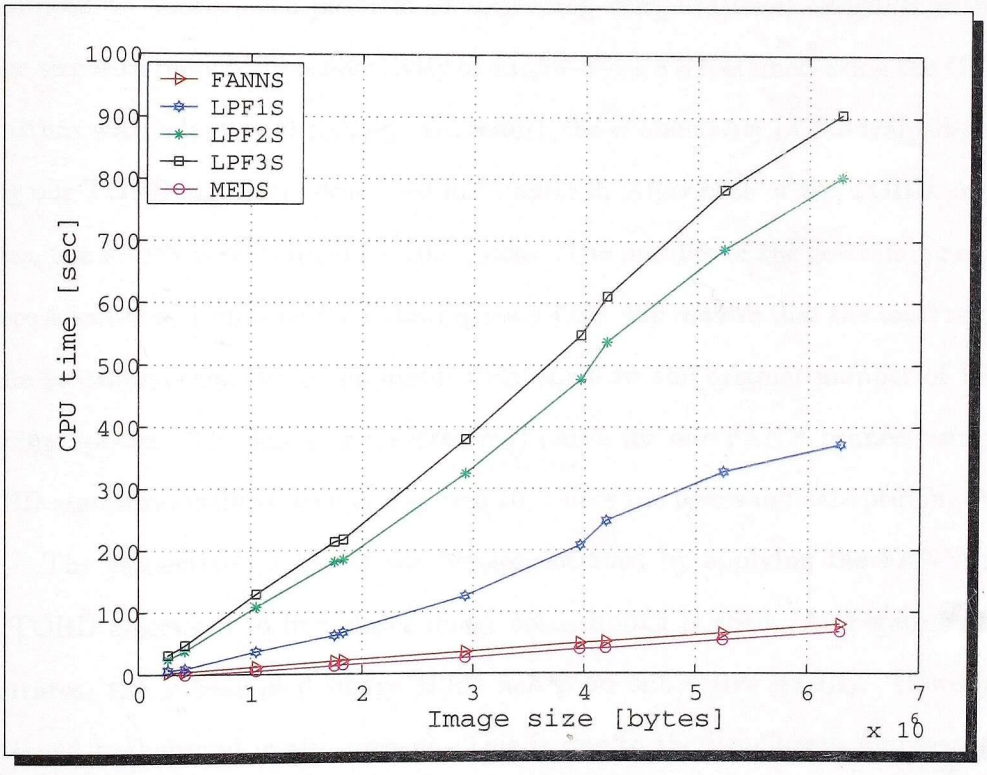


Figure 5.9: Test time [sec] for HOS of large images on an UltraSparc 2 computer. The 16–8–4 FANNS, LPFS and MEDS have been applied twice to subsample the test images.

and HOS testing times are included in Table 5.7. Clearly, our FANNS method outperforms in all cases the LPFS methods in terms of efficiency. At the same time, our FANNS method is slightly outperformed by the MEDS method in terms of efficiency. We also note that the computational demands for the FANN increase significantly with the image size. Moreover, they also increase with the size of the input window.

5.4 Tridiagonally Symmetrical FANNs and the GFABS Algorithm

A solution to address the problem of increasing computational demands with the image size is to reduce the connectivity of our 16–8–4 FANN trained using the GFABS algorithm with adaptive threshold. We reduce the connectivity of the trained FANN using our TOBD algorithm described in Chapter 3. After each of the TOBD pruning stages, the FANN is re–trained for 100 epochs. The number of the re–training epochs is here selected by imposing the following condition. We require that the total number of the re–training epochs be approximately equal to the original number of FANN training epochs. The learning (re–training) curve for our FANN pruned using the TOBD algorithm is illustrated in Figure 5.10, where the peaks indicate pruning steps.

The subjective quality of the images obtained by applying the FANN using the TOBD algorithm to first–order image subsampling is good. As Figure 5.11 (*a*) illustrates, the subsampled image BIRD has good subjective quality. However, a patterned background is also present. This is due to the insufficient number of re–training epochs after each pruning stage. Of course, a solution to eliminate the

Table 5.7: CPU test times [sec] for FOS and MFOS on an UltraSparc 2 computer. The 16–8–4 FANN was trained using 256×256 regions of the images shown in Table 5.5 and the GFABS algorithm with adaptive threshold.

Image size	First–order subsampling					Multi–stage first–order subsampling				
	FANNS	LPF1S	LPF2S	LPF3S	MEDS	FANNS	LPF1S	LPF2S	LPF3S	MEDS
512×512	2.69	3.71	22.43	27.10	2.16	3.26	7.64	25.78	30.97	2.70
576×720	4.22	5.84	35.31	42.93	3.46	5.26	10.25	39.27	46.75	4.78
1024×1024	11.10	14.28	86.39	105.24	8.87	13.00	37.82	110.41	129.50	11.25
1146×1528	17.58	23.79	144.39	175.48	14.34	22.09	63.93	183.40	214.15	19.16
1200×1524	18.51	24.83	151.20	183.42	15.06	23.19	58.81	187.02	217.69	20.79
1999×1465	29.23	39.78	241.60	293.80	25.25	36.43	127.29	326.02	379.83	31.52
2347×1688	40.46	54.21	326.28	396.89	30.28	50.87	210.01	478.13	548.00	45.77
2048×2048	44.59	57.68	349.76	424.94	36.75	53.57	249.71	538.44	610.69	47.11
2560×2048	52.45	71.36	433.88	525.30	48.88	66.84	328.76	688.63	783.20	60.63
3072×2048	66.27	85.71	517.99	630.45	57.74	81.51	373.09	805.14	915.17	74.02

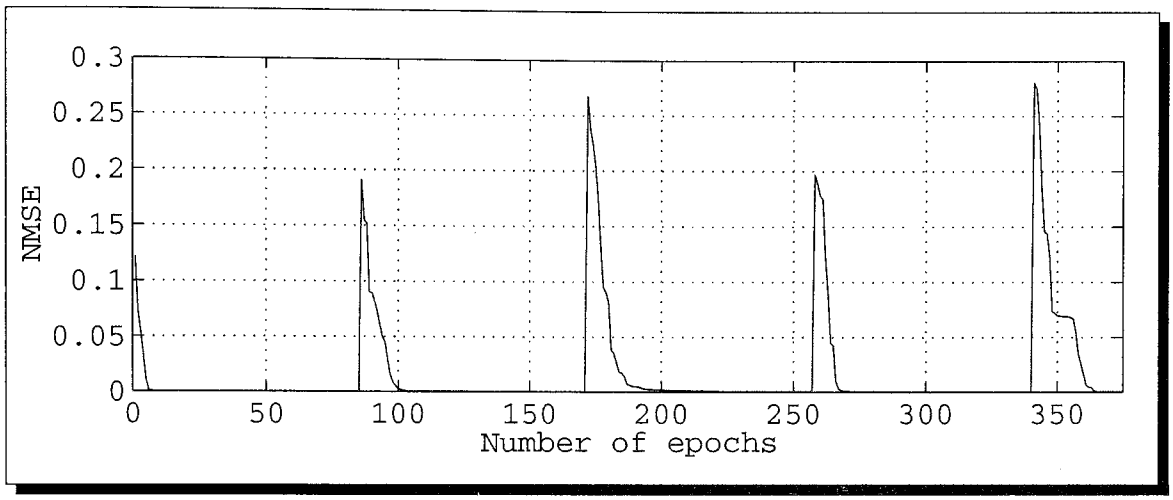
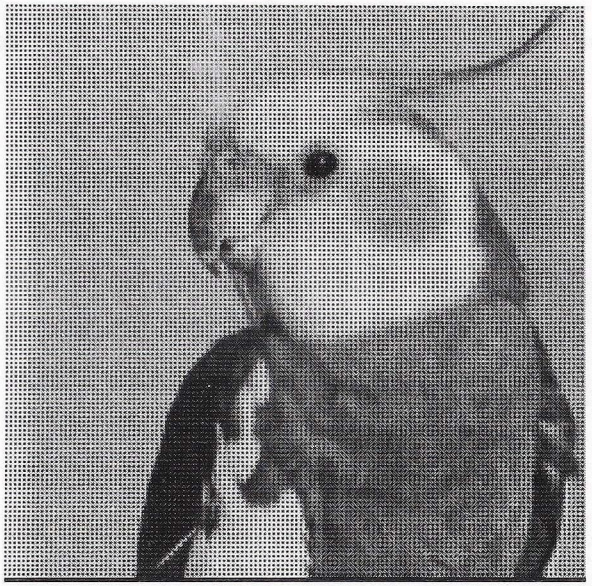


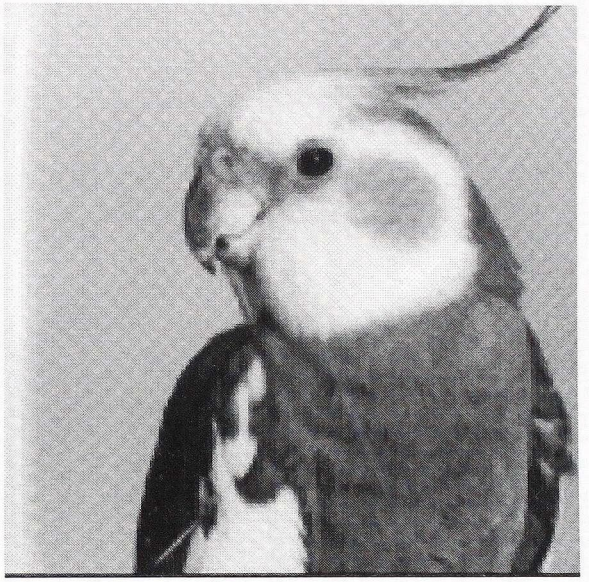
Figure 5.10: Learning curve for a 16–8–4 FANN pruned using the TOBD algorithm. The peaks indicate the first 5 pruning steps. The FANN has been trained for 100 epochs after each pruning step.

patterned background is to increase the number of the re–training epochs. However, this is time consuming. Another solution is to apply a post–processing stage after subsampling. For example, by simply filtering the subsampled image with a median filter using 2×2 image blocks, the image quality is greatly improved. This is illustrated in Figure 5.11 (*b*) which shows that most of the artifacts have been removed. The 512×512 image BIRD reconstructed by cubic interpolation using the subsampled and filtered image in Figure 5.11 (*b*) is illustrated in Figure 5.12. The reconstructed image has good subjective quality as well. We note that it is important that filtering be applied before the interpolation stage. Otherwise, the artifacts are accentuated and the quality of the reconstructed images is worse, even if a filtering stage is applied after interpolation.

The test PSNR values obtained by applying the FANN pruned using the TOBD



(a)



(b)

Figure 5.11: (*a*) Subsampled image BIRD by tridiagonally symmetrical FANN subsampling and (*b*) subsampled image BIRD by tridiagonally symmetrical FANN subsampling followed by median filtering using 2×2 blocks. The size of each subsampled image is 256×256.

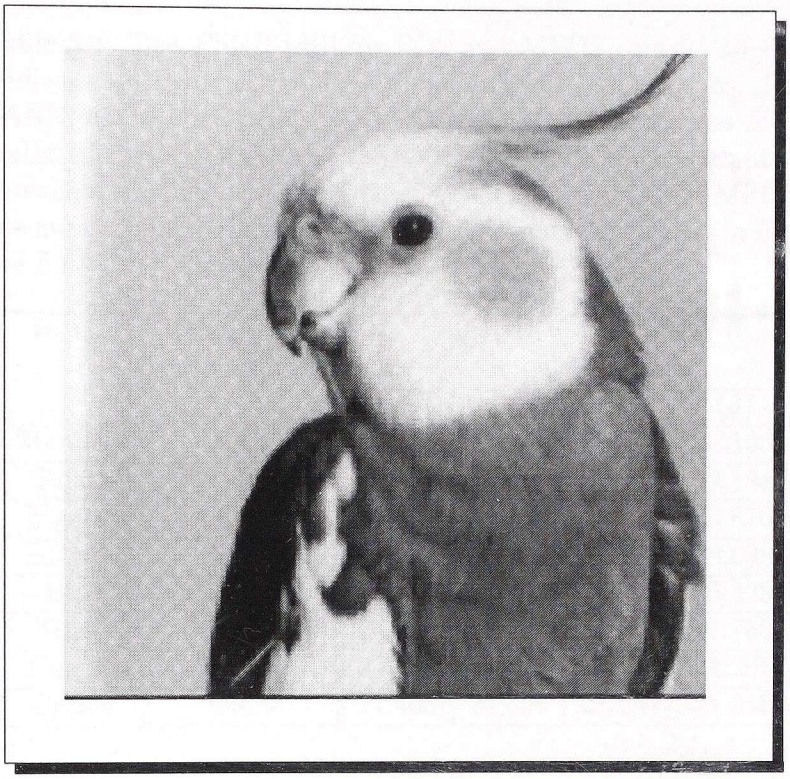


Figure 5.12: Reconstructed 512 × 512 image BIRD by cubic interpolation, using the subsampled image illustrated in Figure 5.11 (*b*).

algorithm to first–order image subsampling followed by median filtering and then reconstructing the subsampled images by cubic interpolation are included in the second column of Table 5.8. As expected, these values are lower (by an average of 1.777 dB) than those in Table 5.3 corresponding to the fully connected FANN. Note, however, that the tridiagonally symmetrical FANN has only 33 weights as compared to the fully connected FANN which has 160 weights.

When applying the tridiagonally symmetrical FANN to MFOS, several solutions to remove the background artifacts introduced by each of the FOS stages can

Table 5.8: Test PSNR [dB] for FOS and MFOS using a 16–8–4 tridiagonally symmetrical FANN and 512×512 still images. The FANN was trained using 256×256 regions of the images illustrated in Figure 5.5 (a) and the GFABS algorithm with adaptive threshold, and pruned using the TOBD algorithm. For MFOS, the notations (a)–(d) refer to the experiments illustrated in Figure 5.13.

Image	First–order subsampling	Multi–stage first–order subsampling			
		(a)	(b)	(c)	(d)
Mandrill	22.449	20.294	21.045	20.721	21.108
Boats	24.052	20.004	22.574	21.694	22.188
Fighter	30.344	20.443	25.119	23.674	23.876
Seismic	36.219	27.672	28.517	28.473	28.698
Bird	32.702	26.927	32.330	30.178	30.829
Truck	30.606	25.563	27.826	26.860	27.760
Couple	27.411	21.784	24.813	23.693	24.298
Crowd	29.688	21.405	24.596	23.264	24.046

be applied. Similarly to the FOS case, a higher number of re–training epochs after each pruning step may be selected. However, this is, again, a time consuming solution. Alternatively, a simple post–processing step such as median filtering can be applied after each of the FOS stages. Finally, the tridiagonally symmetrical FANN can be combined with a fully connected FANN in order to eliminate the artifacts. The latter solutions are illustrated in Figure 5.13. In Figure 5.13 (a), two identical FOS stages using the tridiagonally symmetrical FANN are being employed. As discussed above, median filtering using 2×2 image blocks is performed after each of the first–order subsampling stages. In Figure 5.13 (b) the first of the FOS stages is being performed using the tridiagonally symmetrical FANN. Then, the patterned

background is eliminated by applying median filtering. In the second subsampling stage, a fully connected FANN is applied. It is possible to reverse the order of these FOS stages, that is to first apply the FC−FANN, followed by the TS−FANN and the median filtering stages. This solution is illustrated in Figure 5.13 (*c*). Another option is that illustrated in Figure 5.13 (*d*), where a 16–8–4 TS−FANN and a 4–2–1 FC−FANN are applied to the image. In this case, the second FANN simultaneously eliminates the patterned background and subsamples the image. We note that no additional filtering stage is here needed.

The test PSNR values obtained in the experiments illustrated in Figure 5.13 are included in Table 5.8. For all of the test images, the highest PSNR values are obtained in the experiments illustrated in Figures 5.13 (*b*) and (*d*). All of the PSNRs in these two experiments are higher than those of the best LPFS method in Table 5.6. However, the results in the experiments (*b*) and (*d*) are also lower, by an average of 0.173 dB and 0.675 dB (respectively), than the results obtained using the FC−FANN (also included in Table 5.6). The results in the experiments (*a*) and (*c*) are lower, by an average of 3.014 dB and 1.206 dB (respectively), than those of the FC−FANN. This is expected since the number of weights of the TS−FANN is much smaller than that of the fully connected FANN. In Figures 5.14 and 5.15, which illustrate examples of subsampled and reconstructed images in the experiment shown in Figure 5.13 (*b*), the subjective image quality is clearly good, as well as in Figures 5.16 and 5.17, which illustrate examples of subsampled and reconstructed images in the experiment shown in Figure 5.13 (*d*). In the latter figure, the overall image quality is good, although some feature distortion is present in smooth areas such as the background.

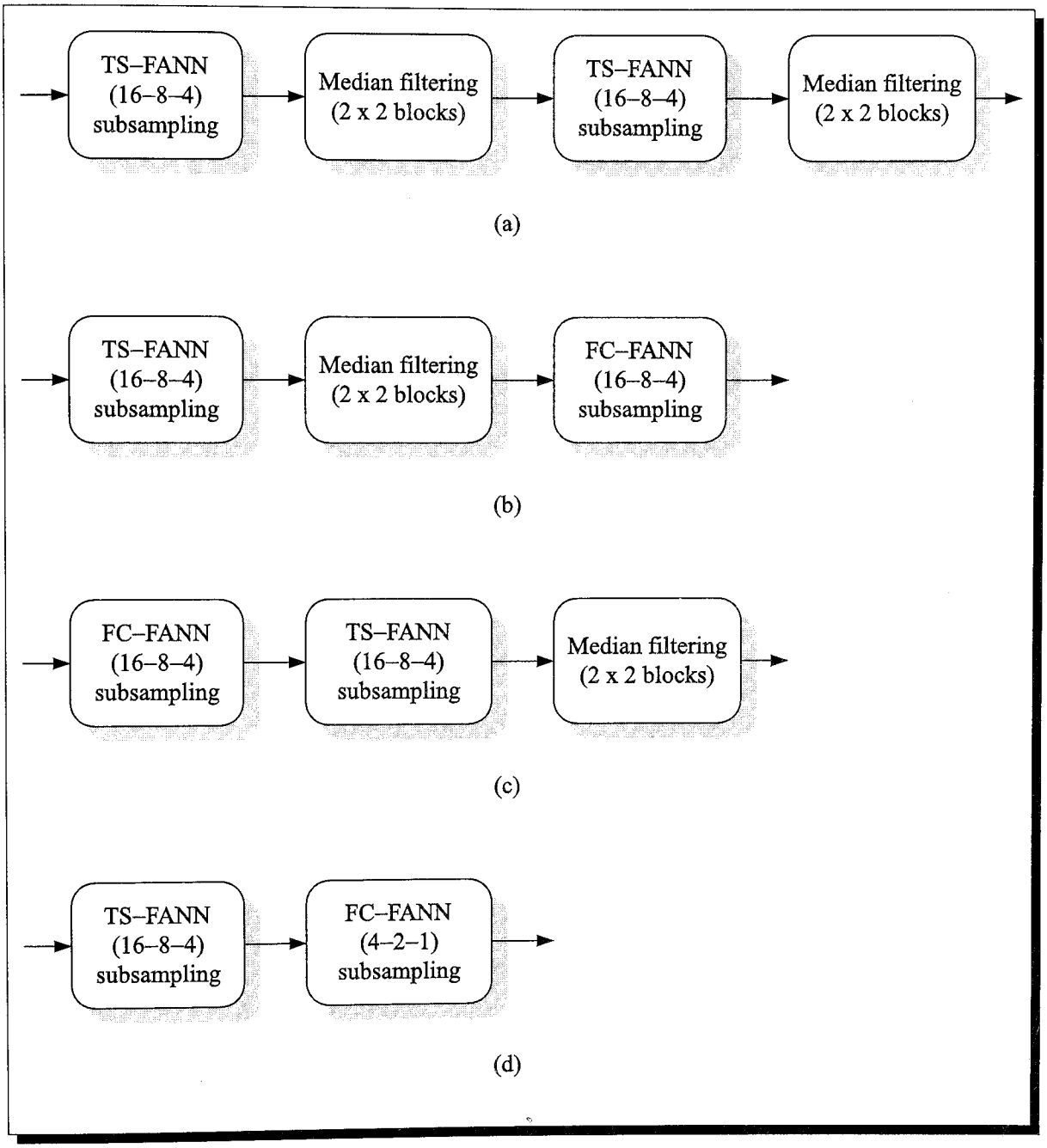


Figure 5.13: Multi–stage FOS options using pruned FANNs. The notations $FC - FANN$ and $TS - FANN$ stand for fully connected FANN and tridiagonally symmetrical FANN, respectively.

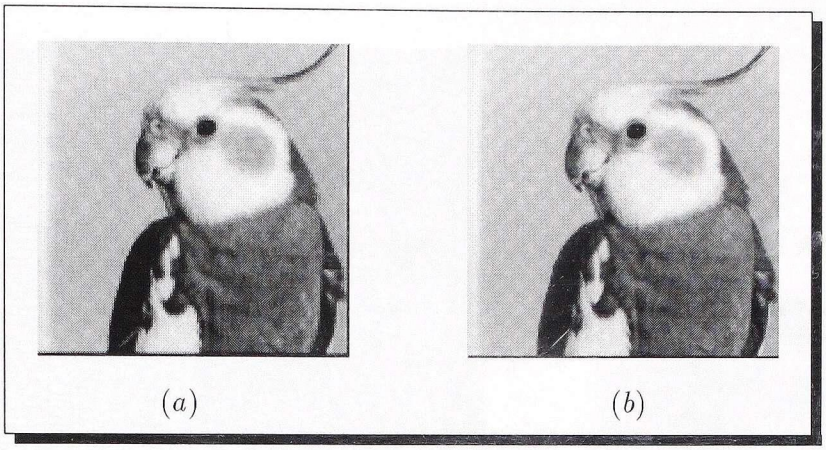


Figure 5.14: Subsampled image BIRD using (a) a 16–8–4 $TS-FANN$ and a $FC-FANN$ for each of the FOS stages, respectively. The image (b) was obtained by inserting a median filtering stage after the first FOS stage.

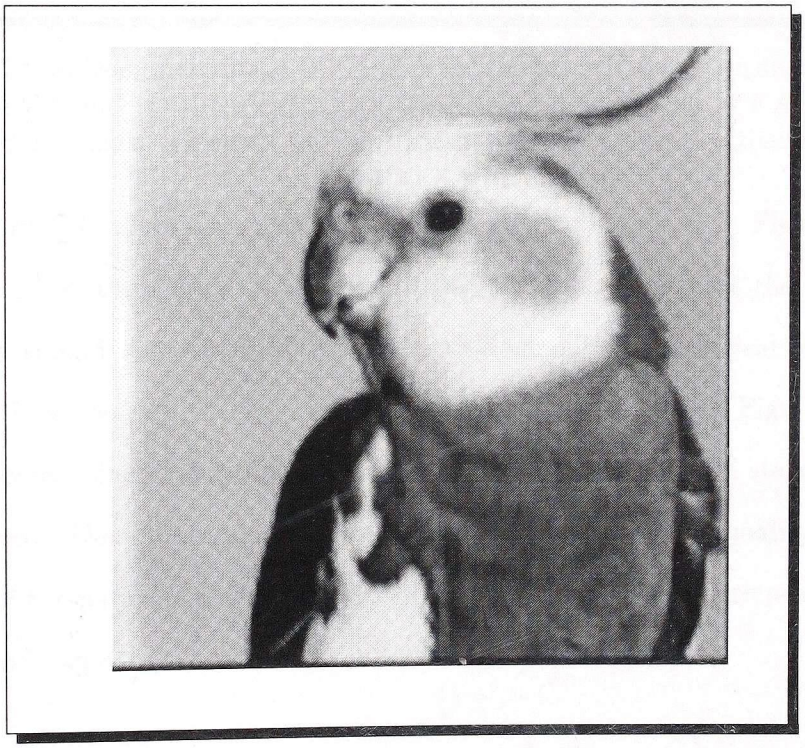


Figure 5.15: Reconstructed 512×512 image BIRD by cubic interpolation using the subsampled image illustrated in Figure 5.14 (b).

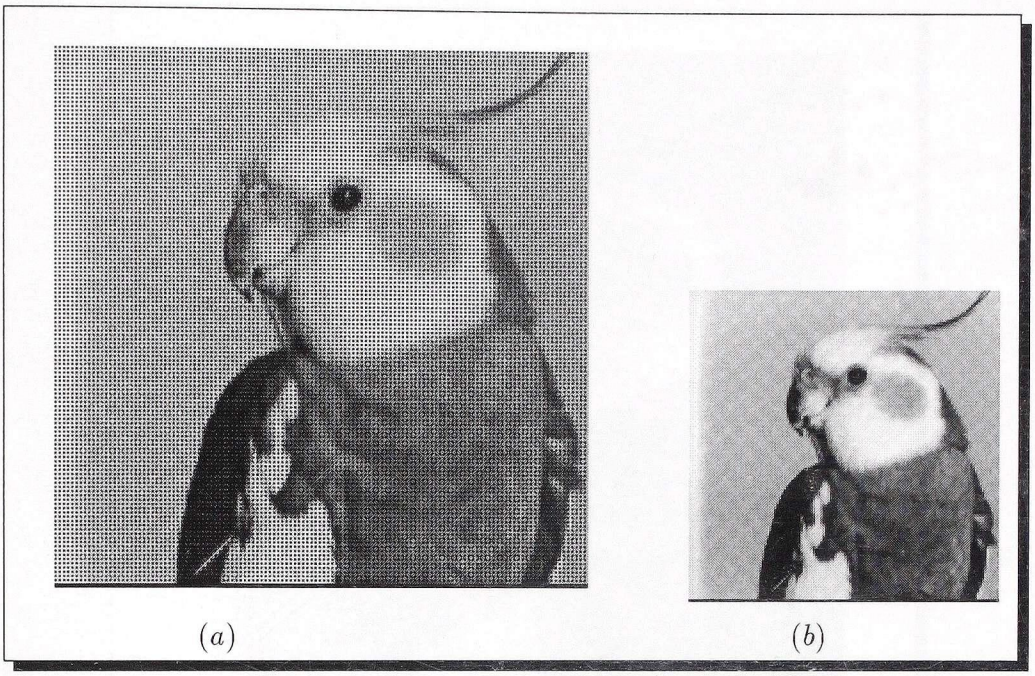


Figure 5.16: Subsampled image BIRD after (a) the first FOS stage using a 16–8–4 $TS - FANN$ and (b) the second FOS stage using a 4–2–1 $FC - FANN$. The sizes of these images are equal to 256×256 and 128×128, respectively.

The PSNR values obtained via the experiment illustrated in Figure 5.13 (c) are slightly lower than those in the experiment (b), suggesting that the first of the FOS stages should be performed using the tridiagonally symmetrical FANN. The lowest PSNR values are obtained in the experiment illustrated in Figure 5.13 (a). This is expected, since the artifacts that result after each of the FOS stages have not been removed. The subsampled images still have good subjective quality (with the exception of the patterned background), although visible artifacts are present in the interpolated images.

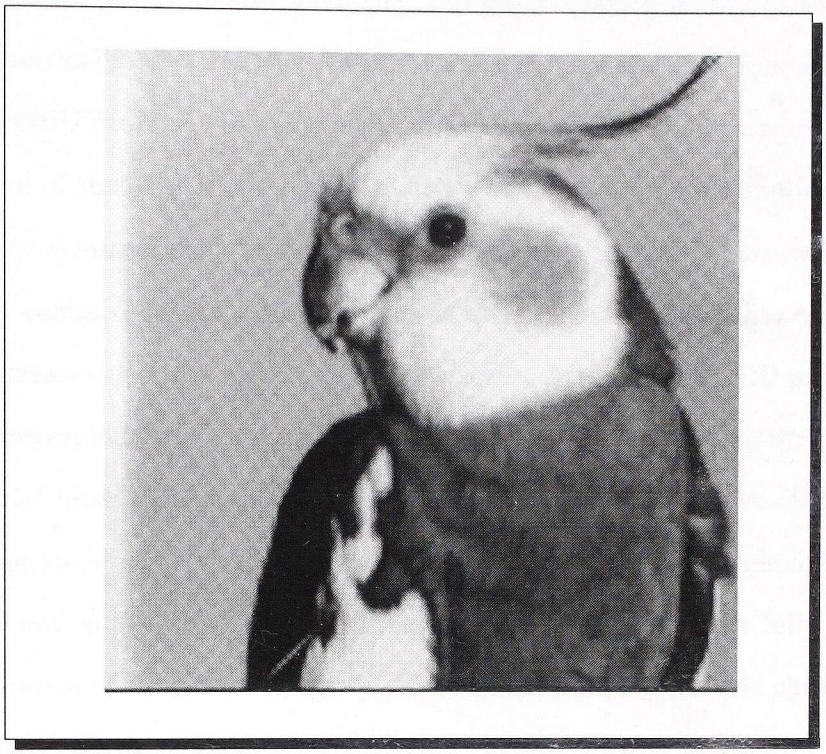


Figure 5.17: Reconstructed 512×512 image BIRD by cubic interpolation using the subsampled image illustrated in Figure 5.16 (*b*).

5.5 Discussion

In this section, we comment on important issues such as the generalization ability of the tridiagonally symmetrical FANN applied to first–order and high–order image subsampling and the computational demands in the testing (subsampling) stage.

5.5.1 FANN Generalization

Based on the results included in Table 5.8, showing that the PSNR values obtained in the experiments illustrated in Figure 5.13 are lower than those obtained by using

fully connected FANNs, it may seem that the generalization ability of the tridiagonally symmetrical FANN decreases after the pruning process. However, we note that after two TOBD pruning steps and their corresponding re–training stages, the PSNR values for all of the test images reconstructed after first–order subsampling using the tridiagonally symmetrical FANN are higher than those in Table 5.3, corresponding to the fully connected FANN. In other words, the generalization ability of the FANN actually increases after pruning some of the weights. After three TOBD pruning steps and their corresponding re–training stages, the PSNR values for all of the test images reconstructed after FOS slightly decrease as compared to the values obtained after two pruning steps. The test PSNR values after three pruning stages for all of the images are still higher than those in Table 5.3 corresponding to the fully connected FANN. In terms of subjective image quality, there are no perceivable changes in the reconstructed images.

Similarly, after two TOBD pruning steps and their corresponding re–training stages, the PSNR values for all of the test images reconstructed after applying the FANN twice are higher than those in Table 5.6 corresponding to the FC–FANN. After three TOBD pruning steps and their corresponding re–training stages, the PSNR values for all of the test images reconstructed after MFOS using the TS–FANNs slightly decrease as compared to the values obtained after two pruning steps. The test PSNR values after three pruning stages for all of the images are still higher than those in Table 5.6 corresponding to the fully connected FANN. In terms of subjective image quality, there are again no perceivable changes in the reconstructed images.

Clearly, the generalization ability of the FANN increases as a result of tridi-

agonally symmetrical pruning using our TOBD algorithm. However, note that the final, pruned FANN has only 33 weights, as compared to the fully connected FANN which has 160 weights. This motivates the loss in the PSNR values if many weights are deleted. In order to continuously increase the PSNR values, a higher number of FANN re–training epochs after each of the pruning steps would be required. This is not an efficient solution. Instead, we have shown that, by re–training the FANN for a small number of epochs after each of the pruning steps, good image quality is obtained when applying the solutions illustrated in Figure 5.13. The experimental results included in Section 5.4 suggest that the best solutions in terms of subjective image quality and PSNR values have been obtained using the systems illustrated in Figures 5.13 (b) and (d).

The generalization ability of the FANNs applied to image subsampling depends on the resolution of the training images. When the test images have resolutions equal or close to those of the training images (as in the case of most of our experiments), the FANN generalizes well. When the test images have resolutions which are very different from those of the training images, as shown in Table 5.4, good results have been obtained although the FANN generalization ability decreases. A solution to address this problem is to introduce some type of scale/resolution invariance in the subsampling process. This invariance can be introduced via the training set or via the training/subsampling algorithm. In the former case, different training sets for each of the FANNs employed in the FOS stages must be used. Each of these training sets consists of images having a resolution which is lower than that of the images in the training set employed for the preceding FANN. In the latter case, additional

processing stages must be employed in the training algorithm.

Finally, we note that we have evaluated both the subsampled and reconstructed images. As stated earlier, cubic interpolation (which is commonly used in many applications) has been employed in order to reconstruct the images. Of course, other types of interpolation may also be selected. Moreover, the interpolation process can be optimized using information about the subsampling algorithm. However, this has not been the focus of our work.

5.5.2 Computational Demands

As illustrated in Section 5.3.2, the fully connected FANN subsampling method is much faster than the LPFS methods and almost as fast as the MEDS method. Our goal was to further increase the speed of FANN subsampling by pruning the FANN with tridiagonally symmetry constraints. In order to asses the efficiency of the resulting TS−FANN with respect to that of the FC−FANN, we now evaluate the computational demands of the former when applied to FOS and MFOS.

The number of weights in the fully connected 16–8–4 FANN is equal to 160. The number of weights in the tridiagonally symmetrical 16–8–4 FANN is equal to 33. Therefore, the number of parameters that need to be saved decreases 4.85 times for the TS−FANN. Moreover, the subsampling time when using the 16–8–4 tridiagonally symmetrical FANN applied to first–order image subsampling is one fifth of that required by the 16–8–4 FC−FANN. Clearly, there is a substantial gain in terms of speed as compared to the fully connected FANN, which is especially desired for images having large sizes.

The number of weights that need to be saved in the MFOS experiments illustrated in Figures 5.13 (a), (b), (c) and (d) is equal to 33, 160, 160 and 43, respectively. Clearly, the number of weights in each of the cases (b) and (c) is equal to that of the fully connected FANN, since both the tridiagonally symmetrical FANN and the fully connected FANN are employed in different subsampling stages. In Figure 5.13 (d), only the parameters of the TS−FANN and those of the 4–2–1 FC−FANN must be saved. The CPU times required by the MFOS experiments (shown in Figure 5.13) are illustrated in Figure 5.18. The test images are, again, those listed in Table 5.4. Note that the lines corresponding to the fully connected FANN and to the median subsampler, which are the same as those having the smallest slopes in Figure 5.6, are the reference with respect to which the solutions (a)–(d) obviously provide higher subsampling speeds.

On average, the MFOS solutions illustrated in Figures 5.13 (a), (b), (c) and (d) are 2.48, 1.84, 1.11 and 3.30 times faster, respectively, than the MFOS using FC−FANNs (see Table 5.7). We note that, although the same number of weights is being used in the experiments (b) and (c), which is equal to that of a fully connected FANN, their subsampling speeds are different. First, the subsampling process in each of these cases is clearly faster than that employing two FC−FANNs. This is due to employing a tridiagonally symmetrical FANN in one of the FOS stages. Which of the FOS stages employs such an FANN has an impact on the subsampling speed. For instance, the subsampling process is faster in the experiment (b) than that in (c), since the tridiagonally symmetrical FANN is applied in the first FOS stage to the original (larger) image. The fastest solutions are provided by the solutions (d) and (a).

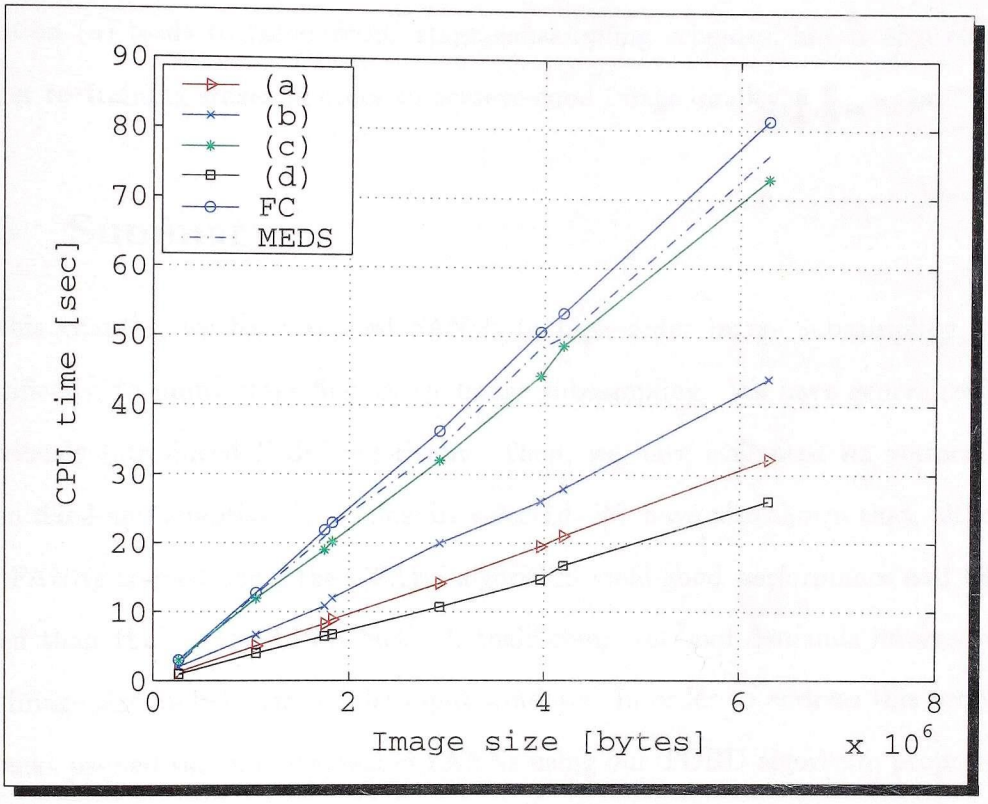


Figure 5.18: CPU times [sec] for MFOS on an UltraSparc 2 computer using a 16–8–4 TS–FANN. For HOS the notations (a)–(d) refer to the experiments illustrated in Figure 5.13. The acronyms FC and $MEDS$ denote the 16–8–4 fully–connected FANN and the median subsampler, respectively.

The best tradeoff is provided by the solution (d), which requires a small number of weights to be stored and which provides the highest speed and a consistently quality of the obtained images. Solution (b) also yields a good tradeoff, more specifically, a doubling of the subsampling speed and the best quality of the obtained images. Solution (a) leads to faster multi–stage subsampling schemes, but it also requires longer re–training stages in order to achieve good image quality.

5.6 Summary

In this chapter, we have applied FANNs to high–order image subsampling, more specifically, to multi–stage first–order image subsampling. We have generalized our previously introduced FABS algorithm. Then, we have evaluated its performance when fixed and adaptive thresholds are selected. We have also shown that, although our FANNs trained using the GFABS algorithm yield good performance and higher speed than that of the LPFS method, their computational demands increase with the image size and the size of the input windows. In order to address this problem, we next pruned the fully connected FANNs using our TOBD algorithm proposed in Chapter 3. The performance and computational demands of the resulting tridiagonally symmetrical structures have been evaluated. We have shown that, depending on the application requirements, different MFOS solutions can be employed, leading to a good subjective quality of the images. Moreover, the resulting systems are approximately 2 to 3.3 times faster than the systems employing the fully connected FANN.

Chapter 6

Conclusions and Future Work

6.1 Thesis Contributions

This thesis has addressed feedforward neural network design with application to image subsampling. Our particular focus has been on both the topological design of FANN structures and the FANN training algorithm, such that fast and compact FANNs, with good subjective and objective performance in first–order and high–order image subsampling are obtained. The main contributions of the thesis are: (1) an algorithm with tridiagonal symmetry constraints for FANN design, (2) a training algorithm for FANNs that leads to good performance in image subsampling, (3) the design, thorough performance and complexity evaluation of fully connected FANNs in FOS and MFOS, and (4) the design, thorough performance and complexity evaluation of tridiagonally symmetrical FANNs in FOS and MFOS.

We proposed the tridiagonally symmetrical Optimal Brain Damage (TOBD) algorithm, first introduced in [110, 112], to design TS–FANN structures. We have

illustrated, via a simulation example, that the number of the FANN parameters is reduced substantially by applying our algorithm, without a significant loss in performance. Moreover, the resulting TS−FANNs are fast, compact, and easy to implement.

Next, we proposed a new training algorithm for FANN structures when applied to image subsampling [111, 113, 114]. This second phase of our research was motivated by an important observation, that is the presence of blocking artifacts in the reproduced images when FANN–based image subsampling is generally performed. We have shown that, our proposed training algorithm addresses this problem by employing a pattern–matching method to extract, during FANN training, geometrical information from each processing window. Results for still images and video sequences were presented, showing the superior performance of the fully connected FANNs trained using our proposed algorithm in image subsampling. We have evaluated the results objectively, subjectively and in the context of a video coding application.

An important result which motivated the third stage of our research is that, despite the fully connected FANN structures being much more efficient than the low-pass filtering and subsampling methods, the testing (subsampling) time in high–order (particularly multi–stage first–order) image subsampling increases linearly with the image size. We therefore reduced the connectivities of our trained FANN structures applied to image subsampling using our FANN design algorithm [112]. We showed that the application of the resulting fast and compact FS−FANNs to FOS and MFOS yields very good tradeoffs in terms of performance and complexity. More specifically, the speed of our FANN–based subsampler increases 2 to 3.3 times while preserving a good quality of the resulting images. Thus, the TS−FANNs are more efficient than

both FC–FANNs and conventional subsampling methods, yielding the same or even better image quality. In all of our experiments, we evaluated image quality by using objective criteria such as the PSNR, and by subjective assessment of the reconstructed images.

6.2 Future Research Directions

The theoretical and practical advantages of our designed FANN structures suggest that further investigation and experimentation are worth pursuing. This dissertation is only a preliminary exploration of what these neural structures can offer. Among the possible research topics to extend and improve the work included in this thesis are:

1. Develop an extension of our design algorithm with tridiagonal symmetry constraints that takes into account a priori information about symmetries in the application problem. This can lead to application–optimized TS–FANNs.

2. Apply TS–FANNs to temporal and spatio–temporal subsampling of video sequences.

3. Address the problem of scale/resolution–dependence in FANN generalization abilities by including invariance to scale/resolution in the training algorithm.

4. Optimize the subjective/objective quality of the reconstructed images by designing an interpolation method that takes into account the particularities of the subsampling process.

Appendix A

The TOBD Algorithm

1. Train the network, until the error $C < \epsilon_6$, with ϵ_6 given. Save the input–hidden and the hidden–output weight matrices, that is,

   $\mathbf{W} = \{w_{ih},\, 1 \leq i \leq M,\, 1 \leq h \leq H\}$ and $\mathbf{V} = \{v_{hj},\, 1 \leq h \leq H,\, 1 \leq j \leq N\}$.

2. Compute the OBD saliencies.

3. IF the minimum saliency weight belongs to an input–hidden connection with weight $w_{i^* h^*}$, THEN

   3.1 IF $i^* \neq h^*$, THEN
   Prune $w_{i^* h^*}$ ($w_{i^* h^*} = 0$). Assign $m^* = \min(i^*, h^*)$.
   ELSE ($i^* = h^*$), assign $m^* = i^*$.
   END.
   For the diagonal element with index m^*, continue with Steps 3.2–3.6

   3.2 Build the $M \times M$ square matrix $\widetilde{\mathbf{W}}$ by padding $\mathbf{W}$ with zeros.

   3.3 Build the vector $\mathbf{q} = [q_1, q_2, \ldots, q_m]^T$ with q_i given by (3.2).

   3.4 Build the matrix $\Im$ using (3.1). Perform the transform $\Im\, \widetilde{\mathbf{W}}\, \Im^{-1}$ and normalize the elements of the resulting matrix.

   3.5 Re–train the network until the error is lower than ϵ_7.

   3.6 IF the weight matrix is not tridiagonal, GO TO Step 2; ELSE END.

   END.

4. IF the minimum saliency weight belongs to a hidden–output connection $v_{h^* j^*}$, with $1 \leq h^* \leq H$ and $1 \leq j^* \leq N$,

   IF $N > 1$, THEN
   GO TO Step 3.1 and apply the subsequent steps for the element $v_{h^* j^*}$.
   ELSE ($N = 1$)
   IF $h^* = \left\lceil \frac{H}{2} \right\rceil$ and H is even, THEN Prune $v_{h^* j^*}$ and $v_{H - h^* + 1, j^*}$.
   IF $h^* = \left\lceil \frac{H}{2} \right\rceil$ and H is odd, THEN Prune $v_{h^* j^*}$ and $v_{H - h^*, j^*}$.
   IF $h^* = \left\lceil \frac{H}{2} \right\rceil + 1$ and $\forall H$, THEN Prune $v_{h^* j^*}$.
   IF The weight matrix is not tridiagonal, GO TO Step 2; ELSE END.
   END.
   END.

Appendix B

The FABS Algorithm

REPEAT

1. FOR each input window (pattern) ξ, where $1 \leq \xi \leq P$ and P is the maximum number of patterns,

   (a) Compute the actual FANN output value $y(\xi)$ given by expression (4.1).

   (b) Compute the median of all possible three–pixel combinations, as illustrated in Figure 4.3 and compare it to the value of the fourth pixel in the window, yielding the values given by the expressions (4.2).

   (c) Compute the minimum $q^*(\xi)$ of the error values given by the expressions (4.2) for the current input window.

   (d) Set the desired output value to $d(\xi) = x^{(l^*)}(\xi)$, where l^* is the value of l for which the minimum in (4.3) is reached. The function $x^{(l^*)}(\xi)$ is equal to $x_{m,n}(\xi)$, $x_{m,n-1}(\xi)$, $x_{m+1,n-1}(\xi)$ or $x_{m+1,n}(\xi)$ if $l^* = 1, 2, 3$ or 4, respectively.

   END.

2. Compute the global error $C(\mathbf{w})$ at the end of one epoch by adding the squared errors $e(\xi)$ for all the input patterns.

3. Modify the weights according to a quasi–Newton rule.

UNTIL the stop condition is met.

Appendix C

The GFABS Algorithm with Adaptive Threshold

REPEAT

1. FOR each input image k

   (a) Compute the standard deviation σ_k of the image.

   (b) Divide the image into 4×4 windows.

   (c) FOR each window (unwrapped into a 16×1 pattern) ψ_k, where $1 \leq \psi_k \leq P_k$ and P_k is the number of patterns for the image k,

   (c1) Compute the actual FANN output value $y(\psi_k)$ given by expression (4.1).

   (c2) Divide the window into 2×2 blocks.

   (c3) FOR each block ξ of size 2×2 with $1 \leq \xi \leq 4$,

   - Compute the standard deviation $\beta_k(\xi)$ of the block ξ.
   - IF $\beta_k(\xi) > \sigma_k$ THEN an edge is present and
     - Compute the median of all possible three–pixel combinations, as illustrated in Figure 4.3. Compare it to the value of the fourth pixel in the window, yielding the values given by the expressions (4.2).
     - Compute the minimum $q^*(\xi)$ of the error values given by the expressions (4.2) for the current input window.
     - Set the desired output value to $d(\xi) = x^{(l^*)}(\xi)$, where l^* is the value of l for which the minimum in (4.3) is reached. The function $x^{(l^*)}(\xi)$ is equal to $x_{m,n}(\xi)$, $x_{m,n-1}(\xi)$, $x_{m+1,n-1}(\xi)$ or $x_{m+1,n}(\xi)$ if $l^* = 1, 2, 3$ or 4, respectively.

     ELSE the block is smooth. Set the desired output value $d(\xi)$ to the average of the four pixels in the block.
     END.

   END.

   (c4) Compute the FANN output error $e(\psi_k)$.

   END.

   END.

2. Compute the global error $C(\mathbf{w})$ at the end of one epoch by adding the squared errors for all the input patterns in all images.

3. Modify the weights according to a quasi–Newton rule.

UNTIL the stop condition is met.

Bibliography

[1] D. E. Rumelhart, J. L. McCLelland, and the PDP Research Group, *Parallel Distributed Processing. Explorations in the Microstructure of Cognition*, vol. 1 and 2. USA: The MIT Press, 1987.

[2] R. Hecht-Nielsen, *Neurocomputing.* USA: Addison–Wesley Publ. Co., 1991.

[3] R. Eberhart and R. Dobbins, *Neural Network PC Tools. A Practical Guide.* USA: Academic Press, Inc., 1990.

[4] C.-H. Lee, "Image surface approximation with irregular samples," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, no. 2, pp. 206–212, 1989.

[5] R. A. Belfor, M. P. Hesp, R. L. Lagendijk, and J. Biemond, "Spatially adaptive subsampling of image sequences," *IEEE Trans. on Image Processing*, vol. 3, pp. 492–500, Sept. 1994.

[6] C. A. Burton, L. Johnston, and E. Sonenberg, "An empirical investigation of thumbnail image recognition." Preprint, 1995.

[7] G. J. Grevera and J. K. Udupa, "Shape–based interpolation of nD grey scenes." Preprint, 1997.

[8] K. Rao and J. Hwang, *Techniques and Standards for Image, Video and Audio Coding.* USA: Prentice–Hall Inc., 1996.

[9] R. J. Clarke, *Digital Compression of Still Images and Video.* USA: Academic Press Ltd., 1995.

[10] P. Vaidyanathan, *Multirate Systems and Filter Banks.* USA: Prentice–Hall, 1993.

[11] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing.* USA: Prentice–Hall, 1983.

[12] E. Eviscito and J. P. Allebach, "The analysis and design of multidimensional FIR perfect reconstruction filter banks for arbitrary sampling lattices," *IEEE Trans. on Circuits and Systems*, vol. 38, pp. 29–41, Jan. 1991.

[13] H. Bischof, *Pyramidal Neural Networks.* USA: Lawrence Erlbaum Associates, Publ., 1995.

[14] N. Peterfreund and Y. Y. Zeevi, "Nonuniform image representation in area–of–interest systems," *IEEE Trans. on Signal Processing*, vol. 4, no. 9, pp. 1202–1212, 1995.

[15] S. Ranganath, "Image filtering using multiresolution representations," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, pp. 426–438, 1991.

[16] A. M. Tekalp, *Digital Video Processing.* USA: Prentice–Hall Inc., 1995.

[17] R. Manduchi, G. Cortelazzo, and G. Mian, "Multistage sampling structure conversion of video signals," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 3, pp. 325–340, Oct. 1990.

[18] R. Mohan, J. R. Smith, and C.-S. Li, "Adapting multimedia internet content for universal access," *IEEE Transactions on Multimedia*, vol. 1, pp. 104–114, Mar. 1999.

[19] C. Manning, "Introduction to digital video coding and block matching algorithms." Preprint, 1995.

[20] J. Hu, N. Sinaceur, *et al.*, "Removal of blocking and ringing artifacts in transform coded images," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, vol. 4, (Munich, Germany), pp. 2565–2568, 1997.

[21] R. Gonzalez and P. Wintz, *Digital Image Processing.* Menlo Park, CA: Addison–Wesley Publishing Company, 1987.

[22] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation.* USA: Addison–Wesley Publ. Co., 1991.

[23] M. S. B. Soucek, *Neural and Massively Parallel Computers. The Sixth Generation.* USA: John Wiley and Sons, 1988.

[24] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing.* USA: John Wiley and Sons, 1993.

[25] J. Zurada, *Introduction to Artificial Neural Systems.* USA: West Publishing Company, 1992.

[26] T. Poggio and F. Girosi, "Networks for approximation and learning," in *Proceedings of the IEEE*, vol. 79, IEEE, 1990.

[27] R. Beale and T. Jackson, *Neural Computing: An Introduction.* UK: Adam Hilger, 1991.

[28] A. Tsoi and A. Back, "Locally recurrent globally feedforward networks: A critical review of architectures," *IEEE Trans. on Neural Networks*, vol. 5, pp. 229–239, Mar. 1994.

[29] B. Dahanayake and R. Upton, "Learning with ease: Smart neural networks," *Proceedings of the International Conference on Neural Networks*, Nov. 1995.

[30] S. Fahlman and C. Lebiere, "The cascade–correlation learning architecture." Technical Report CMU–CS–90–100, School of Computer Science, Carnegie Mellon University, Pittsburgh, US, 1991.

[31] C. H. A. Maren and R. Pap, *Handbook of Neural Computing Applications.* USA: Academic Press, Harcourt Brace Jovanovich Publ., 1990.

[32] E. Fiesler, "Neural network classification and formalization," *Computer Standards and Interfaces*, vol. 16, 1994.

[33] S. Becker, *An Information–Theoretic Unsupervised Algorithm for Neural Networks.* PhD thesis, Graduate Dept. of Computer Science, Univ. of Toronto, Canada, 1992.

[34] S. Haykin, *Neural Networks. A Comprehensive Foundation.* USA: MacMillan College Publ. Co., 1994.

[35] Y. H. Hu, "Special topics in applications of neural networks to signal processing." Lecture notes, Dept. of Electrical and Computer Engineering, University of Wisconsin–Madison, 1994.

[36] E. Bartlett, "Dynamic node architecture learning: An information theoretic approach," *Neural Networks*, vol. 7, no. 1, pp. 129–140, 1994.

[37] E. Levin, N. Tishby, and S. A. Solla, "A statistical approach to learning and generalization in layered neural networks," *Proceedings of the IEEE*, vol. 78, pp. 1568–1573, Oct. 1990.

[38] P. Williams, "Bayesian regularization and pruning using a Laplace prior," *Neural Computation*, vol. 7, pp. 117–143, 1995.

[39] J. Larsen and L. Hansen, "Generalization performance of regularized neural network models," in *Proceedings of the International Conference on Artificial Neural Networks* (J. Vlontzos, J.N.Hwang, and E. Wilson, eds.), 1994.

[40] P. A. Jokinen, "A nonlinear network model for continous learning," *Neurocomputing*, vol. 3, pp. 157–176, Nov. 1991.

[41] B. Amirikian and H. Nishimura, "What size network is good for generalization of a specific task of interest," *Neural Networks*, vol. 7, no. 2, pp. 321–329, 1994.

[42] A. Doering, "Private Communication." Vancouver, Canada, 1995.

[43] Y. H. Hu, "Configuration of feedforward multilayer perceptron neural networks." Preprint, University of Wisconsin–Madison, Dept. of Electrical and Computer Engineering, 1993.

[44] E. Baum and D. Haussler, "What size net gives valid generalization ?," in *Advances in Neural Information Processing Systems* (D. Touretzky, ed.), vol. 1, San Mateo, US: Morgan Kaufmann, 1989.

[45] R. Lippmann, "An introduction to computing with neural nets," *IEEE Magazine on Acoustics, Signal and Speech Processing*, vol. 7, pp. 4–22, Apr. 1987.

[46] M. Lehtokangas, J. Saarinen, and P. Huuhta, "Neural network modeling and prediction of multivariate time series using predictive MDL principle," in *Proceedings of the International Conference on Artificial Neural Networks*, (Amsterdam, The Netherlands), 1993.

[47] M. Lehtokangas, J. Saarinen, and P. Huuhta, "Neural network prediction of nonlinear time series using predictive MDL principle," in *Proceedings of the IEEE Workshop on Nonlinear Digital Signal Processing*, (Tampere, Finland), pp. 7.2.2.1–7.2.2.6, 1993.

[48] H. Akaike, "Fitting autoregressive models for prediction," *Annals of the Institute of Statistical Mathematics*, vol. 21, pp. 243–247, 1989.

[49] B. Jansen, "Time series analysis by means of linear modelling," in *Digital Biosignal Processing* (R. Weitkunat, ed.), pp. 157–179, Elsevier Science Publ., 1991.

[50] M. Cottrell, B. Girard, Y. Girard, M. Mangeas, and C. Muller, "SSM: A statistical stepwise method for weight elimination," in *Proceedings of the International Conference on Artificial Neural Networks*, pp. 601–604, 1994.

[51] Y. Liu, "Unbiased estimate of generalization error and model selection in neural networks," *Neural Networks*, vol. 8, no. 2, pp. 215–219, 1995.

[52] N. Murata, S. Yoshizawa, and S. ichi Amari, "Network information criterion–determining the number of hidden units for an artificial network model." Preprint, Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo, 1992.

[53] E. Fiesler, "Comparative bibliography of ontogenic neural networks," in *Proceedings of the International Conference on Artificial Neural Networks*, pp. 793–796, 1994.

[54] T.-Y. Kwok and D.-Y. Yeung, "Constructive feedforward neural networks for regression problems: A survey." Technical Report HKUST-CS95-43, 1995.

[55] T. Nahban and A. Zomaya, "Toward generating neural network structures for function approximation," *Neural Networks*, vol. 7, no. 1, pp. 89–99, 1994.

[56] E. Alpaydin, "Grow and learn: An incremental method for category learning," in *Proceedings of the Intl. Neural Network Conference*, 1990.

[57] F. F. Mascioli, G. Martinelli, and G. Lazzaro, "Comparison of constructive algorithms for neural networks," in *Proceedings of the International Conference on Artificial Neural Networks*, pp. 731–734, 1994.

[58] M. Frean, "The Upstart algorithm: A method for constructing and training feedforward neural networks," *Neural Computation*, vol. 2, pp. 198–209, 1990.

[59] I. Sethi, "Entropy nets: From decision trees to neural networks," in *Proceedings of the IEEE*, vol. 78, pp. 1605–1613, 1990.

[60] R. Brent, "Fast training algorithms for multilayer neural nets," *IEEE Trans. on Neural Networks*, vol. 2, pp. 346–353, May 1991.

[61] N. Mirghafori, N. Morgan, and H. Bourlard, "Parallel training of multilayer perceptron estimators for speech recognition: A gender–based approach," *Proceedings of the 1994 IEEE Workshop on Neural Networks for Signal Processing*, pp. 289–298, 1994.

[62] J. Lange, H.-M. Voigt, and D. Wolf, "Task decomposition and correlations in growing artificial neural networks," in *Proceedings of the International Conference on Artificial Neural Networks*, pp. 735–738, 1994.

[63] M. Hintz-Madsen, L. K. Hansen, J. Larsen, *et al.*, "Design and evaluation of neural classifiers. Application to skin lesion classification," in *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pp. 484–493, 1995.

[64] J. Gorodkin, L. Hansen, A. Krogh, C. Svarer, and O. Winther, "A quantitative study of pruning by Optimal Brain Damage." Preprint, Electronics Institute, Technical University of Denmark, 1993.

[65] R. Reed, "Pruning algorithms–A survey," *IEEE Trans. on Neural Networks*, vol. 4, pp. 740–747, Sept. 1996.

[66] J. M. J.K. Kruschke, "Benefits of the gain: Speeded learning and minimal hidden layers in back propagation networks," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 21, pp. 273–280, Jan. 1991.

[67] J. Denker, Y. L. Cun, and S. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems* (D. Touretzky, ed.), vol. 2, San Mateo, USA: Morgan Kaufmann, 1990.

[68] T. Cibias, F. Soulie, P. Gallinari, and S. Raudys, "Variable selection with optimal cell damage," in *Proceedings of the International Conference on Artificial Neural Networks*, pp. 327–330, 1994.

[69] B. Hassibi and D. Stork, "Second order derivatives for network pruning: Optimal Brain Surgeon," in *Proc. of NIPS'93* (S. Hanson *et al.*, eds.), San Mateo, CA, US: Morgan Kaufmann Publ. Co., 1993.

[70] E. Karnin, "A simple procedure for pruning back–propagation trained neural networks," *IEEE Trans. on Neural Networks*, vol. 1, no. 2, pp. 239–242, 1990.

[71] L. Hansen, C. Rasmussen, C. Svarer, and J. Larsen, "Adaptive regularization," in *Proc. of the IEEE Workshop on Neural Networks for Signal Processing*, (Ermioni, Greece), 1994.

[72] M. P. L.K. Hansen, "Controlled growth of cascade correlation nets," in *Proceedings of the International Conference on Artificial Neural Networks*, pp. 797–780, 1994.

[73] Y. Chauvin, "A back–propagation algorithm with optimal use of hidden units," in *Advances in Neural Information Processing Systems* (D. Touretzky, ed.), vol. 1, San Mateo, CA, US: Morgan Kaufmann, 1989.

[74] D. P. C. Ji, R.R. Snapp, "Generalizing smoothness constraints from discrete samples," *Neural Computation*, vol. 2, no. 2, pp. 188–197, 1990.

[75] K. Hagiwara and S. Usui, "Numerical experiments on the information criteria for layered feedforward neural nets," in *Proceedings of the International Conference on Artificial Neural Networks*, vol. 1, pp. 493–496, 1994.

[76] L. Hansen and C. Rasmussen, "Pruning from adaptive regularization." Preprint, Electronics Institute, Technical University of Denmark, 1993.

[77] S. Yasui, A. Malinowski, and J. Zurada, "Convergence suppression and divergence facilitation: New approach to prune hidden layer and weights of feedforward neural networks," in *IEEE International Symposium on Circuits and Systems*, (Seattle, WA, US), pp. 121–124, 1995.

[78] M. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in *Advances in Neural Information Processing Systems* (D. Touretzky, ed.), vol. 1, San Mateo, CA, US: Morgan Kaufmann, 1989.

[79] Q. Xue, Y. Hu, and W. Tompkins, "Structural simplification of a feedforward, multilayer perceptron artificial neural network," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, 1991.

[80] S. Kung and Y. Hu, "A Frobenius approximation reduction method (FARM) for determining optimal number of hidden units," in *Proceedings of the IEEE Conference on Neural Networks*, vol. 2, (Seattle, WA, US), pp. 163–168, 1991.

[81] K. Y. Y. Hirose and S. Hijiya, "Backpropagation algorithm which varies the number of hidden units," *Neural Networks*, no. 4, pp. 61–66, 1991.

[82] A. Ivanhnenko, "The group method of data handling–A rival of stochastic approximation," *Soviet Automatic Control*, vol. 1, pp. 43–55, 1968.

[83] R. Barron, "Learning networks improve computer–aided prediction and control," *Computer Design*, vol. 1, pp. 65–70, Aug. 1975.

[84] B. Fritzke, "Growing cell structures, a self–organizing network for unsupervised and supervised learning," *Neural Networks*, vol. 7, pp. 1441–1460, Aug. 1994.

[85] D. T. Y.Q. Chen and M. Nixon, "Generating–shrinking algorithm for learning arbitrary classification," *Neural Networks*, vol. 7, pp. 1477–1489, Aug. 1994.

[86] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing.* Englewood Cliffs, NJ: Prentice Hall, 1989.

[87] D. E. Dudgeon and R. M. Mersereau, *Multidimensional Digital Signal Processing.* USA: Prentice–Hall Inc., 1984.

[88] J. Kovacevic and M. Vetterli, "FCO sampling of digital video using perfect reconstruction filter banks," *IEEE Trans. on Image Processing*, vol. 2, pp. 118–122, Jan. 1993.

[89] Y. Y. Zeevi and E. Shlomot, "Nonuniform sampling and antialiasing in image representation," *IEEE Trans. on Signal Processing*, vol. 41, pp. 1223–1236, Mar. 1994.

[90] A. Skodras and C. Christopoulos, "Downsampling of compressed images in the DCT domain," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, (Rhodos, Greece), pp. 1713–1716, 1998.

[91] J. Kovacevic and M. Vetterli, "The commutativity of up/downsampling in two dimensions," *IEEE Trans. on Information Theory*, vol. 37, pp. 695–698, May 1991.

[92] G. Karlsson and M. Vetterli, "Theory of two–dimensional multirate filter banks," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 38, pp. 925–937, June 1990.

[93] R. Manduchi, "Iterative inter–lattice interpolation," *Electronics Letters*, vol. 32, pp. 533–534, Mar. 1996.

[94] P. Delogne, L. Cuvelier, B. Maison, *et al.*, "Improved interpolation, motion estimation and compensation for interlaced pictures," *IEEE Trans. on Signal Processing*, vol. 3, no. 5, pp. 482–491, 1994.

[95] H. L. Floch and C. Labit, "A scattered data interpolation algorithm for still image subsampling and for motion field representations used to video coding," in *SPIE Proc. Visual Communications and Image Processing*, vol. 3024, pp. 635–645, 1997.

[96] P. Sathyanarayana, P. Reddy, and M. Swamy, "Interpolation of 2–D signals," *IEEE Trans. on Circuits and Systems*, vol. 37, pp. 623–625, May 1990.

[97] H.-M. Adorf, "Interpolation of irregularly sampled data series–A survey," in *Proc. of the Astronomical Data Analysis Software and Systems IV, ASP Conference Series*, vol. 77, 1995.

[98] M.-C. Hong, M. G. Kang, and A. K. Katsaggelos, "A regularized multichannel restoration approach for globally optimal high resolution video sequence," in *SPIE Proc. Visual Communications and Image Processing*, vol. 3024, pp. 1306–1316, 1997.

[99] J. Hawkins and R. Allen, *The Oxford Encyclopedic English Dictionary.* USA: Clarendon Press, Oxford, 1994.

[100] N. S. Cardell, W. H. Joerding, and Y. Li, "Symmetry constraints for feedforward network models of gradient systems," *IEEE Trans. on Neural Networks*, vol. 6, pp. 1249–1254, Sept. 1995.

[101] J. Shawe-Taylor, "Symmetries and discriminability in feedforward neural networks," *IEEE Trans. on Neural Networks*, vol. 4, pp. 816–826, Sept. 1993.

[102] R. Yang, L. Yin, M. Gabbouj, J. Astola, and Y. Neuvo, "Optimal weighted median filtering under symmetry constraints," *IEEE Trans. on Signal Processing*, vol. 43, pp. 591–603, Mar. 1995.

[103] V. K. Madisetti, *VLSI Digital Signal Processors. An Introduction to Rapid Prototyping and Design Synthesis.* US: IEEE Press, 1995.

[104] R. J. Higgins, *Digital Signal Processing in VLSI.* NJ, US: Prentice Hall, 1990.

[105] P. Lapsley, J. Bier, A. Shoham, and E. A. Lee, *DSP Processor Fundamentals.* US: IEEE Press, 1997.

[106] K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation.* US: John Wiley and Sons, 1998.

[107] H. H. Thodberg, "A review of Bayesian neural networks with an application to infrared spectroscopy," *IEEE Trans. on Neural Networks*, vol. 7, pp. 56–72, Jan. 1996.

[108] K. Diamantaras and S.-Y. Kung, "Multilayer neural networks for reduced–rank approximation," *IEEE Trans. on Neural Networks*, vol. 5, pp. 684–697, Sept. 1994.

[109] A. Ghorbani and V. Bhavsar, "Incremental communication for multilayer neural networks," *IEEE Trans. on Neural Networks*, vol. 6, no. 6, pp. 1375–1385, Nov. 1995.

[110] A. Dumitraş and F. Kossentini, "Feedforward neural network design with tridiagonal symmetry constraints." Accepted for publication in *IEEE Trans. on Signal Processing*, 1999.

[111] A. Dumitraş and F. Kossentini, "Fast and high performance image subsampling using feedforward neural networks." Accepted for publication in *IEEE Trans. on Image Processing*, 1999.

[112] A. Dumitraş and F. Kossentini, "Tridiagonally symmetrical pruning for the design of feedforward neural networks with application to high–order image subsampling." Submitted to *IEEE Trans. on Image Processing.*, August, 1999.

[113] A. Dumitraş and F. Kossentini, "Efficient FANN - based subsampling of images," (Hammamet, Tunisia), April 1 - 4, 1998. Proceedings of the *IEEE 2nd IMACS Multiconference: Computational Engineering in Systems Applications* (CESA'98), Vol. 4, pp. 649-653.

[114] A. Dumitraş and F. Kossentini, "FANN - based video chrominance subsampling," (Seattle, US), May 12 - 15, 1998. Proceedings of the *IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP'98), Vol. 2, pp. 1077 - 1080.

[115] A. Dumitraş, D. Liew, A. Jerbi, and F. Kossentini, "A Z - shaped nonlinear transform for image segmentation and classification in intelligent debris analysis," vol. 3, (Chicago, US), pp. 313–317, October 4 - 7, 1998. Proceedings of the *IEEE International Conference on Image Processing* (ICIP'98).

[116] B. Erol, A. Dumitraş, and F. Kossentini, *Emerging MPEG standards: MPEG-4 and MPEG-7.* US: Academic Press, in Handbook of Image and Video Processing, to appear in 1999.

[117] A. Dumitraş, A. Murgan, and V. Lăzărescu, "A quantitative study of evoked potential estimation using a feedforward neural network," in *Proc. of the 1994 IEEE Workshop on Neural Networks for Signal Processing*, (Ermioni, Greece), pp. 606–615, 1994.

[118] A. Dumitraş, A. Murgan, and V. Lăzărescu, "A growing - decreasing method for designing neural filters," in *Proc. of the 1994 IEEE Workshop on Nonlinear Signal and Image Processing*, vol. 2, (Neos Marmaras, Greece), pp. 579–582, 1995.

[119] A. Dumitraş, V. Lăzărescu, and A. Murgan, "On designing a neural filter with adaptive slope of the node's activation function," in *Proceedings of the European Conference on Circuit Theory and Design*, vol. 1, (Istanbul, Turkey), pp. 192–198, 1995.

[120] A. Dumitraş and V. Lăzărescu, "The influence of the MLP's output dimension on its performance in image restoration," in *IEEE International Symposium on Circuits and Systems*, vol. 1, (Atlanta, USA), pp. 329–332, 1996.

[121] A. Dumitraş, V. Lăzărescu, and M. Negoiţă, "Learning as multi–objective optimization in feedforward neural networks," in *Proc. of the First International Conference on Knowledge–Based Intelligent Electronic Systems (KES'97)*, vol. 1, (Adelaide, Australia), pp. 588–593, 1997.

[122] A. Dumitraş, C. Munteanu, and V. Lăzărescu, "Novel cost functions for neural classifiers," in *Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing (EUFIT'97)*, (Aachen, Germany), 1997.

[123] N. Memon, "An interband coding extension of the new lossless JPEG standard," in *SPIE Proc. Visual Communications and Image Processing*, vol. 3024, pp. 47–58, 1997.

[124] S. Mori, C. Y. Suen, and K. Yamamoto, "Historical review of OCR research and development," *Proceedings of IEEE*, vol. 80, no. 7, pp. 1029–1058, 1992.

[125] L. Spirkovska and M. B. Reid, "Connectivity strategies for higher–order nerual networks applied to pattern recognition," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, (San Diego, CA, US), pp. 21–26, 1990.

[126] L. Spirkovska and M. B. Reid, "Coarse–coded higher–order neural networks for PSRI object recognition," *IEEE Trans. on Neural Networks*, vol. 4, no. 2, pp. 276–283, 1993.

[127] L. Spirkovksa and M. B. Reid, "Method and system for pattern analysis using a coarse–coded neural network." US Patent 5333210, Serial number 908141, 1994.

[128] M. B. Reid, L. Spirkovska, and E. Ochoa, "Rapid training of higher–order neural networks for invariant pattern recognition," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, (Washington, DC, US), pp. 689–692, 1989.

[129] S. Kollias and D. Anastassiou, "A unified neural network approach to digital image halftoning," *IEEE Trans. on Signal Processing*, vol. 39, no. 4, pp. 980–984, 1991.

[130] C. B. Atkins, J. P. Allebach, and C. A. Bouman, "Halftone postprocessing for improved highlight rendition," in *Proceedings of the International Conference on Image Processing*, (Santa Barbara, CA), pp. 791–794, 1997.

[131] P. R. Bakic, N. S. Vujovic, D. P. Brzakovic, P. D. Kostic, and B. D. Reljin, "CNN paradigm based multilevel halftoning of digital images," *IEEE Trans. on Circuits and Systems – II: Analog and Digital Signal Processing*, vol. 4, no. 1, pp. 50–53, 1997.

[132] K. R. Crounse, T. Roska, and L. O. Chua, "Image halftoning with cellular neural network," *IEEE Trans. on Circuits and Systems – II: Analog and Digital Signal Processing*, vol. 40, no. 4, pp. 267–283, 1993.

[133] B. L. Shoop and E. K. Ressler, "An error diffusion neural network for digital image halftoning," in *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, (Cambridge, MA, US), pp. 427–436, 1995.

[134] T.-W. Yue and G.-T. Chen, "An auto–invertible neural network for image halftoning and restoration," in *Proceedings of the International Conference on Neural Networks*, vol. 3, (Perth, Australia), pp. 1450–1455, 1995.

[135] Y. Chigusa and K. Suzuki, "An image reconstruction system by neural network with median filter," in *IEEE International Symposium on Circuits and Systems*, vol. 3, (Chicago, US), pp. 2446–2449, 1993.

[136] I. Pitas and A. Venetsanopoulos, *Nonlinear Digital Filters.* USA: Kluwer Academic Publ., 1991.

[137] B. Schmitz and R. Stevenson, "Enhancement of subsampled chrominance image data," in *Proceedings of the World Congress on Neural Networks*, vol. 2, pp. 550–554, 1995.

[138] Telenor Research, "TMN (H.263) encoder/decoder, version 2.0," *TMN (H.263) codec*, June 1996.