# A MULTIRESOLUTION LEARNING METHOD FOR BACK-PROPAGATION NETWORKS

By

Wing-Chung Chan

A Dissertation

submitted in partial fulfillment of the requirements

for the Degree of Master of Philosophy

Division of Computer Science

The Chinese University of Hong Hong

June 1994

# Acknowledgement

As always, it is a great pleasure to acknowledge my debt to the many people involved, directly or indirectly, in the whole progress of this research.

I am very grateful to Dr. Lai-Wan Chan who guided me through the sinuous path of this research. She always provides me many useful suggestions and valuable information. I also wish to thank Prof. Tony Chan who introduced me the preliminary idea of using the multiresolution signal decomposition technique with neural networks. Finally, I want to express my special thanks to Charlotte Yu for her endless support during these two years.

To them all, I deeply appreciate and thank for their kindly help.

# Abstract

A multiresolution representation of a signal provides a simple hierarchical framework for interpretating the information. It is natural to first analyze the signal at a coarse resolution and then gradually increase the resolution. At a coarse resolution, the signal details are characterized by very few samples and the coarse information processing can be performed quickly. The finer details are characterized by more samples and thus take more operations to analyze. However, the prior information derived from the coarser resolution constrains and thus speeds up the computational time at finer resolution. It is believed that such coarse-to-fine strategy provides a possibility for reducing the computational cost of signal operations.

In this dissertation, we proposed to train a back-propagation network using the multiresolution learning method. This learning method involves a group of back-propagation networks. The objective of it is to improve the convergence rate and the recognition ability of the back-propagation networks.

By using the multiresolution signal decomposition technique, we first represent the original input vectors under different resolutions. A group of corresponding back-propagation networks are then built. Each of these networks is responsible to learn the input vectors at a particular resolution. The sequence of the training processes to be carried out by the group of back-propagation networks is from the coarsest level network to the finest level network sequentially. A term called the intermediate stopping criteria is defined for terminating the

training processes of these networks. After a network has been trained on a particular resolution of input vectors, the connection weights of the network are then transformed to the next finer level network. We have considered two different cases of the transformation, i.e., the transformation of connection weights between the input and the hidden layers and the transformation of connection weights between the hidden and the output layers.

# Contents


**1 Introduction** **1**

**2 Multiresolution Signal Decomposition** **5**

2.1 Introduction . . . . . 5

2.2 Laplacian Pyramid . . . . . 6

2.2.1 Gaussian Pyramid Generation . . . . . 7

2.2.2 Laplacian Pyramid Generation . . . . . 7

2.2.3 Decoding . . . . . 8

2.2.4 Limitation . . . . . 9

2.3 Multiresolution Transform . . . . . 9

2.3.1 Multiresolution Approximation of $L^2(R)$ . . . . . 9

2.3.2 Implementation of a Multiresolution Transform . . . . . 12

2.3.3 Orthogonal Wavelet Representation . . . . . 16

2.3.4 Implementation of an Orthogonal Wavelet Representation 18

2.3.5 Signal Reconstruction . . . . . 21

**3 Multiresolution Learning Method** **23**

3.1 Introduction . . . . . 23

3.2 Input Vector Representation . . . . . 24

3.2.1 Representation at the resolution 1 . . . . . 24

3.2.2 Representation at the resolution $2^j$ . . . . . 25

3.2.3 Border Problem . . . . . 26
3.3 Back-Propagation Network Architecture . . . . . 26
3.4 Training Procedure Strategy . . . . . 27
3.4.1 Sum Squared Error (SSE) . . . . . 28
3.4.2 Intermediate Stopping Criteria . . . . . 30
3.5 Connection Weight Transformation . . . . . 31
3.5.1 Weights between the Input and Hidden Layers . . . . . 31
3.5.2 Weights between the Hidden and Output Layers . . . . . 33

**4 Simulations** **36**
4.1 Introduction . . . . . 36
4.2 Choices of the Impulse Response $\tilde{h}(n)$ . . . . . 36
4.3 XOR Problem . . . . . 39
4.3.1 Setting of Experiments . . . . . 39
4.3.2 Experimental Results . . . . . 41
4.4 Numeric Recognition Problem . . . . . 50
4.4.1 Setting of Experiments . . . . . 50
4.4.2 Experimental Results . . . . . 52
4.5 Discussions . . . . . 72

**5 Conclusions** **75**

**A Proof of Equation (4.9)** **77**

**B Proof of Equation (4.11)** **79**

**Bibliography** **81**

# List of Tables


3.1 The structure of the back-propagation networks with 3 levels of resolutions. . . . . 27

4.1 The logic of XOR. . . . . 39

4.2 Configurations of each experiment for the XOR problem. . . . . 40

4.3 The structure of the back-propagation networks for the XOR problem. . . . . 41

4.4 Training results for the XOR problem after 5000 runs. . . . . 42

4.5 Configurations of each experiment for the numeric recognition problem. . . . . 51

4.6 The structure of the back-propagation networks for the numeric recognition problem. . . . . 52

4.7 Training results for the numeric recognition problem after 120 runs. 53

4.8 Recognition results for the 120 runs of the numeric recognition problem. . . . . 64

4.9 The last 10 training epoches of one of the training processes for Job 4 in Experiment 1. . . . . 73

# List of Figures


2.1 Discrete approximations $A^d_{2^j} f$ at the resolution 1, $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{8}$. Depending upon $2^{-j}n$, each dot gives the amplitude of the inner product $\langle f(u), \phi_{2^j}(u - 2^{-j}n)\rangle$. . . . . 15

2.2 Wavelet representation of the signal $A^d_1 f$. The dots give the amplitude of the inner products $\langle f(u), \psi_{2^j}(u - 2^{-j}n)\rangle$ of each detail signal $D_{2^j} f$ depending upon $2^{-j}n$. . . . . 19

2.3 Decomposition of a discrete approximation $A^d_{2^{j+1}} f$ into an approximation at a coarser resolution $A^d_{2^j} f$ and the signal detail $D_{2^j} f$. . . . . 20

2.4 Reconstruction of a discrete approximation $A^d_{2^{j+1}} f$ from an approximation at a coarser resolution $A^d_{2^j} f$ and the signal detail $D_{2^j} f$. . . . . 22

3.1 The training procedure for the multiresolution learning method with three levels of networks, $(B_{2^j})_{-2 \le j \le 0}$. . . . . 29

4.1 Comparisons of the training performance with different values of $\rho$ for the XOR problem. . . . . 43

4.2 Comparisons of the training performance with different values of $\delta$ for the XOR problem. . . . . 44

4.3 Comparisons of the training performance with different transformation schemes for the XOR problem of using the Daubechies' impulse response. . . . . 45
4.4 Comparisons of the training performance with different transformation schemes for the XOR problem of using the average splitting method. . . . . 46
4.5 Comparisons of the training performance with different sets of impulse response coefficients for the XOR problem of using the copy scheme. . . . . 47
4.6 Comparisons of the training performance with different sets of impulse response coefficients for the XOR problem of using the average scheme. . . . . 48
4.7 Comparisons of the training performance with different sets of impulse response coefficients for the XOR problem of using the scale scheme. . . . . 49
4.8 Comparisons of the training performance with different values of $\rho$ for the numeric recognition problem. . . . . 55
4.9 Comparisons of the training performance with different values of $\delta$ for the numeric recognition problem. . . . . 56
4.10 Comparisons of the training performance with different transformation schemes for the numeric recognition problem of using the Daubechies' impulse response. . . . . 57
4.11 Comparisons of the training performance with different transformation schemes for the numeric recognition problem of using the average splitting method. . . . . 58
4.12 Comparisons of the training performance with different sets of impulse response coefficients for the numeric recognition problem of using the copy scheme. . . . . 59

4.13 Comparisons of the training performance with different sets of impulse response coefficients for the numeric recognition problem of using the average scheme. . . . . 60
4.14 Comparisons of the training performance with different sets of impulse response coefficients for the numeric recognition problem of using the scale scheme. . . . . 61
4.15 Comparisons of the recognition results with different values of $\rho$ for the numeric recognition problem. . . . . 65
4.16 Comparisons of the recognition results with different values of $\delta$ for the numeric recognition problem. . . . . 66
4.17 Comparisons of the recognition results with different transformation schemes for the numeric recognition problem of using the Daubechies' impulse response. . . . . 67
4.18 Comparisons of the recognition results with different transformation schemes for the numeric recognition problem of using the average splitting method. . . . . 68
4.19 Comparisons of the recognition results with different sets of impulse response coefficients for the numeric recognition problem of using the copy scheme. . . . . 69
4.20 Comparisons of the recognition results with different sets of impulse response coefficients for the numeric recognition problem of using the average scheme. . . . . 70
4.21 Comparisons of the recognition results with different sets of impulse response coefficients for the numeric recognition problem of using the scale scheme. . . . . 71

# Chapter 1

# Introduction

The back-propagation network has been studied for many years and many researchers have applied it to a wide variety of problems successfully, e.g., pattern recognition, function approximation, control application, and optimization formulation [Hert91], [Kung93], [Rume86].

Unfortunately, it is shown that the back-propagation algorithm, which adopts the steepest descent technique, is slow to converge in a multilayer network [Hush93], [Jaco88]. There are plenty of work to improve the convergence rate of the networks but most of them do increase the computational complexity tremendously [Hush88], [Watr87]. Such limitation prohibits the use of back-propagation networks on the large scale problems, e.g., problems with high dimensionality input space.

Another problem with the multilayer perceptron is that it assumes the individual input neuron acts independently from the other neurons. In fact, in some problems, for example, image recognition problems, use images as the grey level input to the network. The input neurons do have some correlations with their neighboring neurons. However, a multilayer perceptron has not taken this into account.

On the other hand, the human visual system, as an optimal image processor,

can process a huge amount of information quickly. Studies of such system have shown that the retina of the human eye is neither a single receptor aimed by the eyeball muscles nor a uniform array of parallel receptors, but an structured array so as to see a wide angle in a low-resolution way using peripheral vision, while simultaneously allowing high-resolution, detailed perception by the fovea in a small central portion of the viewing region [Levi85]. This finding triggered significant interest in multiresolution signal decomposition and some researchers [Burt83], [Rose84a], [Tani75] have applied this multiresolution technique in many fields of applications, e.g., edge detection, data compression, surface interpolation, and shape analysis.

Recently, several researchers incorporate this technique with neural networks. Yhann and Young [Yhan90] have constructed a multiresolution pyramid representation of a set of sample images and edge features at each level in the pyramid. Using the line and edge features at a chosen scale, a neural network is trained to classify the different textures. Segmentation of the scaled image is then accomplished using the trained network. Evans, Ellacott and Hand [Evan91] have developed a neural network eye detector by combining three simple processes, a multiresolution pyramid, problem decomposition and neural networks. The problem decomposition simplifies the problem domain by breaking the overall goal into smaller goals. This link into the both the multiresolution search and the use of neural networks as the classifiers. Sabourin and Mitiche [Sabo93] have proposed the use of a Kohonen associative memory with selective multiresolution for the task of modeling and classification of shapes and such method is shown to be computationally efficient.

A multiresolution representation of a signal provides a simple hierarchical framework for interpretating the information [Koen84]. In some sense, the signal at a coarse resolution provide the "context" of the signal whereas the finer

resolutions correspond to the particular "modalities". It is natural to first analyze the signal at a coarse resolution and then gradually increase the resolution. This is called a coarse to fine processing strategy. At a coarse resolution, the signal details are characterized by very few samples and the coarse information processing can be performed quickly. The finer details are characterized by more samples and thus take more operations to analyze. However, the prior information derived from the context constrains and thus speeds up the computational time at finer resolutions. It is believed that such coarse-to-fine strategy provides a possibility for reducing the computational cost of signal operations [Rose84b].

In this dissertation, we propose a problem-independent learning method for the back-propagation networks. This learning method adopts the multiresolution signal decomposition technique in order to alleviate the shortcomings of this kind of networks described above. This learning method involves a group of back-propagation networks. The objective of it is to improve the convergence rate and the recognition ability of the networks.

With this multiresolution learning method, the original input vectors for the back-propagation network are represented under different resolutions. A group of corresponding back-propagation networks are then built and each of these networks is responsible to learn the input vectors at a particular resolution. The sequence of the training processes to be carried out by the group of back-propagation networks is from the coarsest level network to the finest level network sequentially. After a network has been trained on a particular resolution of input vectors, the connection weights of the network is then transformed to the next finer level network.

In Chapter 2, we describe the basic concept and the mathematical properties of multiresolution signal decomposition. Two types of multiresolution representations, the Laplacian pyramid and the multiresolution transform, are presented in this chapter also. We then introduce our proposed multiresolution

learning method in Chapter 3. In this chapter, we show how the input vectors are represented under different resolutions, the architecture of the group of back-propagation networks generated, the training procedure strategy, and how the connection weights are transform from one network to another one. In Chapter 4, we show some simulation results of our learning method. Two different problems are simulated. They are the XOR problem and the numeric recognition problem respectively. Finally, we conclude the dissertation and discuss our contributions in Chapter 5.

# Chapter 2

# Multiresolution Signal Decomposition

## 2.1 Introduction

Multiresolution signal analysis has been thoroughly studied in computer vision since the work of Tanimoto and Pavlidis [Tani75] on hierarchical data structure for picture processing, and the Marr theory of low-level vision [Marr82]. Many researchers [Baaz90], [Rose84a], [Szel90], [Unse89] have applied this multiresolution technique in many fields of applications, e.g., edge detection, data compression, surface interpolation, and shape analysis.

A multiresolution decomposition can be regarded as a model of certain types of early processing in natural vision and allows us to have a scale-invariant interpretation of the signal [Burt84]. Given a sequence of increasing resolutions $(r_j)_{j \in Z}$, a scaling invariance can be obtained in a multiresolution representation if $(r_j)_{j \in Z}$ varies exponentially. That is, there exists a resolution step $\alpha \in R$ such that for all integers $j$, $r_j = \alpha^j$.

A multiresolution representation provides a simple hierarchical framework for interpretating the signal information [Koen84]. In some sense, the details of

the image at a coarse resolution provide the "context" of the image whereas the finer details correspond to the particular "modalities". For example, it is difficult to recognize that a small rectangle inside an image is the window (modality) of a house if we did not previously recognize the house (context). Therefore, it is natural to first analyze the signal at a coarse resolution and then gradually increase the resolution. This is called a coarse to fine processing strategy. At a coarse resolution, the signal details are characterized by very few samples and the coarse information processing can be performed quickly. The finer details are characterized by more samples and thus take more operations to analyze. However, the prior information derived from the context constrains and thus speeds up the computational time at finer resolutions. It is believed that such coarse-to-fine strategy provides a possibility for reducing the computational cost of signal operations [Rose84b].

In this chapter, the concept of multiresolution signal decomposition is introduced. We first study the early work of Burt [Burt83] and Crowley [Crow84] on the Laplacian pyramid coding. In which the resulting representations, which form a self-similar structure, are localized in both space and spatial frequency. We then review the multiresolution transform studied by Mallat [Mall89a], [Mall89b], [Mall89c]. He showed that the difference of information between the approximation of a signal at the resolutions $2^{j+1}$ and $2^j$ can be extracted by decomposing this signal on a wavelet orthonormal basis of $L(R^n)$.

## 2.2 Laplacian Pyramid

In computer vision, a common characteristic of images is that neighboring pixels are highly correlated. To represent the image directly in terms of the pixel values is therefore inefficient, i.e., most of the encoded information is redundant. For this purpose, Burt [Burt83] and Crowley [Crow84] have each developed efficient

algorithms to find a representation which decorrelates the image pixels. Such representation is called the Laplacian pyramid and is actually a set of details appearing at different resolutions. Given a sequence of increasing resolutions $(r_j)_{j\in Z}$, the details of an image at the resolution $r_j$ are defined as the difference of information between its approximation at the resolution $r_j$ and its approximation at the lower resolution $r_{j-1}$.

### 2.2.1 Gaussian Pyramid Generation

The first step in the Laplacian pyramid coding is to low-pass filter the original image $g_0$ to obtain an image $g_{-1}$. $g_{-1}$ is a "reduced" version of $g_0$ in that both resolution and sample density are decreased. In a similar way, $g_{-2}$ is formed as a reduced version of $g_{-1}$, and so on. Filtering is performed by a procedure equivalent to convolution with one of a family of local, symmetric weighting functions. An important member of this family resembles the Gaussian probability distribution, so the sequence of images $(g_0, g_{-1}, \cdots, g_{-n})$ is called the Gaussian pyramid. Mathematically, for $-N < j < 0$, $0 \leq x < C_j$, and $0 \leq y < R_j$,

$$g_j(x,y) = \sum_{m=-M}^{M} \sum_{n=-M}^{M} h(m,n) g_{j+1}(2x-m, 2y-n) \tag{2.1}$$

where $h$ is a low-pass filter. Here $N$ refers to the number of levels in the pyramid, while $C_j$ and $R_j$ are the dimensions of the $j$th level.

### 2.2.2 Laplacian Pyramid Generation

The Laplacian pyramid is a sequence of image details $(L_0, L_{-1}, \cdots, L_{-N+1})$. Each is the difference between two levels of the Gaussian pyramid. Thus, for $-N < j \leq 0$,

$$L_j(x,y) = g_j(x,y) - g_{j-1,1}(x,y) \tag{2.2}$$

where $g_{j-1,1}$ is the interpolated level $g_{j-1}$ by a factor of 2 along each axis [Croc81] and is defined as

$$g_{j,n}(x,y) = 4 \sum_{m=-M}^{M} \sum_{n=-M}^{M} h(m,n) g_{j,n-1}(\frac{x+m}{2}, \frac{y+n}{2}) \tag{2.3}$$

and

$$g_{j,0}(x,y) = g_j(x,y). \tag{2.4}$$

Since there is no image $g_{-N}$ to serve as the prediction image for $g_{-N+1}$, we set

$$L_{-N+1} = g_{-N+1}. \tag{2.5}$$

### 2.2.3 Decoding

The Laplacian pyramid is found to be a complete coding and it can be shown that the original image can be reconstructed exactly by expanding, then summing all the levels of the Laplacian pyramid like

$$g_0 = \sum_{j=-N+1}^{0} L_{j,-j}. \tag{2.6}$$

**Proof.** By interpolating both sides of (2.2) and (2.5) with $-j$ times ($j < 0$), we get

$$L_{j,-j} = \begin{cases} g_{j,-j} - g_{j-1,-j+1} & \text{if } j \neq -N+1 \\ g_{j,-j} & \text{if } j = -N+1 \end{cases}$$

$$\begin{aligned} \sum_{j=-N+1}^{0} L_{j,-j} &= \sum_{j=-N+1}^{0} g_{j,-j} - \sum_{j=-N+2}^{0} g_{j-1,-j+1} \\ &= g_0 + \sum_{j=-N+1}^{-1} g_{j,-j} - \sum_{j=-N+1}^{-1} g_{j,-j} \\ &= g_0. \end{aligned}$$

□

A more efficient procedure is to expand $L_{-N+1}$ once and add it to $L_{-N+2}$, then expand this image once and add it to $L_{-N+3}$, and so on until level 0 is reached and $g_0$ is recovered. This procedure simply reverses the steps in the Laplacian pyramid generation.

### 2.2.4 Limitation

This algorithm can decompose an image $g_0$ at a resolution of 1 into an approximation $g_{-J}$ at a coarse resolution $4^{-J}$ and the successive details $(L_j)_{-J<j\leq 0}$.[1] If the image $g_0$ has $A^2$ pixels, each details $L_j$ has $4^j A^2$ samples. Hence, the total number of samples of this representation is approximately $\frac{4}{3}A^2$. Therefore, this simple and elegant algorithm does not define the details from the difference of information between $g_{j+1}$ and $g_j$ [Mall89b]. If it did, the total number of samples representing the image would be the same as in the original image. At different resolutions, the details computed with this algorithm are correlated. It is thus difficult to know whether a similarity between the signal details at different resolutions is due to a property of the signal itself or to the intrinsic redundancy of the representation.

## 2.3 Multiresolution Transform

In this section, we study the basic concept of multiresolution analysis introduced by Mallat [Mall89a], [Mall89b]. The mathematical properties of the operators which transform a function into an approximation at a resolution $2^j$ are presented. The proofs of the theorems and the equations are omitted here and the mathematical foundations are more thoroughly described in [Mall89c].

### 2.3.1 Multiresolution Approximation of $L^2(R)$

Suppose that the original signal $f(x)$ described in this section is measurable and has a finite energy: $f(x) \in L^2(R)$. According to Mallat's definition [Mall89b], we can define the multiresolution approximation of $L^2(R)$.

**Definition.** The approximation of a signal $f(x)$ at a resolution $r$ can be defined as an estimate of $f(x)$ derived from $r$ measurements per unit length. These

[1] In this case, the resolution step $\alpha$ is equal to 4.

measurements are computed by uniformly sampling at a rate $r$ the function $f(x)$ smoothed by a low-pass filter whose bandwidth is proportional to $r$. □

In an approximation operation, when removing the details of $f(x)$ smaller than $r$, the highest frequency of this function is suppressed. In the following, we study only the approximation of a function on a dyadic sequence of resolution $(2^j)_{j\in Z}$.

Let $A_{2^j}$ be the operator which approximates a signal at a resolution $2^j$ and $V_{2^j} \subset L^2(R)$ be the set of all possible approximations at the resolution $2^j$ of functions in $L^2(R)$ and the set of vector space $(V_{2^j})_{j\in Z}$ is called a multiresolution approximation of $L^2(R)$. Then, $A_{2^j}$ is characterized with the following properties [Mall89a]:

- $A_{2^j} \circ A_{2^j} = A_{2^j}$. If $A_{2^j} f(x)$ is the approximation of some function $f(x)$ at the resolution $2^j$, then $A_{2^j} f(x)$ is not modified if we approximate it again at the resolution $2^j$.

- $\forall g(x) \in V_{2^j}$, $\|g(x) - f(x)\| \geq \|A_{2^j} f(x) - f(x)\|$. Among all the approximated functions at the resolution $2^j$, $A_{2^j} f(x)$ is the function which is the most similar of $f(x)$.

- $A_{2^j}$ is an orthogonal projection on $V_{2^j}$.

And the properties of $V_{2^j}$ include:

- $\forall j \in Z$, $V_{2^j} \subset V_{2^{j+1}}$. The approximation of a signal at a resolution $2^{j+1}$ contains all the necessary information to compute the same signal at a smaller resolution $2^j$.

- $\forall j \in Z$, $f(x) \in V_{2^j} \Leftrightarrow f(2x) \in V_{2^{j+1}}$. An approximation operation is similar at all resolution. The spaces of approximated functions should

thus be derived from one another by scaling each approximated function by the ratio of their resolution values.

- Discrete characterization: There exists an isomorphism $I$ from $V_1$ onto $l^2(Z)$.
- Translation of the approximation: $\forall k \in Z$, $A_1 f_k(x) = A_1 f(x-k)$, where $f_k(x) = f(x-k)$.
- Translation of samples: $I(A_1 f(x)) = (\alpha_i)_{i \in Z} \Leftrightarrow I(A_1 f_k(x)) = (\alpha_{i-k})_{i \in Z}$.
- $\bigcup_{j=-\infty}^{+\infty} V_{2^j}$ is dense in $L^2(R)$. When computing an approximation of $f(x)$ at resolution $2^j$, some information about $f(x)$ is lost. However, as the resolution increases to $+\infty$ the approximated signal should converge to the original signal.
- $\bigcap_{j=-\infty}^{+\infty} V_{2^j} = \{0\}$. Conversely as the resolution decreases to zero, the approximated signal contains less and less information and converge to zero.

As the approximation operator $A_{2^j}$ is an orthogonal projection on the vector space $V_{2^j}$, an orthonormal basis of $V_{2^j}$ must be found in order to numerically characterize $A_{2^j}$. The following theorem shows that such an orthonormal basis can be defined by dilating and translating a unique function $\phi(x)$.

**Theorem 2.1** *Let $(V_{2^j})_{j \in Z}$ be a multiresolution approximation of $L^2(R)$. There exists a unique function $\phi(x) \in L^2(R)$, called a scaling function, such that if we set $\phi_{2^j}(x) = 2^j \phi(2^j x)$ for $j \in Z$ (the dilation of $\phi(x)$ by $2^j$), then $\left(\sqrt{2^{-j}}\phi_{2^j}(x - 2^{-j}n)\right)_{n \in Z}$ is an orthonormal basis of $V_{2^j}$.* □

The theorem shows that an orthonormal basis of any $V_{2^j}$ can be built by dilating a function $\phi(x)$ with a coefficient $2^j$ and translating the resulting function on a grid whose interval is proportional to $2^{-j}$. The functions $\phi_{2^j}(x)$ are

normalized with respect to the $L^1(R)$ norm. The coefficient $\sqrt{2^{-j}}$ appears in the basis set in order to normalize the functions in the $L^2(R)$ norm. For a given multiresolution approximation $(V_{2^j})_{j\in Z}$, there exists a unique scaling function $\phi(x)$ which satisfies the above theorem. However, for different multiresolution approximations, the scaling functions are different.

The orthogonal projection on $V_{2^j}$ can then be computed by decomposing the signal $f(x)$ on the orthonormal basis given by Theorem 2.1. Specifically, $\forall f(x) \in L^2(R)$,

$$A_{2^j} f(x) = 2^{-j} \sum_{n=-\infty}^{+\infty} \left\langle f(u), \phi_{2^j}\left(u - 2^{-j}n\right)\right\rangle \phi_{2^j}\left(x - 2^{-j}n\right). \tag{2.7}$$

The approximation of the signal $f(x)$ at the resolution $2^j$, $A_{2^j} f(x)$, is thus characterized by the set of inner products as

$$A^d_{2^j} f = \left(\left\langle f(u), \phi_{2^j}\left(u - 2^{-j}n\right)\right\rangle\right)_{n\in Z} \tag{2.8}$$

and $A^d_{2^j} f$ is called a discrete approximation of $f(x)$ at the resolution $2^j$.

Since $\phi(x)$ is a low-pass filter, this discrete signal can be interpreted as a low-pass filtering of $f(x)$ followed by a uniform sampling at the rate $2^j$. In an approximation operation, when removing the details of $f(x)$ smaller than $2^{-j}$, the highest frequency of this function is suppressed. The scaling function $\phi(x)$ forms a very particular low-pass filter since the family of functions $(\sqrt{2^{-j}}\phi_{2^j}(x - 2^{-j}n))_{n\in Z}$ is an orthonormal family.

### 2.3.2 Implementation of a Multiresolution Transform

In practice, a signal measuring device low-passes the continuous input signal, and a digitizer outputs a uniform sampling. Hence, this measurement corresponds to an approximation of the original signal at a finite resolution. For normalization purposes, it is supposed that this resolution is equal to 1 and let $A^d_1 f$ be the discrete approximation at the resolution 1 that is measured.

Let $(V_{2^j})_{j\in Z}$ be a multiresolution approximation and $\phi(x)$ be the corresponding scaling function. The family of functions $(\sqrt{2^{-j-1}}\phi_{2^{j+1}}(x-2^{-j-1}k))_{k\in Z}$ is an orthonormal basis of $V_{2^{j+1}}$. $\phi_{2^j}(x-2^{-j}n)$ can thus be expanded in this orthonormal basis of $V_{2^{j+1}}$ as

$$\begin{aligned}\phi_{2^j}(x-2^{-j}n) &= 2^{-j-1}\sum_{k=-\infty}^{+\infty}\left\langle\phi_{2^j}(u-2^{-j}n),\phi_{2^{j+1}}(u-2^{-j-1}k)\right\rangle \\ &\quad\cdot\phi_{2^{j+1}}(x-2^{-j-1}k). \qquad (2.9)\end{aligned}$$

By changing variables in the inner products integral, it can be shown that

$$2^{-j-1}\left\langle\phi_{2^j}(u-2^{-j}n),\phi_{2^{j+1}}(u-2^{-j-1}k)\right\rangle = \langle\phi_{2^{-1}}(u),\phi(u-(k-2n))\rangle\,. \quad (2.10)$$

When computing the inner products of $f(x)$ with both sides of (2.9), the following equation is obtained,

$$\begin{aligned}&\left\langle f(u),\phi_{2^j}(u-2^{-j}n)\right\rangle = \\ &\quad\sum_{k=-\infty}^{+\infty}\langle\phi_{2^{-1}}(u),\phi(u-(k-2n))\rangle\left\langle f(u),\phi_{2^{j+1}}(u-2^{-j-1}k)\right\rangle. \qquad (2.11)\end{aligned}$$

Let $H$ be a discrete filter with impulse response

$$\forall n\in Z,\ h(n)=\langle\phi_{2^{-1}}(u),\phi(u-n)\rangle \qquad (2.12)$$

and let $\tilde{H}$ be the mirror filter with impulse response

$$\tilde{h}(n)=h(-n). \qquad (2.13)$$

By inserting (2.12) and (2.13) into (2.11), it can be shown that the discrete approximation of $f(x)$ at a resolution $2^j$, $A^d_{2^j}f$, can be calculated by filtering $A^d_{2^{j+1}}f=(\langle f(u),\phi_{2^{j+1}}(u-2^{-j-1}k)\rangle)_{k\in Z}$ with the discrete filter $\tilde{H}$ and keeping every other sample of the convolution product,

$$A^d_{2^j}f=\left(\sum_{k=-\infty}^{+\infty}\tilde{h}(2n-k)\left\langle f(u),\phi_{2^{j+1}}\left(u-2^{-j-1}k\right)\right\rangle\right)_{n\in Z}. \qquad (2.14)$$

All the discrete approximations $A_{2^j}^d f$, for $j < 0$, can thus be computed from $A_1^d f$ by repeating this process. This operation is called a pyramid transform and the set of discrete approximations $\left(A_{2^j}^d f\right)_{-J<j\leq 0}$ was called a Gaussian pyramid with $J$ levels by Burt and Adelson [Burt83].

In practice, the measuring device gives only a finite number of samples: $A_1^d f = (\alpha_n)_{1\leq n\leq N}$. Thus, each discrete signal $A_{2^j}^d f$ $(j < 0)$ has $2^j N$ samples. In order to avoid border problems when computing the discrete approximations $A_{2^j}^d f$, it is supposed that the original signal $A_1^d f$ is symmetric with respect to $n = 1$ and $n = N$,

$$\alpha_n = \begin{cases} \alpha_{-n+2} & \text{if } -N+2 < n < 1 \\ \alpha_{2N-n} & \text{if } N < n < 2N-1 \\ 0 & \text{otherwise} \end{cases} . \tag{2.15}$$

If the impulse response of the filter $\tilde{H}$ is even, e.g., $\tilde{H} = H$, each discrete approximation $A_{2^j}^d f$ will also be symmetric with respect to $n = 1$ and $n = 2^j N$. Figure 2.1 shows the discrete approximated signal $A_{2^j}^d f$ of a continuous signal $f(x)$ at the resolution 1, $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{8}$.

The following theorem gives a practical characterization of the Fourier transform of a scaling function $\phi(x)$.

**Theorem 2.2** *Let $\phi(x)$ be a scaling function, and let $H$ be a discrete filter with impulse response $h(n) = \langle \phi_{2^{-1}}(u), \phi(u-n) \rangle$. Let $H(\omega)$ be the Fourier series defined by*

$$H(\omega) = \sum_{n=-\infty}^{+\infty} h(n) e^{-inw}. \tag{2.16}$$

*Then, the function defined by*

$$\Phi(\omega) = \prod_{p=1}^{+\infty} H\left(2^{-p}\omega\right) \tag{2.17}$$

*is the Fourier transform of a scaling function $\phi(x)$.* □

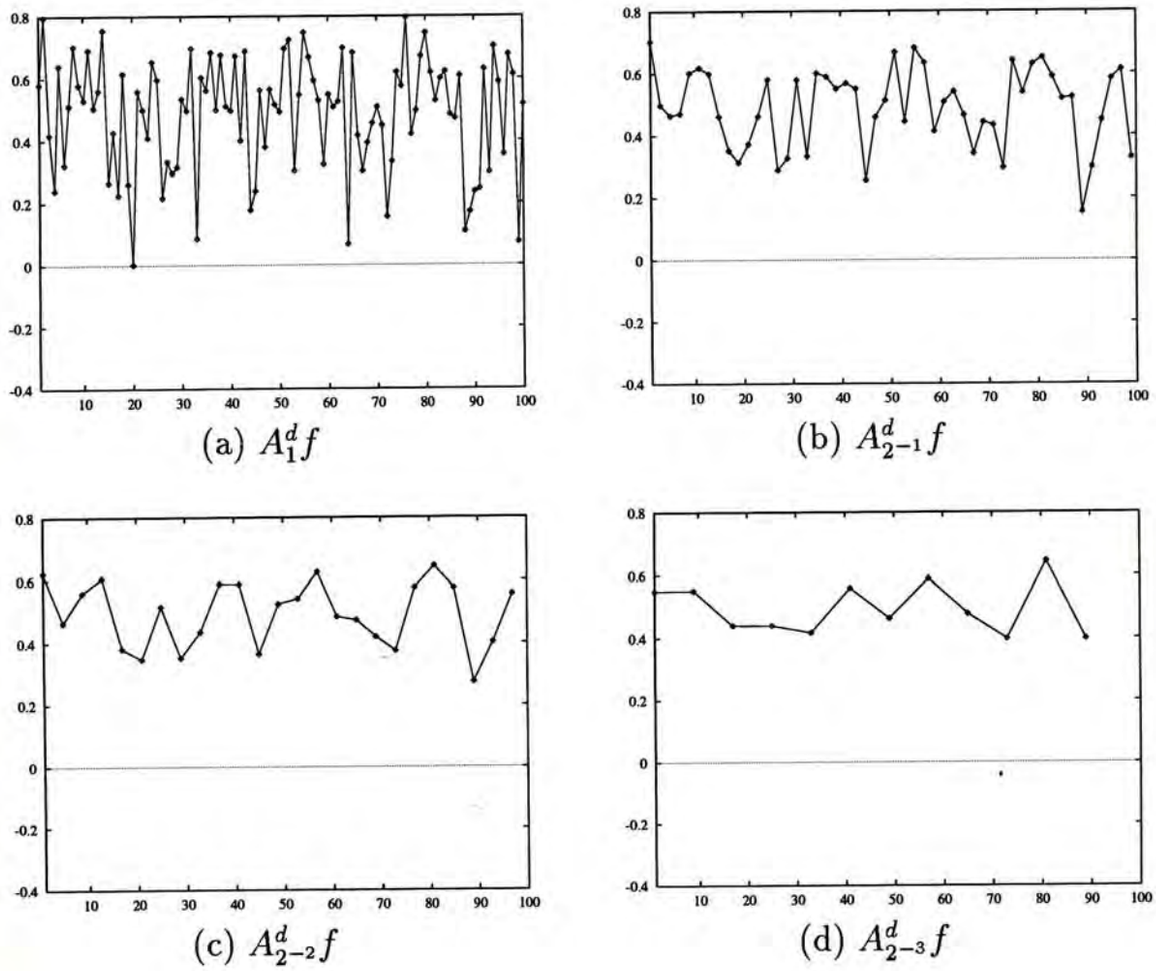



Figure 2.1: Discrete approximations $A_{2^j}^d f$ at the resolution 1, $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{8}$. Depending upon $2^{-j}n$, each dot gives the amplitude of the inner product $\langle f(u), \phi_{2^j}(u - 2^{-j}n)\rangle$.

It is possible to choose $H(\omega)$ in order to obtain a scaling function $\phi(x)$ which has good localization properties in both the frequency and spatial domains. In general, a scaling function $\phi(x)$, which is as smooth as possible and which is well concentrated around 0 in the spatial domain, is preferable.

### 2.3.3 Orthogonal Wavelet Representation

As described in Section 2.3.1, the approximation of a function at a resolution $2^j$ is equal to its orthogonal projection on $V_{2^j}$. The additional precision of the approximation when the resolution increases from $2^j$ to $2^{j+1}$ is thus given by the orthogonal projection on the orthogonal complement of $V_{2^j}$ in $V_{2^{j+1}}$. Let $O_{2^j}$ be this orthogonal complement, i.e.,

- $O_{2^j}$ is orthogonal to $V_{2^j}$.
- $O_{2^j} \oplus V_{2^j} = V_{2^{j+1}}$.

To compute the orthogonal projection of a function $f(x)$ on $O_{2^j}$, an orthonormal basis of $O_{2^j}$ must be found. Much like Theorem 2.1, Theorem 2.3 shows that such a basis can be built by scaling and translating a function $\psi(x)$.

**Theorem 2.3** *Let $(V_{2^j})_{j\in Z}$ be a multiresolution vector space sequence, $\phi(x)$ be the scale function, and $H$ be the corresponding conjugate filter. Let $\psi(x)$ be a function whose Fourier transform is given by*

$$\Psi(\omega) = G\left(\frac{\omega}{2}\right)\Phi\left(\frac{\omega}{2}\right) \tag{2.18}$$

*with $G(\omega) = e^{-i\omega}\overline{H(\omega+\pi)}$. Let $\psi_{2^j}(x) = 2^j\psi(2^j x)$ denotes the dilation of $\psi(x)$ by $2^j$. Then, $\left(\sqrt{2^{-j}}\psi_{2^j}\left(x-2^{-j}n\right)\right)_{n\in Z}$ is an orthonormal basis of $O_{2^j}$ and $\left(\sqrt{2^{-j}}\psi_{2^j}\left(x-2^{-j}n\right)\right)_{(n,j)\in Z}$ is an orthonormal basis of $L^2(R)$. Here, $\psi(x)$ is called an orthogonal wavelet.* □

An orthonormal basis of $O_{2^j}$ can thus be computed by scaling the wavelet $\psi(x)$ with a coefficient $2^j$ and translating it on a grid whose interval is proportional to $2^{-j}$. For computing a wavelet, we can define a function $H(\omega)$ to compute the corresponding scaling function $\phi(x)$ with (2.17) and the wavelet $\psi(x)$ with (2.18). Depending upon the choice of $H(\omega)$, the scaling function $\phi(x)$ and the wavelet $\psi(x)$ can have good localization both in the spatial and Fourier domains.

Let $P_{O_{2^j}}$ be the orthogonal projection on the vector space $O_{2^j}$. As a consequence of Theorem 2.3, this operator can now be written as

$$P_{O_{2^j}} f(x) = 2^{-j} \sum_{n=-\infty}^{+\infty} \left\langle f(u), \psi_{2^j}\left(u - 2^{-j}n\right)\right\rangle \psi_{2^j}\left(x - 2^{-j}n\right). \tag{2.19}$$

$P_{O_{2^j}} f(x)$ yields to the detail signal of $f(x)$ at the resolution $2^j$ and it is characterized by the set of inner products as

$$D_{2^j} f = \left(\left\langle f(u), \psi_{2^j}\left(u - 2^{-j}n\right)\right\rangle\right)_{n \in Z}. \tag{2.20}$$

$D_{2^j} f$ is called the discrete detail signal at the resolution $2^j$ and it contains the difference of information between $A^d_{2^{j+1}} f$ and $A^d_{2^j} f$. The wavelet $\psi(x)$ can also be viewed as a band-pass filter.

It can be proved by induction that for any $J > 0$, the original discrete signal $A^d_1 f$ measured at the resolution 1 is represented by

$$\left(A^d_{2^{-J}} f, (D_{2^j} f)_{-J \le j \le -1}\right). \tag{2.21}$$

This set of discrete signals is thus called an orthogonal wavelet representation. It gives a reference signal at a coarse resolution $A^d_{2^{-J}} f$ and the detail signals at the resolutions $2^j$ for $-J \le j \le -1$. It can also be interpreted as a decomposition of the original signal in an orthonormal wavelet basis or as a decomposition of the signal in a set of independent frequency channels like in Marr's human vision model [Marr82]. The independence is due to the orthogonality of the wavelet functions.

In analogy with the Laplacian pyramid data structure described in Section 2.2, $A^d_{2^{-J}} f$ provides the top-level Gaussian pyramid data, and the $D_{2^j} f$ data provide the successive Laplacian pyramid levels. Unlike the Laplacian pyramid, however, there is no oversampling, and the individual coefficients in the set of data are independent. If the original signal has $N$ samples, then the discrete signals $D_{2^j} f$ and $A^d_{2^j} f$ have $2^j N$ samples each. Thus, the wavelet representation has the same total number of samples as the original approximated signal $A^d_1 f$. Figure 2.2 gives the wavelet representation of the signal $A^d_1 f$ decomposed in Figure 2.1. The energy of the samples of $D_{2^j} f$ gives a measure of the irregularity of the signal at the resolution $2^{j+1}$. The detail signals samples have a high amplitude when the approximations $A^d_{2^j} f$ and $A^d_{2^{j+1}} f$ are locally different.

### 2.3.4 Implementation of an Orthogonal Wavelet Representation

For any $n \in Z$, the function $\psi_{2^j}(x - 2^{-j}n)$ is a member of $O_{2^j} \subset V_{2^{j+1}}$. Similarly to (2.9), this function can be expanded in an orthonormal basis of $V_{2^{j+1}}$,

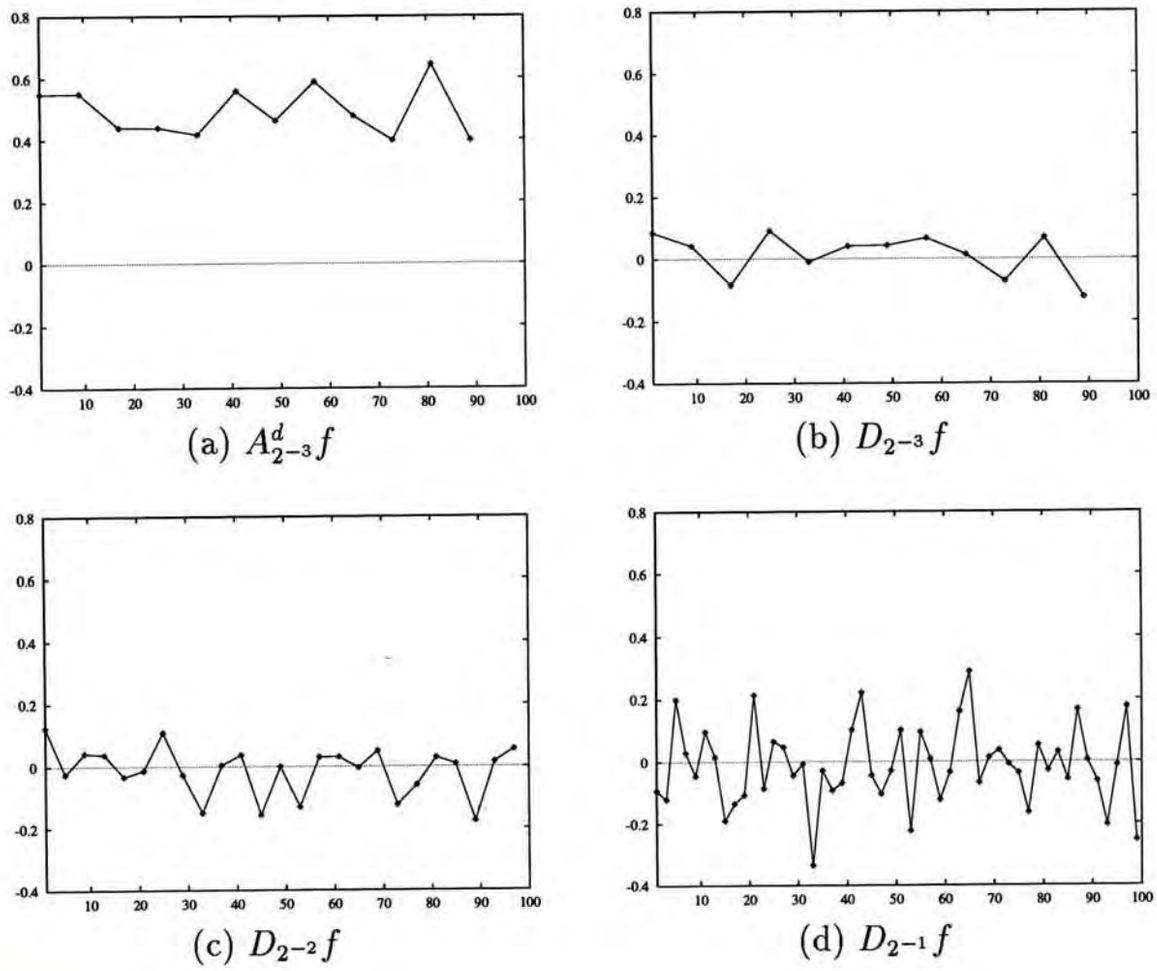
$$\begin{aligned} \psi_{2^j}(x - 2^{-j}n) &= 2^{-j-1} \sum_{k=-\infty}^{+\infty} \left\langle \psi_{2^j}(u - 2^{-j}n), \phi_{2^{j+1}}(u - 2^{-j-1}k) \right\rangle \\ &\quad \cdot \phi_{2^{j+1}}(x - 2^{-j-1}k). \end{aligned} \tag{2.22}$$

By changing the variables in the inner product integral, it can be proved that

$$2^{-j-1} \left\langle \psi_{2^j}(u - 2^{-j}n), \phi_{2^{j+1}}(u - 2^{-j-1}k) \right\rangle = \langle \psi_{2^{-1}}(u), \phi(u - (k - 2n)) \rangle . \tag{2.23}$$

Hence, by computing the inner product of $f(x)$ with the functions of both sides of (2.22), the following equation is obtained,

$$\begin{aligned} &\left\langle f(u), \psi_{2^j}(u - 2^{-j}n) \right\rangle = \\ &\quad \sum_{k=-\infty}^{+\infty} \langle \psi_{2^{-1}}(u), \phi(u - (k - 2n)) \rangle \left\langle f(u), \phi_{2^{j+1}}(u - 2^{-j-1}k) \right\rangle . \end{aligned} \tag{2.24}$$




(a) $A^d_{2^{-3}} f$ (b) $D_{2^{-3}} f$

(c) $D_{2^{-2}} f$ (d) $D_{2^{-1}} f$

Figure 2.2: Wavelet representation of the signal $A^d_1 f$. The dots give the amplitude of the inner products $\langle f(u), \psi_{2^j}(u - 2^{-j}n) \rangle$ of each detail signal $D_{2^j} f$ depending upon $2^{-j}n$.

Let G be a discrete filter with impulse response is given by

$$g(n) = \langle \psi_{2^{-1}}(u), \phi(u-n) \rangle \tag{2.25}$$

and let $\tilde{G}$ be the symmetric filter with impulse response

$$\tilde{g}(n) = g(-n). \tag{2.26}$$

It can then be shown that the detail signal $D_{2^j} f$ can be computed by convolving $A^d_{2^{j+1}} f$ with the filter $\tilde{G}$ and retaining every other sample of the output,

$$D_{2^j} f = \left( \sum_{k=-\infty}^{+\infty} \tilde{g}(2n-k) \left\langle f(u), \phi_{2^{j+1}} \left( u - 2^{-j-1} k \right) \right\rangle \right)_{n \in Z}. \tag{2.27}$$

The orthogonal wavelet representation of a discrete signal $A^d_1 f$ can therefore be computed by successively decomposing $A^d_{2^{j+1}} f$ into $A^d_{2^j} f$ and $D_{2^j} f$ for $-J \leq j \leq -1$ $(J > 0)$. This algorithm is illustrated by the block diagram shown in Figure 2.3.

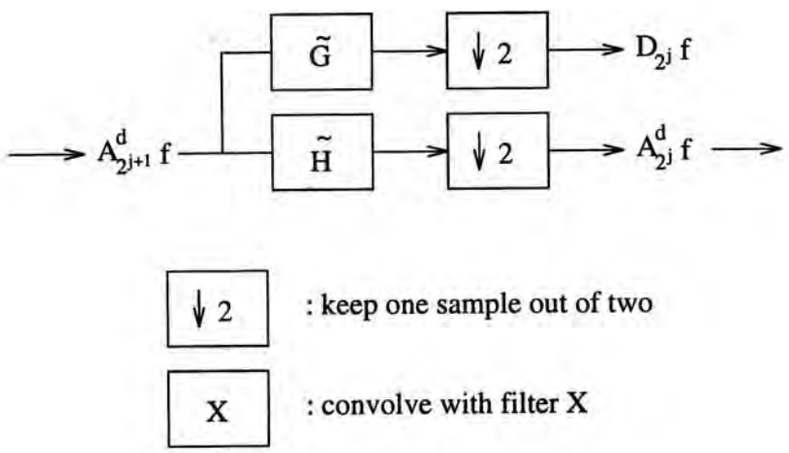



Figure 2.3: Decomposition of a discrete approximation $A^d_{2^{j+1}} f$ into an approximation at a coarser resolution $A^d_{2^j} f$ and the signal detail $D_{2^j} f$.

Also, (2.18) of Theorem 2.3 implies that the impulse response of the filter $G$ is related to the impulse response of the filter $H$ by

$$g(n) = (-1)^{1-n} h(1-n). \tag{2.28}$$

Hence, $G$ is the mirror filter of $H$ and is a high-pass filter. In signal processing, $G$ and $H$ are called conjugate mirror filters. (2.27) can then be interpreted as a high-pass filtering of the discrete signal $A^d_{2^{j+1}} f$.

### 2.3.5 Signal Reconstruction

$(\sqrt{2^{-j}}\phi_{2^j}(x - 2^{-j}n), \sqrt{2^{-j}}\psi_{2^j}(x - 2^{-j}n))_{n\in Z}$ is an orthonormal basis of $V_{2^{j+1}}$ as $O_{2^j}$ is the orthogonal complement of $V_{2^j}$ in $V_{2^{j+1}}$. For any $n > 0$, the function $\phi_{2^{j+1}}(x - 2^{-j-1}n)$ can thus be decomposed in this basis,

$$\begin{aligned}\phi_{2^{j+1}}(x - 2^{-j-1}n) = \\ 2^{-j} \sum_{k=-\infty}^{+\infty} \left\langle \phi_{2^j}(u - 2^{-j}k), \phi_{2^{j+1}}(u - 2^{-j-1}n) \right\rangle \phi_{2^j}(x - 2^{-j}k) \\ +2^{-j} \sum_{k=-\infty}^{+\infty} \left\langle \psi_{2^j}(u - 2^{-j}k), \phi_{2^{j+1}}(u - 2^{-j-1}n) \right\rangle \psi_{2^j}(x - 2^{-j}k). \quad (2.29)\end{aligned}$$

By computing the inner product of each side of (2.29) with the function $f(x)$, the following equation is obtained,

$$\begin{aligned}\left\langle f(u), \phi_{2^{j+1}}(u - 2^{-j-1}n) \right\rangle = \\ 2^{-j} \sum_{k=-\infty}^{+\infty} \left\langle \phi_{2^j}(u - 2^{-j}k), \phi_{2^{j+1}}(u - 2^{-j-1}n) \right\rangle \left\langle f(u), \phi_{2^j}(u - 2^{-j}k) \right\rangle \\ +2^{-j} \sum_{k=-\infty}^{+\infty} \left\langle \psi_{2^j}(u - 2^{-j}k), \phi_{2^{j+1}}(u - 2^{-j-1}n) \right\rangle \left\langle f(u), \psi_{2^j}(u - 2^{-j}k) \right\rangle . \\ (2.30)\end{aligned}$$

As this expression is rewritten by using the filters $H$ and $G$ respectively defined by (2.12) and (2.25), it can then be shown that the original discrete signal can also be reconstructed with a pyramid transform,

$$\begin{aligned}A^d_{2^{j+1}} f = \Bigg( 2 \sum_{k=-\infty}^{+\infty} h(n - 2k) \left\langle f(u), \phi_{2^j}\left(u - 2^{-j}k\right) \right\rangle \\ + 2 \sum_{k=-\infty}^{+\infty} g(n - 2k) \left\langle f(u), \psi_{2^j}\left(u - 2^{-j}k\right) \right\rangle \Bigg)_{n\in Z} . \quad (2.31)\end{aligned}$$

This equation shows that $A^d_{2^{j+1}} f$ can be reconstructed by putting zeros between each sample of $A^d_{2^j} f$ and $D_{2^j} f$ and convolving the resulting signals with the filters $H$ and $G$, respectively. The block diagram shown in Figure 2.4 illustrates this algorithm. The original discrete signal $A^d_1 f$ at the resolution 1 is reconstructed by repeating this procedure for $-J \leq j < 0$ $(J > 0)$.

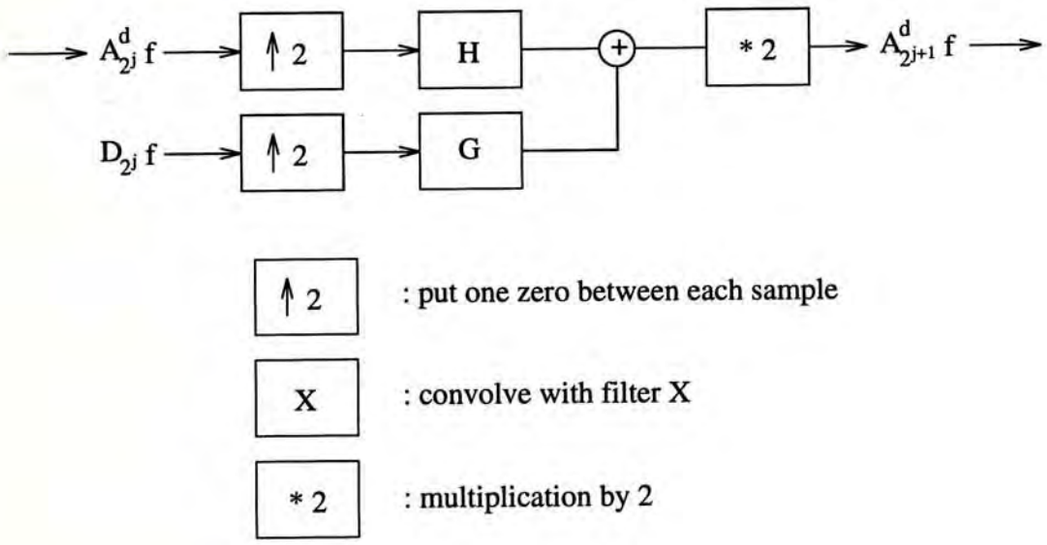



Figure 2.4: Reconstruction of a discrete approximation $A^d_{2^{j+1}} f$ from an approximation at a coarser resolution $A^d_{2^j} f$ and the signal detail $D_{2^j} f$.

# Chapter 3

# Multiresolution Learning Method

## 3.1 Introduction

In this chapter, we proposed to train a back-propagation network using the multiresolution learning method. This learning method involves a group of back-propagation networks. The objective of it is to improve the convergence rate and the recognition ability of the networks. Some of my results have been published and they can be found in [Chan94a] and [Chan94b].

By using the multisolution signal decomposition technique introduced in Section 2.3, we first represent the original input vectors under different resolutions and it is described Section 3.2. In Section 3.3, we then show how a group of corresponding back-propagation networks are built. Each of these networks is responsible to learn the input vectors at a particular resolution. The training procedure is shown in Section 3.4. The sequence of the training processes to be carried out by the group of back-propagation networks is from the coarsest level network to the finest level network sequentially. We also define a term called the intermediate stopping criteria for terminating the training processes

of these networks. After a network has been trained on a particular resolution of input vectors, we then transform the connection weights of the network to the next finer level network (Section 3.5). We consider two different cases of the transformation, i.e., the transformation of connection weights between the input and the hidden layers and the transformation of connection weights between the hidden and the output layers.

## 3.2 Input Vector Representation

First of all, let us define how input vectors for the back-propagation networks are represented under different resolutions. Let $(\vec{x_i})_{1\leq i\leq M}$ be a set of $M$ input vectors. Each vector is with $N$-dimensionality where $\vec{x_i} = (x_{i1}, x_{i2}, \ldots, x_{iN})$ and $x_{ij} \in R$. As the back-propagation algorithm is a kind of supervised learning algorithms, for each $\vec{x_i}$, there is a vector $\vec{d_i}$ called the desired output vectors assigned to it. Hence, the set of $M$ pairs $\left(\left(\vec{x_i}, \vec{d_i}\right)\right)_{1\leq i\leq M}$ are formed as the input for the training process of the networks.

### 3.2.1 Representation at the resolution 1

It has been mentioned in Section 2.3.1 that $A_1^d f$ is the discrete approximation at the resolution 1 and contains a finite number of samples $(\alpha_n)_{1\leq n\leq N}$. We can then define the discrete approximation of $\vec{x_i}$ at the resolution 1, $A_1^d \vec{x_i}$, as

$$A_1^d \vec{x_i} = (x_{in})_{1\leq n\leq N} \tag{3.1}$$

and a set of input pairs at the resolution 1, $\left(\left(A_1^d \vec{x_i}, \vec{d_i}\right)\right)_{1\leq i\leq M}$, can thus be formed.

### 3.2.2 Representation at the resolution $2^j$

By choosing a suitable discrete filter $H$ and applying (2.14), the discrete approximation of $\vec{x}_i$ at the resolution $\frac{1}{2}$, $A^d_{2^{-1}}\vec{x}_i$, is obtained by low-pass filtering the original signal $A^d_1\vec{x}_i$. $A^d_{2^{-1}}\vec{x}_i$ is a reduced version of $A^d_1\vec{x}_i$ in that both resolution and sample density are decreased. In a similar way, $A^d_{2^{-2}}\vec{x}_i$ is formed as a reduced version of $A^d_{2^{-1}}\vec{x}_i$, and so on. As a result, several levels, e.g., $J$ levels, of the input pairs' sets are obtained,

$$\left(\left(A^d_1\vec{x}_i, \vec{d}_i\right)\right)_{1\le i\le M}, \left(\left(A^d_{2^{-1}}\vec{x}_i, \vec{d}_i\right)\right)_{1\le i\le M}, \cdots, \left(\left(A^d_{2^{-J+1}}\vec{x}_i, \vec{d}_i\right)\right)_{1\le i\le M} \tag{3.2}$$

with $\left(\left(A^d_1\vec{x}_i, \vec{d}_i\right)\right)_{1\le i\le M}$ as the finest resolution level and $\left(\left(A^d_{2^{-J+1}}\vec{x}_i, \vec{d}_i\right)\right)_{1\le i\le M}$ as the coarsest one. At any resolution $j$, the discrete approximation $A^d_{2^j}\vec{x}_i$ has $2^jN$ samples, e.g., with $2^jN$-dimensionality.

In general, the discrete approximation of $\vec{x}_i$ at the resolution $2^j$, $A^d_{2^j}\vec{x}_i$, can be defined as

$$A^d_{2^j}\vec{x}_i = \left(\sum_{k=-\infty}^{+\infty} \tilde{h}(2n-k)A^d_{2^{j+1}}\vec{x}_i(k)\right)_{1\le n\le 2^jN} \tag{3.3}$$

where $A^d_{2^{j+1}}\vec{x}_i(k)$ being the $k$-th element of $A^d_{2^{j+1}}\vec{x}_i$. Hence, all the discrete approximations $A^d_{2^j}\vec{x}_i$, for $j < 0$, can thus be computed from $\vec{x}_i$ by repeating this process.

**Example.** Suppose we have a set of 50 input vectors $(\vec{x}_i)_{1\le i\le 50}$. Each vector is with 8-dimensionality, e.g., $\vec{x}_i = (x_{i1}, x_{i2}, \cdots, x_{i8})$, and there is a corresponding vector $\vec{d}_i$ of 2-dimensionality, e.g., $\vec{d}_i = (d_{i1}, d_{i2})$, associated with it. The set of pairs $\left(\left(\vec{x}_i, \vec{d}_i\right)\right)_{1\le i\le 50}$ then forms the finest resolution level. For the consistency of notations, we refer it as $\left(\left(A^d_1\vec{x}_i, \vec{d}_i\right)\right)_{1\le i\le 50}$. By (3.3), two more levels of input pairs' sets can be obtained,

- $\left(\left(A^d_{2^{-1}}\vec{x}_i, \vec{d}_i\right)\right)_{1\le i\le 50}$ where $A^d_{2^{-1}}\vec{x}_i = (y_{i1}, y_{i2}, y_{i3}, y_{i4})$ and
- $\left(\left(A^d_{2^{-2}}\vec{x}_i, \vec{d}_i\right)\right)_{1\le i\le 50}$ where $A^d_{2^{-2}}\vec{x}_i = (z_{i1}, z_{i2})$.

In this case, the dimensionality of $A^d_{2^{-1}}\vec{x}_i$ is 4; while the one of $A^d_{2^{-2}}\vec{x}_i$ is 2. The desired output vectors $\vec{d}_i$ are common to all three levels. □

### 3.2.3 Border Problem

In order to avoid border problems described in Section 2.3.2 when computing the discrete approximations $A^d_{2^j}\vec{x}_i$, it is supposed that the original input vector $A^d_1\vec{x}_i$, is symmetric with respect to $n = 1$ and $n = N$, i.e.,

$$A^d_1\vec{x}_i(n) = \begin{cases} A^d_1\vec{x}_i(-n+2) & \text{if } -N+2 < n < 1 \\ A^d_1\vec{x}_i(2N-n) & \text{if } N < n < 2N-1 \\ 0 & \text{otherwise} \end{cases} . \tag{3.4}$$

If the chosen discrete filter $H$ is even, e.g., $\tilde{H} = H$, each discrete approximation $A^d_{2^j}\vec{x}_i$ will also be symmetric with respect to $n = 1$ and $n = 2^j N$ [Mall89a].

## 3.3 Back-Propagation Network Architecture

After several levels of the input pairs' sets, $\left(\left(\left(A^d_{2^j}\vec{x}_i, \vec{d}_i\right)\right)_{1\le i\le M}\right)_{-J<j\le 0}$ for $J < 0$, have been generated under different resolutions, we then build a group of back-propagation networks $(B_{2^j})_{-J<j\le 0}$. Each back-propagation network $B_{2^j}$ is responsible to learn the mapping between $\left(A^d_{2^j}\vec{x}_i\right)_{1\le i\le M}$ and $\left(\vec{d}_i\right)_{1\le i\le M}$. The input layer of the network $B_{2^j}$ receives the input vectors $A^d_{2^j}\vec{x}_i$ as its input signals for further processing. Hence, the size of it for the network $B_{2^j}$ is equal to the dimension of the input vectors that it learns, which is $2^j N$. The output layer represents the number of categories to be classified in the input vectors. Therefore, the size of it is equal to the dimension of the output vectors $\vec{d}_i$ and is the same among all networks generated.

The required number of neurons in the hidden layer greatly depends on the nature of the problem to be solved [Hush93]. With some specific knowledge about the structure of the problem, and a fundamental understanding of how

the back-propagation networks might go about implementing this structure, one can sometimes form a good estimate of the proper network size. Like the size of the output layer, the size of the hidden layer is the same among all back-propagation networks created.

**Example.** With refer to the example shown in Section 3.2, we can build three different back-propagation networks $(B_{2^j})_{-2\leq j\leq 0}$ to learn the three levels of input pairs' sets generated.

- $B_1$ is used to learn the finest level $\left(\left(A_1^d\vec{x}_i, \vec{d}_i\right)\right)_{1\leq i\leq 50}$.
- $B_{2^{-1}}$ is used to learn the intermediate level $\left(\left(A_{2^{-1}}^d\vec{x}_i, \vec{d}_i\right)\right)_{1\leq i\leq 50}$.
- $B_{2^{-2}}$ is used to learn the coarsest level $\left(\left(A_{2^{-2}}^d\vec{x}_i, \vec{d}_i\right)\right)_{1\leq i\leq 50}$.

If we set the size of the hidden layer as 3 for all networks, each network will then have the structure shown in Table 3.1. □

Table 3.1: The structure of the back-propagation networks with 3 levels of resolutions.

| BP network | Size | | |
|---|---|---|---|
| | Input layer | Hidden layer | Output layer |
| $B_1$ | 8 | 3 | 2 |
| $B_{2^{-1}}$ | 4 | 3 | 2 |
| $B_{2^{-2}}$ | 2 | 3 | 2 |

## 3.4 Training Procedure Strategy

With several levels of the input pairs' sets, $\left(\left(\left(A_{2^j}^d\vec{x}_i, \vec{d}_i\right)\right)_{1\leq i\leq M}\right)_{-J<j\leq 0}$ for $J < 0$, under different resolutions and a group of the corresponding back-propagation networks, $(B_{2^j})_{-J<j\leq 0}$, we can then start the training procedure. First of all, the coarsest level network $B_{2^{-J+1}}$ (the network with the coarsest resolution of vectors

$A^d_{2^{-J+1}}\vec{x}_i$ as input) is trained first. We can initialize the connection weights of this network with small random numbers [Rume86] or by the statistical approach suggested by Wessels [Wess92] before the training process is started. After the network $B_{2^{-J+1}}$ is well trained, we transform the connection weights of it into the connection weights of the finer level network $B_{2^{-J+2}}$ and then start the training process of $B_{2^{-J+2}}$. Such training procedure is repeated until the finest level network $B_1$ has converged.

**Example.** To continue the example shown in Section 3.3, the coarsest level network $B_{2^{-2}}$ will first learn the mapping between $\left(A^d_{2^{-2}}\vec{x}_i\right)_{1\le i\le 50}$ and $\left(\vec{d}_i\right)_{1\le i\le 50}$ after the initialization of connection weights. After it is well trained, the connection weights of it is transformed into the next finer level network $B_{2^{-1}}$ and the training process of $B_{2^{-1}}$ is started to learn the mapping between $\left(A^d_{2^{-1}}\vec{x}_i\right)_{1\le i\le 50}$ and $\left(\vec{d}_i\right)_{1\le i\le 50}$, and so on. The training procedure is finished until the finest level network $B_1$ has converged. Figure 3.1 illustrates the whole training procedure described in this example. □

In the following of this section, we define for how long that a particular network $B_{2^j}$ is trained before the transformation of connection weights is carried out. In the next section, we formulate this transformation between two networks, e.g., $B_{2^j}$ and $B_{2^{j+1}}$.

### 3.4.1 Sum Squared Error (SSE)

Traditionally, the training process of a back-propagation network is repeated until a minimum on sum squared error (SSE) or a point sufficiently close to the minimum is found. Let $L$ be the size of the output layer, $(\vec{o}_i)_{1\le i\le M}$ be the set of the actual output vectors from the output layer after presenting the set of the input vectors $\left(A^d_{2^j}\vec{x}_i\right)_{1\le i\le M}$ to the network $B_{2^j}$, and $\left(\vec{d}_i\right)_{1\le i\le M}$ be the

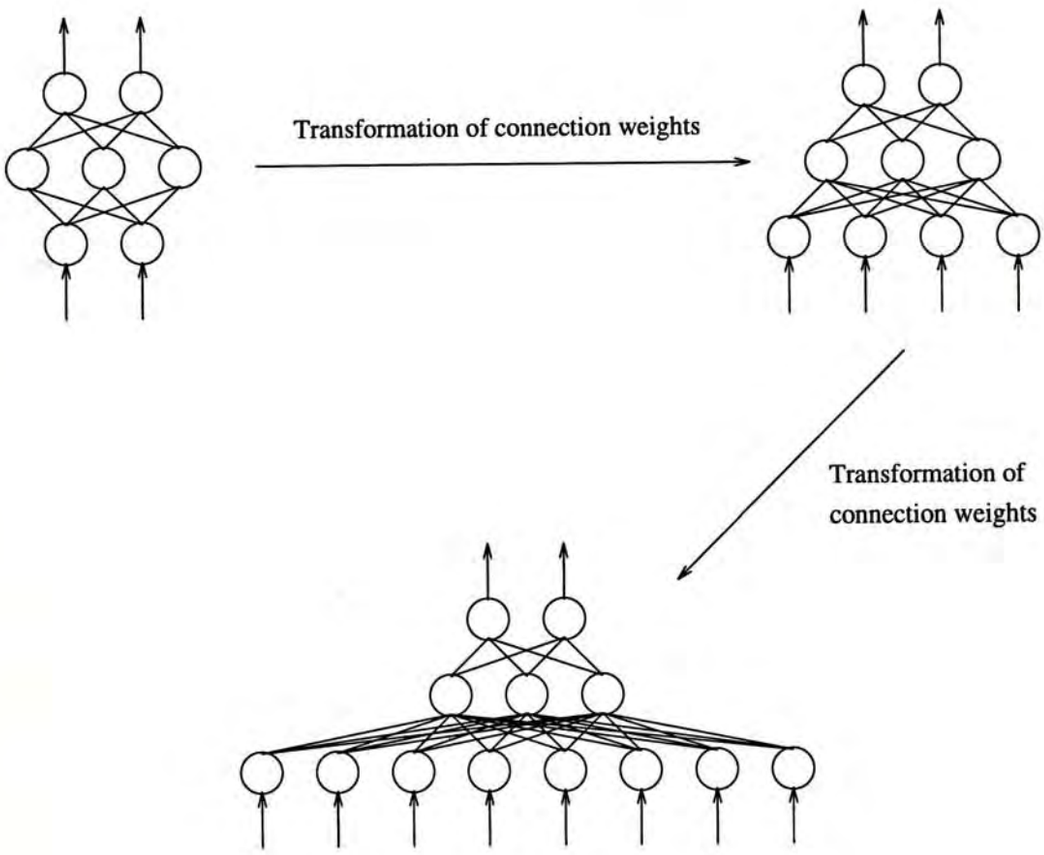



Figure 3.1: The training procedure for the multiresolution learning method with three levels of networks, $(B_{2^j})_{-2 \leq j \leq 0}$.

corresponding set of the desired output vectors. The SSE of the current training cycle $t$ is thus defined as

$$SSE(t) = \sum_{i=1}^{M} \sum_{n=1}^{L} (o_{in} - d_{in})^2 < \varepsilon \tag{3.5}$$

where $\varepsilon > 0$, $\vec{o_i} = (o_{i1}, o_{i2}, \cdots, o_{iL})$, and $\vec{d_i} = (d_{i1}, d_{i2}, \cdots, d_{iL})$.

### 3.4.2 Intermediate Stopping Criteria

However, such a minimum may not be found in the networks we defined except the finest level one $B_1$ (the one with the original input vectors $\vec{x_i}$ as input). It is because the detail signal $D_{2^j}\vec{x_i}$ is lost during the approximation process.

As a result, we define an intermediate stopping criteria for terminating the training processes of the back-propagation networks $(B_{2^j})_{J \le j \le -1}$ which are trained on the discrete signals $\left(\left(A^d_{2^j}\vec{x_i}\right)_{1 \le i \le M}\right)_{J \le j \le -1}$. Let $K$ be the number of hidden neurons, $2^j N$ be the number of input neurons, $w_{pq}$ be the connection weight between input neuron $p$ and the hidden neuron $q$ and it will be updated with $\Delta w_{pq}$ in the current training cycle $t$. Hence, a term $W(t)$ is defined as[1]

$$W(t) = \frac{1-\rho}{2^j NK} \sum_{q=1}^{K} \sum_{p=1}^{2^j N} \left| \frac{\Delta w_{pq}}{w_{pq}} \right| + \rho W(t-1) \tag{3.6}$$

where $0 < \rho < 1$, called a history factor, and $\frac{1}{2^j NK}$ is used for normalization purpose. The intermediate stopping criteria is then defined as

$$W(t) < \delta \tag{3.7}$$

where $\delta > 0$.

[1] (3.6) is somewhat similar to the modified generalized delta rule shown in [Rume86]. The last term of the equation is used to filter out high-frequency variations and prohibits oscillation to be occurred.

## 3.5 Connection Weight Transformation

After the intermediate stopping criteria shown in (3.7) is satisfied in one network $B_{2^j}$, we can then transform the connection weights of it to the next finer level network $B_{2^{j+1}}$. From the coarser level network $B_{2^j}$, we have two sets of connection weights, $\{w_{pq}\}$ and $\{v_{qr}\}$, where $w_{pq}$ is a connection weight from input neuron $p$ to hidden neuron $q$ and $v_{qr}$ is a connection weight from hidden neuron $q$ to output neuron $r$.

### 3.5.1 Weights between the Input and Hidden Layers

First of all, let us consider how the connection weights between the input and the hidden layers $\{w_{pq}\}$ are transformed. The basic idea behind this transformation that we propose here is that the input signal received by each hidden neuron of the finer level network $B_{2^{j+1}}$ from the input layer should be the same as the one received by the corresponding hidden neuron of the coarser level network $B_{2^j}$.

If a discrete signal $A^d_{2^j}\vec{x_i}$ is passed into the coarser level network $B_{2^j}$, the hidden neuron $q$ will then receive

$$\sum_{p=1}^{2^j N} w_{pq} A^d_{2^j}\vec{x_i}(p) \tag{3.8}$$

as its input signal from $2^j N$ input neurons. In order to maintain the same status for the hidden neuron $q$ of the finer level network $B_{2^{j+1}}$ after transformation, the following condition must be held for each hidden neuron,

$$\sum_{o=1}^{2^{j+1}N} \left(w'_{oq} A^d_{2^{j+1}}\vec{x_i}(o)\right) = \sum_{p=1}^{2^j N} \left(w_{pq} A^d_{2^j}\vec{x_i}(p)\right) \tag{3.9}$$

where $w'_{oq}$ is the connection weight of the finer level network from the input neuron $o$ to the hidden neuron $q$.

Based on the assumption shown in (3.9), we can then derive the set of $\left\{w'_{oq}\right\}$

as

$$w'_{oq} = \begin{cases} \sum_{p=1}^{2^jN} \tilde{h}(2p-1)w_{pq} & \text{if } o = 1 \\ \sum_{p=1}^{2^jN} \left(\tilde{h}(2p-o) + \tilde{h}(2p+o-2^{j+2}N) \right. & \text{if } 2 \le o \le 2^{j+1}N - 1 \\ \left. +\tilde{h}(2p+o-2)\right) w_{pq} & \\ \sum_{p=1}^{2^jN} \tilde{h}(2p-2^{j+1}N)w_{pq} & \text{if } o = 2^{j+1}N \end{cases} \tag{3.10}$$

**Proof.** By (3.3) and (3.4), the r.h.s. of (3.9) can be rewritten as

$$\begin{aligned} \sum_{p=1}^{2^jN} \left(w_{pq} A^d_{2^j} \vec{x}_i(p)\right) &= \sum_{p=1}^{2^jN} \left( w_{pq} \sum_{o=-\infty}^{+\infty} \tilde{h}(2p-o) A^d_{2^{j+1}} \vec{x}_i(o) \right) \\ &= \sum_{p=1}^{2^jN} \left( w_{pq} \sum_{o=-2^{j+1}N+3}^{2^{j+2}N-2} \tilde{h}(2p-o) A^d_{2^{j+1}} \vec{x}_i(o) \right). \end{aligned} \tag{3.11}$$

Furthermore, by changing the variable $o$ used in summation and using the property of $A^d_{2^{j+1}} \vec{x}_i(o)$ shown in (3.4), the summation term $\sum \tilde{h}(2p-o) A^d_{2^{j+1}} \vec{x}_i(o)$ of (3.11) can be expressed as

$$\begin{aligned} &\sum_{o=-2^{j+1}N+3}^{2^{j+2}N-2} \tilde{h}(2p-o) A^d_{2^{j+1}} \vec{x}_i(o) \\ &= \sum_{o=1}^{2^{j+1}N} \tilde{h}(2p-o) A^d_{2^{j+1}} \vec{x}_i(o) + \sum_{o=2^{j+1}N+1}^{2^{j+2}N-2} \tilde{h}(2p-o) A^d_{2^{j+1}} \vec{x}_i(o) \\ &\quad + \sum_{o=-2^{j+1}N+3}^{0} \tilde{h}(2p-o) A^d_{2^{j+1}} \vec{x}_i(o) \\ &= \sum_{o=1}^{2^{j+1}N} \tilde{h}(2p-o) A^d_{2^{j+1}} \vec{x}_i(o) + \sum_{o=2}^{2^{j+1}N-1} \tilde{h}(2p+o-2^{j+2}N) \\ &\quad \cdot A^d_{2^{j+1}} \vec{x}_i(2^{j+2}N - o) + \sum_{o=2}^{2^{j+1}N-1} \tilde{h}(2p+o-2) A^d_{2^{j+1}} \vec{x}_i(2-o) \\ &= \sum_{o=1}^{2^{j+1}N} \tilde{h}(2p-o) A^d_{2^{j+1}} \vec{x}_i(o) + \sum_{o=2}^{2^{j+1}N-1} \tilde{h}(2p+o-2^{j+2}N) A^d_{2^{j+1}} \vec{x}_i(o) \\ &\quad + \sum_{o=2}^{2^{j+1}N-1} \tilde{h}(2p+o-2) A^d_{2^{j+1}} \vec{x}_i(o) \end{aligned}$$

$$
\begin{aligned}
&= \tilde{h}(2p-1)A^d_{2^{j+1}}\vec{x}_i(1) + \tilde{h}(2p-2^{j+1}N)A^d_{2^{j+1}}\vec{x}_i(2^{j+1}N) \\
&\quad + \sum_{o=2}^{2^{j+1}N-1}\Big(\tilde{h}(2p-o) + \tilde{h}(2p+o-2^{j+2}N) + \tilde{h}(2p+o-2)\Big) \\
&\quad \cdot A^d_{2^{j+1}}\vec{x}_i(o). \qquad (3.12)
\end{aligned}
$$

Then, by substituted (3.12) into (3.11), we get

$$
\begin{aligned}
&\sum_{p=1}^{2^jN}\Big(w_{pq}A^d_{2^j}\vec{x}_i(p)\Big) \\
&\quad = \sum_{p=1}^{2^jN}\Big(w_{pq}\tilde{h}(2p-1)A^d_{2^{j+1}}\vec{x}_i(1)\Big) \\
&\qquad + \sum_{p=1}^{2^jN}\Big(w_{pq}\tilde{h}(2p-2^{j+1}N)A^d_{2^{j+1}}\vec{x}_i(2^{j+1}N)\Big) \\
&\qquad + \sum_{p=1}^{2^jN}\Bigg(w_{pq}\sum_{o=2}^{2^{j+1}N-1}\Big(\tilde{h}(2p-o) + \tilde{h}(2p+o-2^{j+2}N) \\
&\qquad + \tilde{h}(2p+o-2)\Big)A^d_{2^{j+1}}\vec{x}_i(o)\Big) \\
&\quad = \sum_{p=1}^{2^jN}\Big(\tilde{h}(2p-1)w_{pq}\Big)A^d_{2^{j+1}}\vec{x}_i(1) \\
&\qquad + \sum_{p=1}^{2^jN}\Big(\tilde{h}(2p-2^{j+1}N)w_{pq}\Big)A^d_{2^{j+1}}\vec{x}_i(2^{j+1}N) \\
&\qquad + \sum_{o=2}^{2^{j+1}N-1}\Bigg(\sum_{p=1}^{2^jN}\Big(\tilde{h}(2p-o) + \tilde{h}(2p+o-2^{j+2}N) \\
&\qquad + \tilde{h}(2p+o-2)\Big)w_{pq}\Bigg)A^d_{2^{j+1}}\vec{x}_i(o). \qquad (3.13)
\end{aligned}
$$

By equating the coefficients of the l.h.s. of (3.9) and the coefficients of the r.h.s. of (3.13), the result of (3.10) is worked out. □

### 3.5.2 Weights between the Hidden and Output Layers

After the transformation of the connection weights between the input and the hidden layers is formulated, let us consider how the connection weights between the hidden and the output layers $\{v_{qr}\}$ are transformed from the coarser level network $B_{2^j}$ into the finer level network $B_{2^{j+1}}$. Here we propose three different

schemes for the transformation. They are called the copy scheme, the average scheme, and the scale scheme respectively.

**The Copy Scheme** As shown in Section 3.3 and in Figure 3.1, the number of neurons in the hidden layer and the output layer are the same on both back-propagation networks, $B_{2^j}$ and $B_{2^{j+1}}$. We can then simply copy the values of $\{v_{qr}\}$ to the next finer level network $B_{2^{j+1}}$ as the connection weights between the hidden layer and the output layer. Let $v'_{qr}$ be a connection weight between the hidden neuron $q$ and the output neuron $r$ of the finer level network $B_{2^{j+1}}$. For all $1 \leq q \leq K$ and $1 \leq r \leq L$, we then have

$$v'_{qr} = v_{qr} \tag{3.14}$$

where $K$ is the number of hidden neurons and $L$ is the number of output neurons.

**The Average Scheme** As mentioned in Chapter 2, a discrete approximated signal at a coarse resolution provide the "context" of the signal whereas the finer details correspond to the particular "modalities". Such modalities are found in the finer input vectors $A^d_{2^{j+1}}\vec{x}_i$ but not in the coarser one $A^d_{2^j}\vec{x}_i$. If the modalities of the signal $A^d_{2^{j+1}}\vec{x}_i$ are an important factor for the mapping between the input and the output vectors, it is unfair that the connection weights between the hidden and the output layers are copied one by one from the coarser network $B_{2^j}$ into the finer network $B_{2^{j+1}}$. It is because the distribution of the connection weights between these two layers are bias to the coarser input signal $A^d_{2^j}\vec{x}_i$ after the training process of $B_{2^j}$ and it is difficult for the finer level network $B_{2^{j+1}}$ to extract the properties of those modalities.

Therefore, we propose a scheme that each connection weight between the hidden and the output layers of the finer level network $B_{2^{j+1}}$ takes the

average value of the sum of the connection weights between these two layers of the coarser level network $B_{2^j}$ and it allows the finer back-propagation network to rearrange the mapping between the hidden and the output layers. In other words, every $v'_{qr}$ is assigned to

$$v'_{qr} = \frac{1}{KL} \sum_{q=1}^{K} \sum_{r=1}^{L} v_{qr}. \tag{3.15}$$

**The Scale Scheme** It is believed that large absolute values of the connection weights can cause neurons to be highly active or inactive for all training input vectors, and thus insensitive to the training process [Rume86]. Hence, initializing the net with small random connection weights is employed as the choice of starting parameters for the training process. Studies have been conducted by Wessels and Barnard [Wess92] and they suggested to choose the connection weights to occupy the range $[-\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}]$ where $N$ denotes the number of connection weights leading to a particular neuron.

As a result, we propose another scheme that the connection weights between the hidden and the output layers of the finer level network $B_{2^{j+1}}$ is bounded within the range of $[-\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}]$ where $N$ denotes the number of hidden neurons in the network. That is, every $v'_{qr}$ is assigned to

$$v'_{qr} = \frac{v_{qr}}{\sqrt{N} \max(|v_{qr}|_{1 \le q \le K, 1 \le r \le L})} \tag{3.16}$$

where $\max(|v_{qr}|_{1 \le q \le K, 1 \le r \le L})$ is the maximum absolute value among $\{v_{qr}\}$.

# Chapter 4

# Simulations

## 4.1 Introduction

In this chapter, some computational results are shown to illustrate the performance of the proposed learning method described in Chapter 3. We first choose the impulse response $\tilde{h}(n)$ that is used in the generation of the multiresolution representation of the input vectors for the back-propagation networks and in the transformation of connection weights from the coarser level network to the finer level network. Two different problems are simulated and they are the XOR problem and the numeric recognition problem. For each problem, a number of training processes are carried out with the multiresolution learning method. We then discuss the experimental results in the last section of this chapter. Meanwhile, some of the simulation results can be found in [Chan94a] and [Chan94b].

## 4.2 Choices of the Impulse Response $\tilde{h}(n)$

As shown in (3.3) and (3.10), we must first define the impulse response $\tilde{h}(n)$ before the input vectors can be represented under different resolutions and the transformation of connection weights can be carried out. In other words, $h(n)$

must be defined since $\tilde{h}(n) = h(-n)$ from (2.13). The basic requirement on the coefficients $h(n)$ can be derived as

$$\sum_{n=-\infty}^{+\infty} h(n) = 1. \tag{4.1}$$

**Proof.** Let $(V_{2^j})_{j\in Z}$ be a multiresolution approximation and $\phi(x)$ be the corresponding scaling function. As shown in Theorem 2.1, the family of functions $\left(\sqrt{2^{-j-1}}\phi_{2^{j+1}}\left(x - 2^{-j-1}n\right)\right)_{n\in Z}$ is an orthonormal basis of $V_{2^{j+1}}$. For any $n \in Z$, the function $\phi_{2^j}\left(x - 2^{-j}k\right)$ is also a member of $V_{2^j}$ which is included in $V_{2^{j+1}}$ and it can thus be expanded in this orthonormal basis of $V_{2^{j+1}}$ like

$$\begin{aligned}\phi_{2^j}\left(x - 2^{-j}k\right) &= 2^{-j-1}\sum_{n=-\infty}^{+\infty}\left\langle\phi_{2^j}\left(u - 2^{-j}k\right), \phi_{2^{j+1}}\left(u - 2^{-j-1}n\right)\right\rangle \\ &\quad \cdot \phi_{2^{j+1}}\left(x - 2^{-j-1}n\right).\end{aligned} \tag{4.2}$$

By setting $j = -1$, $k = 0$, and with the use of (2.12), the above equation then becomes as

$$\begin{aligned}\phi_{2^{-1}}(x) &= \sum_{n=-\infty}^{+\infty}\langle\phi_{2^{-1}}(u), \phi(u-n)\rangle\,\phi(x-n) \\ &= \sum_{n=-\infty}^{+\infty} h(n)\phi(x-n)\end{aligned} \tag{4.3}$$

From Theorem 2.1, we know that $\phi_{2^j}(x) = 2^j\phi(2^j x)$. Hence, the above equation can be further simplified to

$$\begin{aligned}2^{-1}\phi\left(2^{-1}x\right) &= \sum_{n=-\infty}^{+\infty} h(n)\phi(x-n) \\ \phi(x) &= 2\sum_{n=-\infty}^{+\infty} h(n)\phi(2x-n).\end{aligned} \tag{4.4}$$

It is called a two-scale difference equation by Strang [Stra89]. We then look for a solution normalized by $\int \phi(x)dx = 1$. If (4.4) is multiplied by 2 and is then integrated, it can be rewritten as

$$2\int \phi(x)dx = 2\sum_{n=-\infty}^{+\infty} h(n)\int \phi(2x-n)d(2x-n) \tag{4.5}$$

and the result comes out. □

There are many ways of choosing the coefficients $h(n)$ [Stra89]. As suggested by Daubechies [Daub88], we can set the impulse response $h(n)$ as

$$h(n) = \begin{cases} \frac{1+\sqrt{3}}{8} & \text{if } n = -1 \\ \frac{3+\sqrt{3}}{8} & \text{if } n = 0 \\ \frac{3-\sqrt{3}}{8} & \text{if } n = 1 \\ \frac{1-\sqrt{3}}{8} & \text{if } n = 2 \\ 0 & \text{otherwise} \end{cases} . \tag{4.6}$$

On the other hand, one always uses the average splitting technique in image processing. That is, the impulse response $h(n)$ is set to

$$h(n) = \begin{cases} 0.5 & \text{if } n = 0 \text{ or } n = 1 \\ 0 & \text{otherwise} \end{cases} . \tag{4.7}$$

With the use of the coefficients shown in (4.6), (3.3) and (3.10) can then be simplified to

$$A^d_{2^j}\vec{x_i} = \left( \sum_{k=2n-1}^{2n+2} \tilde{h}(2n-k) A^d_{2^{j+1}}\vec{x_i}(k) \right)_{1 \le n \le 2^j N} \tag{4.8}$$

and

$$w'_{oq} = \begin{cases} \tilde{h}(1)w_{1q} & \text{if } o = 1 \\ \tilde{h}(0)w_{1q} & \text{if } o = 2 \\ \tilde{h}(-1)w_{\frac{o-1}{2},q} + \tilde{h}(1)w_{\frac{o+1}{2},q} & \text{if } o = 3, 5 \cdots, 2^{j+1}N - 3 \\ \tilde{h}(-2)w_{\frac{o-2}{2},q} + \tilde{h}(0)w_{\frac{o}{2},q} & \text{if } o = 4, 6, \cdots, 2^{j+1}N - 4 \\ \tilde{h}(-2)w_{2^jN-2,q} + \tilde{h}(0)w_{2^jN-1,q} & \text{if } o = 2^{j+1}N - 2 \\ \quad + \tilde{h}(-2)w_{2^jN,q} & \\ \tilde{h}(-1)w_{2^jN-1,q} + \tilde{h}(1)w_{2^jN,q} & \text{if } o = 2^{j+1}N - 1 \\ \quad + \tilde{h}(-1)w_{2^jN,q} & \\ \tilde{h}(-2)w_{2^jN-1,q} + \tilde{h}(0)w_{2^jN,q} & \text{if } o = 2^{j+1}N \end{cases} . \tag{4.9}$$

The proof of (4.9) can be found in Appendix A.

Similarly, with the use of the coefficients shown in (4.7), (3.3) and (3.10) can then be simplified to

$$A_{2^j}^d \vec{x}_i = \left( \sum_{k=2n}^{2n+1} \tilde{h}(2n-k) A_{2^{j+1}}^d \vec{x}_i(k) \right)_{1 \le n \le 2^j N} \tag{4.10}$$

and

$$w'_{oq} = \begin{cases} 0 & \text{if } o = 1 \\ \tilde{h}(0) w_{\frac{o}{2},q} & \text{if } o = 2, 4, \cdots, 2^{j+1}N \\ \tilde{h}(-1) w_{\frac{o-1}{2},q} & \text{if } o = 3, 5 \cdots, 2^{j+1}N - 3 \\ \tilde{h}(-1) w_{2^j N - 1,q} + \tilde{h}(-1) w_{2^j N,q} & \text{if } o = 2^{j+1}N - 1 \end{cases} . \tag{4.11}$$

The proof of (4.11) can be found in Appendix B.

## 4.3 XOR Problem

In this section, the XOR problem is used to illustrate the performance of the proposed learning method. The logic of this problem is that when two binary inputs are of the same value, the output is equal to 0; otherwise, if the inputs are of different values, the output is then equal to 1 (Table 4.1).

Table 4.1: The logic of XOR.

| Input | | Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

### 4.3.1 Setting of Experiments

Two experiments were run with the impulse responses from Daubechies (4.6) and the average splitting method (4.7) respectively. In each experiment, we selected

different values of history factors $\rho$ and intermediate stopping criteria $\delta$ shown in (3.6) and (3.7) respectively, i.e., $\rho = 0.1$, $\rho = 0.5$, $\rho = 0.9$, $\delta = 0.001$, $\delta = 0.005$, and $\delta = 0.009$, for the training processes. To demonstrate our method, a network structure $(B_{2^j})_{-1 \le j \le 0}$, called the 2-level network set, was used. As described in Section 3.4, we first train the coarser level network $B_{2^{-1}}$ and then the finer one $B_{2^0}$. Also, a 1-level network $B_1$, that is a traditional back-propagation network, was built as a control to compare the performance with the 2-level network set. We used all three schemes, that is shown in Section 3.5.2, for the transformation of connection weights between the hidden and the output layers. The training processes of all networks were repeated until the sum square error (SSE) was smaller than 0.001, i.e., $\varepsilon = 0.001$. Therefore, for each experiment, there was a total of 16 training jobs to be carried out (Table 4.2).

Table 4.2: Configurations of each experiment for the XOR problem.

| Job no. | Network type | Transformation scheme | $\rho$ | $\delta$ | $\varepsilon$ |
|---|---|---|---|---|---|
| 1 | 2-level | copy | 0.1 | 0.005 | 0.001 |
| 2 | 2-level | average | 0.1 | 0.005 | 0.001 |
| 3 | 2-level | scale | 0.1 | 0.005 | 0.001 |
| 4 | 2-level | copy | 0.5 | 0.001 | 0.001 |
| 5 | 2-level | average | 0.5 | 0.001 | 0.001 |
| 6 | 2-level | scale | 0.5 | 0.001 | 0.001 |
| 7 | 2-level | copy | 0.5 | 0.005 | 0.001 |
| 8 | 2-level | average | 0.5 | 0.005 | 0.001 |
| 9 | 2-level | scale | 0.5 | 0.005 | 0.001 |
| 10 | 2-level | copy | 0.5 | 0.009 | 0.001 |
| 11 | 2-level | average | 0.5 | 0.009 | 0.001 |
| 12 | 2-level | scale | 0.5 | 0.009 | 0.001 |
| 13 | 2-level | copy | 0.9 | 0.005 | 0.001 |
| 14 | 2-level | average | 0.9 | 0.005 | 0.001 |
| 15 | 2-level | scale | 0.9 | 0.005 | 0.001 |
| 16 | 1-level | — | — | — | 0.001 |

The input vectors for all finer level networks $B_1$ of each experiment are the

one shown in Table 4.1. Hence, the size of the input layer for these networks is set to 2. As the input vectors for the coarser level networks $B_{2-1}$ are with lower resolution, the size of the input layer for these networks is set to 1. For all networks in each experiment, the sizes of the hidden and the output layers are set to 2 and 1 respectively. The learning rate for all of them is set to 0.25 and the momentum rate is set to 0.9 (Table 4.3).

Table 4.3: The structure of the back-propagation networks for the XOR problem.

| BP network | Size | | | Learning rate | Momentum rate |
|---|---|---|---|---|---|
| | Input layer | Hidden layer | Output layer | | |
| $B_1$ | 2 | 2 | 1 | 0.25 | 0.9 |
| $B_{2-1}$ | 1 | 2 | 1 | 0.25 | 0.9 |

### 4.3.2 Experimental Results

All of the experiments were run on a SPARCstation 1+ with 16MB memory. For each training job of each experiment, we have run it for 5000 times with different sets of initial connection weights for the back-propagation networks and finally took the mean value of the convergence time as the results. Table 4.4 show the training results for the two experiments. The convergence time[1] for each training job is presented and a performance index, a ratio to the convergence time of the 1-level network, is measured.

In general, the proposed multiresolution learning method improves the training performance of back-propagation networks when solving the XOR problem, except for the Job 3, 9, and 12 of Experiment 2 that they were a little bit slower than the traditional back-propagation network (Job 16). Figure 4.1 and 4.2

[1]The convergence time for the training jobs which have adopted the multiresolution learning method is measured as the sum of the training time for each network involved and the transformation time for connection weights between networks.

show the comparisons of the training performance with different values of $\rho$ and $\delta$ respectively. As shown in these two figures, it is believed that some optimal values for both $\rho$ and $\delta$ are existed. Figure 4.3 and 4.4 are used to compare the training performance among different transformation schemes between the hidden and the output layers. Generally, the performance of the average scheme is the best one among all three schemes while the performance of the scale scheme is the worst one for the XOR problem. With Figure 4.5, 4.6 and 4.7, it is shown that the Daubechies' impulse response is adequate to be used with the copy scheme and the scale scheme. On the other hand, the average splitting method is adequate to be used with the average scheme.

Table 4.4: Training results for the XOR problem after 5000 runs.

| Experiment 1 (Daubechies' impulse response) | | |
|---|---|---|
| Job no. | Convergence time (sec) | Performance index |
| 1 | 14.27 | 1.11 |
| 2 | 13.75 | 1.15 |
| 3 | 14.27 | 1.11 |
| 4 | 13.05 | 1.21 |
| 5 | 15.26 | 1.04 |
| 6 | 13.75 | 1.15 |
| 7 | 14.52 | 1.09 |
| 8 | 13.87 | 1.14 |
| 9 | 14.63 | 1.08 |
| 10 | 14.05 | 1.12 |
| 11 | 13.84 | 1.14 |
| 12 | 14.23 | 1.11 |
| 13 | 12.99 | 1.22 |
| 14 | 15.83 | 1.00 |
| 15 | 13.44 | 1.18 |
| 16 | 15.80 | 1.00 |

| Experiment 2 (the average splitting method) | | |
|---|---|---|
| Job no. | Convergence time (sec) | Performance index |
| 1 | 15.04 | 1.05 |
| 2 | 11.72 | 1.35 |
| 3 | 16.41 | 0.96 |
| 4 | 14.26 | 1.11 |
| 5 | 13.60 | 1.16 |
| 6 | 15.65 | 1.01 |
| 7 | 14.61 | 1.08 |
| 8 | 12.08 | 1.31 |
| 9 | 16.64 | 0.95 |
| 10 | 15.46 | 1.02 |
| 11 | 11.93 | 1.32 |
| 12 | 16.40 | 0.96 |
| 13 | 14.43 | 1.09 |
| 14 | 14.02 | 1.13 |
| 15 | 15.35 | 1.03 |
| 16 | 15.80 | 1.00 |

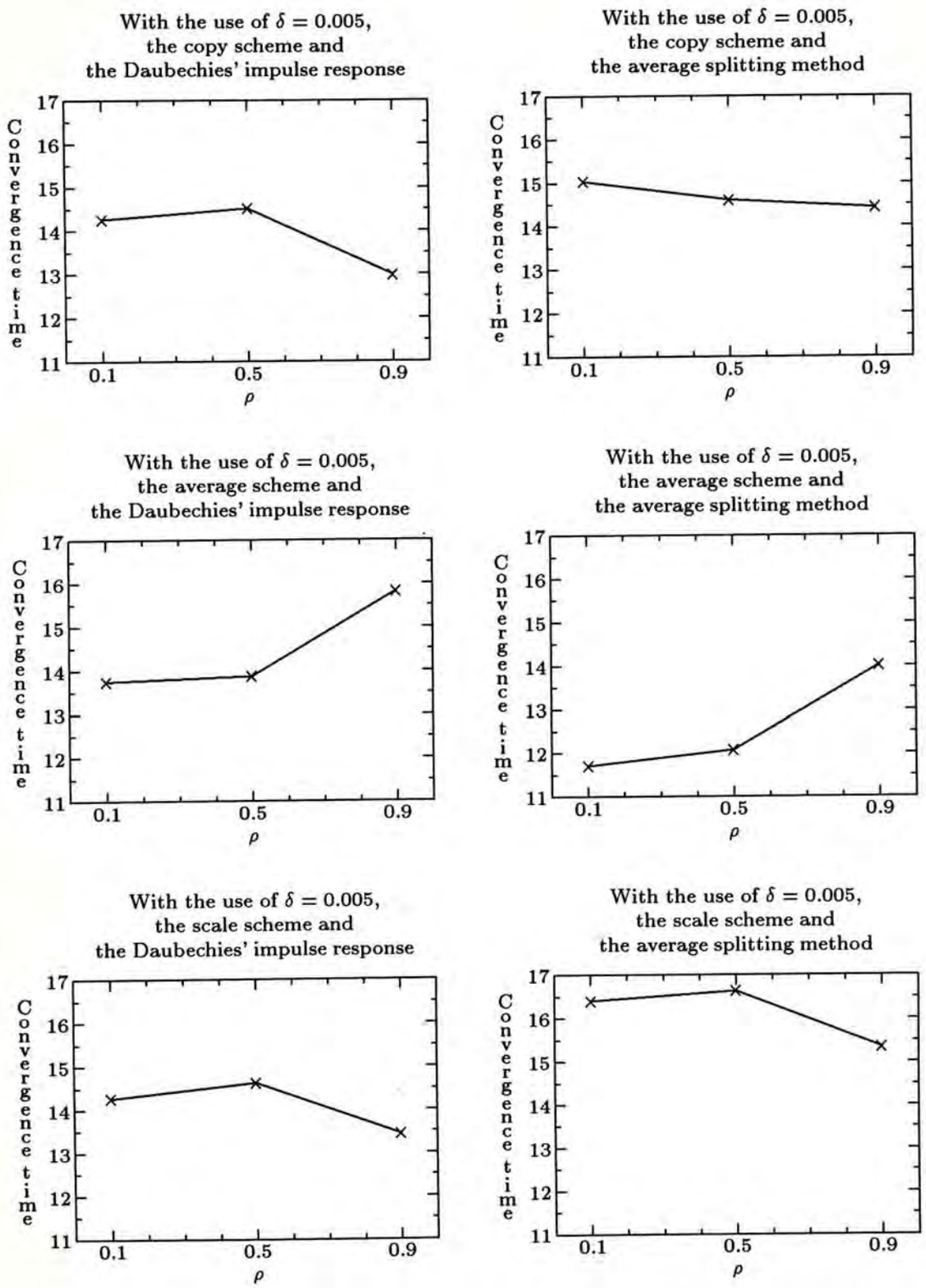



Figure 4.1: Comparisons of the training performance with different values of $\rho$ for the XOR problem.

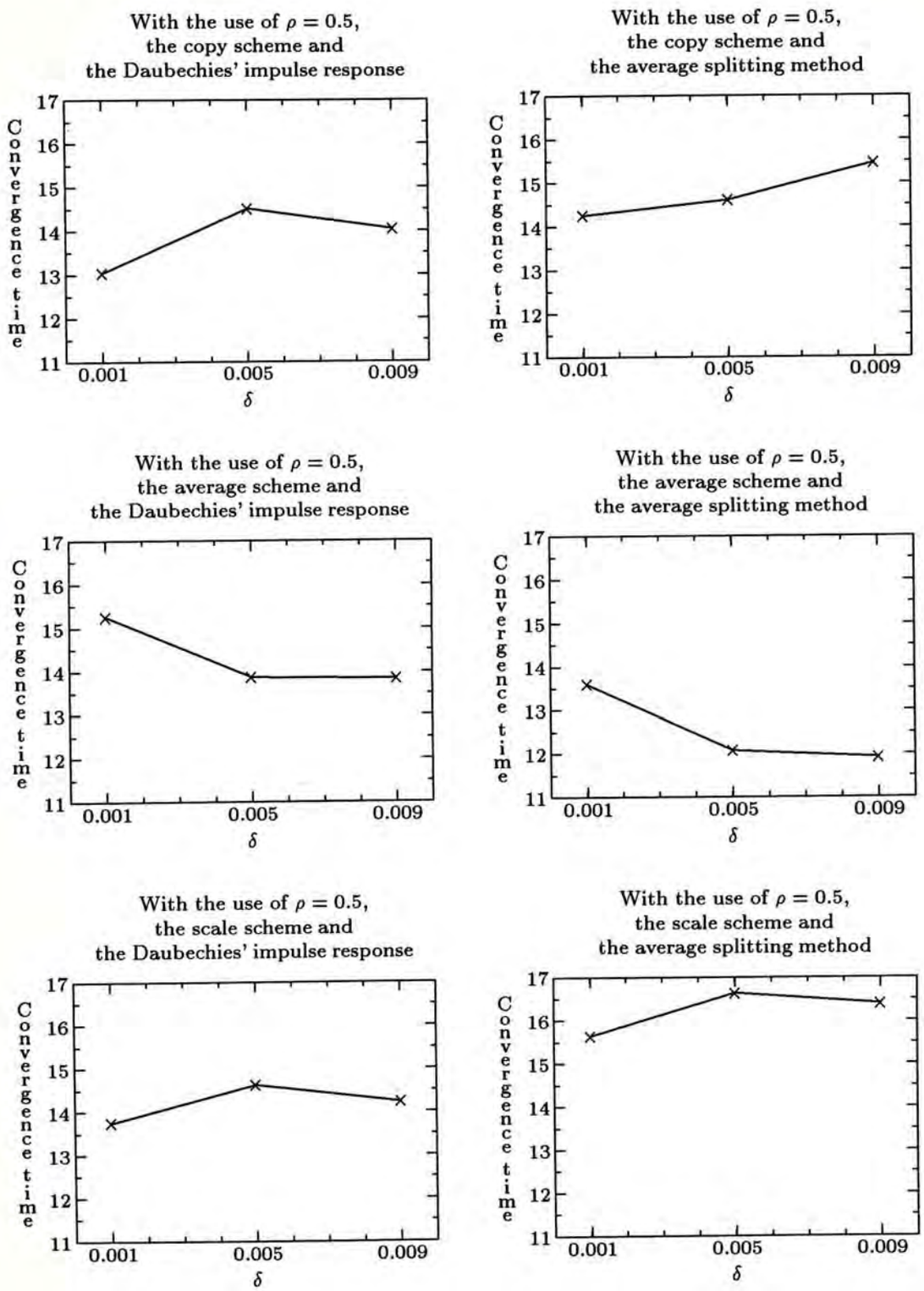



Figure 4.2: Comparisons of the training performance with different values of $\delta$ for the XOR problem.

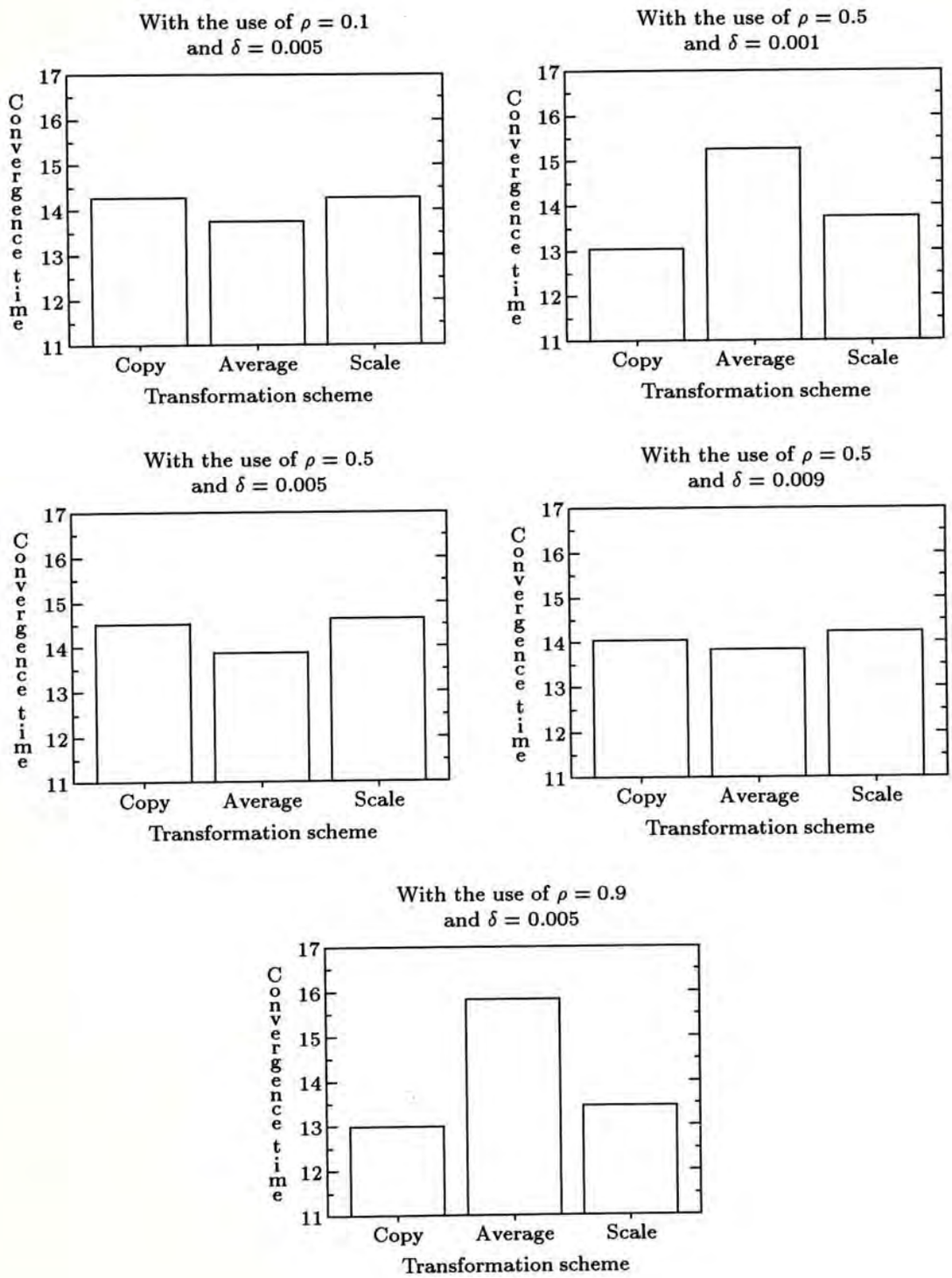



Figure 4.3: Comparisons of the training performance with different transformation schemes for the XOR problem of using the Daubechies' impulse response.

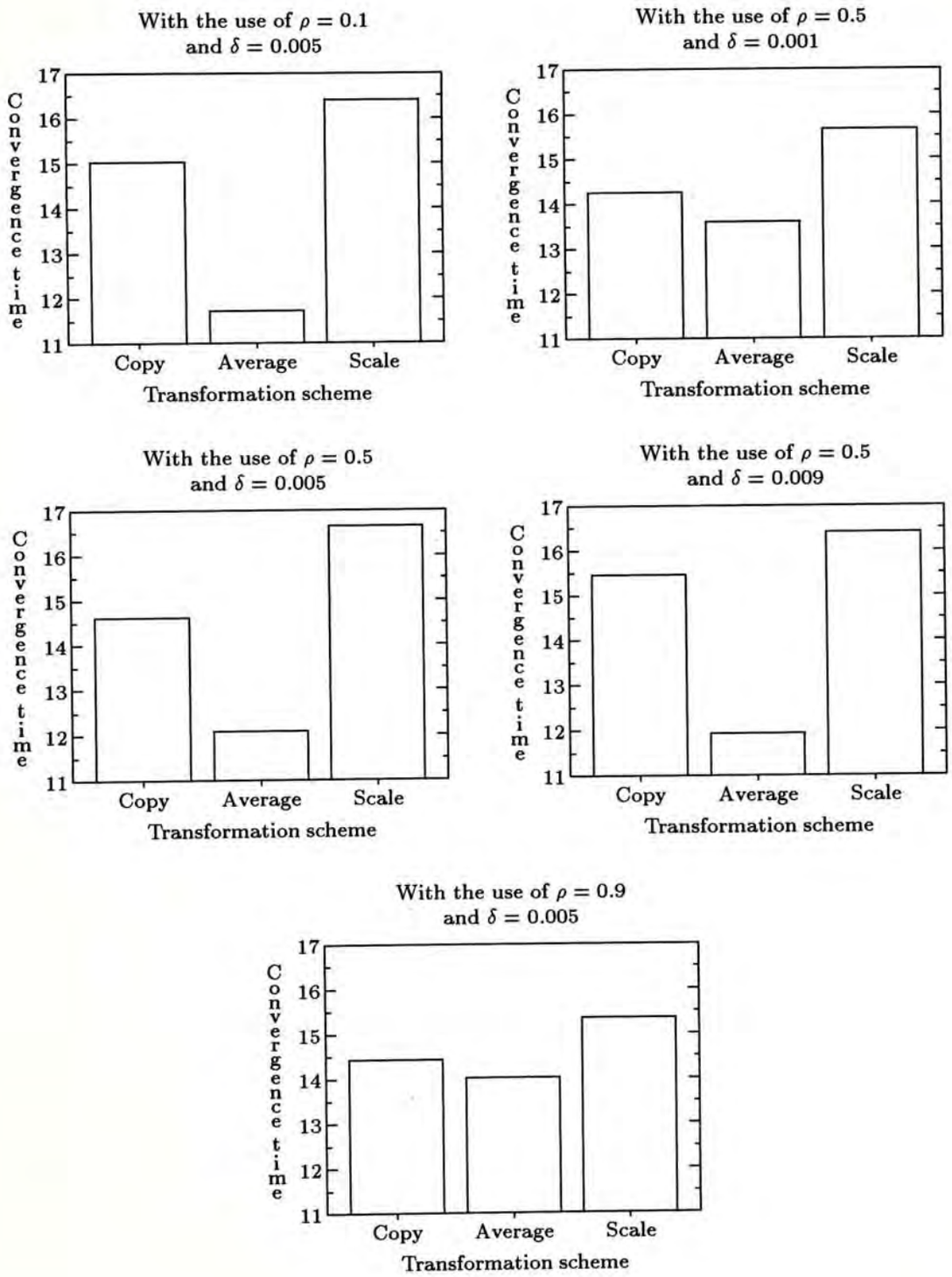



Figure 4.4: Comparisons of the training performance with different transformation schemes for the XOR problem of using the average splitting method.

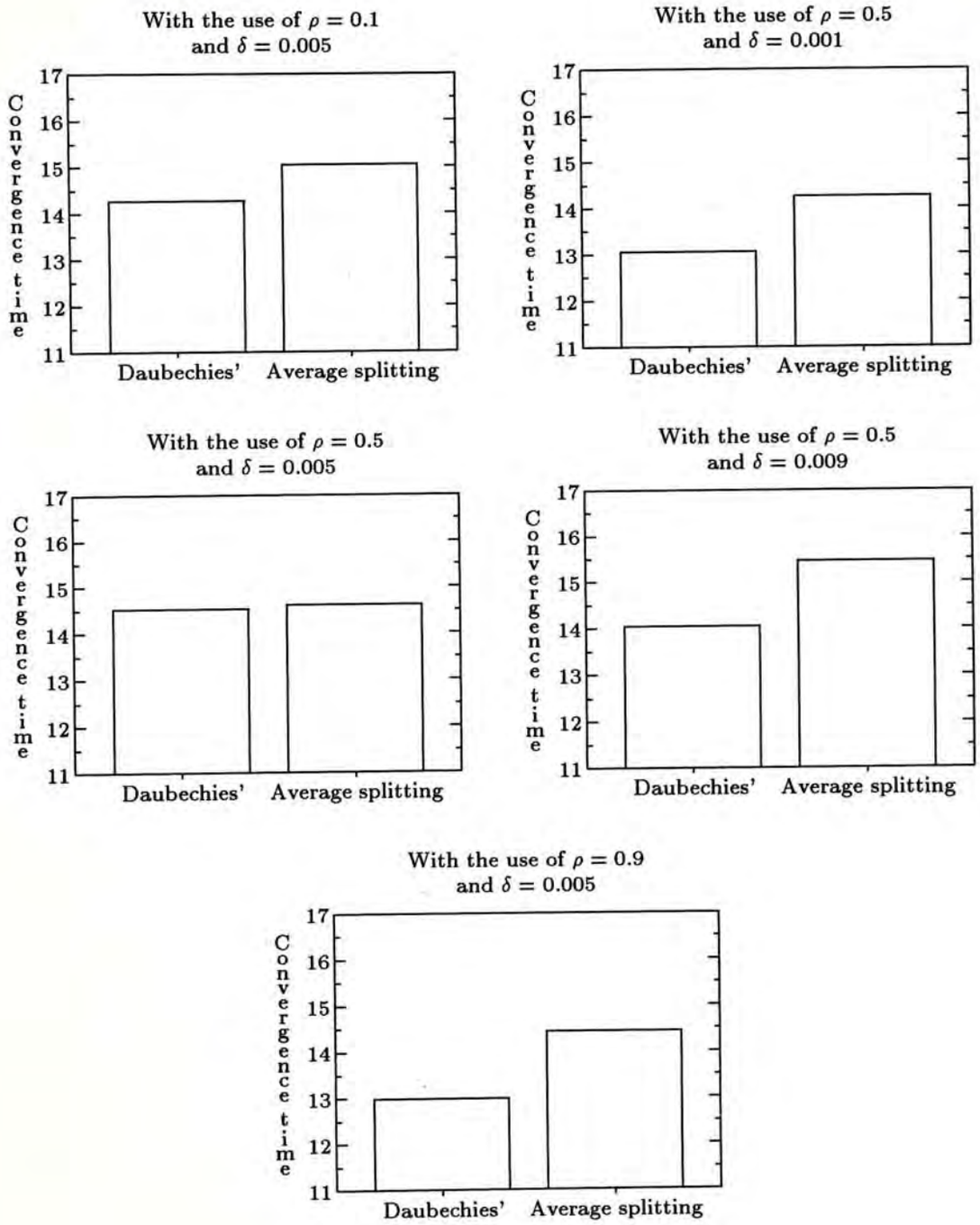



Figure 4.5: Comparisons of the training performance with different sets of impulse response coefficients for the XOR problem of using the copy scheme.

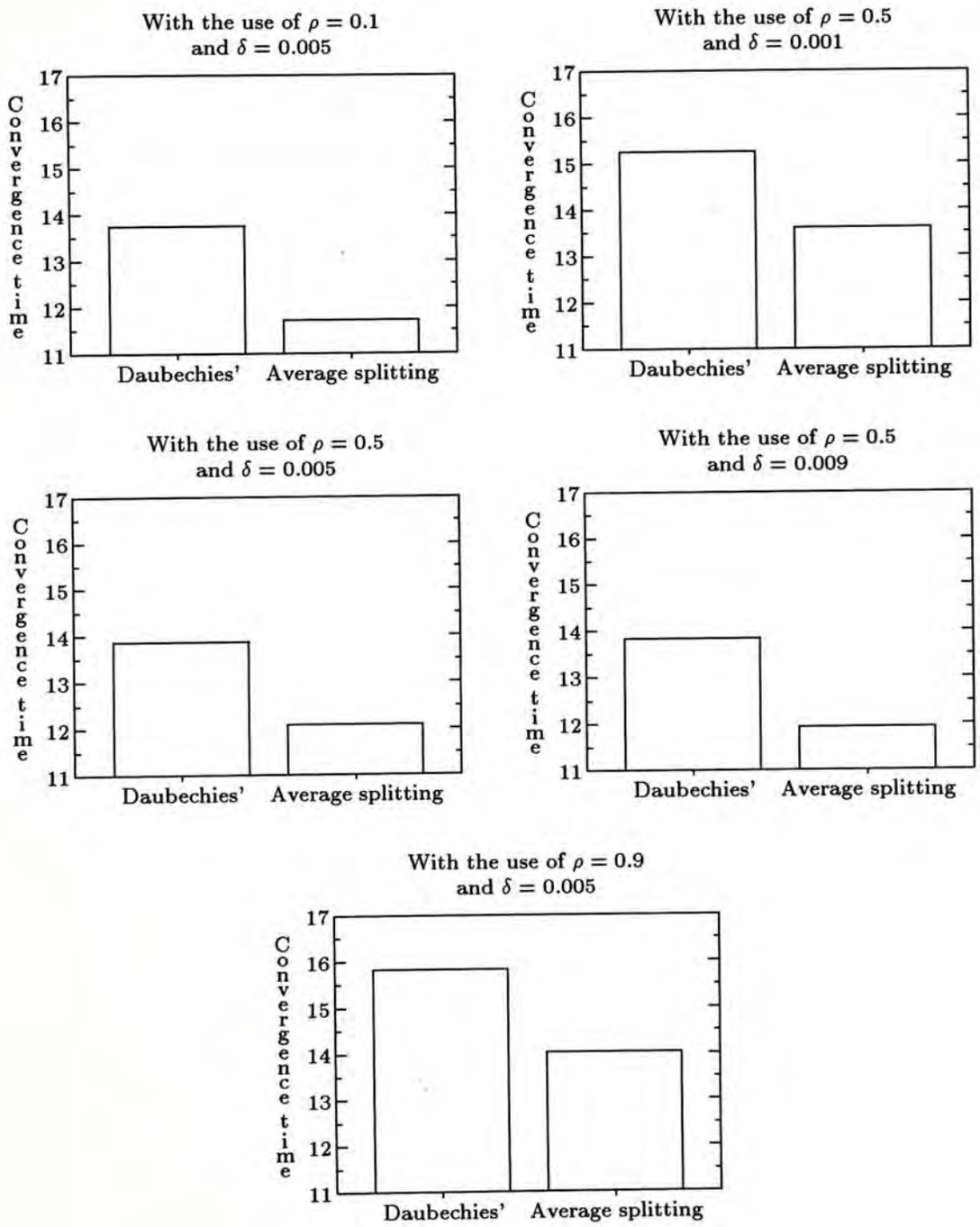



Figure 4.6: Comparisons of the training performance with different sets of impulse response coefficients for the XOR problem of using the average scheme.

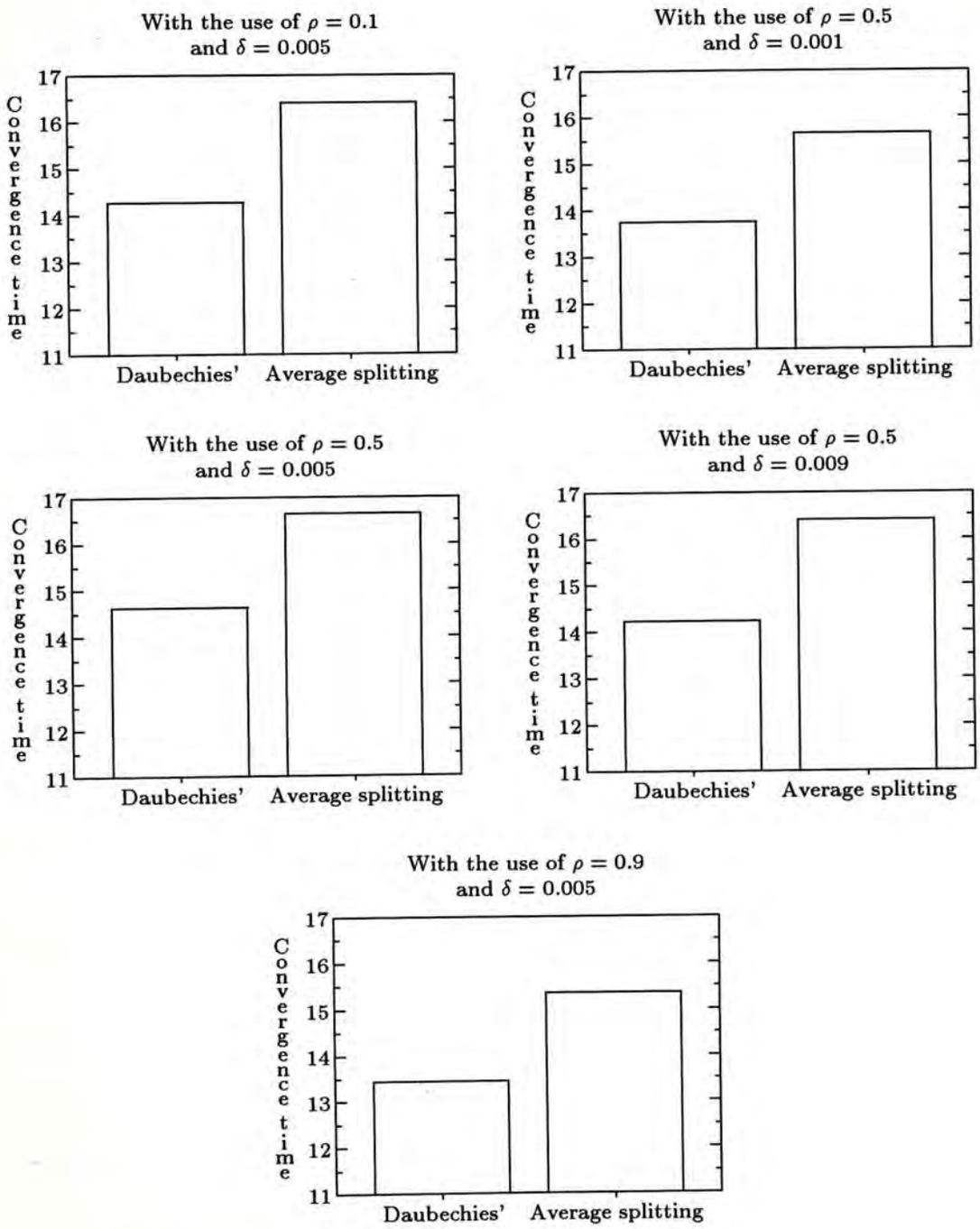



Figure 4.7: Comparisons of the training performance with different sets of impulse response coefficients for the XOR problem of using the scale scheme.

## 4.4 Numeric Recognition Problem

In this section, a numeric recognition problem is used as an example. For this problem, 10 binary patterns of numbers, from 0 to 9, were selected as training examples and the size of them was $32 \times 32$.

### 4.4.1 Setting of Experiments

Two experiments were carried out with the impulse responses from Daubechies (4.6) and the average splitting method (4.7) respectively. In each experiment, we selected different history factors $\rho$ and intermediate stopping criteria $\delta$, i.e., $\rho = 0.1$, $\rho = 0.5$, $\rho = 0.9$, $\delta = 0.001$, $\delta = 0.005$, and $\delta = 0.009$, for the training processes. To demonstrate the proposed learning method, two types of network structure were used, $(B_{2^j})_{-2 \le j \le 0}$ and $(B_{2^j})_{-1 \le j \le 0}$, and were called the 3-level network set and the 2-level network set respectively. For the 2-level network set, we first train the coarser level network $B_{2^{-1}}$ and then the finer one $B_{2^0}$. For the 3-level network set, the training sequence is from $B_{2^{-2}}$ to $B_{2^0}$. Also, a 1-level network $B_1$, which is a traditional back-propagation network was built as a control to compare the performance with the 3-level and the 2-level network sets. We used all three schemes, that is shown is Section 3.5.2, for the transformation of connection weights between the hidden and output layers. The training processes of all networks were repeated until the SSE was smaller than 0.001, i.e., $\varepsilon = 0.001$. Hence, for each experiment, there was a total of 31 training jobs to be carried out (Table 4.5).

Since the binary patterns used in the experiments were all in two dimensions, the multiresolution technique described in Section 2.3 cannot be applied to them directly. However, it has been shown that the two-dimensional multiresolution transform can be seen as a one-dimensional multiresolution transform along the $x$ and $y$ axes [Mall89a]. We first convolve the rows of binary patterns with

Table 4.5: Configurations of each experiment for the numeric recognition problem.

| Job no. | Network type | Transformation scheme | $\rho$ | $\delta$ | $\varepsilon$ |
|---|---|---|---|---|---|
| 1 | 3-level | copy | 0.1 | 0.005 | 0.001 |
| 2 | 3-level | average | 0.1 | 0.005 | 0.001 |
| 3 | 3-level | scale | 0.1 | 0.005 | 0.001 |
| 4 | 3-level | copy | 0.5 | 0.001 | 0.001 |
| 5 | 3-level | average | 0.5 | 0.001 | 0.001 |
| 6 | 3-level | scale | 0.5 | 0.001 | 0.001 |
| 7 | 3-level | copy | 0.5 | 0.005 | 0.001 |
| 8 | 3-level | average | 0.5 | 0.005 | 0.001 |
| 9 | 3-level | scale | 0.5 | 0.005 | 0.001 |
| 10 | 3-level | copy | 0.5 | 0.009 | 0.001 |
| 11 | 3-level | average | 0.5 | 0.009 | 0.001 |
| 12 | 3-level | scale | 0.5 | 0.009 | 0.001 |
| 13 | 3-level | copy | 0.9 | 0.005 | 0.001 |
| 14 | 3-level | average | 0.9 | 0.005 | 0.001 |
| 15 | 3-level | scale | 0.9 | 0.005 | 0.001 |
| 16 | 2-level | copy | 0.1 | 0.005 | 0.001 |
| 17 | 2-level | average | 0.1 | 0.005 | 0.001 |
| 18 | 2-level | scale | 0.1 | 0.005 | 0.001 |
| 19 | 2-level | copy | 0.5 | 0.001 | 0.001 |
| 20 | 2-level | average | 0.5 | 0.001 | 0.001 |
| 21 | 2-level | scale | 0.5 | 0.001 | 0.001 |
| 22 | 2-level | copy | 0.5 | 0.005 | 0.001 |
| 23 | 2-level | average | 0.5 | 0.005 | 0.001 |
| 24 | 2-level | scale | 0.5 | 0.005 | 0.001 |
| 25 | 2-level | copy | 0.5 | 0.009 | 0.001 |
| 26 | 2-level | average | 0.5 | 0.009 | 0.001 |
| 27 | 2-level | scale | 0.5 | 0.009 | 0.001 |
| 28 | 2-level | copy | 0.9 | 0.005 | 0.001 |
| 29 | 2-level | average | 0.9 | 0.005 | 0.001 |
| 30 | 2-level | scale | 0.9 | 0.005 | 0.001 |
| 31 | 1-level | — | — | — | 0.001 |

a one-dimensional filter $\tilde{H}$, retain every other row, convolve the columns of the resulting signals with another one-dimensional filter and retain every other column. Hence, two sets of patterns can be collected with sizes $16 \times 16$ and $8 \times 8$ and they are used as input vectors for networks $B_{2-1}$ and $B_{2-2}$ respectively. For all networks in each experiment, the sizes for the hidden layer and the output layer were set to 10 and 10 respectively. The learning rate for all of them is set to 0.35 and the momentum rate is set to 0.9 (Table 4.6).

Table 4.6: The structure of the back-propagation networks for the numeric recognition problem.

| BP network | Size | | | Learning rate | Momentum rate |
|---|---|---|---|---|---|
| | Input layer | Hidden layer | Output layer | | |
| $B_1$ | $32 \times 32$ | 10 | 10 | 0.35 | 0.9 |
| $B_{2-1}$ | $16 \times 16$ | 10 | 10 | 0.35 | 0.9 |
| $B_{2-2}$ | $8 \times 8$ | 10 | 10 | 0.35 | 0.9 |

### 4.4.2 Experimental Results

All of the experiments were run on a SPARCstation 10/30 with 32MB memory. For each training job of each experiment, we have run it for 120 times with different sets of initial connection weights for the back-propagation networks and finally took the mean value of the convergence time as the results. Table 4.7 shows the training results of the two experiments. The convergence time[2] for each training job is presented and a performance index, a ratio to the convergence time of the 1-level network, is measured.

It is shown in Table 4.7 that the multiresolution learning method improves the training performance of back-propagation networks significantly, from the

[2]The convergence time for the training jobs which have adopted the multiresolution learning method is measured as the sum of the training time for each network involved and the transformation time for connection weights between networks.

Table 4.7: Training results for the numeric recognition problem after 120 runs.

| Experiment 1 (Daubechies' impulse response) | | |
|---|---|---|
| Job no. | Convergence time (sec) | Performance index |
| 1 | 350.07 | 4.11 |
| 2 | 666.64 | 2.16 |
| 3 | 630.77 | 2.28 |
| 4 | 144.94 | 9.92 |
| 5 | 731.99 | 1.96 |
| 6 | 613.87 | 2.34 |
| 7 | 332.14 | 4.33 |
| 8 | 691.22 | 2.08 |
| 9 | 623.02 | 2.31 |
| 10 | 461.88 | 3.11 |
| 11 | 967.85 | 1.49 |
| 12 | 739.97 | 1.94 |
| 13 | 282.87 | 5.09 |
| 14 | 664.81 | 2.16 |
| 15 | 609.93 | 2.36 |
| 16 | 639.27 | 2.25 |
| 17 | 924.62 | 1.56 |
| 18 | 674.52 | 2.13 |
| 19 | 498.82 | 2.88 |
| 20 | 779.65 | 1.85 |
| 21 | 675.79 | 2.13 |
| 22 | 693.56 | 2.07 |
| 23 | 878.38 | 1.64 |
| 24 | 673.50 | 2.14 |
| 25 | 870.13 | 1.65 |
| 26 | 976.48 | 1.47 |
| 27 | 767.38 | 1.88 |
| 28 | 634.77 | 2.27 |
| 29 | 959.69 | 1.50 |
| 30 | 673.51 | 2.26 |
| 31 | 1437.44 | 1.00 |

| Experiment 2 (the average splitting method) | | |
|---|---|---|
| Job no. | Convergence time (sec) | Performance index |
| 1 | 401.08 | 3.59 |
| 2 | 667.52 | 2.16 |
| 3 | 638.43 | 2.25 |
| 4 | 147.86 | 9.73 |
| 5 | 689.17 | 2.09 |
| 6 | 592.48 | 2.43 |
| 7 | 393.92 | 3.65 |
| 8 | 672.43 | 2.14 |
| 9 | 649.16 | 2.22 |
| 10 | 480.56 | 2.99 |
| 11 | 762.49 | 1.89 |
| 12 | 712.23 | 2.02 |
| 13 | 327.59 | 4.39 |
| 14 | 667.90 | 2.15 |
| 15 | 622.98 | 2.31 |
| 16 | 656.60 | 2.19 |
| 17 | 904.94 | 1.59 |
| 18 | 688.19 | 2.09 |
| 19 | 442.41 | 3.25 |
| 20 | 908.51 | 1.58 |
| 21 | 676.41 | 2.13 |
| 22 | 609.22 | 2.36 |
| 23 | 1070.28 | 1.34 |
| 24 | 677.56 | 2.12 |
| 25 | 678.58 | 2.12 |
| 26 | 1068.83 | 1.35 |
| 27 | 753.83 | 1.91 |
| 28 | 553.48 | 2.60 |
| 29 | 962.09 | 1.50 |
| 30 | 659.53 | 2.18 |
| 31 | 1437.44 | 1.00 |

least improvement of 1.34 times faster (Job 23 of Experiment 2) to the best improvement of 9.92 times faster (Job 4 of Experiment 1) than the traditional back-propagation network (Job 31) for the numeric recognition problem. Figure 4.8 and 4.9 show the comparisons of the training performance with different values of $\rho$ and $\delta$ respectively. As shown in these two figures, the history factor $\rho$ does not affect the training performance too much; in other words, the training performance is not varied too much within a range of $\rho$. On the other hands, the convergence rate of the networks will increase as the intermediate stopping criteria $\delta$ decreases. Figure 4.10 and 4.11 are used to compare the training performance among different transformation schemes between the hidden and the output layers. Generally, the performance of the copy scheme is the best one among all three schemes while the performance of the average scheme is the worst one for the numeric recognition problem. With Figure 4.12, 4.13 and 4.14, it is shown that the training performance is not varied too much no matter which impulse response is selected (the Daubechies' impulse response and the average splitting method). It can also be shown from Figure 4.8 to 4.14 that the training performance increases as the network level increases, e.g., the convergence time for a 3-level network is shorter than the one for a 2-level network.

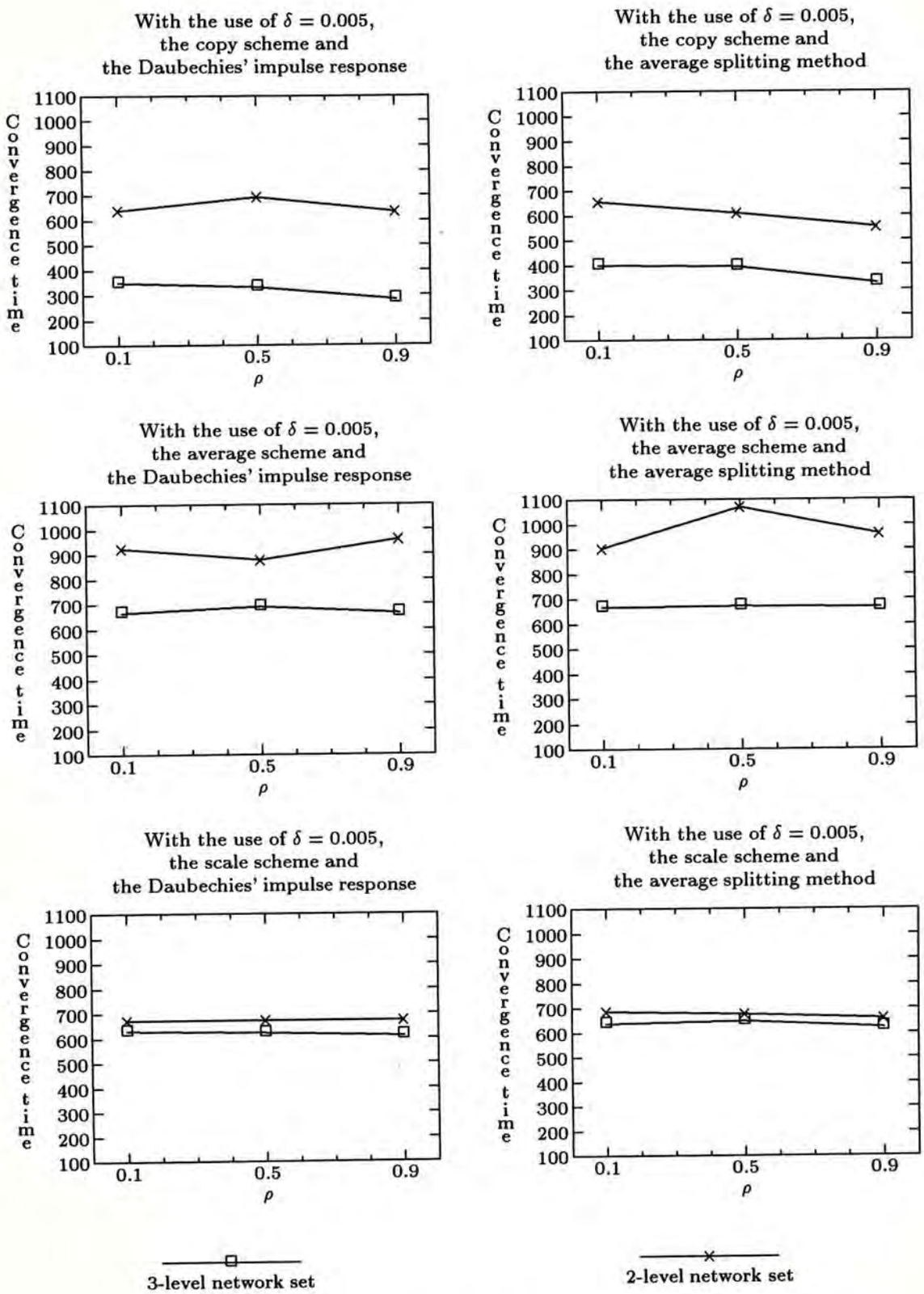



Figure 4.8: Comparisons of the training performance with different values of $\rho$ for the numeric recognition problem.

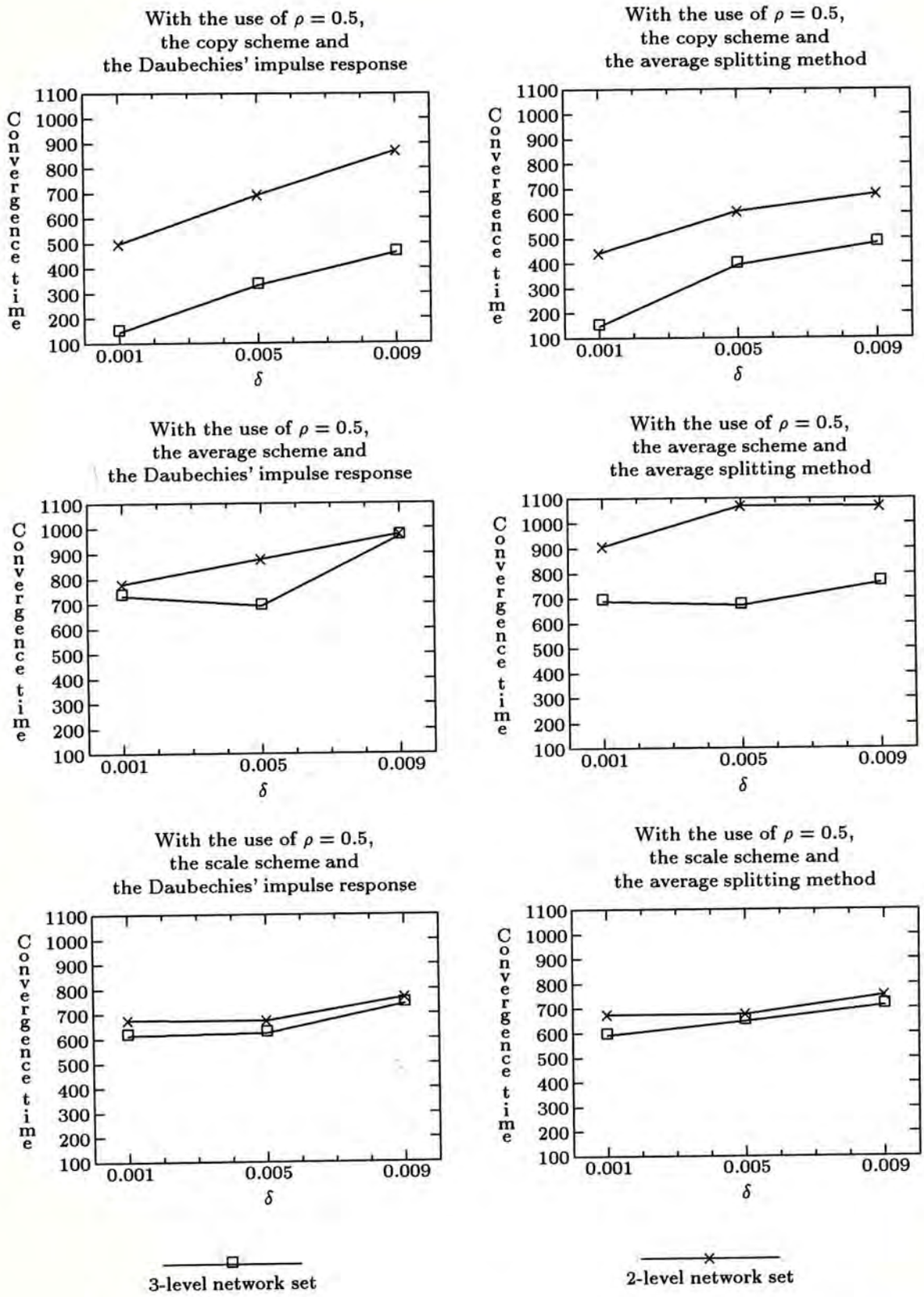



Figure 4.9: Comparisons of the training performance with different values of $\delta$ for the numeric recognition problem.

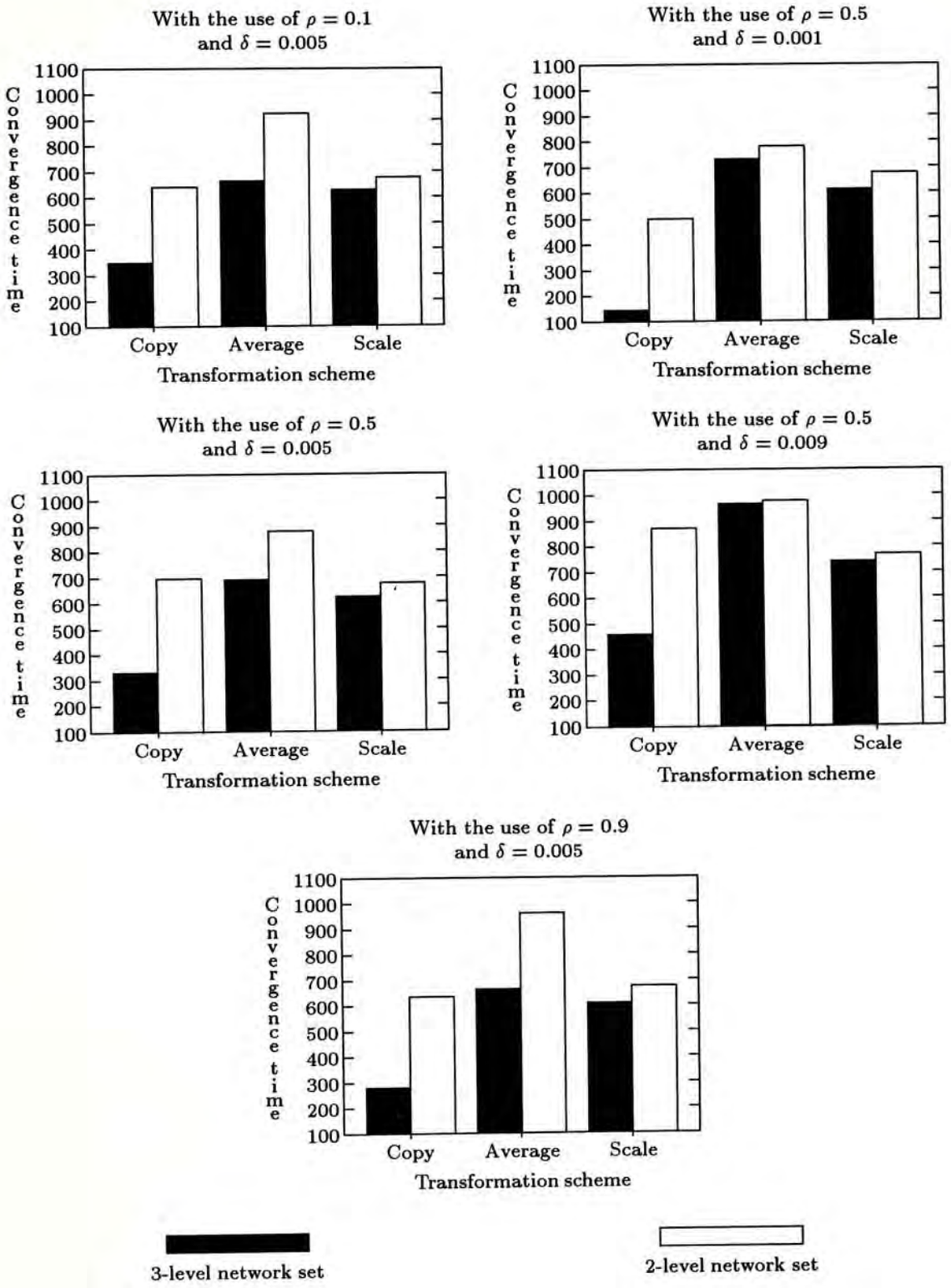



Figure 4.10: Comparisons of the training performance with different transformation schemes for the numeric recognition problem of using the Daubechies' impulse response.

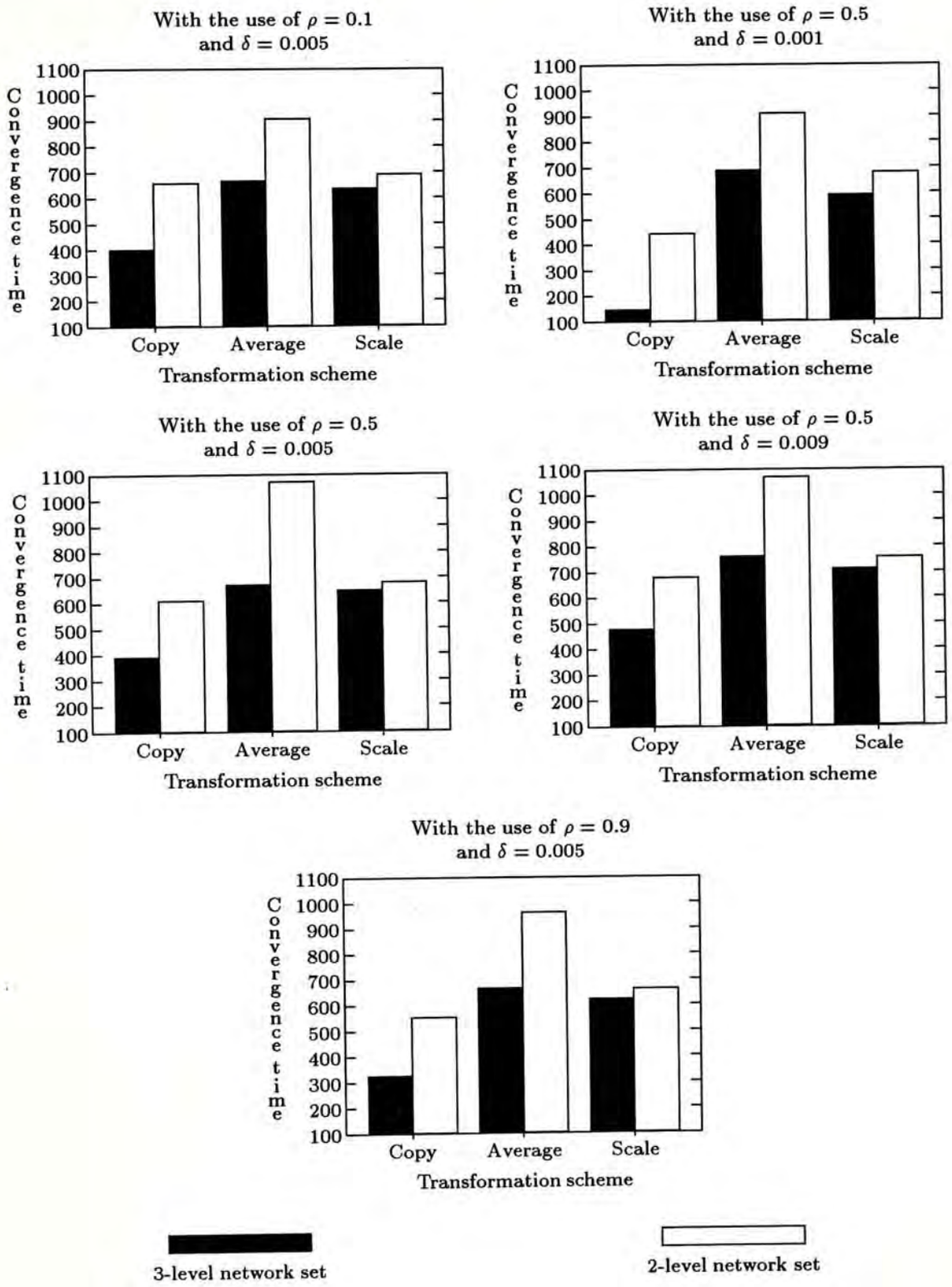



Figure 4.11: Comparisons of the training performance with different transformation schemes for the numeric recognition problem of using the average splitting method.

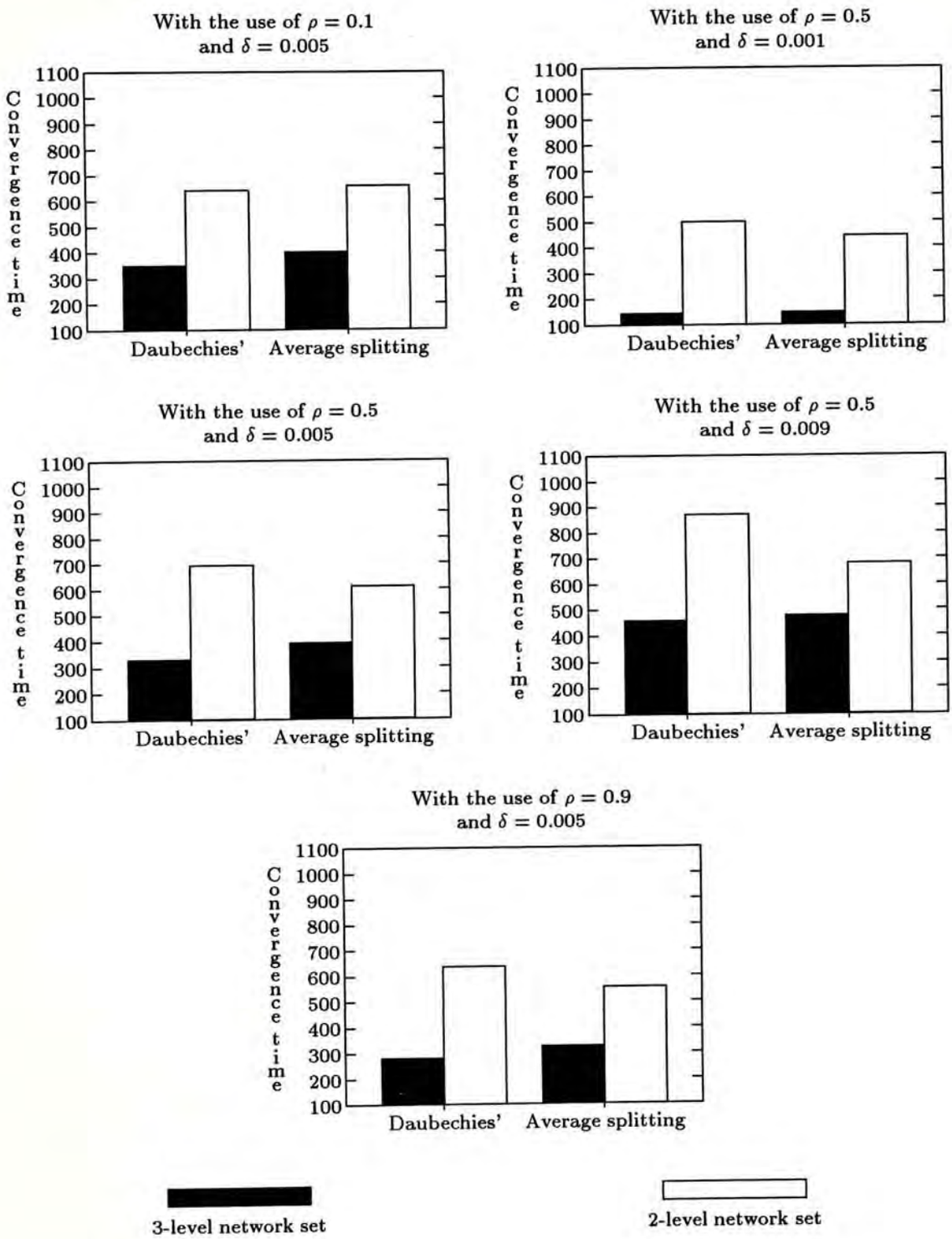



Figure 4.12: Comparisons of the training performance with different sets of impulse response coefficients for the numeric recognition problem of using the copy scheme.

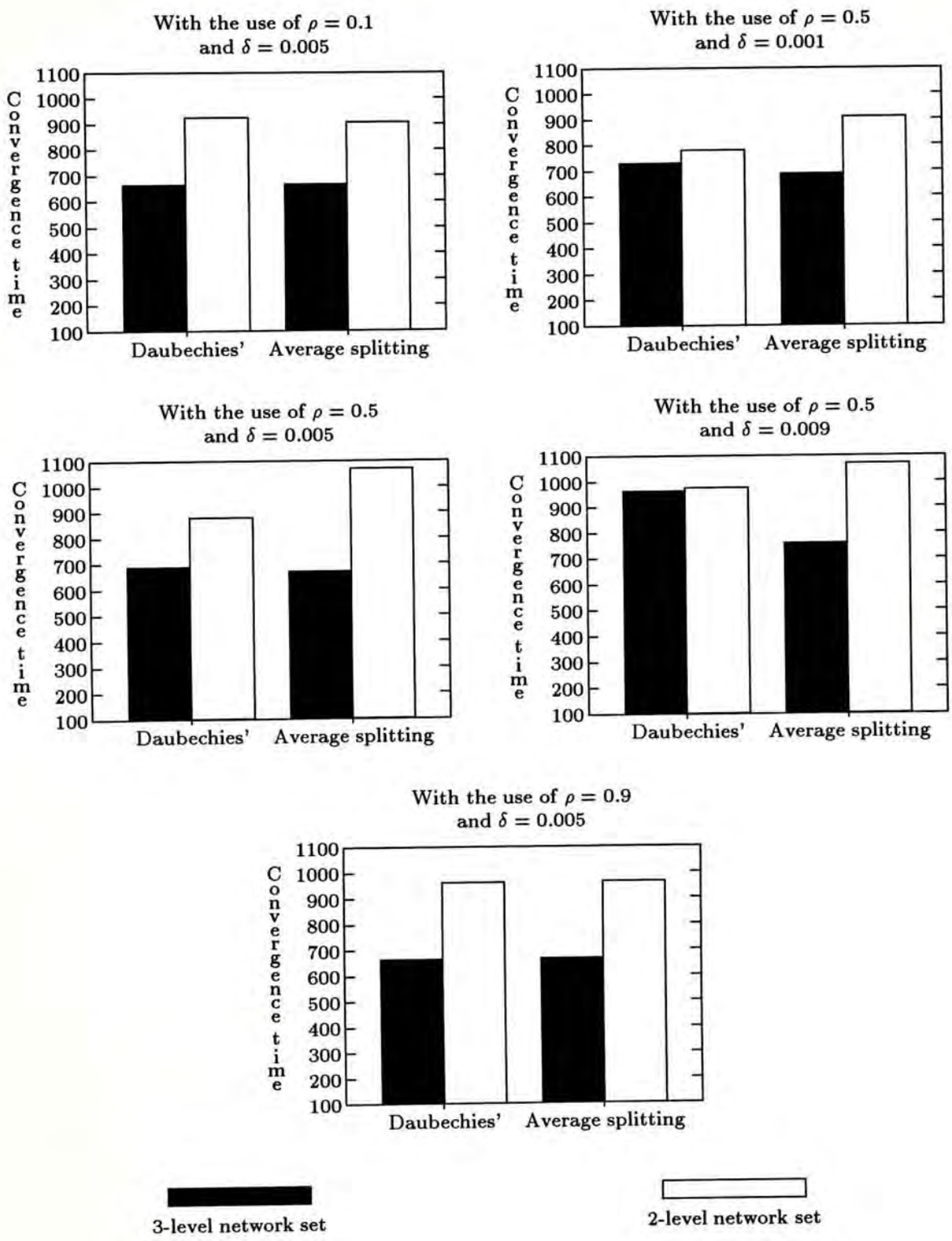



Figure 4.13: Comparisons of the training performance with different sets of impulse response coefficients for the numeric recognition problem of using the average scheme.

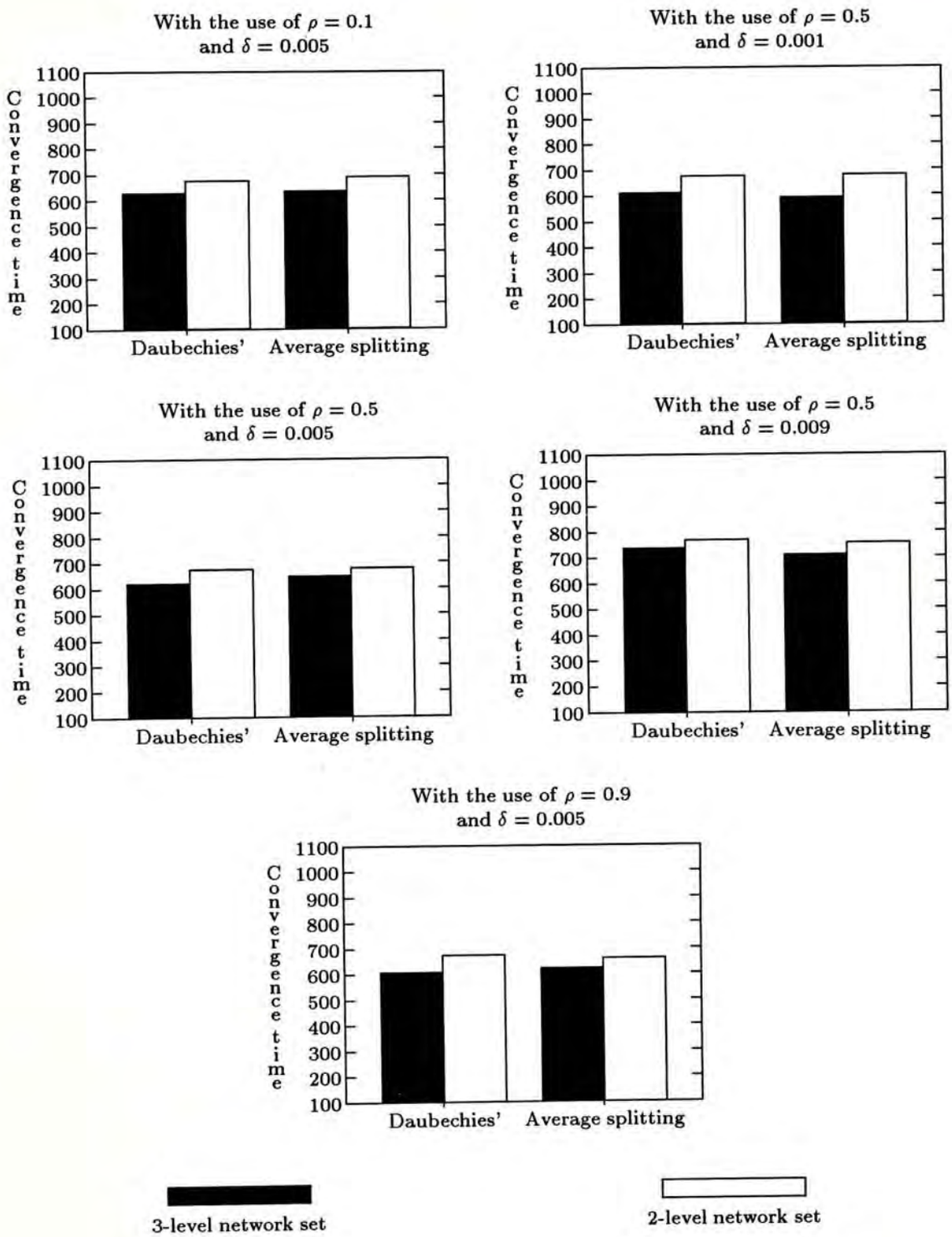



Figure 4.14: Comparisons of the training performance with different sets of impulse response coefficients for the numeric recognition problem of using the scale scheme.

To demonstrate the recognition ability of the trained networks, we constructed a set of testing patterns with 15% uniform noise. The sample size of these testing patterns for each experiment is 500. Table 4.8 shows the recognition results for all 120 runs of the numeric recognition problem. The sum squared errors (SSE) are measured among the output neurons for all samples as shown in (4.12).

$$SSE = \sum_{i=1}^{M} \sum_{n=1}^{L} (o_{in} - d_{in})^2 \tag{4.12}$$

where $M$ is the sample size and $L$ is the size of output layer. An index, called the recognition index, is calculated as a ratio to the sum squared error of the 1-level network for comparisons.

In general, the proposed multiresolution learning method improves the recognition performance of back-propagation networks in solving the numeric recognition, especially when the average and the scale schemes are adopted. Figure 4.15 and 4.16 show the comparisons of the training performance with different values of $\rho$ and $\delta$ respectively. As shown in these two figures, the history factor $\rho$ does not affect the recognition performance of the back-propagation networks too much; in other words, the recognition ability is not varied too much within a range of $\rho$. On the other hand, the recognition ability of the networks will increase as the intermediate stopping criteria $\delta$ increases. Figure 4.17 and 4.18 are used to compare the training performance among different transformation schemes between the hidden and the output layers. Generally, the performance of the average and the scale schemes are much better than the copy scheme for the numeric recognition problem in term of recognition. it is shown in Figure 4.19 that the average splitting method is better than the Daubechies' impulse response when using the copy scheme. On the other hand, it is shown in Figure 4.20 and 4.21 that the recognition performance is not varied too much no matter which impulse response is selected (the Daubechies' impulse response and the average splitting method) when using the average or the scale scheme.

It can also be shown from Figure 4.15 to 4.21 that the recognition performance of a 2-level network is better than the one of a 3-level network if the copy scheme is adopted. For the average and the scale schemes, the performance of them are merely the same.

Table 4.8: Recognition results for the 120 runs of the numeric recognition problem.

| Experiment 1 (Daubechies' impulse response) | | |
|---|---|---|
| Job no. | Sum squared error | Recognition index |
| 1 | 21.59 | 0.70 |
| 2 | 11.15 | 1.35 |
| 3 | 11.70 | 1.29 |
| 4 | 54.93 | 0.27 |
| 5 | 12.44 | 1.21 |
| 6 | 13.34 | 1.13 |
| 7 | 23.43 | 0.64 |
| 8 | 11.12 | 1.36 |
| 9 | 11.52 | 1.31 |
| 10 | 13.64 | 1.11 |
| 11 | 10.52 | 1.43 |
| 12 | 10.53 | 1.43 |
| 13 | 28.69 | 0.53 |
| 14 | 11.42 | 1.32 |
| 15 | 12.12 | 1.24 |
| 16 | 13.85 | 1.09 |
| 17 | 11.36 | 1.33 |
| 18 | 10.97 | 1.37 |
| 19 | 18.56 | 0.81 |
| 20 | 12.12 | 1.24 |
| 21 | 11.90 | 1.27 |
| 22 | 14.09 | 1.07 |
| 23 | 11.59 | 1.30 |
| 24 | 11.28 | 1.34 |
| 25 | 11.77 | 1.28 |
| 26 | 10.94 | 1.38 |
| 27 | 10.99 | 1.37 |
| 28 | 14.58 | 1.03 |
| 29 | 11.91 | 1.27 |
| 30 | 11.38 | 1.33 |
| 31 | 15.08 | 1.00 |

| Experiment 2 (the average splitting method) | | |
|---|---|---|
| Job no. | Sum squared error | Recognition index |
| 1 | 15.78 | 0.96 |
| 2 | 10.05 | 1.50 |
| 3 | 10.16 | 1.48 |
| 4 | 41.67 | 0.36 |
| 5 | 11.48 | 1.31 |
| 6 | 11.79 | 1.28 |
| 7 | 15.96 | 0.94 |
| 8 | 10.20 | 1.48 |
| 9 | 9.76 | 1.55 |
| 10 | 12.53 | 1.20 |
| 11 | 10.55 | 1.43 |
| 12 | 10.21 | 1.48 |
| 13 | 19.52 | 0.77 |
| 14 | 10.38 | 1.45 |
| 15 | 10.45 | 1.44 |
| 16 | 13.51 | 1.12 |
| 17 | 11.29 | 1.34 |
| 18 | 10.52 | 1.43 |
| 19 | 17.44 | 0.86 |
| 20 | 11.76 | 1.28 |
| 21 | 11.35 | 1.33 |
| 22 | 13.71 | 1.10 |
| 23 | 11.28 | 1.34 |
| 24 | 10.93 | 1.38 |
| 25 | 11.64 | 1.30 |
| 26 | 11.18 | 1.35 |
| 27 | 10.87 | 1.39 |
| 28 | 14.54 | 1.04 |
| 29 | 11.20 | 1.35 |
| 30 | 11.27 | 1.34 |
| 31 | 15.08 | 1.00 |

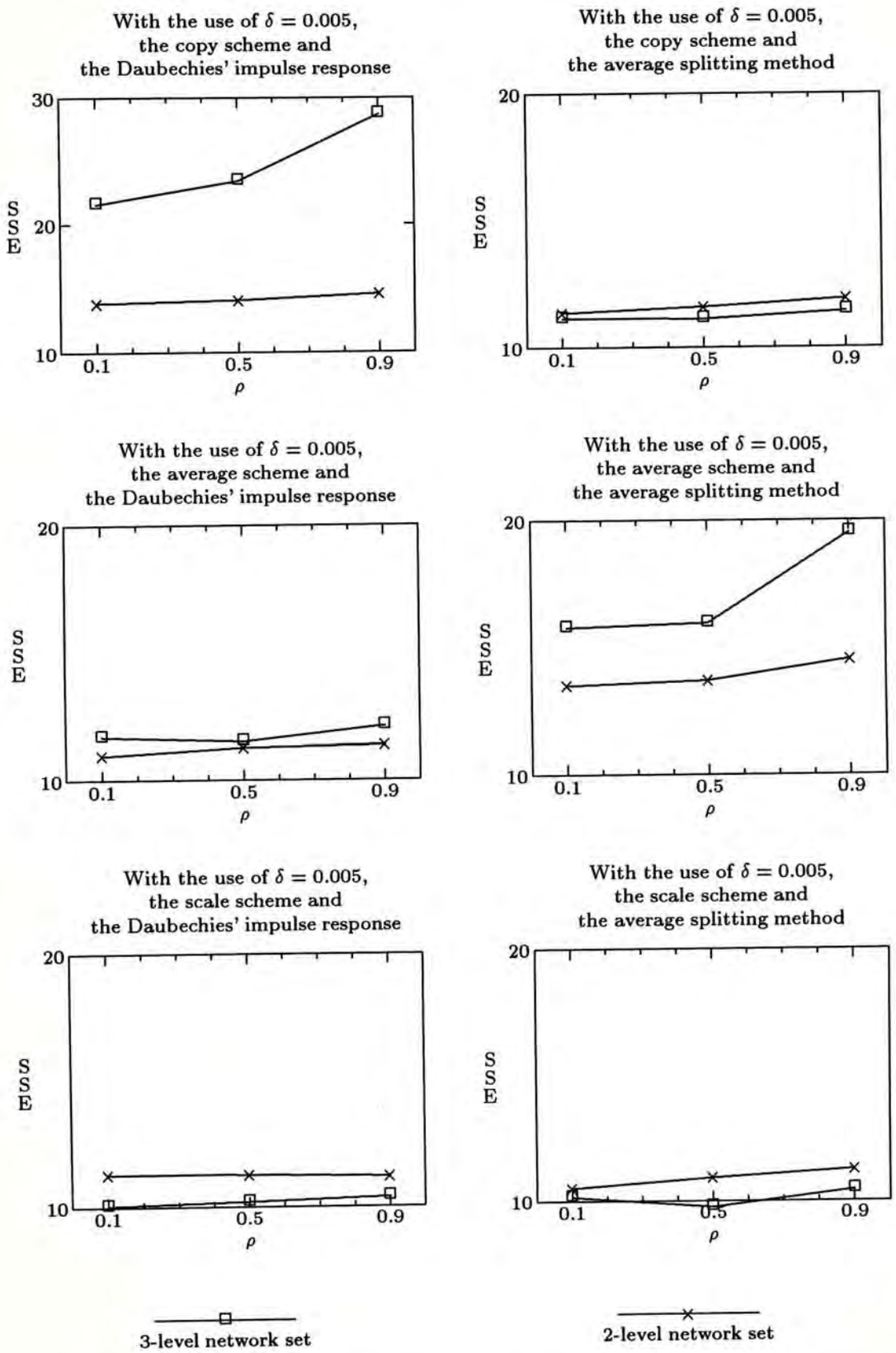



Figure 4.15: Comparisons of the recognition results with different values of $\rho$ for the numeric recognition problem.

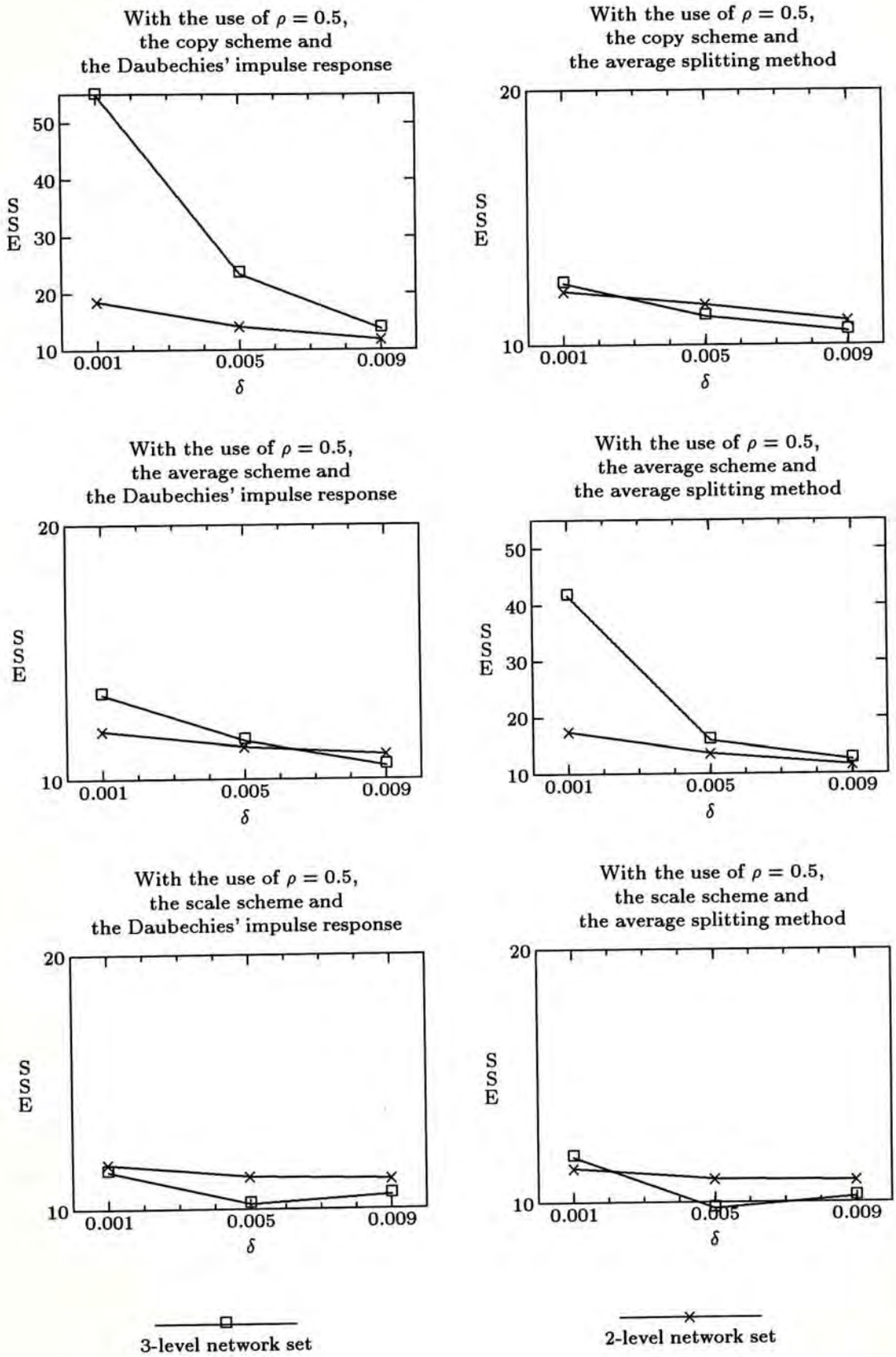



Figure 4.16: Comparisons of the recognition results with different values of $\delta$ for the numeric recognition problem.

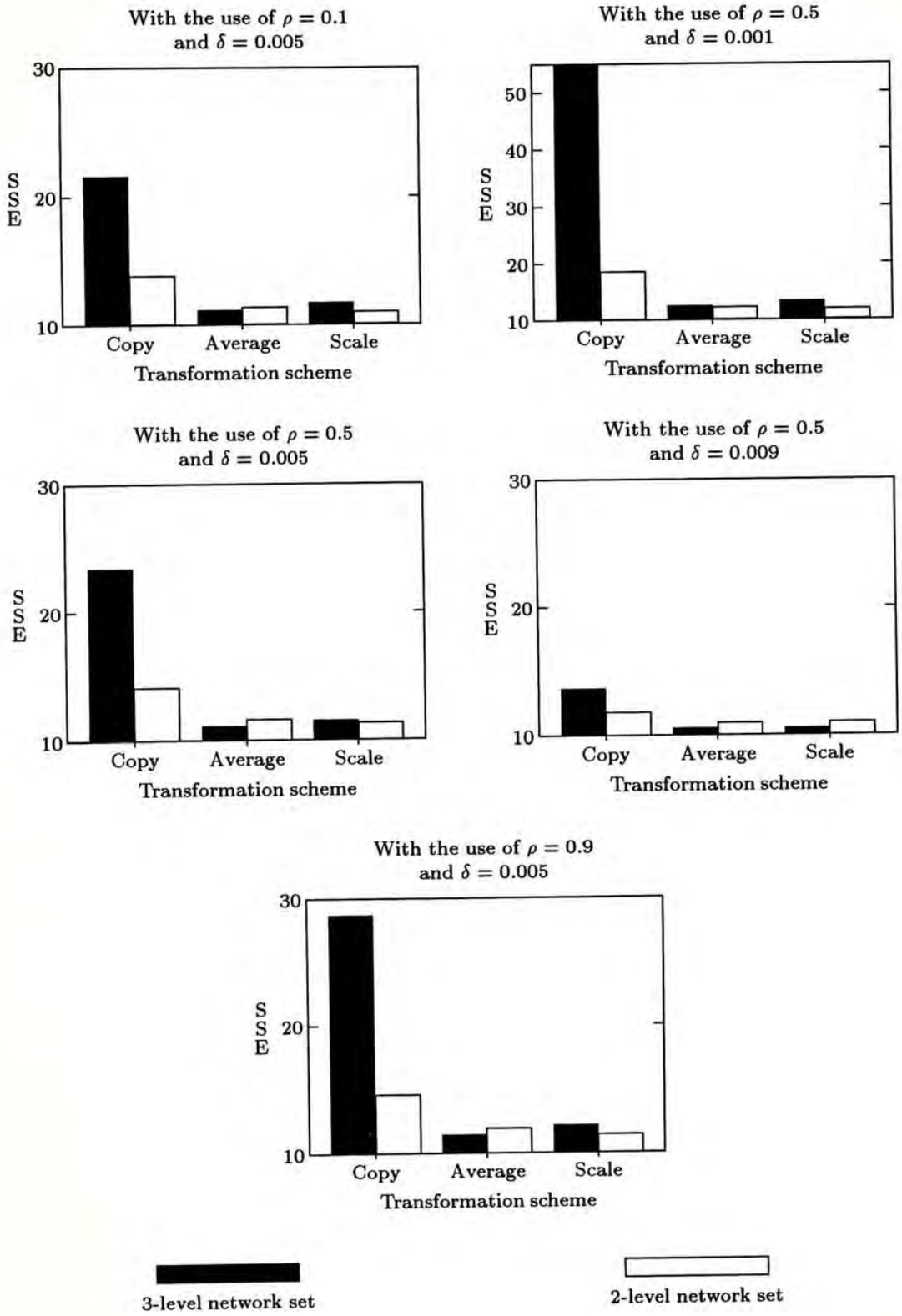



Figure 4.17: Comparisons of the recognition results with different transformation schemes for the numeric recognition problem of using the Daubechies' impulse response.

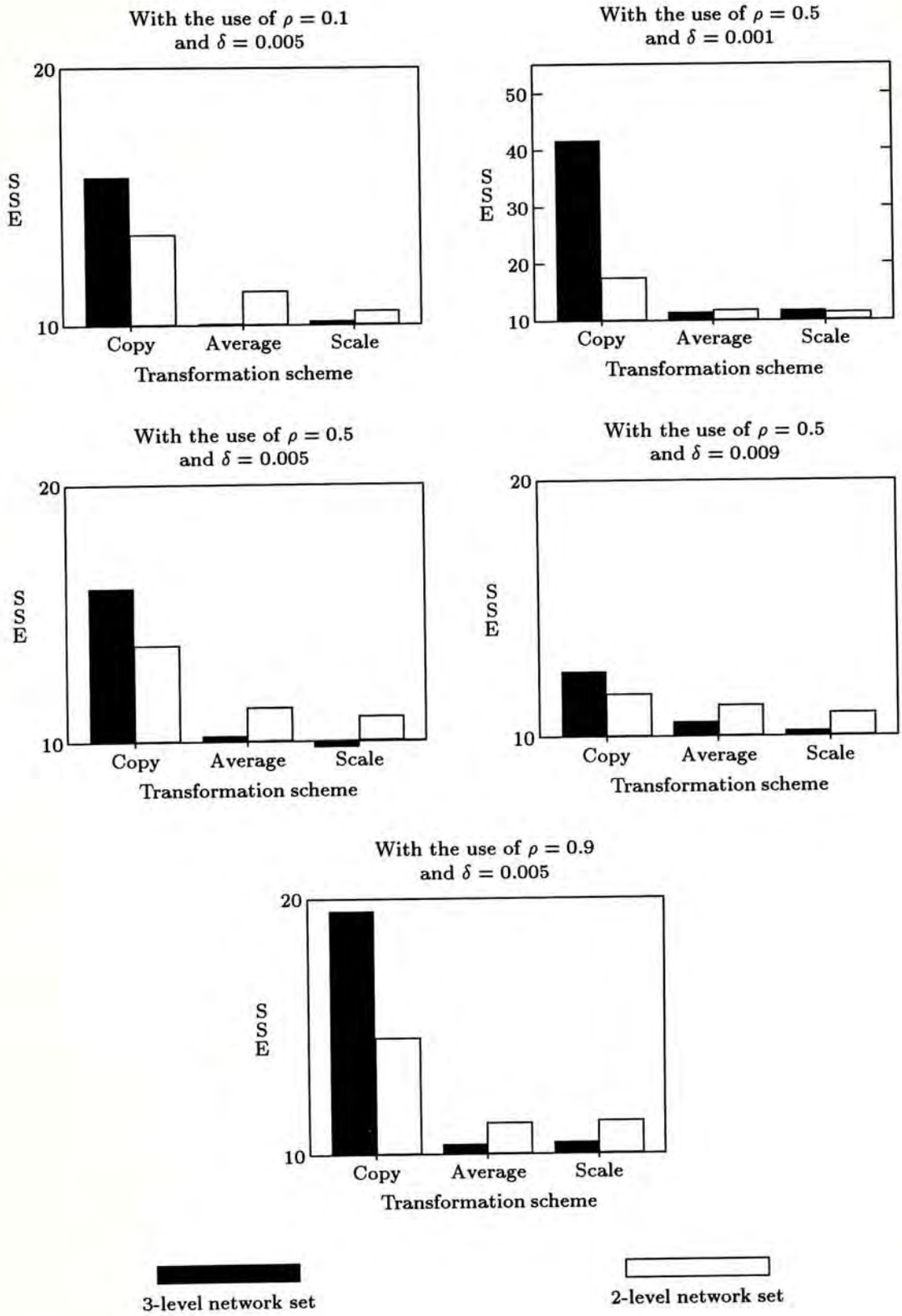



Figure 4.18: Comparisons of the recognition results with different transformation schemes for the numeric recognition problem of using the average splitting method.

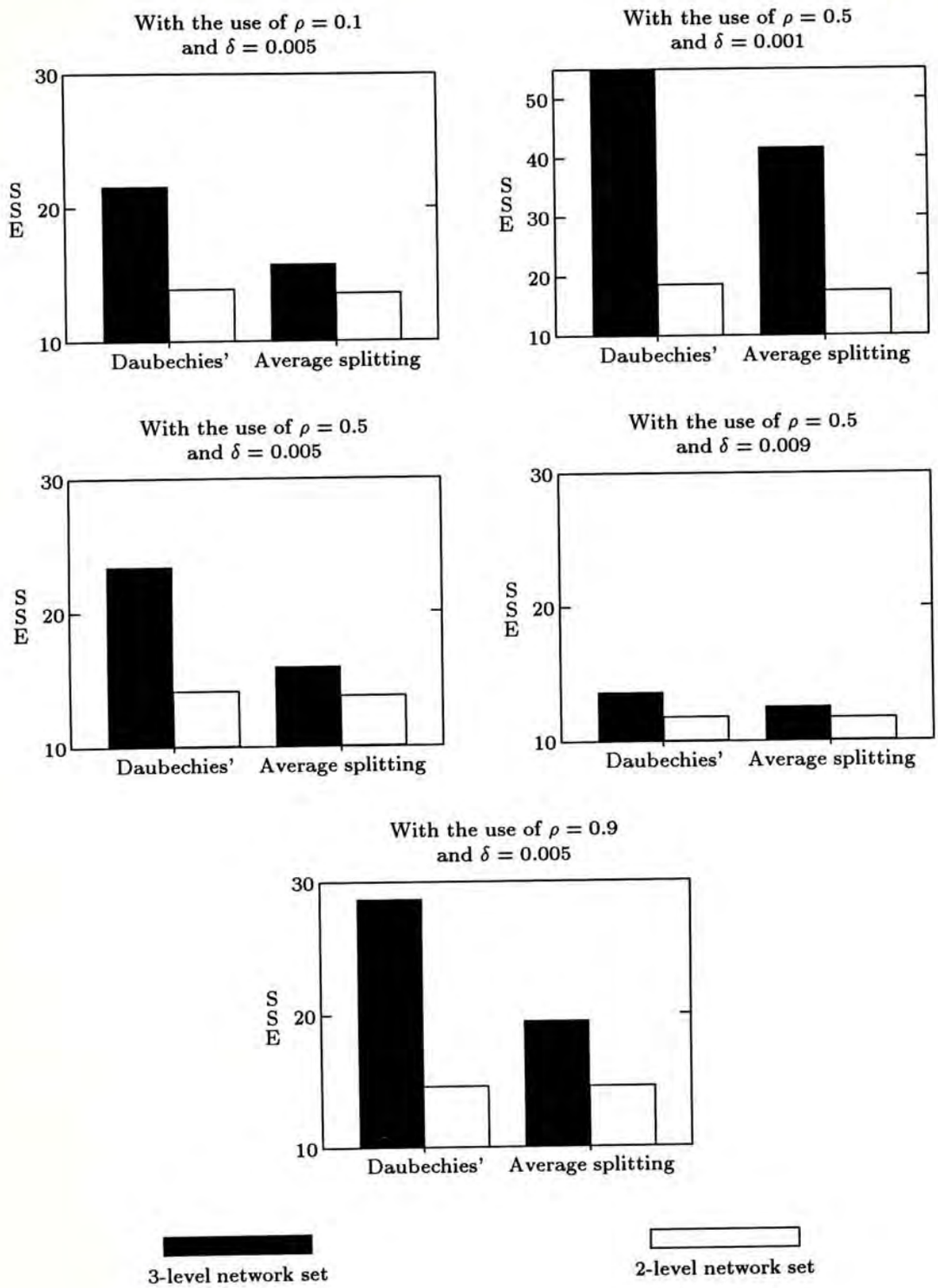



Figure 4.19: Comparisons of the recognition results with different sets of impulse response coefficients for the numeric recognition problem of using the copy scheme.

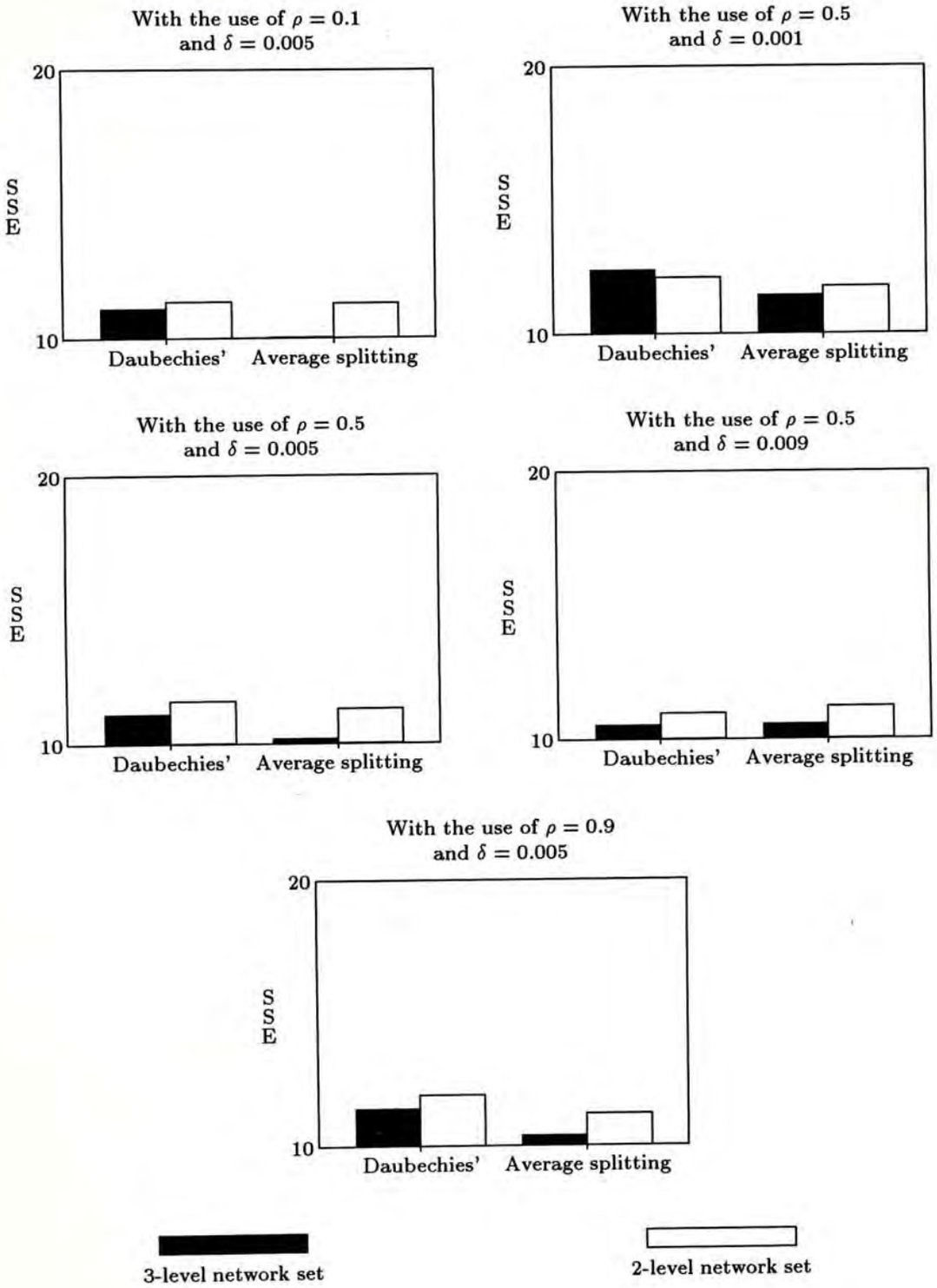



Figure 4.20: Comparisons of the recognition results with different sets of impulse response coefficients for the numeric recognition problem of using the average scheme.

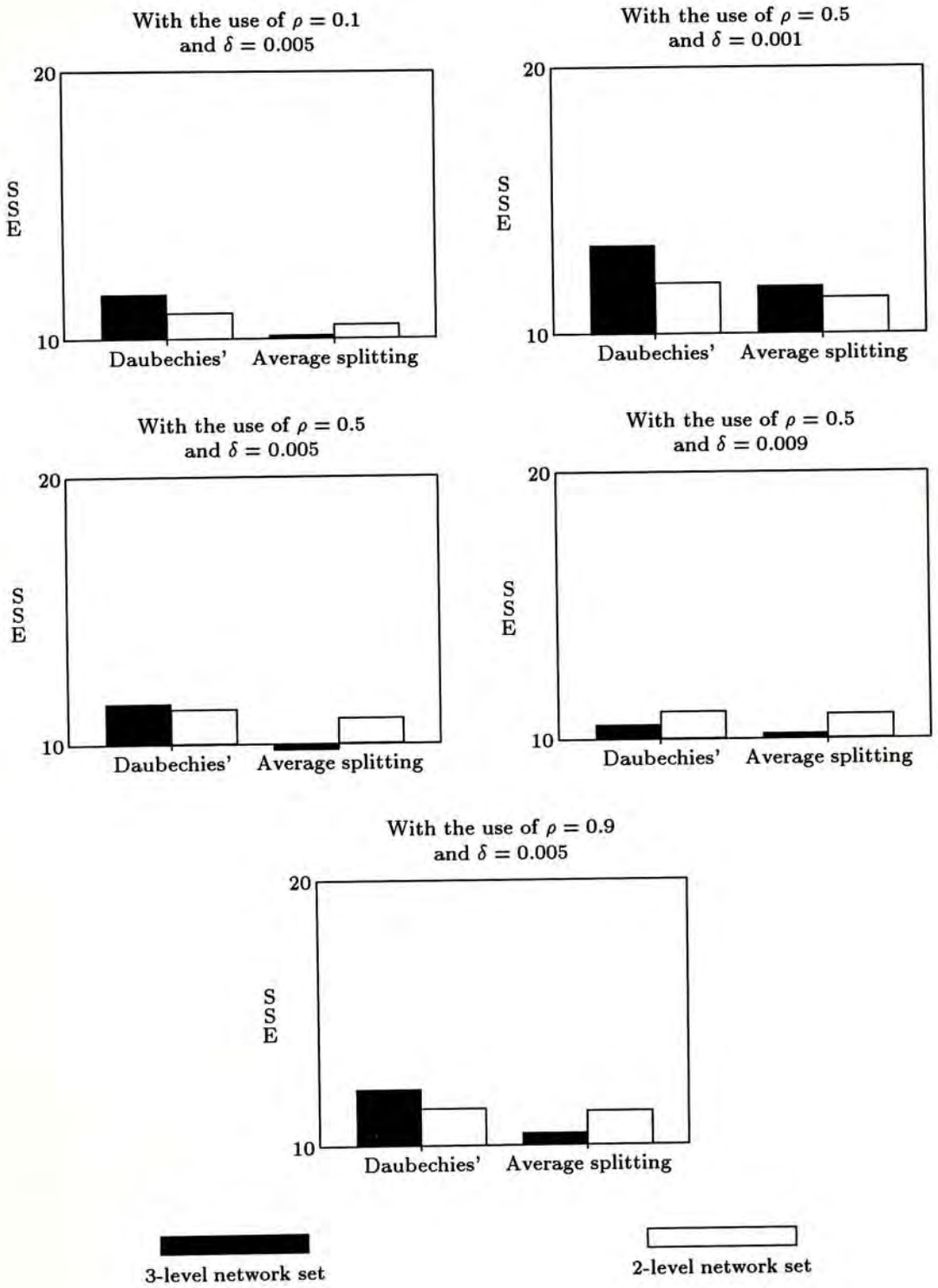



Figure 4.21: Comparisons of the recognition results with different sets of impulse response coefficients for the numeric recognition problem of using the scale scheme.

## 4.5 Discussions

Based on the experimental results shown in the Section 4.3 and 4.4, we discuss the properties of the multiresolution learning method and explain the above results in this section.

First of all, let us investigate why the training performance of a back-propagation network will be improved when the multiresolution learning method is used. With the use of the coarser level network in the learning method, the training examples can be learned in a coarser resolution. Since the architecture of a coarser level network is always simpler than the one of a finer level network, the coarser level networks often take less time in the training processes. Even the examples cannot be fully generalized in this level, the coarser level networks can actually reduce the overhead for the finer level networks in some extents. The function of the finer level networks is to refine the generalization rather than start it from the beginning.

Concerning the recognition ability of the networks in the numeric recognition problem, most of them performed well and could nearly recognize all testing samples correctly. For a standard back-propagation network without using the multiresolution learning method, the connection weights of it are usually initialized with small random numbers. During the training process, the connection weights are adjusted according to the generalized delta rule only. After the training process has been completed, the range of the connection weights will become large. In other words, the network will be more sensitive to some features of a training example than others and the fault tolerance of it to noisy data becomes low. On the other hand, with the use of the multiresolution learning method, the transformation of connection weights, especially the average and the scale schemes, acts as a constraint to limit the variance among connection weights and reduces the possibility of producing a network which is sensitive to some features of a training pattern. Thus, it increases the recognition performance of

the network.

However, the recognition performance of some training jobs of the numeric recognition problem, which adopted 3-level network set and the copy scheme, was not comparable with others, e.g., the recognition index is below 1.00. By monitoring one of the training processes of Job 4 in Experiment 1 (Table 4.9), it is discovered that the criteria $SSE(t) < \varepsilon$ was satisfied earlier than $W(t) < \delta$ for the network $B_{2^{-1}}$. Thus, the training process jumped to the next level network $B_{2^0}$. As it has been shown in Section 3.5, the state of each hidden neuron can remain unchanged after conducting the transformation of connection weights between the input and the hidden layers. Also, the copy scheme does not alter the connection weights between the hidden and the output layers but just copies them from a coarser network to a finer one. So $SSE(t)$ will not increase at this moment but continue to decrease. However, $B_{2^0}$ was the last network used to be trained in the process. The job then ended without any further training because $SSE(t) < \varepsilon$.

Table 4.9: The last 10 training epoches of one of the training processes for Job 4 in Experiment 1.

| Network Type | Time (sec) | $SSE(t)$ | $W(t)$ |
|---|---|---|---|
| $B_{2^{-1}}$ | 106.93 | 0.001024 | 0.002848 |
| $B_{2^{-1}}$ | 107.20 | 0.001021 | 0.002065 |
| $B_{2^{-1}}$ | 107.48 | 0.001017 | 0.001657 |
| $B_{2^{-1}}$ | 107.76 | 0.001014 | 0.001474 |
| $B_{2^{-1}}$ | 108.04 | 0.001010 | 0.001484 |
| $B_{2^{-1}}$ | 108.31 | 0.001007 | 0.001304 |
| $B_{2^{-1}}$ | 108.59 | 0.001004 | 0.001451 |
| $B_{2^{-1}}$ | 108.87 | 0.001000 | 0.001665 |
| $B_{2^{-1}}$ | 109.15 | 0.000997 | 0.001815 |
| $B_{2^0}$ | 110.23 | 0.000996 | 0.001259 |

From Table 4.9, it can be interpreted that the network $B_{2^{-1}}$ had learned sufficient information from the approximation $\{A^d_{2^{-1}}\vec{x}\}$. It then led to $SSE(t) < \varepsilon$.

Hence, the network $B_{2^0}$ could only know what $B_{2^{-1}}$ had learned but it did not have a chance to learn the difference of information between $\{A^d_{2^0}\vec{x}\}$ and $\{A^d_{2^{-1}}\vec{x}\}$, that is the detail approximation $\{D_{2^{-1}}\vec{x}\}$ described in Section 2.3.3. However, the samples used to test the network $B_{2^0}$ could contain those information $\{D_{2^{-1}}\vec{x}\}$, and $B_{2^0}$ would then misclassified them.

As it is shown in Figure 4.8 and 4.9, the convergence rate increases as $\delta$ decreases. It is quite easy to be understood that such improvement is expected. In this case, the low level network, say $B_{2^j}$ contributes more in the whole training process with a small value of $\delta$ and is allowed to learn the information of $\{A^d_{2^j}\vec{x}\}$ as much as possible. The main objective of the high level network $B_{2^{j+1}}$ is to learn the difference of information $\{D_{2^j}\vec{x}\}$ between $\{A^d_{2^{j+1}}\vec{x}\}$ and $\{A^d_{2^j}\vec{x}\}$. Usually, the computational cost for $B_{2^j}$ is smaller than the one of $B_{2^{j+1}}$.

# Chapter 5

# Conclusions

In this dissertation, we proposed a problem-independent learning method for the back-propagation networks to improve the convergence rate and the recognition ability of the networks. Traditionally, the back-propagation algorithm, which adopts the steepest descent technique, is slow to converge in a multilayer network. Such limitation prohibits the use of back-propagation networks on the large scale problems. Also, it assumes the individual input neuron acts independently from the other neurons. In fact, in some problems, for example, image recognition problems, use images as the grey level input to the network. The input neurons do have some correlations with their neighboring neurons. However, a multilayer perceptron has not taken this into account. Therefore, the learning method we proposed adopts the multiresolution signal decomposition techniques in order to alleviate the shortcomings of this kind of networks.

We have described in Chapter 3 the detail of the proposed multiresolution learning method. The learning method involves a group of back-propagation networks. Based on the multiresolution signal decomposition technique, we have defined the input vectors for the back-propagation networks under different resolutions (shown in (3.1) and (3.3)) and have derived the transformation of connection weights between layers from the coarser level network to the finer

one (shown in (3.10), (3.14), (3.15) and (3.16)). The sequence of the training processes to be carried out by the group of back-propagation networks is from the coarsest level network to the finest level network sequentially. We also define a term called the intermediate stopping criteria for terminating the training processes of these networks (shown in (3.6) and (3.7)).

Two different problems are simulated and they are the XOR problem and the numeric recognition problem (Chapter 4) to illustrate the performance of this learning method. Experimental results showed that the proposed method improve the training speed and the recognition ability of the back-propagation networks significantly. Since the architecture of a coarser level network is always simpler than the one of a finer level network, the coarser level networks often take less time in the training processes. Even the examples cannot be fully generalized in such level, the coarser level networks can actually reduce the overhead for the finer level networks in some extents. The function of the finer level networks is to refine the generalization. Also, the convergence rate increases as the intermediate stopping criteria $\delta$ decreases. In this case, the coarser level network contributes more in the whole training process with a small value of $\delta$ and is allowed to learn the input vectors as much as possible. With the use of the multiresolution learning method, the transformation of connection weights, especially the average and the scale schemes, acts as a constraint to limit the variance among connection weights and reduces the possibility of producing a network which is sensitive to some features of a training pattern. Thus, it increases the recognition performance of the network.

# Appendix A

# Proof of Equation (4.9)

By (2.13) and (4.6), we can derive $\tilde{h}(n)$ as

$$\tilde{h}(n) = \begin{cases} \frac{1-\sqrt{3}}{8} & \text{if } n = -2 \\ \frac{3-\sqrt{3}}{8} & \text{if } n = -1 \\ \frac{3+\sqrt{3}}{8} & \text{if } n = 0 \\ \frac{1+\sqrt{3}}{8} & \text{if } n = 1 \\ 0 & \text{otherwise} \end{cases} . \tag{A.1}$$

In other words, for $-2 \leq n \leq 1$, $\tilde{h}(n)$ is a non-zero number; otherwise, it is equal to 0. Let us consider (3.10) with 3 different cases.

- Case 1: $o = 1$

$$\begin{aligned} w'_{1q} &= \sum_{p=1}^{2^j N} \tilde{h}(2p-1) w_{pq} \\ &= \tilde{h}(1) w_{1q} + \tilde{h}(3) w_{2q} + \cdots + \tilde{h}(2^{j+1}N - 1) w_{2^j N, q} \\ &= \tilde{h}(1) w_{1q} \end{aligned} \tag{A.2}$$

- Case 2: $o = 2^{j+1}N$

$$\begin{aligned} w'_{2^{j+1}N,q} &= \sum_{p=1}^{2^j N} \tilde{h}(2p - 2^{j+1}N) w_{pq} \\ &= \tilde{h}(2 - 2^{j+1}N) w_{1q} + \cdots + \tilde{h}(-2) w_{2^j N-1,q} + \tilde{h}(0) w_{2^j N,q} \\ &= \tilde{h}(-2) w_{2^j N-1,q} + \tilde{h}(0) w_{2^j N,q} \end{aligned} \tag{A.3}$$

- Case 3: $2 \le o \le 2^{j+1}N - 1$

The first term $\sum \tilde{h}(2p-o)w_{pq}$ can be rewritten as

$$\sum_{p=1}^{2^jN} \tilde{h}(2p-o)w_{pq}$$

$$= \begin{cases} \tilde{h}(0)w_{1q} & \text{if } o = 2 \\ \tilde{h}(-1)w_{\frac{o-1}{2},q} + \tilde{h}(1)w_{\frac{o+1}{2},q} & \text{if } o = 3, 5 \cdots, 2^{j+1}N - 1 \\ \tilde{h}(-2)w_{\frac{o-2}{2},q} + \tilde{h}(0)w_{\frac{o}{2},q} & \text{if } o = 4, 6, \cdots, 2^{j+1}N - 2 \end{cases} . \quad \text{(A.4)}$$

The second term $\sum \tilde{h}(2p+o-2^{j+2}N)w_{pq}$ can be rewritten as

$$\sum_{p=1}^{2^jN} \tilde{h}(2p+o-2^{j+2}N)w_{pq} = \begin{cases} 0 & \text{if } 2 \le o \le 2^{j+1}N - 3 \\ \tilde{h}(-2)w_{2^jN,q} & \text{if } o = 2^{j+1}N - 2 \\ \tilde{h}(-1)w_{2^jN,q} & \text{if } o = 2^{j+1}N - 1 \end{cases} . \quad \text{(A.5)}$$

The last term $\sum \tilde{h}(2p+o-2)w_{pq}$ can be rewritten as

$$\sum_{p=1}^{2^jN} \tilde{h}(2p+o-2)w_{pq} = 0. \quad \text{(A.6)}$$

By putting (A.2), (A.3), (A.4), (A.5), and (A.6) all together, we then get

$$w'_{oq} = \begin{cases} \tilde{h}(1)w_{1q} & \text{if } o = 1 \\ \tilde{h}(0)w_{1q} & \text{if } o = 2 \\ \tilde{h}(-1)w_{\frac{o-1}{2},q} + \tilde{h}(1)w_{\frac{o+1}{2},q} & \text{if } o = 3, 5 \cdots, 2^{j+1}N - 3 \\ \tilde{h}(-2)w_{\frac{o-2}{2},q} + \tilde{h}(0)w_{\frac{o}{2},q} & \text{if } o = 4, 6, \cdots, 2^{j+1}N - 4 \\ \tilde{h}(-2)w_{2^jN-2,q} + \tilde{h}(0)w_{2^jN-1,q} & \text{if } o = 2^{j+1}N - 2 \\ \quad + \tilde{h}(-2)w_{2^jN,q} & \\ \tilde{h}(-1)w_{2^jN-1,q} + \tilde{h}(1)w_{2^jN,q} & \text{if } o = 2^{j+1}N - 1 \\ \quad + \tilde{h}(-1)w_{2^jN,q} & \\ \tilde{h}(-2)w_{2^jN-1,q} + \tilde{h}(0)w_{2^jN,q} & \text{if } o = 2^{j+1}N \end{cases} . \quad \text{(A.7)}$$

# Appendix B

# Proof of Equation (4.11)

By (2.13) and (4.7), we can derive $\tilde{h}(n)$ as

$$\tilde{h}(n) = \begin{cases} 0.5 & \text{if } n = -1 \\ 0.5 & \text{if } n = 0 \\ 0 & \text{otherwise} \end{cases} . \tag{B.1}$$

In other words, for $n = -1$ or $n = 0$, $\tilde{h}(n)$ is a non-zero number; otherwise, it is equal to 0. Let us consider (3.10) with 3 different cases.

- Case 1: $o = 1$

$$\begin{aligned} w'_{1q} &= \sum_{p=1}^{2^j N} \tilde{h}(2p-1) w_{pq} \\ &= \tilde{h}(1) w_{1q} + \tilde{h}(3) w_{2q} + \cdots + \tilde{h}(2^{j+1}N - 1) w_{2^j N, q} \\ &= 0 \end{aligned} \tag{B.2}$$

- Case 2: $o = 2^{j+1}N$

$$\begin{aligned} w'_{2^{j+1}N,q} &= \sum_{p=1}^{2^j N} \tilde{h}(2p - 2^{j+1}N) w_{pq} \\ &= \tilde{h}(2 - 2^{j+1}N) w_{1q} + \cdots + \tilde{h}(-2) w_{2^j N - 1, q} + \tilde{h}(0) w_{2^j N, q} \\ &= \tilde{h}(0) w_{2^j N, q} \end{aligned} \tag{B.3}$$

- Case 3: $2 \leq o \leq 2^{j+1}N - 1$

The first term $\sum \tilde{h}(2p-o)w_{pq}$ can be rewritten as

$$\sum_{p=1}^{2^jN} \tilde{h}(2p-o)w_{pq} = \begin{cases} \tilde{h}(0)w_{\frac{o}{2},q} & \text{if } o = 2, 6, \cdots, 2^{j+1}N - 2 \\ \tilde{h}(-1)w_{\frac{o-1}{2},q} & \text{if } o = 3, 5 \cdots, 2^{j+1}N - 1 \end{cases}. \quad \text{(B.4)}$$

The second term $\sum \tilde{h}(2p+o-2^{j+2}N)w_{pq}$ can be rewritten as

$$\sum_{p=1}^{2^jN} \tilde{h}(2p+o-2^{j+2}N)w_{pq} = \begin{cases} 0 & \text{if } 2 \leq o \leq 2^{j+1}N - 2 \\ \tilde{h}(-1)w_{2^jN,q} & \text{if } o = 2^{j+1}N - 1 \end{cases}. \quad \text{(B.5)}$$

The last term $\sum \tilde{h}(2p+o-2)w_{pq}$ can be rewritten as

$$\sum_{p=1}^{2^jN} \tilde{h}(2p+o-2)w_{pq} = 0. \quad \text{(B.6)}$$

By putting (B.2), (B.3), (B.4), (B.5), and (B.6) all together, we then get

$$w'_{oq} = \begin{cases} 0 & \text{if } o = 1 \\ \tilde{h}(0)w_{\frac{o}{2},q} & \text{if } o = 2, 6, \cdots, 2^{j+1}N \\ \tilde{h}(-1)w_{\frac{o-1}{2},q} & \text{if } o = 3, 5 \cdots, 2^{j+1}N - 3 \\ \tilde{h}(-1)w_{2^jN-1,q} + \tilde{h}(-1)w_{2^jN,q} & \text{if } o = 2^{j+1}N - 1 \end{cases}. \quad \text{(B.7)}$$

# Bibliography


[Baaz90] Baaziz, N. and Labit, C. "Laplacian pyramid versus wavelet decomposition for image sequence coding". In *Proceedings of ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 1965–1968, Albuquerque, New Mexico, USA, 1990.

[Burt83] Burt, P. J. and Adelson, E. H. "The Laplacian pyramid as a compact image code". *IEEE Transactions on Communications*, 31(4):532–540, 1983.

[Burt84] Burt, P. J. "The pyramid as a structure for efficient computation". In Rosenfeld, A., editor, *Multiresolution Image Processing and Analysis*, pages 6–35. Springer-Verlag, 1984.

[Chan94a] Chan, L. W. and Chan, W. C. "A multiresolution learning method for back-propagation networks". In *Proceedings of World Congress on Neural Networks*, volume 3, pages 46–51, San Diego, California, USA, 1994.

[Chan94b] Chan, W. C. and Chan, L. W. "Transformation of back-propagation networks in multiresolution learning". In *Proceedings of IEEE International Conference on Neural Networks*, volume 1, pages 290–294, Orlando, Florida, USA, 1994.

[Croc81] Crochiere, R. E. and Rabiner, L. R. "Interpolation and decimation of digital signals – a tutorial review". *Proceedings of the IEEE*, 69(3):300–331, 1981.

[Crow84] Crowley, J. L. "A multiresolution representation for shape". In Rosenfeld, A., editor, *Multiresolution Image Processing and Analysis*, pages 169–189. Springer-Verlag, 1984.

[Daub88] Daubechies, I. "Orthonormal bases of compactly supported wavelets". *Communications on Pure and Applied Mathematics*, 41:909–996, 1988.

[Evan91] Evans, M. R., Ellacott, S. W., and Hand, C. C. "A multi-resolution neural network classifier for machine vision". In *IEEE International Joint Conference on Neural Networks*, pages 2594–2599, Singapore, 1991.

[Hert91] Hertz, J., Krogh, A., and Palmer, R. G. *Introduction to the Theory of Neural Computation*, chapter 6, pages 115–162. Addison-Wesley, 1991.

[Hush88] Hush, D. R. and Salas, J. M. "Improving the learning rate of back-propagation with the gradient reuse algorithm". In *Proceedings of the IEEE International Conference on Neural Networks*, volume 1, pages 441–448, 1988.

[Hush93] Hush, D. R. and Horne, B. G. "Progress in supervised neural networks". *IEEE Signal Processing Magazine*, pages 8–29, January 1993.

[Jaco88] Jacobs, R. A. "Increased rates of convergence through learning rate adaptation". *Neural Networks*, 1(4):295–308, 1988.

[Koen84] Koenderink, J. "The structure of images". *Biological Cybernetics*, 50:363–370, July 1984.

[Kung93] Kung, S. Y. *Digital Neural Networks*, chapter 5, pages 145–201. Prentice Hall, 1993.

[Levi85] Levine, M. D. *Vision in Man and Machine*, chapter 3, pages 59–99. McGraw-Hill, 1985.

[Mall89a] Mallat, S. G. "A theory for multiresolution signal decomposition: the wavelet representation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[Mall89b] Mallat, S. G. "Multifrequency channel decompositions of images and wavelet models". *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):2091–2110, 1989.

[Mall89c] Mallat, S. G. "Multiresolution approximations and wavelet orthonormal bases of $L^2(R)$". *Transactions of the American Mathematical Society*, 315(1):69–87, 1989.

[Marr82] Marr, D. *Vision.* W.H. Freeman and Company, 1982.

[Rose84a] Rosenfeld, A., editor. *Multiresolution Image Processing and Analysis.* Springer-Verlag, 1984.

[Rose84b] Rosenfeld, A. "Some useful properties of pyramids". In Rosenfeld, A., editor, *Multiresolution Image Processing and Analysis*, pages 2–5. Springer-Verlag, 1984.

[Rume86] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. "Learning internal representations by error propagation". In Rumelhart, D. E. and

McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1: Foundations, pages 318–362. Bradford Books/MIT Press, 1986.

[Sabo93] Sabourin, M. and Mitiche, A. "Modeling and classification of shape using a Kohonen associative memory with selective multiresolution". *Neural Networks*, 6(2):275–283, 1993.

[Stra89] Strang, G. "Wavelets and dilation equations: a brief introduction". *SIAM Review*, 31(4):614–627, 1989.

[Szel90] Szeliski, R. "Fast surface interpolation using hierarchical basis functions". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):513–528, 1990.

[Tani75] Tanimoto, S. and Pavlidis, T. "A hierarchical data structure for picture processing". *Computer Graphics and Image Processing*, 4:104–119, 1975.

[Unse89] Unser, M. and Eden, M. "Multiresolution feature extraction and selection for texture segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):717–728, 1989.

[Watr87] Watrous, R. L. "Learning algorithms for connectionist networks: applied gradient methods of nonlinear optimization". In *Proceedings of the IEEE First International Conference on Neural Networks*, volume 2, pages 619–628, 1987.

[Wess92] Wessels, L. F. A. and Barnard, E. "Avoiding false local minima by proper initialization of connections". *IEEE Transactions on Neural Networks*, 3(6):899–905, 1992.

[Yhan90] Yhann, S. R. and Young, T. Y. "A multiresolution approach to texture segmentation using neural networks". In *Proceedings of International Conference on Pattern Recognition*, volume 1, pages 513–517, Atlantic City, NJ, 1990.