6 Biomedical signal compression

Chapter: 6
Biomedical signal compression

6.1 Introduction

Need for ECG signal compression exists in many transmitting and storage applications. Transmitting the ECG signal through telephone lines, for example, may save a crucial time and unnecessary difficulties in emergency cases. Effective storage is required of large quantities of ECG information in the intensive coronary care unit, or in long-term (24-28 hours) wearable monitoring tasks [1].

Removing redundancy performs the compression of the signal. The redundancy exhibits itself in terms of statistical dependence between adjacent samples and the non-uniformity of the amplitude probability of the quantized signals [2]. Linear correlation between neighboring samples may be removed, for example by various delta modulation methods or by linear prediction methods [3], while the non-uniform amplitude probability may be handled by entropy coding [3].

Biomedical signal compression methods can be divided into three functional groups:

- **Direct methods:** where the samples of the signal are directly handled to provide the compression.
- **Transformation methods:** where the original samples are subjected to a (Linear) transformation and the compression is performed in the new domain.
- **Parameter extraction method:** where a preprocessor is employed to extract some features that are later used to reconstruct the signal.

Several recent ECG data compression strategies have been proposed which exploit prior information on the locations in time of the heart bit subtraction with residual differencing, long term prediction with entropy coding, cycle pool based compression and vector quantization. Most of these methods utilize pseudo periodic behavior of ECG signals. Vector Quantization is extensively used in data compression systems [4]-[6].

The distortion measures used for evaluating the reconstructed signal includes percentage RMS difference (PRD) and Weighted Diagnostic Distortion (WDD). The WDD is based on comparing the PQRST complex features of the two ECG signals, the original ECG signal and the reconstructed one. The WDD measures the relative

preservation of the diagnostic information in the reconstructed signal: the location, duration, amplitudes, and shapes of the waves and complexes that exist in every beat (PQRST complex). **Figure 6.1** shows some of the diagnostic features:

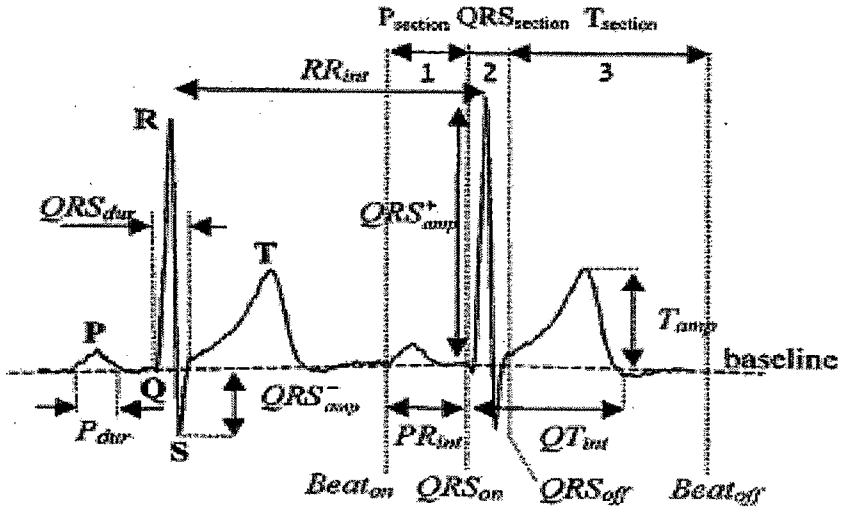


Figure 6.1 Diagnostic Features of ECG wave

6.1.1 Definitions of terms used in the waveform

The **P wave** is caused by atrial depolarization. The P wave is usually smooth and positive. The P wave duration is normally less than 0.12 sec.

The **PR interval** is the portion of the EKG wave from the beginning of the P wave (onset of atrial depolarization) to the beginning of the QRS complex (onset of ventricular depolarization). It is normally 0.12 - 0.20 seconds.

The **PR segment** is the portion on the EKG wave from the end of the P wave to the beginning of the QRS complex. The PR segment corresponds to the time between the end of atrial depolarization to the onset of ventricular depolarization. It is an isoelectric segment, during which the impulse travels from the AV node through the conducting tissue (bundle branches, and Purkinje fibers) towards the ventricles.

The **QRS complex** represents the time it takes for depolarization of the ventricles due to ventricular depolarization. The normal QRS interval range is from 0.04 sec - 0.12 sec measured from the first deflection to the end of the QRS complex.

The **ST Segment** represents the period of ventricular muscle contraction before repolarization. The ST segment is normally isoelectric (no electrical activity is recorded). However, the ventricles are contracting.

The **QT interval** begins at the onset of the QRS complex and to the end of the T wave. It represents the time of ventricular depolarization until venticular repolarization.

The **T wave** is due to ventricular repolarization The wave is normally rounded and positive.
In ECG signal compression algorithms the goal is to achieve a minimum information rate, while retaining the relevant diagnostic information in the reconstructed signal.

Chapter provides a comprehensive study of the work done by the researchers for the signal compression. The ANN models for the signal compression are developed and their performance is compared with other conventional statistical, Vector quantization as well as transformed based techniques

6.2 Classical methods

This subsection gives brief overview of traditional methods for compression.

6.2.1 Adaptive Correlation Technique

A time domain transformation in real time may be used for the data compression of a highly correlated signal such as the ECG.

Several schemes have been proposed [7] for data compression in the past. Nearly all of them employ heuristic approaches to obtain data compression. The methods based on polynomial approximation calls for a predetermined subdivision of the signal into intervals that make optimal use of the polynomial base function. Transform methods such as the DCT or the Haar transform, through exploit a priori information such as periodicity of the signal, are not suitable for non periodic signals and signals received at high data rates. As ECG signals are quasi-periodic and have high inter sample correlation a method of data compression using inter sample correlations of the ECG signal are highly efficient.

Analysis in [8] is built on choosing an initial frame of samples in one beat, in which the signal is assumed to possess stationary characteristics. A frame by frame approach is then restored to for achieving the goal. The idea of constructing a linearly correlated signal of lower range has been introduced and the least square solutions have been arrived at for constructing the transform domain data in a recursive manner. It turns out that a one to one relationship is obtained between the original data domain sequence and the transform domain sequence.

6.2.2 Coding Techniques

Average bit subtraction and residual differencing [9] Hoffman coding is applied to the difference signal. On detection of average beat it is aligned and subtracted from the signal. The detected beat is then used to update the average bit estimate. After beat subtraction the residual signal is first differenced. The difference signal is Huffman encoded and stored along with a record of the best locations. Sample rate and quantization level are chosen to yield maximum fidelity at data rates of approximately 200 bits per second (bps) per channel. **Figure 6.2** depicts compression algorithm.

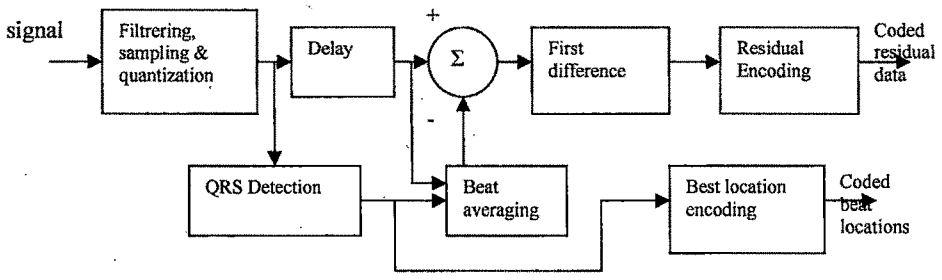


Figure 6.2 Compression Algorithm

Figure 6.3 depicts block diagram of the decompression operations. The residual signal is decoded and first summed to undo the first differencing. The beat locations are also decoded, and average beats are added to the residual signal as indicated by the stored beat locations. As beats are reconstructed they are used to update the average beat estimate. As long as the average beat is initially equal to the average beat used in compression, the quantized ECG will be correctly reproduced.

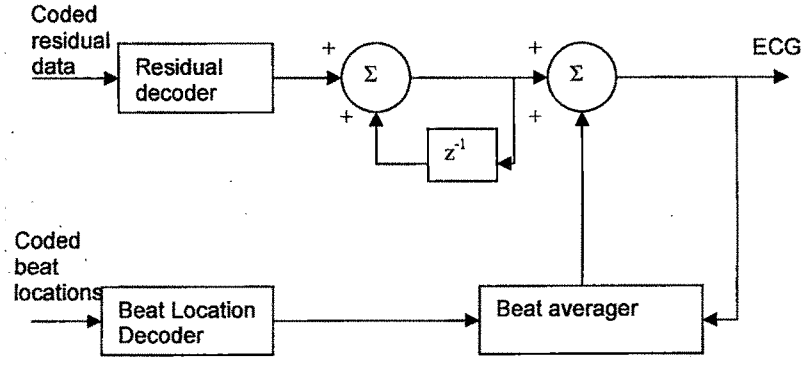


Figure 6.3 Decompression algorithm

[2] is a parameter extraction method based on linear prediction coding (LPC) based on the sub auto regression (SAR) model. In the conventional LPC method, the n^{th}

sample of a signal is predicted by its p past samples. The error between the predicted and actual samples is sent or stored instead of the sample itself.

Since the variance (dynamic range) of the error is less than that of the original signal, it may be represented by fewer bits per sample. It has been suggested in [10], to use a long term prediction (LTP) method where the prediction of the nth sample is made using samples of past beats. It is worthwhile noting that the beat correlation was recently used in a different type of algorithm [11]. The idea of long term correlation is also used in speech analysis for pitch detection. The LTP residual (error) signal is quantized and further compressed using the Huffman code [12]. The residual quantization size is an important parameter of the compression algorithm This lowers the CR (but still maintains higher CR than most CR's reported in the literature). With very low reconstruction errors.

For multi channel ECG data [13] the samples at the same instant may be extracted from all the N channels and treated as a vector of dimension N. A codebook consisting of such vectors is formed using the Linde-Buzo-Gray algorithm [1]. Coding of a vector is done by transmitting the index of that codebook entry, to which the incoming vector will map, with the mapping based on a minimum distortion criterion. The reconstructed output of each channel is extracted from these coded vectors without appreciable distortion. Compression ratios of 18:1 and 16.4:1 have been achieved at an acceptable signal quality. This technique was evaluated on the 15 channel data from 7 subjects independently. To evaluate the performance of technique, four different measures of error are calculated. The performance of the proposed method improves as the number of channel is increases.

An efficient data compression may be achieved only with lossy compression techniques (which allow reconstruction error.) In ECG signal compression algorithms the goal is to achieve a minimum information rate, while retaining the relevant diagnostic information in the reconstructed signal.Zingle, Arnon and Katz [1] developed a new ECG compression algorithm called analysis by synthesis ECG compressor (ASEC) based on analysis by synthesis coding. It consists of a beat codebook, long and short term predictors, and (an adaptive residual quantizer. The compression algorithm uses a defined distortion measure in order to efficiently encode every heartbeat, with minimum bit rate, while maintaining a predetermined distortion level.

The ECG signal may be considered a quasi periodic signal. The main redundancies in the ECG signal exist in the form of correlation between adjacent or past beats (inter beat correlation) and correlation between adjacent samples (intra beat correlation) [2]. The frequent existence of abnormal beats in some pathological cases suggests using a beat codebook. The codebook is used to store "typical" past beats. The intrabeat correlation suggests using a short-term predictor, STP. With LTP, STP and a beat codebook, a predicted beat can be estimated, and a residual signal, which has lower variance, can be calculated. The analysis by synthesis model is used to efficiently code the residual signal, with minimum bit rate, while maintaining a predetermined error.

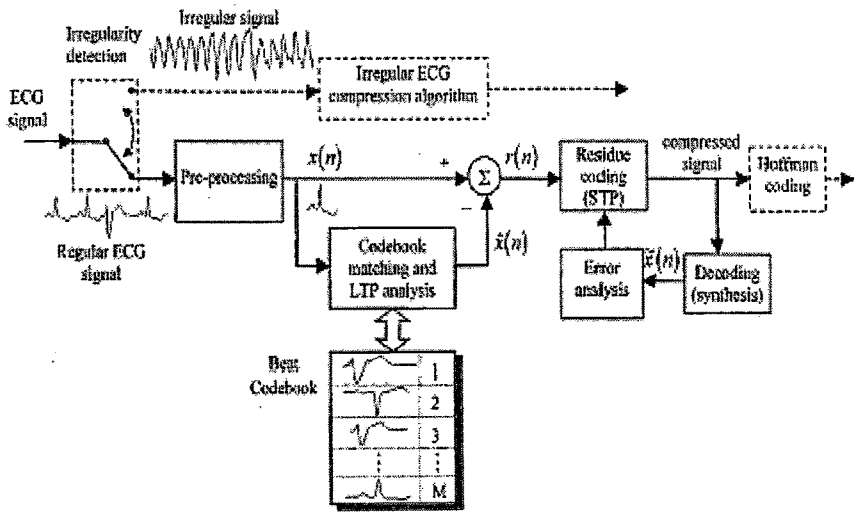


Figure 6.4 General Scheme of ASEC

Figure 6.4 shows the general scheme of the ASEC. The ECG signal is first classified as regular or irregular. However irregular signals, in general are less probable than the regular PQRST signal. The algorithm of irregular signal detection and compression is described in [14].

The ASEC algorithm consists of three main subsystems:

1) preprocessing,

2) coding: codebook matching and long-term prediction (LTP), residue coding, error analysis, and

3) decoding.

The ECG signal is processed beat by beat. The incoming beat is segmented into three time regions (**Figure 6.4**), which are then coded separately. The beat is matched with the Codebook to find the best matching stored beat ("codeword"). LTP coding is performed using the chosen codeword to produce the LTP estimated (predicted) signal. The difference between the original signal and the LTP estimated signal is defined as the residue. The residue undergoes STP coding and adaptive quantization to produce the coded signal. Prior to transmission, the signal to be transmitted is decoded, and the quality of the reconstructed signal is tested (by means of WDD or PRD measure). The residual signal is re-encoded with higher bit rate till the quality of the reconstructed signal is satisfied (below a predetermined distortion threshold).

The decoding system is shown in **Figure 6.5**. This system exists at the transmission side as well as the receiver side. The decoding system consists of bit decoding, beat codebook, and LTP decoding (which consists of an LTP coefficients codebook), which are identical to these elements in the encoder. For every heartbeat (complex), the decoder decodes the bits, and estimates the predicted signal with the LTP and beat codebook.

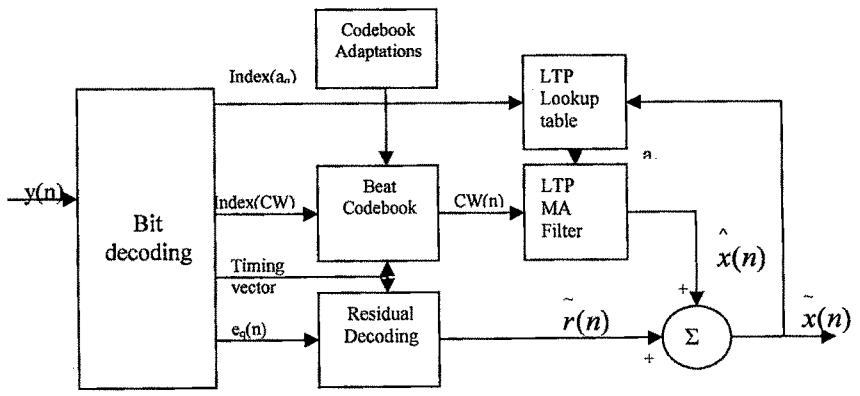


Figure 6.5 Decoding System

The residual signal is reconstructed by residual decoding as shown in **Figure 6.6**. The predicted signal is added to the reconstructed residual and the reconstructed signal is calculated. The reconstructed signal is also used for beat codebook adaptation.

Scan along algorithms may be used to produce compressed data in real time.

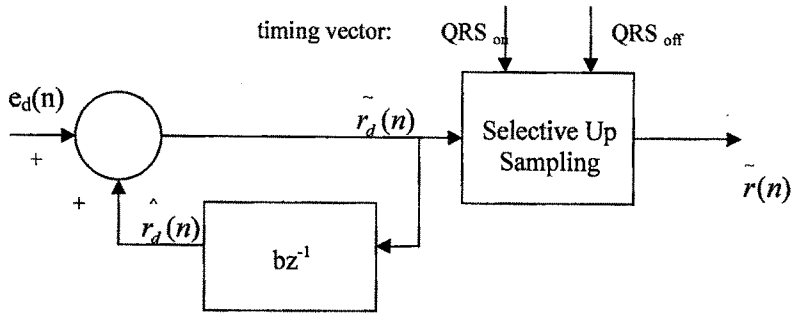


Figure 6.6 Residual Decoder

Ishijima, Shin, Hostetter and Sklansky [15] used polygonal approximation **(SAPA)** for digitized curve to restrict maximum error within a specified limit. A compression ratio of 10/11 on 5 and 3 beat ECG respectively. [66] uses m-SAPA algorithm which exploits inter channel correlation that exists among standard ECG leads for a Two channel data base and 8 channel ECG recording to achieve a compression ratio of 13.6 with PRD of less than error limit of 15%.

6.3 Transformed Methods

In transform methods the original samples are transformed to another domain in the hope of achieving better compression performance, examples include Fourier descriptors, Walsh transform, Karhunen – Loeve transform (KLT), discrete cosine (DCT) [16]-[20] and the recently developed wavelet transform [21]-[24]. In most cases, direct methods are superior to transform methods with respect to system complexity and the error control mechanism; however, transform methods usually achieve higher compression ratios and are insensitive to the noise contained in original ECG signals [25].

ECG signals are usually non stationary and if the quality of a reconstructed ECG signal is not guaranteed so the compression process itself will become less useful. In the case of the direct methods, such as the polygonal approximation methods, the error limit for a reproduced ECG signal is easily controlled by adjusting a user specified error threshold.

The section describes use of methods using orthogonal, non-orthogonal and Wavelet transforms. In [17] the results of an initial study to determine the feasibility of securing electrocardiograph (ECG) data compression via orthogonal transforms such as Haar transform and discrete cosine transforms are discussed.

6.3.1 Wavelet Transforms

The wavelet transform has good localization in both frequency and time domains, having fine frequency resolution and coarse time resolution at lower frequency, and coarse frequency resolution and fine time resolution at higher frequency. Since this matches the characteristic of most signals, it makes the wavelet transform suitable for time-frequency analysis. In data compression, the wavelet transform is used to exploit the redundancy in the signal. After the original signal is transformed into the wavelet domain, many coefficients are so small that no significant information is lost in the signal reconstructed by setting these coefficients to zero.

The technique in [26] is based on a new class of non-orthogonal discrete wavelet transform (DWT). The performance of ECG compression algorithm is measured by its ability to minimize distortion while retaining all clinically significant features of the signal. The percent root-mean square difference (PRD) is used as an accepted standard for measuring the signal distortion. However, there is no standard for measuring the clinically significant features retained after signal reconstruction. The coefficients of DWT are calculated such that the square of the difference between the original signal and the reconstructed one is minimum in least mean square sense. The resulting transforms deal with signals of arbitrary lengths; that means the signal length is not restricted to be multiple of power 2. The results in [27] show that, independent of signal length, the decomposition of the signal up to the fourth level is sufficient for getting minimum PRD.

[28] presents a method based on orthonormal wavelet transform and an adaptive quantization strategy, by which a predetermined percent root mean square difference (PRD) can be guaranteed with high compression ratio and low implementation complexity. Segment of the ECG signal varies with the complex pattern of the segment and one cannot find a predetermined optimal quantization bin size by which the quality of the reconstructed ECG signal is guaranteed at every segment.

[29] describes a discrete orthonormal wavelet transform (DOWT)-based ECG coding system by which a user-specified PRD of the reproduced ECG segments is guaranteed at the minimum entropy as shown in **Figure 6.7.**

A wavelet electrocardiogram (ECG) data based on the set portioning trees (SPIHT) compression algorithm is proposed in [27] The SPIHT algorithm [30] has achieved notable success in still image coding. The algorithm is modified for the one-dimensional case and applied to compression of ECG data.

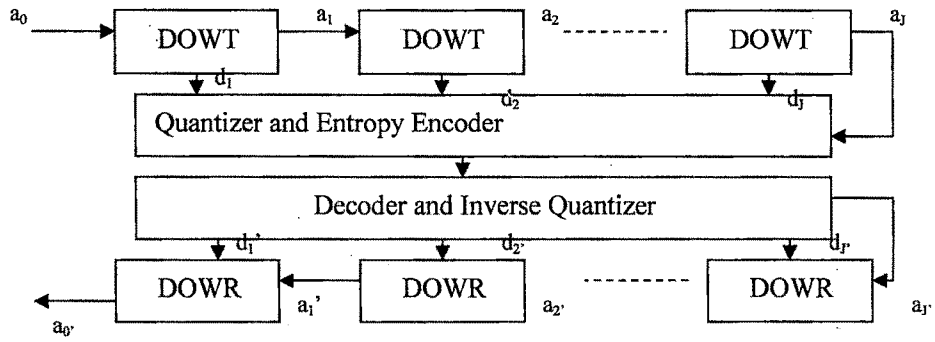


Figure 6.7 A generalized DOWT-based coding system

Experiments on selected records from the MIT-BIH arrhythmia database revealed that the proposed codec is significantly more efficient in compression and in computation than previously proposed ECG compression schemes. The coder also attains exact bit rate control and generates a bit stream progressive in quality or rate.

The fast-forward and inverse wavelet transforms are implemented as tree-structured, perfect- reconstruction filter banks. The input signal is divided into contiguous, non-overlapping blocks of samples called frames and is transformed frame by frame for the forward transform. Within each frame, the analysis filter pair to generate low-pass and high-pass signals, which are then down sampled by a factor of two, filters the input signal. Then this analysis filter pair is applied to the down sampled low-pass signal recursively to generate layered wavelet coefficients.

In implementation, the frame size, number of layers of the wavelet transforms, and the filter pair needs to be appropriately selected. The number of layers determines the coarsest frequency resolution of the transform and should be at least four for adequate compression. The frame size is taken to be a power of two that exceeds the number of layers. The frame should contain several periods of the ECG signal, but should still be short enough for acceptable coding delay and memory usage. Six layers of wavelet decomposition and 1024 sample frames are selected to fulfill the requirements. Among the potential perfect reconstruction filter pairs, ther is selected a bi orthogonal 9/7 tap filters [31], whose coefficients are in **Table 6.1**, have been chosen, because they have proved to offer the best compression performance for wavelet coding of ECG signals among all filters tested in [23]. Since these filters are symmetric, a symmetric (reflective) data extension scheme at the boundaries of the frames to obtain perfect reconstruction at the boundaries in the absence of coding is employed.

Table 6.1: The coefficients of the biorthogonal 9/7 tap filters

Low pass	0.852699	0.377403	-0.11062	-0.023849	0.037829
High pass	0.788485	0.418092	-0.04069	-0.064539	

After the wavelet transform, the SPIHT algorithm is used to encode the wavelet coefficients. The SPIHT algorithm has received widespread recognition for its notable success in image coding [30]. It has also been implemented in the case of one dimension for coding wavelet packet transforms of audio signals and obtained very good compression performance [32]. The SPIHT algorithm may be applied to the wavelet (purely dyadic) transform of ECG signals. The diagram of the encoder and decoder is shown as in **Figure 6.8**.

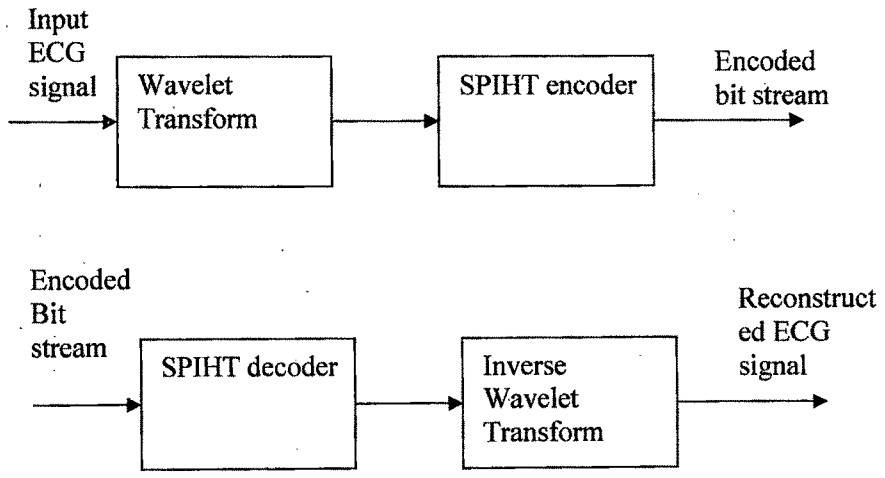


Figure 6.8 Proposed Encoder-Decoder

The principles of the SPIHT algorithm are partial ordering of the transform coefficients by magnitude with a set partitioning sorting algorithm, ordered bit plane transmission and exploitation of self-similarity across different layers. By following these principles, the encoder always transmits the most significant bit to the decoder.

Hybrid algorithm based on wavelet transformation of the linearly predicted error is proposed in [33]

Data compression of ECG signals allows long term digital storage and archiving of ECG recordings. The use of linear prediction after wavelet transformation has been shown [33] to improve the overall performance of the ECG wavelet compression method, giving a lower reconstruction error for the same compression ratio. With

PRD less than 4%, an average compression ratio of 20:1 has been achieved when the proposed algorithm is applied to different normal and abnormal signals from the MIT-BIH arrhythmia database. **Figure 6.9** shows operation of the hybrid method.

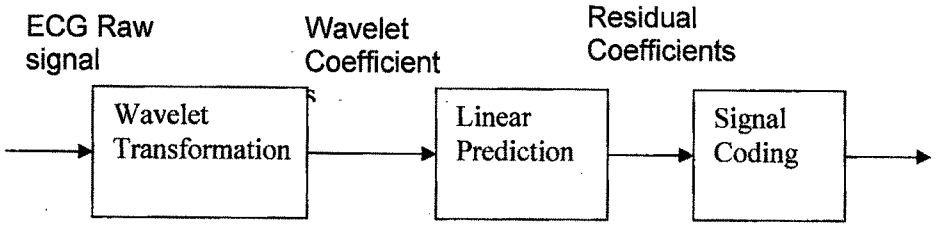


Figure 6.9 Hybrid Compression scheme

This has resulted in a substantial reduction in ECG signal bandwidth in the telemedicine applications and an increased storage capacity of the digital ambulatory records.

[33] describes wavelet packet based algorithm for the compression of single lead ECG. The algorithm combines the efficiency and flexibility of wavelet packet expansions with the methodology of the Karhunen-Loeve transform (KLT).

Associated with each library of wavelet packet is a pair of quadrature mirror filters (QMFs), one designed as a low pass filter, the other as a high pass filter. The wavelet packet transform of a finite sampled signal is computed by successively applying the QMF projections to generate a binary tree of transform coefficients.

Associated with each coefficient are three location parameters level within the tree, vertex along a level, and entry within a vertex- which determine the scale, frequency and position respectively of the underlying wavelet packet pattern.

Any collection of vertices, which satisfies the property that each path from the root to a leaf contains exactly one vertex from the collection, constitutes an orthogonal expansion of the original signal. So for a signal length of n, more than 2^n such expansions exist.

Selection of the one de composition which is to be utilized for further processing is accomplished by first assigning a cost to each vertex in the tree and then determining the orthonormal collection which attains the minimum total cost. The templates,

which comprise the chosen representation, are in some sense beat adapted to the features contained in the signal and, accordingly, are referred to as the best wavelet packet basis.

Given a set of vectors in R^N with N as integer power of 2, the mean vector is computed and subtracted from each member of the set. The resulting residual vectors are individually expanded into wavelet packet trees. The squares of the coefficients contained in these trees are accumulated into a separate tree of variances, upon which a best basis search is performed to determine the joint, or statistical, best basis for the ensemble. The components in the joint best basis are stored in order of the decreasing contribution to the total variance, and the basis is then truncated. Each residual is expressed in terms of the truncated basis to achieve a first stage of compression. Further compression is achieved by performing a KLT on the retained joint best basis coefficients.

6.4 Vector Quantization techniques

Vector quantization (VQ) is another technique that has been used extensively in data compression systems [34]-[36]. VQ has proved to be an effective scheme for image data reduction and it has also been very successful in coding speech parameters. The rate-distortion theory [37] establishes that compression performance can be improved by coding vectors instead of scalars; but this technique up to now has not been extensively applied to compress ECG signals, Examples of the application of the method on ECG signals can be found in [37],[38] and [12], [39], [40]. VQ in [39] follows the finite state VQ approach and the authors report data rates around 200 bps for an acceptable distortion level. The highest CR reported was 150, but is restricted to highly repetitive pattern and requires the additional task of QRS detection [32]. VQ can be employed in conjunction with any of the previously mentioned methods, mainly as a way of quantizing the resulting data [37],[38], [39]. However this technique can also combine redundancy removal with quantization in a single processing stage [12].

The gold washing (GW) adaptive vector quantization (AVQ) (GW-AVQ) [41], adaptive nature of the algorithm provides the robustness for wide variety of the signals. However, the performance of GW-AVQ is highly dependent on a preset parameter called distortion threshold (dth) which must be determined by experience or trial-and-error.

An algorithm is proposed in [41] that allows to assign an initial dth arbitrarily and then automatically progress toward a desired dth according to a specified quality

criterion such as the percent of root mean square difference (PRD) for electrocardiogram (ECG) signals. A theoretical foundation of the algorithm is also presented. This algorithm is particularly useful when multiple GE-AVQ codebooks and, thus multiple dth's are required in a sub band coding framework. Four sets of ECG data with entirely different characteristic are selected from the MIT/BIH database to verify the proposed algorithm. Both the direct GW-AVQ and a wavelet based GW-AVQ are tested. The result show that a user specified PRD can always be reached regardless of the ECG waveforms, the initial selection of dth or whether a wavelet transform is used in conjunction with the GW-AVQ. An average result of 6% in PRD and 410 bits/s in compressed data rate is obtained with excellent visual quality.

A direct waveform mean-shape vector quantization (MSVQ) is proposed in [42] as an alternative for electrocardiographic (ECG) signal compression. The mean values for short ECG signal segment are quantized as scalars and compression of the single-lead ECG by average beat subtraction and residual differencing their wave shapes coded through a vector quantizer. An entropy encoder is applied to both, mean and vector codes, to further increase compression without degrading the quality of the reconstructed signals. The fundamentals of MSVQ are discussed in [42], along with various parameters specifications such as duration of signal segments, the word length of the mean-value quantization and the size of the vector codebook. CR's in excess of 39 have been achieved, yielding low data rates of about 140 bps. This compression factor makes this technique especially attractive in the area of ambulatory monitoring.

Parameter extraction methods extract a set of parameters from the original signal which are used in the reconstruction process. The idea is to quantize a small set of extracted signal features, finely enough to render an almost imperceptible distortion. Among the methods that can be classified in this group Are : peak peaking methods [12], cycle pool-based compression (CPBC) algorithms [43], linear prediction methods [2], [44] and neural network methods [45], [46].

Karhunen – Loeve transform (KLT) [51], Fourier transform (FT) [52], [53] Cosine transform (CT), walsh transform (WT) Legendre transform (LT) [60], the optimally warped transform [61], subband coding [64], and in recent years the wavelet transform (WT)[55-56], [23],[24] has received great attention. However, these techniques, which are relatively insensitive to noise, generally rely on accurate QRS detection.

Table 6.2 summarizes reported performance of a number of ECG compression methods

Mehtod	CR	PRD(%)	SF (Hz)	ADC(bits)
TP[47]	2.0	5.3	200	12
AZTEC[48]	10.0	28.0	500	12
CORTES[10]	4.8	7.0	250	12
FAN/SAPA [49]	3.0	4.0	250	-
MSAPA/CSAPA[49]	5.0	3.5	250	8
ALZ77[50]	13.38	-	250	12
Dual application of KLT [51]	12.0	-	250	12
Fourier descriptors[52]	7.4	7.0	250	12
Adaptive Fourier coefficient estimation [53]	16.0	3.0	500	-
Sub-band coding with extensive Markov system [54]	25.0	-	500	12
Wavelet Transform[55]	9.9	-	500	-
VQ of Wavelet coefficients[56]	10.0	5.5	360	11
Wavelet packet compression[23]	8.06	-	200	12
Peak Peaking(spline) with entropy encoding [57]	10.0	14.0	500	8
Cycle pool based compression algorithms[58]	12.0	11.0	200	12
BP and NN/PCA neural networks [46]	20.0	13.0	360	11
Classified VQ [59]	8.6	24.5	200	12
Long term Prediction[2]	28.17	10.0	250	8

6.5 ANN based techniques

6.5.1 MLPANN for ECG signal compression

6.5.1.1 Introduction

Artificial Neural Network is a massively parallel-distributed processor made up of simple processing units, called neurons, which has a natural characteristic of storing experiential knowledge and making it available for use.

Prior to storage and transmission of representative data in speech, video or multimedia applications of neural networks, the network parameters must be quantized. Basically two approaches can be employed [62] In scalar quantization [63,

64], each network parameter is separately quantized: the quantizer observes a single parameter and selects the nearest approximating value from the predetermined finite set of allowed numerical values. In vector quantization, the parameters of the network prior to quantization are grouped into a vector which is a generic input to the vector quantizer Q comprised of a vector encoder and a vector decoder. The former maps its generic input to a finite set of indexes, each of which is associated with a vector. The set of vectors isomorphic to the set of of indexes comprises the code book of Q. This code book serves as the reproduction set for the vector decoder, which maps the finite set of indexes to this reproduction set. The codebook vectors (code vectors) are designed to be representative of the population of input vectors. An input vector to the encoder is compared with each element of the codebook to produce at the output an index that is associated with the code vector which is the closest, according to a pre selected measure, to the input vector. The elements of this code vector represent the parameters of the network after quantization. To alert the decoder about a code vector closest to the input vector, one either transmits or stores the index associated with the code vector. As the decoder has exactly the same code book as the encoder, it can receive the unique code vector from its index. VQ has the ability to exploit the linear and nonlinear dependencies among the vector components, and is highly versatile in the selection of multidimensional quantizer cell shapes. For a given resolution, use of VQ typically results in a lower distortion than scalar quantization.

Adding a quadratic term to the activation function of linear neurons can greatly enhance the representation ability of multilayer perceptrons without inflating their VC dimensions [63]. Circular back propagation (CBP) networks support both surface-based and prototype-based representations in classification problems; it has been shown that CBP is a unifying model for MLP's and Radial Basis Function (RBF) networks [64]. [65] proves that CBP encompasses vector quantization paradigms as well. Thus one can plug VQ prototypes in a CBP network, with the same number of neurons and supporting the same mapping. The fact that CBP structure can repeat the winner-takes-all (WTA) behaviour enables one to switch safely from one representation to the other, while preserving a network's mapping function. This property is exploited in [63] to initialize BP training.

6.5.1.2 Proposed ANN Implementation for Compression

The technique described here exploits the redundancy that naturally exists in most signals for efficient storage and transmission purpose. Smaller number samples are required to store a particular segment of ECG waveform than the number of analog samples required to represent original signal segment. At the receiver end, encoded compressed signal is decoded into full size segment.

Constructing a Multi Layer Perceptron (MLP) Artificial Neural Network (ANN) carries out ECG signal compression. The method takes definite number of samples as input and produces less number of hidden node outputs, which represent compressed signal samples. Reconstruction is possible at output layer nodes, which are in equal in number as nodes in input layer nodes, so that signal segment is recovered. Any performance criterion used to evaluate an ECG compression algorithm must include two factors: Amount of compression and Resultant reconstruction error. The error associated with ECG compression algorithm (**percent RMS difference (PRD)**) given by:

$$PRD = \left(\frac{\sum (x(i) - x'(i))^2}{\sum x^2(i)} \right)^{1/2} * 100 \qquad \text{.....(6.1)}$$

where x(i) is ith sample of input signal and x'(i) is the ith sample of reconstructed signal. A neural net architecture suitable for solving the ECG signal compression problem is shown in **figure 6.10.** This ANN can give Compression Ratio (Ratio of number of bits required to represent original signal to the number of bits required to represent compressed signal) of 2 as ratio of number of input layer nodes and hidden layer nodes is 2.

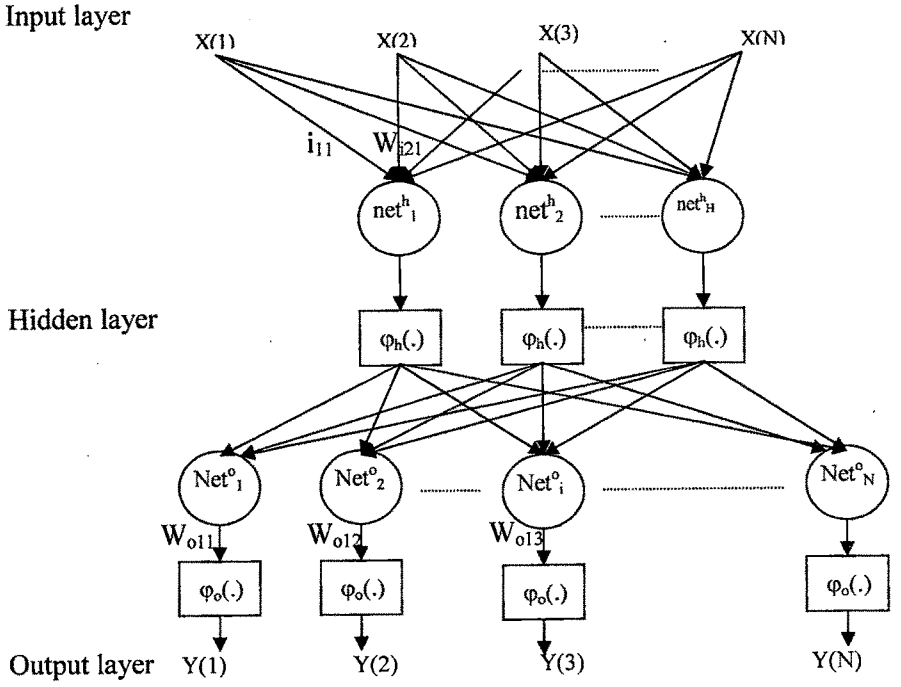


Figure 6.10 MLPANN for ECG signal compression

This type of structure in which input layer with a large number of input nodes are feeding into a small number of hidden layer nodes, which then feeds into a large number of output nodes in output layer is referred to as a **bottleneck type network**. The neural net shown in **figure 6.10** is trained to implement the identity map. A signal presented to the network as input would appear exactly same at the output layer. In this case, the network is used for ECG signal compression by breaking it in two. The transmitter encodes and then transmits the output of the hidden layer. The receiver receives and decodes the hidden layer outputs and generates outputs. Since the network is implementing an identity map, the output at the receiver is an exact reconstruction of the original signal.

A three layer MLPANN is used. Input layer is given total N number of samples from recorded ECG. Hidden layer is Intermediate level and units at the bottom in **figure 6.10** are **output units**. All the units at the lower layer are connected to each unit at the upper layer by the connection strength called weights. All the weights are initialized to small random values at the beginning. Training sets are prepared taking 6 samples at a time. Expected output is same as input. Compression is available at hidden layer. Hence in implementation input to hidden layer becomes part of transmitter and hidden to output layer becomes receiver.

6.5.1.3 MATLAB code

```
%-------------------------------------------------------------------------------
%                          TRAINING PHASE:
%-------------------------------------------------------------------------------
close all;
clear all;
M=500;                    %Number of training sets
TST=500;
I=6;
H=3;%hidden nodes
O=6;%output nodes
Time=5;                   %No. of Epochs
 eta= 0.1;      %Learning Rate
 e1=1;
 e2=1;
 tst=3000;              %Number of samples for testing
 xbi=ones(I,M);
 xbo=ones(H,M);
```

```
Wi=0.5*ones(H,I);%Wi=eye(H,I);
Wo=0.5*ones(O,H);%Wo=eye(O,H);
Wbi=zeros(H,I);
Wbo=zeros(O,H);      %For biasing
%-------------------------------------------------------------------------------
fid = fopen('recg.dat','r');
[X1,count]=  fscanf(fid, '%f',I*M);
fclose(fid);
X1=X1/100;
for P = 0:M-1
Up(:,P+1) = X1(1+P*I :(P+1)*I);              %Up is training input to the MLP
end
  Yp=Up;
%-------------------------------------------------------------------------------
% Reading one more data set for testing: Up1:LECG
%-------------------------------------------------------------------------------
fid = fopen('C3ECG.dat','r');
[X1,count]=  fscanf(fid, '%f',I*TST);
fclose(fid);
X1=X1/100;
for P = 0:TST-1
Up1(:,P+1) = X1(1+P*I :(P+1)*I);             %Up is training input to the MLP
end
Yp1=Up1;
%-------------------------------------------------------------------------------
MSE=zeros(Time,1);
momo(O,H)=0;
momi(H,I)=0;
alfa=0;
for T=1:Time
T
for k = 1:M
xbii=xbi(:,k);
xboo=xbo(:,k);
di=Wi;
do=Wo;
 x=Up(:,k);
 x1=(1.0-exp(-e1*Wi*x-e2*Wbi*xbii))./(1.0+exp(-e1*Wi*x-e2*Wbi*xbii));
```

```
y1=(1.0-exp(-e1*Wo*x1-e2*Wbo*xboo))./(1.0+exp(-e1*Wo*x1-e2*Wbo*xboo));
erro=Yp(:,k)-y1;
tmp = sum(erro.*erro);
MSE(T,1)= MSE(T,1)+tmp;
diffo=(1.0-y1.*y1);
do =  0.5* erro.*diffo;
diffi=(1.0-x1.*x1);
d11=Wo'*do;
d1=d11.*diffi;
for k1=1:H
Wo(:,k1)=Wo(:,k1)+eta*do.*x1(k1,1);
%Wbo(:,k1)=Wbo(:,k1)+eta*do.*x1(k1,1);
 end
for k1=1:I
Wi(:,k1)=Wi(:,k1)+eta*d1*x(k1,1);
%Wbi(:,k1)=Wbi(:,k1)+eta*d1*x(k1,1);
end
end
% MSE = MSE/M;
end
%Test the MLP WITH  Up:
%-------------------------------------------------------------------------------
for k = 1:M
u = Up(:,k);
xbii=xbi(:,k);
xboo=xbo(:,k);
 x1=(1.0-exp(-e1*Wi*u-e2*Wbi*xbii))./(1.0+exp(-e1*Wi*u-e2*Wbi*xbii));
 y1=(1.0-exp(-e1*Wo*x1-e2*Wbo*xboo))./(1.0+exp(-e1*Wo*x1-e2*Wbo*xboo));
Yn(:,k)=y1;
end
k=0;
for i=1:M
for j=1:O
k=k+1;
x(k)=Up(j,i);
dexp(k)=Yn(j,i);
end
end
```

```
x11 = [0 0 0 0 0 dexp];
x22 = [0 0  0  0 dexp 0];
x33 = [0 0 0 dexp 0 0];
x44 = [0 0 dexp 0 0 0];
x55 = [0 dexp 0 0 0 0 ];
x66=  [dexp 0 0 0 0 0 ];
y = (x11 + x22 + x33 +x44 +x55 + x66)/6;
y=y(1:M*O);
figure(1)
title('Waveforms during training');
subplot(4,1,1);plot(x,'r');grid;ylabel('Expected ECG');
subplot(4,1,2);plot(dexp,'g');grid;ylabel('Recovered');
err=x-dexp';
subplot(4,1,3);plot(err,'b');grid;ylabel('Error ');
subplot(4,1,4);plot(y,'r');grid;ylabel('Averaged');
sum = 0;
den = 0;
for i = 1:M*I
sum(1,:) = sum(1,:) + err(i,:).*err(i,:);
den(1,:) = den (1,:) + x(i,:).*x(i,:);
qty(1,:) = sum(1,:)./den(1,:);
prd(1,:) = sqrt(qty(1,:));
prd(1,:)=prd(1,:)*100;
end
%-------------------------------------------------------------------------------
Test the MLP WITH  Up1:
%-------------------------------------------------------------------------------
xbi=ones(I,TST);
xbo=ones(H,TST);
for k = 1:TST
u = Up1(:,k);
xbii=xbi(:,k);
xboo=xbo(:,k);
x1=(1.0-exp(-e1*Wi*u-e2*Wbi*xbii))./(1.0+exp(-e1*Wi*u-e2*Wbi*xbii));
y1=  (1.0-exp(-e1*Wo*x1-e2*Wbo*xboo))./(1.0+exp(-e1*Wo*x1-e2*Wbo*xboo));
Yn(:,k)=y1;
end
k=0;
```

```
for i=1:TST
for j=1:O
k=k+1;
x1(k)=Up1(j,i);
 dexp1(k)=Yn(j,i);
 end
end
x11 = [0 0 0 0 0 dexp1];
x22 = [0 0  0  0 dexp1 0];
x33 = [0 0 0 dexp1 0 0];
x44 = [0 0 dexp1 0 0 0];
x55 = [0 dexp1 0 0 0 0 ];
x66=  [dexp1 0 0 0 0 0 ];
y1 = (x11 + x22 + x33 +x44 +x55 + x66)/6;
y1=y1(1:TST*O);
figure(2)
title('Waveforms during testing');
subplot(4,1,1);plot(x1,'r');grid;ylabel('Expected ECG');
subplot(4,1,2);plot(dexp1,'g');grid;ylabel('Recovered');
err1=x1-dexp1';
subplot(4,1,3);plot(err1,'b');grid;ylabel('Error');
subplot(4,1,4);plot(y1,'r');grid;ylabel('Average');
sum1 = 0;
den1 = 0;
for i = 1:TST*I
sum1(1,:) = sum1(1,:) + err1(i,:).*err1(i,:);
den1(1,:) = den1(1,:)+x1(i,:).*x1(i,:);
 qty1(1,:) = sum1(1,:)./den1(1,:);
prd1(1,:) = sqrt(qty1(1,:));
prd1(1,:)=prd1(1,:)*100;
end
```

6.5.1.4 Results

During training as described earlier, value of error is computed and printed. Error is plotted as a function of time epochs, along with input as well as reconstructed waveform. PRD is calculated on line during training. **Figure 6.10 and Figure 6.11** are the waveforms during training and recall phase respectively with I = O = 6, H = 3 (and hence Compression ratio = 2:1),

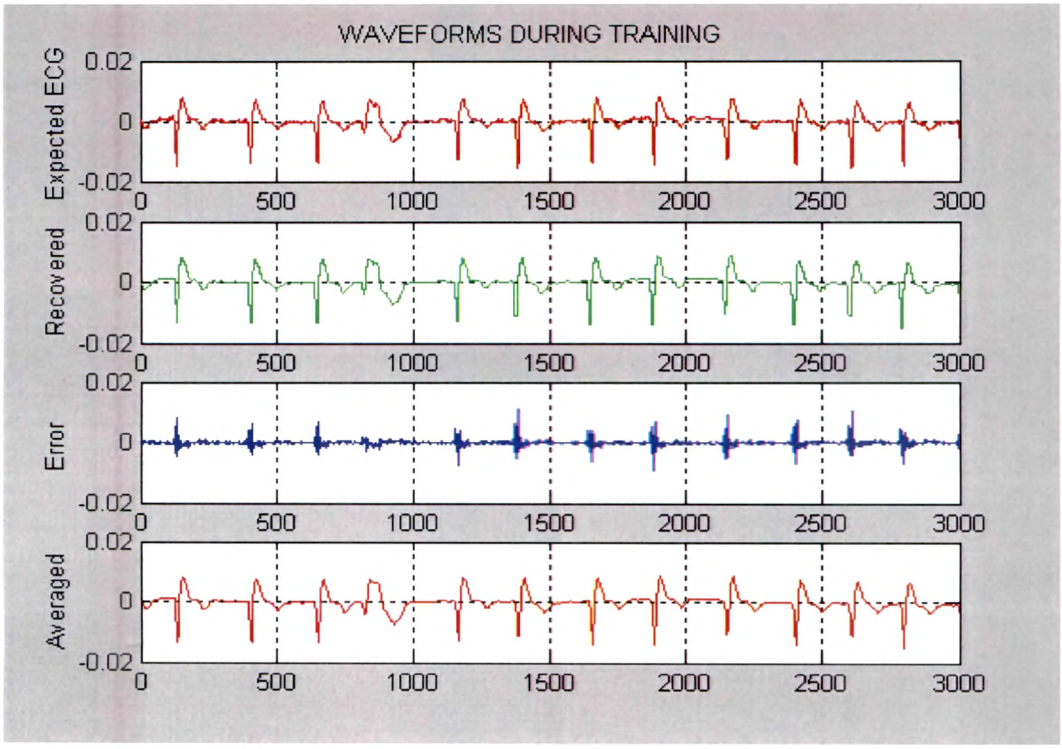
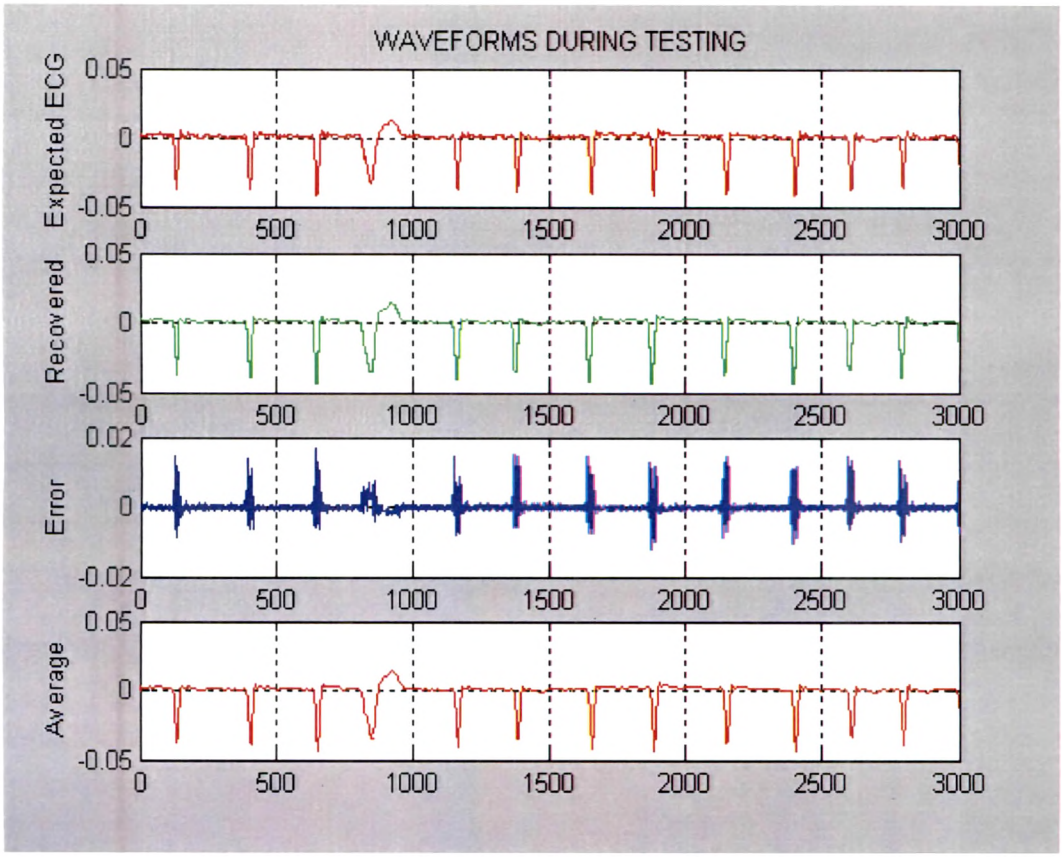


Figure 6.11 Waveforms during training phase for compression using MLPANN



igure 6.12 Waveforms during recall phase for compression using MLPANN

6.5.2 Competitive learning and VQNN for compression

6.5.2.1 Introduction

In competitive learning the output neurons of a neural network compete among themselves to become active. It is this feature that makes competitive learning highly suited to discover statistically salient features that are used to classify a set of input patterns. In the simplest form of competitive learning, the neural network has a single layer of output neurons, each of which is fully connected to the input nodes. The network is called Vector Quantization Neural Network, as illustrated in **Figure 6.13.** The input vector is constructed from a **L**-dimensional space. **K** output neurons are designed to compute the vector quantization codebook. As the neural network is being trained, all the coupling weights will be optimized to represent the best possible partition of all the input vectors.

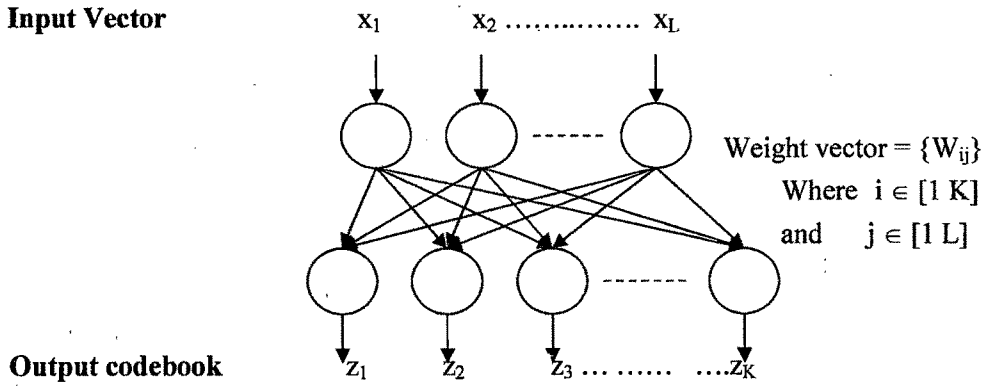


Figure 6.13 Vector Quantization Neural Network

6.5.2.2 Training and Testing Algorithm

To train this network, a group of data samples known to both encoder and decoder is designated as the training set, and the first **K** input vectors of the training data set are used to initialize all the neurons. With this general structure, various learning algorithms are already designed and developed. Some of them are Kohonen's self-organizing feature mapping , competitive learning , frequency sensitive competitive learning, etc. Frequency sensitive competitive learning algorithm is used.

For a neuron k to be the winning neuron, its induced local field S_k for a specified input pattern **x** must be the largest among all the neurons in the network. The output signal Z_k of winning neuron k is set equal to one; the output signals of all the neurons that lose the competition are set equal to zero. We thus write

$$Zk = \begin{cases} 1 & \text{if } S_k > Sj \quad \text{for all } j, j \neq k \\ 0 & \text{otherwise} \end{cases} \qquad (6.2)$$

where the induced local field S_k represents the combined action of all the inputs to neuron k.

A neuron then learns by shifting synaptic weights from its inactive to active input nodes. If a neuron does not respond to a particular input pattern, no learning takes place in that neuron. If a particular neuron wins the competition, each input node of that neuron relinquishes some proportion of its synaptic weight, and the weight relinquished is then distributed equally among the active input nodes. According to the standard **competitive learning rule**, the change Δw_{ij} applied to synaptic weight w_{ji} is defined by

$$\Delta w_{ij} = \begin{cases} \eta(xj - w_{ij}) & \text{if neuron } k \text{ wins the competition} \\ 0 & \text{if neuron } k \text{ loses the competition} \end{cases} \qquad (6.3)$$

where η is the learning-rate parameter. This rule has the overall effect of moving the synaptic weight vector **Wij** of winning neuron **i** toward the input pattern **x**.

When the synaptic weights are properly scaled they form a set of vectors that fall on the same N-dimensional unit sphere. However, for this function to be performed in a "stable" fashion the input patterns must fall into sufficiently distinct groupings to begin with. Otherwise the network may be unstable because it will no longer respond to a given input pattern with the same output neuron.

Under-utilization problem occurs in competitive learning in which some of the neurons are left out of the learning process and never win the competition. Frequency sensitive competitive learning algorithm solves the problem by keeping a record of how frequent each neuron is the winner to maintain that all neurons in the network are updated an approximately equal number of times. To implement this scheme, the local field of the neuron is factored by frequency, which gets incremented some numbers every time that neuron wins.

The above mentioned procedure can be put in form of algorithm as follows:

Step 1: Set L, η, time epochs(t), number of tests(T), internal time epochs(tin).
Step 2: Find total number of samples to read from file of samples.
Step 3: Initialize random values to weights.
Step 4: Form group of L samples as training Set.
Step 5: For time = 1 to t:
Step 6: For tests = 1 to T

a. Find Euclidian sum of distances between input vectors and weight vectors
b. Determine winning output node and modify weight according to (4) for tin times.
c. Increase frequency of that winner node

Step 7: End tests
Step 8: End time

6.5.2.3 Codebook Generation

Proposed algorithm is used for competitive learning of Vector Quantization Neural Network, which is used to generate a codebook for the Vector Quantization of ECG signal. The Competitive Learning Algorithm can successfully group the sample group to generate codebook and reproduce at the time of reconstruction.

A block of **L** samples are taken from ECG signal and treated each sample value as a component of a vector of size or dimension **L**. This vector of source outputs forms the input to the vector quantizer. At both the encoder and decoder of the vector quantizer, there is a set of **L**-dimensional vectors called the **codebook** of the vector quantizer. The vectors in this codebook, known as **code-vectors**, are selected to be representative of the vectors generated from the source output. Each code-vector is assigned an **index**. At the encoder, the input vector is compared to each code-vector in order to find the code vector closest to the input vector. The elements of this code-vector are the quantized values of the source output. In order to inform the decoder about which code vector is the closest to the input vector, there is transmitted index of the code vector. Because the decoder has exactly the same codebook, it can retrieve the code vector given its binary index. A pictorial representation of this process is shown in **Figure 6.14**.

The amount of compression will be described in terms of bits **per sample.** For a codebook of size **K**, and the input vector is of dimension **L**. In order to inform the decoder of which code-vector was selected, we need to use $\log_2 \mathbf{K}$ bits. Thus, the number of bits **per vector** is $\log_2 \mathbf{K}$ bits.

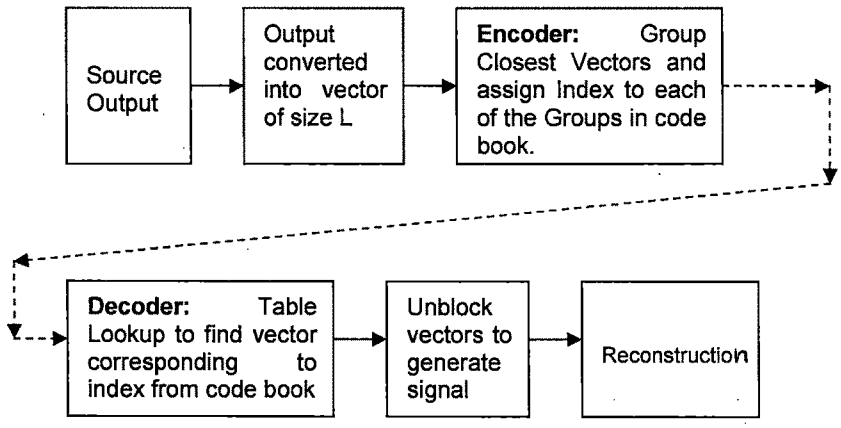


Fig 6.14: Vector Quantization

As each code vector contains the reconstruction values for **L** source output samples, the number of bits **per sample** would be [log2 **K]/L.**
Hence Compression Ratio,

'Bits per Sample' rate
$= B/\{(\log_2 K)/L\}$
$= B*L/\log_2 K$ ..... (6.4)

The percent RMS difference (PRD) given by **equation 6.1**

6.5.2.4 MATLAB Code and waveforms

(a) considering difference of adjacent samples

If we keep as reference first sample and then take difference of adjacent samples, as difference is smaller value than signal itself, it will take less number of bits to code the signal. It requires that first reference sample also be sent to receiver. In this and next algorithm, number of competing nodes and number of training patterns are taken same so that each node can have a chance to win and hence to be associated to that pattern through one unique index.

%VECTOR QUANTIZATION NEURAL NETWORK: Difference of adjacent %samples

```
close all;
clear all;
%-------------------------------------------------------------------------------------------
I=20;% Better training for I=20
H=100;
```

```
eta=0.01;
time=1000;
tests=50;
N=I*tests+1;
fid = fopen('R.dat','r');
[x,count]=  fscanf(fid, '%f',N);
fclose(fid);
%-------------------------------------------------------------------------------------------
% Initialize weights:
%-------------------------------------------------------------------------------------------
w=0.1*ones(H,I)
first=x(1,1);
for j = 1 : tests
n = 1;
m=((j-1)*I+1);
for i = m +1: I*j +1
Up(n,j)= x(i,1)-x(i-1,1);
n = n+1;
end
end       %Up is training input to the MLP
decode=zeros(I,H);
for t = 1:time
   t
f = ones(H,1);
z = zeros(H,1);
for l = 1:tests
max = w(1,:)*Up(:,l)/f(1);
loc=1;
%-----------------------------------------------------
%find loc:
%-----------------------------------------------------
for k = 1:H
if  max <= w(k,:)*Up(:,l)/f(k)
max = w(k,:)*Up(:,l)/f(k);
loc = k;
else
loc=loc;                  %nothing to do
end      %of if
```

```
end     %of k
%-------------------------------------------------------------
decode(:,loc)=Up(:,l);%store the vector
%-------------------------------------------------------------
z(loc) = 1;
f(loc)=f(loc)+1;
for j = 1:I
w(loc,j)=w(loc,j)+eta*(Up(j,l)-w(loc,j)');
end     %of j
end     %of t
end %of tests
%-------------------------------------------------------------------------
%testing for the same input patterns:Up:
%-------------------------------------------------------------------------
code=zeros(tests,1);
ztest = zeros(H,tests);
for i = 1: tests
max=w(1,:)*Up(:,i);
rank= 1;
for j = 1:H
op(j,i) = w(j,:)*Up(:,i);
if max<op(j,i)
max=op(j,i);
rank =j;
else
rank = rank;
end %of if
code(i,1)=rank;
end     %OF J
ztest(rank,i)=1;
Up1(:,i)=decode(:,rank);
end
x1(1,1)=first;
for j = 1 : tests
n = 1;
m=((j-1)*I+1);
for i = m +1: I*j +1
x1(i,1)  = Up1(n,j) + x1(i-1,1);
```

```
n = n+1;
 end
 end
figure(1);
subplot(2,1,1);plot(x,'r');ylabel('source signal');
subplot(2,1,2);plot(x1,'g');ylabel('reconstructed');
perror=((x-x1)./x)*100; %percentage error
figure(2);
plot(perror,'r');ylabel('% error');
err=x-x1;
sum = 0;
den = 0;
for i = 1:I*tests
sum(1,:) = sum(1,:) + err(i,:).*err(i,:);
den(1,:) = den (1,:) + x(i,:).*x(i,:);
qty(1,:) = sum(1,:)./den(1,:);
prd(1,:) = sqrt(qty(1,:));
prd(1,:)=prd(1,:)*100;
end
%-------------------------------------------------------------------
%testing for the different input patterns:Up11:
%-------------------------------------------------------------------
tests=2000/I;
clear x x1
N=I*tests+1;
fid = fopen('R.dat','r');
[x,count]= fscanf(fid, '%f',N);
fclose(fid);
first=x(1,1);
for j = 1 : tests
n = 1;
m=((j-1)*I+1);
for i = m +1: I*j +1
Up(n,j)= x(i,1)-x(i-1,1);
n = n+1;
end
end
code=zeros(tests,1);
```

```
ztest = zeros(H,tests);
for i = 1: tests
max=w(1,:)*Up(:,i);
rank= 1;
for j = 1:H
op(j,i) = w(j,:)*Up(:,i);
if max<op(j,i)
max=op(j,i);
rank =j;
else
rank = rank;
end %of if
code(i,1)=rank;
end     %OF J
ztest(rank,i)=1;
Up1(:,i)=decode(:,rank);
end
x1(1,1)=first;
for j = 1 : tests
n = 1;
m=((j-1)*I+1);
for i = m +1: I*j +1
x1(i,1)  = Up1(n,j) + x1(i-1,1);
n = n+1;
end
end
figure(3);
subplot(2,1,1);plot(x,'r');ylabel('source signal');
subplot(2,1,2);plot(x1,'g');ylabel('reconstructed');
perror=((x-x1)./x)*100; %percentage error
figure(4);plot(perror,'r'); ylabel('% error');
err=x-x1;
sum = 0;
den = 0;
for i = 1:I*tests
sum(1,:) = sum(1,:) + err(i,:).*err(i,:);
den(1,:) = den (1,:) + x(i,:).*x(i,:);
qty(1,:) = sum(1,:)./den(1,:);
```

```
prd1(1,:) = sqrt(qty(1,:));
prd1(1,:)=prd1(1,:)*100;
end
```

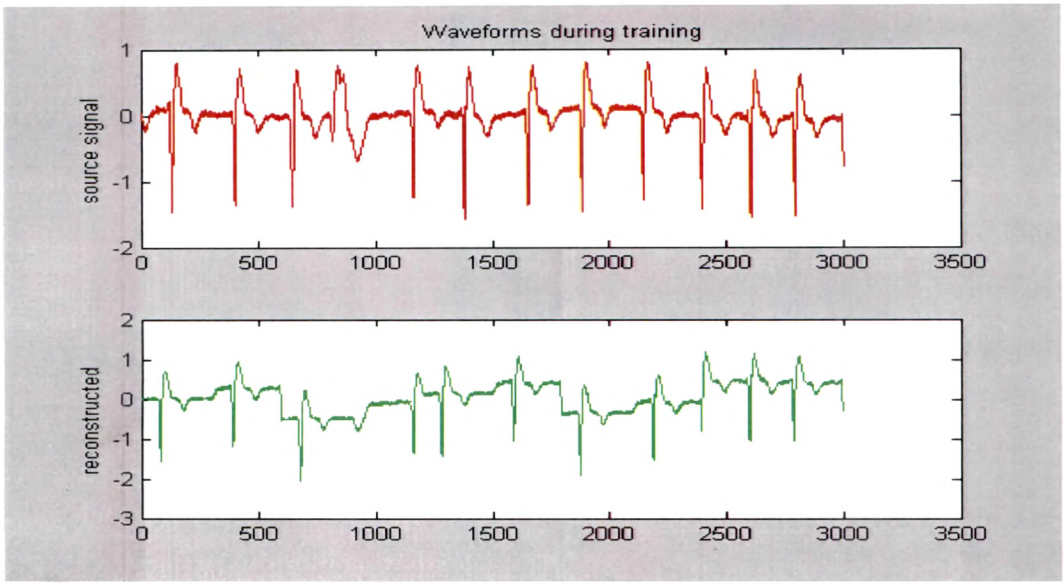


Figure 6.15 Waveforms for Training Phase for VQANN (Difference of adjacent samples)

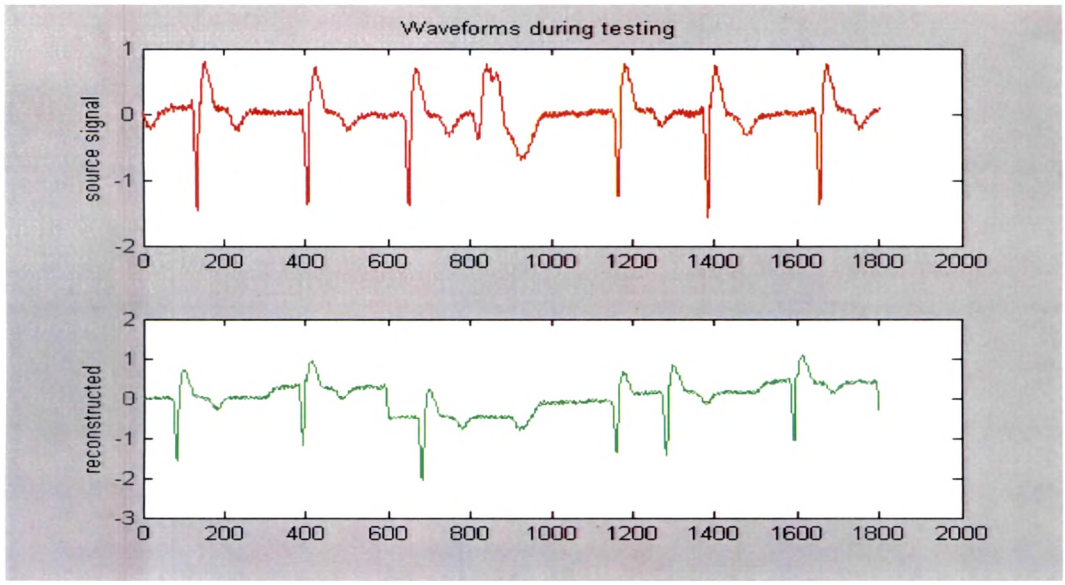


Figure 6.16 Waveforms for Testing Phase for VQANN (Difference of adjacent samples)

If the source output is correlated, vectors of source output values will tend to fall in clusters. More accurate representation of the source output during training and testing

phase can be obtained by selecting the quantizer output points to lie in these clusters. **Figure 6.15 and Figure 6.16** support this.

(b) Considering absolute samples

```
%VECTOR QUANTIZATION NEURAL NETWORK: modified algorithm:
% absolute samples
close all;
clear all;
%-------------------------------------------------------------------------------------------
I=4;% Better training for I=20
H=128;%Hidden nodes
eta=0.05;
time=8;%time epoches
tests=128;
N=I*tests;
fid = fopen('lecg.dat','r');
[x,count]=fscanf(fid,'%f',N);
fclose(fid);
%-------------------------------------------------------------------------------------------
% Initialize weights:
%-------------------------------------------------------------------------------------------
   w=randn(H,I);
%w=ones(H,I)/I;
%-------------------------------------------------------------------------------------------
first=x(1,1);
for j = 1 : tests
n = 1;
m=((j-1)*I+1);
for i = m +1: I*j +1
Up(n,j)= x(i-1,1);%no difference of samples
n = n+1;
end
end
decode=zeros(I,H);
%-------------------------------------------------------------------------------------------
for t = 1:time
      t
```

```
f = ones(H,1);%frequency of winning
z = zeros(H,1);
for l = 1:tests
min = sum(abs(Up(:,l)-w(1,:)'))*f(1);%abs(w(1,:))*abs(Up(:,l))/f(1);
loc=1;
%-------------------------------------------------------
%find loc:
%-------------------------------------------------------
for k = 1:H
if  min> sum(abs(Up(:,l)-w(k,:)'))*f(k)%abs(w(k,:))*abs(Up(:,l))/f(k);
min= sum(abs(Up(:,l)-w(k,:)'))*f(k);%abs(w(k,:))*abs(Up(:,l))/f(k);
loc = k;
else
loc=loc;                    %nothing to do
end      %of if
end      %of k
%-------------------------------------------------------------
decode(:,loc)=Up(:,l);%store the vector
%-------------------------------------------------------------
z(loc) = 1;
f(loc)=f(loc)+30;
for tin =1:4
for j = 1:I
 w(loc,j)=w(loc,j)+eta*(Up(j,l)-w(loc,j)');
 end     %of j
end %of tests
end%tin
end %of t
%-----------------------------------------------------------------
%testing for the same input patterns:Up:
%-----------------------------------------------------------------
code=zeros(tests,1);
ztest = zeros(H,tests);
for i = 1: tests
min = sum(abs(Up(:,i)-w(1,:)'));%max=w(1,:)*Up(:,i);
rank= 1;
for j = 1:H
op(j,i) =  sum(abs(Up(:,i)-w(j,:)'));%w(j,:)*Up(:,i);
```

```
if min>op(j,i) %max<op(j,i)
min=op(j,i);
rank =j;
else
rank=rank;
end %of if
code(i,1)=rank;
end     %OF J
ztest(rank,i)=1;
Up1(:,i)=decode(:,rank);
end %tests
x1(1,1)=first;
for j = 1 : tests
n = 1;
m=((j-1)*I+1);
for i = m +1: I*j +1
x1(i-1,1)  = Up1(n,j);
n = n+1;
end
end          %Up is testing input to the MLP
figure(1);
subplot(3,1,1);plot(x,'r');ylabel('original');
subplot(3,1,2);plot(x1,'g');ylabel('Recovered');
perror=((x-x1)./x)*100; %percentage error
subplot(3,1,3);plot(perror,'b'); ylabel('% error');
diff = sum(((x-x1)./1).^2)/sum(x.^2);
prd1=diff*100;
%-----------------------------------------------------------------
%testing for the different input patterns:Up11:
%-----------------------------------------------------------------
clear x,Up,Up1,rank,op,first;
N=I*tests1;
fid = fopen('lecg.dat','r');
[x,count]=  fscanf(fid, '%f',N);
fclose(fid);
first=x(1,1);
for j = 1 : tests1
n = 1;
```

```
m=((j-1)*I+1);
for i = m +1: I*j +1
Up(n,j)= x(i-1,1);
n = n+1;
end
end
code=zeros(tests1,1);
ztest = zeros(H,tests1);
for i = 1: tests1
min = sum(abs(Up(:,i)-w(1,:)'));%max=w(1,:)*Up(:,i);
rank= 1;
for j = 1:H
op(j,i)  = sum(abs(Up(:,i)-w(j,:)'));%w(j,:)*Up(:,i);
 if min>op(j,i)
min=op(j,i);
rank =j;
else
rank = rank;
end %of if
code(i,1)=rank;
end     %OF J
ztest(rank,i)=1;
Up1(:,i)=decode(:,rank);
end
x1(1,1)=first;
for j = 1 : tests1
n = 1;
m=((j-1)*I+1);
for i = m +1: I*j +1
x1(i-1,1)  = Up1(n,j);
n = n+1;
end
 end          %Up is testing input to the MLP
figure(2);
subplot(3,1,1);plot(x,'r');ylabel('original');
x11 = [0 0 0 0 0 x1'];
x22 = [0 0  0  0 x1' 0];
x33 = [0 0 0 x1' 0 0];
```

```
x44 = [0 0 x1' 0 0 0];
x55 = [0 x1' 0 0 0 0 ];
x66= [x1' 0 0 0 0 0 ];
y = (x11 + x22 + x33 +x44 +x55 + x66)/6;
y=y(1:N);
subplot(3,1,2);plot(y,'g');ylabel('Recovered');
perror=((x-y')./x)*100; %percentage error
subplot(3,1,3);plot(perror,'b'); ylabel('percentage error');
diff = sum(((x-y')./1).^2)/sum(x.^2);
prd2=diff*100;
```

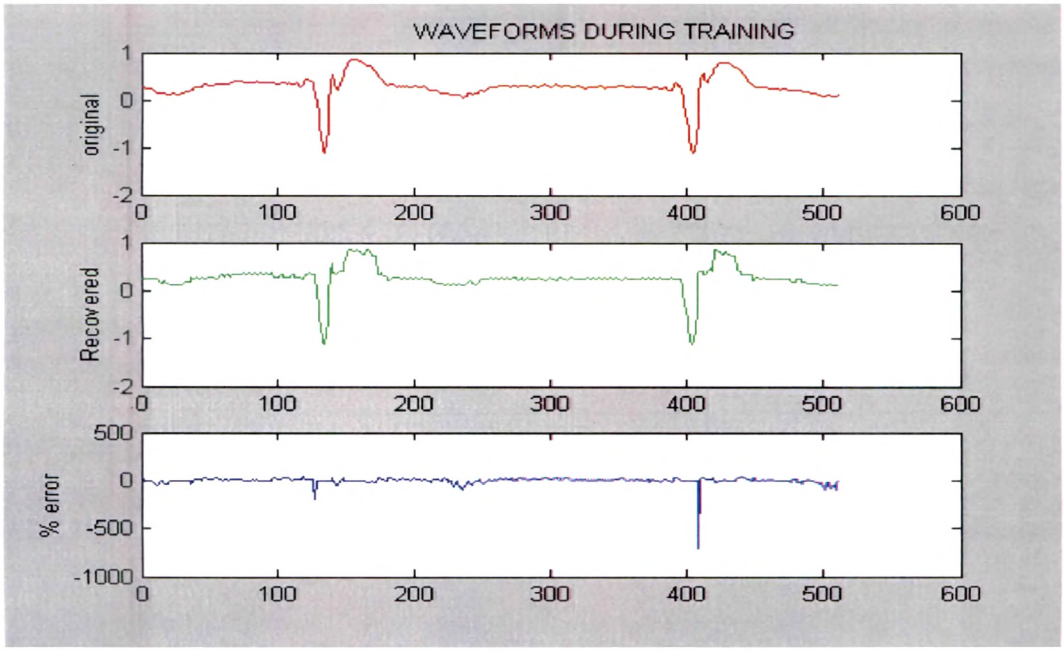


Figure 6.17 Waveforms for Training Phase for VQANN (absolute samples)

As can be seen from the waveforms of **Figure 6.17** during training and waveforms of **Figure 6.18** during testing, there is generated a code book at transmitter where group of L samples (having certain waveshape) is given a unique index. All such possible shapes are identified during training and assigned unique index. Instead of real signal, now sequence of such index is transmitted which takes less number of bits giving high value of CR. Also, as can be seen from these figures, there is a perfect reconstruction. Hence, PRD is also very low value. Better results can be obtained by keeping size of codebook larger.

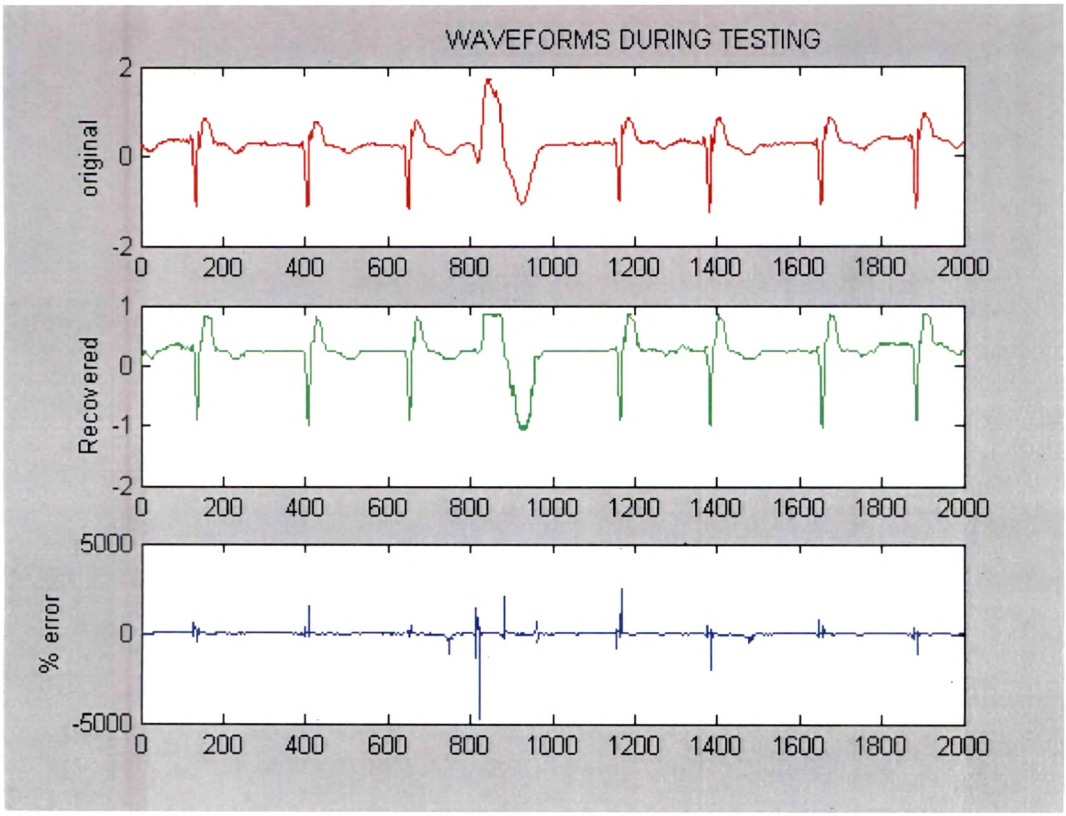


Figure 6.18 Waveforms for Testing Phase for VQANN (Absolute samples)

6.6 Extended backpropogation

The neural network models are used as models for the real brain functions or as computation devices. MLP employing back propagation LMS training technique is popular. Conventional BKP algorithm assumes that there is no correlation between parameters used for the evaluation of net function. Section describes an extension using Sugeno Integral to evaluate net function rather than linear function to capture relationship between parameters. The technique is applied for ECG signal compression.

6.6.1 Algorithm Development

The net value of net_j is obtained by aggregating all inputs to the neuron i. This depends on the Sugeno's Fuzzy Integral. Let

$x_j \in [0,1] \subset \Re$, where $j = 1, \ldots, n_o$ and

All network weights ➔ $w_{ij}^{(l)}$ $[0,1] \subset \Re$, where $l = 1,2$.

Also,

P(j) : be the power set of the set J of network inputs, where : J_{ij} is the j^{th} input of the i^{th} Neuron.

Let

g : P(J) → [0,1] be a function defined for layer 1 ; as follows:

$g(0) = 0$; $g(J_{ij}) = w_{ij}^{(l)}$; $n_o > j > 1$ .....(6.5)

Also

$g(J_{ij\,1}, J_{ij\,2}, \ldots\ldots J_{ij\,r}) = w_{ij\,1}^{(1)} \vee w_{ij\,2}^{(1)} \vee \ldots\ldots \vee w_{ij\,r}^{(1)}$

where $\{ j_1, j_2, \ldots j_r, \} \subset \{ 1, 2, \ldots, n_o \}$

And

$g(J_{i1}, J_{i\,2}, \ldots\ldots J_{i\,n\,0}) = 1$. [Λ → MIN operation, while

V → MAX operation in the unit interval $[0,1] \subset \Re$.]

..... (6.6)

It is proved that function { g } ➔ is a fuzzy measure on P(j) , hence the functional assumes the form of Sugeno integral over the finite reference set J . Let

h : P(I) → [0,1] be a similarly defined function for layer 2.

The net evaluation for the layers will take the form:

Hidden Layer:

$$net_i^{(h)} = \bigvee_{G \in P(J)} \left| \bigwedge_{p \in G} x_p \right| \wedge g(G) ,$$

The activation function is given by:

$$\varphi_h(W_i, X_i , b_i) = \frac{1 - \exp(-net_i^{(h)} - b_i)}{1 + \exp(-net_i^{(h)} - b_i)} \quad \ldots\ldots\ldots\ldots\ldots\ldots (6.7)$$

so that the value $\in [0,1]$.

Output Layer:

$$net_i^{(o)} = \bigvee_{G \in P(I)} \left| \bigwedge_{p \in G} x_p \right| \wedge h(G) ,$$

The activation function is given by :

$$\varphi_h(W_o, X_o , b_o) = \frac{1 - \exp(-net_i^{(o)} - b_o)}{1 + \exp(-net_i^{(o)} - b_o)} \quad \ldots\ldots\ldots\ldots\ldots\ldots (6.8)$$

so that the value $\in [0,1]$.

The back propagation can be used to obtain error signal.

6.6.2 Weight Update for jth output node

Weight update equation at time n+1 is:

$$W_{ji}(n+1) = W_{ji}(n)+\eta\delta_j(n)y_i(n) +\alpha\Delta W_{ji}(n-1) \quad \text{.....(6.9)}$$

Where

$$\delta_j(n) = \phi_j'(v_j(n))e_j(n) \quad \text{.....(6.10)}$$

$\phi_j'(v_j(n))$: is the derivative of the activation function at the input of the j^{th} node $v_j(n)$.

$e_j(n)$: is the difference between actual output and expected output at the j^{th} node, at instant n.

$y_i(n)$: is the input from i^{th} hidden neuron to the neuron j at output layer at time instant n.

α : Momentum factor

η : Learning rate

6.6.3 Weight Update for i^{th} hidden node

Weight update equation at time n is:

$$w_{ik}(n+1) = W_{ik}(n) + \eta\delta_i(n)x_k(n) +\alpha\Delta W_{ji}(n-1) \quad \text{..... (6.11)}$$

Where

$$\delta_i(n) =\phi_i'(v_i(n))\sum_{j=1}^{O}\delta_j(n)W_{ji}(n) \quad \text{..... (6.12)}$$

$\phi_j'(v_j(n))$: is the derivative of the activation function at the input of the j^{th} node $v_j(n)$.

$\delta_j(n)$: is value of δ calculated at j^{th} output node .

$W_{ji}(n)$: is the value of synaptic weight from i^{th} hidden node to j^{th} output node.

x_k: input from the k^{th} input node.

In order to obtain [$w_{ik}(n+1)$] $\in$ [0,1], weights are updated as follows :

[1]. If Expected Value = $\varphi_h(\mathbf{W_o}, \mathbf{X_o}, b_o)$ ➔ No adjustment is required.

[2]. If Expected Value > $\varphi_h(\mathbf{W_o}, \mathbf{X_o}, b_o)$ ➔

if [$w_{ij}(n)$ < Expected Value]

$w_{ij}(n+1) = 1\ \Lambda\ (w_{ij}(n) + \Delta. w_{ij})$

Else

$$w_{ij}(n+1) = w_{ij}(n)$$

[3]. If Expected Value $< \varphi_h(W_o, X_o, b_o a^{(2)}$ ➔

if [$w_{ij}(n) >$ Target Value]

$$w_{ij}(n+1) = 0 \vee (w_{ij}(n) + \Delta.w_{ij})$$

Else

$$w_{ij}(n+1) = w_{ij}(n)$$

With I = O = 6, H = 3 (and hence CR = 2:1), Number of time epochs = 5000. We obtained
PRD =

during training	28.6542
during testing	22.7810

Waveforms for the method are similar to **Figure 6.11 and 6.12**

In conventional methods, data compression is achieved by reducing number of bits representing a particular sample in the hidden layer. On the contrary, in the proposed method, number of bits per samples is reduced to half. Similarly, compression with any integer value is possible giving rise to different values of PRD. The extended back propagation was found to be conversing fast for the same error goal and target values, in comparison with the normal back propagation. It was found that the speed also depends on initial weights.