FFI RAPPORT

COMMUNICATION SIGNAL GENERATION AND AUTOMATIC CLASSIFICATION WITH DETECTION OF UNKNOWN FORMATS USING NEURAL NETWORKS

IVERSEN Alexander, KÅRSTAD Jørn

FFI/RAPPORT-2004/02934

COMMUNICATION SIGNAL GENERATION AND AUTOMATIC CLASSIFICATION WITH DETECTION OF UNKNOWN FORMATS USING NEURAL NETWORKS

IVERSEN Alexander, KÅRSTAD Jørn

FFI/RAPPORT-2004/02934

FORSVARETS FORSKNINGSINSTITUTT
Norwegian Defence Research Establishment
P O Box 25, NO-2027 Kjeller, Norway

FORSVARETS FORSKNINGSINSTITUTT (FFI)
Norwegian Defence Research Establishment

UNCLASSIFIED

P O BOX 25
N0-2027 KJELLER, NORWAY

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

REPORT DOCUMENTATION PAGE

1) PUBL/REPORT NUMBER FFI/RAPPORT-2004/02934	2) SECURITY CLASSIFICATION UNCLASSIFIED	3) NUMBER OF PAGES 51
1a) PROJECT REFERENCE FFI-II/832/113	2a) DECLASSIFICATION/DOWNGRADING SCHEDULE -	

4) TITLE

COMMUNICATION SIGNAL GENERATION AND AUTOMATIC CLASSIFICATION WITH DETECTION OF UNKNOWN FORMATS USING NEURAL NETWORKS

5) NAMES OF AUTHOR(S) IN FULL (surname first)

IVERSEN Alexander, KÅRSTAD Jørn

6) DISTRIBUTION STATEMENT

Approved for public release. Distribution unlimited. (Offentlig tilgjengelig)

7) INDEXING TERMS

IN ENGLISH:	IN NORWEGIAN:
a) Radio Signal Generation	a) Radiosignalgenerering
b) Modulation Recognition	b) Modulasjonsgjenkjenning
c) Neural Networks	c) Nevrale nettverk
d) Auto-Association Network	d) Autoassosiasjonsnettverk
e) Detection	e) Deteksjon

THESAURUS REFERENCE:

8) ABSTRACT

The performance of algorithms for signal classification is often characterized according to their ability to correctly classify various signals belonging to a set of known signal formats. However, in a non-co-operative setting it may also be expected that the classifier will be presented with signals of unknown formats, i.e. signal formats not belonging to any of the categories in the training set. Typically, a neural network classifier will classify an unknown signal to the known signal format that it resembles the most. In EW applications, however, the ability to detect unknown signal formats is imperative. In this report a hybrid classifier scheme is proposed. The performance of this hybrid classifier is tested both with respect to its ability to correctly classify known signal formats as well as its ability to detect outliers. The results indicate that this two-stage classifier is able to detect most of the unknown signal formats, but at the expense of misclassifying some of the known signals. The trade-off between correct classification and detection of new signal formats can be adjusted by setting an appropriate CIV (Class Inherence Verification) threshold. In addition to discussing the classification algorithms and their performance the procedure for generation of test signals is also described.

9) DATE	AUTHORIZED BY This page only	POSITION
2004-09-16	Vidar S. Andersen	Director

ISBN-82-464-0871-2

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

CONTENT

COMMUNICATION SIGNAL GENERATION AND AUTOMATIC CLASSIFICATION WITH DETECTION OF UNKNOWN FORMATS USING NEURAL NETWORKS

1 INTRODUCTION

Classification of communication signals based on technical analysis is an important task within the field of modern COM-EW. The focus for this work has been on developing a scheme for generating semi-realistic radio communications signals for the training and testing of modulation classification methods. The effort put into the signal generation set-up is an attempt to obtain a more realistic assessment of the classification methods than much of the existing research, which is often based on very simplified communication models.

Another shortcoming of existing research in this field is the lack of evaluation of modulation classifiers' abilities to treat spurious signals or modulation types that does not belong to any known class. A hybrid artificial neural network based approach for modulation classification with reliable rejection of unknown signal types have been developed and tested.

Chapter 2 provides a background on the problem area and a brief review of modulation classification methods. A framework and a file format for the generation of signal files are provided in Chapters 3 and 4. Chapter 5 addresses the estimation of bandwidth whereas Chapter 6 describes the feature extraction. In Chapter 7 and 8 the classifier designs and test results are presented. Chapter 9 concludes this report.

2 BACKGROUND

2.1 Electronic Warfare

A definition of electronic warfare (EW) is (1):

Military action to exploit the electromagnetic spectrum which encompasses the interception and identification of electromagnetic emissions; the employment of electromagnetic energy, including directed energy, to reduce or prevent the hostile use of the electromagnetic spectrum and actions to ensure its effective use by friendly forces.

Radio communication signals may be intercepted and analysed in terms of intensity, direction and type for counter measures such as jamming, or be demodulated for the purpose of intelligence and eavesdropping. In the latter case, determining the modulation type of the radio signal is crucial. But also for determining appropriate jamming techniques, knowledge of how the signal is modulated is hugely beneficial.

In a non-co-operative setting, modulation recognition introduces a number of challenging requirements. Factors such as noise, interference, propagation effects and spread spectrum techniques may seriously complicate the task, especially if little or no a-priori knowledge is at hand. Determining modulation types under such conditions usually requires trained operators and advanced signal analysis tools.

2.2 Automatic Modulation Recognition

Automatic modulation recognition (AMR) has been an active research topic since the early 1980s and a number of methods have been proposed. Azzouz and Nandi (2) (3) identify three principal categories of recognition methods: statistical methods, decision theoretic methods and neural networks.

Statistical methods include parametric methods where data belonging to a class is assumed distributed in a particular form, i.e. Gaussian and maximum-likelihood methods. If non-parametric, the distribution model is based on a set of available training data rather than a predefined model, i.e. *k*-nearest neighbour methods. Parametric methods are easy in that few parameters need to be optimised. The drawback is that the distribution model may not represent the true data distribution well. Non-parametric methods give better distribution models but require more parameters to be optimised. Examples of statistical approaches for modulation recognition are the likelihood method (4) and Bayes (5).

In the decision-theoretic approach, a number of key features are extracted from the intercepted signals. The modulation type is then determined by traversing a decision tree where the key features are in turn tested against threshold values in the tree nodes. Once the decision tree is set up, classification is very fast as it can be implemented simply as a set of *if-then-else* statements. It is therefore suitable for online classification and for implementation in resource-limited systems. Due to the simplicity and good classification abilities, many have proposed the decision theoretic approach to modulation recognition, e.g. (6), (7), (8).

Neural networks have been widely applied to AMR. Indeed, a number of studies have shown that neural networks outperform other techniques when it comes to classification rates and robustness to noise (7), (9), (10), (11).

The following sections consider three central matters in pattern recognition tasks such as AMR. These are the nature of the training and testing data (section 2.3), the feature extraction (section 2.4) and the classification proper (section 2.5).

2.3 Radio Signal Generation for Classifier Training and Testing

Non-parametric and neural network classifiers[1] learn to classify based on examples of the various classes. When building a classifier it is common to divide the available examples into two separate and non-overlapping sets for training and testing. The training set is used to train the classifier. The test set, which consists of examples that the classifier has not seen before, is used to assess the quality of the classifier. Sometimes another set, the validation set, is used to continuously evaluate the classifier performance during training. The use of these sets will be illustrated in section 2.5.

An essential requirement when training a classifier is that the training examples are representative of the data that the classifier is to handle when used in practice. In AMR, where classifier training is based on radio signal segments, this can be challenging. Noise, fading, interference and propagation effects in the transmission channel may affect and deteriorate radio signals radically, and finding a representative training set may be difficult. In the AMR research, this has been approached in different ways. Most commonly, classifiers have been trained and tested on synthetically generated signals in a simple additive white Gaussian noise channel model with training and testing data consisting of signals with different signal-to-noise ratios (SNR). A measure of quality has been the SNR level at which the classifier breaks down, i.e. (7), (9). Others have taken account of channel distortions models in addition to the additive white Gaussian to obtain a more realistic picture of the classifier performance. E.g., in (16) and (18) the effects of AMR in channel models representing fading and multi-path effects in rural, small-town and urban environments are investigated. Yet another approach is to base the training on real off-air signals (11). The drawback of this approach is that it can be very time consuming and that it is not possible to control the signal and channel parameters to the same extent as for generated signals. In most practical applications, radio signals are pulse-shaped to meet limited bandwidth requirements. Few have considered AMR for such band-limited signals (17).

[1] Strictly speaking, neural networks are also non-parametric but are mentioned separately for clarity.

2.4 Key Feature Extraction

In pattern recognition it might be necessary to reduce the dimensionality of the data before presenting it to the classifier. This dimensionality reduction is called feature extraction and consists of finding a smaller set of characteristics that helps the classifier to generalise between examples of the same class and to discriminate between examples of different classes. Feature extraction can be data independent, e.g. principal component analysis (PCA), or data dependent, where a-priori knowledge helps extracting useful features.

In AMR, feature extraction is essential as classification often is based on signal segments consisting of thousands of samples. Proposed feature extraction techniques have been based on: spectral analysis (19); instantaneous attributes (amplitude, phase and frequency) (7), (9); higher order statistical moments and cumulants (16), (18); and shape matching of signal constellations (20). Of these, features extracted from the instantaneous attributes and from higher order moments and cumulants seem to have attracted most attention.

The quality of the features is very important for the classification. Good features facilitate the task of the classifier whereas low quality features make the classification process harder. Similarly, a high quality classifier may compensate for features of lower quality. Whereas the feature extraction usually is highly problem-dependent the classifier design is more generic. The next section considers neural network-based classification.

2.5 Neural Network-based Classifiers

The general function of a neural network is to produce an output pattern when presented to a particular input pattern. It can therefore simply be seen as a mapping function. The concept is taken from the brain's ability to recollect on the basis of certain input patterns. Learning these mappings is done in conceptually the same way as the brain, that is, generalising from a number of examples. Neural networks consist of a number of fairly simple computational devices that resemble the neurons in the brain interconnected with weighted connections that resemble dendrites and axons.

Several types of neural networks exist but the most common one used for AMR has been the Multi-layer Perceptron. Before explaining the Multi-layer Perceptron, it is useful to look at its basic constituent: The Perceptron.

2.5.1 The Perceptron

A Perceptron models a neuron by receiving weighted inputs and returning a binary output depending on whether the weighted sum of inputs is less or greater than an adjustable threshold. The Perceptron consists of the weights, the summation processor and the adjustable threshold (Figure 2.1). The inputs and the weights can be positive or negative real values. If the sum of the weighted inputs is greater than, or equal to, the threshold, the Perceptron is said to fire, and outputs a 1. If the weighted sum is below the threshold the output is 0. The

Perceptron's ability to learn is a matter of modifying the values of the weights and the threshold until the required behaviour is obtained.

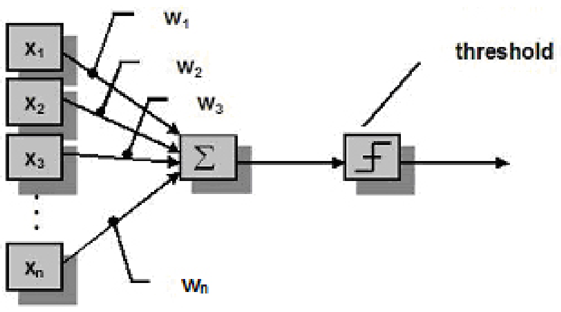
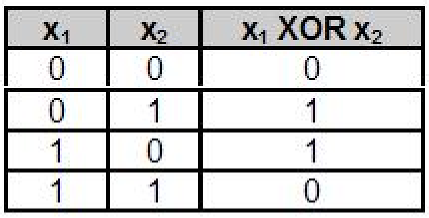


Figure 2.1. A Perceptron. Inputs (x_i), weights (w_i), summation processor and threshold device.

A Perceptron can only be used in linearly separable 2-class classification problems. This is a major limitation, as can be illustrated by looking at the exclusive-OR (XOR) problem. An XOR gate is supposed to output 0 if the two inputs are equal and 1 otherwise. The Perceptron cannot solve this somewhat simple problem because the output classes are not linearly separable (Figure 2.2).



x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

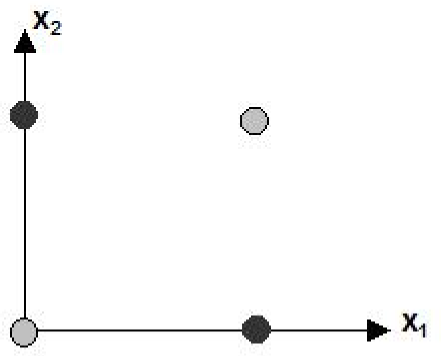


Figure 2.2: The XOR problem. Zeros (grey) and ones (black) are not linearly separable.

The limitations of the Perceptron led to the development of the Multi-layer Perceptron with the back-propagation training algorithm.

2.5.2 Multi-layer Perceptron

A Multi-layer Perceptron (MLP) is able to handle more complex and non-linear classification problems. It consists of one input layer, one or more hidden layers and one output layer of computational nodes (Perceptrons). As opposed to the single Perceptron that outputs either 0 or 1 depending on a threshold value, each node in the MLP can produce any real valued output. The nodes' outputs are determined by their activation function. The most common activation functions are the s-shaped logistic sigmoid (log-sigmoid) function that ranges from 0 to 1 and the hyperbolic tangent sigmoid (tan-sigmoid) function that ranges from –1 to 1.

Each node in a layer is connected to all nodes in the layers immediately behind and in front, but there are no connections between nodes within a layer. The input signal is propagated through the network in a forward direction on a layer-by-layer basis. Figure 2.3 shows an example of an MLP with two hidden layers. In a classification task, such a network typically takes as input a number of key features that are extracted from the data. These values are then propagated through the network to the output layer. If each output node represents a class, the output node with the highest output value determines the predicted class.

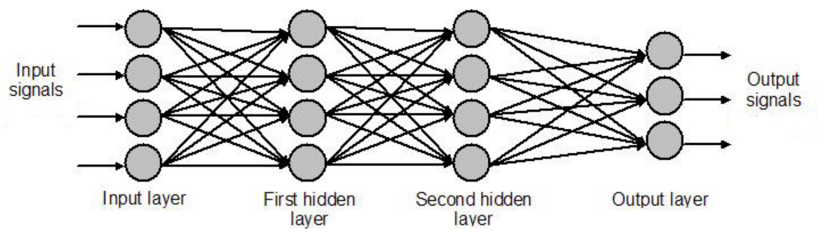


Figure 2.3. A four-input and three-output MLP with two hidden layers.

The design of the MLP is highly problem-dependent and there exist no fixed rules for the number or structure of hidden layers and nodes. The design is often subject to the experience of the designer and a number of heuristics and rules of thumb. Too small a network structure may prevent the MLP from classifying correctly whereas a large network may cause an unnecessary long training- and execution time and over-fitting. Over-fitting occurs when the MLP has adjusted its parameters to fit the training examples to such an extent that it cannot handle unseen data. The result is very good classification results for the training data but poor results for unseen data. The goal is to learn the network to generalise such that it can recognise unseen data as belonging to one of the trained classes. Figure 2.4 shows when the training of the network should be stopped to obtain best generalisation and to avoid over-fitting. This can be achieved using a technique called early stopping. A separate validation set is used during training to evaluate the performance on the unseen validation data. If, during training, the performance of the validation set decreases whilst the performance of the training set continues to increase the training is stopped.

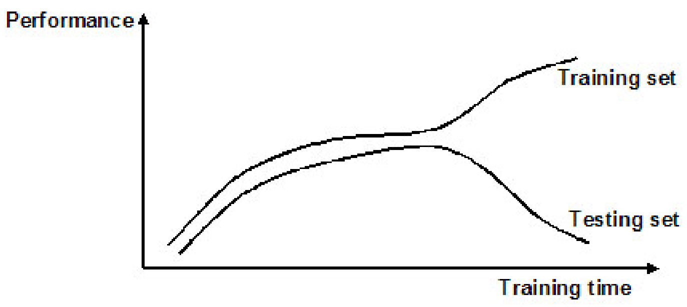


Figure 2.4. Over-fitting starts where the two lines start to diverge.

2.5.3 Back-propagation

Back-propagation learning is a popular algorithm for training an MLP. Basically, the learning consists of two passes through the different layers of the network: forward and backward. When an input is presented to the input layer, it is propagated through the network with fixed and initially random weights on the connections. This is the forward pass. When this is completed, the output from the network is compared with the desired output (usually 1 on the node that represents the class of the input and 0 on the other nodes) to create what is called an error-signal. This is usually the mean-squared error (MSE) – the average squared difference between the outputs and targets. The error-signal is then propagated back through the network from the output layer to the input layer. During this back-propagation, the weights are adjusted to make the network produce an output that is closer to the desired one (12). There are two principal approaches to back-propagation: incremental training and batch training. In incremental training, the back-propagation and weight update is applied after each training example. In batch training, the weights are updated after the presentation of the entire training set. The training time is usually measured in epochs, which represents the number of times the complete training set has been presented to the network during training. There exist many variations of back-propagation learning. The one used in this work is called resilient back-propagation (21).

Once the training is complete, the execution phase is conducted in one forward pass. This is a fast process that involves no other computations than summing the inputs and producing an output at each node in the network.

2.5.4 Detection of Unknown Data Types

When using an MLP for classification, the network attempts to find decision boundaries to enable it to discriminate between the specified classes. So what happens when the network is presented to an input that does not belong to any of the known classes? This may not be an issue if we know that any input belongs to one of the known classes. We know, for example, that a tumour is either benign or malignant. In other cases we want to recognise one or several classes out of a possibly infinite number of cases. Examples of such are entry security system based on retina recognition, where the objective is to recognise the staff and reject all others; and fault diagnosis in engines, where the objective is to detect any deviation from the normal operation.

The principal problem with detection/rejection of unknown patterns or verification of known patterns (general term: novelty detection) is that, most often, only positive examples are available. The MLP approach of determining decision boundaries between the classes is therefore problematical as adjoining classes are unknown. What is required for novelty detection is thus that decision boundaries of the classifier be closed around the examples that represent a class. A classifier that creates open boundaries may classify novel and possibly fundamentally different inputs as belonging to one of the known classes. Figure 2.5 illustrates how a two-dimensional classification problem may respond to novel examples. In this example classifiers A and B are trained to classify between squares and circles using a training data set (the squares and circles in the figure). Classifier A creates an open decision boundary that separates the two known classes. However, the novel triangle examples will also be classified as circles. Classifier B forms closed decision boundaries around the two known classes. This prevents the novel examples being classified as any of the two known classes. Thus, classifier B detects that the triangles does not belong to either of the classes.

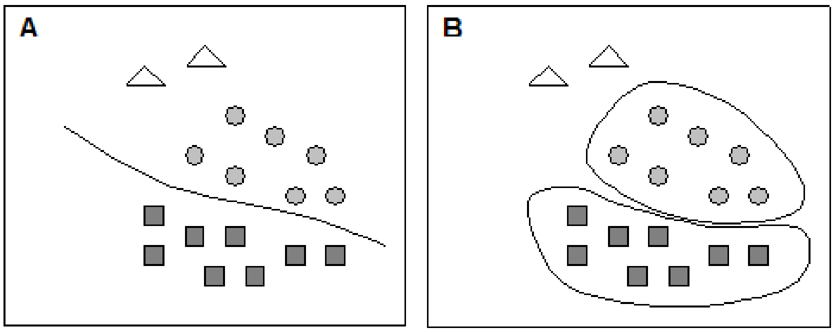


Figure 2.5. The effect of open (A) and closed (B) decision boundaries for novelty detection.

It has been shown that, despite good discriminatory properties, the MLP does not handle novel cases very well. Gori and Scarcelli (13) proved that determining whether an MLP forms closed or open decision boundaries is unfeasible (NP-hard). In problems with an unconfined class-space this property is highly undesirable. There are, however, other pattern recognition methods that handle this problem. Markou and Singh (14), (15) provide an overview over a wide variety of both statistical and neural network-based techniques for novelty detection. The next section describes one of these techniques: the feed-forward auto-associative neural network.

2.5.5 Feed-forward Auto-associative Neural Network

An alternative usage of the MLP is to use it as an associative memory. Associative memory is a memory that can be retrieved based on an incomplete and possibly spurious version of itself. A dramatic example of associative memory is how people that have previously been involved in, say, a traffic accident can create in their minds a complete visualisation of the accident when they, maybe years later, see a car similar to one that was involved. A more moderate example is how we recognise unique and sketchy hand-written versions of characters on a daily basis.

An MLP can be designed as an associative memory as shown in Figure 2.6. The objective is to recreate the input pattern at the output layer. Thus, if a pattern is presented at the input layer, we want the same pattern to appear at the output layer after the values are propagated through the network.

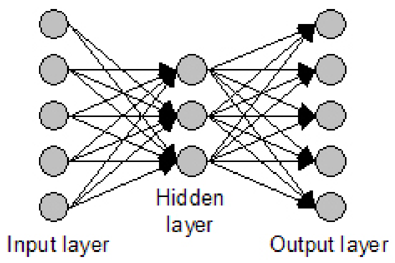


Figure 2.6. Auto-associative MLP with 5 input and output nodes and 3 nodes in hidden layer.

The hidden layer, with fewer nodes than the input and output layers, plays an important role in the auto-association network. If the network manages to recreate the input pattern at the output layer, the activation values of the hidden nodes comprise of a compressed representation of the input pattern. This is analogous to the principal components in a principal component analysis (PCA)(23). The auto-association network can be used for novelty detection by training the network to recreate at the output layer the feature values of training examples of a class. If, after training, a representative example of the class is presented to the network, it will produce an output that is very similar to the input and hence produce a small input-output difference. If a fundamentally different example is presented to the network, the network will fail to reproduce this pattern at the output. The input-output difference will therefore be much greater. Consequently, an input-output difference threshold can be used to detect unrepresentative (novel) examples.

3 RADIO SIGNAL GENERATION

In the open literature concerned with AMR, the vast majority have trained and evaluated their classification methods on synthetically generated signals. As a step towards a more realistic assessment of the limitation and possibilities of classification methods, we initiate the generation of semi-realistic radio communication signals. Here, semi-realistic means signals that are created and distorted (simulated) in software but generated as a radio frequency (RF) signal by a signal generator. It is received and down-converted by a receiver and then sampled by an A/D converter. The signal simulation is convenient as it enables the user to control the distortion. Most practical applications have limited bandwidth and therefore have to apply some form of pulse shaping to the signal to reduce its bandwidth before transmission. This is often ignored in AMR studies, but is considered here. We also perform blind receiver bandwidth estimation.

3.1 Set-up and General Operation

The set-up of the signal generation and receiver is illustrated in Figure 3.1.

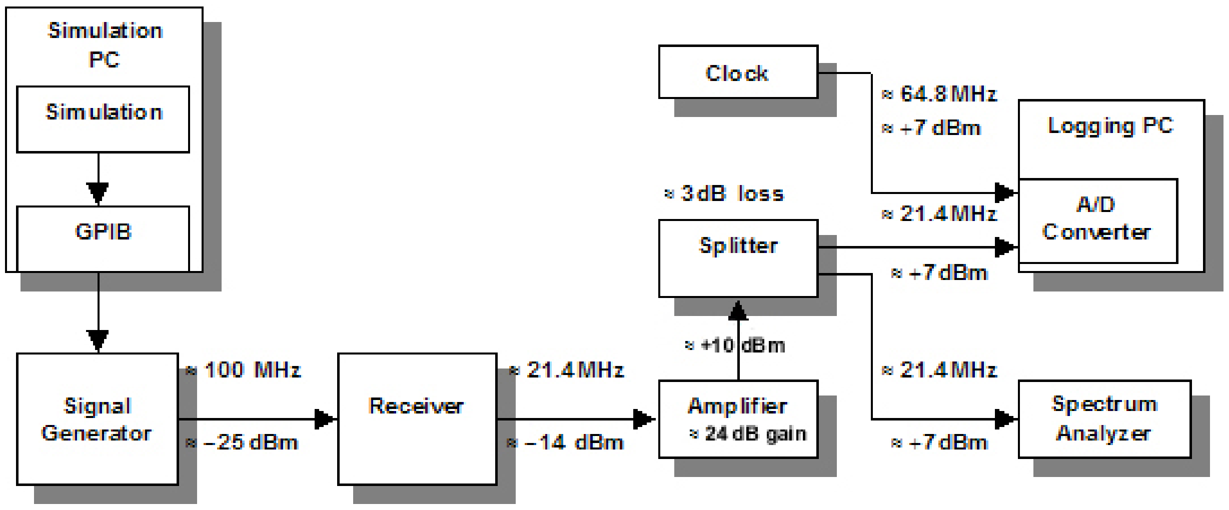


Figure 3.1. Signal generation set-up

With reference to Figure 3.1, the components used are described in Table 3.1.

Label	Description
Simulation PC	Dell Inspiron 4000 PIII laptop, Windows XP, 384 MB RAM
Simulation	WinIQSIM v.4.20, I/Q simulation software, Rhode & Schwarz
GPIB	PCMCIA-GPIB, instrument control card, National Instruments[2]
Signal Generator	SMIQ-06B, signal generator, 300kHz-6.4GHz, Rhode & Schwarz
Receiver	Compact Receiver ESMC, 20-1300 MHz, Rhode & Schwarz
Amplifier	ZKL-2R7, amplifier, Mini Circuits
Splitter	15542 ZSC-2-1W, 3 dB splitter, Mini Circuits
Clock	Signal generator 2022D, 10kHz-1GHz, Marconi Instruments
Spectrum Analyzer	Spectrum Analyzer 8596E, 9kHz-12.8GHz, Hewlett-Packard
A/D Converter	ECDR-21X, 2-channel 14-bit A/D receiver, ECHOTEK
Logging PC	Hewlett-Packard P4 PC, Windows 2000, 512 MB RAM

Table 3.1. Component description

3.2 Signal Generation Stage

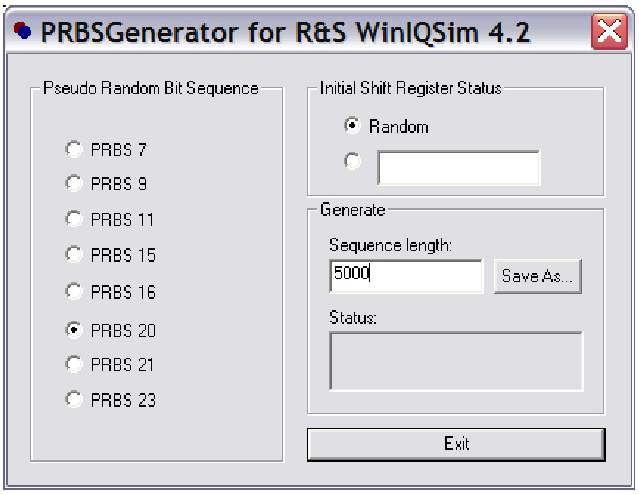
The WinIQSIM software allows for the set-up of digitally modulated single and multi-carrier I/Q baseband signals with a wide range of simulated distortions such as phase and additive noise, interference and multi-path effects.

3.2.1 Data Source

The user can choose the type of bit patterns that are to be modulated onto the signal. In this work, we consistently use a pseudo-random bit sequence (PRBS). For the training of classifiers

[2] After installing the PCMCIA-GPIB driver on some Windows systems, we have experienced that the plug-and-play installation process has not found the proper driver. To avoid this problem, choose to select the appropriate driver manually when prompted.

it is important to use different bit sequences to enable generalisation and prevent it from learning specific bit patterns. The PRBS generator integrated in WinIQSIM (feed-back shift register, CCITT V.52) should be avoided as it always starts off with the same initial shift register status. A detached PRBS generator application has therefore been developed (Figure 3.2). It is based on the same CCITT V.52 standard but starts off with a time-seeded random initial shift register status. Alternatively, the user can manually specify an initial shift register status. The PRBS generator creates files containing the bit sequence in a format suitable for WinIQSIM data source import. Thus, the user should create and import a unique bit sequence file for each signal sequence to generate.


PRBSGenerator for R&S WinIQSim 4.2

Pseudo Random Bit Sequence
- PRBS 7
- PRBS 9
- PRBS 11
- PRBS 15
- PRBS 16
- [✓ PRBS 20]
- PRBS 21
- PRBS 23

Initial Shift Register Status
- [✓ Random]
- ○

Generate

Sequence length: 5000

Save As...

Status:

Exit

Figure 3.2. Pseudo-random bit sequence generator.

Note that the number of bits to generate ("sequence length" in Figure 3.2) is related to the symbol length and the modulation type's bit-rate. It is important to choose a sequence length long enough to avoid repeated bit sequences. E.g., a 2FSK sequence of 20000 symbols carries 20000 bits (1 bit/symbol * 20000), whereas a 16QAM sequence of 20000 symbols carries 80000 bits (4 bit/symbol * 20000).

3.2.2 Modulation Types

Through the WinIQSIM graphical user interface, the user has a choice of variations of PSK, QAM and FSK modulation types. Other modulation types, such as ASK modulations, must be defined by the user and stored to file according to a specific format. Details about the definition of modulation types can be found in the WinIQSIM help files. Appendix C shows examples of 2ASK and 4ASK definitions.

3.2.3 Signal Distortion

Only white noise is added to the signals used in this work. Noise can be defined as signal-to-noise ratio (SNR) or as a bit-energy-to-noise-energy ratio (Eb/N_0). *Eb* can be defined as:

$$Eb = \frac{S}{R}, \tag{3.1}$$

where *Eb* is the energy per bit, *S* is the signal power and *R* is the bit rate. Bit rate is defined as the number of data bits per second. E.g., the bit rate of 2PSK is 1 * symbol rate, whereas the bit rate of 4PSK is 2 * symbol rate. N_0 can be defined as:

$$N_0 = \frac{N}{W} \tag{3.2}$$

where *N* is the noise power and *W* is the receiver bandwidth. Now, Eb/N_0 can be written as:

$$\frac{Eb}{N_0} = \frac{\left(\frac{S}{R}\right)}{\left(\frac{N}{W}\right)} = \frac{S}{N} \cdot \frac{W}{R} \tag{3.3}$$

Rewriting (3.3) we get:

$$\frac{S}{N} = \frac{\left(\frac{Eb}{N_0}\right)}{\left(\frac{W}{R}\right)} \tag{3.4}$$

which in a dB-scale defines the relationship between Eb/N_0 and the signal-to-noise ratio (SNR):

$$SNR = \frac{Eb}{N_0} - 10\log\left(\frac{W}{R}\right) \tag{3.5}$$

The receiver bandwidth (*W*) is not known at the transmitter. When generating signals, the user therefore has to specify noise as Eb/N_0 and provide a noise bandwidth. We consistently choose "full" noise bandwidth, which in WinIQSIM is defined as half of the sampling rate (125 kHz). For the evaluation and comparison of classifiers it is convenient to group signals according to SNR. Since 2- and 4-level modulation types with the same Eb/N_0 give a 3 dB difference in SNR (assuming that the bandwidth is constant), we choose to generate signals with 3 dB intervals. 2ASK, 4ASK, 2PSK, 4PSK, 2FSK and 4FSK sequences, each with 0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30 and 33 dB Eb/N_0 are generated. In addition, we generate MSK and 16QAM signals in the same Eb/N_0-range that will only be used in some testing scenarios.

3.2.4 Generation

Table 3.2 shows some of the parameters that are kept constant for all the generated signals. Note that the signal files' header contain this and other information.

Parameter	**Values**
Symbol rate	10 kHz
Sampling rate	250 kHz
Pulse shaping filter	Cosine (roll-off: 0.95)
Window function	Hanning (impulse length: 50)
Sequence length	20000 symbols
No. of samples (A/D card)	500000

Table 3.2. Summary of fixed parameter values

20000 symbols at a rate of 10 kHz sampled at 250 kHz produce a 2 seconds long sequence consisting of 500000 samples. The SMIQ signal generator receives the baseband signal from WinIQSIM using the GPIB protocol. SMIQ then generates a 100 MHz RF signal corresponding to the received signal and with a power level of –25 dBm.

3.3 Receiver Stage

At the receiver the signal is down-converted to an intermediate frequency (IF) of 21.4 MHz/ -14 dBm. The signal is then amplified with a gain of approximately 24 dB before it arrives at a 3 dB splitter. The power of the signal at the A/D converter and the spectrum analyser is thus approximately +7 dBm. The power level is chosen in order to utilise the full dynamic range of the A/D card (± 32768). The clock signal of the A/D card is 64.8 MHz. With a decimation factor of 256, the I/Q sample frequency is 253125 Hz (64.8 MHz/256). The A/D card has been set up with a numerical controlled oscillator (NCO) frequency of 21.35 MHz, which gives an effective centre frequency of 50 kHz (21.4 MHz – 21.35 MHz) on the digitised signal. We choose to take a sequence of 500000 samples. With the 253125 Hz sample rate of the A/D card this becomes a segment of 1.975 seconds length (500000 samples /253125 Hz). These segments are then stored to files and given names to reflect the modulation type and the Eb/N_0 value. The total file size is approximately 6.8 MB. A signal sample file can now be created using the approach described in the following chapter.

4 RADIO SIGNAL SAMPLES FILE FORMAT

To facilitate fair comparisons and data exchange across different platforms we propose a generic signal sample file format and develop an application to create files of this format.

4.1 General Structure

A file consists of radio communication signal samples with a given modulation format and given signal distortion properties (additive noise, fading, multi-path etc.). It is convenient that the file names consist of the modulation name and the noise level, and that other design specifications are indicated in the folder structures.

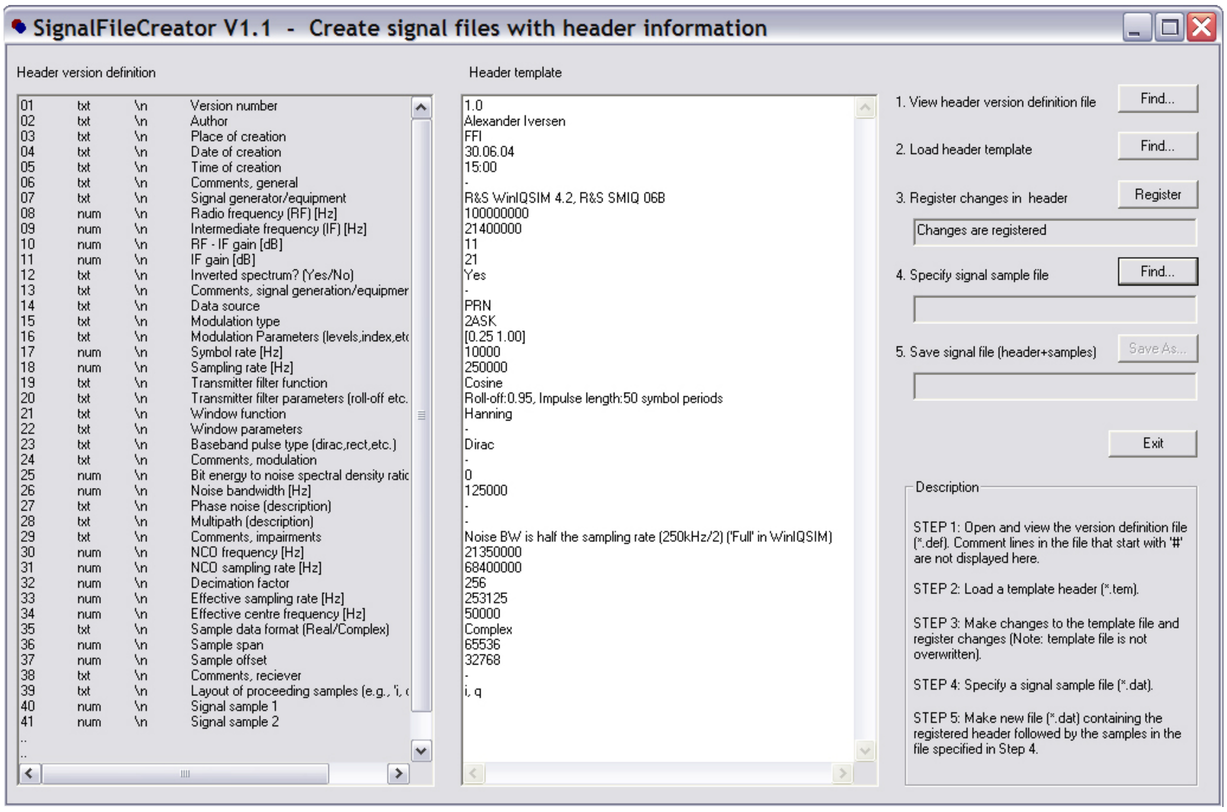
The files are in ASCII format. This format is readable and thus convenient for manual file inspection and is easy to import into other software applications. It is not optimal with respect to size, but the availability and price of data storage makes this a minor problem. The file size can, of course, be reduced using standard file compression programs.

Each file consists of a *header* that contains information about the samples, and a *body* that contains the signal samples. A separate and unalterable *version definition file* defines the structure of the sample file, that is, the number of fields in the header, the field delimiter and the contents. The header fields contain general information such as version number, date and author and, when available, information about the signal generation process, filtering, modulation type, distortion properties and receiver properties. The availability of this information will allow for fairer comparisons as one can take account of signal generation design choices etc.

Appendix A shows the version 1.0 definition file that defines the file structure used for the signal generation in this work, and an example of a portion of a complete signal file.

4.2 Software Application: SignalFileCreator

An application has been developed that lets the user easily create files with header and samples according to a version definition file. The graphical user interface is shown in Figure 4.1.



SignalFileCreator V1.1 - Create signal files with header information

Header version definition

```
01     txt     \n     Version number
02     txt     \n     Author
03     txt     \n     Place of creation
04     txt     \n     Date of creation
05     txt     \n     Time of creation
06     txt     \n     Comments, general
07     txt     \n     Signal generator/equipment
08     num     \n     Radio frequency (RF) [Hz]
09     num     \n     Intermediate frequency (IF) [Hz]
10     num     \n     RF - IF gain [dB]
11     num     \n     IF gain [dB]
12     txt     \n     Inverted spectrum? (Yes/No)
13     txt     \n     Comments, signal generation/equipmer
14     txt     \n     Data source
15     txt     \n     Modulation type
16     txt     \n     Modulation Parameters (levels,index,etc
17     num     \n     Symbol rate [Hz]
18     num     \n     Sampling rate [Hz]
19     txt     \n     Transmitter filter function
20     txt     \n     Transmitter filter parameters (roll-off etc.
21     txt     \n     Window function
22     txt     \n     Window parameters
23     txt     \n     Baseband pulse type (dirac,rect,etc.)
24     txt     \n     Comments, modulation
25     num     \n     Bit energy to noise spectral density ratic
26     num     \n     Noise bandwidth [Hz]
27     txt     \n     Phase noise (description)
28     txt     \n     Multipath (description)
29     txt     \n     Comments, impairments
30     num     \n     NCO frequency [Hz]
31     num     \n     NCO sampling rate [Hz]
32     num     \n     Decimation factor
33     num     \n     Effective sampling rate [Hz]
34     num     \n     Effective centre frequency [Hz]
35     txt     \n     Sample data format (Real/Complex)
36     num     \n     Sample span
37     num     \n     Sample offset
38     txt     \n     Comments, reciever
39     txt     \n     Layout of proceeding samples (e.g., 'i, c
40     num     \n     Signal sample 1
41     num     \n     Signal sample 2
..
..
```

Header template

```
1.0
Alexander Iversen
FFI
30.06.04
15:00
-
R&S WinIQSIM 4.2, R&S SMIQ 06B
100000000
21400000
11
21
Yes
-
PRN
2ASK
[0.25 1.00]
10000
250000
Cosine
Roll-off:0.95, Impulse length:50 symbol periods
Hanning
-
Dirac
-
0
125000
-
-
Noise BW is half the sampling rate (250kHz/2) ('Full' in WinIQSIM)
21350000
68400000
256
253125
50000
Complex
65536
32768
-
i, q
```

1. View header version definition file

2. Load header template

3. Register changes in header

Changes are registered

4. Specify signal sample file

5. Save signal file (header+samples)

Description

STEP 1: Open and view the version definition file (*.def). Comment lines in the file that start with '#' are not displayed here.

STEP 2: Load a template header (*.tem).

STEP 3: Make changes to the template file and register changes (Note: template file is not overwritten).

STEP 4: Specify a signal sample file (*.dat).

STEP 5: Make new file (*.dat) containing the registered header followed by the samples in the file specified in Step 4.

Figure 4.1. Screenshot of SignalFileCreator

The SignalFileCreator requires three files:

- Version definition file (*.def). This file contains information on the number of fields and a description of the field contents.
- Header template file (*.tem). This file contains a header template that the user defines in advance, which is specific to the signal generation. The user can then make the required changes to specific fields in the template and avoid rewriting all the static header information for each file.
- Signal sample file (*.dat). This is the file that contains only the raw samples.

The SignalFileCreator works as follows:

- Open and view the version definition file. The file contents are viewed in the leftmost window. Comment lines in the file that starts with "#" are not displayed.
- Load the header template file. The contents are displayed in the window next to the version definition contents. The fields in the two windows are aligned. Thus, the display of the version definition file is just to make it easier for the user to see what information each field requires.
- Make the required changes to the header template file and register changes using the "register" button. These changes are only stored in memory, thus no changes are stored back to the template file.
- Specify the location of the signal sample file. A handle to this file is stored.
- Specify a file name for the storage of the new file that contains the header and the samples.

When the above steps are completed, the application streams the header contents followed by the samples to the new file. The first 200 samples in the sample file are ignored to avoid transient fluctuations at the start of the signal generation (made by the signal generation equipment). It is important to note that the new file cannot overwrite the original sample file.

5 SEGMENTATION AND BANDWIDTH ESTIMATION

Classification is based on signal segments much smaller than the 500000-samples segment. To be comparable with similar AMR studies (e.g., (7), (9)), it is chosen to operate with 2048-sample segments. One such segment is approximately 8 ms (2048 samples/ 253125 Hz) and will contain approximately 80 symbols (0.008 s * 10000 Hz), which is believed to be adequate for bandwidth estimation and feature extraction.

The bandwidth is estimated based on an analysis of the spectral power density of each 2048-samples segment. It works as follows: Assume we have a symmetric power spectrum plot *P* with length *N* and where the centre frequency is:

$$C = \left\lceil \frac{N}{2} \right\rceil \tag{5.1}$$

Then we compute the integral *S* of *P* by starting at *C* and working outwards alternating on both sides. If initially $S_1 = P_C$, then:

$$S_{2i-2} = S_{2i-3} + P_{C+(i-1)} \text{, and } S_{2i-1} = S_{2i-2} + P_{C-(i-1)} \text{, for all } i = 2, 3, \ldots, C \tag{5.2}$$

A typical *P* curve will have crests where the signal is located and drop sharply down to the noise floor and level out. The *S* curve will consequently rise with a variable slope and level out with a constant slope when *P* contains no signal and only noise. We now take the derivative *S'* of *S*. *S'* will always be positive but the curve will typically have two or three plateaux (depending on the modulation type), the lowest of which will represent only noise and no signal. The double derivative, *S''*, will be most negative at the transition to the lowest plateau of the S' curve and approach zero when the plateau is reached. If i_{min} is the point at the minimum *S''*, *T* is a threshold that indicates that *S''* approaches zero and *A* is a constant, the estimated bandwidth, *W*, is chosen thus:

$$W = i + A \text{, where } i > i_{\min} \text{ and } S''_i > T \tag{5.3}$$

Using this approach, the bandwidths of the generated signals have been estimated. The results are presented in Figure 5.1.

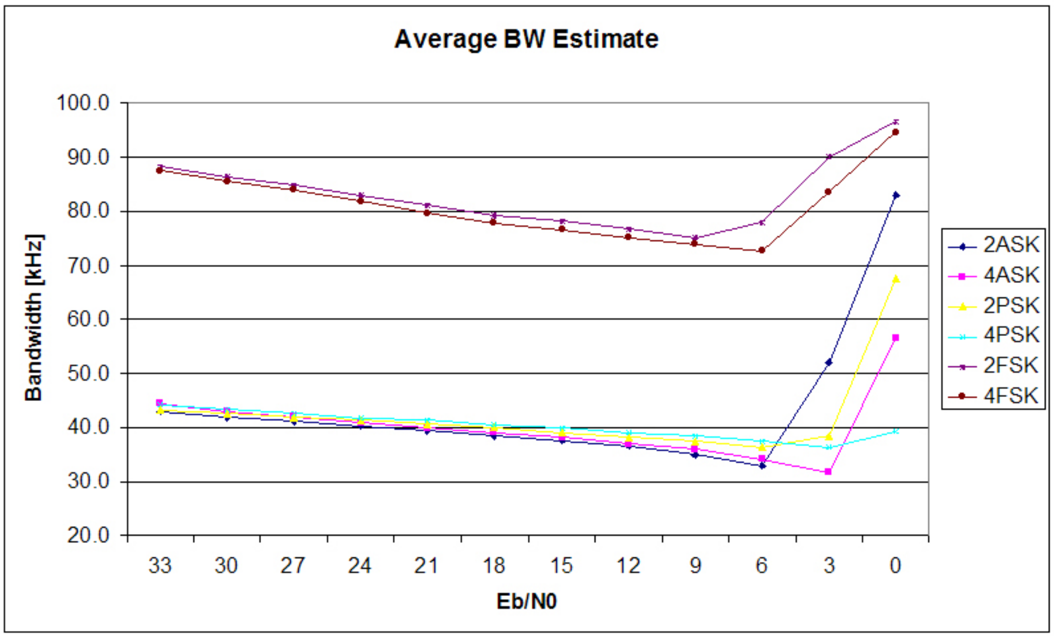


Figure 5.1. Bandwidth estimates of received signals.

Each point of the curve is the average bandwidth estimate of 243 2048-samples segments. The figure shows that the estimates are reasonable down to a noise level of about 6 dB Eb/N_0. With the bandwidth now known, signal-to-noise ratio (SNR) can be calculated according to equation 3.5. The SNR levels are presented in Table 5.1. The values outside the brackets are the true SNR calculations. The bracketed values are rough, generalised estimates, which we refer to when stating SNR levels in the remainder of this report.

Eb/N_0	**Modulation Type**					
	2ASK	**4ASK**	**2PSK**	**4PSK**	**2FSK**	**4FSK**
33	26.7 (27)	29.5 (30)	26.6 (27)	29.6 (30)	23.5 (24)	26.6 (27)
30	23.8 (24)	26.7 (27)	23.7 (24)	26.6 (27)	20.6 (21)	23.7 (24)
27	20.9 (21)	23.8 (24)	20.8 (21)	23.7 (24)	17.7 (18)	20.8 (21)
24	18.0 (18)	20.9 (21)	17.9 (18)	20.8 (21)	14.8 (15)	17.9 (18)
21	15.1 (15)	18.0 (18)	14.9 (15)	17.9 (18)	11.9 (12)	15.0 (15)
18	12.2 (12)	15.1 (15)	12.0 (12)	14.9 (15)	9.0 (9)	12.1 (12)
15	9.3 (9)	12.2 (12)	9.1 (9)	12.0 (12)	6.1 (6)	9.2 (9)
12	6.4 (6)	9.3 (9)	6.2 (6)	9.1 (9)	3.2 (3)	6.3 (6)
9	3.6 (3)	6.5 (6)	3.3 (3)	6.2 (6)	0.3 (0)	3.3 (3)
6	0.9 (0)	3.7 (3)	0.4 (0)	3.3 (3)	-2.9 (-3)	0.4 (0)
3	-3.4 (-3)	1.0 (0)	-2.7 (-3)	0.4 (0)	-6.5(-6)	-3.1 (-3)
0	-8.7 (-6)	-3.7 (-3)	-7.6(-6)	-2.7 (-3)	-9.8 (-9)	-6.7(-6)

Table 5.1. Calculated SNR for the six modulation types (generalised estimates in brackets).

5.1 Bandwidth and SNR of Unknown Signals

The bandwidth estimates and SNR of the MSK and 16QAM signals are presented in Figure 5.2.

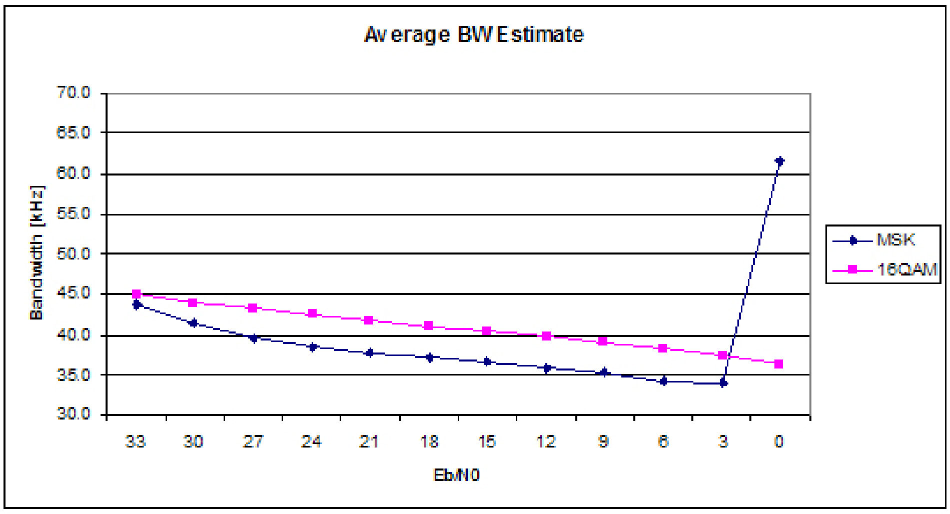


Figure 5.2. Bandwidth estimates of received signals.

Similar to the ASK, PSK and FSK signals, the bandwidth estimation of the MSK and 16QAM signals seems reasonable down to a low Eb/N_0 level. The SNRs are presented in Table 5.2.

	Modulation Type	
Eb/N_0	**MSK**	**16QAM**
33	26.6 (27)	32.5 (33)
30	23.8 (24)	29.6 (30)
27	21.0 (21)	26.7 (27)
24	18.2 (18)	23.7 (24)
21	15.2 (15)	20.8 (21)
18	12.3 (12)	17.9 (18)
15	9.4 (9)	15.0 (15)
12	6.5 (6)	12.0 (12)
9	3.5 (3)	9.1 (9)
6	0.7 (0)	6.2 (6)
3	-2.3 (-3)	3.3 (3)
0	-7.2 (-6)	0.4 (0)

Table 5.2. Calculated SNR for the MSK and 16QAM (generalised estimates in brackets).

6 FEATURE EXTRACTION

Features must be chosen on the basis of what modulation schemes the classifier should handle. Features extracted from the instantaneous attributes (amplitude, phase and frequency) have been successfully used in the past (7), (9), (11) for the classification of low-level ASK-, PSK- and FSK signals. Features used in this work are shown in Table 6.1. These are extracted from signal segments that have been filtered using the bandwidth estimates described in chapter 5.

No.	Feature Name	Description
1	γ_{max}	Maximum power spectral density of normalised-centred instantaneous amplitude (7).
2	σ_{ap}	Standard deviation of the absolute value of the centred instantaneous phase, evaluated over the non-weak intervals[3] of the signal segment (7).
3	σ_{dp}	Standard deviation of the direct (not absolute) value of the centred instantaneous phase, evaluated over the non-weak intervals of the signal segment (7).
4	σ_{aa}	Standard deviation of the absolute value of the normalised-centred instantaneous amplitude (7).
5	σ_{af}	Standard deviation of the absolute value of the normalised-centred instantaneous frequency (9).
6	σ_{da}	Standard deviation of the direct (not absolute) value of the normalised-centred instantaneous amplitude (9).
7	σ_{df}	Standard deviation of the direct (not absolute) value of the normalised-centred instantaneous frequency (9).
8	γ_{maxf}	Maximum power spectral density of normalised-centred instantaneous frequency (9).

Table 6.1. Key Features

6.1 Feature Analysis

In the transition of symbols, the instantaneous attributes have a tendency to contain spikes, which may have a significant effect on the features. To reduce this effect, one can use a median filter window that moves over the instantaneous attributes. The eight features in Table 6.1 have been extracted from the signal segments with SNR ranging from -3 dB to 24 dB and the effects with and without the median filter have been observed. Figure 6.1 shows the feature values when the median filter has not been applied to the instantaneous attributes. Figure 6.2 illustrates the same when a median filter has been applied. Not surprisingly, the feature values tend to come together as the SNR level is reduced for both approaches. However, we clearly see that most of the features separate modulation schemes more clearly when a median filter has been applied. The variations of the feature values, which are displayed as the 10th and 90th percentiles of 243 signal segments, are in both the non-filtered and filtered cases reasonably small.

[3] Non-weak intervals are the samples where the instantaneous amplitude is above a certain threshold.

Based on these results, we choose to use a median filter. The eight features will serve as inputs to the classifiers that are covered in the next chapter.

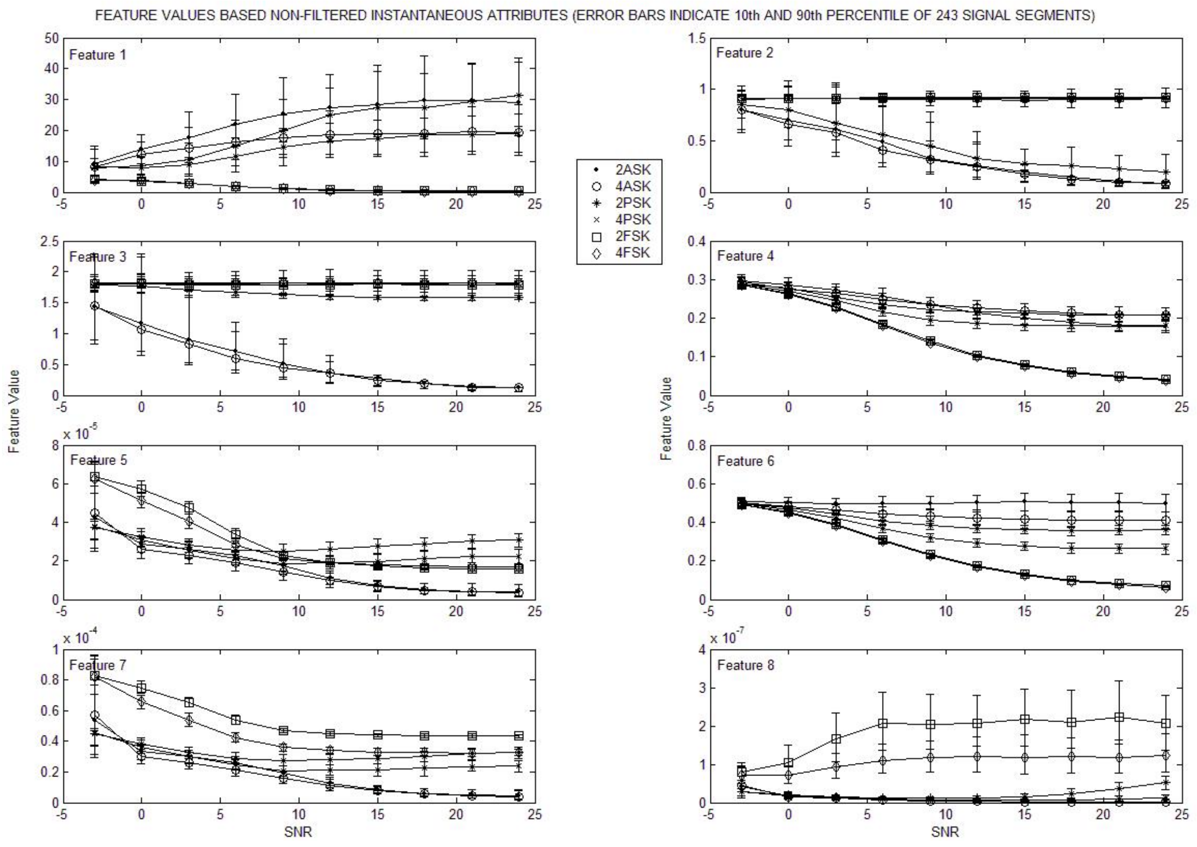


Figure 6.1. Plot of features extracted from non-filtered instantaneous attributes.

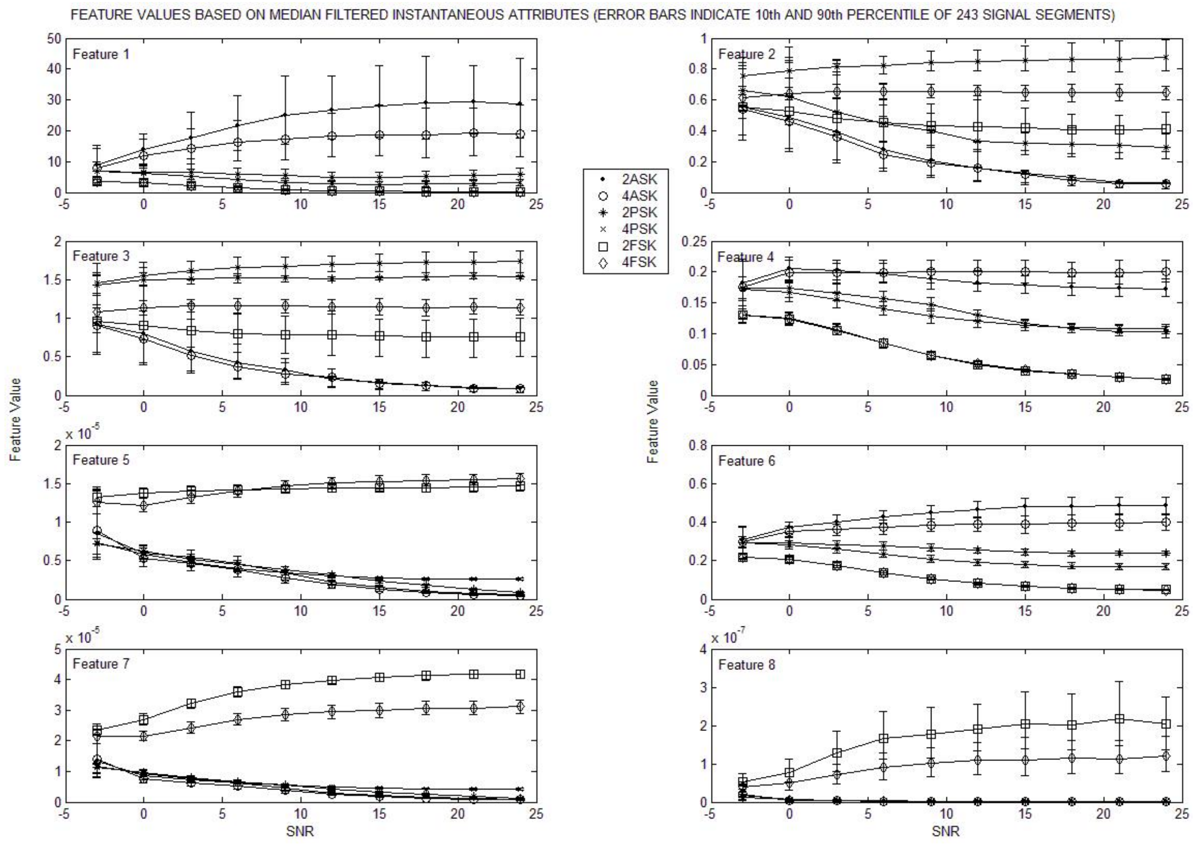


Figure 6.2. Plot of features extracted from median-filtered instantaneous attributes.

7 CLASSIFIER DESIGN

Studies of the classification of modulation types similar to the ones considered here have shown that Multi-layer Perceptrons (MLPs) are well suited for the task (7), (9), (10), (22). However, we are not familiar with any work that investigates the MLPs' abilities to reject signals that they have not been trained to classify. How will, for example, a classifier trained on the six modulation types we consider react to a QAM or a MSK signal?

In this chapter, we develop a standard MLP classifier and investigate both its classification abilities and its behaviour to unknown signal types. We also investigate the potential of auto-association MLPs.

7.1 Standard MLP Classifier

7.1.1 Classifier Structure

The MLP classifier has the following specifications (Table 7.1):

Inputs	8 inputs representing the 8 features in Table 6.1
Hidden layers	2 hidden layers with 6 nodes in each layer
Outputs	6 nodes, each representing one of the 6 modulation types
Activation functions	Tan-sigmoid in hidden layers, log-sigmoid in output layer
Target activation values	0.9 for active outputs, 0.1 for inactive outputs
Training algorithm	Resilient back-propagation with validation set
Pre-processing	Normalise network inputs (features) so that they have zero mean and unity standard deviation

Table 7.1. MLP specifications

The MLP classifier consists of eight inputs, one for each feature, two hidden layers with six nodes each and an output layer with six nodes representing the six modulation types. A network structure consisting of two hidden layers has in comparative studies shown to be advantageous, e.g. (10). Some of the specifications in Table 7.1 are performed to increase the learning speed of the network. The tan-sigmoid is anti-symmetric around zero and creates a better balance of weights than the log-sigmoid. This normally leads to faster convergence. A log-sigmoid was chosen at the output nodes as it produces convenient values in the range 0 to 1. However, the target values, which are used during the supervised training, are offset from [0, 1] to [0.1, 0.9]. The offset targets will be reached faster and hence improve the training speed. As can be seen in Figure 6.2, the range of the features varies greatly. E.g., feature 1 ranges from 0 to 40, whereas feature 8 ranges from 0 to $3*10^{-7}$. During training, this difference of resolution will cause conflicting weight adjustments. Features are therefore normalised before they are fed into the network. The normalisation parameters are based on the training data but must also be used to normalise all consecutive inputs to the network after training is completed.

7.1.2 Training Results

The network was trained using the resilient back-propagation algorithm with early stopping based on the validation set error. From Figure 6.2 we see that some features at as low as 0 dB SNR still show a distinct difference between certain modulation types. We train the classifier on signals in the range 3 dB SNR to 24 SNR with 3 dB intervals (8 levels). At each level, we use 50 segments. Thus, with the 6 modulation types the complete training set consists of 6*50*8 = 2400 examples. Each example consists of 8 features extracted according to the description in Chapter 6. Similarly, the validation set consists of 2400 different examples. The training result is presented in Table 7.2 and was obtained on the first training attempt.

Epochs	370
Training set error (MSE)	0.0045
Validation set error (MSE)	0.0045
Training time[4] (seconds)	63.7

Table 7.2. Training results

Training was stopped after 370 epochs when the validation set error was at a minimum. Note that the training time (in seconds) is highly system dependent and is shown only to give an indication on the elapsed time.

7.1.3 Classification Results

The classifier was tested on signal segments ranging from –3 dB to 24 dB SNR with 3 dB intervals (10 levels). At each level, we use 143 segments. Thus, with the 6 modulation types the complete test set consists of 6*143*10 = 8580 examples. We perform separate tests on each noise level and obtain the results presented in Figure 7.1.

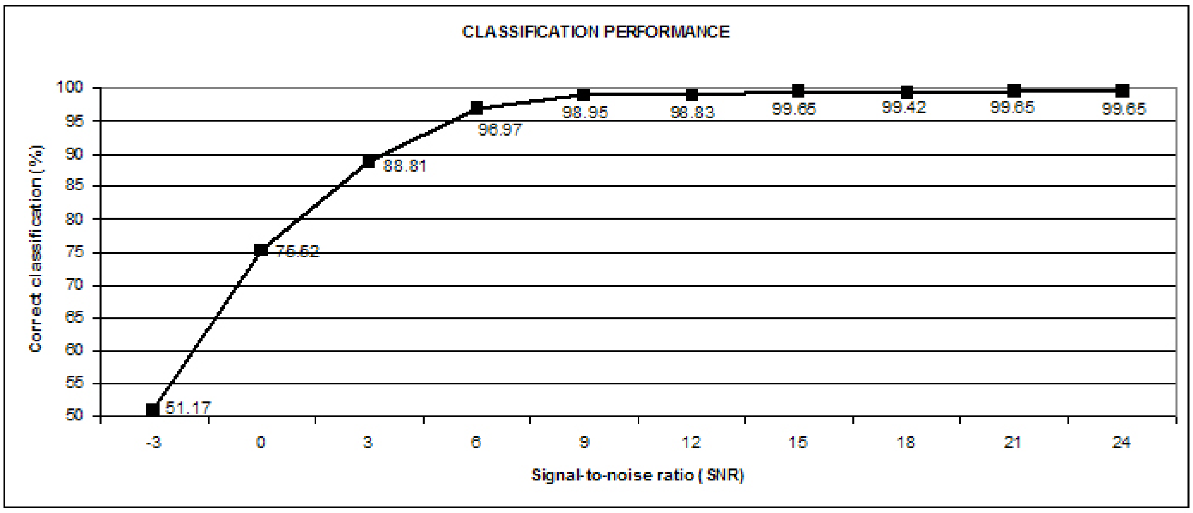


Figure 7.1. Test results.

We see that the classification rate is about 89 % for SNR of 3 dB and close to 100 % for signal segments with SNR of 6 dB and above. At lower SNR levels, the performance drops off significantly, to about 50 % at –3 dB SNR. This drop is expected because of the reduction of the features' discriminatory qualities and the fact that the network is trained only on signals of 3 dB SNR and above. Focusing on test signals of 3 dB SNR and above, we can see what modulation types the classifier mixes up. The confusion matrix presented in Table 7.3 shows that the classifier mainly confuses the 2- and 4-level types of the same modulation type. This is also expected, as their feature values are generally closer to each other (Figure 6.2) than to other modulation types.

[4] Measured on the Simulation PC (Table 3.1).

Actual type	Predicted type (%)					
	2ASK	**4ASK**	**2PSK**	**4PSK**	**2FSK**	**4FSK**
2ASK	96.15	3.76	0.09	0.00	0.00	0.00
4ASK	3.41	96.07	0.17	0.17	0.17	0.00
2PSK	0.00	0.00	96.94	3.06	0.00	0.00
4PSK	0.00	0.00	2.10	97.90	0.00	0.00
2FSK	0.00	0.00	0.00	0.00	99.39	0.61
4FSK	0.00	0.00	0.00	0.00	0.00	100.00

Table 7.3. Confusion matrix for signals of 3 dB and above.

7.1.4 Behaviour to Unknown (Novel) Signals

Especially in a non-co-operative AMR setting, there is no guarantee that the modulation of a received signal is of a type that the classifier is trained on. It is therefore of interest to know how the classifier behaves when presented with such novel signals. E.g., the classifier under consideration has been trained on 2ASK, 4ASK, 2PSK, 4PSK, 2FSK and 4FSK signals. When tested on signals of those types it performs very well. We then present MSK or 16QAM signal segments to the classifier. The ideal situation would be a classifier that produces zero on all outputs to indicate that the input does not match any of the six known modulation types. An alternative approach is to make use of the classification confidence: If, for example, the highest output is 0.99 we might assume that the classifier is very confident. A highest output of 0.5 on the other hand might indicate that the classifier is less confident of the classification (which might indicate an unknown modulation format). Based on this assumption we can set a threshold below which the input is rejected as the output indicates too much uncertainty.

To see if this approach is feasible, we investigate this classification certainty in order to see if unknown signal formats produce low outputs whereas known formats produce high outputs. The distribution of the magnitudes of the highest output for the various signal formats are presented in Figure 7.2 and are based on 1144 test examples for each modulation type in the range 3-24 dB SNR.

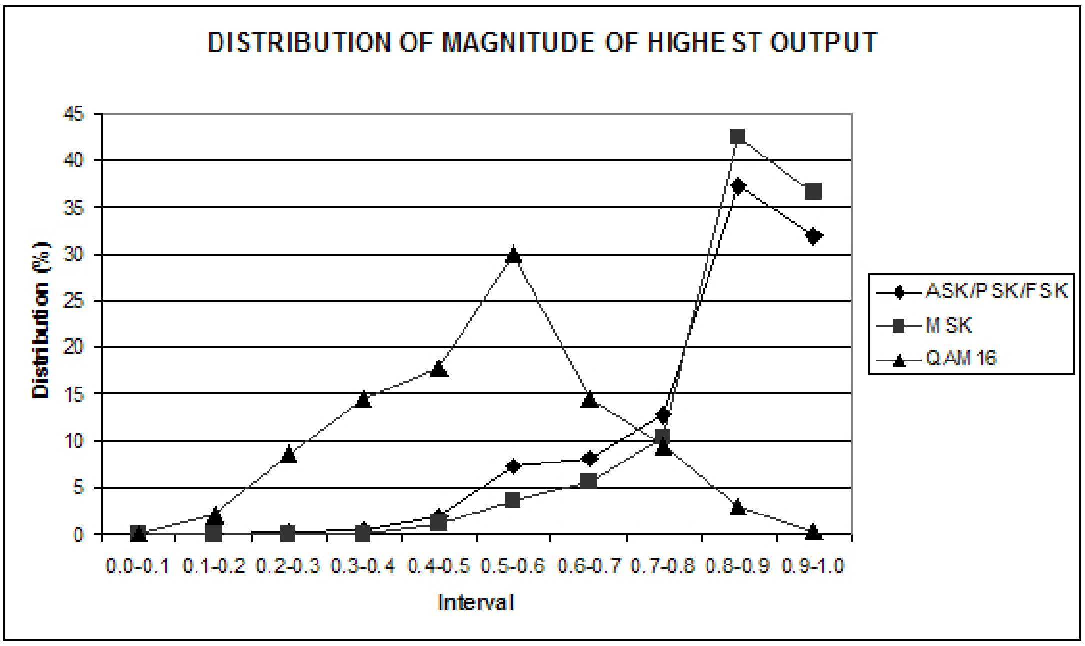


Figure 7.2. Magnitude distribution of the highest output for known and unknown signals.

When the classifier is presented with 16QAM signals, the highest output is usually lower than that of known signals. In this case, it could be possible to set a threshold, below which we reject the input. The MSK signals, however, cause outputs that are very similar to the known signals. In this case it is impossible to tell a known modulation type from an unknown one.

Yet another approach for detecting novelty is possible. One of the findings in (22) was that the difference between the two highest outputs in a correct classification was higher than that of wrong classifications. Figure 7.3 shows the average difference between the two highest outputs of the known and novel examples.

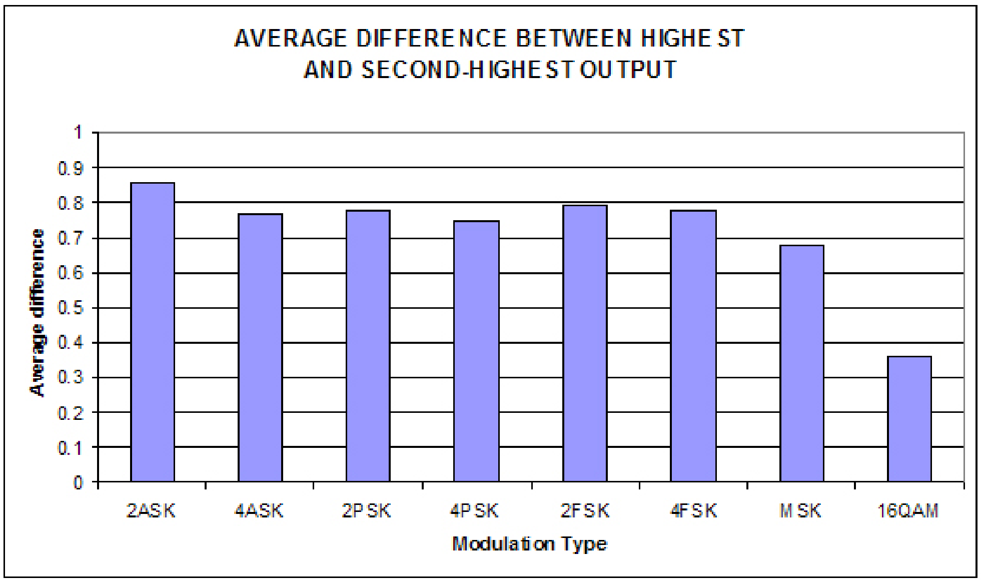


Figure 7.3. Average difference between the two highest outputs.

The figure indicates that the average difference for MSK signals is slightly lower than that of the known signal types. 16QAM has a noticeably smaller difference. The distribution of the highest output magnitude and the output differences are used below to investigate the MLP's ability to detect/reject the unknown signal types (MSK and 16QAM).

From Figure 7.3 we see that by only accepting outputs of more than 0.75 and rejecting outputs below that threshold, we might be able to reject at least some of the novel examples (16QAM). The result of this thresholding is shown in Table 7.4.

	Predicted type (%)						
Actual type	**2ASK**	**4ASK**	**2PSK**	**4PSK**	**2FSK**	**4FSK**	**Novelty**
2ASK	89.51	0.52	0.00	0.00	0.00	0.00	9.97
4ASK	1.14	89.42	0.00	0.00	0.17	0.00	9.27
2PSK	0.00	0.00	92.22	0.79	0.00	0.00	6.99
4PSK	0.00	0.00	0.00	92.48	0.00	0.00	7.52
2FSK	0.00	0.00	0.00	0.00	99.21	0.52	0.26
4FSK	0.00	0.00	0.00	0.00	0.00	99.91	0.09
MSK	0.00	0.00	0.00	84.00	0.00	0.35	15.65
16QAM	8.39	0.26	0.00	3.93	0.00	0.00	87.41

Table 7.4. Confusion matrix for known and novel signals with highest-output thresholding.

We see that 87.41 % of the 16QAM signals have been rejected, whereas only 15.65 % of the MSK-signals have. We also make two other observations: There is a reduced mix-up between the known modulation types but also a reduced rate of correct classification (compared to Table 7.3). The latter is the inevitable cost of the constraints that are introduced.

Figure 7.3 indicates that a difference-threshold of 0.7 could separate novel from known examples. Results are shown in Table 7.5.

	Predicted type (%)						
Actual type	**2ASK**	**4ASK**	**2PSK**	**4PSK**	**2FSK**	**4FSK**	**Novelty**
2ASK	91.26	0.17	0.00	0.00	0.00	0.00	8.57
4ASK	1.40	77.97	0.00	0.00	0.00	0.00	20.63
2PSK	0.00	0.00	92.05	0.44	0.00	0.00	7.52
4PSK	0.00	0.00	0.00	85.84	0.00	0.00	14.16
2FSK	0.00	0.00	0.00	0.00	98.95	0.35	0.70
4FSK	0.00	0.00	0.00	0.00	0.00	95.46	4.55
MSK	0.00	0.00	0.00	46.42	0.00	0.00	53.58
16QAM	9.18	0.09	0.00	1.05	0.00	0.00	89.69

Table 7.5. Confusion matrix for known and novel signals with output difference thresholding.

This approach manages to reject more of the unknown signals on the cost of more known signal types being misclassified as unknown.

An important aspect is that the above analyses were based on the test data, containing both known and "unknown" signal types. In reality we have no a priori knowledge of the nature of the unknown data and there is no guarantee that they will have properties such as those shown here. Classifier design should therefore be based only on the obtained and known training data. This challenge will be handled in the remaining part of this report.

7.2 Auto-association MLP

To investigate the properties of the auto-association MLP (AaMLP) we build six networks that are trained on each of the six known modulation schemes: 2ASK, 4ASK, 2PSK, 4PSK, 2FSK and 4FSK. The AaMLPs have the following specifications (Table 7.6):

Inputs	8 inputs representing the 8 features in Table 6.1
Hidden layers	1 hidden layers with 5 nodes
Outputs	8 nodes, corresponding to the 8 inputs
Activation functions	Tan-sigmoid in hidden layer, linear function in output layer
Target activation values	Target = Input
Training algorithm	Resilient back-propagation with validation set
Pre-processing	Normalise network inputs (features) so that they have zero mean and unity standard deviation

Table 7.6. AaMLP specifications

Each network is trained to reproduce the input at the output pattern. The error that is attempted reduced during training is thus the mean of the square of the differences between the input and the output values. Apart from the target values, the AaMLP operates in the same manner as the classifier in Section 7.1.

7.2.1 Training Results

Each network is trained on signals of one modulation type with SNR ranging from 3 to 24 dB SNR with 3 dB intervals (8 levels). At each level, we have 50 signal segments. Thus, the complete training set consists of 50*8 = 400 examples. We name the six networks after the modulation types they are trained on and achieve the following training results (Table 7.7):

AaMLP-2ASK		**AaMLP-4ASK**	
Epochs	688	Epochs	1509
Training set error (MSE)	0.0318	Training set error (MSE)	0.0318
Validation set error (MSE)	0.0359	Validation set error (MSE)	0.0303
Training time[5] (seconds)	39.247	Training time (seconds)	81.577
AaMLP-2PSK		**AaMLP-4PSK**	
Epochs	278	Epochs	951
Training set error (MSE)	0.03	Training set error (MSE)	0.0245
Validation set error (MSE)	0.039	Validation set error (MSE)	0.0262
Training time (seconds)	14.681	Training time (seconds)	47.398
AaMLP-2FSK		**AaMLP-4FSK**	
Epochs	1866	Epochs	486
Training set error (MSE)	0.0124	Training set error (MSE)	0.0251
Validation set error (MSE)	0.0129	Validation set error (MSE)	0.0282
Training time (seconds)	104.25	Training time (seconds)	20.379

Table 7.7. AaMLP training results.

An important parameter is the networks' training set error. This parameter tells us what MSE can be expected when the network is presented to a signal similar to the one it is trained to reproduce. E.g., if the AaMLP-4PSK is presented with an unknown signal segment, and the MSE is 0.0250, we can assume that the signal is a 4PSK signal. If the MSE is, say, 0.80, we can assume that the signal is not a 4PSK.

7.2.2 AaMLP Performance

The six AaMLP networks are tested on signals of 8 modulation types. The results indicate that the networks are able to discriminate between known and unknown modulation types. Figure 7.4 shows the results from AaMLP-2ASK, which is trained on 2ASK signals. The network produces a markedly lower MSE on 2ASK signals than other signals.

[5] Measured on the Simulation PC (Table 3.1).

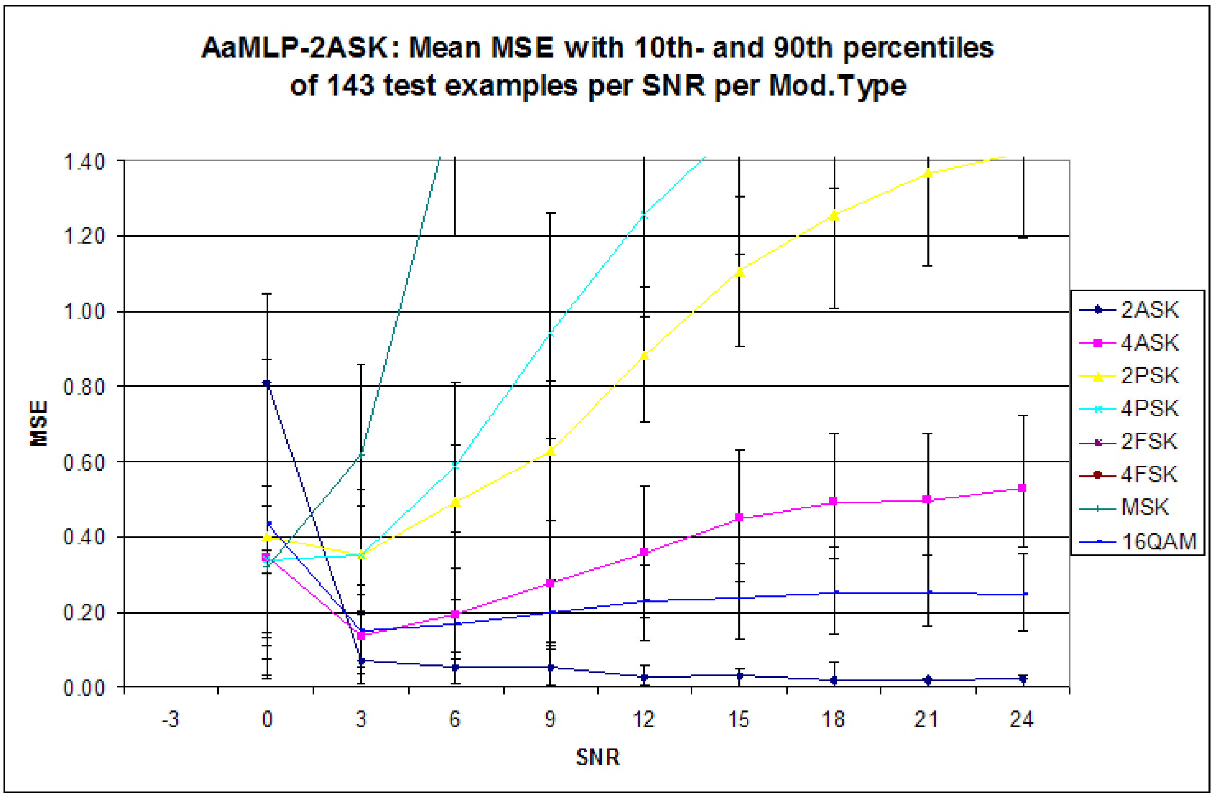


Figure 7.4. AaMLP-2ASK mean-squared error.

Figure 7.5 shows the results from AaMLP-4ASK. Here, 16QAM signal give a MSE that is fairly close to that of 4ASK signals. However, when at the closest, the MSE of 16QAM are still more than 2.6 times larger than the MSE of 4ASK signals.

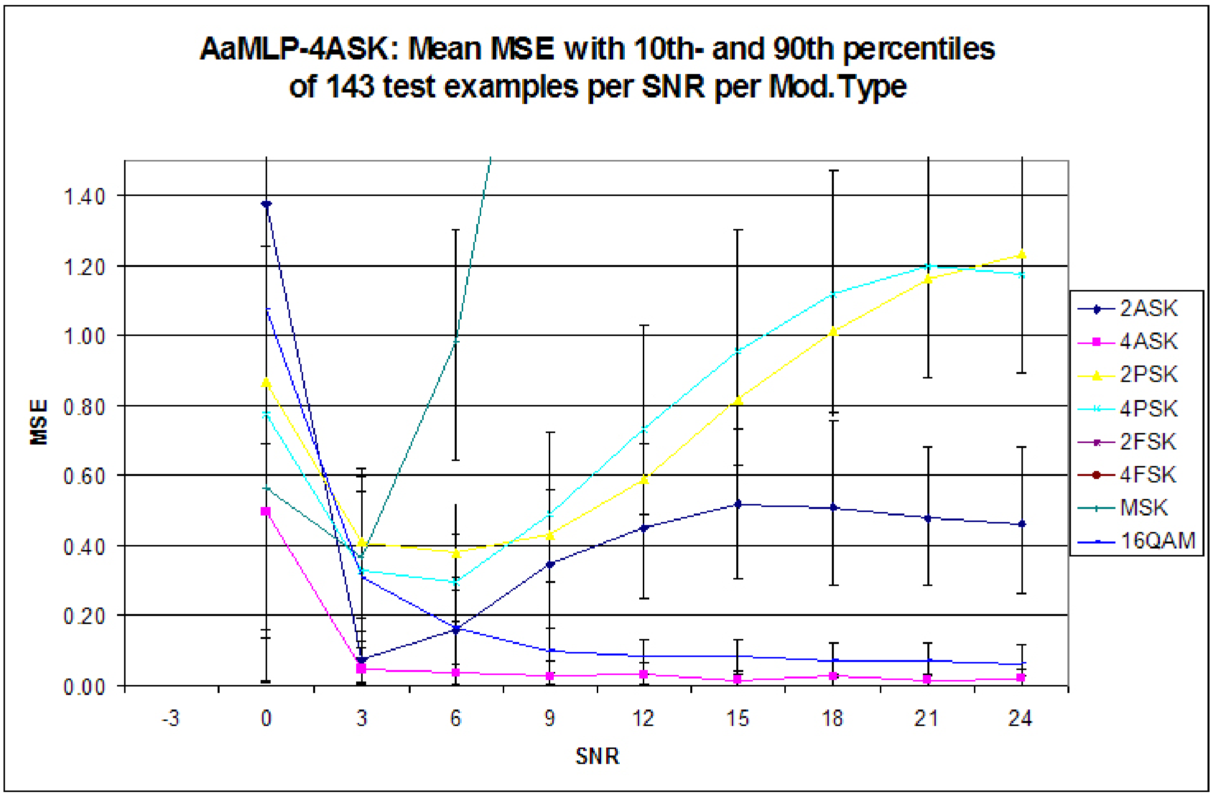


Figure 7.5. AaMLP-4ASK errors.

Figures 7.6 to 7.9 display the results of AaMLP-2PSK, AaMLP-4PSK, AaMLP-2FSK and AaMLP-4FSK respectively. All show favourable results down to a SNR level of about 3 dB (6 dB for AaMLP-4PSK).

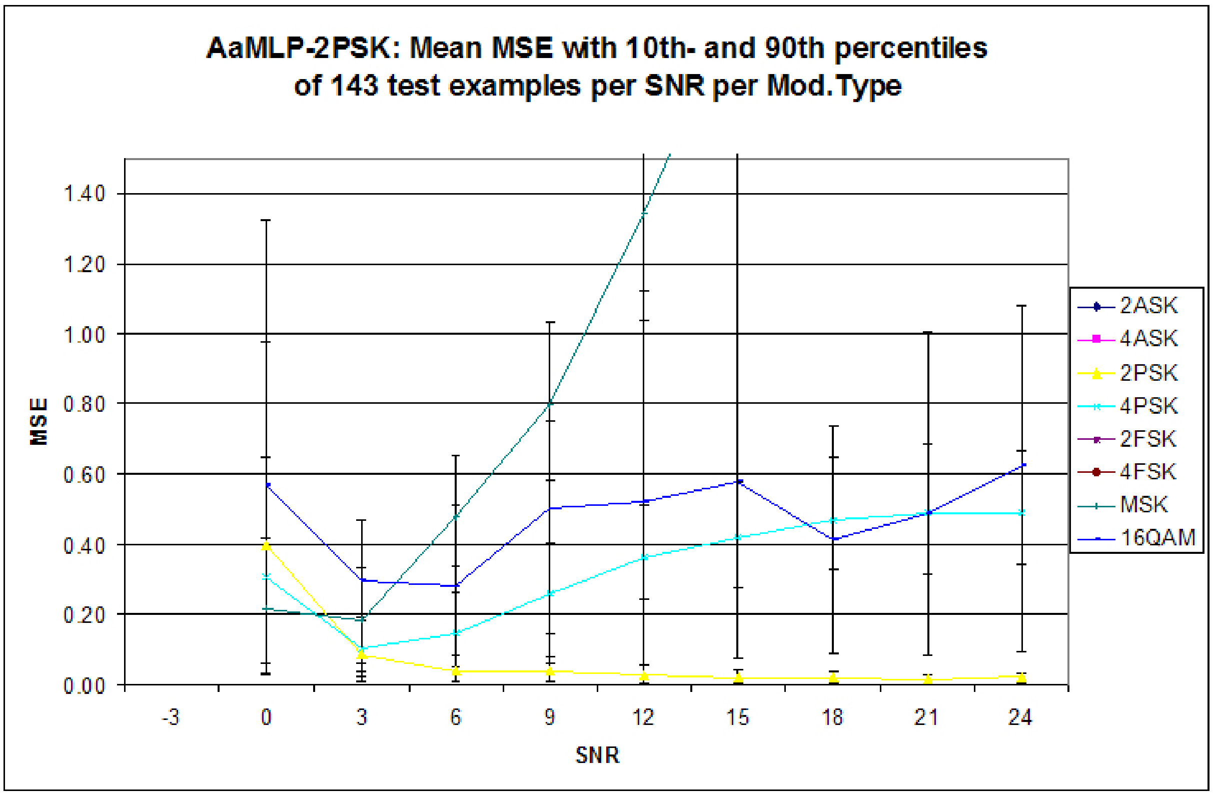


Figure 7.6. AaMLP-2PSK errors.

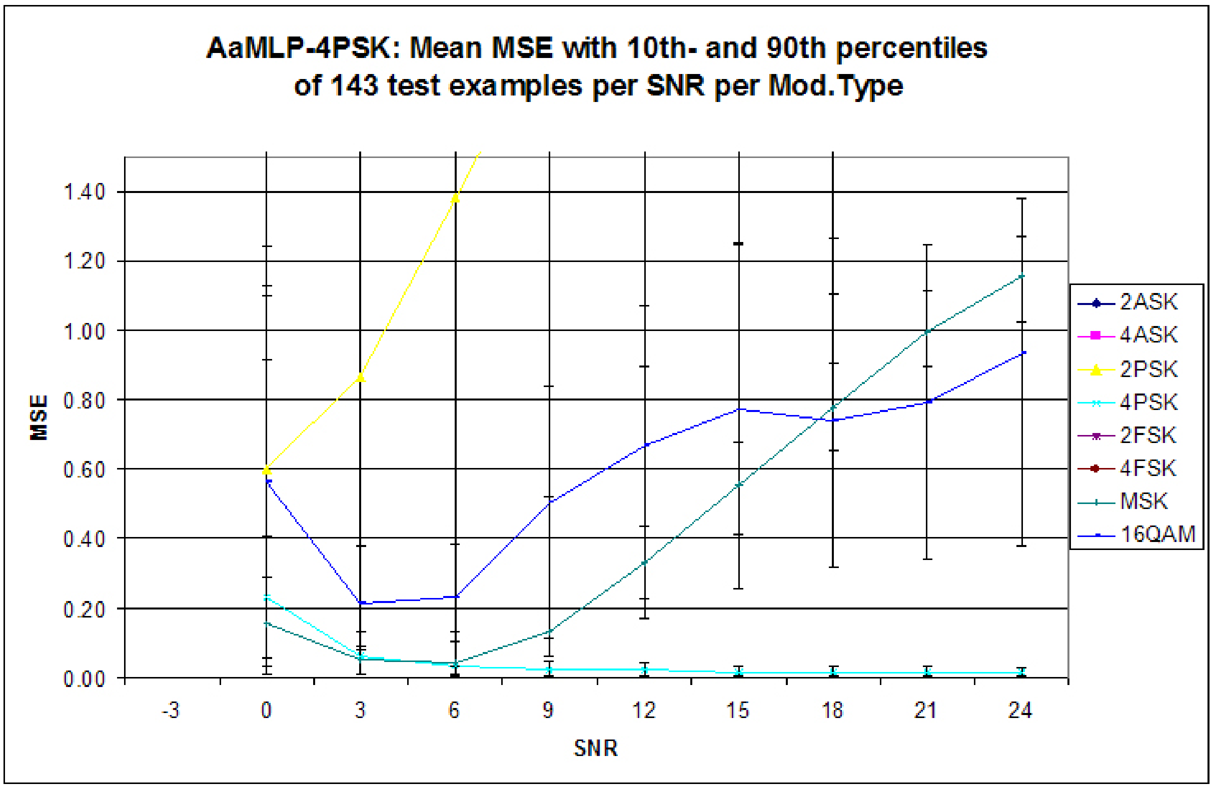


Figure 7.7. AaMLP-4PSK errors.

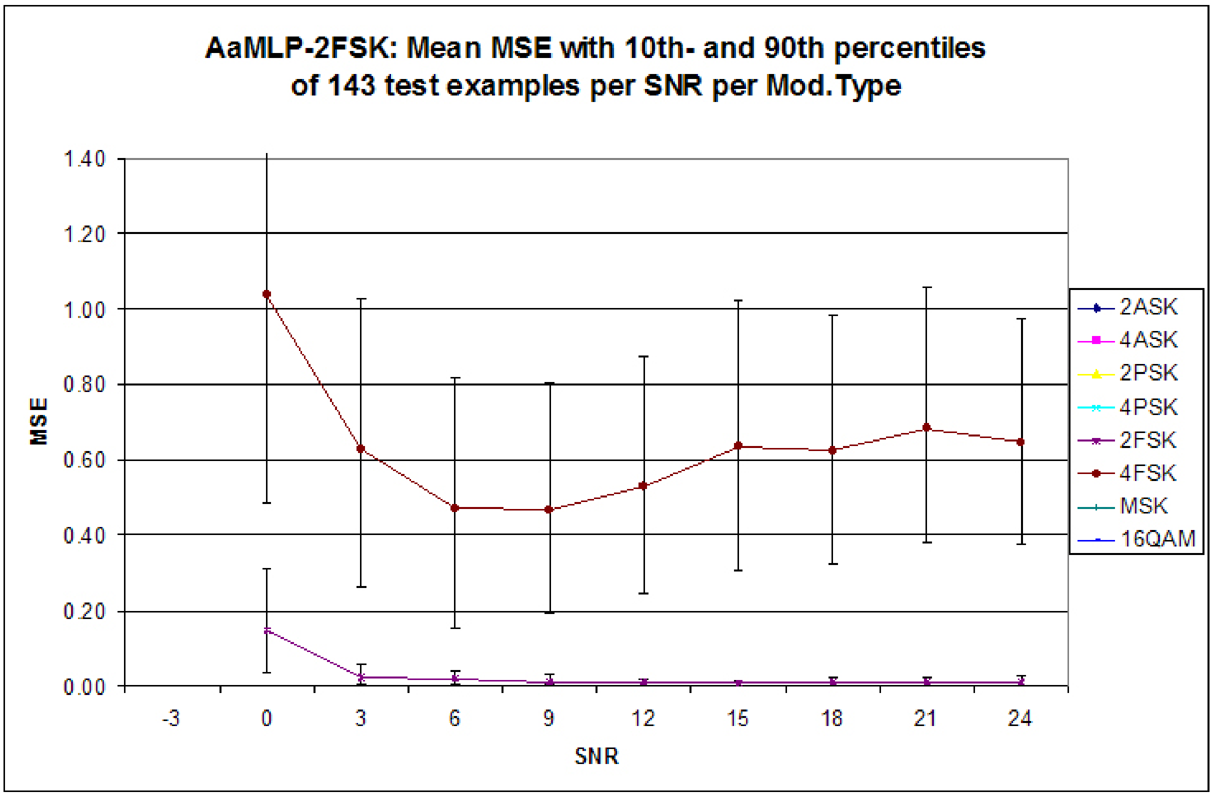


Figure 7.8. AaMLP-2FSK errors.

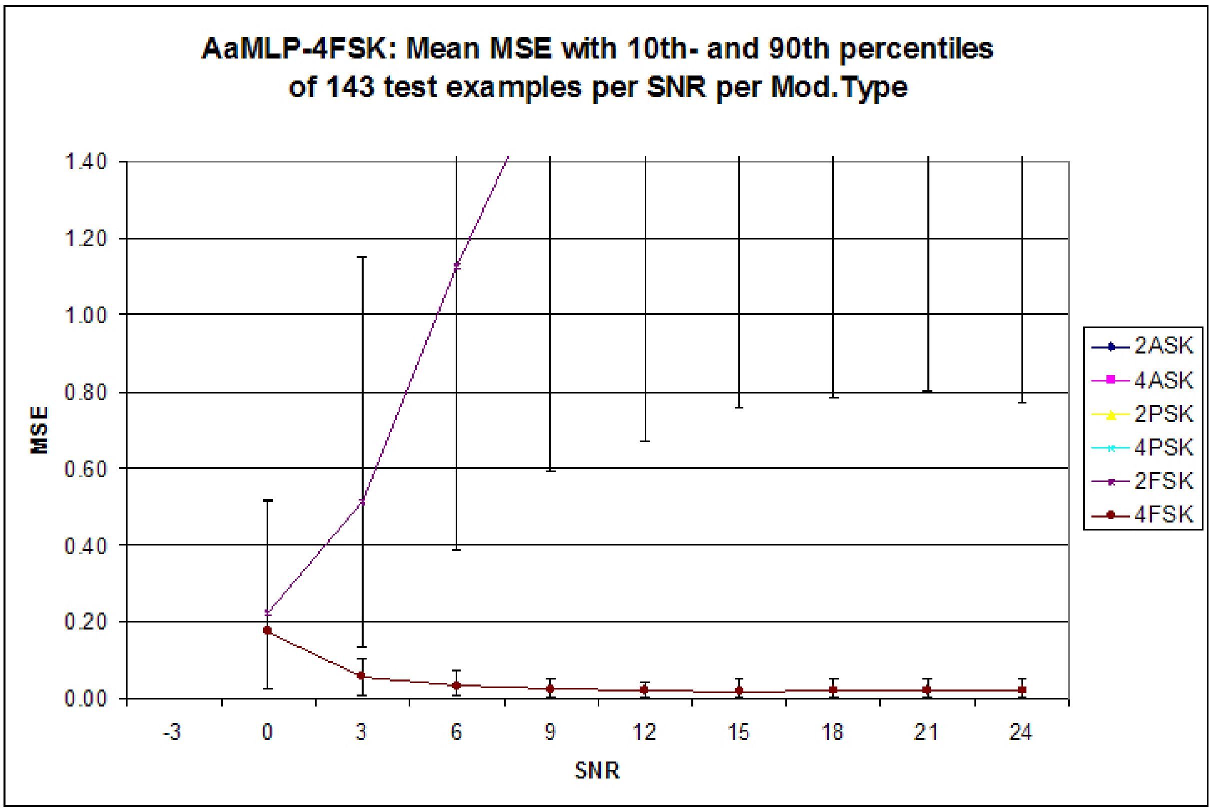


Figure 7.9. AaMLP-4FSK errors.

Table 7.8 shows the absolute difference between the training MSE and the mean MSE of test signals of the same type the networks were trained on (labelled "Corresponding signal type"). The table also shows the mean MSE of the signal type that gave the second lowest MSE ("Closest non-corresponding signal type").

Network	Absolute training and testing MSE difference	
	Corresponding signals	Closest non-corresponding signals
AaMLP-2ASK	0.00433	0.183
AaMLP-4ASK	0.00362	0.086
AaMLP-2PSK	0.00370	0.312
AaMLP-4PSK	0.00217	0.481
AaMLP-2FSK	0.00005	0.573
AaMLP-4FSK	0.00318	1.758

Table 7.8. Absolute training and testing MSE difference (average over 3-24 dB SNR range)

The results indicate that it should be possible to determine a threshold that can be used to accept or reject an incoming signal. The magnitude may also tell how similar an incoming signal segment is to the signals the network has been trained on. In the next chapter we propose a way of integrating the auto-associative networks into a classifier system to enhance its capabilities to reject novel signals.

8 HYBRID CLASSIFIER WITH CLASS INHERENCE VERIFICATION

The preceding chapter showed that the standard MLP, though exhibiting excellent classification capabilities, have problems handling inputs that does not belong to any of the known classes (novel examples). This problem has also been confirmed by other studies, e.g. (13) and (24). We have also investigated the auto-associative networks' capabilities to separate known signal from unknown signals. In the following sections we investigate a method of merging the two approaches to create a hybrid classifier with class-inherence verification, that is, a classifier system with the capacity to reject novel examples.

8.1 General Operation

Figure 8.1 shows an illustration of the proposed system.

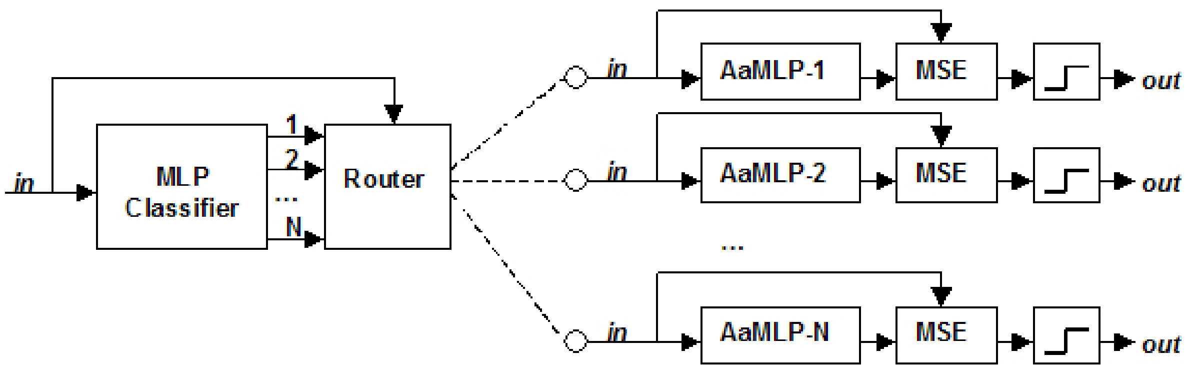


Figure 8.1. Modular classifier: Classification and verification

With reference to the figure, the hybrid classifier works as follows: First, the input vector, *in,* is fed into the standard MLP classifier. The classifier output is then fed into the router together with the input. The router decides which class the input belongs to (usually the class whose corresponding output node has the highest value). This completes the classification stage. Next is the Class Inherence Verification (CIV) stage that verifies that the input actually belongs to the selected class and is not an input of unknown type (ref. Section 7.1.4). Thus, the router supplies the input vector to the auto-association network (AaMLP) that corresponds to the selected class. The mean-squared error (MSE) between the input and output vector of the AaMLP is then thresholded to either verify (*out*=1) or reject (*out*=0) the classification.

To test this approach, we use the MLP classifier described in Chapter 7.1 at the classification stage and the auto-associative networks in Chapter 7.2 at the CIV stage.

8.2 Test Results

Tables 8.1 to 8.3 show the confusion matrices when using the three different CIV thresholds: 0.08, 0.1 and 0.15 respectively. All results are averages for signals in the range 3-24 dB SNR).

	Predicted type (%)						
Actual type	**2ASK**	**4ASK**	**2PSK**	**4PSK**	**2FSK**	**4FSK**	**Novelty**
2ASK	88.37	3.15	0.00	0.00	0.00	0.00	8.48
4ASK	1.84	92.22	0.00	0.00	0.00	0.00	5.94
2PSK	0.00	0.00	89.34	1.31	0.00	0.00	9.35
4PSK	0.00	0.00	1.31	93.09	0.00	0.00	5.59
2FSK	0.00	0.00	0.00	0.00	98.69	0.00	1.31
4FSK	0.00	0.00	0.00	0.00	0.00	94.84	5.16
MSK	0.00	0.00	0.09	23.95	0.00	0.00	75.96
16QAM	0.96	12.76	11.63	2.97	0.00	0.00	71.68

Table 8.1. Confusion matrix for known and novel signals, CIV threshold: 0.08.

Actual type	Predicted type (%)						
	2ASK	4ASK	2PSK	4PSK	2FSK	4FSK	Novelty
2ASK	90.30	3.41	0.00	0.00	0.00	0.00	6.29
4ASK	1.92	93.97	0.00	0.00	0.00	0.00	4.11
2PSK	0.00	0.00	92.66	1.75	0.00	0.00	5.59
4PSK	0.00	0.00	1.40	94.23	0.00	0.00	4.37
2FSK	0.00	0.00	0.00	0.00	98.86	0.00	1.14
4FSK	0.00	0.00	0.00	0.00	0.00	97.12	2.88
MSK	0.00	0.00	0.09	26.14	0.00	0.00	73.78
16QAM	2.10	13.90	15.91	4.28	0.00	0.00	63.81

Table 8.2. Confusion matrix for known and novel signals, CIV threshold: 0.10.

Actual type	Predicted type (%)						
	2ASK	4ASK	2PSK	4PSK	2FSK	4FSK	Novelty
2ASK	93.53	3.67	0.00	0.00	0.00	0.00	2.80
4ASK	2.01	94.93	0.00	0.00	0.00	0.00	3.06
2PSK	0.00	0.00	94.67	2.27	0.00	0.00	3.06
4PSK	0.00	0.00	1.49	96.68	0.00	0.00	1.84
2FSK	0.00	0.00	0.00	0.00	99.30	0.09	0.61
4FSK	0.00	0.00	0.00	0.00	0.00	98.51	1.49
MSK	0.00	0.00	0.09	32.34	0.00	0.00	67.57
16QAM	6.03	15.30	22.03	6.73	0.00	0.00	49.91

Table 8.3. Confusion matrix for known and novel signals, CIV threshold: 0.15.

A low threshold implies a low tolerance. With a CIV threshold of 0.08, more than 70 % of the novel examples are correctly rejected but also some known signal types are incorrectly rejected. A higher threshold means higher tolerance, which implies that more novel examples are incorrectly classified.

The classification rates of the three CIV approaches are compared with the novelty rejection approaches presented in Section 7.1.4 and the results are shown in Figure 8.2. The hybrid CIV classifier displays the best overall classification rates.

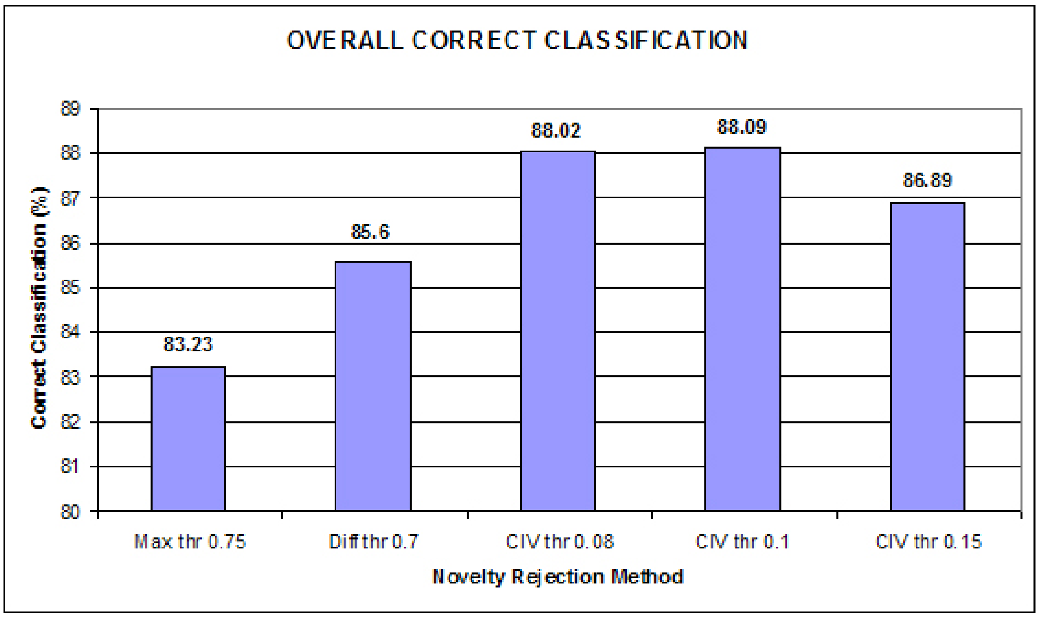


Figure 8.2. Overall correct classification rates for different novelty rejection methods.

Appendix D contains the classification results for the hybrid classifier for each of the eight noise levels.

9 CONCLUSION

This report has considered automatic recognition of radio signal types for electronic warfare applications. The main drawbacks of many methods proposed hitherto are the use of synthetically generated signals in simplified channel models and the lack of evaluation of the classifier's behaviour to unknown signal types. In this work, we have generated a set of semi-realistic signals using a signal generator and created a framework to simplify the production, use and exchange of other signals.

We have evaluated the popular Multi-layer Perceptron's (MLP) ability to handle unknown signal types and found it not to be adequate. We have also investigated MLPs in an auto-associative mode for use as recognisers and found favourable properties. By merging the standard classifier with a bank of auto-association networks, we developed a hybrid classifier with class inherence verification capabilities. This hybrid classifier was able to adequately classify between the known signal types, while at the same time reject a larger amount of unknown signal types than did the stand-alone MLP.

Only a limited number of unknown signal types were used but the inherent properties of the auto-association network should also make it capable to handle other signal types. This is because they are trained to recognise one particular signal type without being dependent on negative training examples. However, comprehensive testing with regards to the hybrid networks' properties remains to be carried out.

References

(1) Smart Sensor Solution Division: Communications Electronic Warfare [online] (2004): NTO Physics and Electronics Laboratory, The Netherlands. HTTP format. [Cited 8th July 2004]. Available from <http://www.tno.nl/instit/fel/div3/feld33-2.html>.

(2) E. E. Azzouz and A. K. Nandi (1997): Automatic Modulation Recognition - I, *Journal of the Franklin Institute* **334**, B, 241-273.

(3) E. E. Azzouz and A. K. Nandi (1997): Automatic Modulation Recognition - II, *Journal of the Franklin Institute* **334**, B, 275-305.

(4) C. -Y. Huang and A. Polydoros (1995): Likelihood Methods for MPSK modulation classification, *IEEE Transactions on Communications* **43**, 1493-1504.

(5) S. S. Soliman and S. -Z. Hsue (1992): Signal classification using statistical moments, *IEEE Transactions on Communications* **40**, 908-916.

(6) D. Boudreau, C. Dubuc, F. Patenaude, M. Dufour, J. Lodge and R. Inkol (2000): A fast automatic modulation recognition algorithm and its implementation in a spectrum monitoring application, In Proceedings of the Military Communications Conference MILCOM2000, 732-736.

(7) A. K. Nandi and E. E. Azzouz (1998): Algorithms for automatic modulation recognition of communication signals, *IEEE Transactions on Communications* **46**, 431-436.

(8) V. Ramakonar, D. Habibi and A. Bouzerdoum (1999): Automatic recognition of digitally modulated communication signals, In Proceedings of the International Symposium on Signal Processing and its Applications ISSPA99, 753-756.

(9) G. Arulampalam, V. Ramakonar, A. Bouzerdoum and D. Habibi (1999): Classification of digital modulation schemes using neural networks, In Proceedings of the International Symposium on Signal Processing and its Applications ISSPA99, 649-652.

(10) E. E. Azzouz and A. K. Nandi (1996): Automatic modulation recognition of communication signals, Kluwer Academic Publishers.

(11) N. Kim, N. Kehtarnavaz, M. B. Yeary and S. Thornton (2003): DSP-based hierarchical neural network modulation signal classification, *IEEE Transactions on Neural Networks* **14**, 1065-1071.

(12) S. Haykin (1999): Neural networks: a comprehensive foundation, Prentice Hall.

(13) M. Gori and F. Scarcelli (1998): Are Multilayer Perceptrons adequate for pattern recognition and verification?, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**, 11, 1121-1132.

(14) M. Markou and S. Singh (2003): Novelty detection: a review - part 1: statistical approaches, *Signal Processing* **83**, 12, 2481-2497.

(15) M. Markou and S. Singh (2003): Novelty detection: a review - part 2: neural network based approaches, *Signal Processing* **83**, 12, 2499-2521.

(16) G. Hatzichristos (2001): Classification of digital modulation types in multipath environments, Electrical engineer's thesis, Naval Postgraduate School, California, USA

(17) V. Ramakonar, D. Habibi, and A. Bouzerdoum (2001): Classification of bandlimited FSK4 and FSK8 signals ,In Proceedings of the International Symposium on Signal Processing and its Applications ISSPA01, 398-401.

(18) J. Venäläinen, L. Terho, and V. Koivunen (2002): Modulation classification in fading multipath channel ,In Proceedings of the Conference on Signals Systems and Computers, 1890-1894.

(19) N. Ghani and R. Lamontagne (1993): Neural networks applied to the classification of spectral features for automatic modulation recognition ,In Proceedings of the Military Communications Conference MILCOM93, 111-115.

(20) B. G. Mobasseri (2000): Digital modulation classification using constellation shape, *Signal Processing* **80**, 2, 251-277.

(21) M. Riedmiller and H. Braun (1993): A direct adaptive method for faster backpropagation learning: the RPROP algorithm, In Proceedings of the IEEE Conference on Neural Networks, 586 - 591.

(22) A. Iversen (2004): Classification of digital modulation schemes using Multi-layered Perceptrons, Technical Report HW-MACS-TR-0016, Heriot-Watt University, Edinburgh.

(23) R. O. Duda, P. E. Hart and D. G. Stork (2001): Pattern Classification. Second Edition, Wiley.

(24) G. C. Vasconcelos, M. C. Fairhurst and D. L. Bisset (1995): Investigating Feedforward Neural Networks with Respect to the Rejection of Spurious Patterns, *Pattern Recognition Letters* **16**, 2, 207-212.

A ABBREVIATIONS

16QAM	16-level quadrature amplitude modulation (modulation format)
2ASK	2-level amplitude shift keying (modulation format)
2FSK	2-level frequency shift keying (modulation format)
2PSK	2-level phase shift keying (modulation format)
4ASK	4-level amplitude shift keying (modulation format)
4FSK	4-level frequency shift keying (modulation format)
4PSK	4-level phase shift keying (modulation format)
A/D	Analog to digital
AaMLP	Auto-association Multi-Layer Perceptron
AMR	Automatic modulation recognition
ASK	Amplitude shift keying (modulation format)
CCITT V.52	Consultative Committee for International Telegraphy and Telephony standard
CIV	Class inherence verification
COM-EW	Communication electronic warfare
Eb/N_0	Bit-energy-to-noise-energy ratio
EW	Electronic warfare
FSK	Frequency shift keying (modulation format)
I/Q	Inphase/quadrature
IF	Intermediate frequency
MLP	Multi-layer perceptron
MSE	Mean squared error
MSK	Minimum shift keying (modulation format)
NCO	Numerically controlled oscillator
PCA	Principal component analysis
PRBS	Pseudo random bit sequence
PSK	Phase shift keying (modulation format)
RF	Radio frequency
SMIQ	Signal generator
SNR	Signal-to-noise ratio
XOR	Exclusive-OR

B SIGNAL FILE FORMAT

Version definition file (*version1-0.def*):

```
# SPECIFICATION OF COMMUNICATION SIGNAL FILES
# VERSION 1.0
# ALEXANDER IVERSEN
# 24 JUNE 2004
#
# - Header consists of 39 fields
# - Fields must not be empty. If data in field not applicable, input '-'
# - Fields are sequenced in the following way:
#   - 1-6    : General information
#   - 7-13   : Generator/equipment
#   - 14-24  : Modulation
#   - 25-29  : Impairments
#   - 30-38  : Reciever
#   - 39     : Layout of following samples values
#   - 40-END : Signal samples
#
# FNUM: Field number
# DTYP: Data type
#       Txt:  Text field of arbitrary length. NB! MUST NOT CONTAIN CHARACTER '\'
#       Num:   Integer or real number
# DLIM: Field delimiter. '\n' represents newline
# DESC: Description of the field contents
#
#
# Note: Use '#' at the start of all comment lines
####################################################################################
#
#FNUM    DTYP     DLIM     DESC
01       txt      \n       Version number
02       txt      \n       Author
03       txt      \n       Place of creation
04       txt      \n       Date of creation
05       txt      \n       Time of creation
06       txt      \n       Comments, general
07       txt      \n       Signal generator/equipment
08       num      \n       Radio frequency (RF) [Hz]
09       num      \n       Intermediate frequency (IF) [Hz]
10       num      \n       RF - IF gain [dB]
11       num      \n       IF gain [dB]
12       txt      \n       Inverted spectrum? (Yes/No)
13       txt      \n       Comments, signal generation/equipment
14       txt      \n       Data source
15       txt      \n       Modulation type
16       txt      \n       Modulation Parameters (levels,index,etc.)
17       num      \n       Symbol rate [Hz]
18       num      \n       Sampling rate [Hz]
19       txt      \n       Transmitter filter function
20       txt      \n       Transmitter filter parameters (roll-off etc.)
21       txt      \n       Window function
22       txt      \n       Window parameters
23       txt      \n       Baseband pulse type (dirac,rect,etc.)
24       txt      \n       Comments, modulation
25       num      \n       Bit energy to noise spectral density ratio (Eb/N0) [dB]
26       num      \n       Noise bandwidth [Hz]
27       txt      \n       Phase noise (description)
28       txt      \n       Multipath (description)
29       txt      \n       Comments, impairments
30       num      \n       NCO frequency [Hz]
31       num      \n       NCO sampling rate [Hz]
32       num      \n       Decimation factor
33       num      \n       Effective sampling rate [Hz]
34       num      \n       Effective centre frequency [Hz]
35       txt      \n       Sample data format (Real/Complex)
36       num      \n       Sample span
37       num      \n       Sample offset
38       txt      \n       Comments, reciever
39       txt      \n       Layout of proceeding samples (e.g., 'i, q')
40       num      \n       Signal sample 1
41       num      \n       Signal sample 2
..
..
END      num      \n       Signal sample END
```

A portion of an example file (*PSK4_EbN0_15.dat*):

```
1.0
Alexander Iversen
FFI
30.06.04
15:00
-
R&S WinIQSIM 4.2, R&S SMIQ 06B
100000000
21400000
11
21
Yes
-
PRN
4PSK
[pi/4 3pi/4 5pi/4 7pi/4]
10000
250000
Cosine
Roll-off:0.95, Impulse length:50 symbol periods
Hanning
-
Dirac
-
15
125000
-
-
Noise BW is half the sampling rate (250kHz/2) ('Full' in WinIQSIM)
21350000
68400000
256
253125
50000
Complex
65536
32768
-
i, q
32775, 32768
32749, 32765
32779, 32783
32837, 32737
32532, 32792
33118, 32824
32682, 32546
31960, 33134
34800, 32534
30339, 32357
33165, 34225
37748, 30598
19751, 33859
56295, 38114
40385, 23224
54595, 25276
11271, 39217
28844, 25172
41140, 21125
42581, 44996
23697, 50946
7948, 30498
21405, 13638
40994, 16947
44803, 36117
28741, 45265
16665, 31171
27200, 14003
50005, 16657
57024, 40838
```

C MANUAL MODULATION DEFINITION IN WinIQSIM 4.20

Example of the definition of 2ASK (*2ASK.imp*)

```
# 2ASK Modulation
# Levels based on the following Matlab modulation:
# mod = (modmap(data,f_symbol,f_sampling,'ask',mod_level)+1.5)/2.5
# Created by Alexander Iversen June 2004
ROHDE&SCHWARZ IQSIM MAPPING FILE
2ASK modulation
QAM
0
2
0.25,   0.00
1.00,   0.00
```

Example of the definition of 4ASK (*4ASK.imp*)

```
# 4ASK Modulation
# Levels based on the following Matlab modulation:
# mod = (modmap(data,f_symbol,f_sampling,'ask',mod_level)+1.5)/2.5
# Created by Alexander Iversen June 2004
ROHDE&SCHWARZ IQSIM MAPPING FILE
4ASK modulation
QAM
0
4
0.20,   0.00
0.46,   0.00
0.73,   0.00
1.00,   0.00
```

D CLASSIFICATION RESULTS FOR HYBRID CLASSIFIER

The tables below show the classification results for the hybrid classifier with a CIV threshold of 0.10. The data are based on 1144 test examples.

Classification results for signals with 24 dB SNR:

	Predicted type (%)						
Actual type	**2ASK**	**4ASK**	**2PSK**	**4PSK**	**2FSK**	**4FSK**	**Novelty**
2ASK	96.50	0.70	0.00	0.00	0.00	0.00	2.80
4ASK	0.00	97.90	0.00	0.00	0.00	0.00	2.10
2PSK	0.00	0.00	99.30	0.00	0.00	0.00	0.70
4PSK	0.00	0.00	0.00	98.60	0.00	0.00	1.40
2FSK	0.00	0.00	0.00	0.00	99.30	0.00	0.70
4FSK	0.00	0.00	0.00	0.00	0.00	99.30	0.70
MSK	0.00	0.00	0.00	0.00	0.00	0.00	100.00
16QAM	1.40	20.98	9.79	0.00	0.00	0.00	67.83

Classification results for signals with 21 dB SNR:

	Predicted type (%)						
Actual type	**2ASK**	**4ASK**	**2PSK**	**4PSK**	**2FSK**	**4FSK**	**Novelty**
2ASK	97.90	0.00	0.00	0.00	0.00	0.00	2.10
4ASK	0.00	97.90	0.00	0.00	0.00	0.00	2.10
2PSK	0.00	0.00	99.30	0.00	0.00	0.00	0.70
4PSK	0.00	0.00	0.00	99.30	0.00	0.00	0.70
2FSK	0.00	0.00	0.00	0.00	98.60	0.00	1.40
4FSK	0.00	0.00	0.00	0.00	0.00	99.30	0.70
MSK	0.00	0.00	0.00	0.00	0.00	0.00	100.00
16QAM	1.40	19.58	11.19	0.00	0.00	0.00	67.83

Classification results for signals with 18 dB SNR:

	Predicted type (%)						
Actual type	**2ASK**	**4ASK**	**2PSK**	**4PSK**	**2FSK**	**4FSK**	**Novelty**
2ASK	95.80	0.70	0.00	0.00	0.00	0.00	3.50
4ASK	0.00	97.90	0.00	0.00	0.00	0.00	2.10
2PSK	0.00	0.00	97.90	0.00	0.00	0.00	2.10
4PSK	0.00	0.00	0.00	100.00	0.00	0.00	0.00
2FSK	0.00	0.00	0.00	0.00	99.30	0.00	0.70
4FSK	0.00	0.00	0.00	0.00	0.00	98.60	1.40
MSK	0.00	0.00	0.00	0.00	0.00	0.00	100.00
16QAM	2.80	18.18	9.79	0.00	0.00	0.00	69.23

Classification results for signals with 15 dB SNR:

	Predicted type (%)						
Actual type	**2ASK**	**4ASK**	**2PSK**	**4PSK**	**2FSK**	**4FSK**	**Novelty**
2ASK	95.80	0.00	0.00	0.00	0.00	0.00	4.20
4ASK	0.00	97.90	0.00	0.00	0.00	0.00	2.10
2PSK	0.00	0.00	98.60	0.00	0.00	0.00	1.40
4PSK	0.00	0.00	0.00	100.00	0.00	0.00	0.00
2FSK	0.00	0.00	0.00	0.00	98.60	0.00	1.40
4FSK	0.00	0.00	0.00	0.00	0.00	98.60	1.40
MSK	0.00	0.00	0.00	0.00	0.00	0.00	100.00
16QAM	0.70	16.78	9.09	0.00	0.00	0.00	73.43

Classification results for signals with 12 dB SNR:

	Predicted type (%)						
Actual type	**2ASK**	**4ASK**	**2PSK**	**4PSK**	**2FSK**	**4FSK**	**Novelty**
2ASK	96.50	0.70	0.00	0.00	0.00	0.00	2.80
4ASK	0.70	95.80	0.00	0.00	0.00	0.00	3.50
2PSK	0.00	0.00	96.50	0.70	0.00	0.00	2.80
4PSK	0.00	0.00	0.00	95.80	0.00	0.00	4.20
2FSK	0.00	0.00	0.00	0.00	99.30	0.00	0.70
4FSK	0.00	0.00	0.00	0.00	0.00	98.60	1.40
MSK	0.00	0.00	0.00	0.00	0.00	0.00	100.00
16QAM	2.80	14.69	20.98	0.00	0.00	0.00	61.54

Classification results for signals with 9 dB SNR:

	Predicted type (%)						
Actual type	**2ASK**	**4ASK**	**2PSK**	**4PSK**	**2FSK**	**4FSK**	**Novelty**
2ASK	88.11	0.00	0.00	0.00	0.00	0.00	11.89
4ASK	0.00	96.50	0.00	0.00	0.00	0.00	3.50
2PSK	0.00	0.00	90.91	1.40	0.00	0.00	7.69
4PSK	0.00	0.00	0.00	98.60	0.00	0.00	1.40
2FSK	0.00	0.00	0.00	0.00	100.00	0.00	0.00
4FSK	0.00	0.00	0.00	0.00	0.00	98.60	1.40
MSK	0.00	0.00	0.00	30.07	0.00	0.00	69.93
16QAM	2.10	12.59	15.39	1.40	0.00	0.00	68.53

Classification results for signals with 6 dB SNR:

	Predicted type (%)						
Actual type	**2ASK**	**4ASK**	**2PSK**	**4PSK**	**2FSK**	**4FSK**	**Novelty**
2ASK	87.41	4.90	0.00	0.00	0.00	0.00	7.69
4ASK	1.40	95.80	0.00	0.00	0.00	0.00	2.80
2PSK	0.00	0.00	87.41	3.50	0.00	0.00	9.09
4PSK	0.00	0.00	0.00	88.11	0.00	0.00	11.89
2FSK	0.00	0.00	0.00	0.00	100.00	0.00	0.00
4FSK	0.00	0.00	0.00	0.00	0.00	95.80	4.20
MSK	0.00	0.00	0.00	89.51	0.00	0.00	10.49
16QAM	3.50	6.29	25.87	14.69	0.00	0.00	49.65

Classification results for signals with 3 dB SNR:

	Predicted type (%)						
Actual type	**2ASK**	**4ASK**	**2PSK**	**4PSK**	**2FSK**	**4FSK**	**Novelty**
2ASK	64.34	20.28	0.00	0.00	0.00	0.00	15.39
4ASK	13.29	72.03	0.00	0.00	0.00	0.00	14.69
2PSK	0.00	0.00	71.33	8.39	0.00	0.00	20.28
4PSK	0.00	0.00	11.19	73.43	0.00	0.00	15.39
2FSK	0.00	0.00	0.00	0.00	95.80	0.00	4.20
4FSK	0.00	0.00	0.00	0.00	0.00	88.11	11.89
MSK	0.00	0.00	0.70	89.51	0.00	0.00	9.79
16QAM	2.10	2.10	25.18	18.18	0.00	0.00	52.45