Appendix D

PROJECT DESIGN AND IMPLEMENTATION

TABLE OF CONTENTS

INTRODUCTION

This document, outlines the processes involved in the successful outcome of this project and the approaches made throughout.

PROBLEM DEFINITION

PROBLEM STATEMENT

In 802.11b wireless packets the preamble of the packet is a bit sequence that allows the receiver to obtain time frequency synchronization and channel estimation for each payload transmission. A preamble has various different attributes, however in this case it will be used for packet detection. In a wireless network if packets are missed, the performance of the network will decrease, therefore detecting when a preamble is present is key. On the contrary, falsely declaring that a packet is present will also reduce the performance of a network causing unnecessary delay increasing unneeded signal processing using extra computation.

The application of deep learning in industry has increased rapidly in the last number of years showing that it can have a vast impact on the performance of companies by reducing errors, identifying new trends and looking at new methodologies in certain domains. It is widely used in speech recognition, image processing, natural language processing and machine translation. It has also become an increasingly popular research topic in communications.

By applying deep learning to the detection of 802.11b wireless packets can it be proven that these methods are more robust than traditional packet detectors. In theory, what is the trade off between deep learning approaches and traditional approaches of detection? The central question to be investigated is:

"Is there a tradeoff between traditional preamble detection techniques, against deep learning architectures in 802.11b wireless packet detection?"

IDEAL PROBLEM SOLUTION

An ideal solution to this problem, is to find an appropriate deep learning architecture that has a high percentage accuracy of the detection of wireless packets. The model must be robust, scalable and must perform well without any prior knowledge of underlying channel models.

RESEARCH

RESEARCH METHODOLOGIES

As defined in the literature review in appendix A, when in the research section of this project a multi vocal approach was taken. To this front, peer reviewed and non peer reviewed papers and articles were considered. The reason behind this approach ensured that a wide range of topics and elements of wireless communications were examined. Wireless communications is a widely researched area, however during the initial literature review it was found that there were small numbers of peer-reviewed and non peer reviewed papers on the specific topic of "Detection algorithms for packet based wireless networks using deep learning". However, elements of the traditional methods of wireless packet detection gave a very in-depth and good understanding of the area. With this, the application of neural networks (NN) with memory were looked at. WIth the lengths of preambles varying, NN's with memory was deemed an interesting topic to be looked at further. Deep research was carried out on the architectures and complexities of three specific NN architectures namely, recurrent neural networks (RNN), long short term memory (LSTM) and bi-directional LSTM (BLSTM).

RESEARCH FINDINGS

The literature review findings proved to be a major factor in the final results of the project, outlined in Appendix E. RNN's were initially looked at as the most efficient architecture for this problem. A feed forward NN with internal memory, therefore considering both the current input and output previously learned. However, it was found that RNN's struggle processing long sequences, when using the ReLU activation function. With this it was ensured when building the architecture activation functions were tested to prove a best fit for this specific problem as premale sequences vary. It was also found that a disadvantage of this architecture covers a vanishing gradient and exploding gradient problem, when gradient computation involving recurrent multiplication on each cell results in a number less than 1 (it will vanish) and greater than 1 (it will explode).

It was found however, that a modified version of the RNN architecture, an LSTM resolves the gradient problem resulting in a more robust and efficient model. An LSTM utilizes four specific gates that control the flow of the model. Namely, an input gate, input modulation gate, forget gate and output gate. The element of difference in this architecture and advantage over RNN's is that the recurrent multiplication of each cell is multiplied by the same value over and over resulting in the gradient problems. In this architecture the forget gate varies ths value each time which is being recurrently multiplied, thus avoiding this problem. However, with this architecture preamble information will only be preserved from the past. Through this research it was found that a re-engineered architecture, namely a BLSTM could be used to understand the context of the preamble and preserve the information from both the past and the future by running input both ways using two LSTM layers.

BLSTMs were found in most peer reviewed papers to improve LSTM model performance in varying classification problems. Using this model, input is presented in forward and backward states to two separate LSTM networks, that are both connected to the same output layer. By preserving information from the future and using two hidden states we can successfully preserve preamble information from both the past and future and compute the output sequence. This will serve as an advantage as the length of 802.11b wireless packets can be extensively long.

The assumptions taken from the literature review were that the BLSTM model would perform well in the detection of wireless packets. However, all architectures were looked at in the process of this project. It was found that this assumption was true using both multipath and additive gaussian white noise data. However, using a binary symmetric channel dataset the RNN model proved to outperform both LSTM and BLSTM data. Although, this dataset consisted of randomly flipped sequences based on an error probability. In industry and real world applications this would not be the case of how a preamble sequence is properly received, however is used extensively in research of such areas.

DATA

As outlined in the final paper, 802.11b wireless packets use direct sequence spread spectrum (DSSS), to reduce overall signal interference. This method spreads a signal across a large bandwidth, thus creating a low power density across various spectrums. Wireless packet preambles, then represent a string of bits represented by logic 1 and 0 data bits. DSSS are represented by code sequences, creating pseudo random sequences that aid in the protection of noise and interference as well as different packets using the same preamble sequence. There are many different code sequences used, namely

- Maximum length sequences
- Gold codes
- Orthogonal codes

This paper utilises maximum length sequences (m-sequences) due to its autocorrelation properties which ensures avoiding a false synchronisation problem. For large sequences, m-codes are the go-to and the main form of code sequences employed in industry and research. Therefore, this approach was taken.

The construction of maximum length sequences is explained in the final paper. Linear feedback time register (LFSR) structures are utilised here to create the sequence codes. In industry, there are two main types of LFSR structures used, namely Galois LFSR and Fibonacci LFSR, although the Galois approach is preferred as its structures run on a faster frequency. The fibonacci approach uses XOR between several bits using multiple XOR gates. In contrast to Galois structures using only two XOR gates with the delay path minimised.

As outlined in equation (1) of the final paper, the sequence is generated outlines that a sequence is made using a generator polynomial and its coefficients. This generator polynomial must be a primitive polynomial. s also outlined in the paper the sequence length is calculated based on the degree (L) of the polynomial, with the sequence length being in the form,

$$N = 2^L - 1$$

For the simulations in this project the primitive polynomial,

$$X^5 + x^2 + 1$$

Was used resulting in a sequence length of 31. As computationally this is effective for the purpose of this research.

As outlined in the final paper, to test a wide range of situations, three datasets were generated to formulate a comprehensive set of results gauging contrasting elements of noise and channel capacities. Each dataset generated contains an input x and the corresponding label y. The input x represents the received signal sequence. The label y represents the binary representation of signal detection. Where 1 represents signal detected and is computed by the reference polynomial and reference initial state. When y is equal to 0, two different states are represented. Namely, a reference polynomial with a random initial state and also, a random polynomial and a random initial state.

DATASET GENERATION

1) BINARY SYMMETRIC CHANNEL (BSC)

BSC channels accept an input sequence, in this case the m-sequence generated by the galois LFSR structure and randomly flips the sequence based on a given error probability. The error probability represents the probability of receiving an incorrect bit, and a range of error probabilities were looked at that can be seen in Appendix E.

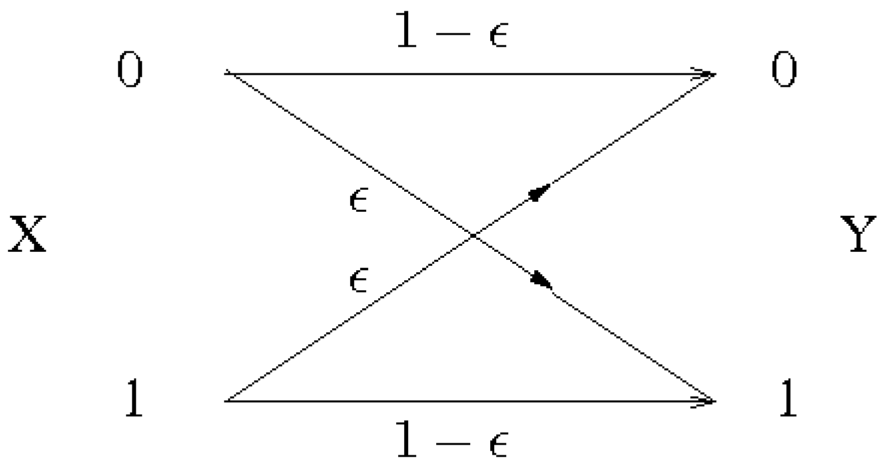


Fig. 1 The binary symmetric channel

In Fig.1 ε represents the error probability rate that bits will be flipped at.

2) ADDITIVE WHITE GAUSSIAN NOISE (AWGN) CHANNEL

The second dataset generated adds AWGN to the m-sequences. The power of noise to the sequences were generated by varying values of signal to noise ratio. As shown in Appendix E, -15 SNR db to 10 SNR dB were addressed. -15 SNR dB represents a large amount of noise in the signal, where 10 SNR dB represents little noise in the signal. Looking at these varying values gave a scope on what was the correct amount of SNR dB to use, and it was concluded that -5 SNR dB is the most relatively common power of noise used in industrial research and a decision was made to use this when generating the final dataset.

3) MULTIPATH CHANNELS

In wireless channels parameters randomly change over time. In a wireless signal the same packet is sent multiple times. Each one with different arrival times and different paths. Resulting in the receiver, receiving multiple packets at multiple different strengths and phase angles. From this, it can be said that a multipath channel changes randomly in respect to time. The output of the channel is expressed as the convolution of the channel input and the normalized channel. How this is generated can be seen in the code snippet below:

```
current_channel = [0 1 (rand(1,6)-1/2)/6];
channel_normalised = current_channel/sqrt(sum(abs(current_channel).^2));
channel_out = conv(channel_normalised, channel_in);
```

AWGN is then added to the output channel to generate the whole dataset. Again -5 SNR dB is set.

PROJECT ARCHITECTURE

Fig. 2 below, outlines the high level architecture of the developed system based on the dataset that is given. It outlines, when a wireless packet preamble is received, and based on the specific dataset what channel the preamble is in. The resulting channel is then passed through the traditional and deep learning classification model, ranging from RNN, LSTM and BLSTM.

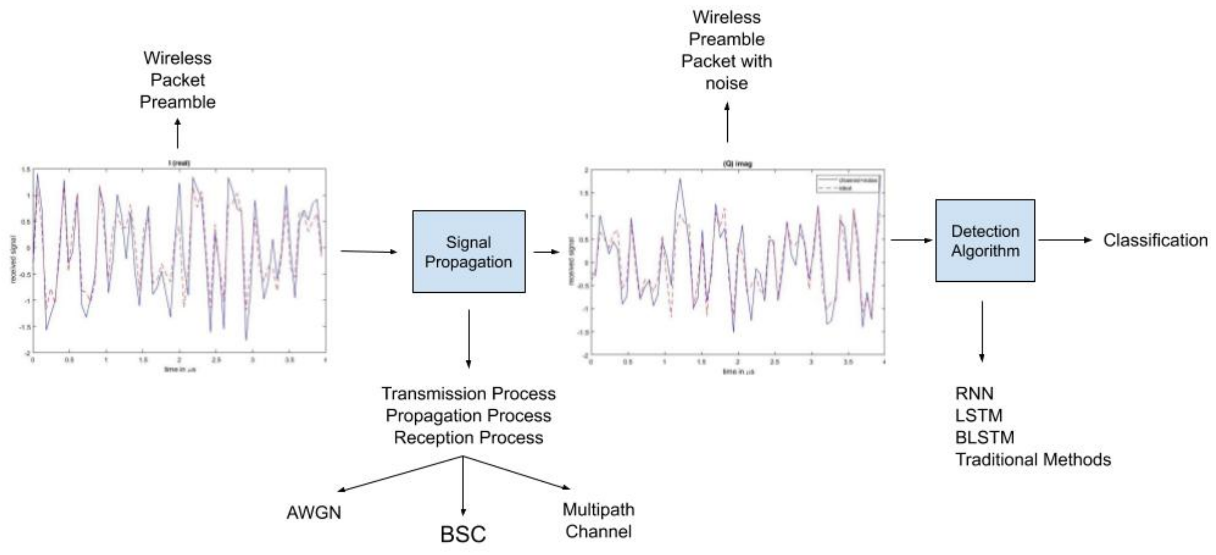


Fig. 2. High Level Architecture

MODEL DESIGN AND ARCHITECTURE

As outlined in the final paper, table I - III represent the final models and implemented architectures. However many more were also looked at. This section breaks down different summaries of models and how they were applied.

DEEP LEARNING TECHNIQUES

Initially, different pooling techniques were trialled. Pooling is essentially downscaling the given input that is obtained from the previous layer. This operation is done using hardcoded tensor operations such as max, global max, average and global average. In this project, global max and global average pooling were specifically looked at.

1) GLOBAL MAX POOLING

In this case, the pool size within the pooling layer is equal to the input size and the max of the entire pool is the output value.

2) GLOBAL AVERAGE POOLING

In this case, the same process applies as above, however the average of the entire pool is the output value.

In all the models tested dense layers are used with the sigmoid activation function. This layer reduces the dimensionality of the output space. The operation implemented by dense represents

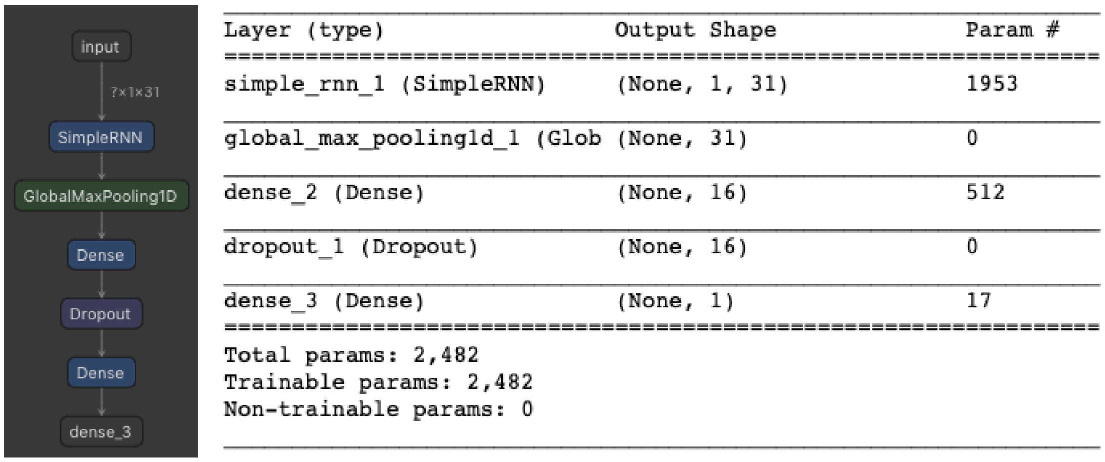
$$Output = activation(dot(input, kernel) + bias)$$

Where activation, in our case sigmoid is the activation function, the kernel represents the weights matrix calculated by the layer and bias represents the bias matrix calculated by the error. This layer is used throughout all models developed in the three architectures.

RNN ARCHITECTURES

A simple RNN model architecture was developed as shown in Fig. 3. This shows a simple RNN layer utilising global max pooling with a dense layer with an output of 16. A dropout rate of 0.2, with the total training parameters of 2,482.

The results of this model can be seen in Appendix E.



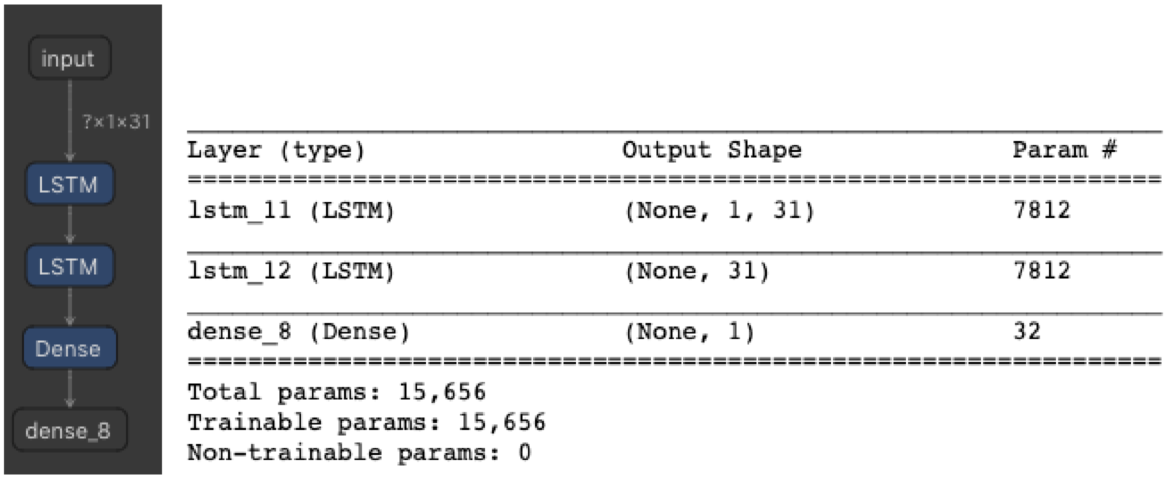
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_1 (SimpleRNN)     (None, 1, 31)             1953
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 31)                0
_________________________________________________________________
dense_2 (Dense)              (None, 16)                512
_________________________________________________________________
dropout_1 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 17
=================================================================
Total params: 2,482
Trainable params: 2,482
Non-trainable params: 0
_________________________________________________________________
```

Fig. 3 RNN Architecture used across all datasets

LSTM ARCHITECTURES

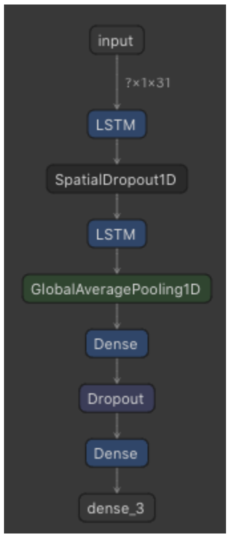
Five different LSTM architectures were tested as seen below in Fig. 4 - Fig. 9 each with varying layers using spatial dropout, dropout, average and max global pooling and dense layers using a sigmoid activation function.

The results of this model can be seen in Appendix E.



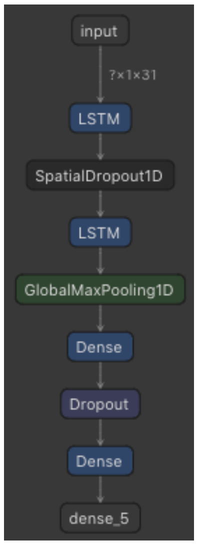
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_11 (LSTM)               (None, 1, 31)             7812
_________________________________________________________________
lstm_12 (LSTM)               (None, 31)                7812
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 32
=================================================================
Total params: 15,656
Trainable params: 15,656
Non-trainable params: 0
```

Fig. 4. Two Layer LSTM



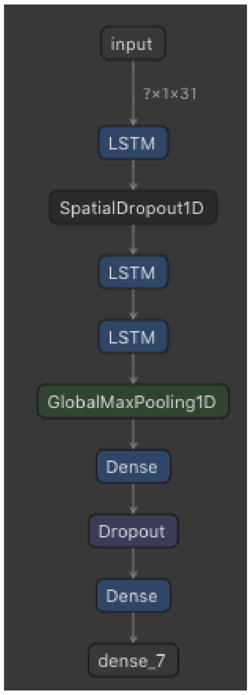
```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 1, 31)             7812
_________________________________________________________________
spatial_dropout1d_3 (Spatial (None, 1, 31)             0
_________________________________________________________________
lstm_3 (LSTM)                (None, 1, 31)             7812
_________________________________________________________________
global_average_pooling1d_1 ( (None, 31)                0
_________________________________________________________________
dense_2 (Dense)              (None, 16)                512
_________________________________________________________________
dropout_1 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 17
=================================================================
Total params: 16,153
Trainable params: 16,153
Non-trainable params: 0
```

Fig. 5. Two layer LSTM, using spatial dropout and global average pooling



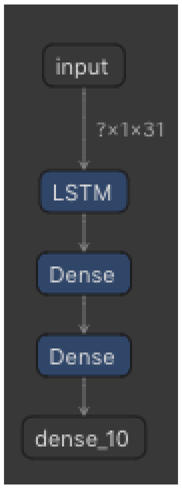
```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_6 (LSTM)                (None, 1, 31)             7812
_________________________________________________________________
spatial_dropout1d_5 (Spatial (None, 1, 31)             0
_________________________________________________________________
lstm_7 (LSTM)                (None, 1, 31)             7812
_________________________________________________________________
global_max_pooling1d (Global (None, 31)                0
_________________________________________________________________
dense_4 (Dense)              (None, 16)                512
_________________________________________________________________
dropout_2 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 17
=================================================================
Total params: 16,153
Trainable params: 16,153
Non-trainable params: 0
```

Fig. 6. Two layer LSTM, using spatial dropout and global max pooling



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_8 (LSTM)                (None, 1, 31)             7812
_________________________________________________________________
spatial_dropout1d_6 (Spatial (None, 1, 31)             0
_________________________________________________________________
lstm_9 (LSTM)                (None, 1, 31)             7812
_________________________________________________________________
lstm_10 (LSTM)               (None, 1, 31)             7812
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 31)                0
_________________________________________________________________
dense_6 (Dense)              (None, 16)                512
_________________________________________________________________
dropout_3 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 17
=================================================================
Total params: 23,965
Trainable params: 23,965
Non-trainable params: 0
```

Fig. 7. Stacked LSTM, using spatial dropout and global max pooling



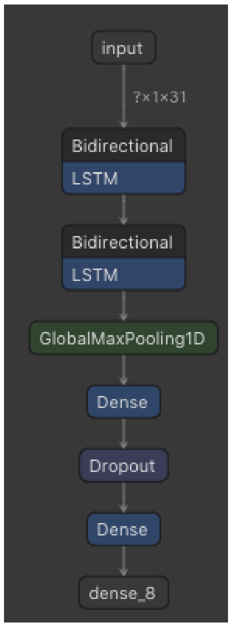
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_13 (LSTM)               (None, 1, 31)             7812
_________________________________________________________________
dense_9 (Dense)              (None, 1, 15)             480
_________________________________________________________________
dense_10 (Dense)             (None, 1, 1)              16
=================================================================
Total params: 8,308
Trainable params: 8,308
Non-trainable params: 0
```

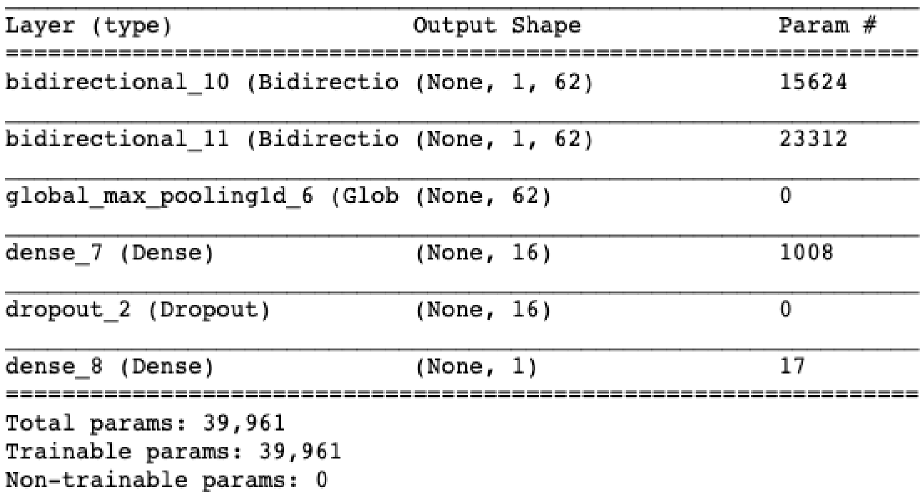
Fig. 8. A low complexity LSTM with two dense layers

BLSTM ARCHITECTURES

Fig.9 outlines the BLSTM architecture built using two BLSTM layers and a global max pooling layer. Two dense layers are also used here reducing the dimensionality from 62 to 16, and 16 to 1 respectively. Total trainable parameters is 39,961.

The results of this model can be seen in Appendix E.





```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_10 (Bidirectio (None, 1, 62)             15624
_________________________________________________________________
bidirectional_11 (Bidirectio (None, 1, 62)             23312
_________________________________________________________________
global_max_pooling1d_6 (Glob (None, 62)                0
_________________________________________________________________
dense_7 (Dense)              (None, 16)                1008
_________________________________________________________________
dropout_2 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 17
=================================================================
Total params: 39,961
Trainable params: 39,961
Non-trainable params: 0
```

Fig. 9. Two layer BLSTM using global max pooling

MODEL TRAINING AND EVALUATION METRICS

MODEL COMPILATION

As outlined in the as this is a binary classification problem, binary cross entropy is used as the loss function. This loss function evaluates the log probability of a packet being detected or not. This loss function effectively measures the effectiveness of the model learning.

An optimizer increases the learning speed of the model. The efficient adam optimizer is also used throughout, however the use of SGB optimizer was also looked at however this was deemed to have a lower performance rate. The benefits of using the adam optimizer for this type of preamble detection, are that it is computationally efficient, and has little memory requirements and is also very well suited to large data and sequences with noise.

MODEL EVALUATION METRICS

NEYMAN PEARSON (NP) CURVES

NP curves are used to evaluate the trade off between traditional detection techniques and the trained deep learning models. False alarm rates are plotted against missed detection rates to outline the performance of the different methods used. If a packet is detected, where the prediction is wrong the error represents a false alam, since we mistakenly detected the wrong packet received. This is shown on the x axis. In contrast to not detecting a packet, when it is a packet represents missed detection shown on the y axis. With a probability threshold of, λ as it decreases or increases the curve becomes smaller or larger. As can be seen in Appendix E, the trained model curves are significantly inside the traditional method resulting in the level of detection being higher. The closer the models NP curve comes to the traditional curve shows that the level of detection is reducing. Fig. 10, outlines the miss detection and false alarm probabilities.

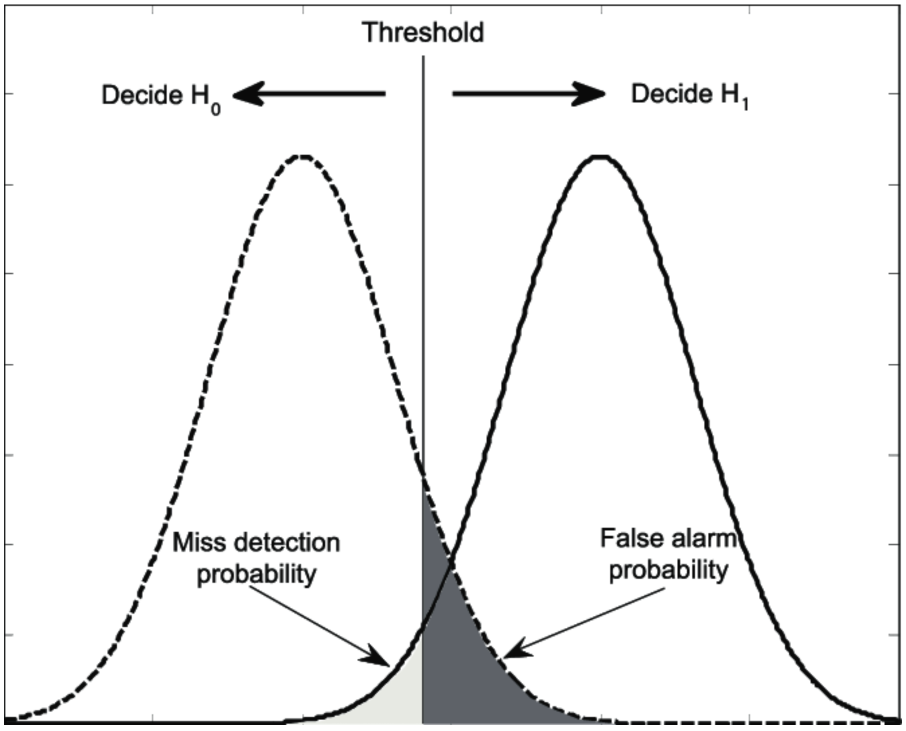


Fig. 10 Missed detection and False alarm probabilities based on a threshold

PRECISION AND ACCURACY

In binary classification problems, precision and accuracy are widely used in evaluating the performance of a model. They are calculated using four prior metrics that are calculated using a confusion matrix. Namely,

- True Positives (TP): Predicted positive and it’s true. Here, the model predicted a packet is detected and that is true.
- False Positives (FP): Predicted positive and it’s false. Here, the model predicted a packet is detected but it is actually wasnt.
- True negatives (TN): Predicted negative and it’s true. Here, the model predicted a packet is not detected and it is actually wasnt.
- False Negatives (FN): Predicted negative and it’s false. Here, the model predicted a packet is not detected but it actually was.

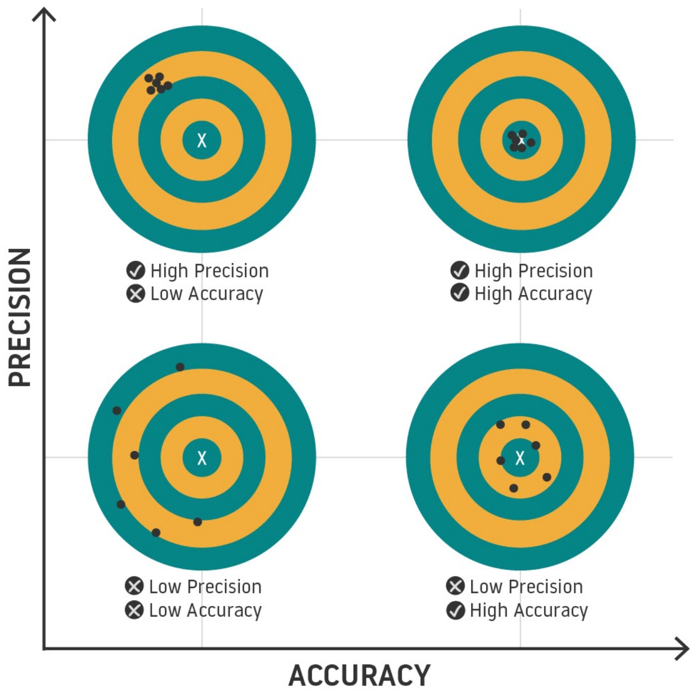


Fig. 11. Precision Vs Accuracy

As can be seen in Fig. 11, it is of upmost importance that the model has high values of precision and accuracy. Precision, evaluates the accuracy of positive predictions the classifier makes. For example, as seen in Table IV in the final paper, the RNN model has a precision score of 0.97 and accuracy of 0.98. This specifies when a packet is correctly detected, the model is correct 96% of the time, while being 98% accurate. while also indicating a correct detection of 98% of packets. Precision is defined as,

$$Precision = \frac{TP}{TP + FP}$$

Although the accuracy metric is used, having a high accuracy does not ensure a high performing model. Precision and recall metrics along with ROC and AUC metrics give a wider overview of performance evaluation. Accuracy is defined as,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

RECALL

Recall is another evaluation metric used along side precision and accuracy. This evaluates the ration of packets correctly identified by the classifier. In table IV of the final paper, recall is 0.98 indicating a correct detection of 98% of packets. Recall is defined by,

$$Recall = \frac{TP}{TP + FN}$$

RECEIVER OPERATING CHARACTERISTIC (ROC) CURVES & AREA UNDER CURVE (AUC)

A ROC curve represents the true positive rate plotted against the false positive rate, the amount of negative observations that are incorrectly classified. ROC curves are widely used today in real world machine learning projects and are used to great effect with AUC scores. An AUC score of 1, is the optimal score. Fig. 12, shows a range of ROC curves with specific AUC scores for clarity.

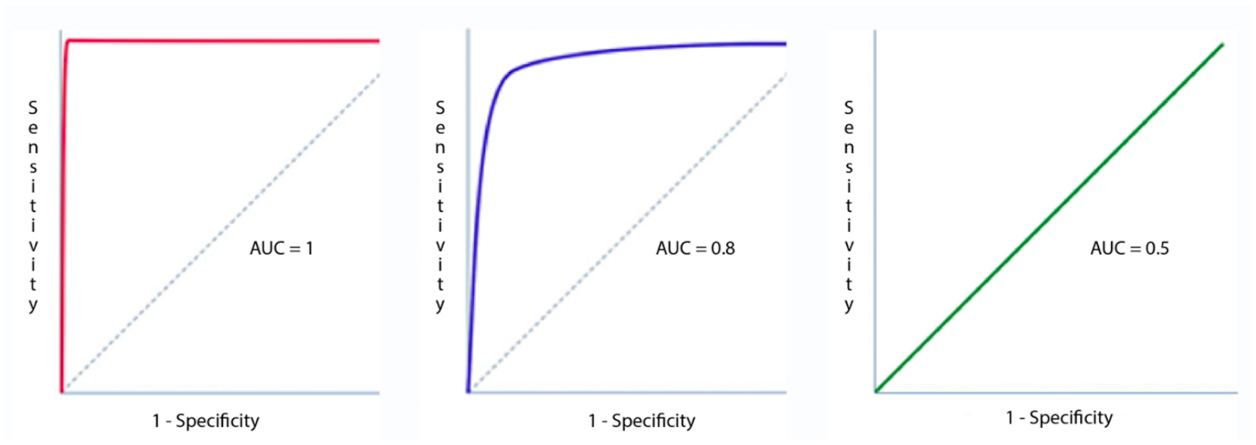


Fig. 12. Ranging ROC curves with AUC

CONCLUSION

To conclude, the extensive research in the initial literature review paved the way for the successful outcome of this project. The decision to generate data using m sequences using different datasets worked very well under all conditions. The assumptions made in most cases were correct and the findings of this project will now be brought forward to be used in industry. Which, was the goal of this project from the beginning.