Neural Network Scheme for Channel Coding

Satya Kumar Vankayala[1], Tirtankar Mittra[1], Seungil Yoon[2]

***Abstract*—Channel Coding adds additional bits of information to the message bits so that erroneous bits can be corrected at the decoder. There are many popular channel coding schemes like Low Density Parity Check (LDPC) codes, invented by Gallager, that have revolutionized communication systems by being capacity approaching codes. With the advent of 5G communications, vRAN and O-RAN technologies, it is now feasible to use neural networks and other software based machine learning algorithms so in this paper we explore how our Neural Network scheme works in comparison with the benchmark LDPC decoder.**

I. Introduction

In this section, the paper gives a brief description of a Neural Network(NN). Neural Networks were built to mimic the functionality of a human brain it is a type of artificial neural network that consists of multiple hidden layers. Each layer consists of multiple neurons, and every neuron of a particular layer is connected to a neuron of the next layer. For every connection, the network will assign a weight (random numerical value), which will be adjusted if the algorithm was not able to recognise a particular pattern effectively. The depth of the network depends on the number of layers that compose it.

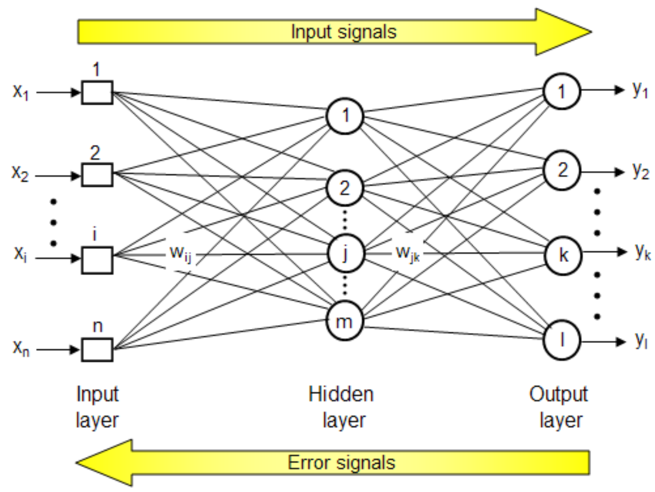


Fig. 1. Forward and back propagation in neural networks

The dynamics of the Neural Network system is governed by forward propagation and backward propagation. For example Equation [1] calculates loss(J) during forward propagation for a one hidden layer neural network. Similarly, Equation [2] depicts how the weight matrices $[W^{(1)}, W^{(2)}...]$ are updated in gradient descent using backward propagation.

$$J = l(\phi_2(W^{(2)}\phi_1(W^{(1)}x)), y) + \frac{\lambda}{2}\left(||W^{(1)}||_F^2 + ||W^{(2)}||_F^2\right) \tag{1}$$

$$W^{(i)} = W^{(i)} - \alpha\frac{\partial J}{\partial W^{(i)}} \tag{2}$$

II. Proposed Scheme

This section discusses the algorithm used for the proposed scheme in detail. **Algorithm 1** shows the steps for training our Neural Network Model, where as **Algorithm 2** shows the steps for evaluating our model and Fig. (2) pictorially represents our Neural Network(NN) decoder model.

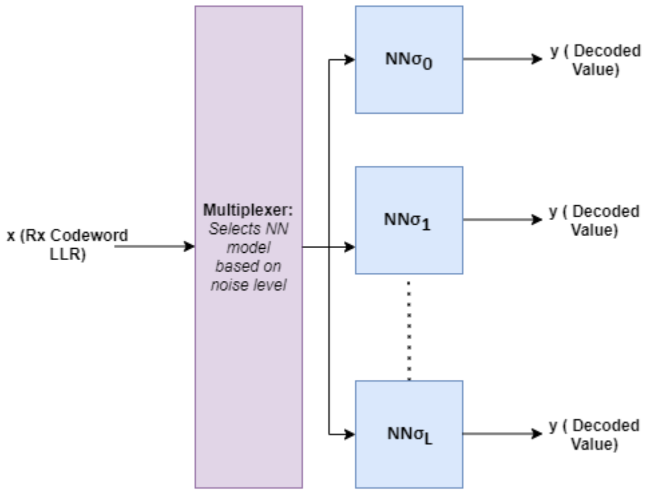


Fig. 2. Neural Network Model of our Decoder

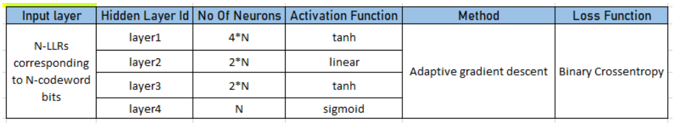
Algorithm 1 Training Phase Of Our Model.

```
1:  Create $A_{tr}$ random message bits of size $K$.
2:  Add parity bits which increases message bit size to $N$. Store it in $Y_{tr}$.
3:  Modulate using (BPSK) and the add different levels of AWGN noise $[\sigma_0, \sigma_1, ..., \sigma_L]$. Store it in $[X_{\sigma_0}, X_{\sigma_1}, ..., X_{\sigma_L}]$.
4:  Divide $[X_{\sigma_0}, X_{\sigma_1}, ..., X_{\sigma_L}]$ into training and validation set $[X_{tr\sigma_0}, X_{tr\sigma_1}, ..., X_{tr\sigma_L}]$ and $[X_{val\sigma_0}, X_{val\sigma_1}, ..., X_{val\sigma_L}]$ respectively.
5:  Initialize Neural Network models $[NN_{\sigma_0}, ..., NN_{\sigma_L}]$.
6:  For $\sigma_i$ in $[\sigma_0, \sigma_1, ..., \sigma_L]$
7:      Do
8:          Train $NN_{\sigma_i}$ by passing $X_{tr\sigma_i}$.
9:          Calculate error($Err$) on validation set $X_{val\sigma_i}$.
10:         If $Err > \delta$
11:             Modify hyper-parameters of $NN_{\sigma_i}$.
12:         Else
13:             Break
14:     While true
```

Algorithm 2 Testing Phase Of Our Model.

```
1: Create $A_{test}$ random message bits of size $K$.
2: Add parity bits which increases message bit size to $N$. Store it in $Y_{test}$.
3: Modulate using (BPSK) and the add different levels of AWGN noise $[\sigma_0, \sigma_1, ..., \sigma_L]$. Store it in $[X_{\sigma_0}, X_{\sigma_1}, ..., X_{\sigma_L}]$.
4: Randomly shuffle $[X_{\sigma_0}, X_{\sigma_1}, ..., X_{\sigma_L}]$ into one set $X_{test}$
5: For $x$ in $X_{test}$
6:     Determine $\sigma_i$ for $x$.
7:     Predict using $NN_{\sigma_i}$ for $x$.
8:     Compare predicted value with $y$ in $Y_{test}$.
9: Plot BLER vs SNR graph.
```

III. Results



Input layer	Hidden Layer Id	No Of Neurons	Activation Function	Method	Loss Function
N-LLRs corresponding to N-codeword bits	layer1	4*N	tanh	Adaptive gradient descent	Binary Crossentropy
	layer2	2*N	linear		
	layer3	2*N	tanh		
	layer4	N	sigmoid		

Fig. 3. Neural Network Descriptor Table

In order to generate BLER V/S SNR graph for Neural Network Decoder we took the codeword size $(N = 32)$ and message bit size $(K = 22)$. BLER was calculated on test data using the appropriate Neural Network model(as explained earlier different Neural Network models are used based on the noise level). For the above simulation we found the optimal Neural Network(NN) blocks for different noise levels σ which is shown in Fig. (3). The BLER V/S SNR comparison is shown in Fig. (4).

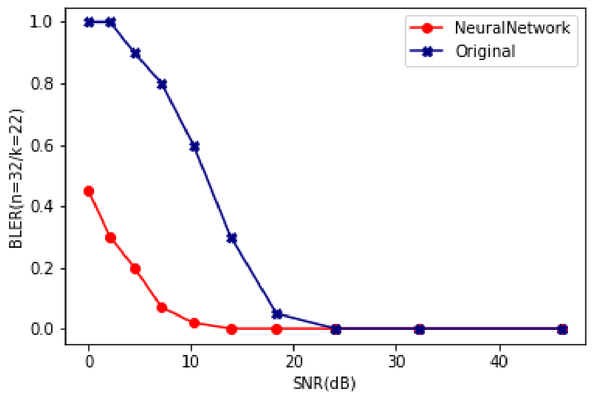


Fig. 4. Block error rate comparison of proposed Neural Network method versus known original method(no approximations) for $n = 32$, $k = 22$.

IV. Conclusion

Our Neural Network(NN) decoder model far exceeds the LDPC decoder used in terms of performance, as shown in Fig. (4). LDPC decoder achieves its optimal performance for large codewords and is dependent on the parity check matrix(H) but in our simulations we have used small codewords so our future work should explore how this model compares with LDPC decoder for higher sized codewords. Also, for our simulations we have used a dense deep neural network, instead to make the computations less expensive we could selectively remove some connections from the neurons, for example its advisable to connect those bits together which constitutes one parity check equation from the input layer to the first hidden layer.

References