Implementing Transmitter, Channel and Receiver using Autoencoder

- Apoorva Jha (15MI420), Harish (15MI421)

OBJECTIVE:

To implement layers of a communication system (receiver, channel, transmitter). We try to accurately obtain the message signal when passed through the architecture, with different noises emulating the channel.

METHODOLOGY:

To implement such an end-end system we use an autoencoder.
The encoder layers act as transmitter and decoder layers for receiver. We add a gaussian layer for noise in the channel.
During training,

1. We train the system for one particular SNR (signal to noise ratio) value, and passing the message signal to the entire autoencoder model and expect to receive the same.

During testing,

1. We separately define the encoder model for transmitter and decoder model for receiver.
2. We use this to predict the output that is transmitted.
3. Then mix it with different noise, representing the channel noises. This is the signal received.
4. To see the actual message, we pass through the receiver
5. Compared this with the message sent.

DATA:

1. The parameters for structuring the message signals are:
    a. The message sent is one of the possible M messages. Here M is 4.
    b. $M=2^k$, k bits are used to send the message. Say k=2 (0 or 1), M will be something of the format [0,1,0,0].
    c. n defines the number of discrete ways the channel is used. Here it is 2.
    d. R is the communication rate of the channel given by n/k.
2. For training, we create 5000 sample messages, with bit 1 at different positions and also separately maintain the position of bit.
3. Four different messages are possible: [1,0,0,0] [0,1,0,0] [0,0,1,0] [0,0,0,1]
4. For testing, we create 1000 data samples similar to training data.

5. For evaluating model, we consider SNR(dB) from -15 to 15.

ARCHITECTURE:

We use an autoencoder for our system.

Transmitter/Encoder:

1. The first layer receives the message signal.
2. It is the dense layer with M neurons and has ReLu activation function.
3. We use ReLu in the start because we know are output has no negative values.
4. This is passed to another dense layer with activate function softmax and n number of neurons.
5. We use an L2 regularizer in this layer to avoid overfitting

Channel:

1. We use a gaussian noise layer at SNR(dB=7), mean=0 and standard deviation = $\sqrt{\frac{1}{2*R*SNR}}$

Receiver/Decoder:

1. We use 2 dense layers with M neurons.
2. The first layer has ReLu activation function.
3. The second layer has Softmax function

The model is optimized using Adam optimized at 0.001 learning rate and crossentrpy loss function is used.

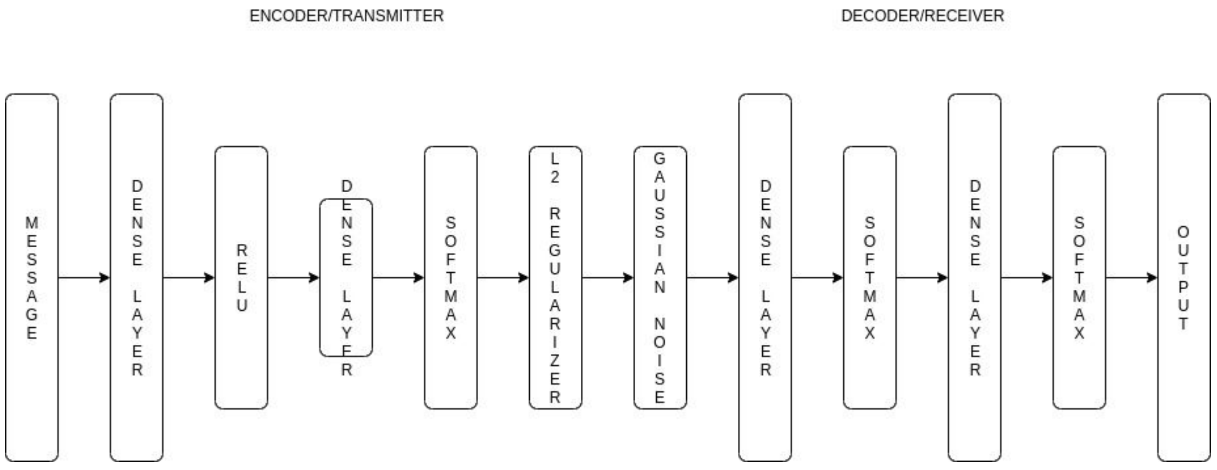

Fig 1:Autoencoder architecture

TRAINING:

We give the training data for both input and output and train for 100 epochs with batch size 16.

TESTING:

For testing, we evaluate the block error rate for different SNR. We take 1000 test data samples. We then evaluate all of them for SNR ranging from -15 to 15. We compare the predicted output with actual message signal. Block Error Rate (BER) is errors in predicted and actual message signal by total number of input messages. We plot the graph of SNR vs BER.

RESULTS:

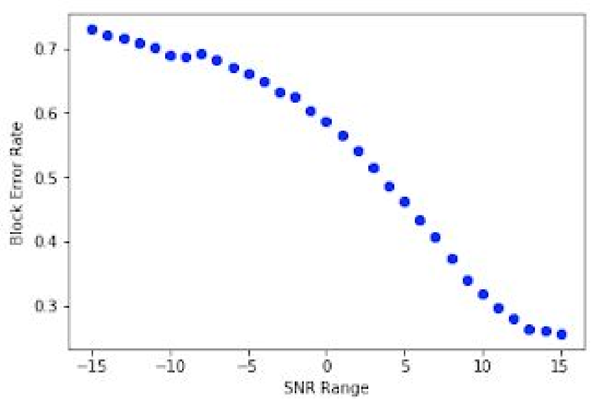

Fig 2: SNR vs BER

The graph shows that as SNR increases, the BER decreases. The relation between SNR and BER is curvilinear instead of linear. The graph shows the SNR vs BER curve which is similar to the ones obtained from traditional systems.
The different SNR and BER values are given as:

SNR: -5 BER: 0.661
SNR: -4 BER: 0.649
SNR: -3 BER: 0.633
SNR: -2 BER: 0.624
SNR: -1 BER: 0.603
SNR: 0 BER: 0.586
SNR: 1 BER: 0.565
SNR: 2 BER: 0.54

SNR: 3 BER: 0.514
SNR: 4 BER: 0.485
SNR: 5 BER: 0.462

The BER can be further increased, by using strong regularizer, increasing the batch size, etc

REFERENCES:

1. An Introduction to Deep Learning for the Physical Layer, Tim O’Shea, Senior Member, IEEE, and Jakob Hoydis, Member, IEEE