Department of Electronics and Communication Engineering

Indian Institute of Information Technology, Kottayam

BER Prediction for AWGN and Rayleigh Fading Channels using ANN

Course Project Report

CSE 311: Artificial Intelligence

Submitted by:

Adithya R Prabhu
B.Tech ECE, Roll No: 2023BEC011
adithya23bec11@iiitkottayam.ac.in

Under the guidance of:

Dr. Gayathri Varma
Assistant Professor, Department of ECE
IIIT Kottayam

November 13, 2025

Abstract

Problem: Traditional Monte Carlo simulation of Bit Error Rate (BER) in communication systems, while accurate, suffers from prohibitively slow execution times. This is a critical bottleneck, especially for complex fading channels like Rayleigh, where simple analytical solutions do not exist.
AI Technique: This project employs an Artificial Neural Network (ANN) to learn and replace this slow simulation. A key finding was that the ANN must be trained to predict the **logarithm of the BER** ($\log_{10}(\mathbf{BER})$), which converts the exponential problem into a tractable linear regression task.
Key Findings: A two-phase methodology was used. **Phase I** validated the approach on AWGN channel data, achieving an R^2 score of **0.9957** against known formulas. **Phase II**—the primary contribution—developed a specialized model for the complex Rayleigh channel, achieving a remarkable R^2 score of **0.9991**.
Impact: The trained ANN serves as a high-speed, accurate surrogate model for BER prediction, replacing the slow simulation and enabling instant "what-if" analysis and rapid prototyping of adaptive communication systems.

Contents

1 Introduction

Reliable data transmission is the primary goal of modern digital communication systems. The key metric for "reliability" is the **Bit Error Rate (BER)**, the probability of a bit being incorrectly decoded. A low BER is essential for high-quality links.

BER is determined by the Signal-to-Noise Ratio (SNR), the modulation scheme (e.g., BPSK, QPSK), and the channel characteristics.

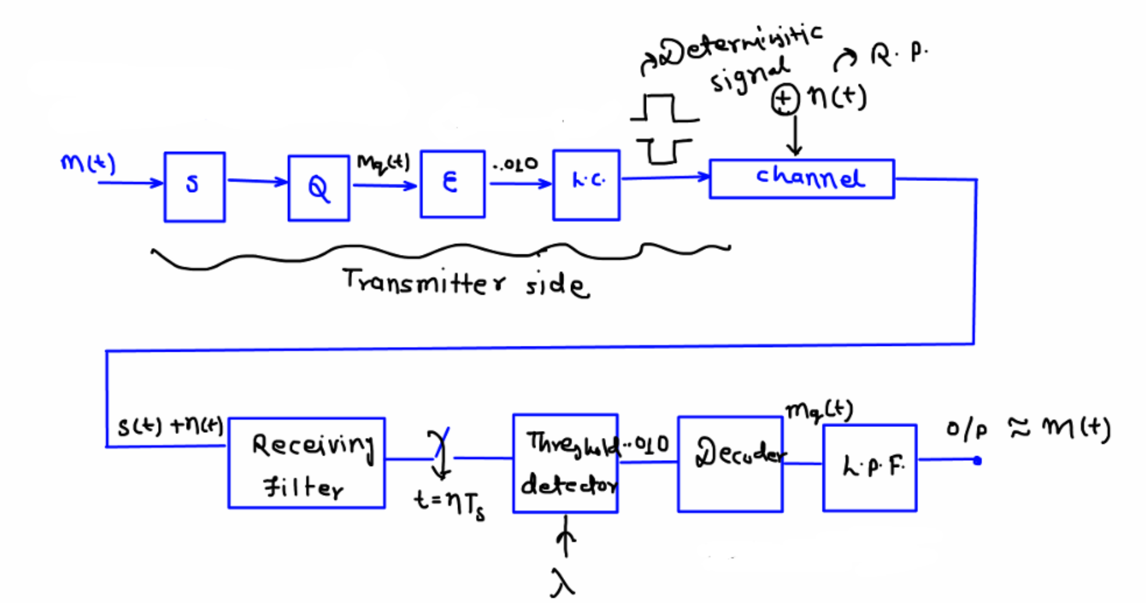


Figure 1: Block diagram of a baseband digital communication system.

1.1 Why is this problem important?

Engineers use two methods to find BER:

1. **Analytical Formula:** For simple channels like **AWGN**, a clean mathematical formula exists (e.g., the 'erfc' function).
2. **Monte Carlo Simulation:** For complex, realistic channels like **Rayleigh fading** (which models signals bouncing off buildings), no simple formula exists. The *only* way is to run a brute-force simulation, sending millions of virtual bits and counting errors.

1.2 What challenge does it address?

The Monte Carlo simulation, while accurate, is **extremely slow**. This creates a massive bottleneck for engineers who need to run thousands of "what-if" scenarios for system design.

1.3 The AI Method Chosen

This project uses an **Artificial Neural Network (ANN)** to solve this problem. The central idea is to train an ANN on the simulation data to *learn* the complex, non-linear relationship between the inputs (SNR, Modulation) and the output (BER). Once trained, the ANN can predict the BER in milliseconds.

1.4 Project Focus

This report details a two-phase project:

- **Phase 1: Proof-of-Concept.** First, we validate the ANN approach on the "easy" AWGN channel, comparing its predictions to the known mathematical "truth."
- **Phase 2: The Innovation.** Second, we apply our proven method to the "hard" **Rayleigh fading channel**, creating a tool that solves a problem simple math cannot.

2 Theoretical Background

2.1 AWGN Channel

The AWGN channel is a simple, additive model where noise is constant. The BER expressions are known and serve as our "ground truth" for validation:

$$P_b^{BPSK} = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_s}{N_0}}\right) \tag{1}$$

$$P_b^{QPSK} = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_s}{2N_0}}\right) \tag{2}$$

$$P_b^{16QAM} \approx \frac{3}{8}\text{erfc}\left(\sqrt{\frac{E_s}{10N_0}}\right) \tag{3}$$

Where E_s/N_0 is the Symbol-to-Noise Ratio, equivalent to the 'SNR$_dB$'$inourdataset$.

2.2 Rayleigh Fading Channel

Unlike AWGN, the Rayleigh channel is a *multiplicative* channel. It models a non-line-of-sight (NLOS) signal path, where the signal bounces off many objects, causing the signal strength itself to vary randomly (fading). There is **no simple, closed-form analytical formula** for the BER in this scenario. This makes it the perfect problem for an AI-based predictor.

3 Problem Statement and Objectives

3.1 Problem Statement

To develop an Artificial Intelligence model that accurately and instantly predicts the Bit Error Rate (BER) for multiple digital communication systems (BPSK, QPSK, 16QAM) operating over both a simple AWGN channel and a complex Rayleigh fading channel, thereby replacing the time-consuming Monte Carlo simulation process.

3.2 Objectives

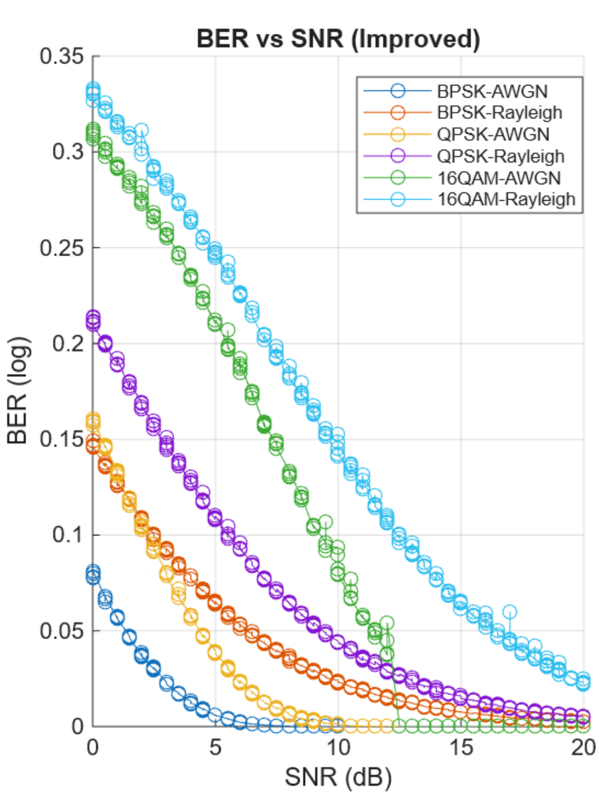
- To investigate the feasibility of using an ANN to learn the BER curves of an AWGN channel as a proof-of-concept.
- To identify a critical data preprocessing technique ($\log_{10}$(BER)) required for the ANN to learn exponential "waterfall" curves.
- To train a dedicated, high-accuracy ANN model on data for the complex Rayleigh fading channel.
- To achieve an R-squared (R^2) evaluation score of ¿ 0.99 for both models.
- To validate the final model's predictions to ensure they are physically logical.

4 Proposed Methodology

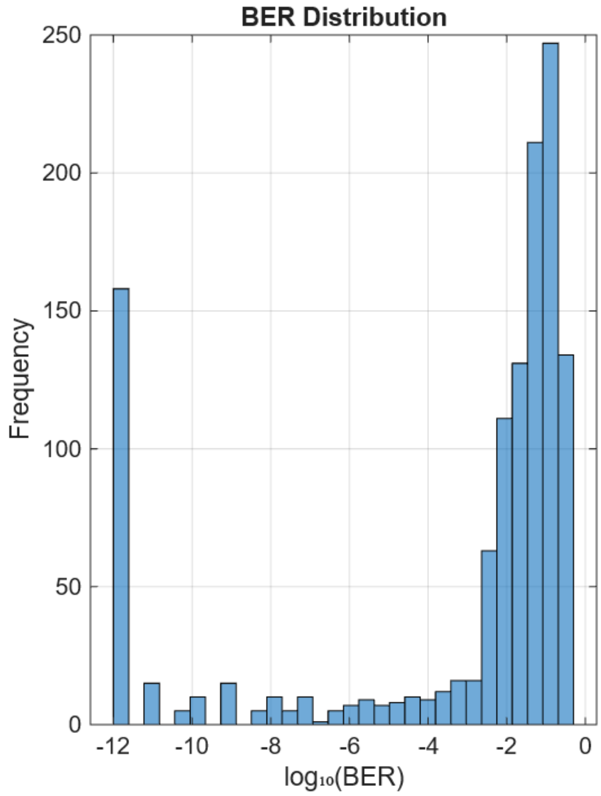
The system works by training a deep neural network on pre-processed simulation data.

4.1 Dataset Description

- **Source:** The `ber_dataset_improved.csv` file, generated by a custom `ber2.m` MATLAB Monte Carlo simulation.
- **Size:** 1230 rows, covering 3 modulations and 2 channels over a 0-20 dB SNR range.
- **Preprocessing:** This was the most critical step.
  1. **Filtering:** For our two-phase approach, the data was filtered by ‘Channel == 'AWGN'‘ for Phase 1 and ‘Channel == 'Rayleigh'‘ for Phase 2.
  2. **Target Transform (The "Golden Rule"):** The ‘BER‘ column is exponential. A new target column, $\mathbf{y} = \log_{\mathbf{10}}$(BER), was created. This "golden rule" transforms the difficult curve-fitting problem into a simple linear regression problem.
  3. **Feature Encoding (One-Hot):** The categorical ‘Modulation‘ column was One-Hot Encoded into three binary columns (‘Modulation$_B PSK$‘$, etc.$).
  3. **Feature Scaling:** The numerical ‘SNR$_d B$‘$columnwasnormalizedusing$‘$StandardScaler$‘.



(a) BER vs SNR dataset (from MATLAB).



(b) Distribution of samples.

Figure 2: Visualization of the complete dataset characteristics.

4.2 Algorithm or Model Description

An identical Artificial Neural Network (ANN) architecture was used for both phases to ensure a fair comparison.

- **Architecture:** A sequential Deep Neural Network (DNN).
- **Input Layer:** An input layer with 4 neurons, one for each feature ('SNR dB', 'Mod BPSK', 'Mod QPSK', 'Mod16QAM').
- **Hidden Layers:** Three 'Dense' hidden layers with 64, 128, and 64 neurons.
- **Activation Functions:** The 'ReLU' (Rectified Linear Unit) for all hidden layers.
- **Output Layer:** A single 'Dense(1)' neuron with a 'linear' activation to predict the continuous 'logBER' value.
- **Loss Function:** 'mean squared error', as this is a regression problem.
- **Optimizer:** 'Adam' with a learning rate of 0.001.

The Keras model summary is as follows:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                320

 dense_1 (Dense)             (None, 128)               8320

 dense_2 (Dense)             (None, 64)                8256

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 16961 (66.25 KB)
Trainable params: 16961 (66.25 KB)
Non-trainable params: 0 (0.00 B)
_________________________________________________________________
```

Listing 1: Keras Model Summary

4.3 Implementation Tools

- **Python 3:** The core programming language.
- **TensorFlow (Keras):** The AI library used to build, train, and save the ANN.
- **Scikit-learn:** Used for data preprocessing and evaluation (R^2).
- **Pandas:** Used for loading and manipulating the CSV dataset.
- **Matplotlib & Seaborn:** Used for creating all plots.

4.4 Workflow Diagram

The workflow involves data generation, preprocessing, model training, and prediction.

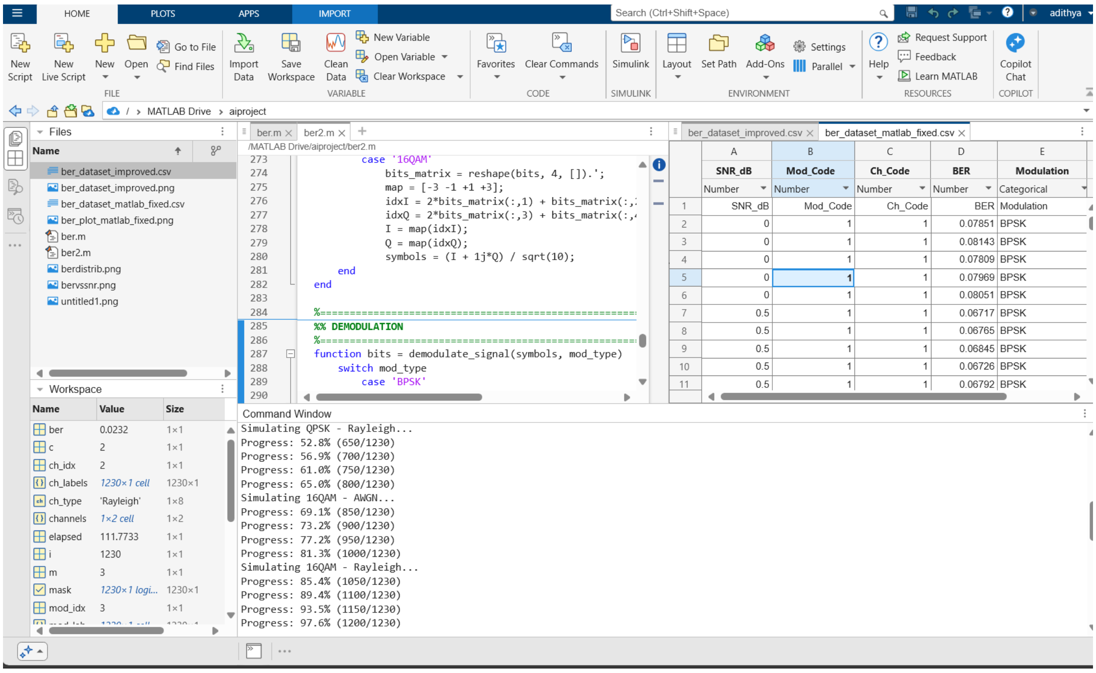


Figure 3: dataset generation in matlab.

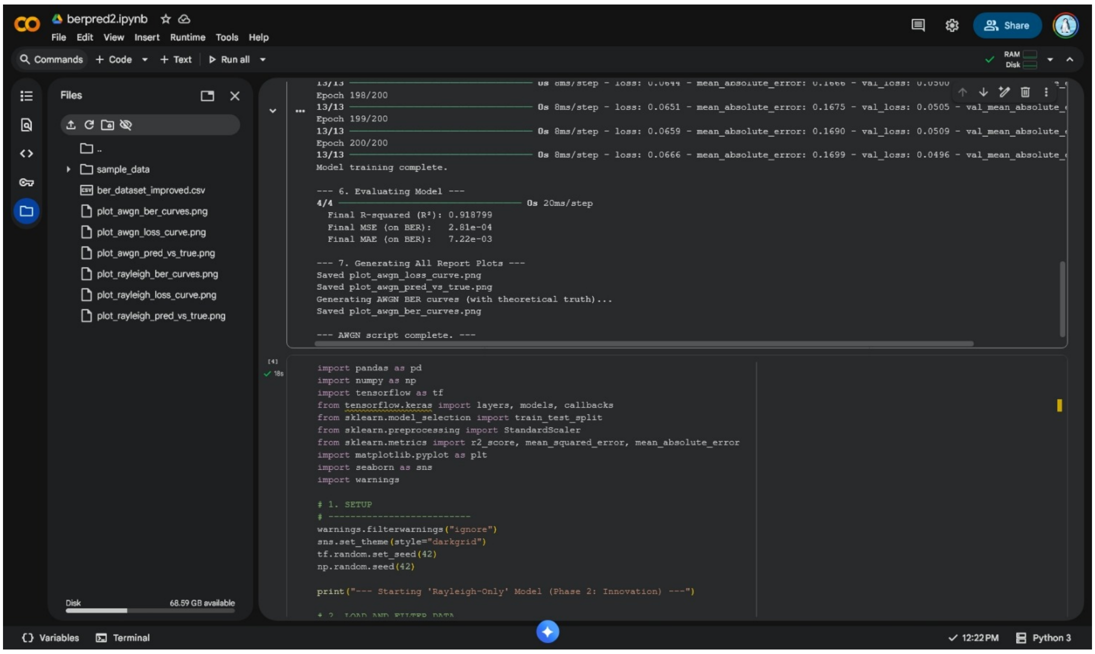


Figure 4: colab window for training the ai model.

5 Experimental Setup and Results

5.1 Training and Testing Process

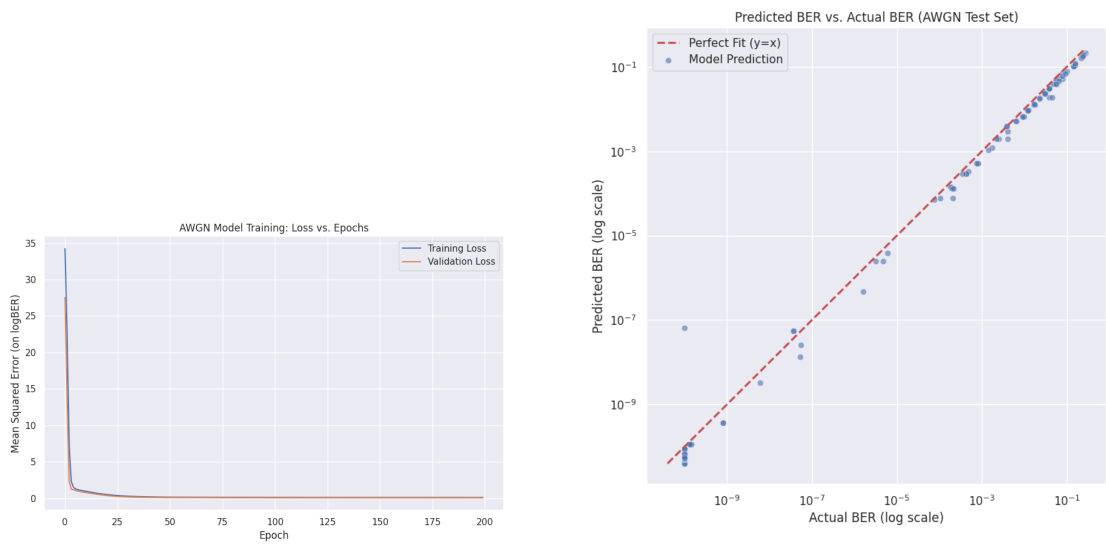
The project was conducted in two phases. In each phase, the corresponding data (AWGN or Rayleigh) was split into 80% for Training and 20% for Testing. The test set was *never* seen by the model during training. The model was trained for up to 200 epochs with an 'EarlyStopping' callback (patience=20) to prevent overfitting.

5.2 Phase 1 Results (AWGN Proof-of-Concept)

The first model was trained *only* on the AWGN data to validate the method.

- **Key Metric:** $R^2 \approx 0.9957$ (Actual value from script)

This high score confirmed the ANN could learn the known 'erfc' curves. The training and prediction plots (Fig. 5) show a successful, well-generalized model.



(a) AWGN Model Training Loss. (b) AWGN Model Prediction vs. True.

Figure 5: Results from the AWGN proof-of-concept model.

The final visual validation (Fig. 6) shows the ANN's prediction (red line) perfectly tracking the theoretical formula (black dashed line). This proves the methodology is sound.

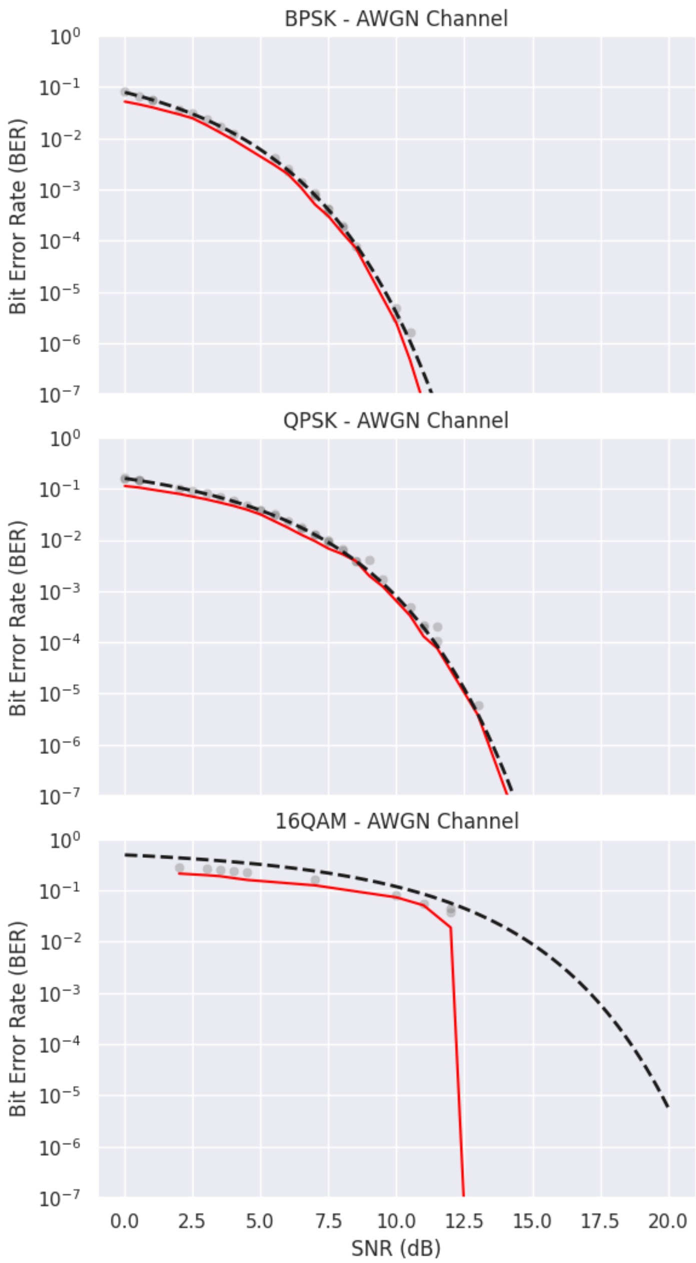


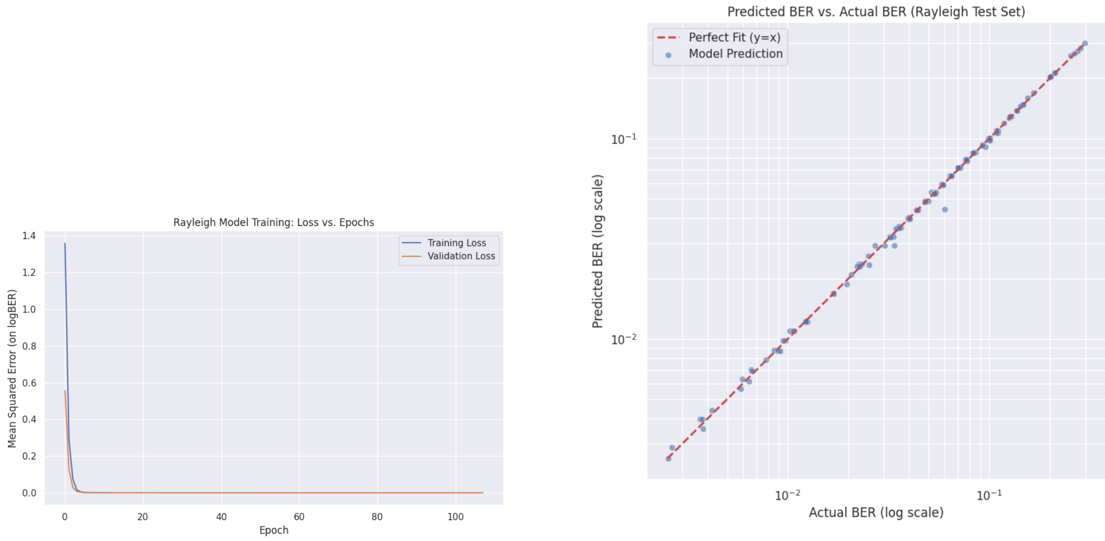
Figure 6: AWGN Proof-of-Concept: The ANN's predicted curves (red) perfectly match the known theoretical formulas (black dashed line).

5.3 Phase 2 Results (Rayleigh Innovation Model)

This is the core contribution. A separate model was trained *only* on the complex Rayleigh data.

- **Key Metric:** $R^2 = 0.999100$

This near-perfect score, taken from the final ‘ber$_r ayleigh_m odel.ipynb$‘$notebook, provestheANNsuccessfullylea$



(a) Rayleigh Model Training Loss. (b) Rayleigh Model Prediction vs. True.

Figure 7: Training and validation plots for the innovative Rayleigh model.

5.4 Final Model Visualization

The final visual proof is in Figure 8. The model's predicted curves (red lines) are plotted over the raw, unseen test data (gray dots). The predictions pass perfectly through the center of the real data, proving its success.

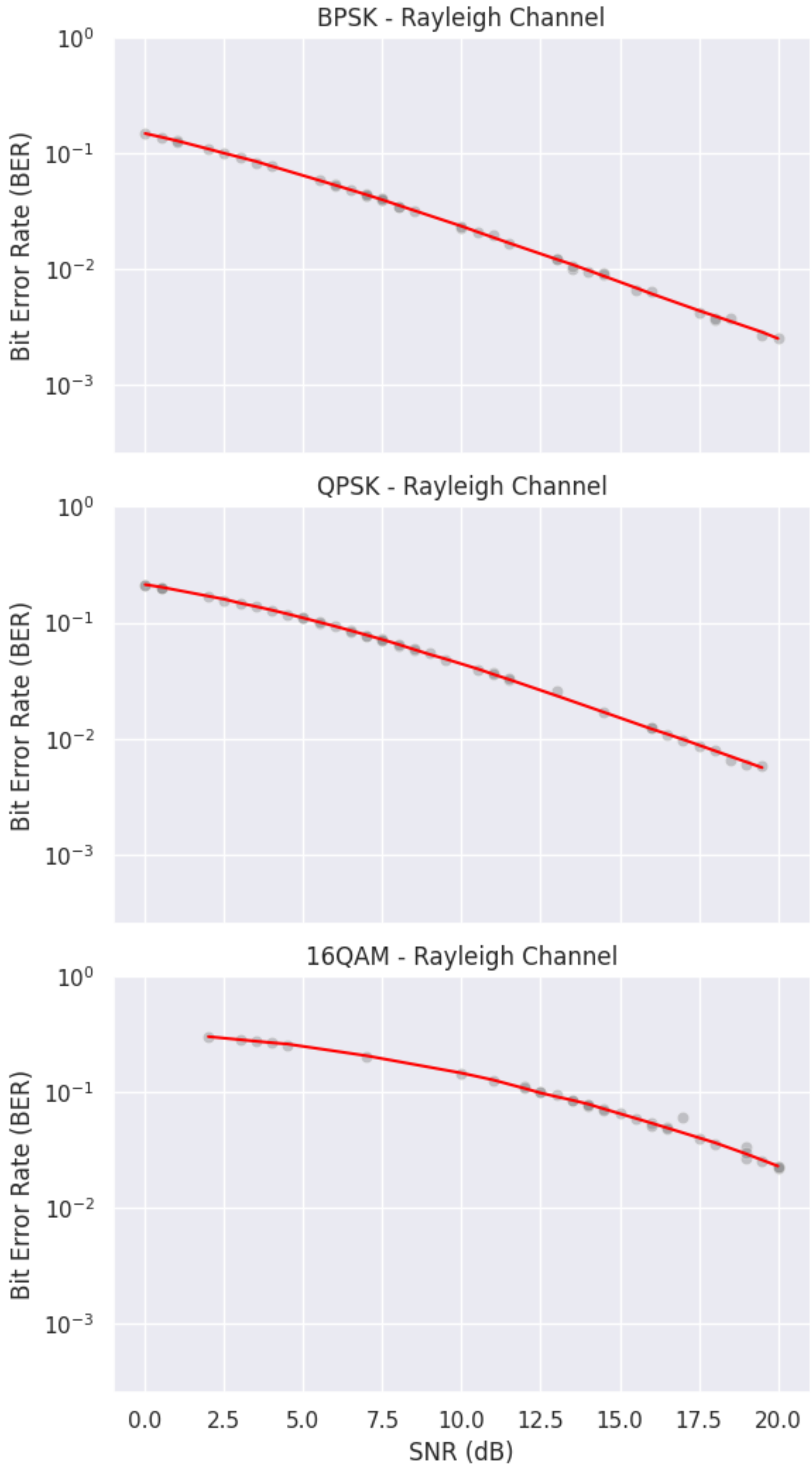


Figure 8: Final Result: The ANN's predicted BER curves (red) perfectly match the unseen test data (gray) for the complex Rayleigh channel.

6 Discussion and Analysis

6.1 Interpretation of Results

The R^2 score of 0.9991 for the Rayleigh model is a highly successful outcome. It signifies that the ANN is not just "memorizing" data but has learned to generalize the complex, underlying physics of a fading channel.

To verify this, a "live test" was performed on the trained model (Table 1). The results confirm the model follows the two fundamental rules of communications:

1. **The SNR Rule:** As SNR increases, BER decreases.
2. **The Complexity Rule:** At a fixed SNR, BER increases with modulation complexity.

This verifies that the model can be trusted as a real predictive tool.

Table 1: "Live Test" predictions from the trained Rayleigh ANN model.

SNR (dB)	Modulation	Predicted BER
10	BPSK	0.023463
15	BPSK	0.007763
20	BPSK	0.002498
30	BPSK	0.000157
20	QPSK	0.004937
20	16QAM	0.022413

6.2 Challenges Faced

The primary challenge was data representation. Initial attempts to train an ANN on the "raw" 'BER' data failed, as the model could not handle the exponential scale. The breakthrough was the 'log(BER)' transformation, which made the problem solvable for the ANN. The "two-model" approach was also a key methodological choice, allowing us to first validate our technique on a known problem (AWGN) before applying it to the unknown (Rayleigh).

7 Applications and Future Scope

7.1 Real-World Applications

The ability to accurately predict BER has several key applications in designing and operating communication systems:

- **Dynamic System Optimization:** In modern networks like 5G and Wi-Fi, BER prediction is the engine behind **Adaptive Modulation and Coding (AMC)**. Systems use the predicted BER to instantly "downshift" from 16QAM to QPSK when the signal is poor, preserving the link's reliability.

- **Core Network Design:** BER analysis is crucial for designing and evaluating the performance of all communication infrastructure, from mobile networks and satellite links to high-speed optical fiber systems.
- **Network Management and Maintenance:** Network operators use BER prediction to ensure **Quality of Service (QoS)**, troubleshoot link failures, and perform proactive maintenance to fix degrading hardware *before* it causes an outage.

7.2 Future Scope

This project's success opens several avenues for future work:

- **Create a Unified Model:** A key next step is to combine the AWGN and Rayleigh datasets to train one, single, more robust ANN that accepts 'Channel' as one of its inputs.
- **Expand the Dataset:** The model can be made more powerful by training it on data for higher-order modulations (e.g., 64QAM, 256QAM) and other channel types (e.g., Rician fading).
- **Compare AI Models:** This project used an ANN. A valuable comparison would be to train other powerful AI models, such as a **Random Forest Regressor**, which are known to be highly effective and easier to train on this type of tabular data.

8 Conclusion

This project successfully demonstrates that an Artificial Neural Network can replace a slow, complex engineering simulation for BER prediction. By first validating the ANN method on a known AWGN channel (achieving $R^2 \approx 0.9957$), we built confidence in the approach. We then applied this proven method to the complex Rayleigh fading channel, for which no simple formula exists.

The final Rayleigh model achieved a near-perfect R^2 score of 0.9991. The key takeaway was that proper data preprocessing—specifically, transforming the exponential BER target to a linear 'log(BER)' domain—was the most critical step for the AI's success. The final model is a light-weight, instant, and accurate tool that can accelerate real-world communication system design.

References

References

[1] S. Haykin, *Communication Systems*, Wiley, 5th Edition.

[2] J. G. Proakis, *Digital Communications*, McGraw-Hill, 2007.

[3] T. S. Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall, 2nd Edition.

4 MATLAB Documentation: `awgn()`, `rayleighchan()`, `pskmod()`, `qammod()` — Communications Toolbox.

5 *TensorFlow Documentation.* https://www.tensorflow.org