Project Title: Learning and Prediction of OTFS Channel Coefficients using a Variational Autoencoder

1. Introduction & Motivation:

Accurate and timely Channel State Information (CSI) is crucial for optimizing the performance of Orthogonal Time Frequency Space (OTFS) communication systems, especially in dynamic wireless environments. Traditional channel estimation methods can be computationally intensive or suffer from pilot overhead. This project explores the use of Variational Autoencoders (VAEs), a powerful deep generative model, to learn the complex relationships between fundamental channel properties and the resulting OTFS channel coefficients. By leveraging the VAE's ability to create a structured latent representation, we aim to predict CSI more efficiently and robustly, paving the way for AI-driven adaptive transceivers.

2. Project Goal:

The primary goal of this project is to design, implement, train, and evaluate a Variational Autoencoder (VAE) model in Python using a deep learning framework (TensorFlow/PyTorch). The VAE will learn to predict OTFS Delay-Doppler channel coefficients based on a set of input channel properties/features from a pre-generated dataset.

3. Specific Objectives:

Upon successful completion of this project, the student will be able to:

- Familiarize themselves with the theoretical foundations of Variational Autoencoders (VAEs), including their architecture, loss functions (reconstruction loss and KL divergence), and generative capabilities.
- Understand how to map diverse channel features (e.g., Path Loss, LOS/NLOS status, SNR, Doppler spread) to OTFS Delay-Doppler channel coefficients using a neural network.
- Design and implement a suitable VAE architecture for complex-valued channel coefficient prediction.
- Preprocess and prepare the input dataset (channel features and coefficients) for VAE training.
- Train the VAE model effectively, tune hyperparameters, and monitor its convergence.
- Evaluate the performance of the trained VAE in predicting OTFS channel coefficients using relevant metrics (e.g., Normalized Mean Squared Error (NMSE)).

• Analyze the learned latent space to gain insights into how the VAE represents different channel states.

4. Methodology & Tasks:

1. Literature Review (Weeks 1-3):

• Study the principles of Variational Autoencoders (VAEs) and Conditional VAEs (CVAEs).

• Review existing deep learning approaches for wireless channel estimation and prediction.

• Understand the structure of the provided OTFS channel dataset.

2. Dataset Preparation (Weeks 4-6):

• Load the provided channel dataset (features and coefficients) into Python.

• Implement necessary preprocessing steps: normalization/scaling of channel features, handling of complex-valued channel coefficients (e.g., separate real/imaginary parts or use complex layers if available).

• Split the dataset into training, validation, and test sets.

3. VAE Architecture Design (Weeks 7-10):

• Design the encoder and decoder neural network architectures. Consider using:

- Encoder: Dense layers or Convolutional Neural Networks (CNNs) to process channel features and output the mean (μ) and variance (σ^2) vectors for the latent space.

- Decoder: Dense layers or Transposed CNNs to generate complex-valued OTFS channel coefficients from a sampled latent vector z.

• Determine appropriate activation functions, layer sizes, and the dimension of the latent space.

• Consider implementing a Conditional VAE (CVAE), where some or all input features are also fed directly to the decoder to guide generation.

4. VAE Implementation (Weeks 11-14):

• Implement the VAE model using a deep learning framework (TensorFlow or PyTorch).

• Define the VAE loss function, combining the reconstruction loss (e.g., Mean Squared Error for real/imaginary parts of coefficients) and the Kullback-Leibler (KL) divergence term.

• Set up the training loop, including optimizer selection, learning rate scheduling, and batching.

5. Training and Hyperparameter Tuning (Weeks 15-16):

- Train the VAE model on the prepared dataset.
- Experiment with different hyperparameters (e.g., latent space dimension, learning rate, batch size, number of layers, network complexity) to optimize performance.
- Monitor training and validation loss for convergence and overfitting.

6. Performance Evaluation & Analysis (Weeks 17-18):

- Evaluate the trained VAE's prediction accuracy on the unseen test set using metrics like Normalized Mean Squared Error (NMSE) between predicted and actual channel coefficients.
- Visually compare predicted channel coefficients (e.g., magnitude and phase plots over delay-Doppler grid) against ground truth.
- Analyze the learned latent space: use dimensionality reduction techniques (e.g., t-SNE, PCA) to visualize the distribution of different channel states in the latent space and observe how they relate to the input features.

5. Expected Outcomes:

- A well-documented Python codebase implementing the VAE/CVAE model for OTFS channel coefficient prediction.
- A trained VAE model capable of generating channel coefficients from input channel features.
- A comprehensive performance analysis report detailing the model's accuracy, training process, and insights into the latent space.
- A written bachelor thesis detailing the methodology, implementation, evaluation, and conclusions.
- A presentation summarizing the project's findings.

6. Required Skills:

- Proficiency in Python programming.
- Experience with deep learning frameworks (TensorFlow or PyTorch).
- Basic understanding of machine learning concepts (neural networks, training, validation).
- Basic understanding of wireless communication fundamentals and OTFS.

- Familiarity with data manipulation libraries (NumPy, Pandas) and plotting (Matplotlib).