Project Name

Reproducing Dual-Domain Joint Encoder (DDJE) for Speech Separation

Project Background

This project aims to reproduce the **Dual-Domain Joint Encoder (DDJE)** proposed in the paper *Improved Speech Separation via Dual-Domain Joint Encoder in Time-Domain Networks*. The model leverages both **time-domain and frequency-domain features** to enhance speech separation accuracy and robustness.

Objectives

1. **Understand the paper**, including the DDJE structure and training methodology.
2. **Prepare datasets**, selecting suitable open-source speech separation datasets (e.g., WSJ0-2mix, LibriMix).
3. **Implement the Dual-Domain Joint Encoder (DDJE) model**, including:
    - **Time-Domain Encoder**
    - **Frequency-Domain Encoder**
    - **Feature Fusion Module**
    - **Speech Separation Network**
4. **Train the model** using **Permutation Invariant Training (PIT)** and optimize hyperparameters.
5. **Evaluate performance** based on SDR, SI-SDR, PESQ, and compare results with Conv-TasNet and DPRNN baselines.
6. **Deploy the model**, providing inference support for real-time speech separation.

Implementation Steps

Step 1: Literature Review and Technical Research

- Analyze the DDJE model structure and training techniques.
- Study related works such as Conv-TasNet and DPRNN.
- Decide on the deep learning framework: **PyTorch (preferred) or TensorFlow**.

Step 2: Dataset Preparation

- Select datasets: **WSJ0-2mix**, **LibriMix**.
- Preprocess data:
    - Normalize audio files and resample if needed.
    - Compute **Short-Time Fourier Transform (STFT)** for frequency-domain features.
    - Generate training samples (mixture and target speech).

Step 3: Model Implementation

- **Time-Domain Encoder**: CNN + LSTM/Transformer.
- **Frequency-Domain Encoder**: STFT transform + CNN.
- **Feature Fusion**: Cross-attention or gated mechanism.
- **Speech Separation Network**: Integrate DDJE with **TasNet/DPRNN**.

Step 4: Model Training and Optimization

- Use **Permutation Invariant Training (PIT)** to handle speaker permutations.
- Define loss functions:
  - **SI-SDR Loss**
  - **MSE Loss**
- Set hyperparameters:
  - Optimizer: **AdamW**
  - Learning rate schedule: **Cosine Annealing / Warm-up**
  - Batch size: **16-32**
  - Epochs: As required for convergence.

Step 5: Model Evaluation and Comparison

- Metrics:
  - **Signal-to-Distortion Ratio (SDR)**
  - **Scale-Invariant SDR (SI-SDR)**
  - **Perceptual Evaluation of Speech Quality (PESQ)**
- Compare results with **Conv-TasNet, DPRNN**.

Step 6: Model Deployment

- Export trained weights.
- Optimize inference using **ONNX / TensorRT**.
- Develop an inference interface to test real-world speech separation.

Technology Stack

- **Programming Language**: Python
- **Deep Learning Framework**: PyTorch / TensorFlow (PyTorch recommended)
- **Audio Processing**: Librosa, PyDub
- **Training Framework**: PyTorch-Lightning
- **Evaluation Tools**: mir_eval, PESQ, torchmetrics
- **Deployment Tools**: ONNX, TensorRT

Expected Deliverables

1. **Fully reproduced DDJE model** with successful training on WSJ0-2mix or LibriMix.
2. **Achieve competitive speech separation performance** (SDR > 10dB).
3. **Provide reproducible PyTorch code** for training, inference, and evaluation.
4. **Deploy a working model**, capable of real-time speech separation.