**Noise Reduction using Deep Learning project.**

**High Level Design**

1. Input: Noisy Audio Data

- Data Sources: Gather a dataset containing paired noisy and clean audio samples. Popular datasets include:

- VoiceBank-DEMAND: Noisy speech corpus with corresponding clean speech samples.

- Noisy Speech Database: A variety of environmental noises added to speech samples.

- Input Format:

- Audio is typically represented as a waveform, but for deep learning models, it is often transformed into a spectrogram or Mel-spectrogram for better feature extraction.

2. Preprocessing Layer

- Audio Preprocessing:

- Resampling: Ensure all audio samples have the same sampling rate (e.g., 16kHz).

- Frame Splitting: Split long audio files into smaller frames (e.g., 1-second or 5-second chunks).

- Normalization: Normalize the audio signals to standardize amplitude.

- Feature Extraction:

- Convert the audio waveform into a log-Mel spectrogram or MFCC (Mel-frequency cepstral coefficients).

- Spectrogram extraction helps the model better understand frequency information over time, essential for reducing noise.

3. Model Architecture

- Pre-Trained Model: Use a pre-trained model, such as:

- Wav2Vec 2.0: Pre-trained on audio data and suitable for speech processing tasks.

- UNet for Audio: An encoder-decoder architecture often used for image and audio denoising.

- Model Modification:

- Encoder: Extract high-level features from noisy audio spectrograms.

- Decoder: Reconstruct the clean audio from the learned features by denoising the spectrogram.

- Fine-Tuning:

- Freeze the early layers (encoder) of the pre-trained model to retain learned features.

- Fine-tune the later layers (decoder) on the new noise reduction task using your noisy-clean audio pairs.

- Loss Function:

- Use a reconstruction loss function such as Mean Squared Error (MSE) between the predicted clean audio and the actual clean audio.

- Optionally, add perceptual loss (e.g., STOI or PESQ) to ensure the audio is perceptually better.

4. Training Pipeline

- Input: Feed noisy spectrograms into the model.

- Output: The model will output a cleaned spectrogram.

- Training:

- Use paired noisy and clean audio samples to supervise the learning.

- Use augmentation techniques like adding synthetic noise to ensure model robustness.

- Batching & Epochs:

- Use mini-batch gradient descent with small batches of audio samples.

- Train the model over multiple epochs until convergence (monitored by validation loss).

5. Post-Processing Layer

- Spectrogram Inversion: Convert the cleaned spectrogram back to a waveform using Griffin-Lim Algorithm or inverse short-time Fourier transform (ISTFT).

- Smoothing: Apply smoothing techniques to reduce artifacts introduced by the denoising process.

- Output: Generate the clean waveform as the final output audio.

6. Evaluation and Metrics

- Objective Evaluation Metrics:

- Signal-to-Noise Ratio (SNR): Measures the level of noise reduction.

- Perceptual Evaluation of Speech Quality (PESQ): Evaluates speech quality.

- Short-Time Objective Intelligibility (STOI): Measures intelligibility improvement.

- Subjective Evaluation:

- Perform listening tests to evaluate how clear the cleaned audio sounds to humans.

7. Deployment Pipeline

- User Interface:

- Design a simple interface (e.g., a web app using Flask or Streamlit) that allows users to upload noisy audio files and download the denoised output.

- Model Serving:

- Deploy the model using a framework like TensorFlow Serving or PyTorch Serve.

- Ensure scalability if the system will be used in real-time or for a larger user base.

- API:

- Create a REST API for the model where the client sends a noisy audio file, and the server returns the cleaned audio.

8. Workflow Overview

1. User Input: User uploads a noisy audio file (e.g., voice recording with background noise).

2. Preprocessing: Convert audio to spectrogram, normalize, and split into frames.

3. Model Inference: Feed the noisy spectrogram into the fine-tuned model.

4. Post-Processing: Convert the cleaned spectrogram back to an audio waveform and apply smoothing.

5. User Output: User downloads the cleaned audio file.

This high-level design focuses on using a pre-trained model for noise reduction.