# Technical Report

**Task**: Train a Low-Latency Speech Enhancement Network for Edge Devices

## Overview

In this project, I designed and implemented a real-time, low-latency speech enhancement model using PyTorch, optimized for streaming applications such as hearing aids, voice calls, and smart assistants. The system enhances noisy audio on the fly by processing audio in 4ms chunks, leveraging a Transformer-based architecture to model long-term dependencies while maintaining causal constraints. The design emphasizes both algorithmic efficiency and audio quality to meet the stringent requirements of edge deployments.

## Design Considerations

### Model Choice

Speech enhancement is inherently a time-series modeling problem. Traditional architectures like CNNs and RNNs (including LSTMs) have been applied in this domain, but they suffer from inherent limitations. CNNs have restricted temporal context due to their fixed kernel sizes, while LSTMs can struggle with vanishing gradients and limited parallelism during training.

Transformers, with their self-attention mechanism, offer a compelling alternative due to their ability to model long-range dependencies and parallelizable computations. Recent advancements show that even in audio and time-series domains, Transformers outperform conventional methods in both accuracy and stability.

### Preprocessing and Architectural Adaptation

Feeding raw audio samples directly into Transformer layers is suboptimal, as raw waveforms are highly sparse and lack a structured representation. To address this, I employed a **Conv1D encoder** to project the raw audio into a higher-dimensional latent space (with d_model=32). This transformation enriches the representation and makes it suitable for Transformer-based learning. At the output stage, a **Conv1D decoder** maps the transformed features back to the waveform domain.

### Context Management for Streaming Inference

To ensure temporal continuity and reduce perceptual artifacts, the model maintains short-term memory across inference windows:

1. **Conv1D Encoder Context**: Retains K-1 past raw audio samples to provide local continuity during feature extraction.

2. **Transformer Context**: Maintains N past encoded chunks to allow the Transformer to operate over an extended receptive field.

3. **Conv1D Decoder Context**: Retains K-1 samples from previous Transformer outputs to ensure seamless stitching of the enhanced waveform.

This architectural memory enables streaming-friendly inference with zero lookahead and no future dependency, achieving 4ms algorithmic latency, making the model highly suitable for real-time and edge-based applications.

### Loss Function

The objective of speech enhancement is to produce audio that closely matches the clean ground truth. Initially, I experimented with **mean squared error (MSE)** loss, which is theoretically aligned with maximizing signal-to-noise ratio (SNR). However, due to vanishing gradient issues near convergence, I transitioned to using **L1 loss**, which offers stronger gradients during late-stage training and is widely adopted in speech enhancement literature for its robustness and stability.

## Model Architecture

- **Conv1D Encoder**: Transforms each 64-sample raw audio chunk into a 32-dimensional latent representation using a 1D convolutional layer with a kernel size of 5.

- **Transformer Encoder**: Consists of 4 layers with 2 attention heads, operating on both the current and previous latent chunks to model temporal dependencies via self-attention.

- **Conv1D Decoder**: Maps the Transformer output back to the waveform domain using a 1D transposed convolution layer with a kernel size of 5.

This lightweight, modular architecture balances efficiency and effectiveness, enabling real-time processing with only **40K parameters**, making it well-suited for deployment on resource-constrained edge devices.

## Dataset Construction

I created a custom dataset using a 2-hour subset of the **LibriSpeech** corpus. To simulate real-world noise conditions, I synthetically added:

- Environmental sounds (e.g., babble, restaurant, subway, sidewalk, children playing)
- Additive white Gaussian noise (AWGN)

Noise levels were varied between -5dB and +10dB SNR. The paired clean-noisy dataset was used to train the model over 5 epochs using a batch size of 8 under constrained compute resources.

## Future Directions

This work lays a strong foundation for real-time, low-latency speech enhancement on edge devices. Future improvements may include:

1. **Scaling to Larger and Diverse Datasets:** Expanding training on more extensive and environmentally diverse datasets, including real-world recordings, will improve generalization and robustness.
2. **Perceptual Optimization:** Integrating perceptual loss functions such as PESQ, SI-SNR, or STOI can help align model outputs more closely with human auditory perception, improving subjective quality.
3. **Model Compression:** Exploring quantization, pruning, and knowledge distillation can significantly reduce memory and compute requirements while preserving performance, ideal for deployment on resource-constrained devices.
4. **Architectural Search:** Exploring lightweight architectural variants, including efficient Transformer hybrids, neural architecture search (NAS), and dilated

convolutions, may lead to better performance in terms of latency.

5. **Extended Training Time:** The current model was trained for a limited number of epochs due to computational constraints. Validation loss was still decreasing at the end of training, suggesting further improvements are likely with extended training.