Speech Enhancement using Deep Convolutional Neural Network

Student: Trần Anh Quân
Instructor: Dr. Phan Duy Hùng

Bachelor of Computer Science
Hoa Lac Campus - FPT University
4/5/2021

Outline

1. Introduction
2. Data Collecting and Preprocessing
3. Methodology
4. Experiments
5. Conclusion and Future Works

Introduction

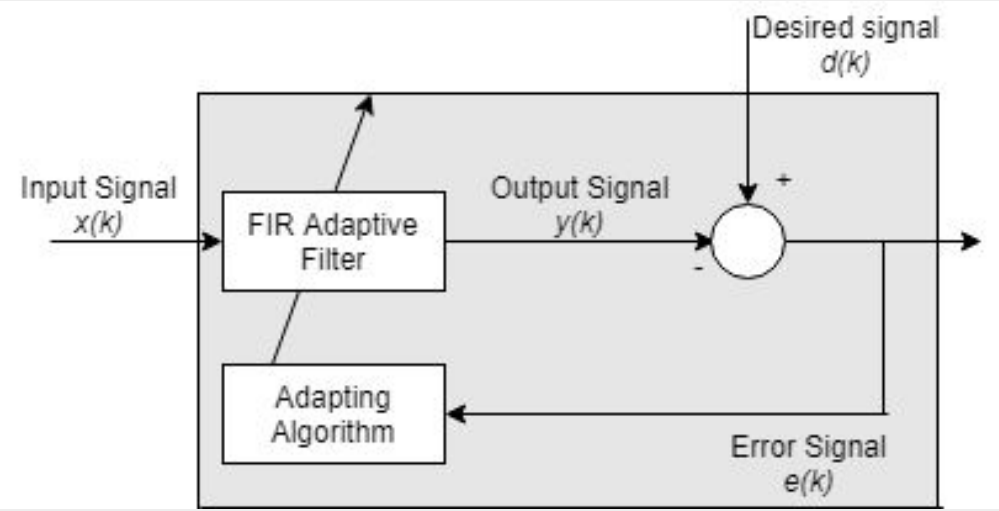
1.1 Overview
1.2 Idea and motivation
1.3 Related Works
1.4 Contribution

1.1 Overview

- What is Speech Enhancement?
- The need for Speech Enhancement

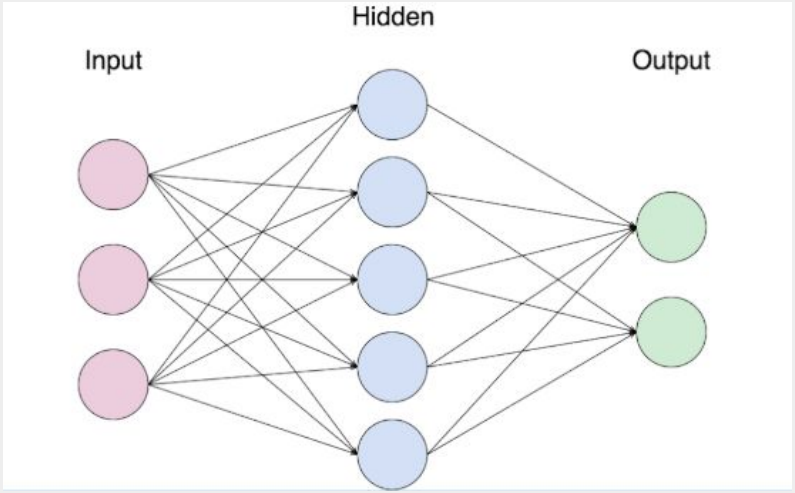
1.1 Overview

- Typical traditional approach: Digital Signal Processing



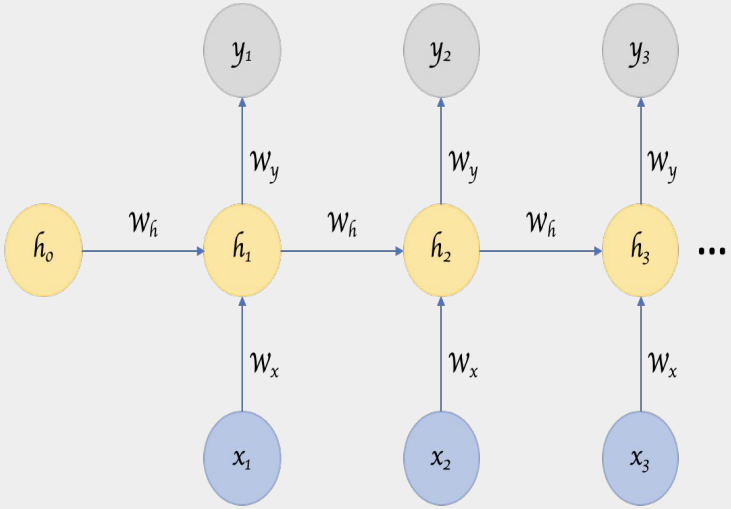
1.1 Overview

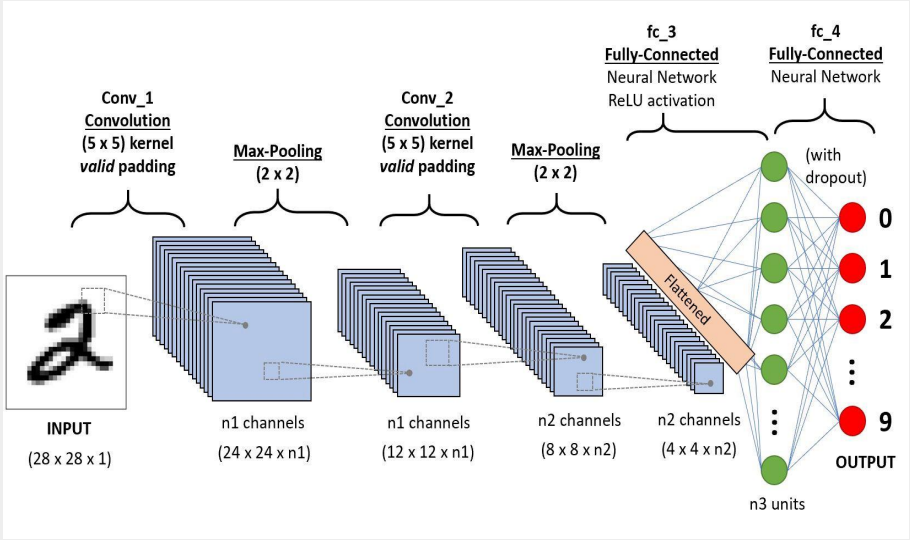
- The trend of using DNNs for Speech Enhancement



1.1 Overview

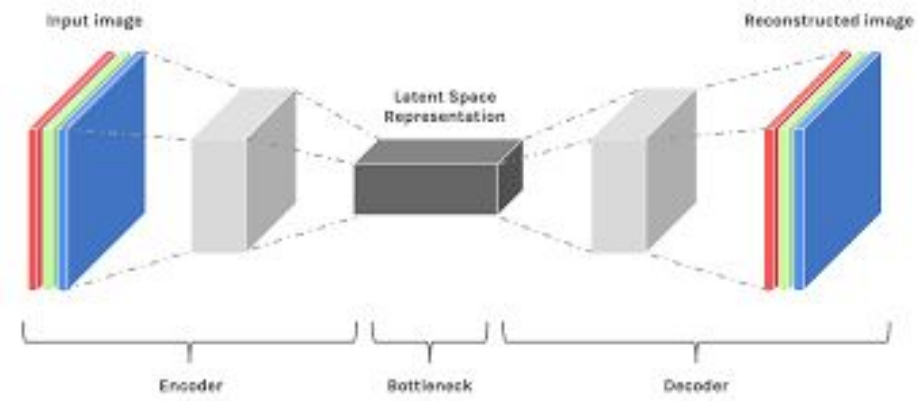
- DNN approaches: RNN and CNN

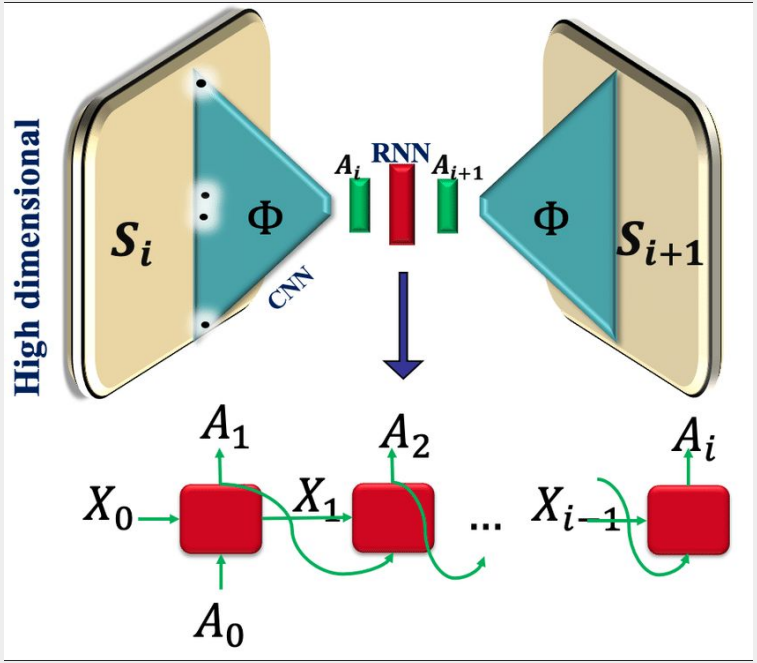




1.1 Overview

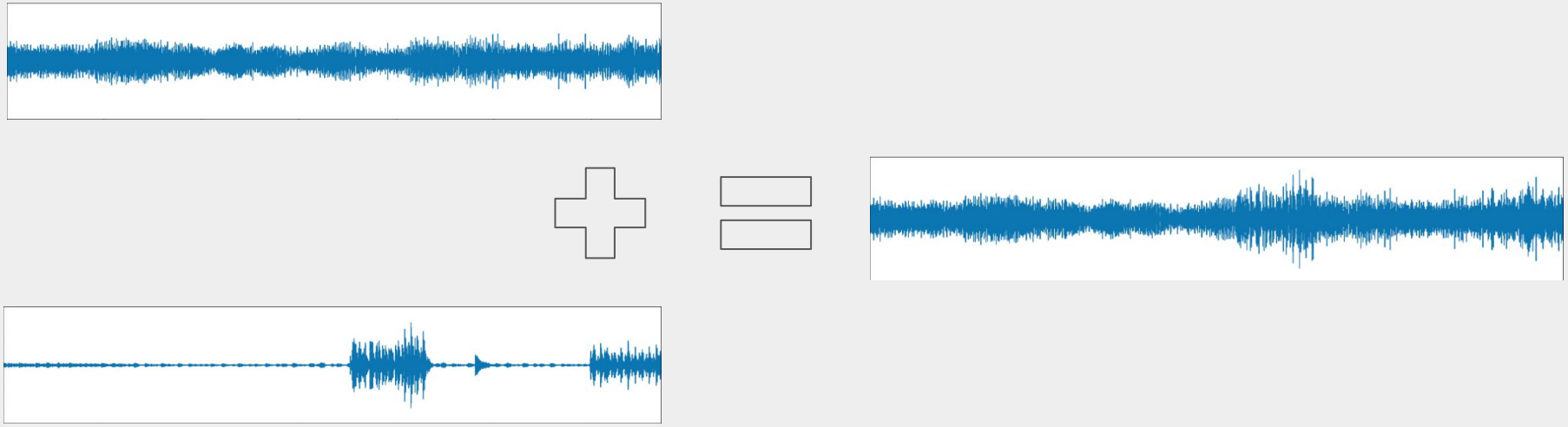
- Contemporary DNN trend: CDAE, CRNN





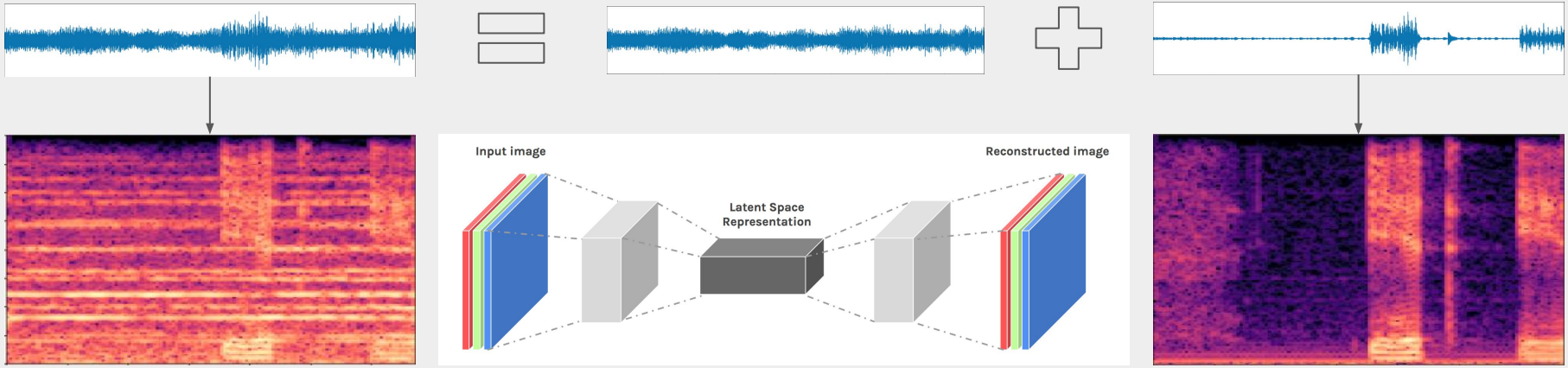
1.1 Overview

- Data



1.1 Overview

- Common flow



1.1 Overview

- Challenges
  - Data
  - Model complexity

1.2 Idea and Motivation

- The success of using Convolutional Encoder-Decoder for Speech Enhancement
- The complexity of previous high-performance models
- The shortage of data in previous works

1.3 Related Works

- A fully convolutional neural network for speech enhancement (Park & Lee, 2016)

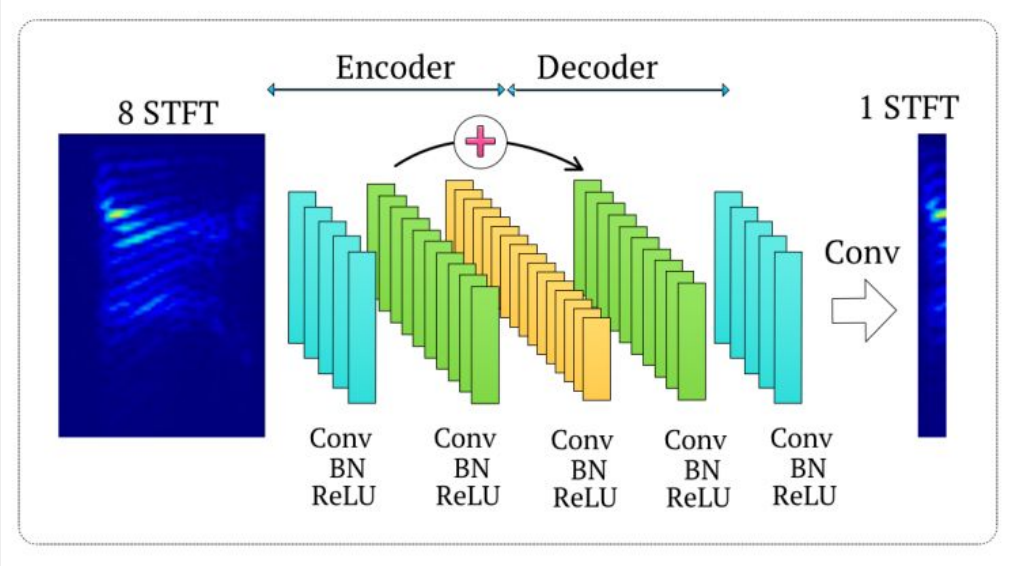


Figure 1.1: R-CED

1.3 Related Works

- A Convolutional Recurrent Neural Network for Real-Time Speech Enhancement (Tan & Wang, 2018)

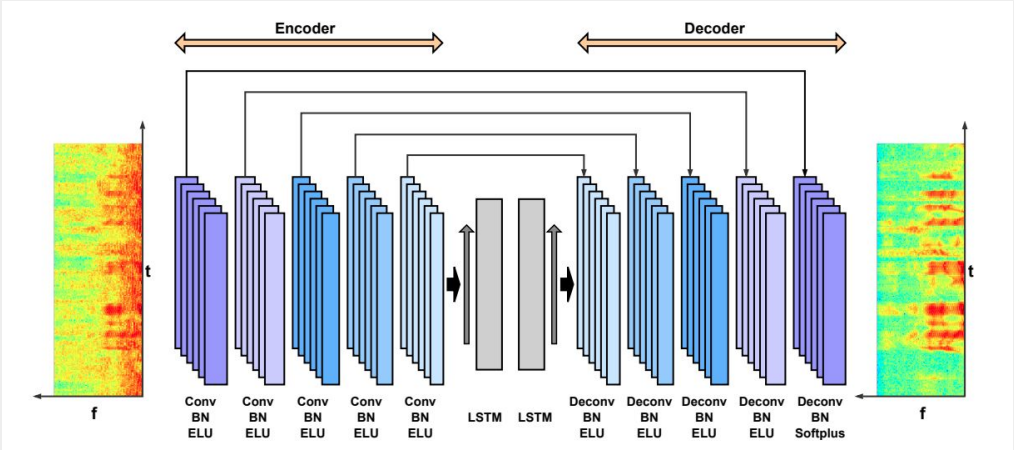


Figure 1.2: CRN

1.3 Related Works

- TCNN: Temporal Convolutional Neural Network for Real-Time Speech Enhancement in the Time Domain (Pandey & Wang, 2019)

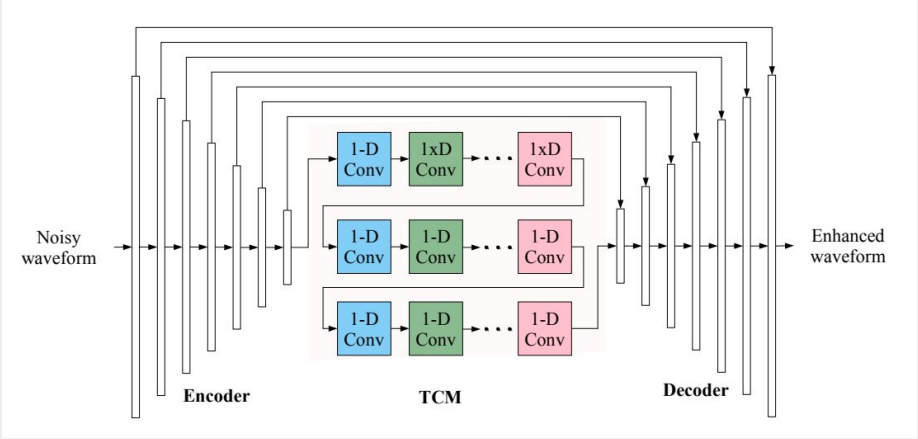


Figure 1.3: TCNN

1.3 Related Works

- Speech enhancement using progressive learning-based convolutional recurrent neural network (Li et al., 2020)

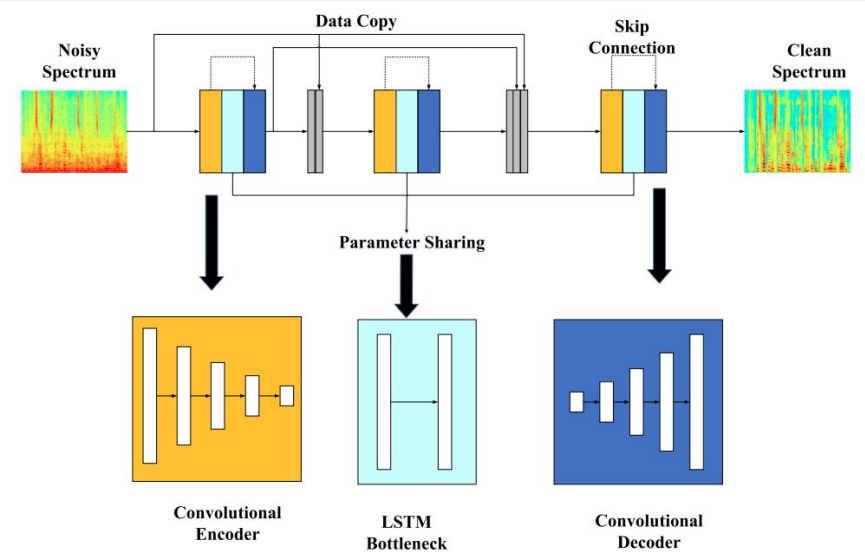


Figure 1.4: PL-CRNN

1.3 Related Works

- Real Time Speech Enhancement in the Waveform Domain (Defossez et al., 2020)

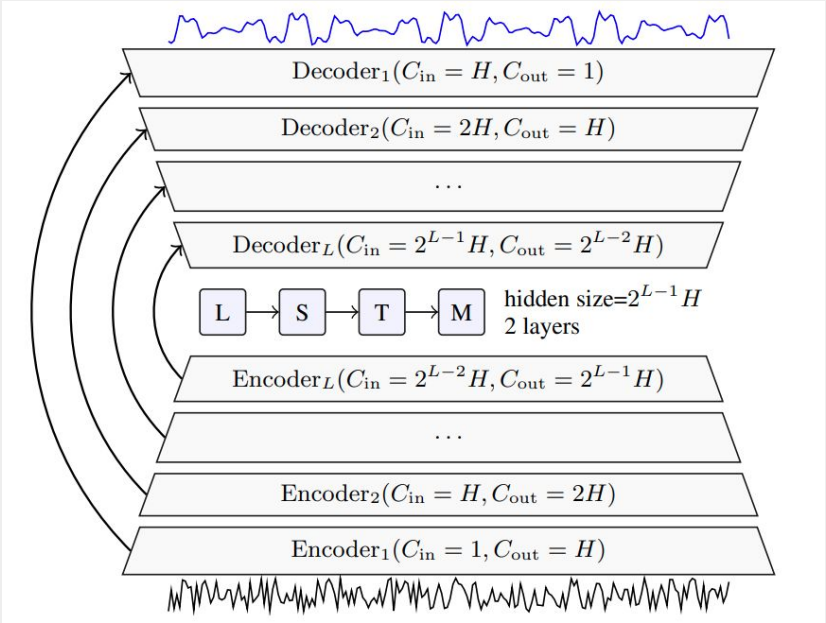


Figure 1.5: DEMUCS

1.4 Contribution

- Main contribution: robust dataset for speech enhancement
- Secondary contribution: applying Unet architecture

Data collecting and preprocessing

2.1 Data collecting
2.2 Data preprocessing

2.1 Data collecting

- Why choose to collect clean voice data and noise data separately?

2.1 Data collecting

- Clean voice: LibriSpeech (train-clean-100)

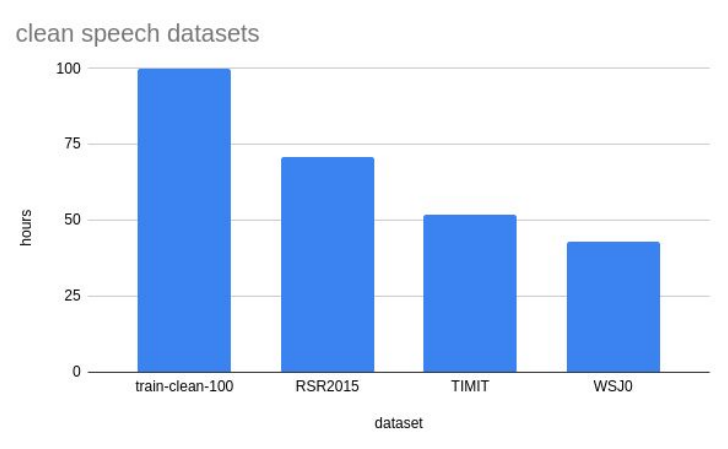


Table 2.1: Clean voice data

subset	hours	per-spkr minutes	female spkrs	male spkrs	total spkrs
dev-clean	5.4	8	20	20	40
test-clean	5.4	8	20	20	40
train-clean-100	100.6	25	125	126	251

2.1 Datasets

- First noise dataset (for training): ESC-50
- Compare ESC with others (DEMAND, NOISEX, NOIZEUS, AURORA)
- Second noise dataset: UrbanSound (fold 1) (for validation and testing)

Table 2.2: Classes in noise dataset

Animals	Natural soundscapes and water sounds	Human, non-speech sounds	Interior/ domestic sounds	Exterior/ urban noises
Dog	Rain	Crying baby	Door knock	Helicopter
Rooster	Sea waves	Sneezing	Mouse click	Chainsaw
Pig	Crackling fire	Clapping	Keyboard typing	Siren
Cow	Crickets	Breathing	Door, wood creaks	Car horn
Frog	Chirping birds	Coughing	Can opening	Engine
Cat	Water drops	Footsteps	Washing machine	Train
Hen	Wind	Laughing	Vacuum cleaner	Church bells
Insects (flying)	Pouring water	Brushing teeth	Clock alarm	Airplane
Sheep	Toilet flush	Snoring	Clock tick	Fireworks
Crow	Thunderstorm	Drinking, sipping	Glass breaking	Hand saw

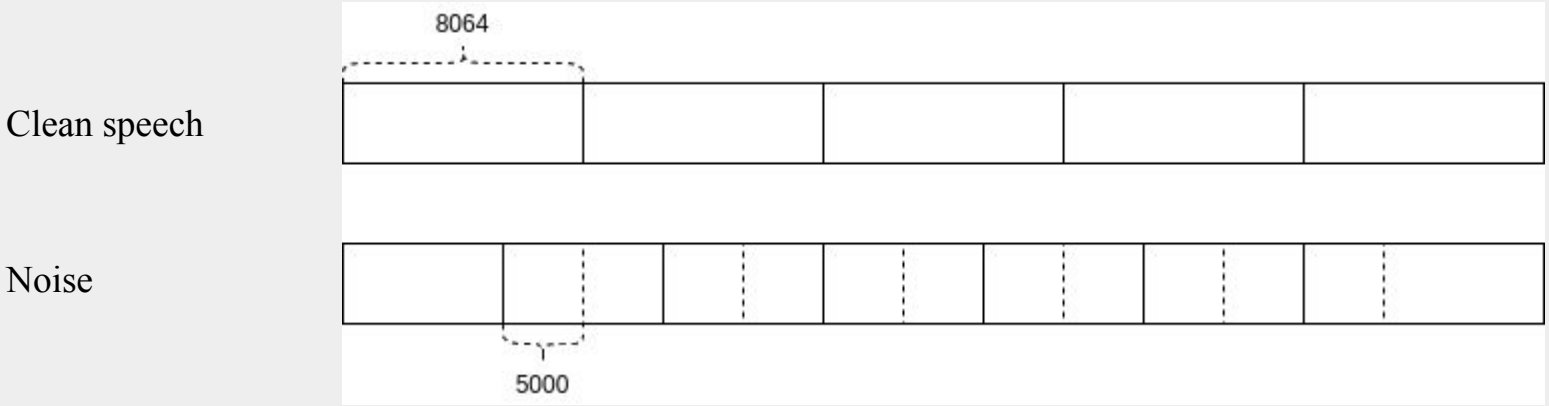
2.2 Data preprocessing

$$x = s + \alpha \times n$$

x is noisy speech, s is clean speech and n is noise

$$\alpha = 10^{SNR\_dB/10}$$

2.2 Data preprocessing



Methodology

3.1 Problem Formulation
3.2 Model
3.3 Objective function

3.1 Problem formulation

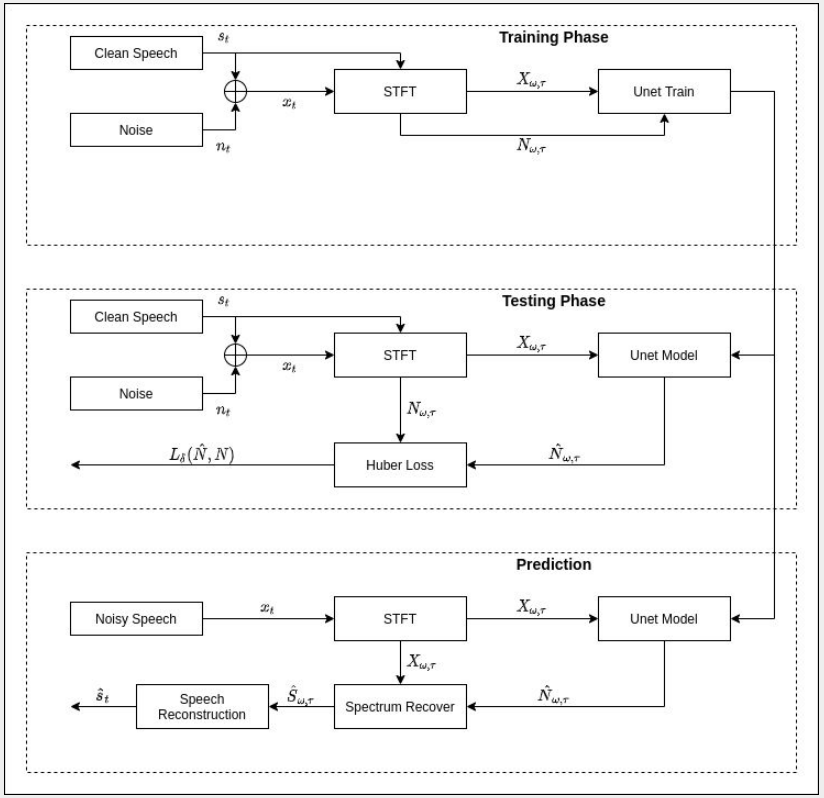


Figure 3.1: System pipeline

3.2 Model

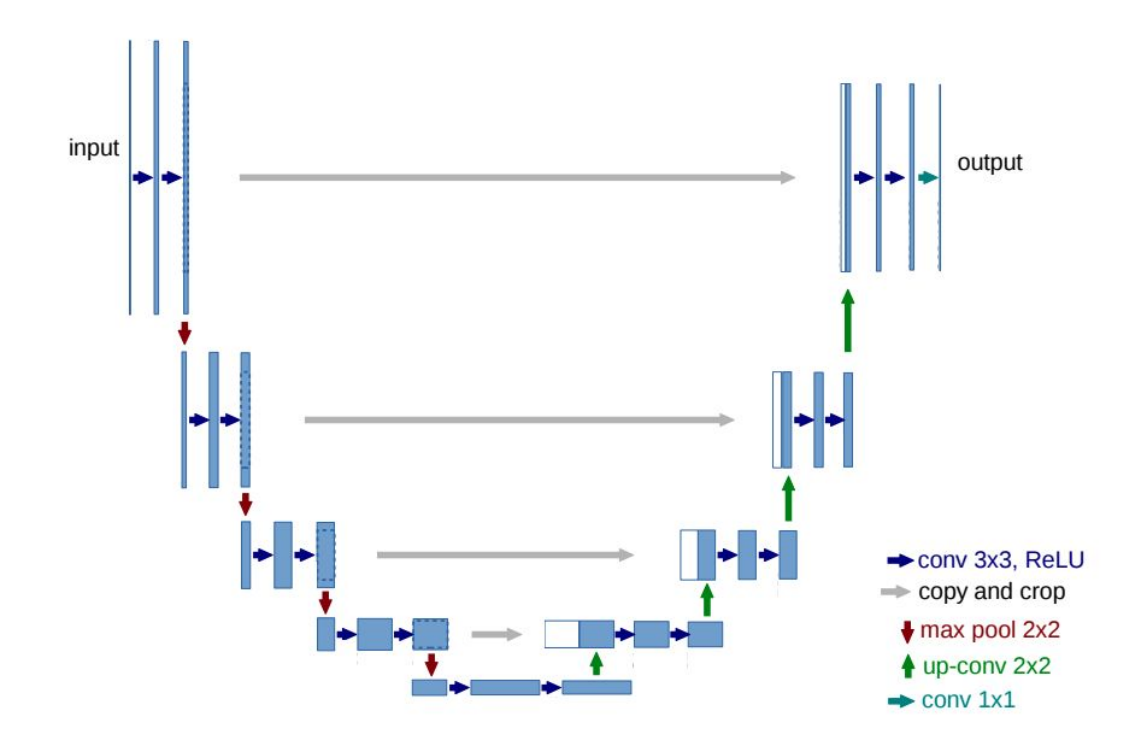


Figure 3.2: Unet pipeline

3.2 Model

Table 3.1: Unet architecture

layer name	output shape	hyperparameters	connected to
input_1	(None, 128, 128, 1)	_	_
conv2d	(None, 128, 128, 16)	160	input_1
conv2d_1	(None, 128, 128, 16)	2320	conv_2d
conv2d_2	(None, 64, 64, 32)	4640	conv2d_1
conv2d_3	(None, 64, 64, 32)	9248	conv2d_2
conv2d_4	(None, 32, 32, 64)	18496	conv2d_3
conv2d_5	(None, 32, 32, 64)	36928	conv2d_4
conv2d_6	(None, 16, 16, 128)	73856	conv2d_5
conv2d_7	(None, 16, 16, 128)	147584	conv2d_6
conv2d_8	(None, 8, 8, 256)	295168	conv2d_7
conv2d_9	(None, 8, 8, 256)	590080	conv2d_7
conv2d_10	(None, 16, 16, 128)	131200	conv2d_9
concatenate	(None, 16, 16, 256)	_	conv2d_7, conv2d_10
conv2d_11	(None, 16, 16, 128)	295040	concatenate
conv2d_12	(None, 16, 16, 128)	147584	conv2d_11
conv2d_13	(None, 32, 32, 64)	32832	conv2d_12
concatenate_1	(None, 32, 32, 128)	_	conv2d_5, conv2d_13
conv2d_14	(None, 32, 32, 64)	73792	concatenate_1
conv2d_15	(None, 32, 32, 64)	36928	conv2d_14
conv2d_16	(None, 64, 64, 32)	8224	conv2d_15
concatenate_2	(None, 64, 64, 64)	_	conv2d_3, conv2d_16
conv2d_17	(None, 64, 64, 32)	18464	concatenate_2
conv2d_18	(None, 64, 64, 32)	9248	conv2d_17
conv2d_19	(None, 128, 128, 16)	2064	conv2d_18
concatenate_3	(None, 128, 128, 32)	_	conv2d_1, conv2d_19
conv2d_20	(None, 128, 128, 16)	4624	concatenate_3
conv2d_21	(None, 128, 128, 16)	2320	conv2d_20
conv2d_22	(None, 128, 128, 2)	290	conv2d_21
conv2d_23	(None, 128, 128, 1)	3	conv2d_22
total parameters		1,941,093	

3.3 Objective function

Huber loss

$$L_\delta(\hat{N}, N) = \begin{cases} \frac{1}{2}(\hat{N} - N)^2 & \text{for } |\hat{N} - N| \leq \delta, \\ \delta|\hat{N} - N| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$$

Experiments

4.1 Datasets
4.2 Experiment settings
4.3 Evaluation
4.4 Results

4.1 Datasets

- Six sub-experiments, each uses the datasets:
  - Training set: 100 hours - 344909 samples
  - Validation set: 5 hours - 17872 samples
  - Testing set: 5 hours - 17980 samples
  - Noise:
    - ESC-50: 2.5 hours, mixed with training data
    - UrbanSound: 2.5 hours, mixed with validation and testing data
- All the data are sampled at 8kHz
- Each sample contains 8064 frames (over 1s)

4.2 Experiment settings

- Platform: Google Colab
- Environment: Tensorflow
- GPU: NVIDIA Tesla K80
- STFT: 256 points Hann window, 64 window shift
- Batch size: 64
- Optimizer: Adam
- Learning rate: 0.001
- Epochs: 10

4.3 Evaluation

- Metrics: STOI (Short-Time Objective Intelligibility) vs PESQ (Perceptual Evaluation of Speech Quality)

Table 4.1: STOI measure results

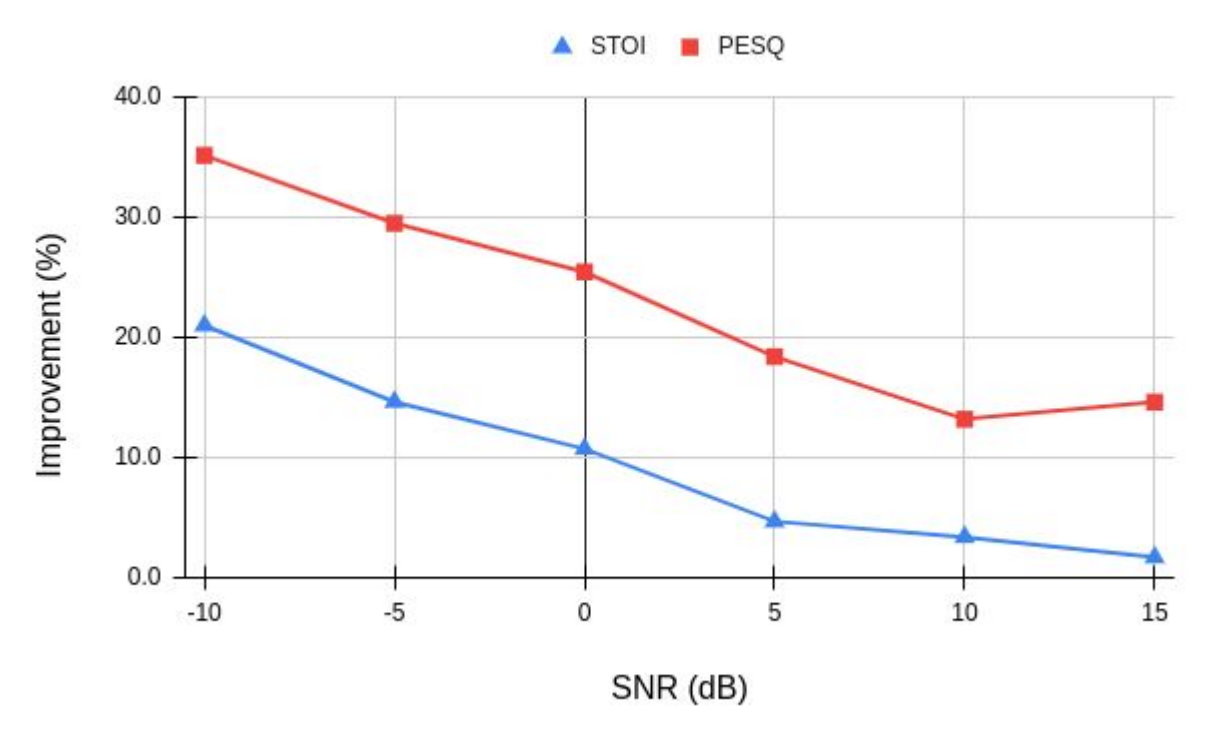
Evaluation metrics	STOI(%)						
SNR	-10	-5	0	5	10	15	Avg.
unprocessed	61	72.5	79.3	87.7	91.6	94.6	81.1
train-clean-100	73.8	83.1	87.8	91.8	94.7	96.2	87.9

Table 4.2: PESQ measure results

Evaluation metrics	PESQ						
SNR	-10	-5	0	5	10	15	Avg.
unprocessed	1.68	2.07	2.32	2.72	2.96	3.22	2.50
train-clean-100	2.27	2.68	2.91	3.22	3.35	3.69	3.02

4.3 Evaluation

Figure 4.1: Improvements in Evaluation Scores



4.4 Results

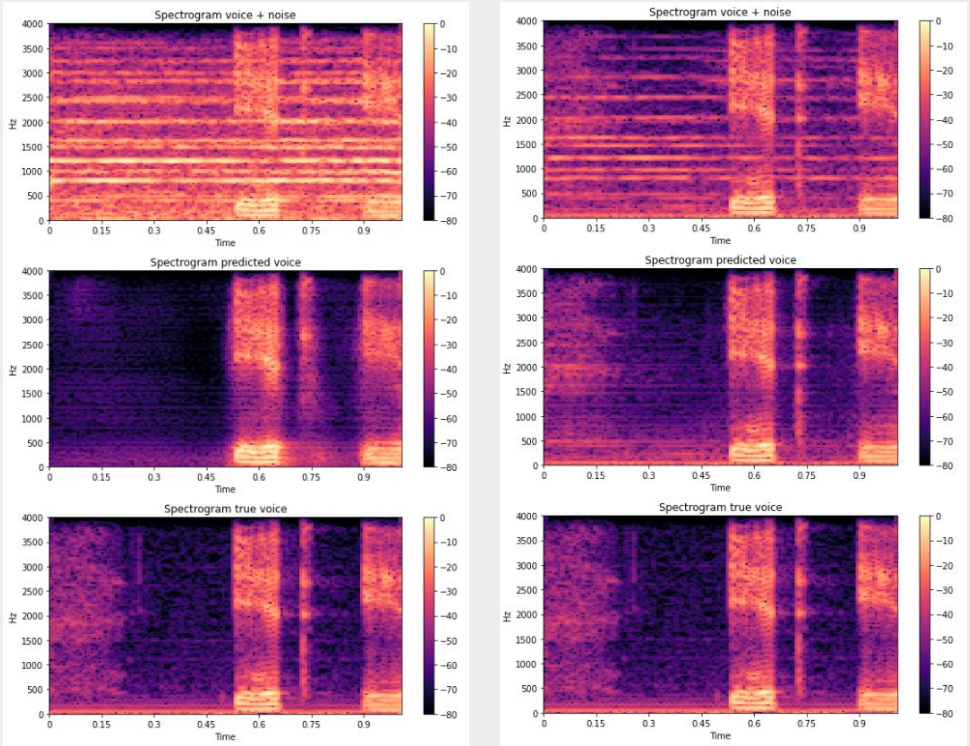


Figure 4.2: Spectrograms with different levels of SNR

4.4 Results

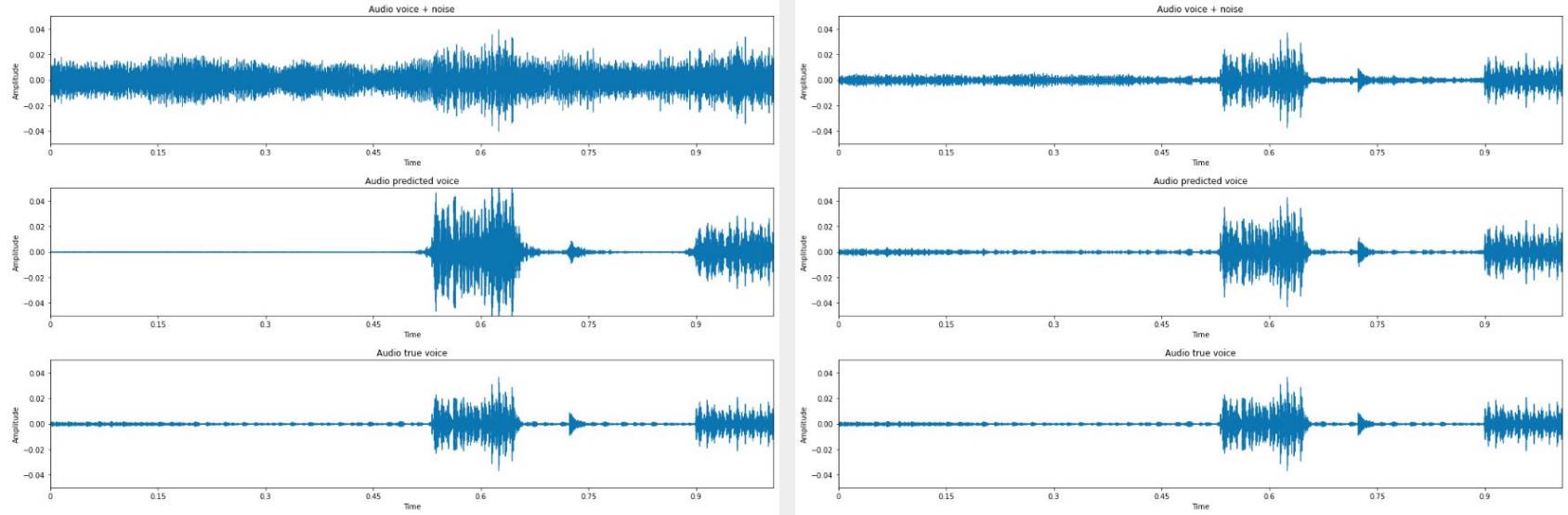


Figure 4.3: Waveforms with different levels of SNR

Conclusion and Future Works

5.1 Conclusion
5.2 Future Works

5.1 Conclusion

- Robust dataset
- Applying Unet
- Common flow

5.2 Future Works

- Shortcomings:
  - Long training time (45 minutes to 1 hour per epoch)
  - Not training with train-clean-360
  - Huber loss is not a robust objective function

5.2 Future Works

- Proposed future improvements:
  - Reduce the size of the model
  - Raise the performance of the model
    - Adapt model architecture
    - Data augmentation (especially noise)

Thank you
for your attention!

Q & A