RESEARCH PAPER

Speech Enhancement Using U-Net

K. Kishore Kumar[1], Kakumanu V V Nagendra Babu[2], Ranjith Kumar Chinnam[3], D. Bhavana[*4], K. Pushpavalli[1], VLN Manaswini[1], A Mary Lavanya[1], M Havila[1], D. Ravi Tej[5]

[1]Assistant Professor, Dept. of CSE, Andhra Loyola Institute of Engineering and Technology, Polytechnic Post Office, Vijayawada, 520008, AP, India
[2]Assistant Professor, Department of Information Technology, Swarnandhra College of Engineering and Technology, Narsapur, Andhra Pradesh, India
[3]Assistant Professor, Department of Artificial Intelligence and Machine Learning, Aditya University, Surampalem, Andhra Pradesh, India
*[*4]Associate Professor, Dept. of ECE, Green Fields, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur District, AP, India*
[5]Assistant Professor, Dept. of ECE, SRK Institute of Technology, Enikepadu, Vijayawada, AP, India.

ABSTRACT

Speech enhancement plays a critical role in various real-world applications, including hearing aids, mobile communications, and healthcare technologies. This paper explores the effectiveness of the U-Net model, an end-to-end neural network architecture inspired by Wave-U-Net, for enhancing speech signals. The model's performance is assessed using two datasets: the publicly available VCTK corpus and a noise-contaminated variant of the Libri Speech dataset. Different loss functions are evaluated, including Mean Squared Error (MSE), L1 norm, and a combination of both. Experimental results demonstrate that the U-Net architecture consistently delivers superior performance compared to existing state-of-the-art approaches, particularly in terms of speech intelligibility and audio quality. Among the tested loss functions, the MSE-based approach yielded the best results.

Keywords: American Sign recognition, CNN, Background Subtraction, OpenCV, Hand tracking and segmentation, Feature extraction.

How to cite this article: Kishore Kumar K, Nagendra Babu KVV, Chinnam RK, Bhavana D, Pushpavalli K, Manaswini V, Lavanya AM, Havila M, Ravi Tej D. Speech Enhancement Using U-Net. Int J Drug Deliv Technol. 2026;16(20s): 118-124. DOI: 10.25258/ijddt.16.20s.14

Source of support: Nil
Conflict of interest: None

1. INTRODUCTION

Audio signals can be represented in various forms, ranging from raw waveforms to time-frequency transformations [1]. Selecting an appropriate representation is critical for achieving high system performance. Among these transformations, spectrograms have proven particularly effective for audio analysis. A spectrogram is a two-dimensional visualization derived from the Short Time Fourier Transform (STFT), where the time component and frequency content are plotted on the axes, and intensity reflects the amplitude of frequency components over time. This visual format makes spectrograms well-suited for applying Convolutional Neural Networks (CNNs), which are traditionally used in image processing, to audio tasks. Between the two types of spectrograms—magnitude and phase—the magnitude spectrogram retains the core features of the signal, whereas the phase spectrogram generally shows limited variation in both time and frequency domains [2]. This study focuses on developing a speech enhancement system that reduces background noise to improve speech clarity.

This study involves using magnitude spectrograms to represent audio signals as shown in figure 1 and figure 2, which will help in estimating a noise profile that can be removed from a spectrogram of a noisy speech signal.

The estimated noise profile is then subtracted from the noisy magnitude spectrogram to obtain a cleaner representation of the speech signal. This process allows the model to suppress unwanted background noise while preserving the essential characteristics of the speech components. By operating in the time–frequency domain, the system can more effectively distinguish between speech patterns and noise artifacts.

The enhanced spectrogram is subsequently transformed back into the time-domain waveform using the inverse Short Time Fourier Transform (ISTFT). This approach enables improved speech intelligibility and overall audio

Corresponding Author: *ranjith61ch@gmail.com*

quality, making it suitable for applications such as speech recognition, communication systems, and hearing assistance technologies.

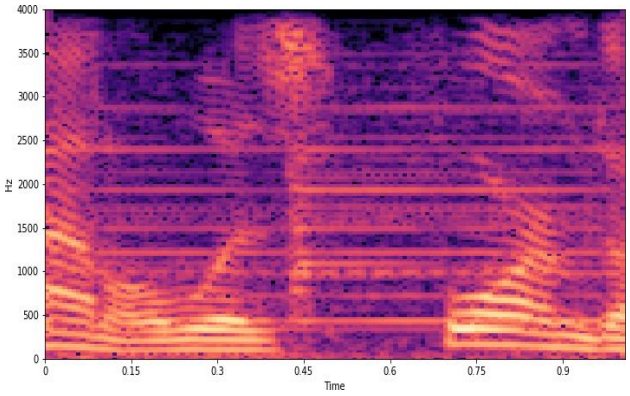

Figure 1: Time Frequency Spectrogram

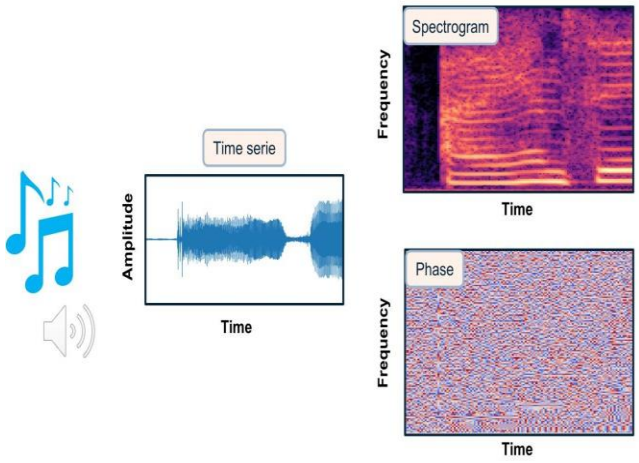

Figure 2: Spectrogram and phase

2. DESIGN AND STUDY

2.1 Various Components Used

Python is a high-level, interpreted programming language known for its simplicity and readability [3]. Its syntax relies heavily on indentation to define code structure, which encourages clean and understandable coding practices. Designed with both beginners and experienced developers in mind, Python supports various programming paradigms, including procedural, object-oriented, and functional programming [4]. It is dynamically typed and includes automatic memory management through garbage collection.

Often referred to as a "batteries included" language, Python offers a rich standard library that simplifies many programming tasks. The language was created by Guido van Rossum in the late 1980s as a successor to the ABC language and was first released in 1991 as version 0.9.0. Python 2.0 followed in 2000, introducing features like list comprehensions and reference-counting garbage collection [5]. Support for Python 2 ended with version 2.7.18 in 2020. Python 3.0, released in 2008, marked a significant evolution of the language. It introduced many improvements, though it was not fully backward-compatible with Python 2. Today, Python remains one of the most widely used programming languages, valued for its versatility and ease of use across a broad range of applications [6]. The following images sheon in Figure 3 represents the clean voice samples used in this study.

Figure 3: Voice samples and images

The clean speech recordings used in this study were primarily sourced from LibriSpeech, an automatic speech recognition (ASR) corpus derived from public domain audiobooks. Additional audio samples were also taken from the SiSec dataset. Environmental noise samples were collected from the ESC-50 dataset and from publicly available sounds hosted at Columbia University's sound archive [7]. This study focused on ten specific types of environmental noise, including a ticking clock, footsteps, bells, a handsaw, alarms, fireworks, insect sounds, tooth brushing, vacuum cleaner noise, and snoring. An illustration of these noise categories is provided below, using images sourced from Unsplash.

MATLAB is a proprietary programming language and numerical computing environment developed by MathWorks [8]. It supports multiple programming paradigms and is widely used for tasks such as matrix operations, data visualization, algorithm development, user interface design, and integration with code written in other languages. While MATLAB is primarily designed for numerical computation, it also offers symbolic computation capabilities through an optional toolbox powered by the MuPAD engine [9]. Additionally, the Simulink package extends MATLAB's functionality by enabling graphical modelling, simulation, and model-based design for dynamic and embedded [10]. systems. As of 2020, MATLAB is used by over four million individuals' worldwide, spanning fields such as engineering, science, and economics. As shown in Figure 4, a Convolution Neural Network is composed of multiple layers that progressively transform the input data

using differentiable functions. The key layers that form the structure of a CNN include convolution layers, pooling layers, and fully connected layers. When these layers are arranged in a specific sequence, they collectively form the CNN architecture. Each of these layers plays a vital role in extracting features that are distinctive to particular patterns or properties within the input image. These extracted features are then used to perform tasks such as classification or recognition. To optimize the network's performance, a loss function is defined and minimized during training. This minimization is achieved using a mathematical formulation, which guides the network to improve its predictions by adjusting its internal parameters.

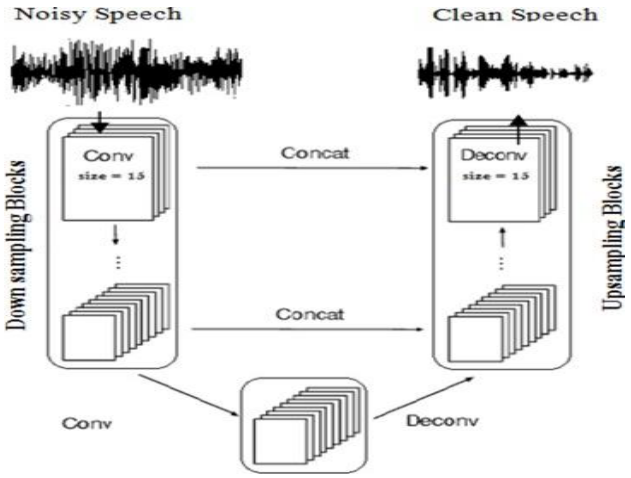

Figure 4: Proposed CNN block diagram

2.2 How is the process carried out

To develop a speech enhancement system, it is essential to gather appropriate data, select suitable algorithms, and build a training model capable of producing the desired output. Below is a step-by-step breakdown of the approach followed in this study:

1. **Data Preparation:** Audio samples were collected from various online sources and open-access libraries to create a comprehensive dataset.
2. **Model Training:** A Python-based program was developed by referencing existing denoising models. This step involved training the model on the prepared dataset.
3. **Prediction:** In the final phase, the trained model was used to generate enhanced speech by reducing background noise in test samples.

2.3 Data Preparation

To prepare the training dataset, English speech recordings and background noise clips were gathered from a range of open-source repositories. Most of the clean voice samples came from the LibriSpeech dataset, which is based on audiobooks in the public domain. Additional speech clips were included from the SiSec collection. Background noise samples were taken from the ESC-50 database and the sound archive maintained by Columbia University. All audio files were converted to a sampling rate of 8 kHz. For the purpose of training, validation, and testing, segments slightly longer than one second were used. To increase the variety in noise data, augmentation was performed by choosing noise snippets from various positions in the original recordings.

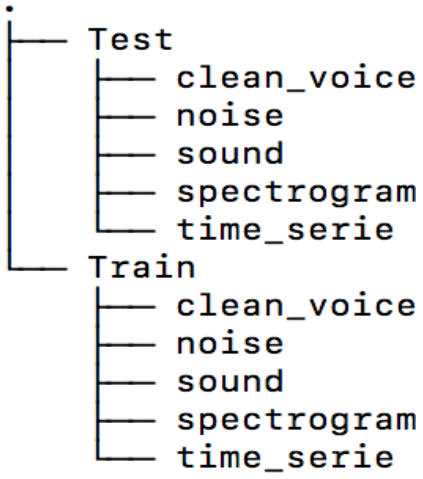

Figure 5: Overlay of the speech

These were then combined with clean speech samples, adjusting the noise level randomly between 20% and 80% to simulate different environments. Altogether, the dataset contained roughly 10 hours of paired noisy and clean speech for training, along with an additional hour reserved for validation.

To configure the data preparation process as shown in Figure 5, the paths in the args.py study are updated, which handles the program's default settings. Place environmental noise files in the noise_dir folder and clean speech samples in the voice_dir folder. The number of training frames to be generated is set using the nb_samples parameter, either directly in args.py or via a terminal argument. While the demo version uses a default of 50 samples, for a full-scale application, at least 40,000 samples are recommended. This study will combine random clean voice samples with environmental noise from the specified directories and generate various output files. These include spectrograms of the noisy mixtures, isolated noise, and clean speech, as well as their corresponding phase components, time-domain signals, and audio files—useful for quality checks or testing alternative models. The args.py settings file also defines key parameters for the Short-Time Fourier Transform (STFT), such as the frame size and the step between frames (hop length). By default, each segment is transformed into a spectrogram of size 128 × 128. The primary inputs for

training are the magnitude spectrograms of both noisy and clean speech signals.

2.4 Training

The architecture employed in this study is based on the U-Net model, which was initially developed for segmenting biomedical images. In this case, it has been adapted for the task of removing noise from audio spectrograms. The network receives magnitude spectrograms of noisy audio as input and is trained to estimate the noise by learning the difference between the noisy and clean versions. To ensure stable training, both input and output data are normalized within a range from -1 to 1.

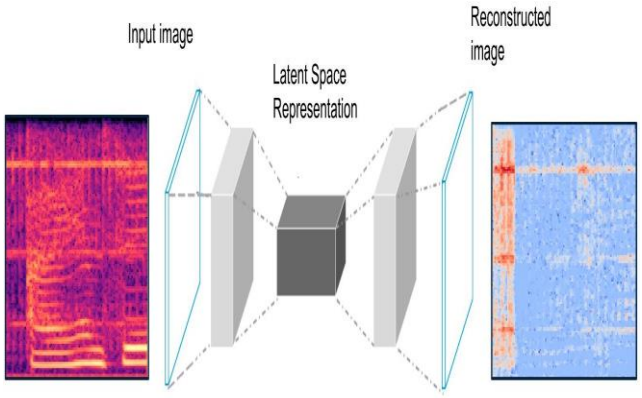

Figure 6: U-Net System

Various model configurations were explored during training. The most effective setup featured an encoder composed of 10 convolutional layers, each using LeakyReLU activations, along with max pooling and dropout layers to prevent overfitting. The decoder followed a mirrored structure, using skip connections to retain high-resolution features. A hyperbolic tangent (tanh) function was applied at the final layer to ensure output values were scaled between -1 and 1. When training the model from the beginning, weights were initialized using a normal distribution. The model was compiled with the Adam optimizer, and the Huber loss was chosen as the objective function—a balanced choice between L1 and L2 losses. Training the model from scratch typically takes a few hours on a modern GPU. This study uses parameters set in the args.py configuration file. By default, the model begins training with randomly initialized weights, though this behavior can be changed by setting the corresponding flag in the config file. You can also resume training using pre-trained weights found in the weights directory—specifically, a provided file named model_unet.h5. The number of training epochs and batch size can be adjusted through the epochs and batch_size parameters. The model automatically saves the best-performing weights during training to a file named model_best.h5. If working with limited memory, you can use fit_generator to load data in smaller batches during training. In this study, training was carried out using Google Colab's free GPU resources. An example notebook, Train_denoise.ipynb, is included in the ./colab directory. If you're working with a large dataset and sufficient storage, it's also possible to preload your data to local or cloud storage and load portions as needed using TensorFlow's data generators. Due to limited space on

Google Drive, training data was split into 5 GB chunks in advance. Weights were saved and reloaded between training sessions. The final model achieved a training loss near 0.002129, while the validation loss settled around 0.002406.

A visual representation of how these losses changed during the training process is provided in Figure 7

2.5 Prediction

During inference, each noisy audio clip is split into overlapping time-series segments, each slightly longer than one second. These segments are processed using the Short-Time Fourier Transform (STFT) for extracting the magnitude and phase information from the signal in spectrogram form. The magnitude spectrograms of the noisy inputs are then fed into the U-Net model, which estimates the noise component for each segment (refer to the graph below for illustration). On a standard CPU, the time taken to process and predict the noise for one spectrogram window is approximately 80 milliseconds.

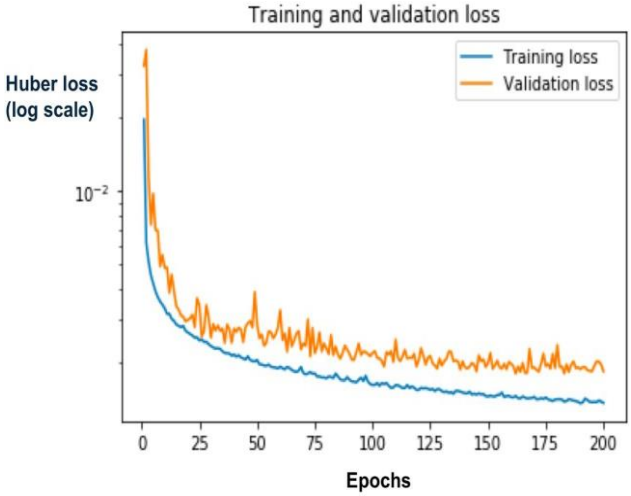

Figure 7: Training and validation loss

2.6 Model Under Execution

Shown below in Figure 8 and Figure 9 are some validation results using noise types such as alarm, insects, vacuum

cleaner, and bells. For each example, you'll find three spectrograms: the original noisy input, the output produced by the model after denoising, and the actual clean speech spectrogram. The results indicate that the model effectively learns to identify and reduce different types of background noise, creating a less noisy representation of the spectrogram that aligns well with the original clean audio. Additional validation examples can be found in the GIF preview located at the top of the study repository.

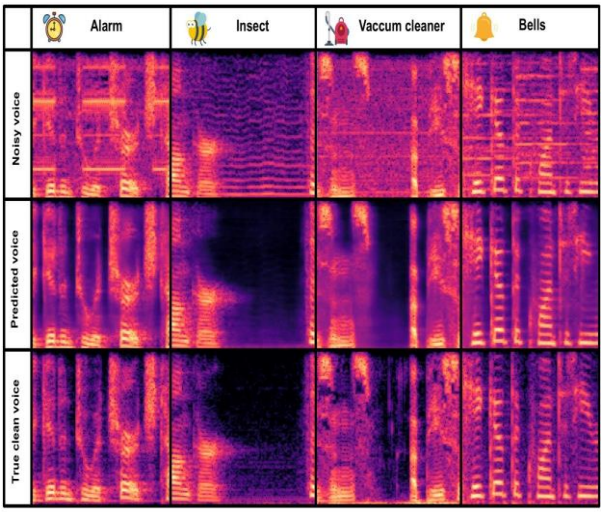

Figure 8: Sound Spectrums

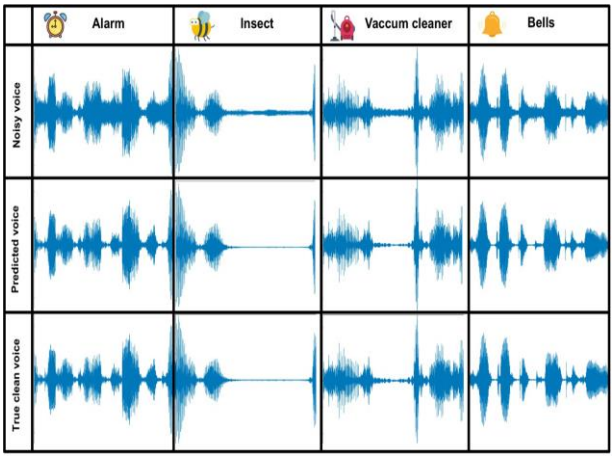

Figure 9: Noise in Time series spectrum

3. Results

Once the datasets were prepared and the model was trained using the selected platform, I used PyCharm to manage and execute the complete workflow—from training through validation. Below are screenshots

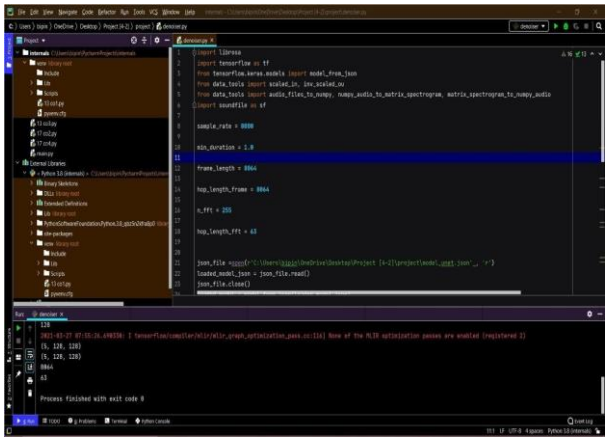

Figure 10: Pycharm 1

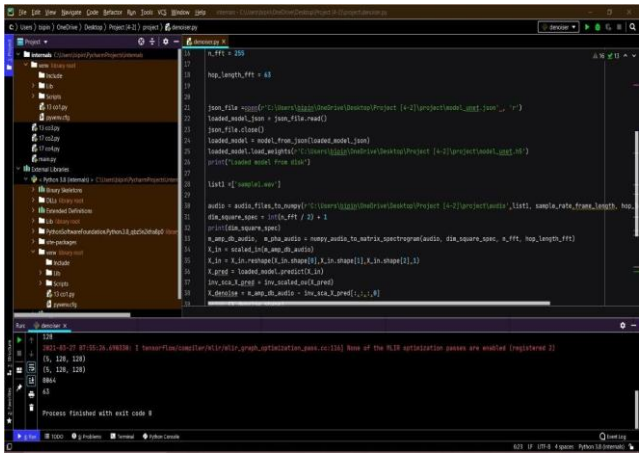

Figure 11: Pycharm 2

3.1 Applications

This technology has a wide range of potential applications. When the audio output is clean and free from background noise, it significantly enhances communication quality, allowing conversations to happen more clearly and without misunderstanding. The specific signals considered important depend on the use case. For example, in mobile speech enhancement, the primary focus is on isolating the user's voice, while any surrounding voices or ambient sounds—like traffic, wind, or rain—are treated as unwanted noise. In contrast, a "cocktail-party" scenario involves multiple people speaking at once in the same space, where each voice is relevant and must be preserved. In such cases, background noise might come from sounds like clinking glasses or footsteps rather than competing speech.

3.2 Discussion

This system has been trained using a range of pre-identified noise types—such as alarm sounds and heavy footsteps that have already been classified as background disturbances in prior research. The model development focused on these familiar noise categories. However, its performance can vary when exposed to unfamiliar or untrained noise types. Additionally, the system currently outputs the user's voice at a slightly reduced volume compared to the original input. Looking ahead, we plan to address these limitations to enhance overall performance. One of our goals is to improve the model's accessibility by creating a more user-friendly interface and eventually making the system publicly available. The core functionality of the machine learning model is to isolate and remove noise from audio recordings. How effectively it performs depends heavily on the intended application, which may vary depending on the user's specific needs. Noise removal is a complex task, as the model must first accurately distinguish between the desired signal and unwanted background sounds. A major challenge lies in building a system capable of making this distinction, which requires extensive training on diverse and well-prepared datasets. The tools used for data preparation, along with the quality and variety of the audio samples, play a crucial role in shaping the model's effectiveness. In this study, we successfully filtered out commonly recognized noise elements from audio input—those that are typically considered disruptive by general users. It's important to note that the filtering process may differ depending on the nature of the audio samples, highlighting the importance of tailoring the approach to specific noise environments.

4. Conclusion

In this study, we successfully developed a system capable of removing unwanted background noise from audio recordings. This significantly enhances the clarity and overall quality of the output, bringing it to a more usable and professional standard. The model was trained and tested using publicly available datasets and online resources. Existing architectures were adapted and refined to meet the specific objectives of our work.

The proposed approach focuses on improving speech intelligibility while preserving the original characteristics of the audio signal. By applying advanced signal processing and deep learning techniques, the system is able to effectively distinguish between useful speech components and unwanted noise.

REFERENCE

1. Jansson, A., Humphrey, E., Montecchio, N., Bittner, R., Kumar, A., & Weyde, T. (2017). Singing voice separation with deep U-Net convolutional networks.
2. Grais, E. M., & Plumbley, M. D. (2017, November). Single channel audio source separation using convolutional denoising autoencoders. In Proceedings of the IEEE Global Conference on Signal and Information Processing (GlobalSIP) (pp. 1265–1269). IEEE.
3. Piczak, K. J. (2015, October). ESC: Dataset for environmental sound classification. In Proceedings of the 23rd ACM International Conference on Multimedia (pp. 1015–1018).
4. Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-Net: Convolutional networks for biomedical image segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention (pp. 234–241). Springer.
5. Das, N., Chakraborty, S., Chaki, J., Padhy, N., & Dey, N. (2021). Fundamentals, present and future perspectives of speech enhancement. International Journal of Speech Technology, 24(4), 883–901.
6. Chao, R., Cheng, W.-H., La Quatra, M., Siniscalchi, S. M., Yang, C.-H. H., Fu, S.-W., & Tsao, Y. (2024, December). An investigation of incorporating mamba for speech enhancement. In Proceedings of the IEEE Spoken Language Technology Workshop (SLT) (pp. 302–308). IEEE.
7. Richter, J., Welker, S., Lemercier, J.-M., Lay, B., & Gerkmann, T. (2023). Speech enhancement and dereverberation with diffusion-based generative models. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 31, 2351–2364.
8. Fu, S.-W., Yu, C., Hsieh, T.-A., Plantinga, P., Ravanelli, M., Lu, X., & Tsao, Y. (2021). MetricGAN+: An improved version of MetricGAN for speech enhancement. arXiv preprint arXiv:2104.03538.
9. Zheng, C., Zhang, H., Liu, W., Luo, X., Li, A., Li, X., & Moore, B. C. (2023). Sixty years of frequency-domain monaural speech enhancement: From traditional to deep learning methods. Trends in Hearing, 27, 1–28.
10. Babaev, N., Tamogashev, K., Saginbaev, A., Shchekotov, I., Bae, H., Sung, H., et al. (2024). FINALLY: Fast and universal speech enhancement with studio-like quality. In Advances in Neural Information Processing Systems, 37, 934–965.
11. Veeraiah, V., Ramesh, J.V.N., Koujalagi, A., Talukdar, V., Namdev, A., Gupta, A. (2024). Health Fitness Tracker System Using Machine Learning Based on Data Analytics. In: Marriwala, N.K., Dhingra, S., Jain, S., Kumar, D. (eds) Mobile Radio Communications and 5G Networks. MRCN 2023. Lecture Notes in Networks and Systems, vol 915. Springer, Singapore. https://doi.org/10.1007/978-981-97-0700-3_57
12. Mamatha, G., Gupta, A., Koujalagi, A., Pooja, P., Pramanik, S., & Roy, A. (2025). Applying Generative AI in a Global Language School to Improve Curriculum Planning.

Advances in Computational Intelligence and Robotics, 151-184. https://doi.org/10.4018/979-8-3693-8744-3.ch007

13. Raju Dr. S. Hrushikesava et al 2020 IOP Conf. Ser.: Mater. Sci. Eng. 981 022067 DOI 10.1088/1757-899X/981/2/022067

14. Sujatha, B., Koujalagi, A., Harika, A., Kumari, V.S. (2024). An In-Depth Convolution Neural Network for Chest X-Ray Image Assessment Using CXRIA-Net. In: Bhateja, V., Tang, J., Polkowski, Z., Simic, M., Chakravarthy, V.V.S.S.S. (eds) Information System Design: AI and ML Applications. ISDIA 2024. Lecture Notes in Networks and Systems, vol 1107. Springer, Singapore. https://doi.org/10.1007/978-981-9

15. Sreekantha, D.K., Koujalagi, A., Girish, T.M., Sairam, K.V.S.S.S.S. (2021). Internet of Things (IoT) Enabling Technologies and Applications—A Study. In: Chiplunkar, N.N., Fukao, T. (eds) Advances in Artificial Intelligence and Data Engineering. AIDE 2019. Advances in Intelligent Systems and Computing, vol 1133. Springer, Singapore. https://doi.org/10.1007/978-981-15-3514-7_107

16. Sreekantha, D.K., Koujalagi, A., Girish, T.M., Sairam, K.V.S.S.S.S. (2021). Internet of Things (IoT) Enabling Technologies and Applications—A Study. In: Chiplunkar, N.N., Fukao, T. (eds) Advances in Artificial Intelligence and Data Engineering. AIDE 2019. Advances in Intelligent Systems and Computing, vol 1133. Springer, Singapore. https://doi.org/10.1007/978-981-15-3514-7_107