Investigation of Machine Learning Techniques for Restoring the Quality of Compressed Audio

Artem Panchenko[1], Denys Yarkin[2], Vyacheslav Frolov[3], Ievgen Meniailov[4], Maryna Vladymyrova[5] and Victoriya Kuznietcova[6]

V.N. Karazin Kharkiv National University, 4 Svobody sq., Kharkiv, 61101, Ukraine

Abstract

This paper investigates machine learning methods for restoring the quality of audio signals compressed with lossy algorithms. A dedicated data preparation methodology was developed, which involved converting WAV files into MP3 format (96 kbps) and subsequently decoding them back into WAV. Three neural network architectures were explored: a linear model, a convolutional neural network (CNN), and a U-Net-like model. The convolutional model demonstrated the best performance for single-sample restoration, but offered the additional advantage of reconstructing entire audio segments rather than individual samples. The results confirm the potential of neural networks for enhancing the quality of compressed audio signals. Visual spectrogram analysis further revealed a noticeable reduction in compression-induced noise. The main limitation of the study was the available computational resources, which constrained the scale and complexity of experiments.

Keywords

neural networks, machine learning, data samples restoring, convolutional neural network, UNet model

1. Introduction

The restrictions introduced during the COVID-19 pandemic in 2020 forced businesses worldwide to transition, wherever feasible, to remote work [1]. However, even after the pandemic subsided, a significant proportion of companies retained remote or hybrid work arrangements. In such settings, communication between employees is predominantly mediated through audio calls or video conferencing. Nevertheless, despite the current era of pervasive digitalization, even densely populated regions still contain areas with unstable connectivity and limited internet bandwidth, which severely complicates or in some cases renders impossible—effective remote work for residents of these regions. Ensuring robust and high-quality communication under conditions of unreliable connectivity thus represents an urgent challenge for the development of advanced communication tools. The present study addresses this issue by proposing a solution aimed at improving the quality of audio transmission.

A wide range of formats exists for the storage of audio information, which can be broadly divided into two categories: lossless formats, which preserve the entirety of the original data, and lossy formats, which achieve compression by discarding certain portions of the information [2]. One of the most well-known, widespread, and technically straightforward lossless formats is WAV (Waveform Audio File) [3]. In this representation, the audio signal is stored as a sequence of discrete samples of the amplitude of the sound wave. To produce such a representation, the audio signal is measured at regular intervals in time, with each recorded value rounded to the nearest available level from a discrete set. The number of available levels is determined by the bit depth (e.g., 16-bit), while the frequency of these measurements is referred to as the sampling rate. This form of representation is often termed a waveform.

In contrast, the most widely used lossy format for audio storage is MP3 (formally, MPEG-1 Audio Layer III or MPEG-2 Audio Layer III), which gained global popularity due to its ability to drastically

ProfIT AI'25: 5th International Workshop of IT-professionals on Artificial Intelligence, October 15–17, 2025, Liverpool, UK

artem.panchenko@karazin.ua (A. Panchenko); denys.yarkin@gmail.com (D. Yarkin); v.v.frolov@karazin.ua (V. Frolov); ievgen.meniailov@karazin.ua (I. Meniailov); vladymyrova@karazin.ua (M. Vladymyrova); vkuznietcova@karazin.ua (V. Kuznietcova)

0000-0001-5865-6158 (A. Panchenko); 0000-0002-2770-3385 (V. Frolov); 0000-0002-9440-8378 (I. Meniailov); 0009-0000-9868-2617 (M. Vladymyrova); 0000-0003-3882-1333 (V. Kuznietcova)

reduce file size by leveraging psychoacoustic models [4]. The underlying algorithm analyzes the audio signal and selectively discards components that are imperceptible to the human auditory system, while encoding the remaining content with relatively minor perceptual loss.

This principle can be applied to optimize the transmission of audio between devices: compressing the signal on one device, transmitting the reduced data, and subsequently reconstructing it on the receiving device. It should be noted, however, that the problem of signal recovery in the presence of packet loss during transmission remains a critical aspect of ensuring communication reliability.

The aim of this study is to develop a neural network to validate the feasibility of employing this approach for restoring the quality of audio after compression. Furthermore, the research seeks to compare the performance of several neural network architectures and to identify the one most suitable for addressing the problem of post-decompression audio quality restoration.

This paper should be regarded as both an introduction to the research area and it is structured as follows:

- Section 3 presents the dataset employed in the study, along with the details of its preprocessing.
- Section 4 describes the configuration and application of Linear Neural Networks for addressing the defined tasks.
- Section 5 outlines the setup and utilization of Convolutional Neural Networks (CNNs) for solving the proposed problems.
- Section 6 details the configuration and use of the UNet model in the context of the study objectives.
- Section 7 provides a comparative analysis of the obtained results.

2. Related Work

Prior to the widespread adoption of neural networks for audio processing, a variety of methods and conceptual approaches were proposed to address the problem of restoring degraded audio quality.

In [5], the authors introduced the idea of estimating the noise spectrum during pauses in speech and subsequently subtracting it from the active signal spectrum. This method demonstrated a reduction of quantization noise and the characteristic “buzzing” artifacts in the reconstructed signal.

In [6], a phase decoder was employed to mitigate phase distortions by reintroducing missing harmonics and smoothing phase fluctuations.

However, with the rapid development of machine learning, research efforts shifted towards neural network–based methods.

In [7], the authors presented a comprehensive review of the challenges and limitations of applying machine learning (ML) to audio restoration. The study examined the role of dataset selection (paired noisy/clean vs. unpaired real-world recordings), the issue of domain shift when synthesizing artifacts (e.g., Gaussian and pink noise, codec distortions, clicks), and the comparative performance of key neural architectures, including CNNs, RNNs, autoencoders/variational autoencoders (AE/VAE), GANs, and transformers. Special attention was given to the choice of loss functions (MSE/MAE, spectral convergence, log-STFT), as well as critical constraints such as overfitting, inference latency (<10 ms for real-time applications), and computational cost. The authors concluded that while ML-based approaches exhibit substantial potential for audio restoration, their practical implementation is hindered by the scarcity of representative real-world datasets and the mismatch between objective evaluation metrics and subjective human perception of quality.

In [8], a lightweight Wavenet-inspired architecture was introduced to enhance audio quality on resource-constrained devices such as hearing aids, enabling real-time noise suppression under challenging acoustic conditions.

In [9], the signal was decomposed into harmonic components, and the resulting representation was processed with architectures originally designed for image analysis, offering an alternative paradigm for audio enhancement.

In [10], the authors proposed a hybrid architecture combining encoder–decoder structures with recurrent LSTM layers. This model, comprising approximately 3.7 million parameters, achieved first

and second place in the “Interspeech 2020 Deep Noise Suppression (DNS) Challenge” in the categories of real-time and offline speech enhancement, respectively.

Overall, the review of existing work demonstrates that, despite current challenges, neural networks and related ML methods hold significant promise for audio restoration. To ensure robust model training, however, careful dataset design and the selection of appropriate evaluation metrics are essential.

A notable limitation of prior studies is their focus on restoring either entire audio segments or the last sample within a sliding window. None of the works considered the possibility of reconstructing a central sample within a window, which could leverage a richer contextual neighborhood and potentially improve performance. Furthermore, existing studies generally do not compare multiple models and approaches on a unified dataset, making it difficult to establish which architectures are more effective under consistent experimental conditions. Similarly, little systematic attention has been devoted to analyzing which specific neural network parameters most strongly influence performance.

The present study seeks to address these gaps by systematically evaluating architectures, training configurations, and contextual prediction strategies for audio restoration.

3. Data preparation

For the construction of the training dataset, the LJ Speech Dataset [11] was selected, which comprises 13,100 short audio recordings of a single speaker reading passages from an English-language book, provided in WAV format.

There exist several approaches to representing audio signals. In this study, we adopt the waveform representation, as it is the most practical for real-time audio restoration tasks. This choice eliminates the need for prior signal transformations before applying the model, thereby ensuring that the experimental setup closely resembles real-world conditions. Under this representation, the responsibility for identifying dependencies and patterns within the data is entirely delegated to the model developed in the course of this work.

During dataset preparation, each original WAV audio track was converted into the MP3 format at a bitrate of 96 kbps using the open-source encoder LAME (Lame Ain’t an MP3 Encoder). Subsequently, the MP3 files were reconverted back into WAV format with identical parameters (16-bit, 22050 Hz). This process produced paired audio samples: a “high-quality” version encoded at the original bitrate and a “degraded” version with lower quality, both represented in waveform form.

To construct the training dataset, the degraded signal was segmented into overlapping windows of neighboring samples. In this work, each window consisted of 201 samples, corresponding to approximately 9 milliseconds of audio. These windows were used to predict the original sample located at the center of the window. This design choice was motivated by the intent to provide the model with a broader temporal context for the restoration of the target sample. As a result, each training instance consisted of 201 input values and a single target value.

It should be emphasized that, due to computational limitations and the requirements of training relatively large models for this task, a reduced dataset of 5 GB was utilized. While this volume is insufficient for building a state-of-the-art model, it nonetheless enables the investigation of how different model parameters affect restoration quality, the identification of optimal configurations, and the formulation of directions for future development of the proposed architectures.

4. Linear model

The linear model is defined as the simplest architecture among those tested, consisting of an input layer followed by several dense layers, each accompanied by an activation function. Additionally, a BatchNorm1d layer was inserted between the dense layers to ensure more stable training.

To determine the most suitable activation function for the linear model—specifically, the one that allows the model to achieve the lowest average loss over an equal number of epochs—a configuration

Table 1
Comparison of the achieved loss values and training time across different numbers of layers with various activation functions

	Loss (7 l., 5000 e.)	Time (7 l., 5000 e.)	Loss (15 l., 10000 e.)	Time (15 l., 10000 e.)
Tanh	2.22e-4	29.39 min	2.51e-4	78.66 min
ReLU	2.55e-4	28.69 min	4.11e-4	76.26 min
Tanh	2.13e-4	30.62 min	5.08e-4	84.54 min

Table 2
Comparison of the Minimum Achieved Loss Values Across Models with Varying Numbers of Parameters and Layers

	5 000	50 000	500 000
1 layer	1.72e-4	1.69e-4	1.92e-4
3 layers	1.65e-4	1.61e-4	1.63e-4
5 layers	1.68e-4	1.92e-4	1.73e-4
7 layers	1.8e-4	2.07e-4	2.16e-4
11 layers	1.8e-4	2.09e-4	2.44e-4
19 layers	2.08e-4	2.21e-4	3.11e-4

with three layers and approximately 100,000 parameters was selected. The best results were obtained with the tanh activation function, yielding a loss value of 1.69e-4.

However, it should be noted that despite the inclusion of BatchNorm1d, the use of tanh in deeper models with a larger number of layers may still lead to issues such as the exploding gradient and vanishing gradient problems. For this reason, it is reasonable to further evaluate the three activation functions that demonstrated the most promising performance—Tanh, ReLU, and SiLU—on a different architecture consisting of seven layers with the same parameter count (100,000). In addition to the achieved loss, special attention is given to the training time required for the same, relatively small number of epochs.

It is important to note that, for the deeper model, a smaller learning rate was employed. This adjustment was necessary because the mathematical function represented by such a model becomes significantly more complex, and a larger learning rate may cause the optimization process to "overshoot" local minima. Under these conditions, the Tanh activation function continued to yield superior results compared to ReLU and SiLU, while maintaining approximately the same training time. Consequently, Tanh was selected as the activation function for all subsequent experiments with the linear model.

To determine the optimal number of layers, models consisting of 1, 3, 5, 7, 11, and 19 layers were evaluated. Since different configurations require distinct hyperparameter settings—specifically with respect to the scheduler and learning rate—multiple training runs were conducted for each version, and the best-performing result was selected for comparison.

The 3-layer architecture with 50,000 parameters achieved the lowest loss value. Consequently, this configuration will be considered the "baseline," and subsequent models with a similar number of parameters will be evaluated relative to it.

Additionally, several types of one-dimensional pooling layers were tested:

- MaxPool1d, which selects the maximum value from each one-dimensional window of the input signal
- AvgPool1d, which computes the average value for each one-dimensional window of the input signal
- LPPool1d, which, for each input window, computes L_p norm. The latter generalizes the notion of distance or magnitude by raising the absolute values of the elements within the window to the power of p summing them, and subsequently taking the p-th root of this sum

Table 3
Achieved loss values obtained with different types of pooling layers

MaxPool1d	AvgPool1d	LPPool1d (p=2)	LPPool1d (p=3)
1.78e-4	1.66e-4	2.2e-4	2.37e-4

Table 4
Comparison of the achieved loss values in the convolutional model when using different filter sizes and dilation values

	3 ks	5 ks	7 ks
1 dil	1.61e-4	1.62e-4	1.68e-4
2 dil	1.6e-4	1.58e-4	1.68e-4
4 dil	1.65e-4	1.63e-4	1.78e-4
8 dil	1.73e-4	1.8e-4	1.93e-4
10 dil	1.81e-4	1.84e-4	2.09e-4

As can be observed, in this case the use of pooling layers does not improve the model's accuracy. It is also worth noting that training the model with LPPool1d requires at least 2.5 times more time compared to other pooling functions, which have only a minor impact on the overall training time.

5. Convolutional Neural Network

The convolutional model is structurally similar to the linear model, but with a crucial distinction: several of the initial dense layers are replaced with convolutional (Conv1D) layers, which, in our case, are applied to the processing of one-dimensional audio signals.

Initially, we test different values of kernel size and dilation. At this stage, our objective is not to identify the most "optimal" parameters for solving the task, but rather to determine which parameter values are meaningful in our case. For this purpose, we construct a simple model consisting of an input layer, a convolutional layer, a flatten layer, a single fully connected layer, and an output layer, with a total parameter count of approximately 50,000. The default value of dilation is set to 1, meaning that the kernel elements are positioned adjacent to one another.

As can be observed, the lowest loss value was achieved with a convolutional layer configured with dilation = 2 and kernel size = 5.

Next, this layer will be duplicated while maintaining the total number of model parameters at approximately 50,000. This approach enables the determination of the optimal number of convolutional layers. The architecture with three convolutional layers demonstrated the highest efficiency; therefore, it will be employed in subsequent experiments.

A common approach in the design of convolutional neural networks is to combine layers with different kernel sizes, often arranging them in order from the layer with the broadest receptive field (i.e., the greatest distance between the first and last elements of the kernel) to the one with the narrowest. This strategy allows the model to capture large-scale dependencies in earlier layers, while later layers specialize in detecting small, localized patterns. Perhaps the most well-known example of such an architecture is AlexNet, which was employed for object detection and recognition in images and had a profound impact on the development of deep learning. This architectural principle can be adapted for time series analysis, which in our case corresponds to audio signals.

It can be observed that the lowest loss was achieved with the configuration 7ks-5ks-3ks and dilation=2; that is, in a model where the first convolutional layer has a kernel size of 7, the second 5, and the third 3. This demonstrates that employing a combination of layers with varying kernel sizes is meaningful and leads to improved performance.

Pooling layers are frequently employed in convolutional neural networks. They allow an increase in the number of filters in convolutional layers without proportionally increasing the total number of

Table 5
Results of testing convolutional models with three convolutional layers using different kernel sizes and dilation values

	dilation=2	dilation=4	dilation=6
5ks-3ks-3ks	1.65e-4	1.71e-4	1.8e-4
5ks-5ks-3ks	1.61e-4	1.74e-4	1.79e-4
5ks-5ks-5ks	1.51e-4	1.68e-4	1.8e-4
7ks-5ks-3ks	1.39e-4	1.67e-4	1.82e-4
7ks-5ks-5ks	1.45e-4	1.68e-4	1.94e-4
7ks-7ks-3ks	1.59e-4	1.67e-4	2.01e-4
7ks-7ks-5ks	1.54e-4	1.72e-4	1.98e-4
9ks-5ks-3ks	1.57e-4	1.81e-4	2.11e-4

Table 6
Results of testing the convolutional model with different types of pooling layers

MaxPool1d	AvgPool1d	LPPool1d (p=2)	LPPool1d (p=3)
1.43e-4	1.37e-4	1.62e-4	1.7e-4

Table 7
Minimal achieved loss values of convolutional models with varying numbers of fully connected layers following the convolutional layers

1 dense l.	3 dense l.	5 dense l.	7 dense l.
1.39e-4	1.42e-4	1.46e-4	1.55e-4

model parameters. We tested this approach on the three-layer architecture obtained in the previous stage.

t can be observed that the use of AvgPool1d slightly improved the model's accuracy, though the effect is not substantial. Thus, while this approach can be effective in general, in our specific case it does not produce a radical improvement.

In convolutional models, it is common to add one or more dense layers after the convolutional layers to perform a form of "analysis" on the extracted features. We will attempt to determine the optimal number of these dense layers by incorporating them into the model developed in the previous stage. It is important to note that we will maintain the total number of parameters close to 50,000, which requires slightly reducing the size of the convolutional layers.

It can be observed that adding fully connected layers, in this case, does not contribute to achieving lower loss values.

6. UNet model

When training the UNet model, an additional crucial modification to the dataset is required, beyond merely reshaping the target data. First, in the development of the linear and convolutional models, we used an input window of 201 samples to provide the model with equal context on both sides of the target sample. However, for the UNet model, this window length is unnecessary because the model restores an entire segment at once. Second, each encoder level reduces the window length by half; therefore, to ensure correct operation of the algorithm, it is preferable to select an input window length that is divisible by two at each encoder stage. Consequently, we slightly reduce the window length from 201 to 192, which is the product of 64 and 3, allowing the encoder to accommodate up to six levels of depth.

We will test the model based on the UNet architecture across different parameter sizes. Unlike the

Table 8
Results of testing the UNet architecture with varying parameter counts, specifically the minimum achieved loss values

50 000	200 000	400 000	1 200 000	5 000 000
3.82e-4	3.42e-4	3.18e-4	4.03e-4	4.98e-4

Table 9
Minimum achieved loss values of the UNet architecture model with varying numbers of encoder levels

2 levels	4 levels	6 levels	8 levels
3.44e-4	3.11e-4	3.24e-4	3.32e-4

Table 10
Results of testing UNet models with varying convolutional kernel sizes in the convolutional layers

ks = 3	ks = 5	ks = 7	ks = 9
3.11e-4	3.43e-4	3.64e-4	3.81e-4

analogous testing conducted for the linear model, this evaluation will be performed on significantly larger parameter counts. For comparison, we will also test the model using the same number of parameters employed in the convolutional model experiments—50,000. Additionally, in this architecture, it is possible to vary the number of "depth levels" in the encoder and, correspondingly, in the decoder; however, for the present tests, we will fix this number at three.

It can be observed that the model with 400,000 parameters, given sufficient training time, achieves significantly better results than the model with 50,000 parameters and slightly outperforms the model with 200,000 parameters, likely due to the need for a larger dataset and extended training time. A similar situation occurs with models containing 1,200,000 and 5,000,000 parameters, which require substantially greater computational resources for training. However, the achieved loss values for these larger models still remain higher than those obtained during testing of the linear and convolutional models, due to the same limitations—high resource demands for training. Consequently, further experiments will focus on architectures with 400,000 parameters.

As noted above, in this architecture we can experiment with the number of levels, that is, the number of convolutional layers in the encoder (and, correspondingly, in the decoder, since they are symmetrical).

It can be observed that the model with four depth levels demonstrates superior performance compared to the other models, suggesting that this depth may be "optimal" for our task. Next, we will test the UNet architecture with four depth levels, 400,000 parameters, and varying convolutional kernel sizes.

It can be observed that the minimum achieved loss increases with larger kernel sizes, indicating that the optimal choice is to retain the initial kernel size of 3. Although the loss values obtained with this architecture are higher than those achieved with the linear and convolutional models, it is important to note that from a practical application perspective, this architecture may be more effective in certain use cases due to its ability to reconstruct a larger number of samples simultaneously.

7. Results analysis

Two neural network architectures were developed and tested for reconstructing the central sample within a sliding window over an audio signal:

- Linear architecture: A model where the primary computations are performed by fully connected (dense) layers.
- Convolutional architecture: A model in which the main computations are carried out by convolutional layers.

The optimal configuration for the linear model was identified as follows:

- Activation function: Tanh
- Number of hidden layers: 3
- Total parameters: 50,000

Configurations with a greater number of layers and parameters were also tested; however, they produced inferior results. This is typically attributable to insufficient training data or an inadequate number of training epochs. Given the limitations in computational resources, a significantly larger dataset could not be employed. Additionally, training more complex models generally requires a lower learning rate, while the increased number of parameters necessitates longer training times. Considering these constraints, fully training large models to their optimal capacity would have demanded a duration incompatible with rapid parameter adjustments and extensive testing of multiple configurations, or would have been infeasible given the available resources.

Within the scope of the experiment, the linear model was tested with pooling layers inserted between the fully connected layers; however, this approach did not result in any improvement in the loss function.

The convolutional model, in its optimal configuration, demonstrated significantly better performance than the linear model, achieving approximately 15% lower loss in one of the tested configurations. Specifically, the minimum loss value for the convolutional model was obtained with the following parameters:

- Dilation = 2 across all layers
- Layer configuration:
  - First layer: kernel_size = 7
  - Second layer: kernel_size = 5
  - Third layer: kernel_size = 3
- Application of AvgPool1d following each convolutional layer
- Three hidden layers

During the development of the convolutional model, we encountered the same limitations regarding model size as observed with the linear model. Therefore, it is reasonable to assume that with greater computational resources, even better performance could be achieved.

An architecture based on UNet was also developed and tested for reconstructing an entire segment of the audio signal at once. This model exhibited inferior performance in terms of the minimum achieved loss value; however, it is important to note that it represents a considerably more complex architecture than the previously tested linear and convolutional models. Consequently, it likely requires substantially more data and training time. Additionally, this model reconstructs an entire signal segment simultaneously—192 samples instead of a single sample as in the linear and convolutional models—which may be advantageous in certain applications.

The best results in testing the UNet-based architecture were obtained under the following configuration:

- 4 levels in the encoder (and, correspondingly, in the decoder)
- 400,000 parameters
- Convolutional kernel size of 3 at each encoder level
- Pooling layer type: MaxPool1d

We now proceed to evaluate the best-performing model (in terms of the achieved loss) on a real audio recording to assess its practical performance. For this purpose, we selected a fragment of an audio track in WAV format and applied the same preprocessing operations used during dataset preparation, thereby producing a degraded WAV file with reduced quality. The trained model was then applied to this audio fragment, and we compared the spectrograms of the degraded and the reconstructed signals.

Table 11
Comparison of the best results achieved by different models along with the corresponding model configurations

Model	Linear	Convolutional	UNet-like
Input window size	201	201	192
Number of output samples	1	1	192
Number of parameters	50000	50000	400000
Minimum achieved loss value (MSE)	1.61e-4	1.37e-4	3.11e-4

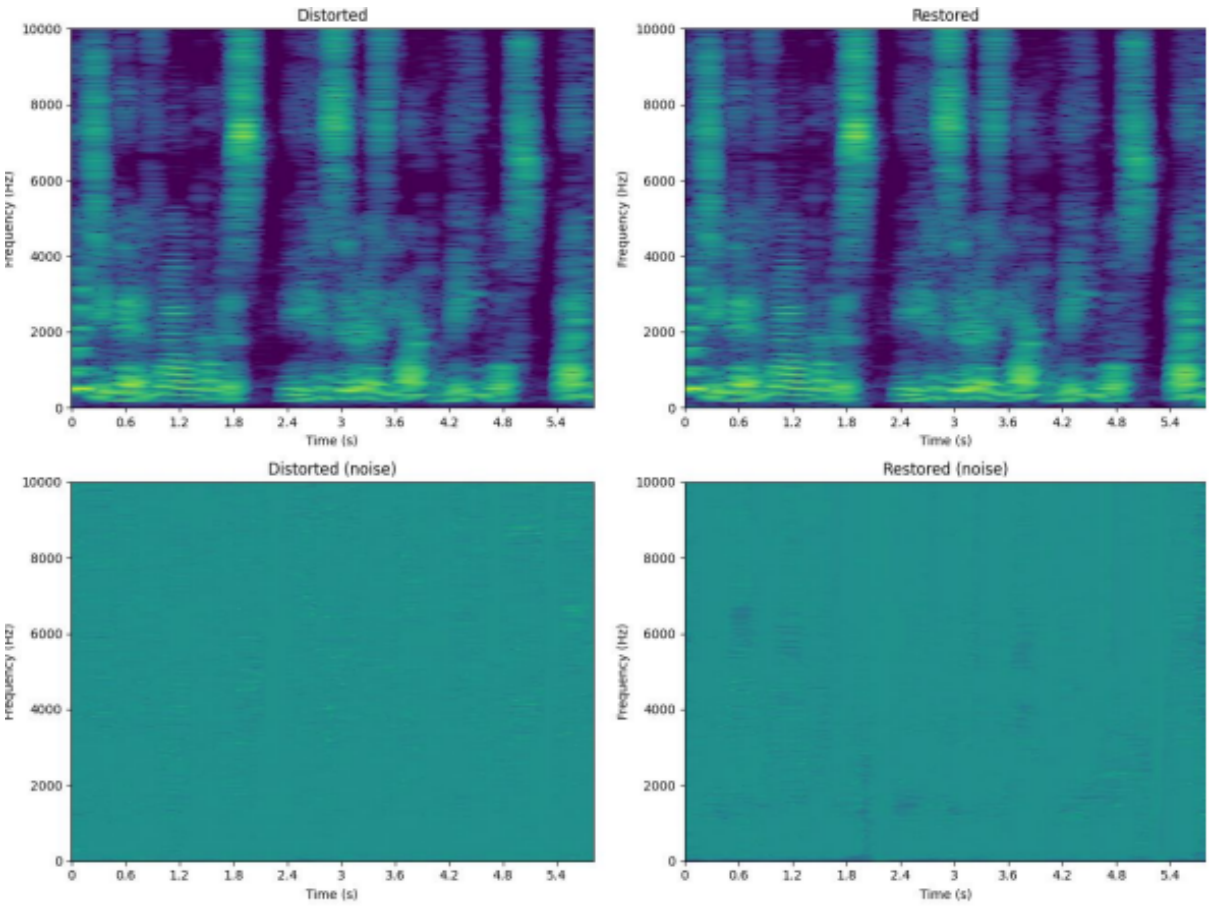

Figure 1: Comparison of the spectrograms of the degraded (left) and reconstructed (right) audio fragments. The lower panels present the spectrograms of the residual noise for these fragments, computed relative to the original audio recording

A detailed examination of the presented spectrograms reveals that, although some additional noise components appear in the reconstructed audio, a considerable proportion of the smaller noise artifacts are effectively suppressed. This observation indicates the potential of the tested architecture.

It should be noted that during the course of this study, no additional evaluation methods for assessing model performance were implemented — in particular, perceptual metrics or listening tests were not conducted. This limitation is primarily due to the need for professional audio equipment and controlled acoustic environments to ensure reliable perceptual assessment. Moreover, such metrics inherently involve a degree of subjectivity, as they rely on human perception and auditory judgment, which can vary significantly across listeners.

For these reasons, the study relied on the analysis of spectrograms as the primary means of evaluating the model's output quality. Spectrogram-based evaluation provides an objective, reproducible representation of the audio signal's frequency–time characteristics, allowing for a more systematic

comparison between distorted and reconstructed signals. While this approach does not fully capture perceptual nuances, it offers a consistent and technically sound basis for assessing improvements in signal reconstruction within the scope of the current research.

8. Conclusions

As a result of the conducted study, three distinct neural network architectures were developed and compared for the task of restoring the quality of audio signals degraded by compression into lossy formats.

A data preparation methodology was designed, involving the conversion of audio files into MP3 format with a low bitrate (96 kbps) followed by reconversion into WAV format, thereby generating pairs of "degraded–original" signals for model training. A sliding window approach of fixed length was proposed, enabling the restoration of the central sample in the case of the linear and convolutional models, or an entire segment in the case of the UNet-based architecture.

As a result of the comparative analysis of the three architectures, the following observations were made:

- Convolutional model: This architecture demonstrated the best performance, achieving an MSE of 1.37e-4, which is approximately 15% lower compared to the linear model (MSE 1.61e-4). The most effective configuration of the convolutional network was identified as one comprising three layers with varying kernel sizes (7–5–3) and a dilation factor of 2.
- Linear model: The linear architecture produced satisfactory results with an optimal configuration consisting of three hidden layers, the Tanh activation function, and approximately 50,000 parameters.
- UNet-like model: Although this architecture exhibited a higher loss value (MSE 3.11e-4), it presents a potential advantage in its ability to reconstruct entire audio segments (192 samples) at once, which may be beneficial in specific practical applications.

A visual analysis of the spectrograms of the restored audio signals confirmed the effectiveness of the developed models, particularly through the observed reduction of “minor” noise in the reconstructed signals.

The primary limitation of this study was the available computational resources, which did not allow for the full training of more complex models with a larger number of parameters. Nevertheless, the experimental results obtained using smaller models and a limited dataset convincingly demonstrate the potential of neural networks for restoring the quality of audio signals. This opens avenues for future research, including the use of larger datasets, more sophisticated architectures, and parameter optimization to achieve state-of-the-art performance.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to Grammar and spelling check. After using these tool, the authors reviewed and edited the content as needed and takes full responsibility for the publication’s content.

References

[1] Herby, Jonas and Jonung, Lars and Hanke, Steve H., Did lockdowns work? : The verdict on Covid restrictions, number 1 in IEA Perspectives, IEA, 2023. URL: https://iea.org.uk/publications/did-lockdowns-work-the-verdict-on-covid-restrictions/.

[2] D. Pan, A tutorial on mpeg/audio compression, IEEE MultiMedia 2 (1995) 60–74. doi:10.1109/93.388209.

[3] Q. Liu, A. H. Sung, M. Qiao, Spectrum steganalysis of wav audio streams, in: P. Perner (Ed.), Machine Learning and Data Mining in Pattern Recognition, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 582–593.
[4] brandenburg karlheinz, mp3 and aac explained, journal of the audio engineering society (1999).
[5] S. Boll, Suppression of acoustic noise in speech using spectral subtraction, IEEE Transactions on acoustics, speech, and signal processing 27 (2003) 113–120.
[6] J. Laroche, M. Dolson, New phase-vocoder techniques for pitch-shifting, harmonizing and other exotic effects, in: Proceedings of the 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. WASPAA'99 (Cat. No. 99TH8452), IEEE, 1999, pp. 91–94.
[7] O. Casey, R. Dave, N. Seliya, E. R. S. Boone, Machine learning: Challenges, limitations, and compatibility for audio restoration processes, in: 2021 International Conference on Computing, Computational Modelling and Applications (ICCMA), IEEE, 2021, pp. 27–32.
[8] K. Fisher, A. Scherlis, Wavemedic: convolutional neural networks for speech audio enhancement, 2016.
[9] S. R. Park, J. Lee, A fully convolutional neural network for speech enhancement, arXiv preprint arXiv:1609.07132 (2016).
[10] Y. Hu, Y. Liu, S. Lv, M. Xing, S. Zhang, Y. Fu, J. Wu, B. Zhang, L. Xie, Dccrn: Deep complex convolution recurrent network for phase-aware speech enhancement, arXiv preprint arXiv:2008.00264 (2020).
[11] K. Park, T. Mulc, Css10: A collection of single speaker speech datasets for 10 languages, arXiv preprint arXiv:1903.11269 (2019).