Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Curriculum Ingegneria Biomedica, Elettronica e delle Telecomunicazioni

Advanced Deep Learning Techniques for Analysis and Reconstruction of Audio Signals

Ph.D. Dissertation of:
Carlo Aironi

Advisor:
Prof. Stefano Squartini

Curriculum Supervisor:
Prof. Franco Chiaraluce

XXXVI edition - new series

Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Curriculum Ingegneria Biomedica, Elettronica e delle Telecomunicazioni

---

Advanced Deep Learning Techniques for Analysis and Reconstruction of Audio Signals

Ph.D. Dissertation of:
Carlo Aironi

Advisor:
Prof. Stefano Squartini

Curriculum Supervisor:
Prof. Franco Chiaraluce

XXXVI edition - new series

Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Facoltà di Ingegneria
Via Brecce Bianche – 60131 Ancona (AN), Italy

Acknowledgments

Before starting with the discussion of this thesis, I would like to express my gratitude and appreciation to my advisor, Prof. Stefano Squartini, who has constantly motivated, supported and guided me during this journey, allowing me to grow personally. I would like to extend my gratitude to my colleagues at A3lab and the staff of the UNIVPM, whose invaluable support and insightful discussions have been crucial during the course of the research experience, hoping to continue the collaboration with them in the years to come. Finally, I would like to thank my family and friends for their unconditional support and continuous encouragement during these academic years and during the writing of this thesis.

Ancona, Maggio 2024

Carlo Aironi

Abstract

Recently, there has been a significant increase in the utilization of digital audio-video communication systems, even more due to the altered lifestyle imposed by the recent pandemic, which necessitated minimizing interpersonal interactions. The potential of remote communication has thus been unveiled, opening up futuristic scenarios that go beyond simple audio or video interactions but look toward immersive and interactive experiences. Despite the significant advancements in networking technologies, real-time transmissions continue to face challenges due to the possibility of data loss. Especially in voice communications, this loss not only compromises sound quality but also reduces overall intelligibility. Notably, machine learning and deep neural networks are revolutionizing ever more daily activities, among them, speech enhancement showed remarkable results in improving the speech signals affected by imperfections.

The objective of this thesis is twofold: first, to provide a novel perspective for the analysis of audio signals used in deep neural networks, and second, to develop methodologies, based on generative neural networks, for the restoration of transmission errors, potentially occurring in packet-switched networks. In the first study, we propose a novel way to model an audio signal by deriving a graph representation from its spectrogram, and exploiting graph neural network (GNN) learning models for the task of sound event classification (SEC). We then used a graph structure to exploit label co-occurrence information and improve the performance of a standard audio feature-based classifier, in a weakly labeled SEC task.

With regard to the restoration of signals that suffer packet losses, we speak of "concealment" of loss meaning that the approach used at the receiving end is to provide a reconstruction that makes the listener unaware of the loss event, thus eliminating listening fatigue. Inspired by the latest proposed solutions for packet loss concealment (PLC), we present several approaches based on generative neural networks, for loss mitigation. Each of these approaches aim at improving with respect to the most critical issues of the problem under consideration, namely the maximum width of addressable lost gaps, and the computational complexity, which may affect the ability to operate in real-time scenarios. Evaluations conducted with simulated losses and traces observed on real VoIP calls, showed state-of-the-art capabilities in modeling either speech and music signals.

Finally, GNNs were applied in a different context, to solve a well-known combinatorial optimization problem, the Linear Sum Assignment Problem (LSAP), with the goal of providing a learnable and differentiable framework, potentially useful in tasks where such assignment problems occur.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the realm of artificial intelligence, the field of deep learning has emerged as a transformative force, revolutionizing various domains by leveraging complex neural networks to extract intricate patterns from vast sets of collected data. Among its manifold applications, speech enhancement and restoration stand out as vital components, essential for improving communication systems and auditory experiences. The increasing prevalence of digital communication platforms, coupled with the omnipresence of background noise and network imperfections, has underscored the need for sophisticated techniques to enhance the quality of transmitted speech signals.

1.1 Motivation and objectives

The motivation behind this research lies in addressing the challenges posed by the efficient representation of the audio signal, whether it contains voices, music or artificial sounds, in order to extract the salient features of the contained event and fully exploit them with computational methods, based on machine learning and deep neural networks. Second, to address the problem posed by packet losses and deteriorated audio quality in communication systems. Packet loss, a common issue in networked environments, can result in information loss and degrade the quality of transmitted audio signals. Additionally, audio inpainting techniques are essential for reconstructing missing or corrupted portions of signals. This study aims to delve into the field of generative deep learning, exploring innovative solutions for speech enhancement, with a specific focus on packet loss concealment and audio inpainting. Nonetheless, some interesting research work will be presented on the use of graph-structured data, both for the analysis of acoustic features and for the semantic characterisation of a weakly labeled dataset in a sound event recognition context.

The primary objective of this research is to develop advanced deep learning models capable of mitigating the impact of packet loss on speech signals transmission, potentially present in faulty networks, and explore novel audio inpainting techniques for restoring degraded speech segments. Through a comprehensive investigation, this study seeks to contribute significantly to the refinement of communication systems, fostering clearer and more intelligible speech transmission in the presence of network irregularities. The investigation encompasses a spectrum of challenges, including understanding the dynamics of packet loss in communication networks, devising effective evaluation criteria, and exploring innovative concealment and inpainting methodologies.

1.2 Organization of the thesis

This thesis is organized into distinct chapters, each dedicated to a specific facet of the research.

Chapter 2 aims to provide the reader who already has a solid understanding of artificial neural networks, with founding concepts of the topics that will be applied in the research papers presented next, and in particular the techniques of data representation and extraction of information from audio signals, tipycally used with neural networks, the tasks of sound event classification and sound event detection, the topic of graph neural networks and methods for learning on graph-structured data, and the generative neural networks, used to produce new data by approximating probability distributions.

Chapter 3 opens with a study of existing works using graph neural networks for audio processing tasks, and continues with a literature review on the topic of multi-label classification in various fields of application. Two research works are then presented, the first on the development of an innovative methodology for the representation of audio information extracted from audio spectrogram, in graph form, and its subsequent use for a sound event classification task. The second paper deals with the evaluation of different techniques for node embedding within a hybrid framework for multi-label sound event classification, which uses a graph structure to model the relationships between the various classes of events, potentially present within a recording.

Chapter 4 introduces the packet loss concealment problem and illustrates the techniques that have been adopted in the literature for its solution, categorizing them into algorithmic and DNN-based approaches. It also illustrates the concept of audio inpainting used in the subsequent works, through generative adversarial networks for the generation of the corrupted context in the time-frequency domain. The work presented in subchapter 4.2 deals with the repair of corrupted recordings on signals containing speech, while the one presented in section 4.3, deals with the repair of signals containing musical sequences. This latter further exploits the symbolic representation given by the sequence of notes, in order to enhance the reconstruction capability. The last work presented in subsection 4.4 also discusses the spectrogram inpainting problem, emphasizing the latency and computational efficiency requirements that are often crucial for PLC systems. In order to optimize these aspects, it uses a neural network fed with complex-valued spectrograms, thereby optimally reconstructing the phase of the speech signal and reducing the computational demand. In addition, it introduces an adaptive mode for the inference process that allows the user for a trade-off between reconstruction quality and admissible latency time.

Finally, Chapter 5 presents a contribution arising from the study of graph neural networks for solving a well-known combinatorial optimization problem, the linear sum assignment problem, along with a potential application in the context of power smart grids.

Conclusions are then presented and possible future directions of the research work are outlined.

Chapter 2

Background

2.1 Audio representation for Deep Learning

In the field of academic research, a variety of audio representations have proven their effectiveness in applications related to audio analysis and synthesis. Comparisons of different sound forms have often been made to investigate the optimal representation for a given deep learning architecture. Initial experiments often involve the exploration of raw representations either in time domain or in time-frequency domain. However, contemporary studies have delved into more advanced forms, including statistical descriptors, as well as embeddings extracted from state-of-the-art DNN and vocoders, with the goal of providing richer and more meaningful descriptions.

Raw waveform

In engineering contexts, the term *raw audio* generally refers to a waveform encoded through pulse code modulation (PCM). This involves sampling the local pressure deviation from the ambient atmospheric pressure, which is continuous in both time and amplitude, presenting the waveform as a sequence of numbers (see fig. 2.1). Each number indicates an amplitude level at a given sampling instant. In order to capture all the essential information, the highest frequency of the signal must conform to the Nyquist-Shannon theorem, according to which, only frequencies less than half the sampling rate can be accurately reproduced from the sampled signal. The most common sample rate values for audio applications are 16 kHz, 22.050 kHz, 44.1 kHz 48 kHz and, less frequently, 96 kHz. In this context, each real number is mapped to an approximate fixed value within a finite set of discrete numbers; optionally a μ-law non linear companding can be used to reduce the quantization range of the raw audio. The prevailing quantization levels are 256, 65536 and 16.8 million, encoded with 8, 16 and 24 bits, respectively. This representation is considered highly informative, requiring low computational impact for direct use in deep learning networks.

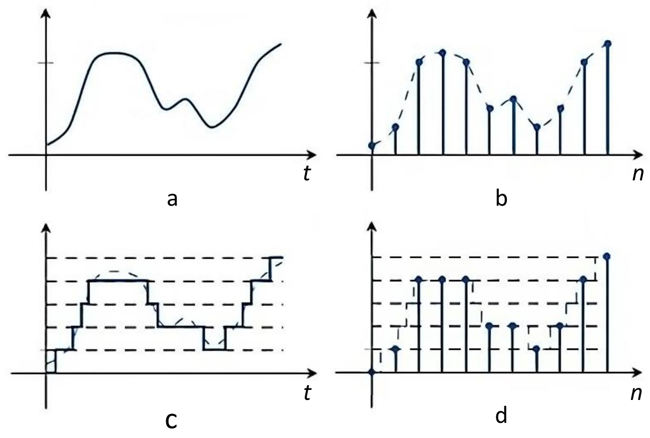


Figure 2.1: Analog waveform (a) going through the processes of time sampling (b), amplitude quantization (c), and both (d).

Spectrograms

Spectrogram is the typical bi-dimensional representation of sound in the joint time-frequency (T-F) domain. It is generated through the Short Time Fourier Transform (STFT), in which the Fourier Transform is applied to overlapping segments of the waveform (eq. 2.1),

$$STFT[f,t] = \sum_{n=0}^{L-1} s[n] \cdot w[t] e^{-j2\pi fn} \tag{2.1}$$

where $n = 0, 1, ..., L-1$ denotes the segment index, while $w[t]$ is a tapered window function, like the Hanning function, which reduces spectral leakage and improves the resolution in time.

Although the STFT operation produces a complex-valued spectrogram, most work adopting this representation for audio tasks neglects the information from the signal phase while retaining only the absolute values of STFT. In this way, deep learning architectures can be directly applied and easily borrowed from the field of image processing, due to their ability to process two-dimensional input, however, a growing body of work has demonstrated the obvious advantage of including the phase in spectrogram processing as well.

Mel and log-Mel spectrograms

Another common representation of audio signals in T-F domain is the mel-spectrogram. Its introduction stems from psychoacoustic studies according to which the human ear perceives differences between lower frequencies more easily than higher frequencies. Therefore, the mel transformation is based on the principle that equal distances on the scale have the same perceptual distance. The feature computation involves the STFT calculation and the subsequent flow through a triangular filterbank (fig. 2.2) that simulates the overall frequency selectivity of the human auditory system, expressed by the frequency warping of eq. 2.2:

$$Mel(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.2}$$

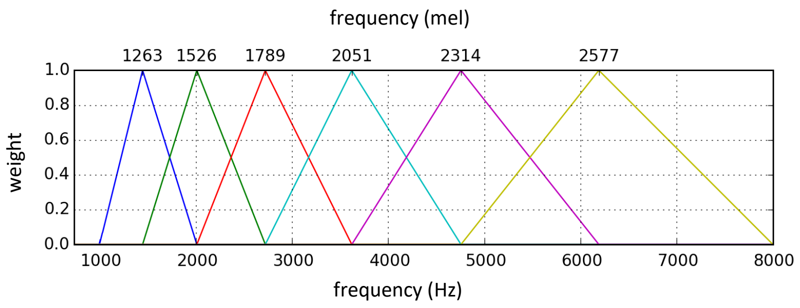


Figure 2.2: Mel filterbanks basis function including 6 bands.

Frequently, to enhance the performance of DNN architectures, logarithmic scaling is applied hence obtaining the log-Mel spectrogram.

CQT spectrograms

The CQT spectrogram is a time-frequency representation that offers a more perceptually relevant frequency resolution than standard STFT, by using logarithmically spaced frequency bins. This makes it well-suited for tasks that involve the analysis of audio signals with diverse and varying tonal content, mainly musical audio signals, but several works use CQT even on speech applications. The discrete CQT operation can be compactly expressed as follows:

$$\mathrm{CQT}\left\{x[n]\right\}(f_k) = \frac{1}{N_k} \sum_{n<N_k} x[n] w_{N_k}[n] e^{-j2\pi n \frac{Q}{N_k}} \tag{2.3}$$

where:

$$N_k = \left\lceil Q \frac{f_s}{f_k} \right\rceil \tag{2.4}$$

is the width of the k-th basis function, while

$$Q = \frac{2\pi f_k}{\Delta_k} = \left(2^{\frac{1}{b}} - 1\right)^{-1} \tag{2.5}$$

is the *Quality factor* and $w_{N_k}[n]$ is the n-th windowing function. In the above equations, f_s denotes the signal sampling rate while b is the desired number of bins per octave.

The CQT representation exhibits higher resolution for musical instruments with lower registers, and higher time resolution at higher frequencies. Furthermore, the frequency axis is equivariant to pitch translation: when altering the pitch of a note, all harmonics in logarithmic scale experience a constant shift, as they are approximately integer multiples of the fundamental frequency. This helps convolutional architectures to share structural similarities between different pitches.

Mel-frequency cepstral coefficients

Mel-frequency cepstral coefficients (MFCCs) provide a compact representation of the spectral envelope of a sound signal, which are commonly used in speech recognition systems (ASR). The calculation of the coefficients undergoes the following steps:

1. **Pre-emphasis**: The audio signal is often pre-emphasized to amplify high-frequency components. This is achieved by applying a first-order high-pass filter.

2. **Framing**: The pre-emphasized signal is divided into short overlapping frames. Each frame is typically around 20 to 30 milliseconds long, and adjacent frames overlap by a certain amount (e.g., 50 %).

3. **Windowing**: A window function (e.g., Hamming window) is applied to each frame to reduce spectral leakage.

4. **Fast Fourier Transform (FFT)**: The spectrum of each framed signal is obtained by applying the FFT. This provides information about the frequency content of each frame.

5. **Mel Filtering**: The spectrum is then passed through a bank of Mel filters, such as those presented in the previous section.

6. **Logarithm**: The logarithm of the magnitude of the filter bank outputs is taken. This helps in capturing the dynamic range of the audio signal.

7. **Discrete Cosine Transform (DCT)**: The resulting mel-frequency cepstral coefficients are obtained by applying the DCT to the log-filter bank energies. The DCT is used to decorrelate the coefficients and compactly represent the information.

The number of coefficients considered in practical applications is typically between 10 and 20, since they embody the most significant features of the vocal tract. The first few coefficients describe the coarse spectral shape, with the very first coefficient representing the average power in the spectrum and the second coefficient approximating the broad shape of the spectrum and related to the spectral centroid. The higher-order coefficients represent finer spectral details. A common method for extracting information about transitions between phonemes is to determine the first difference of MFCC features, known as the *delta* (Δ) of a feature, as well as the second difference, known as *delta-delta* (Δ^2) (see fig. 2.3).

2.2 Sound Event Detection and Classification

Sound Event Classification (SEC) consists in the automatic recognition of different sound events within a given audio recording. Furthermore, the task of discerning the temporal characteristics of each sound event, specifically the onset and offset times, is known as Sound Event Detection (SED). These processes, SEC and SED, finds a wide range of applications among different domains, like autonomous driving, wearable devices, Human-Computer Interfaces (HCI) for people with hearing impairments [3], home automation [4], surveillance systems [5], hazardous environment monitoring or as a part of Acoustic

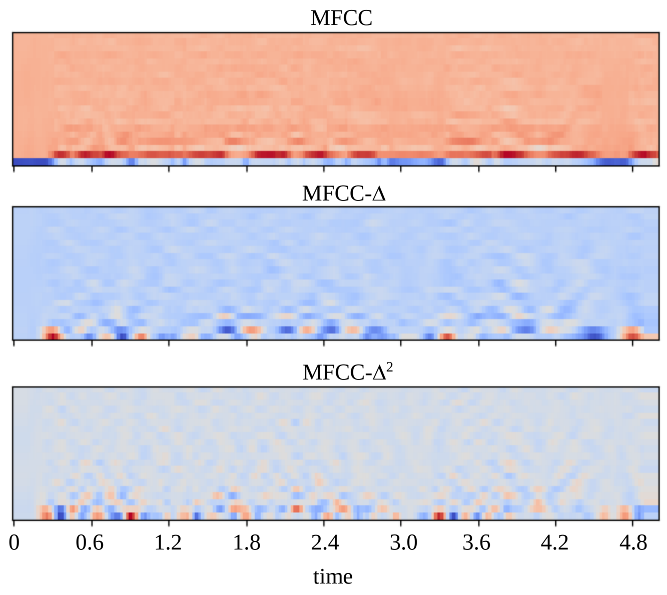


Figure 2.3: Example of MFCC, MFCC *delta*, and MFCC *delta-delta* of a speech audio file.

Scene Recognition (ASR) [6]. For the SED task, datasets with strong labels are necessary [7,8], encompassing annotations of active sound events along with their corresponding onset and offset times. Conversely, the SEC task relies on weakly labeled datasets, which solely provide annotations for the set of active sound events within each recording [9–11]. In terms of complexity, annotating strongly labeled datasets is more labor-intensive compared to weakly labeled datasets.

Both SEC and SED problems were historically first addressed with classical machine learning algorithms like Gaussian Mixture Models (GMM) [12], Support Vector Machines [13], Hidden Markov Models [14], using handcrafted features, such as MFCC coefficients [15], Mel and log-Mel filterbank features, gammatone coefficients [16] and wavelet features [17].

A significant growth in automatic recognition accuracy has been achieved by using Deep Neural Network (DNN) based methods. Addressing the Sound Event Classification (SEC) task has predominantly involved the utilization of Convolutional Neural Network (CNN) architectures [9, 10, 18, 19]. In contrast, the Sound Event Detection (SED) task, requiring the temporal localization of sound events, has demonstrated consistently favorable outcomes through the adoption of a joint architecture involving CNNs and recurrent neural networks, commonly known as Convolutional Recurrent Neural Network (CRNN) [20]. Notably, recent findings highlighted in [8] indicate that on extensive SED datasets, the performance of CNN architectures is comparable to CRNN architectures, particularly when the detection is carried out at a one-second resolution.

Several approaches have emerged aiming to simultaneously acquire knowledge in Sound Event Classification (SEC) and Sound Event Detection (SED) using solely weakly labeled data [21–23]. In a previous study [21], various well-established Convolutional Neural Network (CNN) architectures from the realm of computer vision were employed for this purpose. However, these methods operated under the assumption that weak labels were consistently active throughout the recording during training, a paradigm termed as Strong Label Assumption Training (SLAT). This assumption has been demonstrated to result in subpar SEC performance, as evidenced in [23]. An alternative approach was introduced by the authors in [24]. They proposed a method based on Fully Convolutional Network (FCN), enabling learning from weakly labeled datasets without presuming the continuous presence of weak labels throughout the recording. This training strategy is subsequently denoted as Weak Label Assumption Training (WLAT). Similar FCN-based WLAT approaches were also suggested in [11,23,25]. However, these methods were solely evaluated on smaller datasets, and their effectiveness on larger datasets remains uncertain.

2.3 Graph Neural Networks: an overview of methods and applications

Graphs have recently become a powerful component for representing diverse and intricate real-life data structures, encompassing social networks, traffic networks, information systems, knowledge graphs, atoms and molecule interactions, and physical system networks. As a versatile form of data organization, graph structures inherently capture the hidden relationships within these entities. Consequently, they can effectively depict a multitude of non-Euclidean structures essential across various disciplines and domains due to their adaptable nature. For instance, in representing a social network as a graph, individual users are portrayed as nodes, and the connections between users, such as friendships or work relationships, are represented as edges. In the biological context, proteins can be represented as nodes, and the interactions between proteins, such as dynamic interactions, are depicted as edges.

By systematically analyzing and extracting insights from graph-structured data, we gain a thorough understanding of the underlying significance within the information.

Over the past decade, there has been substantial development in the creation of machine learning algorithms tailored for learning from graph-structured data. Within this spectrum, conventional graph kernel methods [26] typically decompose graphs into distinct atomic substructures and subsequently employ kernel functions to assess the similarity between all pairs of these substructures. While graph kernels offer insights into modeling graph topology, these approaches often derive substructures or feature representations based on predefined, handcrafted criteria. These rules tend to be heuristic, susceptible to high computational complexity, consequently leading to weak scalability and suboptimal performance.

Definitions

A graph is usually defined as a set of elements $G = (V, E, X^n, X^e)$, where $V \in \{v_1 ... v_{|V|}\}$ is the set of nodes and E is the set of edges. A directed edge pointing from v_i to v_j is denoted by $e_{i,j}$, while the neighborhood of a node v is defined as $N(v) = \{u \in V | (v, u) \in E\}$ which identifies all nodes directly connected to v. The network connectivity is represented by the Adjacency matrix $\mathbf{A} \in \mathbb{R}^{|V| \times |V|}$, whose elements $A_{i,j} = 1$ if $e_{i,j} \in E$ and $A_{i,j} = 0$ if $e_{i,j} \notin E$. If the branch is weighted by the value $w_{i,j} \in \mathbb{R}$ then $A_{i,j} = w_{i,j}$.

$$\mathbf{A} = \begin{bmatrix} A_{1,1} & \dots & A_{1,N} \\ \vdots & \ddots & \vdots \\ A_{N,1} & \dots & A_{N,N} \end{bmatrix} \tag{2.6}$$

In case of undirected graphs, $\mathbf{A}$ is symmetrical since for every connection $i \to j$ the opposite connection $j \to i$ also exists. In directed graphs, by contrast, symmetry does not apply. The elements on the main diagonal represent the connection of a node with itself, which is denoted as *self loop*.

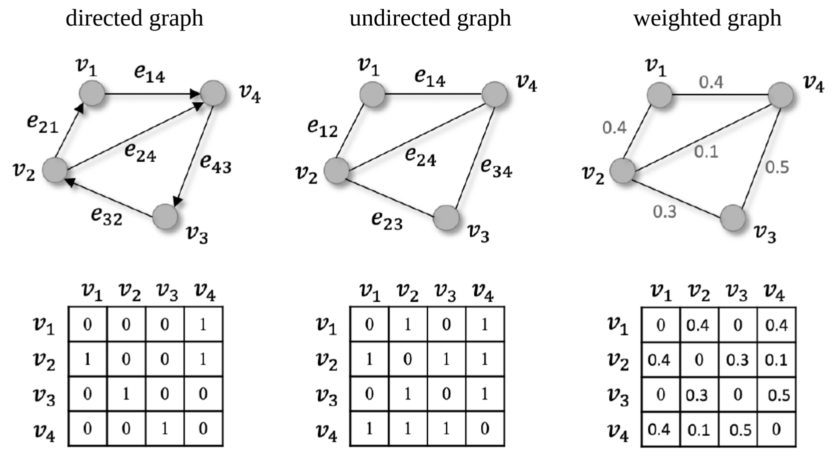
The adjacency matrix may be computationally inconvenient when the graph has many nodes and a low density of connections, a condition that occurs in most real-world graphs; in this case the matrix while having considerable size is “sparse”, that is, it has a very low density of nonzero elements. As an alternative to the matrix representation, the connections of a graph can be described with a list of coordinates indicating only the connected nodes (COO format, or Edge List).

The *degree* k_i of node i in an undirected graph is defined as the number of edges converging at node i. The average number of connections per node, $\bar{k}$ is:

$$\bar{k} = \frac{1}{|V|}\sum_{i=1}^{|V|} k_i = \frac{2|E|}{|V|} \tag{2.7}$$

The degree information of all nodes in the graph is contained in the *degree matrix* $\mathbf{D}$, whose elements $D_{i,j} = k_i$ for $i = j$ and $D_{i,j} = 0$ otherwise. The elementwise difference between entries of $\mathbf{D}$ and $\mathbf{A}$ defines the Laplacian Matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$, which is an effective way to describe the properties of a graph in frequency domain, through so-called *spectral analysis.*

A graph may have C-dimensional node attributes, identified with $X^n \in \mathbb{R}^{|V|\times C}$, as well as D-dimensional edge attributes $X^e \in \mathbb{R}^{|E|\times D}$.



$$\begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & 0 & 0 & 0 & 1 \\ v_2 & 1 & 0 & 0 & 1 \\ v_3 & 0 & 1 & 0 & 0 \\ v_4 & 0 & 0 & 1 & 0 \end{array} \quad \begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & 0 & 1 & 0 & 1 \\ v_2 & 1 & 0 & 1 & 1 \\ v_3 & 0 & 1 & 0 & 1 \\ v_4 & 1 & 1 & 1 & 0 \end{array} \quad \begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & 0 & 0.4 & 0 & 0.4 \\ v_2 & 0.4 & 0 & 0.3 & 0.1 \\ v_3 & 0 & 0.3 & 0 & 0.5 \\ v_4 & 0.4 & 0.1 & 0.5 & 0 \end{array}$$

Figure 2.4: Different types of graphs and their corresponding adjacency matrix representations.

GNN taxonomy

Graph neural networks can be broadly classified into four categories [27]: Convolutional Graph Neural Networks (ConvGNNs), Recurrent Graph Neural Networks (RecGNNs), Graph Autoencoders (GAEs), which deal with generative tasks, and Spatiotemporal Graph Neural Networks (STGNNs), which aim to learn from time-varying structures. The first two categories will be described in detail below as they are of interest to the work presented in the next chapter.

Convolutional Graph Neural Networks (ConvGNNs) arise from the attempt to generalize the notion of convolution, widely popular in Convolutional Networks (CNNs), by applying it to the graph domain. The reason CNNs are extremely effective is that they operate on data inherently described on regular domains, like images, which are in fact composed of the arrangement of pixels on an ordered grid, for which concepts of "distance" and "closeness" between two pixels can be easily defined.

Convolutional Graph Neural Networks can be classified into two macro-categories based on the domain in which they operate: spectral-based and spatial-based. The spectral-based approach defines convolution from the perspective of graph signal processing, in the Fourier domain; assuming an undirected graph a complete representation of it is given by the Laplacian matrix:

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{2.8}$$

or in some cases from its normalized version:

$$\tilde{\mathbf{L}} = \mathbf{I}_n - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \tag{2.9}$$

$\mathbf{L}$ is real symmetrical semidefinite positive and can be factorized in the form:

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top \tag{2.10}$$

in which the matrix $\mathbf{U}$ is composed of the eigenvectors of $\mathbf{L}$ forming an orthonormal space: $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$.

Let X be the matrix of node attributes, whose rows are also called *graph signals*, the spectral convolution of the graph with a kernel g can be defined by resorting to the definitions of Fourier transform (eq. 2.11) and the convolution property (eq. 2.12):

$$\mathcal{F}(X) = \mathbf{U}^\top X \tag{2.11}$$

$$X * g = \mathcal{F}^{-1}\left(\mathcal{F}(X) \cdot \mathcal{F}(g)\right) = \mathbf{U}\left(\mathbf{U}^\top X \cdot \mathbf{U}^\top g\right) \tag{2.12}$$

The spectral method has some limitations, first, the learned information is strongly related to the topology of the structure and cannot be transposed to different graphs, furthermore, the computational complexity related to the calculation of eigenvalues is proportional to $\mathcal{O}\left(|V|^3\right)$, so it may pose problems for graphs with high dimensions $|V|$.

Spatial-based methods, on the other hand, are defined by having the graph placed in a spatial domain that allows for the knowledge of network structure localized to the individual node and a set of its “neighbors”; as a result, network weights can be shared among multiple regions of the network that have similar local structures.

The early models outlined in Gori et al [28] and further elaborated in Scarselli et al [29], implicitly define the *Spatial Convolution* operator, which was later formalized with the concept of *Message Passing* (MP) mechanism; it tries to capture information by the graph manifold, edges and node feature vectors, by aggregating informative “messages” from a neighborhood of nodes.

ConvGNNs generalize the operation of convolution, which is a widely popular concept on image processing field, from grid data to non-Euclidean graph data. A general framework is described by eq. 2.13 and depicted in Fig. 2.5, it expresses the rule to update the attribute of node v_i, at network layer k:

$$\mathbf{x}_i^{(k)} = \gamma^{(k)}\left(\mathbf{x}_i^{(k-1)}, m_i^{(k)}\right) \tag{2.13}$$

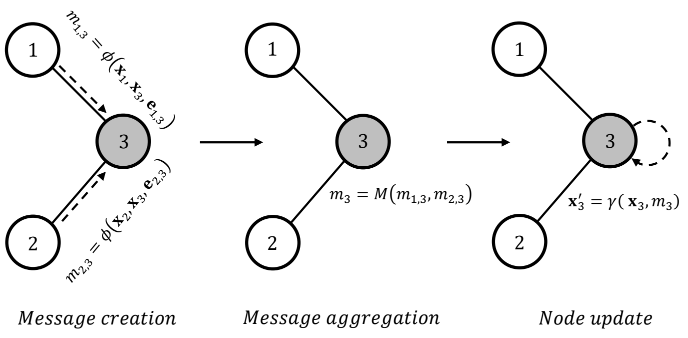
where $m_i^{(k)}$ denotes the *message*, obtained by aggregating node v_i's previous features $\mathbf{x}_i^{(k-1)}$, neighbors' features $\mathbf{x}_j^{(k-1)}$ and their edge features $\mathbf{e}_{j,i}$:

$$m_i^{(k)} = M_{j\in\mathcal{N}(i)}\left(\phi^{(k)}\left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i}\right)\right) \tag{2.14}$$

In the previous equation, M represents a differentiable, permutation invariant function (tipically, *sum*, *mean* or *max*), while γ and ϕ denote differentiable parametric functions such as MLPs (Multi Layer Perceptrons).

An interesting strenght point of ConvGNNs, compared with other architectures such as dense networks or CNNs, is that they better scale with the graph size, due to the localized action of the MP mechanism, which allows efficient parameters sharing and, consequently, memory savings.

Recurrent Graph Neural Networks (RecGNNs), iteratively employ a set of parameters across nodes within a graph to derive an high-level node representation. Due to computational limitations, initial research predominantly concentrated on directed acyclic graphs. Based on an information diffusion



$m_{1,3} = \phi(\mathbf{x}_1, \mathbf{x}_3, \mathbf{e}_{1,3})$
$m_{2,3} = \phi(\mathbf{x}_2, \mathbf{x}_3, \mathbf{e}_{2,3})$
$m_3 = M(m_{1,3}, m_{2,3})$
$\mathbf{x}'_3 = \gamma(\mathbf{x}_3, m_3)$
Message creation
Message aggregation
Node update

Figure 2.5: The three main steps of the Message Passing paradigm in GNNs.

mechanism, RecGNN updates nodes' states by exchanging neighborhood information recurrently until a stable equilibrium is reached. A node's hidden state is continuously updated by

$$\mathbf{h}_v^{(t)} = \sum_{u \in N(v)} f\left(\mathbf{x}_v, \mathbf{x}^e_{(v,u)}\mathbf{x}_u, \mathbf{h}_u^{(t-1)}\right) \tag{2.15}$$

where $f(\cdot)$ is a parametric function, $\mathbf{x}_v$ is the node feature vector and $\mathbf{h}_v^{(0)}$ is a random initial representation of node v hidden state. The sum operator enables the RecGNN to be applicable regardless of the amount and order of neighbors.

Graph pooling

Because of the potentially large number of nodes and thus the high dimensionality of the node features involved in a graph neural network (GNN), directly processing them for the final task may pose computational challenges. Therefore, a downsampling strategy becomes necessary. This strategy is given different names depending on its objective and role within the network however its goal is to reduce parameter size by downsampling nodes to generate smaller representations, thus addressing concerns such as overfitting, permutation invariance, and computational complexity. Graph pooling can be roughly categorized into *flat* pooling and *hierarchical* pooling (fig. 2.6), according to its role in graph-level representation learning. Flat pooling directly generates a mono-entity graph-level representations in one step, while the hierarchical approach coarsens the graph gradually into a smaller sized graph. In earlier approaches,

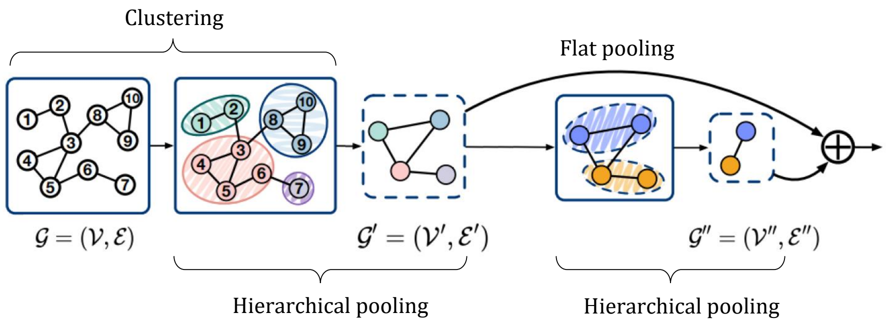

Clustering
Flat pooling
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$
$\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$
$\mathcal{G}'' = (\mathcal{V}'', \mathcal{E}'')$
Hierarchical pooling
Hierarchical pooling

Figure 2.6: An illustrative example of graph pooling.

graph coarsening algorithms utilized eigen-decomposition to condense graphs based on their topological structure. However, these methods encountered issues with time complexity. An alternative to eigen-decomposition, Graclus algorithm [30], calculates a clustering version of the original graph. Most pooling operators follow a hierarchical scheme in which the pooling regions correspond to graph clusters that are combined to produce a coarser graph. These clusters generalize the notion of local neighborhood exploited in traditional CNNs and allow for pooling graphs of varying sizes. The cluster assignments can be obtained via deterministic algorithms [31, 32] or be learned in an end-to-end fashion [33, 34]. Furthermore, node embeddings, graph topology [35], or both can be leveraged to pool graphs.

Node embedding

Node embedding plays a significant role in learning useful information from graph structured data. It has recently attracted significant interest due to its wide applications in areas such as graph visualization, link prediction, node clustering and node classification. Broadly, node embedding refers to the task of mapping each node of a graph in a lower dimensional vector space, preserving "similarity" between native and embedded domains. Embeddings should capture the graph topology along with relationships between nodes and further relevant inherent information (see fig. 2.7).

Many algorithms to generate node embeddings have been proposed [36], differing in particular in how node similarity is defined, which is a crucial aspect for effective graph-structured data modeling. On a successive work presented

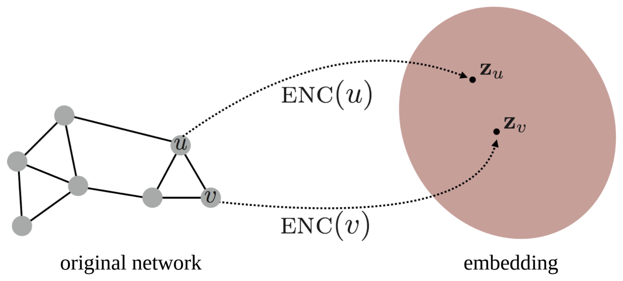


Figure 2.7: Node embedding from non-Euclidean domain to vector space.

in this thesis we focus on *node2vec* [37], a *random walk* based method which uses a random walk approach to generate network neighborhoods for nodes by stochastic sampling, and which defines similarity between nodes u and v as the probability that both u and v co-occur in the same walk over a network.

The *node2vec* algorithm was presented by Stanford University researchers on 2016, inspired by the seminal work on DeepWalk algorithm [38], which introduces for the first time the concept of random walk for embedding generation.

In *node2vec*, an encoder function $ENC(\cdot)$ maps the transition between graph domain and the embedding space; it is a lookup learnable matrix $\boldsymbol{Z}$:

$$ENC(v_i) = \mathbf{z}_i = \mathbf{Z}\mathbf{v}_i, \tag{2.16}$$

where $\mathbf{v}_i \in \mathbb{I}^{|\mathcal{V}|}$ is an indicator vector.

The "similarity" of a pair of nodes v_i and v_j is defined as the probability of visiting node v_j on a random walk of fixed length, starting from node v_i:

$$s_G(v_i, v_j) = Pr(v_j|v_i). \tag{2.17}$$

On the other hand, the concept of "similarity" in the embedding space is modeled as a softmax of the dot product between node's embedding vectors:

$$s_E(\mathbf{z}_i, \mathbf{z}_j) = \frac{\exp\left(\mathbf{z}_i^\top \mathbf{z}_j\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\mathbf{z}_i^\top \mathbf{z}_k\right)}, \tag{2.18}$$

where $\mathcal{N}_i$ is the neighbourhood set of node v_i, sampled during the random walk.

The objective of embedding is to maximize the likelihood of random walk co-occurrences, defined by the following loss function:

$$\mathcal{L} = \sum_{v_i, v_j \in V} \log \left(s_E(\mathbf{z}_i, \mathbf{z}_j) \right). \tag{2.19}$$

The strategy implemented in *node2vec* (illustrated in fig. 2.8) to define node neighbours is to use biased random walks that can trade off between local and global views of the network. Specifically, given a graph and starting from a specific node, two parameters p and q define the next hop probability during the walk, allowing to choose between a global macro-view of neighborhood (*Depth First Search* approach) or by first exploring the nearest nodes (*Breadth First Search* approach).

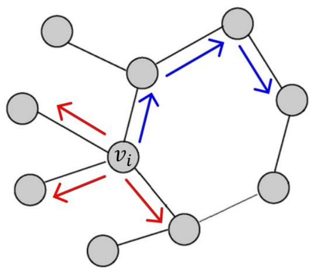


Figure 2.8: *node2vec* sampling strategies: BFS-like walks (red) vs DFS-like walks (blue), starting from node v_i.

2.4 Generative Deep Learning

Generative models in artificial intelligence are algorithms or architectures designed to learn and replicate patterns inherent in a given dataset. Unlike discriminative models, which aim to predict labels or outputs based on input data, generative models focus on understanding the underlying structure of the data and generating new samples that resemble the training data distribution. To elaborate formally, given a training data distribution $p_{data}(x)$, the objective is to acquire a distribution $p_{model}(x)$ that closely resembles the former. This task constitutes a specific instance of density estimation, a fundamental issue in unsupervised learning. There exist primarily two methodologies for addressing these challenges. One approach involves explicit density estimation, where $p_{model}(x)$ is explicitly defined and solved for. Alternatively, implicit density estimation entails learning a model capable of sampling from $p_{model}(x)$ without explicit definition. Typically, the training process of a generative model aims to optimize a score assessing the disparity between the generated distribution and the actual data distribution. For instance, Jensen-Shannon (JS) or the Wasserstein distances serve as metrics for quantifying the dissimilarity between two probability distributions. Generative models are employed across various domains and tasks, including image generation, text generation, audio generation, and more. They play a crucial role in tasks such as data augmentation, data synthesis, anomaly detection, and simulation.

A taxonomy of generative models, reported in [39] are depicted on fig. 2.9. The approaches on the left branch explicitly express the likelihood of the distribution and try to maximize it while the ones on the right address it indirectly, as in Generative Adversarial Networks (GANs) that defer to the discriminator the task of evaluating the closeness of the distributions. Between the explicit models, it is possible to distinguish the ones that work with a tractable density, and the ones that work with an intractable one and therefore have to approximate it. In the next section, the operation of only GAN-type networks will be discussed in detail, as these are the ones that will be of interest to the audio signal reconstruction methods outlined in the following chapters.

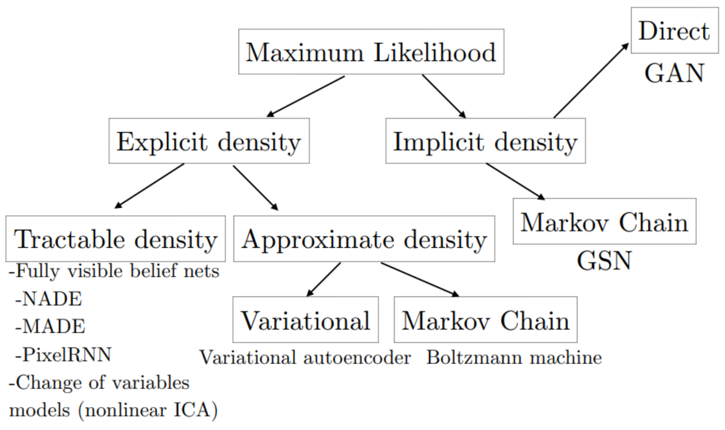


Figure 2.9: Deep generative models that can learn via the principle of maximim likelihood.

Generative Adversarial Networks

Generative Adversarial Networks (GANs, fig. 2.10 a) [40] have emerged in the past years as a powerful generative modeling technique; their objective is to replicate a data distribution in an unsupervised way. A typical GAN consists of two networks, a generator (G) and a discriminator (D). Given an input of random values sampled from a normal distribution, z (latent variable), the generator performs an upsampling in order to obtain a sample of suitable dimensions. On the other hand, the discriminator acts as a binary classifier, trying to distinguish "real" samples x (belonging to the dataset distribution) from "fake" samples, generated by G.

Both G and D are trained simultaneously in a min–max competition with respect to binary cross-entropy loss. (eq. 2.20) The final objective for G is to output samples that follow as close as possible the "real" data distribution, while D learns to spot the fake samples from real ones, by penalizing G for producing implausible results.

$$\min_{G} \max_{D} \mathcal{L}_{GAN}\left(D, G\right) = \mathbb{E}_{x}\left[\log\left(D(x)\right)\right] + \mathbb{E}_{z}\left[\log\left(1 - D(G(z))\right)\right] \tag{2.20}$$

A Conditional Generative Adversarial Network (cGAN, fig. 2.10 b) is a type of generative model that incorporates additional conditioning information c to guide the generation process. This additional information could be any kind of auxiliary information relevant to the generation task. For example, in the context of image generation, the conditioning information might include class labels indicating the type of object to be generated, or even specific attributes such as color, style, or orientation. Despite being a simple technique, it has proven to prevent mode collapse, which is a serious problem afflicting normal GANs, and allows the output of the generative network to be controlled toward a desired output. The objective of a cGAN is as follows (eq. 2.21):

$$\begin{split}\min_{G} \max_{D} \mathcal{L}_{cGAN}\left(D, G\right) =& \mathbb{E}_{x,c}\left[\log\left(D(x|c)\right)\right] + \\ &+ \mathbb{E}_{z,c}\left[\log\left(1 - D(G(z|c))\right)\right]\end{split} \tag{2.21}$$

Least Squares Generative Adversarial Networks (LSGANs) [41] are a variant of the traditional GANs that employ a different loss function to stabilize training and produce higher quality generated samples, and at the same time enhance stability and mitigate the vanishing gradient problem. LSGANs use a least squares loss function, which replaces the cross-entropy loss, with the aim to penalize the generator based on the discrepancy between the generated samples and the real samples in terms of their feature space distances.

Minimizing the objective function of LSGAN (eq. 2.22 and 2.23) yields minimizing the Pearson χ^2 divergence between the real and the approximated data distributions.

$$\min_D \mathcal{L}_{LSGAN}(D) = \frac{1}{2}\mathbb{E}_x\left[(D(x)-b)^2\right] + \frac{1}{2}\mathbb{E}_z\left[(D(G(z))-a)^2\right] \tag{2.22}$$

$$\min_G \mathcal{L}_{LSGAN}(G) = \frac{1}{2}\mathbb{E}_z\left[(D(G(z))-c)^2\right] \tag{2.23}$$

where a and b are the labels for fake data and real data, respectively, and c denotes the value that G wants D to believe for fake data.

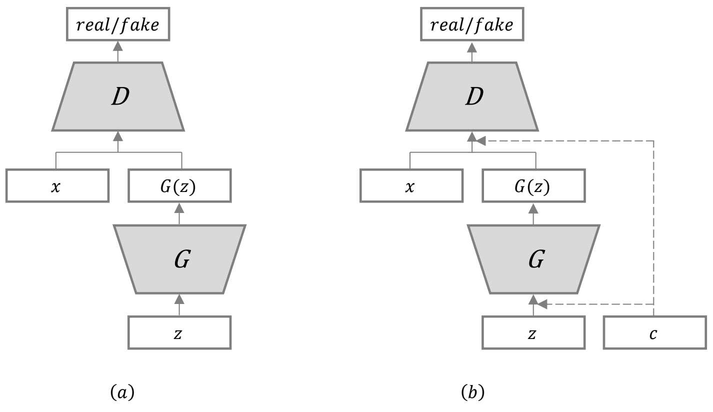


Figure 2.10: Architecture of a GAN (a) and a conditional GAN (b)

Chapter 3

Graph-based Approaches for Audio Signal Analysis

3.1 Introduction

Applications of Graph Neural Networks in Audio Processing

The prevailing methods for audio and speech processing make use of sequence-to-sequence approaches, with recurrent or convolutional architectures, either working on a set of high-level features or low-level descriptors. While works employing graphs to learn audio representations are currently limited, the interest of the research community is steadily growing. A recent study has demonstrated that graphs can effectively model audio samples, resulting in lightweight and accurate models for speech emotion recognition [42]. This study employed a basic cyclic graph to characterize a given audio data sample. Similarly, a GNN was employed to capture the interconnections among different speech segments from speakers engaged in conversation, aiming at speech emotion classification [43]. In a separate recent study [44], individual audio channels are considered as nodes when forming a speech graph for the purpose of speech enhancement tasks. This facilitates the exploration of spatial correlations among multiple channels. Furthermore, Graph Neural Networks (GNNs) have been utilized to integrate information from various heterogeneous modalities [45,46]. In [47], an ontology-informed strategy for acoustic event classification has been introduced, employing both feedforward dense layers and Graph Convolutional Networks (GCNs) as two distinct subnetworks. Another study [48] explores the likelihood of co-occurrences among acoustic events by initially extracting audio features using a CNN-based network. Modeling relationships between labels with graphs has also proved useful in few-shot scenarios [49], for example, when dealing with limited datasets. Furthermore, multitask GCN has been applied in the literature to mitigate the impact of label noise and leverage the hierarchical structure, yielding successful outcomes in audio tagging [50]. Recently, in [51] a Graph Convolutional Network has been utilized to solve the task of similarity retrieval in musical sequences, which approximates human similarity judgments between extended musical playing techniques. Finally, sound source localization on distributed microphone arrays (DMAs) [52] also benefited from the use of GNNs.

Graph Neural Networks for Multilabel Classification across different domains

Typically, *single-label classification* is about learning from a dataset where each example is linked to a unique label l, drawn from a subset of distinct labels L, where $|L| > 1$. When $|L| = 2$ this learning task is termed a *binary classification* problem or *filtering*. Conversely, when $|L| > 2$, it's referred to as a *multi-class classification* problem. In contrast, in *multi-label classification*, examples are associated with a set of labels $Y \subseteq L$, this is a common problem across several domains, including image, sound or text classification, protein function prediction or recommender systems, just to name a few.

One simplistic but common approach to tackle multi-label classification problems involves treating individual instance in isolation, converting the multi-label problem into a series of binary classification tasks, aimed at predicting the presence or absence of each class of interest. Leveraging the significant advancements in single-label classification achieved by deep neural networks, the efficacy of binarization methods has notably improved. However, these techniques are inherently constrained by their disregard for the intricate topology structure there might exists between classes. This limitation has stimulated research into effective strategies for capturing and exploring label correlations in multi-label classification. Various methods, including those based on Recurrent Neural Networks (RNNs) [53], probabilistic graph models [54, 55] or attention [56, 57], have been proposed to explicitly model label dependencies.

There is a growing trend to leverage graph neural networks for exploiting implicit relationships between data and labels in multi-label classification tasks. For instance, Saini et al. [58] conceptualize extreme classification as link prediction within a document-label bipartite graph, employing graph neural networks alongside attention mechanisms to enhance node representations through graph convolutions, across various neighborhood orders. ML-GCN [59] utilizes CNNs to generate image representations and captures label correlations by constructing a label-label graph from the label co-occurrence matrix. LaMP [60] treats labels as nodes within a label-interaction graph, computing hidden representations of each label node conditioned on the input via attention-based neural message passing. Similarly, for multi-label image recognition, Chen et al. [61] establish a directed graph over object labels, where each label node is represented by word embeddings, and GCNs are employed to map this label graph into a set of inter-dependent object classifiers. These aforementioned approaches, along with others [62–64] demonstrate the growing interest and advances in the use of graph neural networks for multi-label classification tasks and was the starting point for our research on the label-informed SEC method, illustrated in sec. 3.3.

3.2 Graph-based Representation of Audio signals for Sound Event Classification

In this section, we describe an innovative method for representing information extracted from audio spectrograms, by deriving a graph structure which can be employed by already established Graph Deep-Neural-Networks techniques to perform a wide variety of assignments. We evaluate this approach on a Sound Event Classification task by employing the widely used ESC [10] and Urbansound8k [65] datasets and compare it with a Convolutional Neural Network (CNN) based method. We show that such proposed graph-based approach is extremely compact and used in conjunction learned CNN features, allows for a significant increase in classification accuracy over the baseline with more than 50 times fewer parameters than the original CNN method. This suggests that, the proposed graph-based features can offer additional discriminative information on top of learned CNN features.

This work was presented at the 2021 European Signal Processing Conference (*EUSIPCO*) [66].

3.2.1 Proposed method

The proposed method is based on the key idea to define a graph whose informative elements (nodes, edges and their attributes) are derived from the log-Mel scale spectrogram of a signal, using an image processing approach. A segmentation procedure is performed over the log-Mel representation to isolate several different graphical entities we call *regions*. Each of these entities encloses a portion of the spectrum where the energy of the registration exceeds a certain threshold, and is then identified as a node in the target graph.

In our preliminary experiments we found out that most commonly segmentation methods used in image processing (like superpixel segmentation with N-cut [67] or SLIC [68] algorithm) perform very poorly if applied directly to spectrograms, due to the lack of both color depth and sharp edges. We instead use a Level-Set method [69] to define level curves which enclose *regions* with constant acoustic energy.

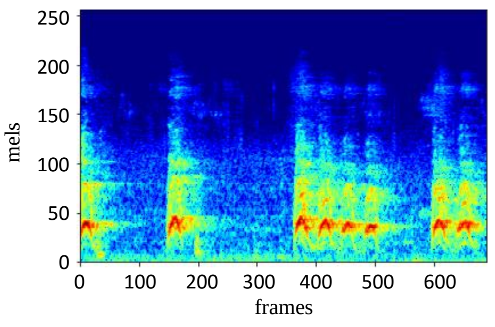
In order to reduce the amount of resulting *regions*, and thus the number of nodes, a 2D Gaussian filter with square kernel is used before the segmentation step to obtain a smoothed version of the log-Mel spectra. After smoothing, the amplitude levels are normalized to the range $[0, 1]$. Figure 3.1 shows a log-Mel scale spectrogram belonging to a clip extracted from Urbansound8k dataset (a) and its smoothed and segmented version, (b) and (c).

We denote with $x(t, f)$ the normalized and smoothed log-Mel spectra, where $0 \leq f < F$ and $0 \leq t < T$ are the Mel band and frame indexes, while F and T are the total number of Mel bands and frames. *Regions* are isolated by considering a finite set of K thresholds $\tau = [\tau_1, \dots, \tau_K]$, with each threshold $\tau_i \in (0, 1)$. Level-sets are then extracted applying a function to the normalized log-Mel spectrogram defined as follows:

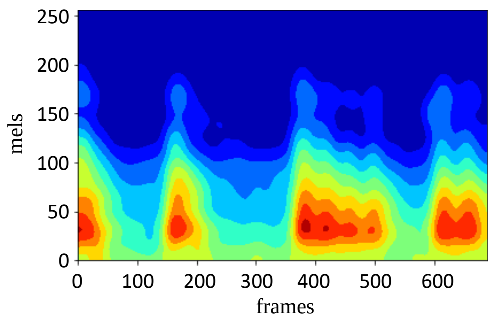
$$f\left(x(t,f), \tau_i\right) = \begin{cases} 0 & \text{if } x(t,f) \leq \tau_i \\ 1 & \text{if } x(t,f) > \tau_i \end{cases}. \tag{3.1}$$

This step returns a set of $K + 1$ discrete *levels* as it can be seen in figure 3.1 (c) where, for example, 10 *levels* in which the acoustic energy falls above a certain threshold are identified.

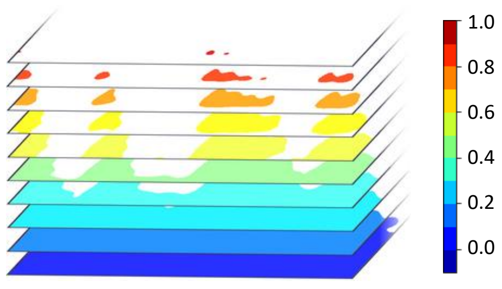
As defined in equation 3.1, each log-Mel spectra *level* is a binary matrix, it is thus possible to apply a trivial image segmentation procedure to isolate *regions* for each *level.* In detail, we can isolate each *region* $I\left(t; f\right)$, by taking



(a)



(b)



(c)

Figure 3.1: Log-Mel spectrogram of an audio clip from Urbansound8k dataset (a), smoothed and segmented version (b), exploded view (c).

each maximally contiguous area where $f(x(t,f),\tau_i) = 1$; in figure 3.1 (c) for example, there are 4 different regions in the third *level* from the top. This segmentation procedure can be implemented very efficiently using dynamic programming.

Node Attributes and Graph Edges

Regions arising from the segmentation step are assigned to graph nodes which are characterized through attributes encoding each *region* geometric shape and position. These attributes are derived from the ith and jth order image moments $M_{i,j}$, central moments $\mu_{i,j}$, and covariance matrix $cov\left[I\left(t;f\right)\right]$ of the *region* $I\left(t;f\right)$.

The ith and jth order image moments are defined as:

$$M_{ij} = \sum_t \sum_f t^i f^j \cdot I\left(t;f\right), \tag{3.2}$$

where, as before, t and f denote the frame and Mel band indexes (x-y image coordinates), while the corresponding central moments are:

$$\mu_{ij} = \sum_t \sum_f \left(t - t_c\right)^i \left(f - f_c\right)^j \cdot I\left(t;f\right), \tag{3.3}$$

where t_c and f_c are the *region* centroid along frame axis and Mel band axis, defined as:

$$t_c = \frac{M_{10}}{M_{00}},\ f_c = \frac{M_{01}}{M_{00}}. \tag{3.4}$$

The covariance matrix is obtained from the central moments:

$$cov\left[I\left(t;f\right)\right] = \begin{bmatrix} \mu_{20}/\mu_{00} & \mu_{11}/\mu_{00} \\ \mu_{11}/\mu_{00} & \mu_{02}/\mu_{00} \end{bmatrix} = \begin{bmatrix} \mu'_{20} & \mu'_{11} \\ \mu'_{11} & \mu'_{02} \end{bmatrix} \tag{3.5}$$

In this work we use eight attributes defined as follows:

- Area (which corresponds to moment M_{00}).
- Perimeter (corresponding to the moment M_{00} of the region boundary).
- The centroid spatial coordinates, t_c, f_c, and z_c (along *level* axis).

$$f_c = \frac{M_{10}}{M_{00}} \qquad t_c = \frac{M_{01}}{M_{00}} \qquad z_c \in [0.0...1.0] \tag{3.6}$$

- Orientation, defined as the angle θ between major axis and the vertical axis of an ellipse with the same image moment of the region. It can be obtained from the covariance matrix elements:

$$\theta = \frac{1}{2} arctan\left(\frac{2\mu'_{11}}{\mu'_{20} - \mu'_{02}}\right) \qquad \theta \in \left[-\frac{\pi}{2}; \frac{\pi}{2}\right] \tag{3.7}$$

- Eccentricity E, defined as the ratio between focal distance and the semi-major axis of an ellipse with the same image moment of the region. It can be obtained from the eigenvalues λ_i of the covariance matrix:

$$E = \sqrt{1 - \frac{\lambda_{min}}{\lambda_{max}}} \tag{3.8}$$

- Solidity, which encodes if the shape is convex or concave and is defined as the ratio between the area of the region and the area of a convex hull, the smallest polygon enclosing the region.

Edges of graph are defined by the following empirical rule: two nodes i, j, each corresponding to *regions* $I_i(t,f)$ and $I_j(t,f)$ are connected if they intersect: $I_i(t,f) \cap I_j(t,f) \neq \emptyset$. Edges orientation are based on the relative *levels* of the two *regions*: from lower to higher. A directed graph is thus obtained. Figure 3.2 shows the actual graph obtained from the segmented regions of figure 3.1 (c) and the criterion of connections presented above. Other heuristic criteria for defining edges have been explored, but they have led to worse performance.

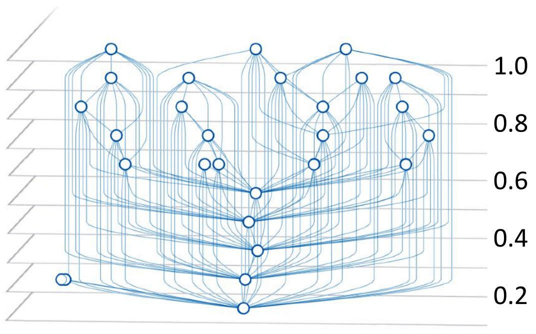


Figure 3.2: Graph originated from fig. 3.1 (c)

3.2.2 Datasets

We evaluate the performance of the proposed method using two datasets widely employed for SEC: ESC10 which is a subset of the wider ESC50 collection [70], and Urbansound8k [65]. We describe them thereafter.

ESC10 consists in 400 audio clips, grouped in 10 classes belonging to three general groups of sounds: transient/percussive sounds (*sneezing*, *dog barking*, *clock ticking*), sound events with strong harmonic content (*crying baby*, *crowing rooster*) and structured noise/soundscapes (*rain*, *sea waves*, *fire crackling*, *helicopter*, *chainsaw*). Each clip has a duration of 5 seconds and is sampled at 44100 Hz. Due to the scarcity of audio clips and to be directly comparable with [70], we applied the following data augmentation techniques :

- Random pitch shift, between -4 and +5 semitones
- Random time stretch, between -5% and +10%
- Random time shift, between 0 and +22050 samples (equivalent to a range of $[0, 0.5]\,s$.

In this way, we obtain 4400 different audio clips. Furthermore, frames with an energy content of less than -70 dB (which are presumably silence intervals) are discarded, as they can lower the generalization capability of the model. This practice induces a slight imbalance in dataset items between classes, especially those with burst (short duration) sounds, and long silence. However, this imbalance will not have a noticeable impact on classification performance.

Urbansound8k is composed by 8732 registrations of urban environmental sounds, grouped in 10 classes:

1. *air conditioner*
2. *car horn*
3. *children playing*
4. *dog bark*
5. *drilling*
6. *engine idling*
7. *gun shot*
8. *jackhammer*
9. *siren*
10. *street music*

These registrations are slightly more noisy than the ESC10 ones. Clips on Urbansound8k have different lengths and different sampling frequencies, so a resampling to 22050 Hz and a zero padding in time is performed. Sequences resulting from data augmentation and pre-processing are used in the same form to generate the log-Mel and delta spectrograms dataset and the graph dataset, and both are linked with a `clip_id` key field to ensure synchronization of the inputs on the multimodal framework training phase.

3.2.3 Experimental Setup

To evaluate the efficacy of the proposed graph-based representation we considered two different DNN classification approaches for each dataset:

- Graph-only (GNN), in which only the proposed graph-based features are used and a GNN is employed for classification at graph level.
- Hybrid (GNN+CNN), in which the graph-only approach is combined using a stacking ensemble approach with more standard CNN-based features extracted from log-Mels. The two high-level features are then combined using a Multi-Layer-Perceptron (MLP).

We compare these two approaches with a state-of-the-art CNN-based architecture proposed in [10]. More in detail, we use the short-segment majority voting architecture from this latter work as our baseline system (CNN) as well as for the Hybrid (GNN+CNN) approach. This model takes in input the log-Mel spectra of the audio signal along with the deltas, and processes them with a cascade of 2D convolutional layers with Rectified Linear Unit (ReLU) activations followed by two fully connected layers with ReLU non-linearity and an output linear layer. 60 Log-Mel bands are employed with a window of 1024 samples and 50 % overlap. All networks are trained to convergence by using early stopping and halving the learning rate if no improvement is observed for 5 epochs.

Graph-only architecture

The GNN employed in this work belongs to the category of Message Passing Neural Networks (MPNN) [71, 72] which are composed of several graph convolutional layers (GNNConv). The architecture is depicted in figure 3.3. Due to the novelty of the proposed method, could not be defined a well performing model on a graph dataset like the one we built, so the choice was made looking at models tailored on datasets with similar statistical properties, like nodes and edges distributions, and class numbers. The MPNN framework used here is described in [71] and later in [72], where it is applied to classic benchmark datasets (point cloud, chemical and social network graphs).

Each of the GNNConv layers transforms the input graph into another one with same topological structure but whose nodes have an higher dimensional feature vector. GNNConv operator is defined as:

$$x_i' = \mathbf{\Theta} x_i + \frac{1}{|N(i)|} \sum_{j \in N(i)} x_j \cdot h_{\mathbf{\Gamma}} \left(e_{i,j} \right) \tag{3.9}$$

where x_i is the input node, $\mathbf{\Theta}$ is a learnable $I \times C$ linear transformation, and $h_{\mathbf{\Gamma}}$ is a non-linear transformation with learnable parameters $\mathbf{\Gamma}$ (here we use a MLP with ReLU), fed with the node distances $e_{i,j}$ between node i and its neighboring nodes $N(i)$.

After every GNNConv layer, an Exponential Linear Unit (ELU) activation is applied and the graph is shrunk through a graph-level pooling phase applied on clusters of two nodes, as paired by the Graclus algorithm [30]. Graclus algorithm is a graph clustering method designed for large-scale graphs; it operates by iteratively coarsening the graph into a hierarchy of smaller graphs, until a desired size is reached. The process involves two main steps: coarsening and clustering. In the coarsening step, the algorithm aggregates nodes into *supernodes*, effectively reducing the graph size. The clustering step involves partitioning the coarsened graph using a spectral clustering technique. Graclus optimizes a quality measure, considering both internal edge density within clusters and external edge density between clusters. This process is repeated recursively on each coarsened level, leading to a hierarchical clustering of the original graph. The algorithm efficiently exploits the mathematical equivalence between general shear targets and k-means with weighted kernel, to work in the spatial domain, thus avoiding time-consuming computation of eigenvectors. The last layer rejects the informative content of the graph structure by keeping only node attributes (node-level pooling) which are then embedded in a single vector describing the whole graph (global-pooling). Finally, an MLP with one hidden layer and ReLU activation, is used to obtain class logits.

We used PyTorch Geometric library [73] for the implementation of the project in Python language. Each GNNConv layers has 32 channels C for $\mathbf{\Theta}$, and each

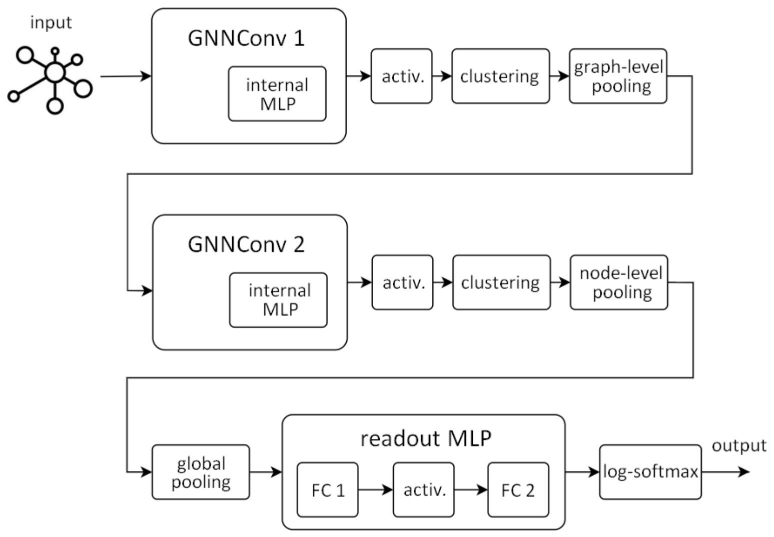


Figure 3.3: GNN architecture used in both Graph-only and Hybrid approaches.

MLP has an hidden size of 64 neurons. The readout MLP has an hidden size of 128 neurons. We use for training Adam optimizer [74], with a batch size of 32 and a learning rate of 0.001. For ESC10 we use 2 layers of GNNConv and 3 layers for Urbansound8k.

Hybrid architecture

In the Hybrid approach we combine the high-level graph-based features extracted with the MPNN described in previous section, with high-level CNN extracted features, in a stacking ensemble fashion. Regarding the CNN, we use the same architecture as in [10], which is also our baseline model.

Here, we consider for this hybrid approach only the top convolutional layers of the baseline CNN architecture [10] and concatenate the embeddings as extracted from such layers with the one obtained by the MPNN before the readout MLP. This hybrid representation is fed to an MLP and then to a linear output layer which outputs class logits. The architecture described above is depicted in figure 3.4.

We re-use the pre-trained MPNN from Graph-only approach as well as pre-trained CNN layers obtained by a re-implementation of the network from [10]. In the training phase only the fusion MLP and the output layer are updated, the CNN and GNN branches are kept frozen. We use Stochastic Gradient Descent

(SGD) optimizer with Nesterov momentum, batch size 64 and learning rate of 0.001. The fusion MLP has 1024 hidden neurons and ReLU activations.

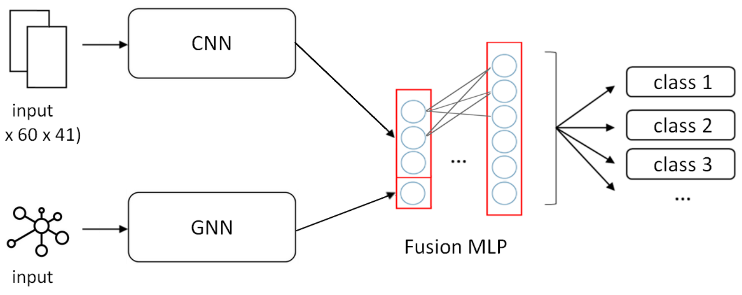


Figure 3.4: Hybrid GNN-CNN ensembling scheme

3.2.4 Results

In the following, we report and discuss the results obtained by the previously defined Graph-only and Hybrid classifiers. To be comparable with [10], we calculate accuracy by using 5-fold cross-validation for ESC10 and 10-fold cross-validation for Urbansound8k. In both cases, the fold partitions are the same defined by the dataset guidelines. We give the neural networks trainable parameters counts in table 3.1 and report accuracy in table 3.2 as well as in figure 3.5 were we show box-plots. In table 3.2 we highlight in bold best results validated through a Paired Student t-test [75] with a confidence level of 95 %, performed on 10 different runs (10 different folds for Urbansound8k and 10 for ESC10).

It can be observed that the Graph-only approach (GNN) has overall lower accuracy than the CNN-based approach, with a significant difference especially for Urbansound8k which is more noisy. On the other hand, the GNN-based classifier has more than 100 times fewer parameters and, in addition, the size of the proposed graph-based features is significantly lower than the log-Mel features. In fact, for an ESC10 audio clip the full size of log-Mels as employed in [10] is 25840 while for the proposed graph-based features only 30 nodes (on average on ESC10) are extracted, with each node having 8 scalar features as described in section 3.2.1. Thus the proposed representation is extremely compact and this can explain the difference in performance.

Nonetheless, these very compact features are able to bring considerable improvement when combined with the CNN features in the Hybrid (GNN+CNN) approach. This Hybrid model is still considerably smaller than the CNN baseline due to the use of a small fusion MLP. This result suggests that the proposed graph-based approach is able to supply additional discriminative information with respect to CNN learned features, despite the modest size of proposed representation.

Table 3.1: Total learnable parameters for different structures.

	ESC10	Urbansound8k
CNN [10]	26 M	26 M
GNN	84 K	152 K
GNN+CNN	**369 K**	**437 K**

Table 3.2: Overall classification Accuracy for 5-folds (ESC-10) and 10-folds (Urbansound8k).

Method	ESC10	Urbansound8k
CNN [10]	0.775	0.700
GNN	0.737	0.635
GNN+CNN	**0.800**	**0.730**

3.2.5 Final remarks

In this work we presented a novel method which allows to represent the information contained in the log-scaled Mel spectrograms through a graph using a segmentation step based on constant energy level curves and image processing techniques. This graph-based representation is remarkably dense and suitable for resource constrained device and edge-computing devices. The proposed approach is applied to a Sound Event Classification task using two real-world datasets and compared with a state-of-the-art CNN based model. We found that although the proposed graph-based representation is not able to compete with current state-of-the-art CNN-based models, due to its modest size, it is able to offer additional discriminative capability when used in conjunction with standard CNN learned features, significantly boosting performance and allowing to reduce drastically the size of the network.

Future work includes exploring different GNN models that could potentially further improve both the computational footprint and performance as well

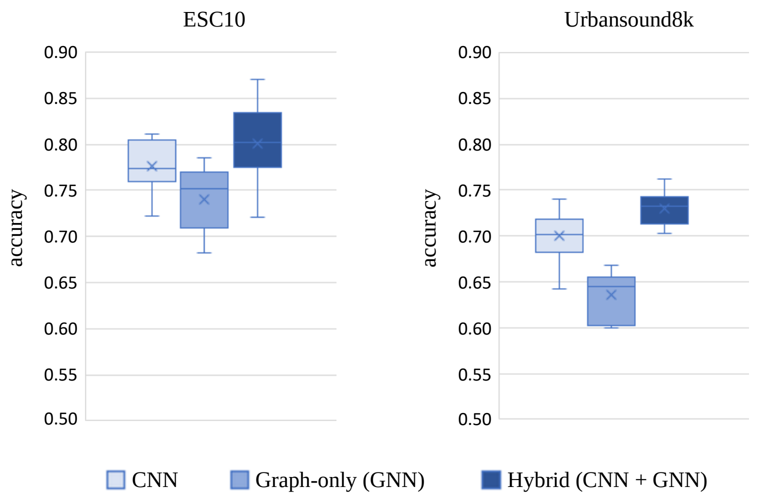


Figure 3.5: Box plots for classification accuracy for the baseline (CNN) and the proposed configurations (GNN, GNN+CNN), for ESC10 dataset (left) and Urbansound8k dataset (right).

as devising a method for learning to extract the graph-based representation without relying on any a-priori assumption, and finally, different GNN models can be evaluated or custom-tailored to the specific graph audio dataset.

3.3 Graph Node Embeddings for ontology-aware Sound Event Classification

Multi-label Sound Event Classification (SEC) is a challenging task which requires to handle multiple co-occurring sound event classes. Recent works [47,50] proposed an ontology-aware framework for multilabel SEC in which a Graph Neural Network (GNN) approach is trained to exploit labels co-occurrence information and improve the performance of a standard audio-feature based classifier via late-fusion. This GNN is fed a graph-based representation of the training set labels. In this work we adopt such framework and perform an in-depth study on how the labels embeddings used to construct the graph representation can affect the performance. We perform our experiment on the FSD50K dataset and compare different embedding strategies, of which two have already been used by previous works and two that have not yet been considered for SEC applications. Our results show that *node2vec*-generated embeddings lead to substantial performance improvements with respect to other algorithms used in previously ontology-aware SEC works. Our best node2vec model leads to an absolute improvement of 3.39 % in mean average precision, with respect to the best competing embedding strategy, with a lower number of trainable parameters.

This work was presented at the 2022 European Signal Processing Conference (*EUSIPCO*) [76].

3.3.1 Multi-label Sound Event Classification

Multi-label Sound Event Classification, also referred as multi-label Audio Tagging, aims to predict the presence of certain sound events in an audio recording. As multiple, different, sound events can occur in the same recording, in multi-label SEC class labels are not considered mutually exclusive, and algorithms must be able to handle multiple, co-occurring events. Typical SEC systems are based on deep neural networks (DNN) like, for instance, convolutional neural network (CNN) classifiers, working on either spectrogram-based features [9, 18] or directly from the raw waveform [19]. More recently, many methods use recurrent neural networks (RNN) [77–79] or convolutional recurrent neural networks (CRNN) [11, 80, 81], which can also capture temporal information of sound events, or even graph neural networks (GNN) for audio feature representation [66].

Most of these methods, do not take advantage of the intrinsic relationships between different co-occurring sound events which are detected within the same audio clip. In fact, some sound events are more likely to occur together as they can belong to the same category e.g. musical instruments or to the same acoustic scenario e.g. office environment. Some approaches have been proposed to embed information retrieved from labels relationships for classification tasks, primarily in computer vision [61, 82–84], natural language processing [85, 86] and on audio domain [47, 50]. Both [47] and [50] propose an ontology-aware SEC framework in which a graph-based DNN approach is used to exploit prior knowledge about sound events ontology and co-occurrences. This graph DNN is fed a graph representation of the training set labels relationships, e.g. in [50] graph edges are defined by the labels correlation matrix and nodes embeddings by one-hot encoded vectors. The output of this graph DNN is used in both works [47, 50] to improve SEC performance of a more “classic”, audio-features based classifier via late fusion. Hereafter, building upon these previous work, we study how the node embedding strategy employed in the graph-based DNN can affect SEC performance. In [50] only one-hot encoded vectors were considered for use as labels node embeddings. While in [47] pre-trained GloVe [87] word embeddings are used. Here we study the use of two additional node embeddings strategies which haven't been previously explored for SEC tasks: node2vec [37] learned embeddings and pre-trained fastText [88] word embeddings. We perform experiments on the recently proposed FSD50K dataset [1] and show that node2vec embeddings bring substantial performance improvements over both one-hot and pre-trained fastText and GloVe word embeddings. Importantly, these improvements come at a modest increase in the number of trainable parameters when compared to the original audio-feature classifier used alone without the ontology graph DNN branch.

3.3.2 Recurrent Graph Neural Networks

The notion of graph neural networks was initially outlined in Gori et al. [28] and further elaborated in Scarselli et al. [29]. These early models implicitly define what will be called *Spatial Convolution* [71, 89, 90], and later formalized with the concept of *Message Passing* (MP) mechanism, in which we aim to acquire and propagate information by aggregating informative *messages*, gathered from neighboring nodes. Furthermore, they fall into the category of Recurrent Graph Neural Networks (RecGNNs) [91], in which a target node's representation is learned by propagating neighbor information in an iterative manner, sharing memory parameters, both between nodes dimension and time domain, until a stable fixed point is reached.

On this work we focus on a later development proposed by Li et al. called Gated Graph Neural Network (GGNN) [92] which employs a gated recurrent unit (GRU) [93] as a recurrent function, reducing the recurrence to a fixed number of steps.

Given an initial node embedding $\mathbf{x}_i^{(0)}$, from which to define a hidden representation $\mathbf{h}_i^{(0)}$ as:

$$\mathbf{h}_i^{(0)} = \mathbf{x}_i \,\|\, \mathbf{0}, \tag{3.10}$$

where $\|$ denotes the concatenation operation. A node hidden state $\mathbf{h}_i$ at timestep t is obtained aggregating its previous hidden state and its neighboring hidden states through the following equations:

$$\mathbf{m}_i^{(t+1)} = \sum_{j \in \mathcal{N}(i)} e_{j,i} \cdot \mathbf{\Theta} \cdot \mathbf{h}_j^{(t)}, \tag{3.11}$$

$$\mathbf{h}_i^{(t+1)} = \mathrm{GRU}(\mathbf{m}_i^{(t+1)}, \mathbf{h}_i^{(t)}), \tag{3.12}$$

where $\mathbf{m_i}$ is the aforementioned *message*, built at node i, $\mathbf{\Theta}$ is a learnable parameter matrix and $e_{j,i}$ denotes the edge weight from source node j to target node i. GNN unrolls the recurrence for a fixed number of steps T and use backpropagation through time in order to compute gradients.

3.3.3 Ontology-Aware Framework

As in [47,50], also here we adopt an ontology-aware framework composed of two main modules: an *audio embedding module* and a label (graph) *co-occurrence learning module*. The proposed approach is illustrated in figure 3.6.

The audio embedding module is used to extract an high-level, condensed representation from feature vectors extracted from the audio waveform, such as log-Mel filterbank energies (LFBE). On the other hand, the label co-occurrence learning module is based on a graph neural network. This latter learns node mappings via multi-layer GGNN and is fed with a fixed pre-defined graph, where each node has an initial embedding representation. We describe both modules in detail thereafter.

Audio embedding module

In this work, we consider for the audio embedding branch the Convolutional-Recurrent Neural Network (CRNN) architecture proposed in [20] and employed as a baseline method in [1]. Other previously proposed classifiers, such as [9, 11, 18, 19, 77–81], can be employed in principle. This model is fed in input a feature vector $V \in \mathbb{R}^{F \times T}$ obtained from a clip segment, where F and T are the dimension of each feature vector (e.g. number of Mel bands) and the number of time frames of the input feature, respectively. These features are then processed by three convolutional blocks. Each block convolves the input feature map with two-dimensional filters; then, ReLU, max-pooling and batch-normalization are applied in this order. Following, a single-layer Bidirectional Gated Recurrent Unit (BiGRU) is applied on the output of the last convolutional layer. The CRNN architecture allows robust feature extraction

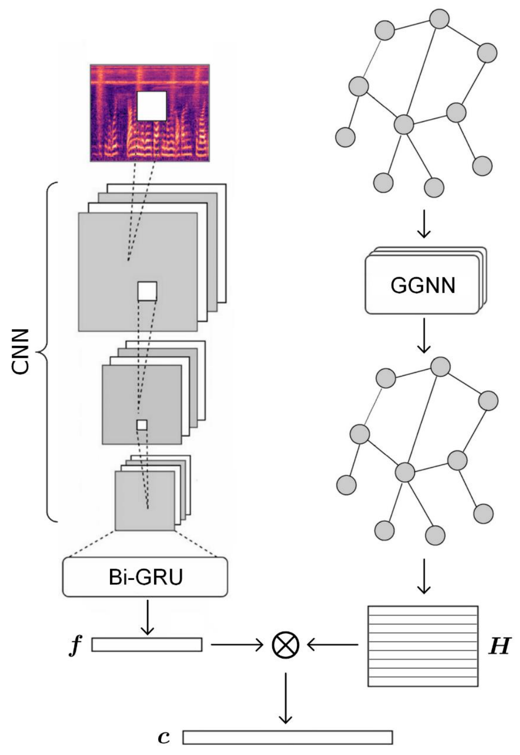


Figure 3.6: Audio embedding module (left branch) and graph co-occurrence learning module (right branch) of the proposed CRNN-GGNN architecture.

against time and frequency shifts, which often occur in SEC tasks. The forward and backward output vectors are then concatenated and placed as input of a single-layer dense network which outputs a final embedding vector $\boldsymbol{f} \in \mathbb{R}^D$ with dimension D which matches the output size of the label co-occurrence learning module.

Label co-occurrence learning module

The graph learning module depends on the initial labels node embeddings and the label correlation matrix M. These two components are in fact used to build up a graph representation where each node i is a vector of embedding size $\boldsymbol{x}_i \in \mathbb{R}^D$ and the graph edges are defined by the labels correlation matrix.

Previous works [47,48,61,84], propose several approaches to build this correlation matrix. For example, [47] builds the correlation matrix as a binary matrix, where entries $M_{i,j}$ which represents edges between i-th and j-th label are either one or zero wether or not they share a common Audioset [94] parent class label.

By contrast, here we adopt a strategy similar to [48, 61, 84] and focus on the label co-occurrence matrix of training data to build the correlation matrix, which is used to represent the structured graph of label relationships. Let N be the total number of classes, the generation process is as follows: using the training data ground-truth labels we compute the label co-occurrence matrix $M \in \mathbb{R}^{N \times N}$ whose elements $M_{i,j}$ counts the times of appearance of pairwise events (e_i, e_j); then the total occurrence of each label L_i in the training set is counted and the conditional probability matrix is calculated by:

$$M'_{i,j} = M_{i,j}/L_i. \tag{3.13}$$

The actual label co-occurrence matrix of the 200-class FSD50K dataset is depicted on fig. 3.7. Contrary to [48, 61, 84], we further remove diagonal elements from the resulting matrix, which correspond to self-loop connections. We found in our experiments that these degraded performance. The labels co-occurrence graph is then constructed from this correlation matrix M, which defines the graph edges and the node embeddings. This graph is then fed to a GNN which transforms such initial graph representation and outputs final learned node embeddings $\boldsymbol{H} \in \mathbb{R}^{N \times D}$.

Multilabel class logits $\boldsymbol{c}$ are obtained by multiplying these learned node embeddings $\boldsymbol{H}$ with the audio embedding module feature vector $\boldsymbol{f}$:

$$\boldsymbol{c} = \boldsymbol{H}\boldsymbol{f}^{\top} \tag{3.14}$$

3.3.4 Experimental Setup

In this study, we consider four different strategies for deriving the node embeddings of the label co-occurrence module. In detail, we compare one-hot vectors as used in [50], pre-trained GloVe word embeddings as used in [48, 61, 84], fastText word embeddings, as used in [61] and node2vec embeddings. These latter two have not yet been explored for SEC tasks. We will compare the performance of various configurations for such node embeddings within the framework described in sec. 3.3.3.

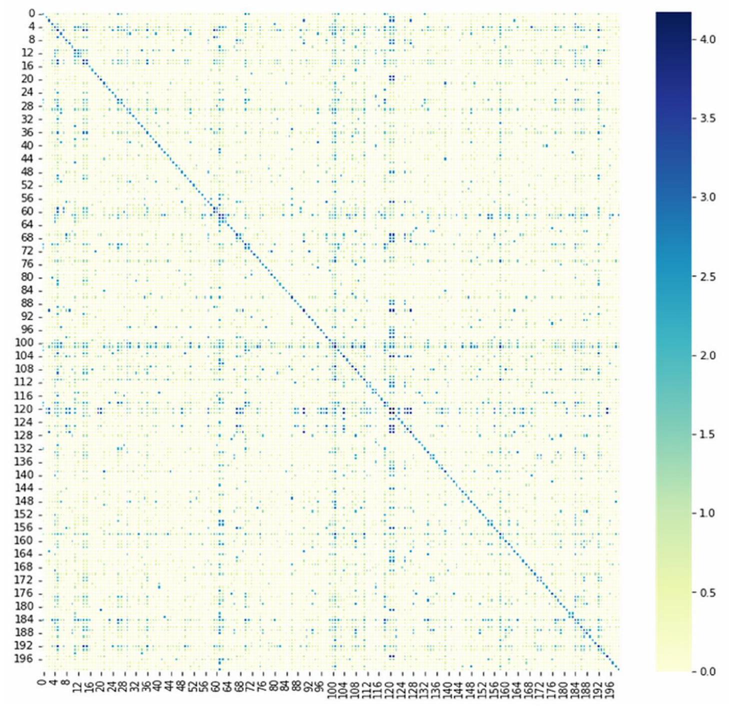


Figure 3.7: Label co-occurrence matrix of the 200-class FSD50K dataset, before removal of self-connections.

Dataset

We perform our SEC experiments using FSD50K [1], a recently introduced dataset of sound event clips with over 100 hours of manually labeled audio and 200 classes, drawn from the larger AudioSet ontology [94]. More in detail, FSD50K contains 37134 audio clips for training, 4170 audio clips for validation, and 10231 audio clips for evaluation. The clips have variable length from 0.3 s to 30 s. We used the provided training, validation and test splits to respectively train, tune and test the models employed in our experiments. Figure 3.8 shows the distribution of labels in the train and validation sets jointly, (fig. 3.8 top) and in the test set (fig. 3.8 bottom).

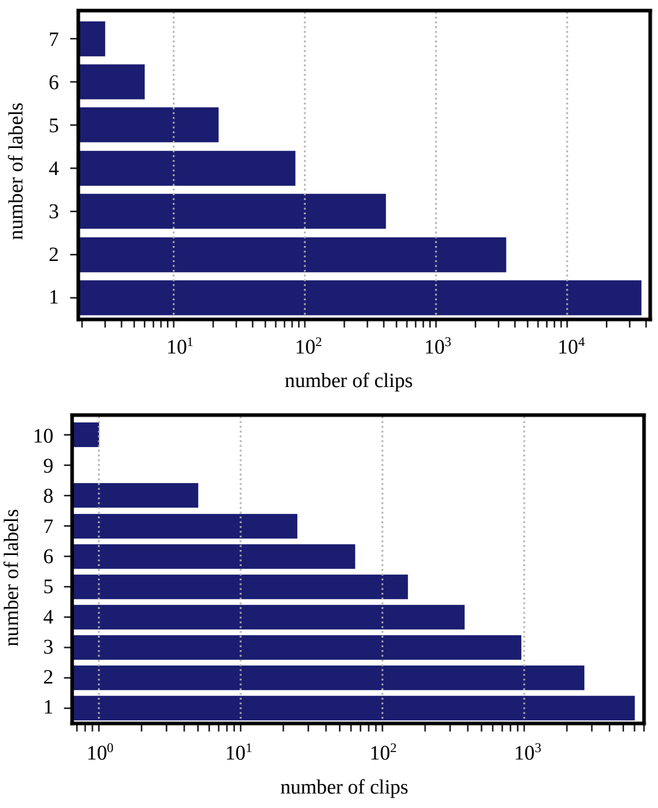


Figure 3.8: Label distribution in the train + validation set (top) and test set (bottom). Images are taken from [1].

Feature Extraction

Following [1], we resample FSD50K from the original 44.1 kHz samplerate to 22.050 kHz and extract 96-band, LFBEs to use as input features for the CRNN branch. To deal with variable-length clips, we feed to the CRNN LFBEs extracted from audio chunks of 1-second length with 50 % overlap. LFBEs are computed with a Short-Time Fourier transform window size of 30 ms with 10 ms stride. With these settings, input CRNN features have shape $(F, T) = 96 \times 101$. The label associated with each chunk is inherited from the source clip label as the dataset is weakly labeled.

Networks structure and node embedding

In the CRNN model we use 128 filters, $(5, 5)$ kernel size and unitary stride for all convolutional layers. The pooling sizes for the max-pooling layers are $(F, T) = (5, 2)$, $(4, 2)$ and $(3, 2)$. The bidirectional GRU block has an hidden layer with size 64. Regarding the graph co-occurrence branch, we employ a GGNN model composed of three hidden layers with node embeddings of the same size as input. We investigated several parameters involved in the *node2vec* algorithm, and found the best configuration to be as follows:

- $walk\_length = 20$,
- $num\_walks = 20$,
- $p = 0.1$ (return parameter),
- $q = 1$ (in-out parameter).

Evaluation metrics

We use four evaluation metric scores, which are widely used in SEC [1]: mean Average Precision (mAP), mean Area Under the Curve (mAUC), and sensitivity index d-prime (d'); Mean Average Precision is an approximation of the area under the Precision-Recall curve, which is more informative of performance when dealing with imbalanced datasets. Similarly, mean Area Under the Curve (mAUC) metric is defined as the area under the ROC curve, averaged over all sound classes; d-prime index is used in signal detection theory for similar purposes and is closely connected to AUC; it is defined as the difference between z-scores of True Positive Rate (TPR) and False Positive Rate (FPR):

$$d' = z(\text{TPR}) - z(\text{FPR}). \tag{3.15}$$

Finally, the plain label-ranking average precision (*lrap*) measures the average precision of retrieving a ranked list of relevant labels for each test clip: the system ranks all the available labels, then the precisions of the ranked lists down to each true label are averaged. Here we employ the "label-weighted" variant which calculates the precision for each label in the test set and gives them all equal contribution to the final metric:

$$l\omega lrap = \frac{1}{\sum_s |C(s)|} \sum_s \sum_{c \in C(s)} lrap(c, s), \tag{3.16}$$

where $|C(s)|$ is the number of true classes for sample s.

Model Training

Models were trained up to 60 epochs with a random weight initialization for networks on both modules; learning rate was initially set to the value of $5 \cdot 10^{-4}$, and then halved if validation mAP is not improved within 5 epochs. Adam [74] was used as optimization algorithm, with L_2 weight decay of $5 \cdot 10^{-4}$. Each model was trained with a binary cross-entropy loss with separate logits for each class, as we perform multi-label classification. We used a batch size of 256 to maximize the GPU utilization. Once the training is over, the model checkpoint with best validation mAP is selected and evaluated on the test set. For all graph-based models we tuned the size of the node embedding dimension with respect to mAP obtained on the validation set. To improve generalization, during training, we employed *mixup* augmentation [95] with parameter λ drawn from a beta distribution $Beta(\alpha, \alpha)$ with $\alpha = 0.2$.

3.3.5 Results

We report our results in Table 3.3. In detail, we compare the CRNN classifier used alone (CRNN baseline) with the approach described in Section 3.3.3, where we use a GGNN to learn co-occurrences prior knowledge as an additional information for SEC. In detail, for this latter approach, we compare different node embedding strategies while keeping the rest of the framework the same. Firstly we can see that using one-hot encoding, as in [50], improves only $l\omega lrap$ score with respect to the baseline and instead leads to a degradation of the other metrics. Secondly, we can observe that using GloVe pre-trained word embeddings for the nodes, as used in [47], leads to slighly lower performance for most metrics in our task. This could be due to the fact that FSD50K dataset is significantly more challenging than the dataset used in [47]. Instead, we found the proposed node2vec-based approach to lead in general superior results both with respect to the CRNN baseline model used alone and with the other embeddings. The most noticeable increase in mAP and mAUC values are observed for the 128 node2vec embedding dimension (0.4035 and 0.9340 respectively) but a slight performance increase is observed already with a modest 64 embedding size model. Increasing further the embedding size leads to a degradation of mAP and mAUC scores and, in general, mixed results: the 200-dim model obtains worse results than the baseline model while the 300-dim model obtains the highest d' and $l\omega lrap$ figures. These are however only marginally better than ones obtained with 128 embedding size. In general we can conclude that using node2vec with 128 embedding size leads to the best trade-off between the number of trainable parameters and performance.

Table 3.3: Classification performance for the considered architecture with different node embeddings. For each model, we report the number of trainable parameters. The metrics used are described in detail in Section 3.3.4

Model	node embedding	trainable param.	mAP	mAUC	d'	$l\omega lrap$
CRNN baseline	-	0.923 M	0.3676	0.9307	2.0950	0.5113
CRNN-GGNN	200-dim one-hot	1.285 M	0.3532	0.9264	2.0501	0.5252
CRNN-GGNN	300-dim GloVe	1.748 M	0.3644	0.9266	2.0517	0.5114
CRNN-GGNN	300-dim fastText	1.748 M	0.3696	0.9283	2.0599	0.5248
CRNN-GGNN	64-dim node2vec	0.943 M	0.3752	0.9336	2.1254	0.5434
CRNN-GGNN	128-dim node2vec	1.062 M	**0.4035**	**0.9340**	2.1310	0.5513
CRNN-GGNN	200-dim node2vec	1.285 M	0.3551	0.9255	2.0990	0.5373
CRNN-GGNN	300-dim node2vec	1.748 M	0.3725	0.9310	**2.1668**	**0.5516**

3.3.6 Final remarks

In this work we compared different node embedding strategies for ontology-aware SEC. Building from previous works, we adopt a SEC framework which is able to exploit information of sound events classes co-occurrence via a learned GNN-based module. This module is fed a graph whose nodes are labels embeddings and whose edges are defined by the various labels co-occurrences obtained from the training set. The output of this module is then combined with the output of a conventional CRNN architecture which is fed audio-related features. Using this framework, we compared different node embeddings strategies using the FSD50K dataset to perform our experiments. We show that node2vec node embeddings can outperform other embeddings strategies used in previous works on ontology-aware SEC. Our best node2vec-based method improves the SEC absolute performance up to 3.39 % in terms of clip-level mAP score, 0.0711 points on sensitivity index (d') and 0.0265 points on lωlrap score, compared with the best competing embedding approach (fastText). Future works may include exploring different audio modules, as well as extending the technique to SED, by managing the event localization at frame-level rather than at the audio clip-level.

Chapter 4

Mitigating Packet Loss Effects in Audio Transmissions: Reconstruction Techniques

4.1 Packet Loss Concealment and Audio Inpainting

Speech signals are often subject to localized distortions or even total loss of information, when data is transmitted through unreliable channels. This happens, for example, in applications such as mobile digital communications, video-conferencing systems and Voice over Internet Protocol (VoIP) calls. In such scenarios, audio frames are often encapsulated into packets, which are then routed individually through the network, sometimes taking different paths, resulting in out-of-order delivery. At the destination, the original sequence may be reassembled in the correct order, based on the packet sequence numbers. Hence, a variety of issues can occur, like packet losses, over-delay or jitter [96, 97].

The type of errors in the transmission channel can originate from bit-errors, that is transmission faults where individual bits are either omitted, added, or flipped. These errors, often encountered in wireless links, are of a low-level nature; channels vary in their error characteristics, with some being more prone to isolated bit-errors, while others experience bursts of errors where consecutive bits are affected. In packet-switched networks, bit-errors are typically addressed either at the transport layer or by identifying entire packets as lost. One method for concealing bit-errors involves establishing transition probabilities between bit configurations of consecutive packets. However, this approach becomes overly intricate with more flexible packet structures.

In this thesis we deal with the problem from a higher perspective than the telecommunications approach, so we will consider an entire packet as the smallest unit that can be affected by corruption. Lost packets occur when a packet contains a significant number of bit-errors, rendering it completely corrupted, or when network congestion makes packet delivery unfeasible. Congestion can lead to queue overflow, resulting in discarded packets, or delay transmission to the point of rendering packets effectively useless and thus "lost". Similar to bit-errors, packets can be lost individually or in bursts. The statistical distribution of error types significantly influences the choice of the most appropriate concealment method.

Finally, delayed packets represent another type of error typically caused by network congestion. Delayed packets are received correctly but experience significant delays, making timely playback impossible. Since concealment methods must be applied at the expected playback time, errors in reconstruction and potential error propagation may occur. In applications where latency is not a tight constraint, the use of delayed packets may facilitate faster recovery of the correct output.

Several approaches have been proposed in the literature to the problem of reconstructing audio sequences corrupted by the loss of fragments, where we mainly refer to Packet Loss Concealment [98–101] and Audio Inpainting [102–106]. The differences between the two terms are subtle and there is no clear-cut boundary in the use of either, but we can say that the former is mostly used in the telecommunication domain, thus in the presence of constraints posed by operating conditions such as the need to reproduce the repaired audio stream in real time and with low latency. Attention is also often focused on the computational complexity of the reconstruction algorithm in the hypothesis that this must be performed by devices with reduced capacities. Audio inpainting, on the other hand, is referred to in works that perform signal reconstruction by borrowing restoration techniques from the field of image processing, that are applied to time-frequency representation (spectrograms) of the corrupted sequence, seen as a two-dimensional image.

However, since there is no common opinion in the literature on the difference between the terms Packet Loss Concealment and Audio Inpainting, other works differentiate them on the basis of the objective that is pursued: in this case the word "concealment" is associated to any technique that attempts to overcome the packet-loss problem, by masking the lost fragments by an estimated reconstruction, which should be meaningful and consistent with the informative content of the speech message. The concealment system should also prevent audible artifacts and decrease listening fatigue, so that the listener remains unaware of any problems that have occurred. “Inpainting”, by contrast, refers to techniques that aim to precisely reconstruct the missing waveform.

4.1.1 Problem modeling

In order to experiment with Packet Loss Concealment techniques, there needs to define a model that could simulate the packet-loss behaviour. Modeling packet loss in a real-time setting is essential to be able to solve the concealment problem. PLC techniques need to be tested in realistic scenarios in order to be robust and successful. Several models has been developed for this purpose, including Bernoulli models, two-state Markov models, also referred as Gilbert models, and three-state Markov models [96].

The Bernoulli model is arguably the most basic model for packet loss, according to which each data packet is independently lost with a fixed probability p, that equals the loss rate (PLR):

$$PLR = p \tag{4.1}$$

Models based on Markov chains attempt to represent the correlation between losses and free losses in communication networks and can also be represented resorting to finite-state machines. In these models, N denotes the “Non-loss” state, to represent a correctly received packet, while L denotes the “Loss” state, to represent a lost packet. The transition probability from N to L is indicated with p, while the opposite transition is denoted by q. A two-state Markov chain model, also known as *simple* Gilbert model, is depicted in fig. 4.1. In such a model $p \neq q$ and $p + q < 1$. Unlike the Bernoulli model, it is able to capture the dependence between consecutive losses in the network. For this model, the packet loss rate is given by:

$$PLR = \frac{p}{p+q} \tag{4.2}$$

In order to better capture the temporal dependence of the packet loss process and model consecutive binary losses in burst-noise channels, the problem must be posed from a higher perspective than previous approaches. In the following models, B indicates a “bad reception” state, which does not necessarily imply

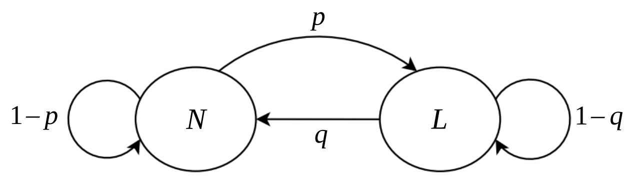


Figure 4.1: Two-state Markov chain for the *simple* Gilbert model. It matches the Bernoulli model for $q = p$.

that the received packet is considered lost, and dually, G indicates a "good reception" state. Thereby, two additional parameters, k and h, must be introduced to determine the probability of discarding the packet in each of the two states G and B; a loss can occur as independent event, with a probability of $1-k$ and $1-h$ respectively. The model proposed by Gilbert in 1960 has $k = 1$ (i.e., no loss occurs in the "good" state) and $0 \leq h \leq 1$ (see fig. 4.3); its final PLR is given by:

$$PLR = \left(\frac{p}{p+q}\right)(1-h) \tag{4.3}$$

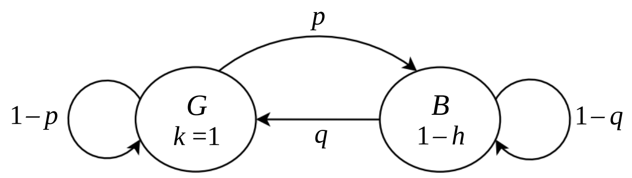


Figure 4.2: Gilbert model.

A later estension to the Gilbert model, called Gilbert-Elliot model (fig. 4.3), includes the possibility of losses in both states by introducing variability also for the parameter k. The values of k and h can be chosen arbitrarily, with $0 \leq h \leq 1$ and $0 \leq k \leq 1$. The resulting PLR is thus:

$$PLR = \left(\frac{q}{p+q}\right)(1-k) + \left(\frac{p}{p+q}\right)(1-h) \tag{4.4}$$

Three-state Markov models introduce a new "intermediate" state (I) between the "good" (G) and "bad" (B) states of the Gilbert-Elliot model. They have been shown to be more effective in reproducing trace characteristics of real samples in channels with higher losses, however, the study of complex models,

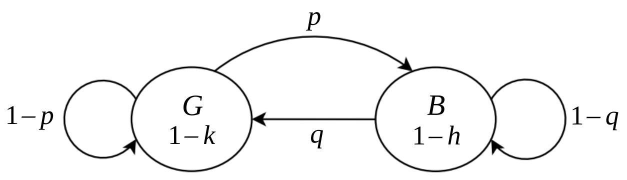

$$\begin{tikzcd} \overset{G}{1-k} \arrow[r, bend left, "p"] \arrow[loop left, "1-p"] & \overset{B}{1-h} \arrow[l, "q"] \arrow[loop right, "1-q"] \end{tikzcd}$$

Figure 4.3: Gilbert-Elliot model.

along with their variants, is beyond the scope of this thesis and can be found in the extensive review of Gouvea et al. [96].

4.1.2 Classical PLC approaches

Traditionally, Packet Loss Concealment (PLC) operates within the feature space of the codec utilized for speech data packetization, encoding, and decoding, such as EVS [107], Opus [108], and AMR-WB [109], prior to the decoding phase. This becomes particularly crucial if the codec relies on sequential packet information for decoding. These conventional approaches have undergone iterative enhancements with newer codec generations. While they generally perform well under moderate packet loss conditions, they exhibit significant degradation in speech quality during high or burst packet loss scenarios. Modern codecs enhance their performance by categorizing missing frames into various types (e.g., silence, voiced speech, non-periodic) and employing diverse prediction techniques for each frame type. Regeneration-based strategies leverage knowledge of the audio compression algorithm to extract codec parameters and reconstruct lost packets by utilizing surrounding packet parameters. Although effective, such recoveries entail considerable implementation costs. In addition to these methods, many codecs incorporate Forward Error Correction (FEC), where upon detecting adverse network conditions, the sender transmits redundant information concerning past frames to better compensate for short losses if subsequent frames are already available. However, this approach introduces network overhead and additional latency.

Receiver-based techniques, which operate independently of sender interaction, include Insertion-based schemes, wherein losses are recovered by inserting fill-in packets, often comprising noise or silence (zero-fill) sequences. While straightforward to implement, these techniques compromise speech quality after repairs.

Linear predictive coding (LPC) [110] is a technique that has been widely adopted since GSM technologies. It operates by predicting a value $\hat{x}_n$ for a lost frame based on K preceding frames $x_{n-K} \dots x_{n-1}$, using a linear recurrence:

$$\hat{x}_n = \sum_{i=1}^{K} c_i \cdot x_{n-i} \tag{4.5}$$

Another popular approach is the use of hidden Markov models (HMM) [111,112] in which previous packets help to build a statistical prior and estimate the missing packet.

Sparse representation has proven to be competitive with contemporary methods [102,113,114], exploiting the local sparse structure of audio frames in a Gabor dictionary for recovering missing or distorted audio data using extensions of the Orthogonal Matching Pursuit (OMP) algorithm.

Interpolation-based schemes recover losses by identifying matching patterns or performing interpolation to derive replacement packets resembling the originals. Although computationally intensive, these techniques outperform insertion-based schemes. Notably, the *odd-even* interpolation technique [115] separates speech samples into adjacent odd-sample and even-sample packets, with recovery accomplished by interpolating based on the available correctly received sample set, however it fails to recovery the loss if both the odd-sample and even-sample packets in a pair are lost. Waveform substitution techniques leverage the speech signal before or after the loss to find alternatives for recovering the lost segment, particularly effective when the speech signal remains relatively stable during the loss. A specialized form known as Pitch Waveform Replication (PWR) [110] is proposed for voiced segments with distinctive pitch periods P, then the missing segment can be filled by repeatedly copying the last P ms of received speech before the loss, after the loss or both. Instead, for non-speech segments, simple repetition of previous packets is used. Recent advancements introduce methods for restoring long-duration gaps in audio signals by utilizing similarity graphs to model non-local dependencies [116, 117]. Large hole inpainting is achieved by extracting features from surrounding data and identifying similar regions based on these border contents.

4.1.3 Deep PLC approaches

The advent of Deep Neural Networks (DNN) has significantly improved the quality of several audio processing tasks, leading to successful explorations of DNN architectures for deep PLC and Audio Inpainting.

Two primary frameworks define the operation of Deep Packet Loss Concealment. Firstly, in real-time scenarios, upon receipt of each segment, it immediately undergoes post-processing through the PLC algorithm. The resulting frame after post-processing could be either the original non-lost frame or the concealed frame. Typically, these segments are short in length, resembling practical packet sizes (10-20 ms). Secondly, the alternative framework involves processing larger audio segments, encompassing one or more lost packets, employing deep generative models such as Generative Adversarial Networks (GANs) or Autoencoders. Generally, the non-lost segments offer a broader context, facilitating PLC techniques in concealing the lost segments. However, the approaches in the second framework are more suited for offline processing due to their utilization of broader contexts, potentially including future segments (lookahead). Additionally, the utilization of deep models and processing larger segments makes them less suitable for real-time low-latency processing.

In the following, the works that first pioneered the use of a specific structure for PLC will be discussed, although in the years to follow numerous variants have been proposed and nowadays, approaches based on generative inference were found to be he most promising.

The work of Lee et al. [99] is the first that addresses the packet-loss problem using deep learning. They introduce a framework for a PLC algorithm consisting on a feed-forward neural network driven by the features of P consecutive past frames, which predicts the features of the succeeding frame. In the inference phase, each reliable frame is decoded as it is, while, to conceal the lost frame, the network predicts a potential reconstruction and inverts it to generate the predicted waveform. Previous work comes with two major limitations, that is it strictly relies on invertible features and uses a limited context of P preceding frames to estimate the lost frame.

Successive approaches address these issues while maintaining the same framework. The authors in [118] use the raw audio as input in an end-to-end fashion, thus removing the need for feature selection and their need to be invertible. Additionally, they apply LSTMs [119] on raw audio frames directly to estimate the succeeding frame. One of the main advantages of using recurrent cells like

LSTMs is handling long-term dependencies in sequential data, by maintaining internal memories. A crucial advantage introducted in [118] is adapting online-training into the framework, where every frame that is not lost, can be used at inference time to enhance the model by also making a training step. This would make the trained model more enhanced when it is used for a long time.

Several works [104, 120] resort to an encoder-decoder architecture, often shaped as U-Net [121], in which the inputs are encoded into a lower-dimensional representation that is then used to reconstruct the signal. The encoder relies on a traditional convolutional neural network (CNN) architecture that is passed the spectrogram representations of the pre- and post-gap context segments and encodes these into a single vector. The resulting encoding is then reconstructed by the decoder which uses deconvolutional layers to produce a single TF representation of the gap only. The authors of [122] train the U-Net using deep feature losses by employing a VGG [123] feature extractor network, wherein the network is trained to minimize the difference between the features extracted from the reconstructed spectrogram and those of the original spectrogram.

The authors in [124] first succesfully addressed PLC using a GAN variant called Speech Enhancement GAN (SEGAN) [125] that operates in the time domain by producing raw audio signals directly. The SEGAN generator G is constructed as an autoencoder, where the audio is encoded by using successive convolutional layers into a vector. This is concatenated with a vector of random noise and together, they are passed to the decoder which has a mirrored structure to that of the encoder. The decoder learns to recreate an enhanced version of the audio input to the encoder. In order to not lose low-level details of the input audio, the authors use skip-connections between the corresponding layers of the encoder and the decoder to allow information such as phase or alignment to pass. On the other hand, given a pair of an impaired speech and its enhanced version, the discriminator D is trained to classify if the enhanced speech is “real”(actually from the dataset) or “fake” (i.e. a bad imitation of the dataset).

Early methods fall into the category of *informed inpainting*, since they require as input a mask specifying which segments have been lost. However, moving toward real use cases, where such information is unlikely to be available, recent work converges on so-called *blind inpainting*, where the task of identifying the temporal location of the lost packet is included in the reconstruction process.

In order to return the signal to a time domain representation via inverse

STFT, methods working with magnitude spectra must reconstruct the phase information, so approximate algorithms such as Griffin-Lim [126] or Phase Gradient Heap Integration (PGHI) [127] are usually employed. Less frequently phase vocoders are used since these have far greater computational demands. For this reason, methods that favor the real-time aspect often agree in choosing to operate in the time domain. Alternatively, it is still possible to exploit the time-frequency representation by operating phase inpainting as well, with the processing of complex-valued spectrograms, as we proposed in our last and most recent work (section 4.4).

4.1.4 Evaluation metrics for PLC algorithms

The evaluation of a specific PLC mechanism poses a complex challenge. This assessment can be classified as a form of Speech Enhancement (SE) evaluation. The primary challenge arises from the multitude of factors influencing the perceived speech quality. While various metrics capture certain aspects of these factors, they still fall short of fully modeling realistic speech distortions [128]. Mean Opinion Score (MOS) is often regarded as the benchmark evaluation method, involving listeners manually rating enhancements on a five-point scale. Subsequently, these ratings are averaged [128]. However, while this approach can produce reliable results with a sufficient number of raters, it is also costly and time-consuming. Additionally, it does not facilitate rapid iteration on new insights, which could greatly benefit researchers in their work. Moreover, there are objective measures for SE tasks. Standard metrics can be used as objective measures, for example, Mean Squared Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Mel Cepstral Distance (MCD) and Concordance Correlation Coefficient (CCC) which measures data reproducibility, and is trending in Speech Emotion Recognition. Additional sound specific measures commonly considered are: Log Spectral Distance (LSD) [129], Signal to Noise Ratio (SNR), Signal to Interference Ratio (SIR), Signal to Distortion Ratio (SDR), and Signal to Artifacts Ratio (SAR) [130]. Perceptual Evaluation of Speech Quality measure (PESQ) emerged as a valid objective metric on a competition to develop metrics for SE tasks [128]. The PESQ algorithm operates by simulating human perception of speech quality and assigning a scores ranging from -0.5 to 4.5. Short-Time Objective Intelligibility (STOI) [131] operates on short-time segments of speech signals, typically utilizing a time-frequency representation such as the Short-Time Fourier Transform (STFT). It calculates a correlation-based measure between the processed speech and the reference speech in each time-frequency bin, aiming to capture the perceptual intelligibility of the processed speech. These metrics, while quick and easy to compute, may not have a very strong correlation with human ratings, and may be insufficiently exact when trying to compare two relatively similar models. They also require an aligned reference, which limits their use to scenarios where such a reference is available. Particularly in scenarios involving packet loss concealment with a jitter buffer and timescale modification, which are commonly implemented, the reference signal is typically unaligned, potentially leading to additional errors.

Non-intrusive DNN-based metrics were succesfully applied to the PLC problem. Despite the fact that they were trained for other type of tasks many researchers consider them to be sufficiently correlated with reconstruction quality in the presence of missing segments. Among the most widely used is the Deep Noise Suppression Mean Opinion Score (DNSMOS) [132]. In its first version it was developed as a non-intrusive metric predicting the scores from ITU-T Rec. P.808 subjective evaluation, whose the goal is to reflect the overall quality of the audio clip. Subsequently DNSMOS was related to the P.835 standard which provides 3 different scores: speech quality (SIG), background noise quality (BAK), and overall quality (OVRL) of the audio. The developed metric is proven to be highly correlated with human ratings, with a Pearson's Correlation Coefficient (PCC) of 0.94 for SIG and 0.98 for BAK and OVRL. Recently, at the 2022 Interspeech conference, as part of an initiative to foster research on the PLC topic, Microsoft researchers present PLCMOS [133], a DNN-based score in which a neural network is trained to estimate the ratings human raters would assign to an audio file. Unlike the previously described metrics, PLCMOS model has been trained with audio degraded through lossy transmissions, with real packet loss traces observed in VoIP calls and then healed using several different PLC algorithms. PLCMOS is fully non-intrusive and therefore does not require a reference signal; it has become very popular in recent times as a method of comparing PLC algorithms.

4.2 A Time-Frequency Generative Adversarial Based Method for Audio Packet Loss Concealment

In this study, we introduce a system based on a generative adversarial approach, which aims to repair the lost fragments during the transmission of audio streams. Inspired by the powerful image-to-image translation capability of Generative Adversarial Networks (GANs), we propose *bin2bin*, an improved *pix2pix* [134] framework to achieve the translation task from magnitude spectrograms of audio frames with lost packets, to non-corrupted speech spectrograms. In order to better maintain the structural information after spectrogram translation, we adopt the combination of two STFT-based loss functions, mixed with the traditional GAN objective. Furthermore, we employ a modified PatchGAN structure as discriminator and we lower the concealment time by a proper initialization of the phase reconstruction algorithm.

Experimental results show that this solution, while preserving global temporal and spectral information along with local information, can outperform competing approaches, based either on classical digital signal processing solutions or learning methods.

This work was presented at the 2023 European Signal Processing Conference (EUSIPCO) [135].

Generative Deep Learning in Speech Processing

Generative Adversarial Networks (GANs) [40] have emerged in the past years as a powerful generative modeling technique. Given the success achieved in the field of image processing, GANs have also been effective in speech processing tasks. In this regard, WaveGAN [136] represents the pioneering attempt to adapt a deep convolutional GAN (DCGAN) structure for speech, by compressing the two-dimensional image input into one-dimensional. It laid the foundations for GAN-based practical audio synthesis and for converting different image generation GANs to operate on waveforms.

Several extensions have been derived from WaveGAN; to name a few, cWaveGAN [137], which allows conditioning both G and D with additional information to drive the generation process, and Parallel WaveGAN [2], which uses a multi-resolution STFT loss along with the adversarial loss.

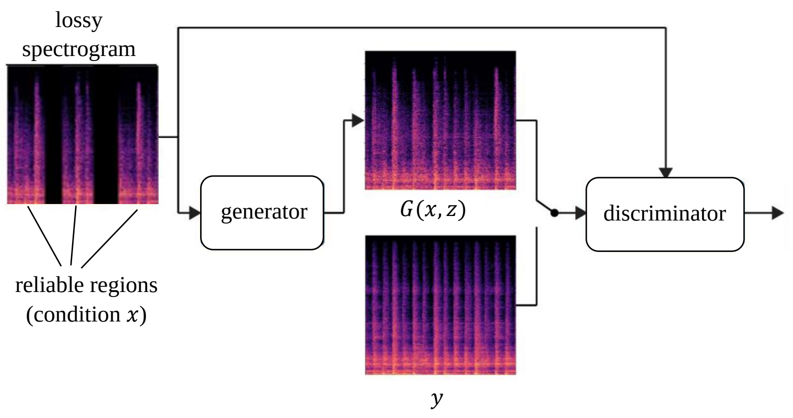


Figure 4.4: Spectrogram inpainting approach based on adversarial learning.

As outlined in [136], in the generative setting, working with compressed time-frequency representations may be problematic as the generated spectrograms are non-invertible, hence they cannot be listened to without lossy estimations, nevertheless, the practice of bootstrapping image recognition algorithms for audio tasks has become commonplace; examples include SpecGAN [136], MelGAN [138], VocGAN [139] and StyleGAN [140].

Pix2pix

Pix2pix [134] is a conditional GAN (cGAN) originally developed in 2017 by Phillip Isola, et al. for synthesizing photos from label maps, reconstructing objects from edge maps and colorizing images. Unlike a vanilla GAN which uses only random noise seeds to trigger generation, a cGAN introduces a sort of supervision by feeding the generator with the target information c, categorical labels or contextual samples. The discriminator is also conditioned by c, to help distinguish more accurately the matching and alignment of two images.

Unlike other cGAN-based works (e.g. [141] [142]), Isola et al. demonstrate that the input noise vector z does not have a significant impact if the conditioning information is strong enough, so they removed it, getting the same stochastic behavior by adding dropout layers to the generator.

The operating principle of the network we propose is illustrated in figure 4.4, it is a straightforward adaptation of the image inpainting task, for which pix2pix has been shown to be capable.

4.2.1 Neural Concealment Architecture

An overview of our bin2bin architecture is presented in Fig. 4.5. The main contribution of this paper is the adaptation of the pix2pix architecture, for the audio packet loss concealment task, through an in-depth evaluation of both generative and discriminative processes, optimized to inpaint spectrograms gaps. We adopt the term bin2bin as a direct translation of pix2pix, inspired by the fundamental unit (bin) of the discretized time and frequency axes of the spectrogram.

Generator

In the proposed bin2bin scheme, the generator architecture makes use of the U-Net [121] structural design with the insertion of skip-connections between affine layers. The U-Net is composed of a convolutional encoder that downsamples the input image in the first half of the architecture, and a decoder that upsamples the latent representation applying 2D transposed-convolutions.

The clean signal s and its lossy counterpart $\tilde{s}$, are first transformed into time-frequency spectrograms. In the provided implementation, all STFTs are computed with a 512 points Hann window, corresponding to 32 milliseconds at the sample rate of 16000 Hz, and a hop size of 64. The STFT parameters have been chosen to ensure a balanced resolution between the regions to be reconstructed and the reliable parts acting as conditioning contexts.

Our generator G accepts $1 \times 256 \times 256$ inputs, where each dimension represents, respectively, the number of *Channels*, *Frequency* and *Time* bins, hence, a portion of such size is extracted at a random time, from the aforementioned spectrograms S and $\tilde{S}$, regardless of the amount of lost fragments present inside.

Only the log-magnitude spectrogram is fed into the generator; for the training stage, the phase information is discarded, while for the test stage it is used to initialize the Griffin-Lim [126] phase reconstruction algorithm.

Discriminator

The discriminator is built on a custom architecture, specifically designed for the pix2pix framework, called PatchGAN [134]. It is basically a fully convolutional network that maps the input image into an $N \times N$ feature map of outputs Y, in which each patch y_{ij} indicates whether the corresponding portion of input

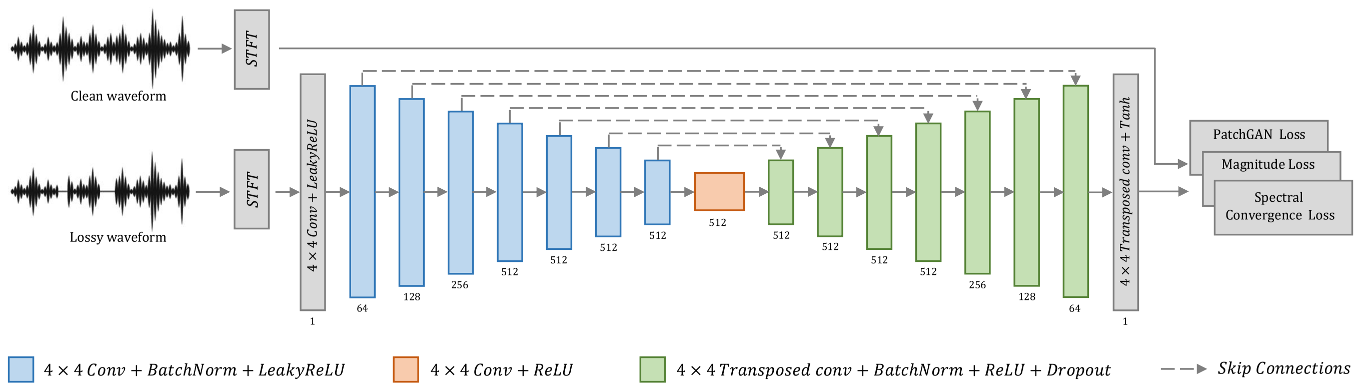


Figure 4.5: The proposed framework is composed of the U-Net for spectrogram inpainting. Deep feature loss for training the U-Net is obtained by ensembling the discriminator loss (binary cross-entropy between patches), along with the spectral distances ($\mathcal{L}_{mag}$ and $\mathcal{L}_{sc}$), between the representations of the recovered and the actual STFT log-magnitudes.

is real or fake. The patches originate from overlapped receptive fields, which can be retrieved through simple backtracking operations.

In the original paper [134], an ablation study was conducted to determine the best configuration of D (number of convolutional layers, kernels size), to maximize the evaluated metrics. In this work we focused on a similar aspect: we tested the effect of varying the size of the discriminator convolutional kernels, to achieve a rectangular receptive field, instead of the square dimension (70×70 pixels) used in pix2pix. We motivated this decision by observing that the portions of the spectrogram to be concealed extend over the entire frequency dimension, and a relatively small part of the time dimension. We traded-off between the complexity of D and the desired shape, obtaining an optimal receptive field of 162×24, with rectangular 8×2 kernels for all conv layers.

Post-processing

The generator output represents the magnitudes of the TF coefficients, both of the reliable and lost regions. The synthesis by the inverse STFT introduces an inherent cross-fading, which significantly reduces artifacts. For the phase reconstruction we used a modified version of the Fast Griffin-Lim [143] algorithm, by providing the phase of the lossy frame as an initial estimate. In this way the synthesis of the reconstructed waveform is considerably sped up; we can obtain maximum quality, with less than 10 iterations of the algorithm. The original version of the Griffin-Lim algorithm [126] is based on the redundancy of the short-time Fourier transform (STFT). A pseudo-algorithmic description is provided below:

Algorithm 1 Griffin-Lim algorithm (GLA)

```
1: Input: magnitude spectrogram $|S_0|$
2: Set: initial phase guess $\phi_0$
3: Initialize: $\hat{S}_0 = |S_0| \cdot e^{j\phi_0}$
4: for $i \leftarrow 1$ to $N$ do
5:     $\hat{S}_n = STFT(ISTFT(|\hat{S}_0| \cdot e^{j\phi_{n-1}}))$
6:     $\phi_n = arg(\hat{S}_n)$
7: Output: $|S_0| \cdot e^{j\phi_N}$
```

Loss functions

The generator model is trained by mixing the GAN objective with a traditional pixel-wise loss, between the generated reconstruction of the source spectrogram and the expected target spectrogram. In the original paper, L_1 and L_2 distances were evaluated; though both produce blurry results on image generation tasks, the former is preferred as it introduces fewer artifacts.

Differently from the original paper, we have found it more beneficial to use loss functions related to the perceptual quality of the audio signal: log-STFT magnitude loss ($\mathcal{L}_{mag}$) and Spectral Convergence loss ($\mathcal{L}_{sc}$), defined as follows:

$$\mathcal{L}_{mag}\left(S, \tilde{S}\right) = \frac{\sum_{t,f} |\log|S_{t,f}| - \log|\tilde{S}_{t,f}||}{T \cdot N} \tag{4.6}$$

$$\mathcal{L}_{sc}\left(S, \tilde{S}\right) = \frac{\sqrt{\sum_{t,f}\left(|S_{t,f}| - |\tilde{S}_{t,f}|\right)^2}}{\sqrt{\sum_{t,f}|S_{t,f}|^2}} \tag{4.7}$$

where $|S_{t,f}|$ and $|\tilde{S}_{t,f}|$ represent the STFT magnitude vector of s and $\tilde{s}$ respectively, at time t, while T and N denote the number of time bins and frequency bins of a frame.

As outlined in [144], $\mathcal{L}_{sc}$ highly emphasizes large spectral components, which helps especially in early phases of training, while $\mathcal{L}_{mag}$ accurately fits small amplitude variations, which tends to be more important towards the later phases of training.

The goal of the adversarial loss is to drive the generator model to output T-F representations that are plausible in the target domain, whereas the spectral losses regularize the generator model to output spectrograms that are a plausible translation of the source context. The combination of the adversarial loss and the spectral losses is controlled by the hyperparameters λ_1 and λ_2, both set to 250, since it has been observed that the spectral loss is more important for reconstruction than the adversarial one.

$$\mathcal{L} = \mathcal{L}_{cGAN} + \lambda_1 \mathcal{L}_{mag} + \lambda_2 \mathcal{L}_{sc} \tag{4.8}$$

The discriminator model is trained in a standalone manner in the same way as in a traditional GAN model, minimizing the negative log-likelihood of identifying real and fake images, although conditioned on the clean spectrogram, which is concatenated with $G(\tilde{S})$ to form the input of D.

We followed a common practice in training generative networks [39], which consists in balancing the evolution of training by iterating n_G times the generator weights update, for every one of D. We used the value $n_G = 10$.

The models were trained for 50 epochs, following an early stopping policy based on the spectral losses observed on the validation set. We used the Adam [74] optimizer with a learning rate of 0.0002 for both the generator and the discriminator, and a batch size of 8.

4.2.2 Datasets

We used the VCTK Corpus (Centre for Speech Technology Voice Cloning Toolkit) [145] set of data to simulate loss traces, for training and evaluation of the speech PLC model.

VCTK contains about 44 hours of clean speech from 109 English speakers, 47 males and 62 females, with different accents. To comply with the policy followed by the methods under comparison, we downsampled the audio to 16 kHz, trimmed leading and trailing silence, and split into three subsets: train, validation and test, the latter containing 5 speakers held out from the train and validation sets. We assumed that the lost packets have a duration multiple of 20 ms, and were simulated by zeroing samples of the clean waveform, finally we limited to 120 ms the maximum gap length, equivalent to 6 consecutive packets. Fig. 4.6 shows the distribution of lost gaps, obtained by injecting packets with rates in the range 10 % - 40 %.

4.2.3 Results and comparisons

The proposed PLC method has been compared with three algorithmic solutions, represented by the general purpose codecs Opus [108], WebRTC [146] and Enhanced Voice Services (EVS) [107], and against four state-of-the-art deep PLC methods: the wave-to-wave generative adversarial network (PLCNet) [147], the mel-to-wave non-autoregressive adversarial auto-encoder (PLAAE) [101], the wave-to-wave adaptive recurrent neural network (RNN) [118] and the time-frequency hybrid generative adversarial network (TFGAN) [100]. In addition, the evaluation metrics obtained by simply zero-filling the lost gaps were also reported as a baseline.

We evaluated the performances of the proposed generative inpainting method, in terms of Wide-Band Perceptual Evaluation of Speech Quality (PESQ) [148]

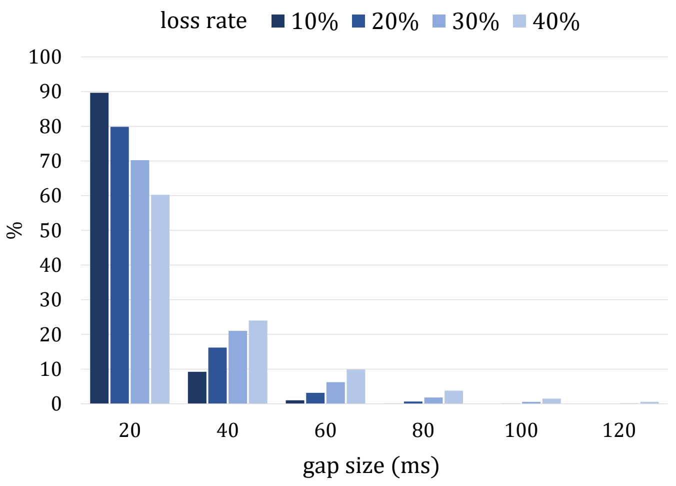


Figure 4.6: Gap size distribution obtained by injecting zero-valued frames, with different rates.

and Short-Time Objective Intelligibility (STOI) [131]. The implementations used in this paper are from [149] for PESQ, and from [150] for STOI.

Table 4.1 shows the experimental results for PESQ and STOI, under different packet loss rates, compared with the PLCNet method, It can be seen that the proposed model can achieve a significant improvement in performance, the more the loss rate increases, so it is also able to cope better with large gaps of adjacent lost packets. The improvement is notable on PESQ scores; it ranges from +6.0 % (loss rate 10 %) to +27.5 % (loss rate 40 %). The STOI shows less noticeable gains, only for higher loss rates: +2.3 % (loss rate 30 %) and +7.8 % (loss rate 40 %).

Table 4.2 and fig. 4.7 summarize the results of the proposed method with all the competing approaches. Values represent the average score of PESQ and STOI under all packet loss rates investigated. Compared with the best performing network among previous state-of-the-art systems (PLCNet), bin2bin improves PESQ by 15.3 % and STOI by 2.4 %, while, in comparison with the best codec-based concealment (EVS), the improvement rises up to 43.9 % for PESQ and 12.8% for STOI.

Figure 4.8 shows the qualitative results of a concealed 120 ms wide gap, within a test sample. This represents the worst scenario, in terms of extent of lost fragments, the network is trained to face. The system is unaware of the quantity, location and extension of lost fragments presents inside the input.

In addition, we timed the forward execution of the bin2bin inpainting process, both in a CPU environment (Intel core i7-6850K) and a GPU environment (Nvidia Titan Xp), obtaining real-time (RT) factor values of 0.17 and 0.11 respectively.

Table 4.1: Objective scores for bin2bin and PLCNet, under different packet loss rates

	Packet Loss Rate	zero-fill	PLCNet	bin2bin
PESQ	10 %	2.13	3.12	**3.31**
	20 %	1.04	2.60	**2.87**
	30 %	0.89	2.04	**2.50**
	40 %	0.81	1.71	**2.18**
STOI	10 %	0.86	**0.93**	0.92
	20 %	0.81	**0.90**	**0.90**
	30 %	0.73	0.85	**0.87**
	40 %	0.61	0.77	**0.83**

Table 4.2: Average objective scores for the comparison PLC solutions, under packet loss rate 10 %-40 %

	PESQ	STOI
bin2bin	**2.72**	**0.88**
PLCNet	2.36	0.86
PLAAE	2.04	0.84
TF-GAN	1.97	0.81
RNN	1.91	0.77
EVS	1.89	0.78
Opus	1.77	0.77
WebRTC	1.70	0.70
zero-fill	1.22	0.75

4.2.4 Final remarks

In this work, we proposed an end-to-end pipeline for spectrogram inpainting and audio concealment using a cGAN-based architecture, inspired by the popular pix2pix framework. We combined the classical discriminative loss with a linear combination of two loss functions, that are correlated with the perceptual quality of speech. In addition, we adapted the receptive field of the

PatchGAN discriminator and we used a custom initialization of the Griffin-Lim algorithm to speed up post-processing. We demonstrated experimentally that the proposed method is capable of simultaneously identifying and recovering missing parts, thus outperforming the state-of-the-art DNN method by +15.3 % on PESQ and +2.4 % on STOI, respectively. Finally, inference time evaluation suggests that this approach can be integrated into a real-time application, even with a mid-range hardware setting.

As future developments we plan to investigate the generator to directly process complex-valued spectrograms, in order to incorporate the phase reconstruction directly into the generative model.

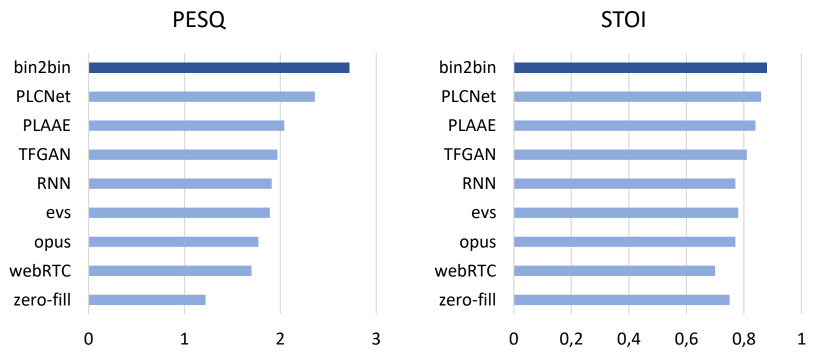


Figure 4.7: Bar graphs representing PESQ and STOI results for comparison.

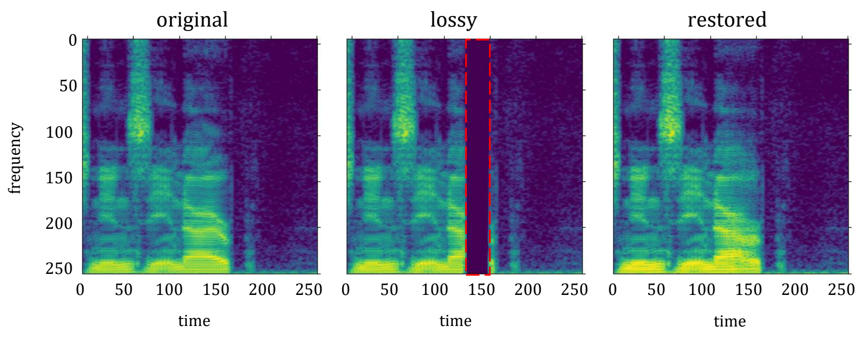


Figure 4.8: Magnitude spectrograms (in dB) of an example reconstruction. Left: original signal. Center: lossy signal with 120 ms wide gap (in red-dashed box) Right: reconstruction by the bin2bin network. The axes of the plots indicate the frequency bin and the frame index.

4.3 A Score-aware Generative Approach for Music Signals Inpainting

In this work we explore the ability to conceal missing gaps on musical signals, through a conditional-GAN (cGAN) based framework operating in the time-frequency (TF) domain, with the specific aim of inpainting large audio gaps. We address this problem by exploiting a well-established cGAN framework proposed for computer vision tasks and incorporate several modifications tailored to this specific use case. This study extends from our activities in the field of speech signal inpainting [135]. Two important novelties with respect this previous work are the introduction of a parallel objective criterion to support the GAN training and the use of CQT spectrograms. Both these novelties are motivated by the application domain i.e. musical signals. Furthermore, we add polyphonic pitch estimation into the generator flow, to translate the candidate inpainted frame from the TF domain to a piano-roll representation. This latter is then used to minimize the differences with the ground-truth roll, extracted by the aligned MIDI file. Compared to a similar approach [151] which relies on a-priori knowledge of the score in the form of a MIDI file (and enforces a tight execution of it), our method can be extended to any type of music execution, be it improvised or not. Our method will be also compared to a state-of-the-art method for audio inpainting, called GACELA [104].

This work was presented at the 4th International Symposium on the Internet of Sounds, in October 2023 [152].

4.3.1 Time-Frequency features

To efficiently process audio files, we adopt a TF representation to extract relevant input features. Commonly used methods for preprocessing audio signals include the Short-Time Fourier Transform (STFT), Mel Filterbank Cepstrum Coefficient (MFCC), Mel-Scale Spectrogram and constant-Q transform (CQT) [153].

As pointed out in most of the state-of-the-art works on Chord Recognition [154,155], Automatic Music Transcription [156,157] and Pitch Detection [158], the CQT offers advantages over STFT or Mel-scaled spectrograms in note identification. This makes it particularly well-suited for processing musical audio signals. In CQT, the frequency bins are logarithmically spaced, as a result, it exhibits higher resolution for instruments with lower registers, and higher time resolution at higher frequencies (see fig. 4.9). The combination of these resolutions makes it possible to recover pitch and timing information for individual notes, even if played simultaneously. Another significant feature of the CQT representation, particularly relevant to AMT applications is the equivariance of the frequency axis to pitch translation. When altering the pitch of a note, all harmonics in logarithmic scale experience a constant shift, as they are approximately integer multiples of the fundamental frequency. This helps convolutional architectures to share structural similarities between different pitches.

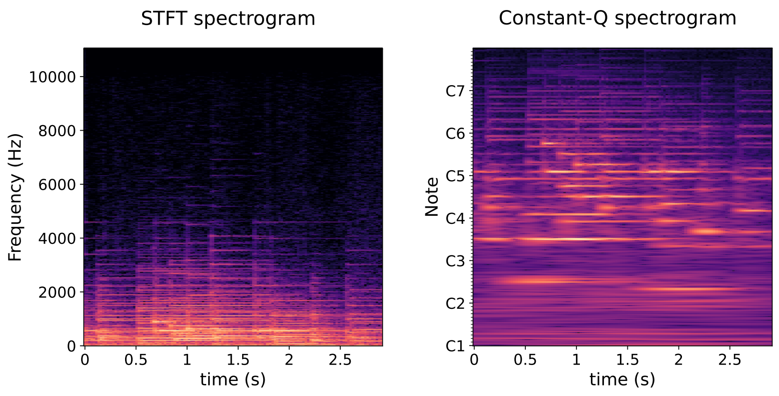


Figure 4.9: Comparison between the STFT of a piano play (left) and the CQT of the same clip (right). The time axis are consistent between the two representations, while the frequency axes are not: the STFT frequency bins spans from 0 to $f_s/2$, while CQT frequency bins ranges from 32.7 Hz (note C1) and 4186.0 Hz (note C8), as a design choice.

4.3.2 Inpainting Architecture

The proposed architecture is a generative neural inpainter that improves *pix2pix* by using different arrangements for the task of musical signals time-frequency inpainting; an overview of it is depicted in fig. 4.10. As also explained in our previous work [135], the term *bin2bin* draws inspiration from the fundamental unit (TF bin) of the time and frequency axes of the spectrogram.

Generator

The *bin2bin* scheme utilizes a generator architecture based on the U-Net [121] design with added skip-connections between homogeneous layers. The U-Net consists of a convolutional encoder that down-samples the input image in the first half and a decoder that upsamples the latent representation using 2D transposed-convolutions. Both the clean signal (s) and its lossy version ($\tilde{s}$) undergo transformation into CQT spectrograms, denoted in the following by S and $\tilde{S}$, respectively. In detail, 84 pitch levels are covered, corresponding to as many notes, from C1 (32.7 Hz) to C8 (4186.0 Hz) spanning a total of 7 octaves, each represented by 72 bins. At the sampling rate of 22050 Hz, the FFT hop size was set to 128 samples, to ensure a balanced resolution between the regions to be reconstructed and the reliable parts that serve as conditioning contexts.

The generator G is fed spectrograms of size 504×504, where each dimension represents, respectively, the number of frequencies and time bins. Only the magnitude spectrogram is fed into the generator, since it processes real-valued inputs, hence the phase information is discarded.

PatchGAN Discriminator

The discriminator module is based on a custom architecture known as PatchGAN. It operates as a fully convolutional network, transforming the input image into an $N \times M$ feature map of outputs Y. Each patch $y_{i,j}$ in this feature map indicates whether the corresponding portion of the input is real or fake. The authors who first introduced PatchGAN [134] conducted an ablation study to find the best configuration, in terms of number of convolutional layers and kernel sizes, to optimize the evaluated metrics. Similarly, in our work, we tested the effect of varying the size of the discriminator's convolutional kernels to achieve a rectangular receptive field instead of the square dimension commonly used in CV. Our motivation for this decision came from observing that the portions of the spectrogram to be inpainted span the entire frequency dimension but cover a smaller part of the time dimension. Following our previous study [135] we employed a receptive field of 162×24, using 8×2 rectangular kernels for all convolutional layers.

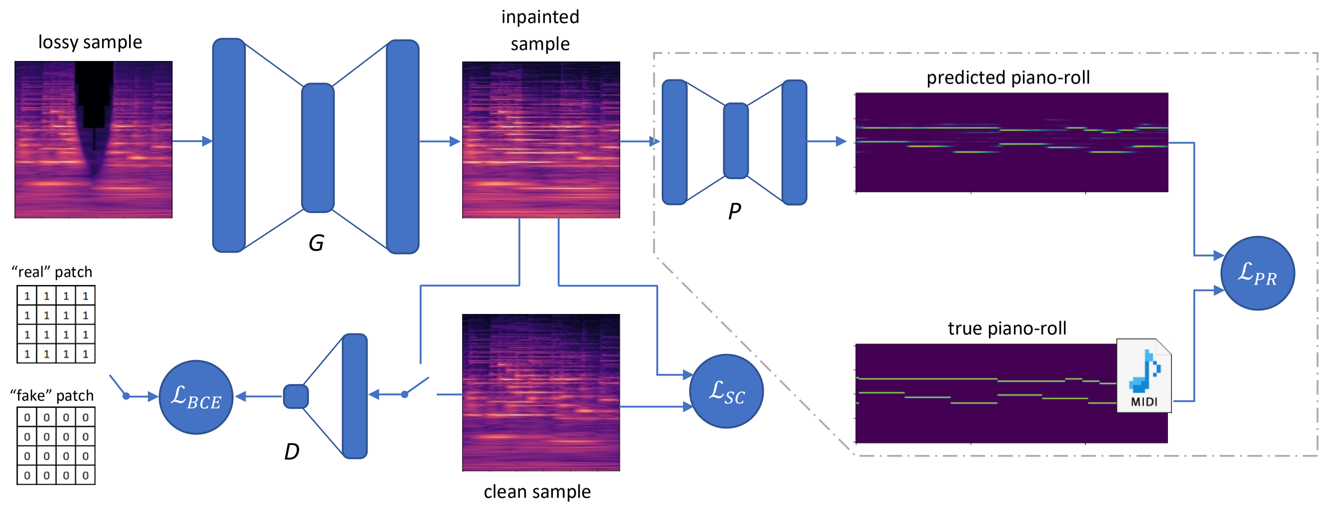


Figure 4.10: The proposed framework is composed of the U-Net for CQT spectrogram inpainting (G). Multi-objective loss for training the U-Net is obtained by ensembling the discriminator loss (binary cross-entropy between patches, $\mathcal{L}_{BCE}$), along with the Spectral Convergence $\mathcal{L}_{sc}$, and the MSE between the true and the predicted piano-roll ($\mathcal{L}_{PR}$). The dashed area encloses the novelty of this study, i.e. the pitch estimation module (P) and the symbolic loss, focused on inpainting music signals.

4.3.3 Multi-objective training

Multi-objective optimization [159] has received considerable attention in recent years as many real world problems have multiple conflicting objectives. In general, since the multi-objective deep learning methods use richer information to achieve the desired goal, their performance in comparison to the other methods should be better.

In the proposed framework, we adopt the concept of multi-objective optimization by training the generator with an ensemble of three loss functions: the first is the Binary Cross-Entropy loss $\mathcal{L}_{BCE}$ as in a classic GANs. We also propose a score loss $\mathcal{L}_{PR}$, that is the MSE computed between two piano rolls (represented as 2D images): the predicted one and the reference one, extracted from the aligned MIDI file. Its rationale is to help the generator to inpaint the signal with musical events that are likely to be present in the original score. In other words, the generator should learn to fill gaps with note pitches and timings that are harmonically and rhythmically compatible with the surrounding context (thus likely to be the ones in the original score). Finally, we employ the Spectral Convergence Loss $\mathcal{L}_{sc}$, defined as

$$\mathcal{L}_{sc}\left(S,\tilde{S}\right)=\frac{\sqrt{\sum_{t,f}\left(|S_{t,f}|-|\tilde{S}_{t,f}|\right)^2}}{\sqrt{\sum_{t,f}|S_{t,f}|^2}} \tag{4.9}$$

Reducing this loss helps improving the perceptual quality of the audio signal [144].

4.3.4 Pitch estimation

To build the piano-roll representation used in the pitch estimation module, note-on and note-off messages were extracted along with their timestamps, from each MIDI file associated with the music clip, and transformed into a 2D binary matrix consisting of 84 rows and 504 columns. The time dimension (504) coincides with the size of the aligned input CQT spectrogram, while the 84 rows represent as many pitch levels chosen in the symbolic representation. (see fig. 4.11).

The pitch estimation module we adopted was the result of a preliminary study conducted on three types of neural network layers commonly used in polyphonic pitch estimation, i.e. fully connected, LSTM and convolutional, arranged as a stack of several layers (as it is done, e.g. with MLP and CNN) or

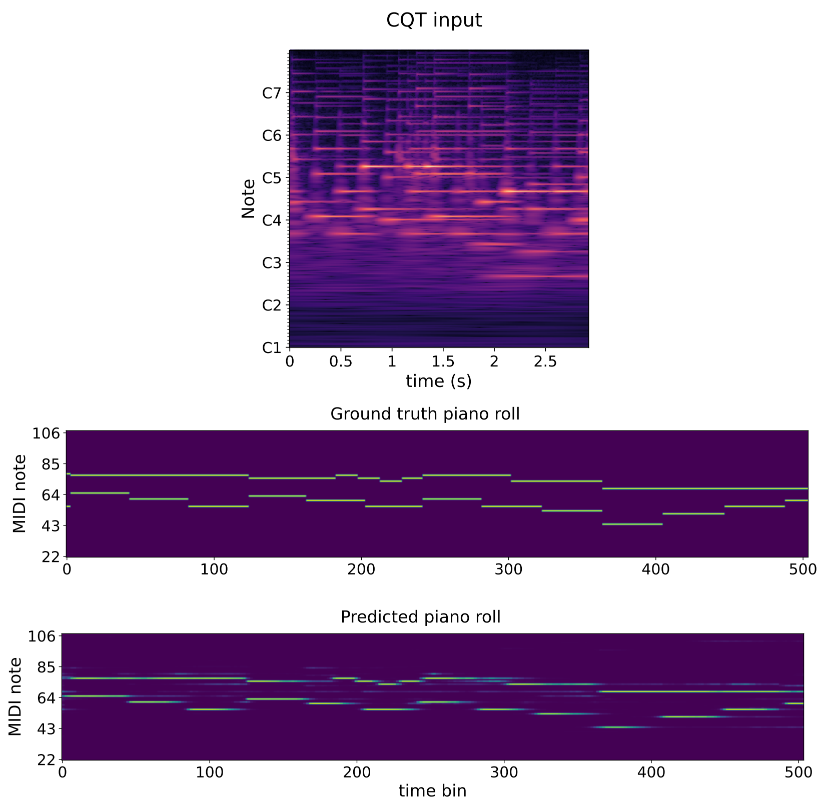


Figure 4.11: Example of a CQT frame (top) and its aligned piano-roll extracted from the MIDI file (middle). The bottom figure represents the pattern obtained placing the CQT spectrogram as input to the pitch estimation module.

in a U-Net fashion. The best choice was to use convolutional layers shaped as a U-Net, as also supported in a recent study comparing different architectures for multipitch estimation [160]. The architecture was derived, for simplicity from the same U-Net with skip-connections used in the *bin2bin* generator module, with the addition of an appropriate ConvTranspose2D layer in the upsample branch, to achieve the desired shape of the output.

The module was trained for 200 epochs, following the criterion of minimizing the Mean Squared Error (MSE) between the predicted and actual roll. The MSE criterion is used, since the estimated piano-roll contains continuous values in $[0, 1]$, and thus comparison with the ground truth piano-roll is not strictly a binary classification task.

4.3.5 Experiments

During our experiments, we found that training the network with damaged frames larger than those actually used during testing, increases the network performance. For this reason we trained the models with lossy gaps of 900 ms and tested the model with either "small" or "medium" lossy gaps of 375 ms and 750 ms, respectively. The training gap width is the largest we could employ without affecting the network ability to converge during training. In contrast, GACELA is constrained to use the same gap size in both training and test. The lossy gaps are always at the center of the context frame (see fig. 4.12). We conducted the experiments both with and without the support of the piano-roll based loss, obtaining two different architectures which will be later referred to as *bin2bin* and *bin2bin-MIDI*.

Dataset

Our experiments were conducted using the MAESTRO dataset [161], which includes nearly 200 hours of paired audio and MIDI recordings from nine years of International Piano-e-Competition events. More detailed statistics are reported on Table 4.3. The MIDI data in this dataset provides information on key strike velocities and sustain pedal positions, although we used only the note-on note-off informations. Both the audio and MIDI files are precisely aligned with an accuracy of approximately 3 ms and divided into individual musical pieces. The uncompressed audio in the dataset is of CD quality or higher, with a sampling rate of either 44.1 or 48 kHz and a 16-bit PCM stereo format. To ensure proper training, validation, and testing, we adhere to the specific split configuration proposed by the dataset's creators. This configuration ensures that the same musical composition, even if performed by multiple contestants, does not appear in multiple subsets.

Table 4.3: Statistics of the MAESTRO dataset

Split	Duration (hours)	Size (GB)	Notes (million)
Train	159.2	96.3	5.66
Validation	19.4	11.8	0.64
Test	20.0	12.1	0.74
Total	**198.6**	**120.2**	**7.04**

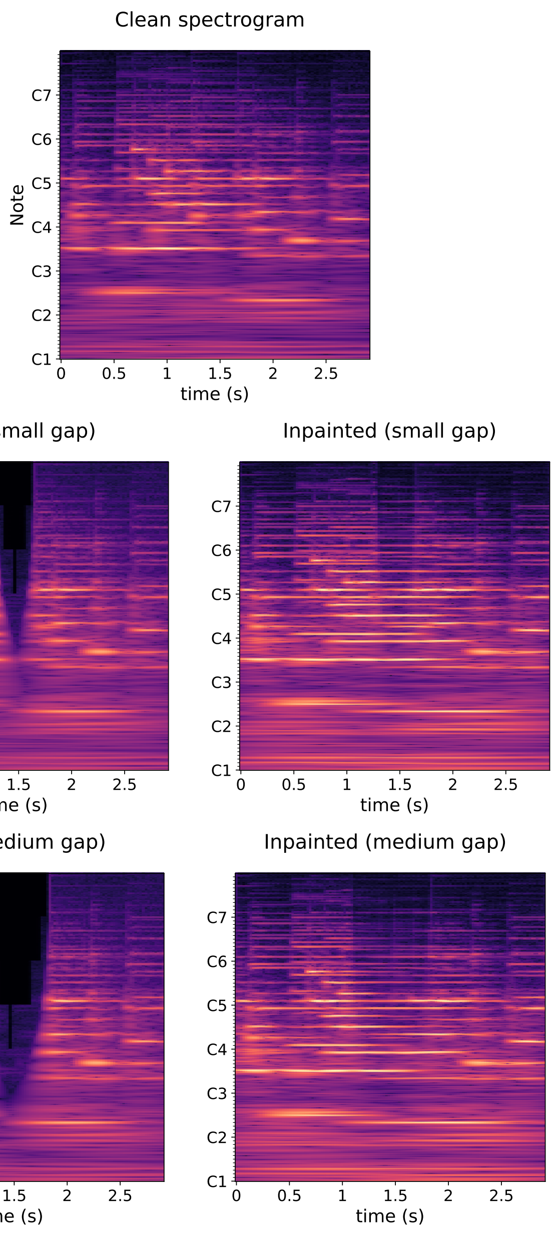


Figure 4.12: Example of CQT spectrograms inpainted with the *bin2bin* framework. Due to the variable length of the CQT basis filters, the damaged gaps affect a portion of the spectrogram unevenly distributed over time.

Post-processing

Since the generator output represents the magnitudes of the CQT coefficients of both the damaged regions and the reliable surrounding contexts, we had to apply a phase reconstruction using the Fast Griffin-Lim algorithm (FGLA) [143], with a map of all zeros as the initial phase guess. The phase recovery algorithm is a critical issue in audio processing approaches working with real-valued spectrograms. We made this choice favoring the quality of the reconstruction, but the high number of iterations required by FGLA makes it challenging to use in real-time applications.

Next, we minimized the reconstruction error, introduced by both the generator and the time conversion, by merging the inpainted corrupted region with the reliable surrounding context, using a cross-fade operation. We adopt the ascending and descending profiles of a Hann function, to respectively fade-in and fade-out the amplitude of the waveforms involved.

Comparative method: GACELA

As a baseline we used the GACELA context encoder [104], which is currently considered state-of-art in music inpainting. Briefly, GACELA is a cGAN framework proposed in 2020 by A. Marafioti et al, which performs Mel-scaled spectrogram inpainting using five parallel discriminators with increasing resolution of receptive fields. It relies only on cross-entropy loss while it does not employ reconstruction losses, neither in the frequency nor in the time domain, and this aspect, in our opinion, strongly penalizes the final reconstruction. The authors provide results based solely on subjective listening tests, practiced on a subset of files, thus we replicated the experiments evaluating two different types of objective metrics, on all files in the test set.

Evaluation metrics

To evaluate the inpainting methods numerically, we calculate the Objective Difference Grade (ODG) [162] and the Structural Similarity Index (SSIM) [163], between the outputs and the ground truth magnitude spectrograms.

Objective Difference Grade is the overall quality measure introduced in PEAQ (Perceptual Evaluation of Audio Quality) [162] which is considered standard for audio quality evaluation. PEAQ algorithm performs a direct comparison between a recovered signal and a reference signal and is designed to mimic perceptual quality ratings made by a human listener. Its output ranges from 0 (imperceptible difference) to -4 (very annoying artifact). We used the MATLAB implementation from [164] which is based on the ITU-R recommendation (BS 1387) of 1999 [165].

The Structural Similarity Index (SSIM) [163] is a pixel-wise quality metric first introduced in Computer Vision but quite popular also for audio synthesis [151,166]. It is defined as the weighted product of three factors: luminance (l), contrast (c) and texture (s), consisting of the local means, standard deviations and cross-covariance statistics for spectrograms S and $\tilde{S}$:

$$\mathrm{SSIM} = \left[l\left(S, \tilde{S}\right)\right]^{\alpha} \times \left[c\left(S, \tilde{S}\right)\right]^{\beta} \times \left[s\left(S, \tilde{S}\right)\right]^{\gamma} \tag{4.10}$$

4.3.6 Results

The results of the evaluation tests are reported on the following tables. The experiments demonstrate that our model significantly outperforms the GACELA framework, being able to produce a more plausible and artifact-free gap inpainting.

Table 4.4 and the box-plot depicted in Fig. 4.13 show that *bin2bin* achieves a gain of about 10.5 % (without the piano-roll loss) and 13.3 % (with the piano-roll loss), for the “small” gap condition, while the performance improvement is about 8.5 % and 10.3 %, for the “medium” gap condition, in terms of ODG score with respect to the baseline. Given the narrow margin of gain between the two variants of the *bin2bin* method, the evidence of ODG improvement, obtained by introducing the additional loss, was also successfully validated with a statistical Z-test on the mean, with a significance level of 0.05.

Table 4.5 and the box-plot of Fig. 4.14 show the results obtained on the same test files in terms of SSIM index. Here the advantage between GACELA and *bin2bin* is noticeable as well. Our method allows an improvement of 12.6 % (“small” gap) and 13.6 % (“medium” gap), however, the SSIM index between the two training schemes, with and without the piano-roll loss, is about the same magnitude, with no evidence of statistical difference.

Finally, in fig. 4.15 we show in detail the clipping of a reconstructed area, compared with the original one. It can be seen that the *bin2bin* method succeeds better than GACELA, in reproducing the sharpness of the fundamental frequencies along with the harmonics of piano notes, especially in the lower octaves, thus reproducing a sound that is more faithful to the original.

Table 4.4: ODG score values for the three compared methods, and the two gap width configurations.

Gap size	Method	ODG score ↑
Small (375 ms)	GACELA	-3.232 ± 0.232
	bin2bin	-2.892 ± 0.510
	bin2bin-MIDI	**-2.800 ± 0.491**
Medium (750 ms)	GACELA	-3.318 ± 0.202
	bin2bin	-3.039 ± 0.495
	bin2bin-MIDI	**-2.976 ± 0.456**

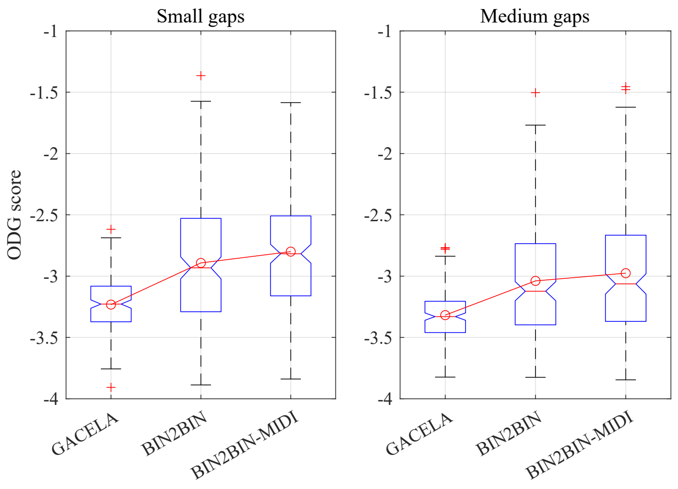


Figure 4.13: ODG score box plots, for the three compared methods, and the two gap width configurations.

Table 4.5: SSIM index values for the three compared methods, and the two gap width configurations.

Gap size	Method	SSIM index ↑
Small (375 ms)	GACELA	0.809 ± 0.035
	bin2bin	0.911 ± 0.053
	bin2bin-MIDI	**0.913 ± 0.036**
Medium (750 ms)	GACELA	0.796 ± 0.054
	bin2bin	0.904 ± 0.036
	bin2bin-MIDI	**0.905 ± 0.036**

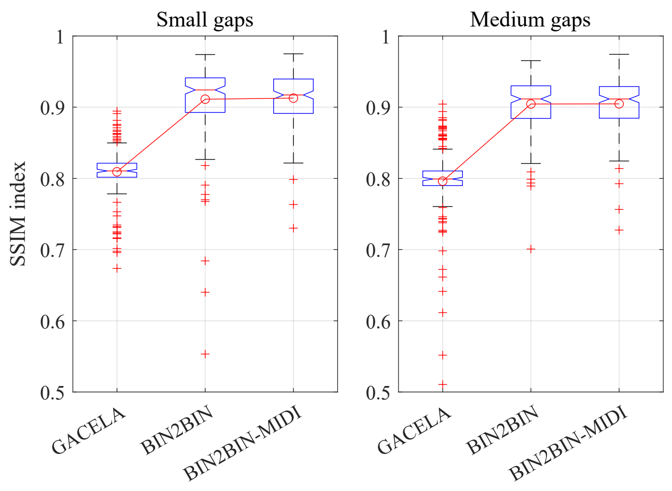


Figure 4.14: SSIM index box plots, for the three compared methods, and the two gap width configurations.

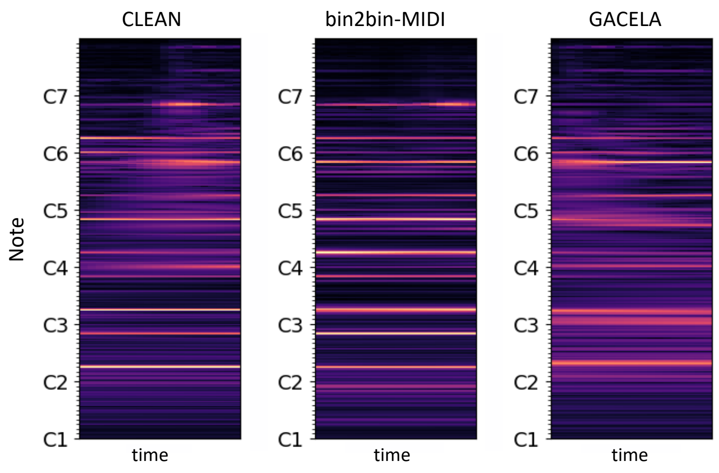


Figure 4.15: Cropped portions of the CQT spectrograms affected by a 750 ms gap. Clean reference (left), *bin2bin-MIDI* inpainting (center), GACELA inpainting (right).

4.3.7 Final remarks

In this paper, we presented an end-to-end pipeline for CQT spectrogram inpainting and music signal gap concealment. The pipeline utilizes a cGAN-based architecture, initially introduced in our previous research on speech PLC, and inspired by the widely known *pix2pix* framework.

We integrated the traditional cross-entropy loss of a PatchGAN discriminator with two additional loss functions: one correlated with the perceptual quality of speech and another representing the MSE difference between the ground truth piano-roll and the predicted piano roll, obtained through a pre-trained pitch estimation module. The evaluation shows that our framework outperforms a well-consolidated cGAN-based method, which tends to produce segments that are richer in notes but often dissonant with the surrounding context. The reader is encouraged to personally evaluate some listening examples we provide in the accompanying demo website [1].

An interesting aspect that has arisen during the development of this project is the possibility of applying the inpainting process at the piano-roll level, rather than (or in conjuction with) the spectrogram level. In fact, we believe that a lower-dimensional representation can be considerably helpful to the training of the GAN, allowing it to generate lost contexts in a more musically meaningful way, and thus addressing gaps that easily exceed a thousand ms. We plan to develop metrics and tools to evaluate this concept as a future development.

4.4 Complex-bin2bin: A Latency-Flexible Generative Neural Model for Audio Packet Loss Concealment

On this section we propose a time-frequency generative framework for PLC, aiming to address challenges posed by computational overhead and joint magnitude-phase recovery. Inspired from our previous studies on PLC through spectrogram inpainting [135,152], we name the proposed method *complex bin2bin* (also referred to as *cplx-bin2bin*), to emphasize that the restoration involves the real part as well as the imaginary part of the acoustic spectrogram, while the term *bin2bin* points to the mapping of the fundamental discrete time and frequency units (bin).

[1] `https://aircarlo.github.io/bin2bin_music_inpainting`

4.4.1 Proposed method

One of the main innovations we have introduced in this work is the possibility of performing the repair of damaged segments in “adaptive” mode. In contrast to the totality of work in the literature, in which the forward settings (such as operating context window and stride step) must be predetermined in the training phase, our model performs a training procedure under more general conditions, which allows it to be flexible and operate in different inference conditions. In this way, the proposed model is capable of repairing damaged segments regardless of their numerosity and position within a context window of varying size, from 20 ms to 1024 ms. This operating modality is designed to give the user the ability to trade-off between computational latency and reconstruction quality in real time and without the need to change the backbone model at runtime.

The diagram in Fig. 4.16 illustrates the generic operation mode of the proposed framework. Prediction of lost frames in the current context is done based on both the buffer and the context itself, the former of which contains either correctly received and previously reconstructed frames. Then, according to a binary mask that labels the state of each packet, only the actually reconstructed packets flow through to form the final sequence.

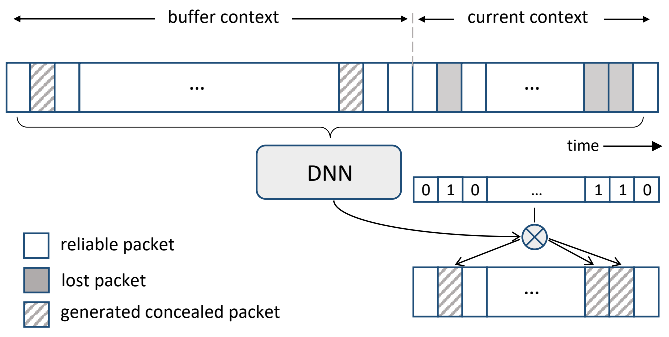


Figure 4.16: Overview of the proposed PLC mechanism, operating on a given time frame.

Network structure

An overview of our complex bin2bin architecture is presented in fig. 4.17. One of the main contributions of this paper is the adaptation of the TCN-DenseUNet

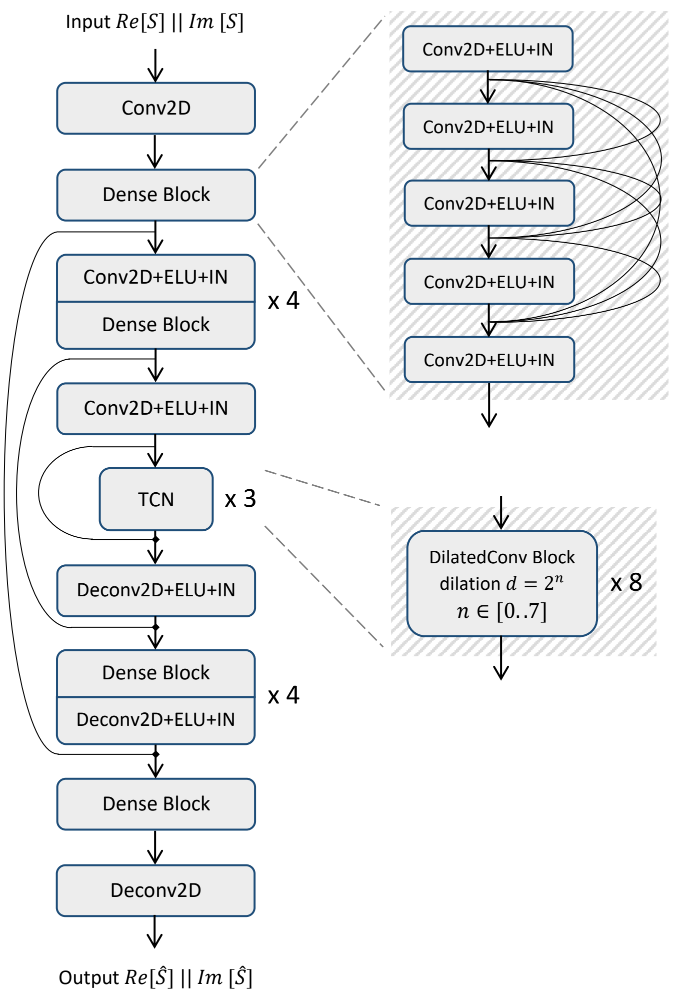


Figure 4.17: Generator network composed of the U-Net with temporal convolutional bottleneck.

architecture for the audio packet loss concealment task, through an in-depth evaluation of both generative and discriminative processes, optimized to in-paint complex-valued spectrograms gaps. We chose this architecture given its large use in various speech enhancement tasks such as speaker separation [167] and speech dereverberation [168]. The structure is built on a U-Net [121] with skip-connections and DenseNet blocks [169] at multiple frequency scales in the encoder and decoder. A temporal convolutional network (TCN) [170] at the middle section leverages long-range information by using dilated convolutions along time. Exponential Linear Unit (ELU) activations and Instance Normalizations (IN) are used after convolution and deconvolution blocks. The network takes as input a real-valued tensor with shape $C \times F \times T = 2 \times 257 \times 257$, where C is the number of channels, F the number of STFT frequenciy bins and T the number of STFT frames. The RI components of the lossy spectrogram, S are concatenated along the channel axis and fed to the network, while the output $\hat{S}$ yields the same size as input.

This work evolves from our previous researches on spectrogram inpainting using conditional GANs [135,152]. Unlike the latter, the use of complex-valued spectrograms throughout the generation process allows for multiple advantages, first of all, the ability to convert the repaired spectrogram back in time domain with a single inverse STFT operation, without the need to resort to approximate algorithms for phase estimation, which are known to be the bottleneck of methods operating on magnitude spectrograms.

Loss criteria

As widely experienced in multiple research works on speech enhancement operating in the time-frequency domain, combining multiple resolution loss criteria can have beneficial effects on several aspects, even more so in the case of generative adversarial networks, where convergence and stability are critical issues. The first to introduce such an approach being Yamamoto et al [2].

To facilitate the generation of high-resolution slices of inpainted spectrograms, we used a multiresolution STFT criterion for the generator, which is based on the evaluation of the repaired and target spectrograms, at three different resolutions in time and frequency. We defined each individual STFT loss as the weighted sum of three contributions: the spectral convergence loss ($\mathcal{L}_{sc}$), the log-STFT magnitude loss ($\mathcal{L}_{mag}$) and the phase loss ($\mathcal{L}_{pha}$):

$$\mathcal{L}_{STFT}(G) = \lambda_1 \cdot \mathcal{L}_{sc}\left(S, \hat{S}\right) + \lambda_2 \cdot \mathcal{L}_{mag}\left(S, \hat{S}\right) + \lambda_3 \cdot \mathcal{L}_{pha}\left(S, \hat{S}\right) \tag{4.11}$$

where $S \in \mathbb{C}$ and $\hat{S} \in \mathbb{C}$ denote respectively the STFT of the clean signal (s) and the repaired spectrogram. The optimal weights were chosen as $\lambda_1 = \lambda_2 = 1$ and $\lambda_3 = 0.1$, during the hyperparameter tuning process. The individual loss terms are defined as follows:

$$\mathcal{L}_{sc}\left(S, \hat{S}\right) = \frac{\sqrt{\sum_{t,f}\left(|S_{t,f}| - |\hat{S}_{t,f}|\right)^2}}{\sqrt{\sum_{t,f}|S_{t,f}|^2}} \tag{4.12}$$

$$\mathcal{L}_{mag}\left(S, \hat{S}\right) = \frac{\sum_{t,f}|\log|S_{t,f}| - \log|\hat{S}_{t,f}||}{T \cdot N} \tag{4.13}$$

$$\mathcal{L}_{pha}\left(S, \hat{S}\right) = \frac{\sum_{t,f}\left(\angle S_{t,f} - \angle \hat{S}_{t,f}\right)^2}{T \cdot N} \tag{4.14}$$

where $|\cdot|$ and $\angle$ represent the STFT magnitude and phase components respectively, while T and N denote the number of time bins and frequency bins of a frame.

As outlined in [144], $\mathcal{L}_{sc}$ highly emphasizes large spectral components, which helps especially in early phases of training, while $\mathcal{L}_{mag}$ accurately fits small amplitude variations, which tends to be more important towards the later phases of training. Finally $\mathcal{L}_{pha}$ helps in phase estimation, although most of the insight about the structure of the speech is obtained from the magnitude [171].

The multiple resolutions of $\mathcal{L}_{STFT}$ are given by the parameters sets reported in Table 4.6. All three terms are then averaged and summed to an additional contribution, $\mathcal{L}_{PMSQE}$. The latter is a perceptual-related metric operating in the time domain, defined by the combination of the MSE criterion and a differentiable implementation of the PESQ algorithm:

$$\mathcal{L}_{PMSQE} = \frac{1}{T}\sum_{t}\left(MSE + \alpha \cdot D^{(s)} + \beta \cdot D^{(a)}\right) \tag{4.15}$$

where α and β are weighting factors experimentally determined, T is the number of frames in the training batch, $D^{(s)}$ and $D^{(a)}$ are two disturbance terms inspired by the PESQ algorithm. To delve into the details of $\mathcal{L}_{PMSQE}$ see [172].

Table 4.6: Parameters used to settle multi-resolution losses. The window sizes and hop lengths in ms are derived from [2] by fitting the actual sampling rate.

Loss #	FFT size	Window size	Hop length
$\mathcal{L}_{STFT,1}$	1024	400 (25 ms)	80 (5 ms)
$\mathcal{L}_{STFT,2}$	2048	800 (50 ms)	160 (10 ms)
$\mathcal{L}_{STFT,3}$	512	160 (10 ms)	32 (2 ms)

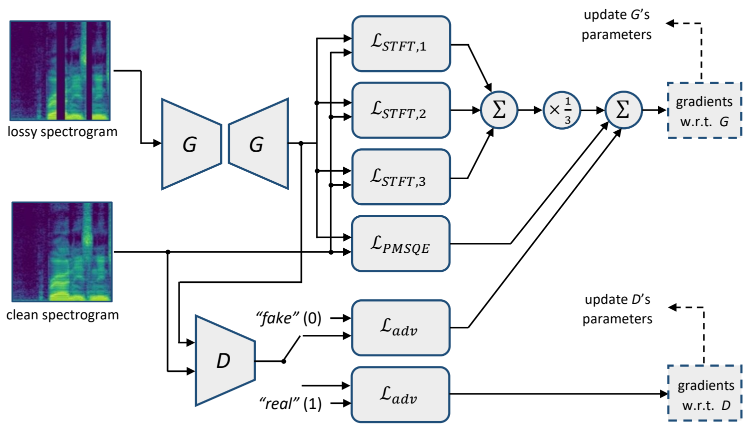


Figure 4.18: Illustration of the adversarial training strategy, assisted by the multi-resolution STFT loss and the perceptual loss.

Finally, we used the least squares loss function instead of the sigmoid cross-entropy loss function for the discriminator, as in Least squares GANs (LS-GANs) [41] (see section 2.4). The objective functions for joint conditional and least squares GAN (which will be referred as LSCGAN) can be defined as follows:

$$\min_D \mathcal{L}_{LSCGAN}(D) = \frac{1}{2}\mathbb{E}_{x,c}\left[(D(x|c) - 1)^2\right] + \frac{1}{2}\mathbb{E}_{z,c}\left[(D(G(z)|c))^2\right] \tag{4.16}$$

$$\min_G \mathcal{L}_{LSCGAN}(G) = \frac{1}{2}\mathbb{E}_{z,c}\left[(D(G(z)|c) - 1)^2\right] \tag{4.17}$$

Figure 4.18 shows the operative adversarial training scheme, used in generator and discriminator update.

4.4.2 Experimental Setup

The evaluation of the proposed method was carried out using two different criteria for simulating lost packets, according to the most common scenarios used by works dealing with the PLC problem.

First, a series of experiments were conducted using a synthetic dataset, generated from clean speech recordings taken from the VCTK corpus [145]. To simulate the occurrence of lost packets, fragments of 20 ms duration were filled with zeros, each selected randomly, regardless of the state of the preceding packets. As experienced in our previous work, the most effective strategy for training is to use over-corrupted recordings, i.e. with a higher loss rate than that used in the test phase. In addition, to ensure a stable and fast convergence of the generative network, each training sequences were corrupted with varying amount of losses, ranging from 10 % to 60 %.

The second operational scenario was to consider corrupted speech recordings with loss traces observed in actual VoIP calls. For this purpose, the dataset provided for the Microsoft PLC challenge 2022 [173] was used, which consists of clean audio clips taken from radio podcasts, and separate lost packet descriptor files, that can be coupled with clean registrations to form a potentially large dataset for our purpose. The traces of lost packets from real video calls have a slightly higher variability than those obtained by randomly simulating losses. The statistical distribution of gap width for both data sets is shown in figure 4.19.

Additionally, we applied a set of data augmentation techniques, directly on the raw waveforms, with the aim to improve model performance and generalization abilities. Augmentations include:

- Gaussian noise injection,
- Time stretch, with a rate in $[0.8, 1.25]$,
- Pitch shift, within -4 and $+4$ semitones.

Training is conducted with an early stop criterion. Specifically, to obtain a real-world evaluation of the training progress, we employ one of the evaluation metrics, described in 4.4.2. The validity of this approach is later discussed in 4.4.3 after the experimental results are described.

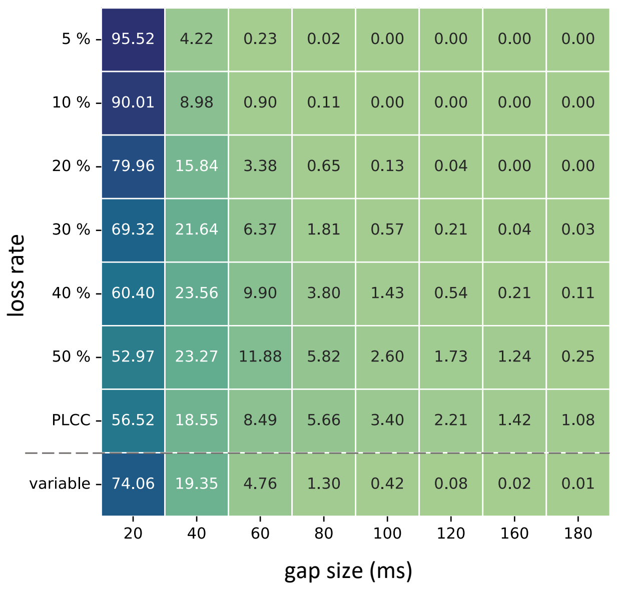


Figure 4.19: Heatmap showing the distribution of gap widths characterising the datasets. The first six rows refer to the manually injected gaps, at different rates, *PLCC* refers to the Microsoft PLC challenge dataset, while *variable* indicates the distribution obtained with varying rates in $[10\,\%, 60\,\%]$. The dashed line separates test configurations (above) from train one (bottom).

Evaluation metrics

We assessed the performance of the proposed model in terms of several criteria, some of which were also used by the models taken as comparisons. In both sets of experiments, with the synthetic dataset (VCTK) and the real-traces dataset, we calculated the values of PESQ [148], STOI [131], DNSMOS [174], PLCMOS and Word Error Rate [133].

Perceptual Evaluation of Speech Quality measure (PESQ) [148] emerged as a valid objective metric on a competition to develop metrics for speech enhancement tasks. The PESQ algorithm operates by simulating human perception of speech quality and assigning a scores ranging from -0.5 to 4.5.

Short-Time Objective Intelligibility (STOI) [131] operates on short-time segments of speech signals, typically utilizing a time-frequency representation such as the Short-Time Fourier Transform (STFT). It calculates a correlation-based measure, expressed as a percentage value, between the processed speech and the reference speech in each time-frequency bin, aiming to capture the perceptual intelligibility of the processed speech.

The latter metrics, while quick and easy to compute, may not have a very strong correlation with human ratings, and may be insufficiently exact when trying to compare two relatively similar models. They also require an aligned reference, which limits their use to scenarios where such a reference is available. Particularly in scenarios involving packet loss concealment with a jitter buffer and timescale modification, which are commonly implemented, the reference signal is typically unaligned, potentially leading to additional errors.

Non-intrusive deep neural network (DNN)-based metrics were also effectively utilized in addressing the PLC problem. One of the most prevalent metrics is the Deep Noise Suppression Mean Opinion Score (DNSMOS) [174]. Despite being originally trained for different tasks, many researchers consider it sufficiently aligned with reconstruction quality, especially in scenarios with missing segments. DNSMOS was initially conceived as a non-intrusive metric to predict scores from the ITU-T Rec. P.808 subjective evaluation, which aims to capture the overall quality of an audio clip, was later upgraded to the P.835 standard. This standard delineates three distinct scores: speech quality (SIG), background noise quality (BAK), and overall audio quality (OVRL). Authors stated that the DNSMOS metric exhibits a high correlation with human ratings, showing a Pearson's Correlation Coefficient (PCC) of 0.94 for SIG and 0.98 for BAK and OVRL.

PLCMOS [133] is a newly implemented DNN-based metric formulated by Microsoft researchers as part of an effort to advance research on Packet Loss Concealment. The scoring system employs a neural network trained to predict the ratings that human evaluators would assign to an audio file. Unlike the previously described DNSMOS, the PLCMOS model is trained using audio degraded by lossy transmissions, incorporating real packet loss traces observed in VoIP calls, and subsequently restored using various PLC algorithms. As a fully non-intrusive method, PLCMOS does not necessitate a reference signal. It has gained significant popularity as a means of comparing different PLC algorithms in recent times.

Word Error Rate (WER) is the primary accuracy metric used to evaluate Automatic Speech Recognition (ASR) systems, so it plays an important role in judging the correct gap reconstruction. Obviously, the impact of small and sparse gaps is significantly smaller than bursts of close gaps can have, so we expect different and non-comparable WER values between the two datasets considered. To calculate WER, we adopted the pre-trained ASR Whisper framework [175]. Specifically, the `medium.en` model was chosen since it is adequately accurate and lightweight at the same time, to allow feasible evaluations. The ASR model first alignes the reference and the recognized transcription by minimizing a proper distance (e.g. the *Levenshstein* or *edit* distance) between the two texts. After this, it counts the number of insertions (I), substitutions (S) and deletion (D) word errors and outputs the proportion of the total number of errors over the number of words (N) in the reference:

$$WER = \frac{I + S + D}{N} \cdot 100\,\% \tag{4.18}$$

Comparative methods

The baseline systems used for evaluation of VCTK data include two strictly causal solutions, DNN and CRN, two methods designed for offline use, SEGAN and Wave-U-Net, and a flexible solution, TFGAN-PLC, that allows two latency options, 20 ms and 160 ms, but still requires the models to be trained separately.

DNN [176] is a deep approach to predicting lost speech frames by resorting to the FFT features of previous correctly received frames. Specifically, two DNNs with three hidden layers and 2048 neurons each are employed to separately predict the magnitude and phase of the candidate frame. CRN [177] is a convolutional encoder-decoder architecture with LSTMs which has achieved excellent results in speech enhancement with magnitude-only mapping. The SEGAN-based speech enhancement approach [124] works end-to-end with the raw audio signals and reconstructs the lost frames directly in the time domain. Unlike the original SEGAN paper, a reduced configuration is used for the PLC task, with fewer output channels and shorter time frames. Wave-U-Net [178] is an application of the 1D convolutional U-Net architecture, originally designed to perform end-to-end speech enhancement, for the PLC task. TFGAN-PLC [100] is an end-to-end PLC approach adopting a time-frequency hybrid generative adversarial network with the integration of time-domain and frequency-domain discriminators.

Finally, real-world traces from MS dataset were tested with LPCNet [179], an autoregressive neural vocoder that improves on WaveRNN [180] using linear prediction. It allows causal operation, reconstructing 20 ms lost packets as they occur, or by looking at 5 ms lookahead, which will be considered having 25 ms stride. In addition, a variant of LPCNet called LPCNet-dc is tested, in which the authors state a special handling for DC offsets.

4.4.3 Results

Comparative results are provided in the following for all the aforementioned methods, according to PESQ, STOI, DNSMOS and PLCMOS scores. Table 4.8 reports PESQ scores at different loss rates for the reference clean speech, the lossy speech (with zero-fill where packets are lost) and the neural network approaches that accept a stride of 20 ms or 160 ms. In addition we provide results for the cplx-bin2bin network with a 1024 ms (being the only one that can accept such a long stride). As can be seen, the proposed method outperforms all the comparative methods in terms of PESQ. With very high loss rates (40 % and 50 %) there is no data for the comparative methods, but we tested our method showing that an improvement in terms of PESQ can be still achieved with respect to the zero-fill lossy speech. Please note that according to PESQ the clean audio gets a slightly lower result than the ideal score (5.0).

Evaluations are also conducted on the same data using the STOI index, as shown in Table 4.10. In this case, the TFGAN-PLC scores better with low loss rates (5 % with stride 160 ms, and 5-10 % with stride 20 ms), but the proposed method scores almost identically and scores better with higher loss rates, making it more suitable to heavy loss scenarios.

In addition to PESQ and STOI we also evaluated the PLCMOS and DNSMOS scores for the proposed method, which are shown in Table 4.9 and 4.12. In all cases, the proposed algorithm is able to increase both the PLCMOS and the DNSMOS scores of the lossy speech. Specifically, the PLCMOS score is increased from a minimum of 39% (loss rate 50%, stride 20 ms) to a maximum of 86% (loss rate 20%, stride 1024 ms).

Overall, these results show the effectiveness of the proposed approach. To keep the results in context, on table 4.7 we also provide an insight on the number of trainable parameters for each of the compared solutions.

For the sake of completeness, we also conducted tests with the cplx-bin2bin network with varying length of the stride, to assess the PESQ, STOI, PLCMOS and DNSMOS performance. These are shown in figures 4.21, 4.22, 4.23 and 4.24. All scores follow a similar pattern, i.e. that with shorter strides the concealment performance increases. Specifically, the performance rises quickly in terms of PESQ and PLCMOS when the stride increases from 20 ms to values between 300-400 ms. Then a plateau is reached, meaning that the added con-

Table 4.7: List of trainable parameters in each architecture. For the adversarial-based models, only the generator was considered.

Model	trainable params (M)
DNN	24.37
CRN	17.50
SEGAN	20.49
Wave-U-Net	10.13
TFGAN-PLC	1.85
LPCNet	5.90
cplx-bin2bin	3.14

text between 400 and 1000 ms is of little help to increase the reconstruction quality. It is interesting to note that with high loss rates, an increase in the stride can highly improve the STOI, the PLCMOS and the DNSMOS.

Another method to evaluate the proposed method and its ability to restore the original speech signal is to assess the WER on the reconstructed signal. Table 4.11 shows that with a stride of 20 ms the network can only slightly increase the performance[2]. However, with a larger stride the network benefits from the added context and is capable of reducing the WER up to a half, with loss rates as high as 50 %.

To visually assess the qualitative results of the proposed model, we report on fig. 4.20 (top-right) the spectrogram of a 1024 ms long segment (equivalent to nearly 50 packets) corrupted by losses of varying size.
The example is taken from a real validation sample. The regions affected by losses are not sharply demarcated because the STFT operation introduces an inherent cross-fade that smooth the transition in time. On fig. 4.20 (top-left) is the spectrogram of the same clean segment, while the image below shows the output of the reconstruction network. Although this visual evaluation does not allow for quantifying the presence of other types of distortions potentially introduced by the network, it is observed that the differences between the reconstructed and clean spectrogram are imperceptible, and the typical formant frequencies of the speech signal are reconstructed seamlessly.

[2]With loss rates 5-10 % the WER of the reconstructed signal is slightly higher than the one computed on the lossy speech, which may imply that pre-trained ASR Whisper model used in this work is robust to rare drops of small audio segments.

Table 4.8: PESQ scores for complex-bin2bin and the comparative DNN solutions, evaluated at different loss rates and stride.

	stride	Loss rate					
Model	(ms)	5 %	10 %	20 %	30 %	40 %	50 %
Clean	-	4.64	4.64	4.64	4.64	4.64	4.64
Lossy (zero-fill)	-	2.61	1.84	1.33	1.18	1.10	1.03
DNN	20	2.73	1.89	1.54	1.39	-	-
CRNN	20	2.79	1.93	1.66	1.48	-	-
TFGAN-PLC	20	2.94	2.16	1.87	1.63	-	-
cplx-bin2bin	20	**3.30**	**2.64**	**1.99**	**1.75**	1.51	1.47
SEGAN	160	2.76	1.95	1.63	1.49	-	-
Wave UNet	160	2.87	2.11	1.76	1.54	-	-
TFGAN-PLC	160	3.24	2.59	2.14	1.86	-	-
cplx-bin2bin	160	**3.73**	**3.23**	**2.65**	**2.26**	1.94	1.68
cplx-bin2bin	1024	3.76	3.41	2.89	2.49	2.15	1.87

Table 4.9: PLCMOS scores for complex-bin2bin evaluated at different loss rates and stride.

	stride	Loss rate					
Model	(ms)	5 %	10 %	20 %	30 %	40 %	50 %
Clean	-	4.274	4.274	4.274	4.274	4.274	4.274
Lossy	-	4.025	3.586	2.721	2.179	1.822	1.547
cplx-bin2bin	20	4.179	3.982	3.567	3.163	2.853	2.613
cplx-bin2bin	160	4.229	4.131	3.939	3.728	3.477	3.168
cplx-bin2bin	1024	4.222	4.174	4.064	3.927	3.742	3.518

Table 4.10: STOI scores for complex-bin2bin and the comparative DNN solutions, evaluated at different loss rates and stride.

	stride	Loss rate					
Model	(ms)	5 %	10 %	20 %	30 %	40 %	50 %
Clean	-	100	100	100	100	100	100
Lossy	-	95.45	91.20	83.94	75.75	68.87	63.99
DNN	20	95.73	92.57	85.36	78.84	-	-
CRNN	20	96.25	92.77	86.11	79.24	-	-
TFGAN-PLC	20	**97.69**	**94.68**	88.93	83.72	-	-
cplx-bin2bin	20	97.15	94.63	**89.72**	**84.59**	79.92	75.39
SEGAN	160	96.82	94.20	87.03	81.37	-	-
Wave UNet	160	97.15	94.23	87.68	82.17	-	-
TFGAN-PLC	160	**98.45**	95.82	90.11	86.39	-	-
cplx-bin2bin	160	97.74	**96.00**	**93.05**	**89.99**	86.30	81.71
cplx-bin2bin	1024	97.81	96.78	94.45	91.65	88.41	84.25

Table 4.11: Word Error Rates obtained on the synthetic dataset, with different loss rates.

	stride	Loss rate					
Model	(ms)	5 %	10 %	20 %	30 %	40 %	50 %
Clean		1.76	1.76	1.76	1.76	1.76	1.76
Lossy		1.95	2.19	2.94	3.86	6.44	12.98
cplx-bin2bin	20	2.10	2.27	2.82	3.74	5.51	12.01
cplx-bin2bin	160	1.95	2.04	2.53	3.02	3.89	6.84
cplx-bin2bin	1024	1.89	1.99	2.28	3.0	3.46	6.30

Table 4.12: DNSMOS scores for complex-bin2bin evaluated at different loss rates and stride.

	stride	Loss rate																	
		5 %			10 %			20 %			30 %			40 %			50 %		
Model	(ms)	ovrl	sig	bak	ovrl	sig	bak	ovrl	sig	bak	ovrl	sig	bak	ovrl	sig	bak	ovrl	sig	bak
Clean		3.18	3.46	3.99	3.18	3.46	3.99	3.18	3.46	3.99	3.18	3.46	3.99	3.18	3.46	3.99	3.18	3.46	3.99
Lossy		3.12	3.40	3.97	3.00	3.28	3.92	2.64	2.91	3.75	2.19	2.43	3.55	1.79	1.98	3.34	1.51	1.64	3.20
cplx-bin2bin	20	3.14	3.45	4.00	3.06	3.38	3.96	2.90	3.23	3.89	2.71	3.05	3.81	2.49	2.83	3.69	2.24	2.58	3.56
cplx-bin2bin	160	3.17	3.47	4.01	3.12	3.43	3.99	3.04	3.35	3.96	2.95	3.27	3.93	2.86	3.19	3.90	2.76	3.08	3.86
cplx-bin2bin	1024	3.12	3.43	3.99	3.10	3.41	3.99	3.05	3.36	3.98	3.00	3.31	3.97	2.93	3.25	3.95	2.86	3.83	3.93

Table 4.13: Overall metrics for testing cplx-bin2bin and LPCNet on MS-PLC real traces dataset.

	stride ms	PESQ	STOI	PLCMOS	DNSMOS (ovrl)	DNSMOS (sig)	DNSMOS (bak)	WER
Clean (reference)		4.56	100.00	4.33	3.24	3.58	3.94	9.85 %
Lossy (zero-fill)		2.19	83.91	2.68	2.56	2.77	3.56	20.07 %
LPCNet causal	20	2.70	90.82	3.58	3.11	3.47	3.85	16.93 %
LPCNet-dc causal	20	2.71	90.95	3.54	3.13	3.47	3.91	17.39 %
LPCNet noncausal	25	2.76	91.36	3.62	3.10	3.45	3.87	17.23 %
LPCNet-dc noncausal	25	2.77	91.49	3.59	3.13	3.47	3.91	17.02 %
cplx-bin2bin	20	**3.16**	**92.56**	**3.95**	**3.15**	**3.48**	**3.92**	**15.66 %**

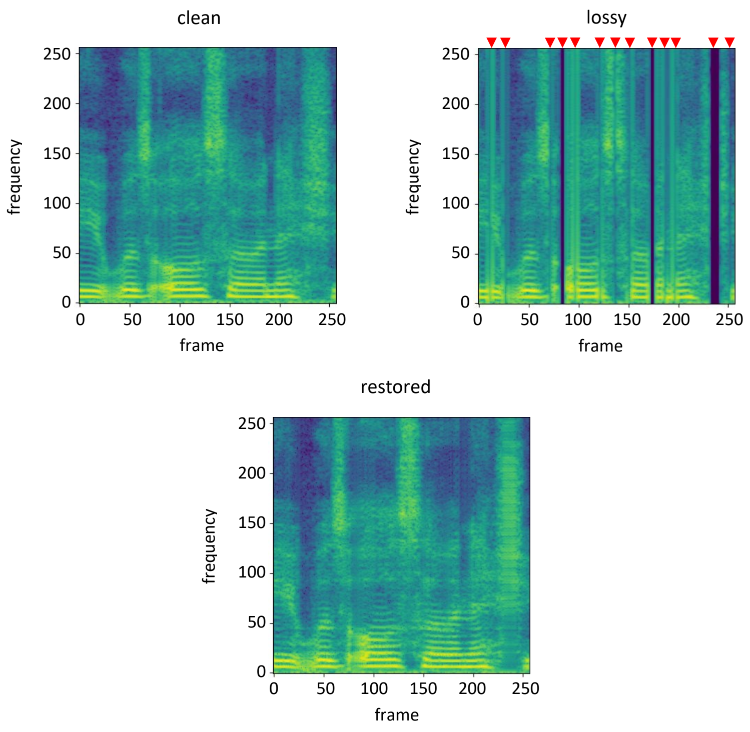


Figure 4.20: Magnitude spectrograms (in dB) of an example reconstruction. Top-left: target signal, top-right: lossy signal with red markers indicating gap displacements, bottom: reconstruction by the complex bin2bin network. The axes of the plots indicate the frequency bin and the frame index.

Choice of the early stopping metric

In Section 4.4.2 we described the validation method and the early stop criterion. These are based on one of the evaluation metrics, the PLCMOS, in order to stop the training according to a real-world assessment of the audio signal. The underlying assumption is that the choice of PLCMOS is not arbitrary and correlates well with any other of the metrics, including the PESQ, which is one of the addends in the training loss, and could be well used in its spite.

From the experimental data, we report scatter plots, in Figure 4.25 that shows the PLCMOS score of each test sample as y-coordinate and each of the other three metrics (PESQ, STOI, DNSMOS OVRL) in the x-coordinate. As can be seen the PLCMOS is positively correlated (PCC > 0) to the other indices used for evaluations. The positive association between the metrics motivates

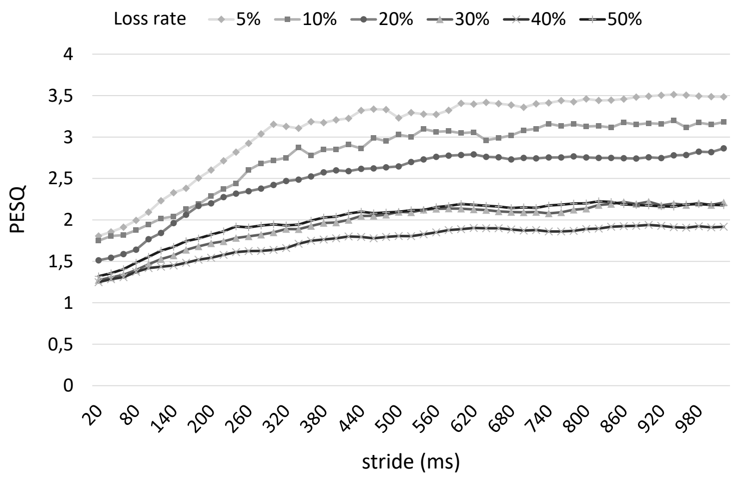


Figure 4.21: Trend of PESQ values, for recovering a selected file, by varying the stride and loss rate, through the entire admissible ranges.

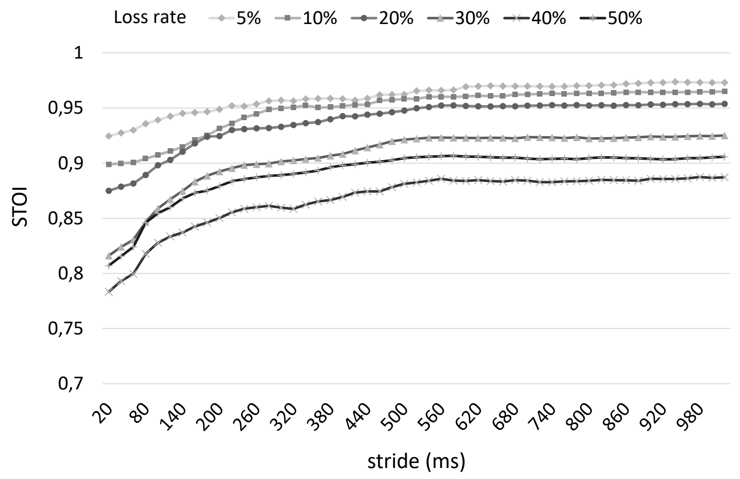


Figure 4.22: Trend of STOI values, for recovering a selected file, by varying the stride and loss rate, through the entire admissible ranges.

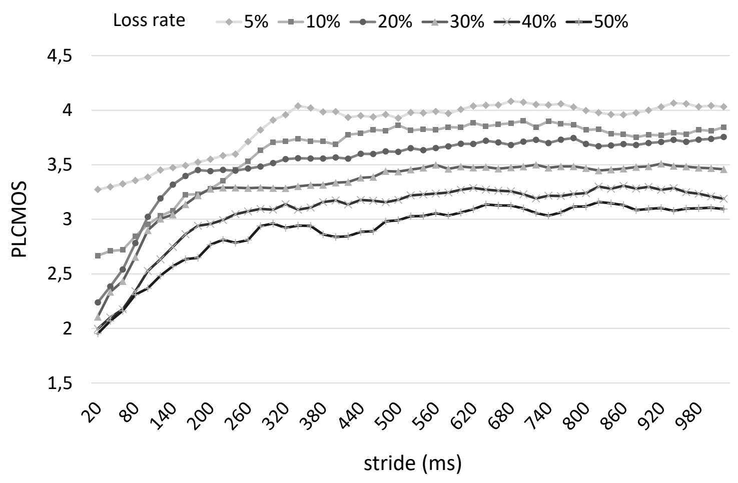


Figure 4.23: Trend of PLCMOS values, for recovering a selected file, by varying the stride and loss rate, through the entire admissible ranges.

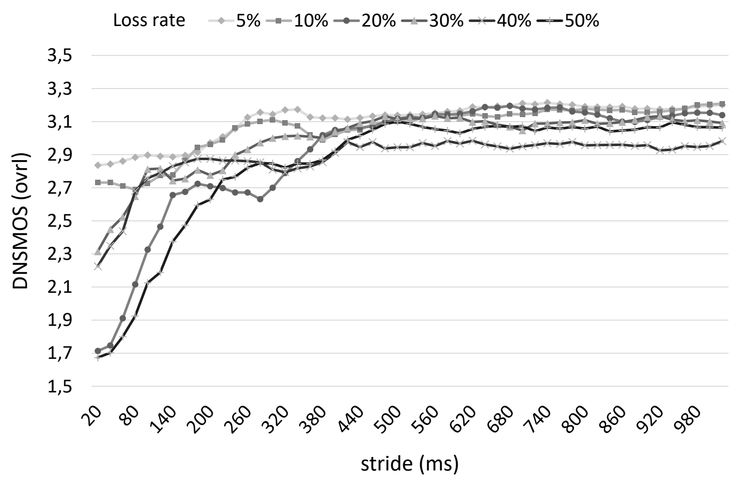


Figure 4.24: Trend of DNSMOS ovrl values, for recovering a selected file, by varying the stride and loss rate, through the entire admissible ranges.

the use of the PLCMOS as a criterion for early stopping, alternative to the losses used during training, to provide a more robust training focused for PLC.

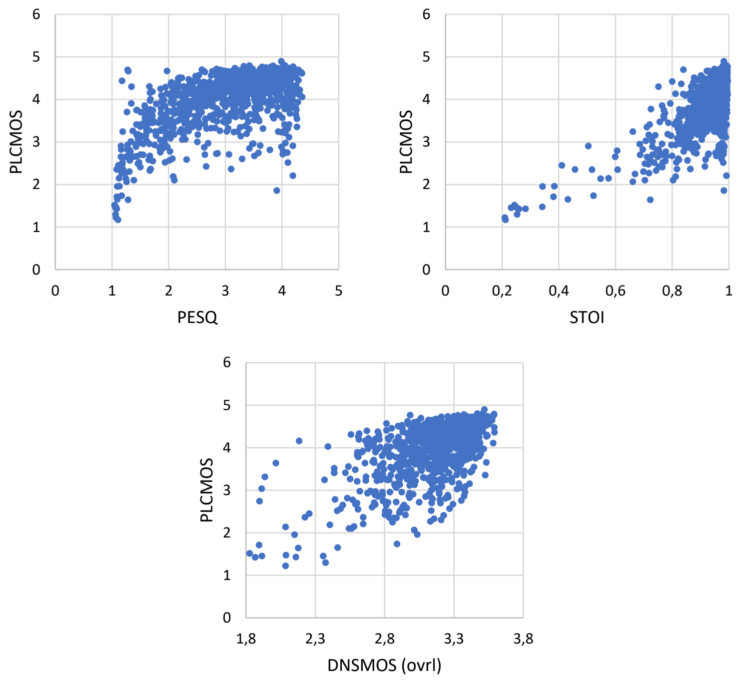


Figure 4.25: Scatter plots showing the correlation between PLCMOS and PESQ (top-left), PLCMOS and STOI (top-right), PLCMOS and DNSMOS (bottom). Data points are extracted from the tests using cplx-bin2bin with 1024 ms stride.

4.4.4 Final remarks

In this work we proposed a novel approach for Audio Packet Loss Concealment that provides flexible handling of latency and is comparable or superior to state of the art DNN solutions. Its flexibility lies in the ability of recovering spectrograms without prior knowledge on the segment to be restored, therefore, it can be employed to repair the latest audio packet, as well as any other in the input temporal context. The approach is based on a generative bin2bin network that handles complex spectrograms, thus restoring the phase and magnitude information jointly. The system also employs an audio quality metric, PESQ, implemented as a differentiable DSP algorithm, in order to use it as a loss function during the training.

Experiments were conducted on different datasets. On voice recordings with randomly inserted gaps, at rates ranging from 5 % to 50 %, the proposed model outperformed five recent alternative approaches, based on neural networks, with improvements up to 22.2 % (20 ms latency) and 23.8 % (160 ms latency) for PESQ, and improvements of 1.04 % (20 ms latency) and 4.17 % (160 ms latency) for STOI. Furthermore, experiments conducted on corrupted signals with actually lost packet distributions over communications networks, the complex-bin2bin model showed improvements of 14.08 % (PESQ), 1.17 % (STOI), 26.2 % (PLCMOS), 0.64 % (DNSMOS ovrl), 0.29 % (DNSMOS sig), 0.26 % (DNSMOS bak), and succeeding in lowering the word error rate perceived by an automatic recognition system by 1.27 percentage points.

Despite the excellent results obtained by our novel PLC approach, two aspects leave room for future investigations. First is the need to reduce the computational footprint of the model, which is currently the second lightest among those examined. This can be achieved by a proper data distillation technique applied to the TCN-DenseUNet. Second, to improve the performance of the model in terms of word error rate, especially at minimum latency where results have been shown to be negative, we plan to incorporate an additional loss criterion that minimizes the expected WER.

Chapter 5

Other contributions

5.1 A graph-based neural approach to linear sum assignment problems

5.1.1 Introduction

Linear assignment [181] is a fundamental problem in operations research; it aims at assigning the elements of one finite set to the elements of another set. This is done under the condition of one-to-one correspondence, so that the resulting assignment satisfies some optimality criterion, such as minimum cost or, in a dual form, maximum profit. When the sum of the costs is the objective to be minimized, the problem is called a Linear Sum Assignment Problem (LSAP). This type of problem is found in many image processing applications such as point matching [182], handwritten character and mathematical expression recognition [183], multiple object tracking (MOT) [184], and object segmentation [185]. In wireless communication systems, it plays an important role in tasks such as mode selection for device-to-device communications [186], resource allocation in MIMO systems [187] and unlicensed channel management for LTE systems [188]. On audio processing field, the permutation ambiguity problem of multiple source separation [189] is closely related to LSAP, as well as to end-to-end neural diarization [190] and, in general, for metrics computation, such as diarization error rate [191] or permutation-invariant word error rate [192] for automatic speech recognition.

Related works

A well-established method for linear assignments is the Hungarian algorithm [193], developed by H. Kuhn in 1955 and revised by J. Munkres in 1957 [194], which succeeds in obtaining the optimal solution without a greedy search. However, it does not scale well with the size of the problem N, since its computational complexity is $\mathcal{O}(N^3)$.

Bertsekas et al. proposed the auction algorithm [195] to solve LSAP problems, so called because it mimics the actual auction process. This method comprises two steps: a bidding phase and an assignment phase. Later, an extension based on a scaling strategy, further enhanced the performance of the auction algorithm. While being close to the ideal solution, it still has a high computational complexity, which affects its outcomes for linear assignment issues.

A variant of the Hungarian algorithm that uses the shortest alternating paths to supplement primal solutions has been proposed by Jonker and Volgenant. [196]. It begins with an initialization phase based on a naive auction algorithm.

Several algorithms, including the interior point method [197, 198] and dual forest method [199], solve linear assignment problems using general linear programming techniques. Ramakrishnan et al. [198] modified the Karmarkar interior-point method [200] and created an approximate dual projective algorithm.

Additionally, a lot of heuristic methods based on greedy tactics try to find quick approximate solutions. For the generalized assignment problems, Trick et al. suggest a greedy heuristic technique [201] and demonstrate that adding some randomization to the greedy approach can help find better solutions. Another similar approach is the greedy randomized adaptive search procedure (GRASP) [202]. Naiem et al. develop the Deep Greedy Switching (DGS) algorithm [203, 204], which starts with a random initial guess and attempts to discover a better solution by searching inside a well defined neighborhood. Although these methods are much faster than the previous heuristics having polynomial complexity, they often get stuck when the value of the objective function reaches a local optimum. Moreover, because the gradients of these heuristic solutions are somewhat difficult to describe, they cannot be directly included in learning frameworks.

Recently, deep neural networks (DNNs) have achieved promising results on mathematical optimization problems, with practical applications such as wireless resource management allocation [205], link scheduling optimization [206] or interference management [207]. Several data-driven algorithms have also been proposed for linear assignment problems. In [208], the assignment task was converted into an equivalent continuous linear programming problem solved by a recurrent neural network. Recent works address smaller sub-tasks by breaking the $N \times N$ assignment problem into N multi classification tasks, using stacked perceptrons layers [209], bidirectional long short-term memory neural network (Bi-LSTM) [210] and bidirectional recurrent neural networks (Bi-RNN) [211].

This work is an extension of the one presented in [212], where we propose a graph-based LSAP description and a DNN technique built on graph learning is used to address the assignment challenge. This extended version of the article includes a case study of a real application on smart meter scheduling and an in-depth analysis of GNN network performance as the number of hidden layers and batch-size used in training vary. As a comparison, we build upon the framework proposed by Lee et al. [209], where LSAP of different dimensionality N are decomposed into N independent sub-assignment problems, and two types of DNNs, a feed-forward MultiLayer Perceptron (MLP) and a Convolutional Neural Network (CNN) are applied to address the sub-assignments as independent classification tasks.

In this work we show that using an MLP is the worst approach because it forces the problem to be treated as independent assignments. A better modeling strategy is arguably to process the entire cost matrix as input. However, for example, in the CNN approach as convolutional kernels cover a narrow receptive field, it is required to stack several layers to cover the entire cost matrix as the size of the problem increases. Moreover, CNNs have finite receptive field, thus this solution cannot scale to cover arbitrarily large cost matrices. In contrast, the proposed graph representation of the cost matrix together with graph-based DNN techniques, allow for an efficient information spread, exploiting relations between all agent-job pairs, even for networks with limited depth, with obvious advantages in terms of scalability.

This work originated from our contribution to the Second International Conference on Applied Intelligence and Informatics, AII 2022 [212], and was later extended for publication on the International Journal of Neural Systems, in January 2024 [213].

5.1.2 Problem formulation

Typically, in the literature, assignment problems are described by resorting to a toy-example in which N *jobs* must be assigned to as many different *workers* or *agents*, taking into account a quantity associated with each assignment (e.g., a cost to be minimized or a profit to be maximized). This of course can be extended to any other context in which two sets of the same cardinality are involved and all elements must be paired one by one.

Given two finite sets $I = \{1, 2, ..., N\}$ and $J = \{1, 2, ..., N\}$, let assume that the assignment of element $i \in I$ to element $j \in J$ incurs a cost $c_{i,j}$. The problem has a straightforward integer programming formulation, in which the decision variable $x_{i,j}$ is equal to 1 or 0 to indicate a feasible or infeasible assignment, respectively; therefore, it can be expressed as the minimization of the following objective function:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c_{i,j} x_{i,j} \tag{5.1}$$

subject to the constraints:

$$\sum_{i=1}^{N} x_{i,j} = 1 \qquad j = 1..N \tag{5.2}$$

$$\sum_{j=1}^{N} x_{i,j} = 1 \qquad i = 1..N \tag{5.3}$$

$$x_{i,j} \in \{0, 1\} \qquad i, j = 1..N \tag{5.4}$$

Another common way of modeling assignment problems is by means of graph theory, through a complete bipartite graph, a structure where the set of vertices can be divided into two disjoint sets or classes, and the only edges connect vertices from one class to those of the other class. Let N be the problem dimensionality, the associated graph has $2N$ nodes, N of which represent agents while the rest represent jobs. The assignment costs, properly arranged in a $N \times N$ adjacency matrix, C, thus represent the weights of connections between nodes.

The resulting assignments can be also expressed as a permutation ϕ of the elements inside set I or set J, or with a permutation matrix X_ϕ, whose elements $x_{i,j} = 1$ if $j = \phi(i)$, and $x_{i,j} = 0$ if $j \neq \phi(i)$, the latter being the *Adjacency matrix* of a bipartite assignment graph (fig. 5.1).

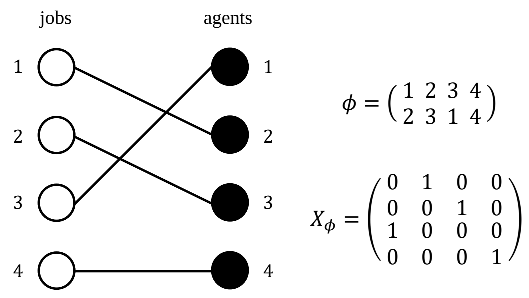

jobs
agents
1
2
3
4
1
2
3
4
$\phi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 1 & 4 \end{pmatrix}$
$X_\phi = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Figure 5.1: Different representations for the same assignment permutation.

5.1.3 Hungarian algorithm

Formally, the Hungarian algorithm involves manipulating the weights of the bipartite graph in order to find a stable, minimum-weight perfect matching [214]. However, most implementations use dynamic programming techniques by acting on cost matrix values, given the key observation that if a number is added to or subtracted from all of the entries of any one row or column of the cost matrix, then an optimal assignment for the resulting matrix is also an optimal assignment for the original cost matrix. The next page presents the pseudo-code formulation of the iterative algorithm on which most software libraries are based. When the algorithm ends, the entries of the returned matrix A containing ones indicate the optimal assignments, which ensure that all constraints are satisfied.

Algorithm 2 Hungarian Algorithm

```
**Input**: Cost matrix $C \in \mathbb{R}^{N \times N}$
**Output**: Assignment matrix $A \in \{0; 1\}^{N \times N}$
**Step 1:** Subtract the smallest element in each row from all elements in that row.
for $i \leftarrow 1$ to $N$ do:
    $m_i = min(C_{i,:})$
    for $j \leftarrow 1$ to $N$ do:
        $C_{i,j} = C_{i,j} - m_i$
**Step 2:** Subtract the smallest element in each column from all elements in that column.
for $j \leftarrow 1$ to $N$ do:
    $m_j = min(C_{:,j})$
    for $i \leftarrow 1$ to $N$ do:
        $C_{i,j} = C_{i,j} - m_j$
**Step 3:** Draw the minimum number of lines to cover all zeros in $C$.
$l \leftarrow$ minimum_lines_to_cover_zeros($C$)
if $l == N$ then:
    if $C_{i,j} == 0$ then:
        $A_{i,j} \leftarrow 1$
    else:
        $A_{i,j} \leftarrow 0$
    return $A$
else:
    **Step 4a:** Extract the submatrix $C^*$ by selecting the columns and rows not yet covered.
    $C^* = C - \{$covered_rows_and_columns$\}$
    **Step 4b:** Find the smallest element in $C^*$
    $n \leftarrow \min(C^*)$
    **Step 4c:** Subtract $n$ from each row of $C^*$,
    for $i \leftarrow 1$ to $|C^*|$ do:
        $C_{i,:} = C_{i,:} - n$
    **Step 4d:** Add $n$ to each column of $C^*$,
    for $j \leftarrow 1$ to $|C^*|$ do:
        $C_{:,j} = C_{:,j} + n$
    Go back to **Step 3**
```

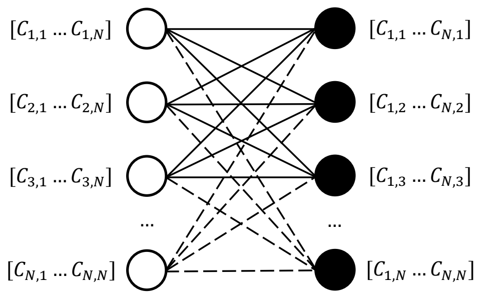
5.1.4 GNNs for combinatorial optimization problems

In the field of combinatorial optimization (CO), GNNs have already demonstrated their practical value. They have been used in various contexts, either to produce a solution directly or as an integrated component of an existing solver. Most of the previous works on such topic focused towards finding feasible, optimal or near-optimal solutions, while a smaller number tried to quantify the optimality of the proposed solution, or prove its infeasibility. Below, we briefly review the main works involving GNNs for basic CO problems. For a more extensive review, see [215].

Prates et al. [216] trained a GNN in a supervised manner to solve small-scale instances (up to 105 cities) of the Traveling Salesman Problem (TSP). A similar structure was further extended by Lemos et al. [217] for the Graph Coloring Problem. To solve the TSP, Joshi et al. [218] suggested using Residual Graph Convolutional Neural Networks [219] in a supervised manner. Instead of producing a valid TSP tour, the model provides the probability that each edge belongs to the tour. Li et al. [220] used Graph Convolutional Networks [221] on combinatorial problems easily reducible to Maximum Independent Set (MIS) problems. Li et al. [222] studied the use of GNN for Graph Matching, that is, to search for an alignment between two graphs or sub-graphs, such that a cost function is minimized. Fey et al. [223] proposed an extended architecture for the same matching problem, in the first stage of which a GNN learns a node embedding to compute a similarity score between nodes based on local neighborhoods. A GNN-based architecture known as GraphSIM is presented by Bai et al. [224] to address the challenges of graph edit distance and maximum common sub-graph problems, in an end-to-end pipeline. A similar challenge to the one addressed in this paper is presented by Nowak et al. [225] who trained a GNN in a supervised manner to predict solutions to the Quadratic Assignment Problem (QAP). They modeled the QAP instances as two adjacency matrices and used the two corresponding graphs as input to the GNN.

5.1.5 Proposed method

In the following, given the assignment problem, the cost overview has been modeled with a fully connected bipartite graph, a structure where the set of vertices can be divided into two disjoint sets or classes, and the only edges connect vertices from one class to those of the other class. Let N be the problem dimensionality, the associated graph has $2N$ nodes; N of which represent agents



$[C_{1,1} \dots C_{1,N}]$
$[C_{2,1} \dots C_{2,N}]$
$[C_{3,1} \dots C_{3,N}]$
$[C_{N,1} \dots C_{N,N}]$
$[C_{1,1} \dots C_{N,1}]$
$[C_{1,2} \dots C_{N,2}]$
$[C_{1,3} \dots C_{N,3}]$
$[C_{1,N} \dots C_{N,N}]$

Figure 5.2: Illustration of the nodes and edges definition, from the cost matrix to the corresponding bipartite graph.

while the rest represent jobs.

The input raw feature vectors of the nodes, $[\mathbf{x}_1 \dots \mathbf{x}_{2N}]$, are initialized with the cost values between source agents and receiving jobs, according to the cost matrix $C \in \mathbb{R}^{N \times N}$:

$$\mathbf{x}_i = [C_{i,1} \dots C_{i,N}] \qquad i = 1..N \quad agents \tag{5.5}$$

$$\mathbf{x}_{N+j} = [C_{1,j} \dots C_{N,j}] \qquad j = 1..N \qquad jobs \tag{5.6}$$

Conversely, all the raw attributes of the edges $\mathbf{e}_{ij}$ in the constructed graph are initialized with zero-valued vectors, hence they are not taken into account by the convolution operator.

The GNN structure is composed of K layers, through which the graph preserves its bipartite layout, while node feature vectors are updated according to the message passing (MP) operator expressed, for the generic node v and all its neighbors w, by the following equation:

$$\mathbf{x}_v^{(k)} = \frac{1}{|\mathcal{N}(v)|} \sum_{w \in \mathcal{N}(v)} MLP\left(\mathbf{x}_v^{(k-1)} | \mathbf{x}_w^{(k-1)}\right) \tag{5.7}$$

Comparing eq. 5.7 with the general structure described in section 2.3, it can be seen that messages are generated by a learnable function (MLP), a dense network whose input vector is obtained by concatenating the attribute vector of node v with that of each of its neighbors. Then, the average acts as the aggregation function of messages from neighboring nodes, collected in a set of K hops.

Finally, the feature map at last layer, $X^{(K)} = [\mathbf{x}_1^{(K)} \dots \mathbf{x}_{2N}^{(K)}]^\top$ is transformed to the actual output $Y \in \mathbb{R}^{N \times N}$ through a linear projection with learnable parameters $\mathbf{\Theta} \in \mathbb{R}^{N \times 2N}$, to obtain the estimated assignment matrix:

$$Y = \mathbf{\Theta} X^{(K)} \tag{5.8}$$

Loss fuction and evaluation metric

The proposed model solves the assignment task as $2N$ separate classifiers to jointly comply both constraints of the assignment problem formulation (eq. 5.2-5.3); at train stage, the predicted scores are obtained from the output logits Y by applying softmax operations, in both row-wise and column-wise directions:

$$\mathbf{r}_i = softmax\left(\mathbf{y}_{i,1} \dots \mathbf{y}_{i,N}\right) \tag{5.9}$$

$$\mathbf{c}_j = softmax\left(\mathbf{y}_{1,j} \dots \mathbf{y}_{N,j}\right) \tag{5.10}$$

then, given the ground truth binary assignment matrix $\hat{Y} \in \mathbb{R}^{N \times N}$, the cross-entropy loss is computed for each of the $2N$ separate classifiers, in the form of *negative-log likelihood*:

$$\mathcal{L}_r = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbf{r}_{ij} \cdot \log\left(\hat{\mathbf{y}}_{i,j}\right) \tag{5.11}$$

$$\mathcal{L}_c = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbf{c}_{ij} \cdot \log\left(\hat{\mathbf{y}}_{i,j}\right) \tag{5.12}$$

finally, the total loss $\mathcal{L} = \mathcal{L}_r + \mathcal{L}_c$ is backpropagated to update the network weights.

At inference stage, the output prediction matrix Y passes through a threshold criterion, to obtain a binary assignment map, whose rows and columns are one-hot encoded vectors. The criterion allows for “collision” avoidance (e.g. multiple jobs assigned to the same agent or multiple agents tasked with the same job); it consists of an iterative procedure, in which the assignment with the highest output value (thus the highest probability of being a correct match) is selected from the prediction matrix. Then the corresponding row and column are deleted, and the process is repeated until a single entry remains.

To benchmark our proposed approach we use accuracy as defined in [209], that is, the amount of jobs correctly matching their optimal agents, divided by N. This also lets us to compare directly with methods proposed in [209].

Network architecture

The Graph Neural Network consisted of $K = 2$ layers, which was experimentally proven to be the best performing depth for each of the problem dimensionality (N) evaluated. A higher number of layers showed the same level of accuracy, with the drawback of parameters, memory and time consumption increase, as evidenced by the ablation study of Section 5.1.6.

The MLP network used for message propagation has one input layer of size $2N$, an hidden layer with 128 neurons, ReLU activation and an output layer of size N. This configuration was applied identically for all investigated values: $N = \{2, 4, 8, 12, 16\}$. The choice of a single hidden layer for the MLP follows the network design made by the authors of [226] and [227], where the MP operator (5.7) was originally introduced.

Dataset

The policy we adopted to generate data samples follows the one implemented in the reference paper: we generated 100.000 synthetic cost matrices, drawing samples from a continuous uniform distribution: $c_{i,j} \sim U[0, 1)$, then, 80% of such matrices were used for training, while the remaining 20 % were reserved for the validation process.

When running experiments, several times with identical settings, we ensured that the chosen amount of data is sufficiently large to avoid the model to overfit. With the same criterion we generated 20.000 samples which are used for testing. In order to further reduce the dependence of the results with respect to a specific data distribution, we investigated a minor variation on train phase by generating different samples at every epoch, but did not obtain a noticeable improvement.

The ground truth decision matrix $\hat{Y}$ is obtained at runtime for each sample, using the Hungarian algorithm; specifically we used the *munkres* [228] Python package which implements the original algorithm.

Compared learning approaches

The MLP and CNN approaches we took as comparison [209] address LSAP by first decomposing it into N separate sub-assignment problems on how to assign one of N jobs to agent j. Only constraints of eq. 5.2 and eq. 5.4 are strictly guaranteed, while the constraint of eq. 5.3 is not taken in consideration at train time, hence there may exist some collisions such that one job may be assigned to different agents simultaneously. To prevent this issue, a greedy collision-avoidance rule is applied to finally state the actual assignment.

The MLP and CNN architectures developed in [209] consisted of N models in parallel; the former has four layers with 32, 64, 256, and N hidden neurons, and ReLU non-linearity, while the latter (CNN) includes five convolutional layers, each containing 32, 32, 32, 32, and N kernels of size 1×1, and an output projection map. In both cases, cross-entropy is taken as objective criterion, while Adam [74] is used as optimization algorithm.

5.1.6 Experiments and results

Models were trained up to 50 epochs with a random weight initialization for networks; the learning rate was initially set to the value of $6 \cdot 10^{-3}$, and then halved if validation loss is not improved within a patience interval of 5 epochs. Empirically, it has been found that the learning rate is halved only once within the entire training stage; an extension of the training interval does not lead to further improvements. Stochastic Gradient Descent (SGD) was used as optimization algorithm, with L_2 weight decay of $5 \cdot 10^{-4}$. Once the training is over, the model checkpoint with best validation accuracy is selected and evaluated on the test set.

Results

We report in table 5.1 the results obtained in terms of accuracy, in conjunction with the bar plot of fig. 5.3, where we compared the proposed graph approach with the other solutions described on sec. 4.4: MLP and CNN.

The proposed GNN model exhibited relative performance improvements of 0.2 % ($N = 2$), 2.2 % ($N = 8$), 18.9 % ($N = 12$) and 17.3 % ($N = 16$) if compared with the conventional CNN. Improvements rise to 1.5 % ($N = 2$), 20.8 % ($N = 8$), 32.7 % ($N = 12$) and 31.1 % ($N = 16$) if compared with the conventional MLP. Conversely, a slight worsening of accuracy has been observed for $N = 4$, however, the significant improvement achieved as N increases suggests that large LSAP problems may benefit more from the GNN approach than the CNN or MLP ones. An interesting aspect which characterizes the GNN framework is the limited amount of memory required to store network parameters.

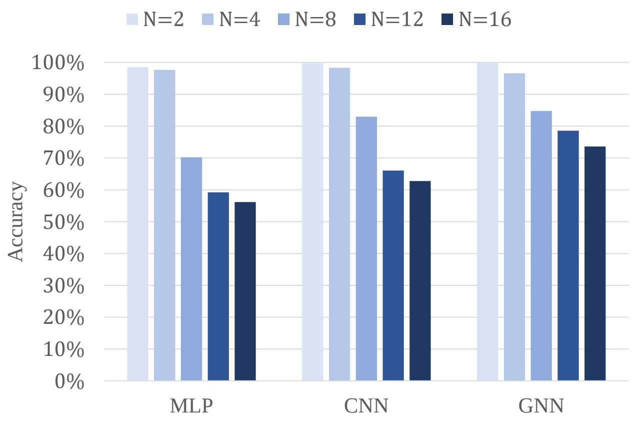


Figure 5.3: Accuracy comparison bar chart between the proposed GNN architecture and reference learning approaches.

Table 5.1: Accuracy performance comparison for the proposed GNN architecture and reference learning approaches, for different sizes N.

Size N	2	4	8	12	16
MLP [209]	0.9849	0.9763	0.7019	0.5918	0.5614
CNN [209]	0.9974	**0.9829**	0.8295	0.6605	0.6274
GNN	**0.9997**	0.9660	**0.8477**	**0.7856**	**0.7361**

Table 5.2: Learnable parameters required by different models and sizes N.

Size N	2	4	8	12	16
MLP [209]	38.8 k	81.3 k	183.1 k	317.7 k	497.4 k
CNN [209]	9.7 k	13.7 k	32.1 k	64.4 k	125.9 k
GNN	2.5 k	4.9 k	9.6 k	14.4 k	19.2 k

Table 5.3: MAC operations required by different models and sizes N.

Model	Size N				
	2	4	8	12	16
MLP [209]	38.1 k	79.8 k	180.2 k	313.3 k	491.5 k
CNN [209]	26.4 k	215.5 k	1.79 M	6.29 M	15.46 M
GNN	18.4 k	147.5 k	1.18 M	3.98 M	9.44 M

Table 5.4: Measurements of peak RAM memory usage for the execution of a single LSAP instance. Values are in kB.

Model	Size N				
	2	4	8	12	16
MLP [209]	6.1	12.2	24.6	37.3	50.2
CNN [209]	8.7	67.4	548.3	1900.4	4629.1
GNN	17.4	72.6	319.5	786.1	1480.0
Hungarian [194]	56.0	176.0	608.0	1296.0	2240.0

Table 5.5: Measurements of CPU average load time when processing a single LSAP instance. Times are in μs.

Model	Size N				
	2	4	8	12	16
MLP [209]	177.2	340.8	639.2	1023.6	1314.4
CNN [209]	221.5	454.0	934.0	1731.0	2324.4
GNN	331.1	353.8	446.7	583.5	813.3
Hungarian [194]	28.7	88.2	518.6	1719.2	4197.5

Since the MLP and the CNN approached the LSAP problem as N different classification sub-problems, the parameters are shared to a limited extent within each of these classifiers; moreover, each of them needs the full cost matrix as input, which causes a noticeable overhead. On the other hand, the message passing operator (eq. 5.7) allows for efficient reuse of the internal MLP, since it operates at node-level. Tables 5.3 and 5.2 report the exact count of learnable parameters and MAC, Multiply and ACcumulate operations involved for each of the considered architectures for different graph sizes; the values have been estimated by performing inference with a single-batch input sample. Figures on the next page help to better visualize the trends of required parameters (fig. 5.4) and MAC (fig. 5.5) as N increases.

Table 5.4 and and fig. 5.6 show the peak RAM memory use, for each of the considered methods, when solving an instance of the assignment problem. The results reveal that the CNN method has significantly more memory consumption than the MLP-based one. This is largely due to the fact that the allocation of the 2D convolutional kernels is quite costly. It is important to note, as previously mentioned, that these two methods actually use N models in parallel on sub-instances of the entire problem.

The GNN network is positioned between MLP and CNN in terms of RAM

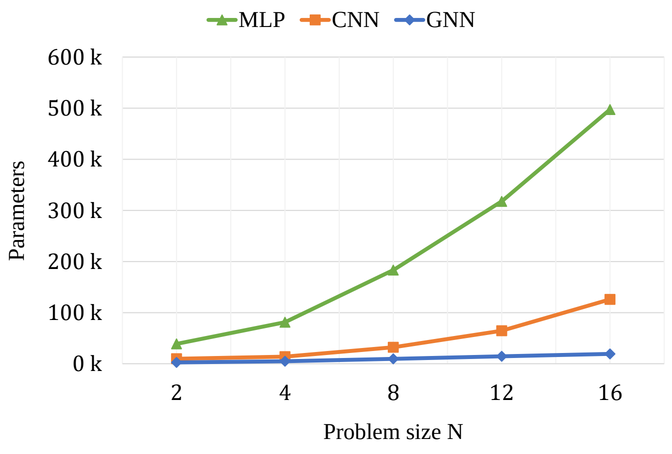


Figure 5.4: Trend in demand for memory parameters requirements.

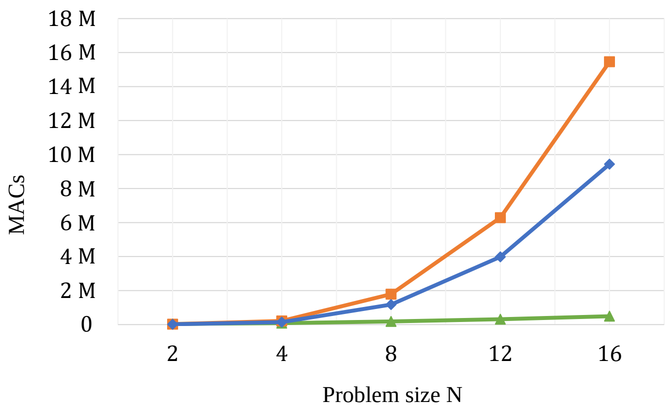


Figure 5.5: Trend in demand for MAC operations requirements.

usage, mainly due to the locality of the graph-convolutional operator and the fact it does not require N models in parallel. As we can see it is always less demanding with respect to the Hungarian algorithm, for all addressed case studies (different N values).

The peak RAM occupation allows us to estimate the maximum memory footprint of a specific solver, and hence the hardware minimum requirements; in order to get a better insight into resource consumption (it is almost always possible to trade-off memory occupation for execution time) we perform also, in parallel, a time-profile analysis, reporting on tab. 5.5 and fig. 5.7 an estimation of mean execution time, obtained over 10.000 runs, between the proposed approach, the comparison ones, and the Hungarian algorithm implemented in plain Python language [228]. We can note a cubic trend in time complexity, which makes the Hungarian algorithm impracticable to apply on resource-constrained devices, even for small dimensions.

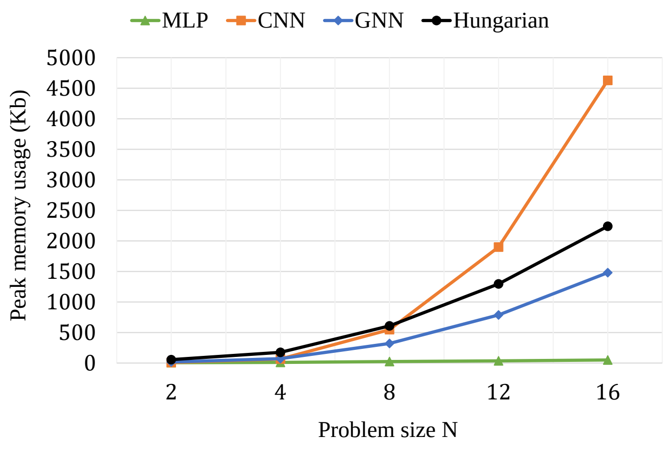


Figure 5.6: Peak RAM memory usage for the different algorithms in exam.

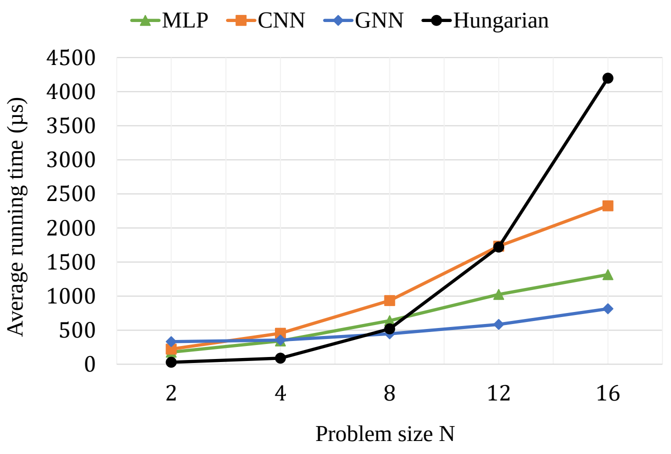


Figure 5.7: CPU average running times for the different algorithms in exam.

Looking at figure 5.5 we observe that the MAC operations required by the MLP are significantly lower than those required by the GNN, however figure 5.7 seems to indicate an opposite trend. As before, this behavior is because execution times do not depend exclusively on the number of raw operations, but are also very sensitive to other time-greedy operations such as read/write memory accesses. Indeed, we expect that the GNN, making heavy use of parameter sharing, can be significantly fast even if it experiences a high load in terms of MACs.

Figure 5.8 further emphasizes the trade-off between effectiveness and efficiency introduced by the graph-based approach, compared with the Hungarian algorithm.

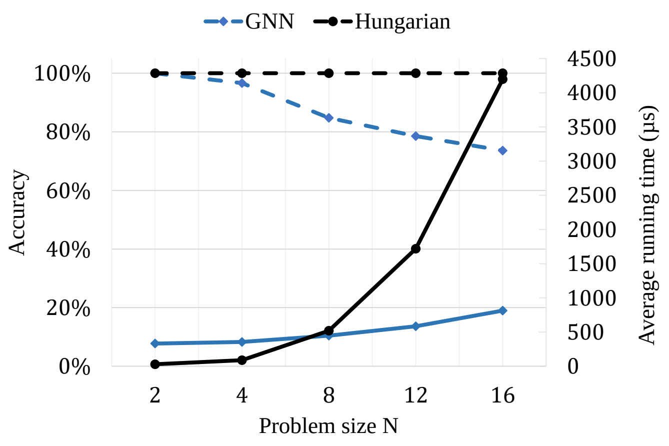


Figure 5.8: Inference time (solid lines) versus accuracy (dashed lines).

Finally, figure 5.9 shows the comparison between the accuracy obtained in terms of correct assignments and the cost accuracy. The latter quantifies the deviation of the obtained assignment cost from the optimal one. As already mentioned, the GNN is a sub-optimal solver, however, an interesting aspect emerges from this graph: although the number of correct assignments produced by the GNN is significantly lower than the optimal solution, the error in terms of total cost is quite small. This indicates that the GNN is very efficient in determining those assignments that have a major impact on minimizing total cost, while errors are concentrated on the remaining assignments which have little impact on total cost. To further confirm this behavior, we performed additional experiments with $N = 24$ and $N = 32$, although previous analyses stop at $N = 16$ to comply with the configuration used in the comparison methods, MLP and CNN.

All experiments were conducted on an Ubuntu 16.04 machine, with six Intel(R) Core(TM) i7-6850K CPUs @ 3.60GHz and 32GB RAM; network models were developed in Python language with PyTorch and PyTorch geometric [73] framework libraries.

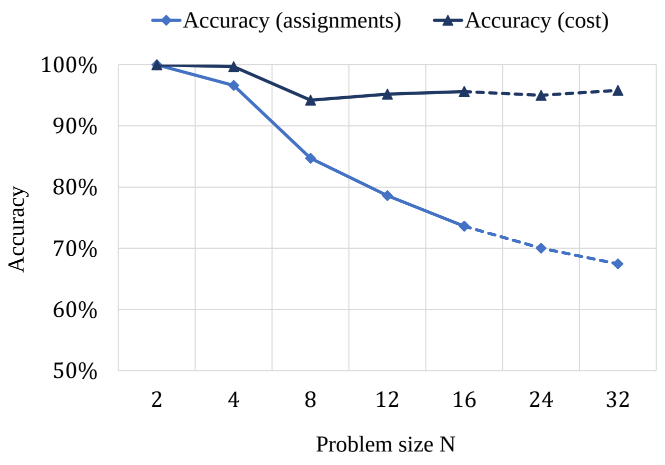


Figure 5.9: Trends in accuracy of exact assignments and deviation from the minimum attainable cost.

Ablation studies

For ablation studies on the proposed model, we conducted experiments with different parameters, including batch size (BS) and number of GNN hidden layers (K), to explore how these factors may affect the overall accuracy.

Figure 5.10 demonstrates that an high batch value decelerates the convergence in terms of total required epochs. On the other hand, contrary to what reported in [209], a low batch size did not lead to unstable convergence behavior. In fact we were able to achieve the best results in the least number of epochs, using the batch value of 1.

Figure 5.11 shows the results in terms of accuracy and inference time, achieved for several values of GNN hidden layers: $K = \{2, 3, 4, 5\}$.

The lack of improvement in accuracy, as depth increases, is due to a phenomenon, known in the GNN literature as *over-smoothing* [229–231], according to which node features tend to converge to the same vector and become nearly indistinguishable as the result of applying multiple graph convolutional layers.

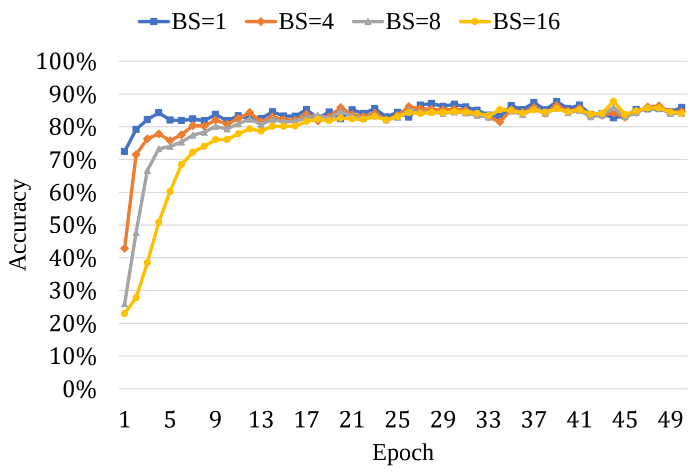


Figure 5.10: Validation accuracy at different training epochs, for different batch size values.

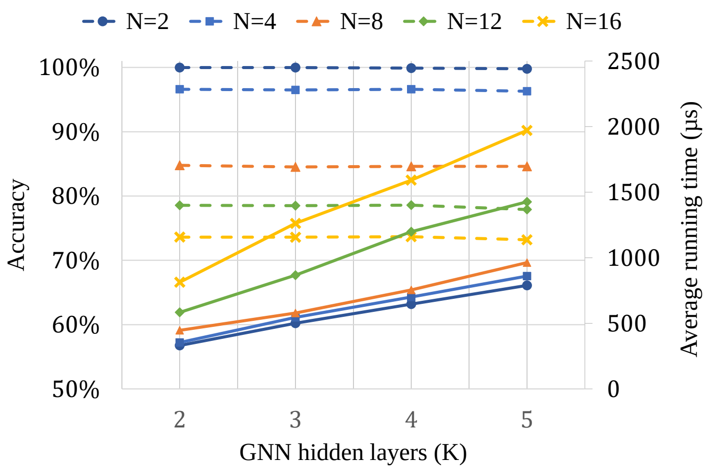


Figure 5.11: Comparison plot, showing the relationship between GNN network depth (K), accuracy level (dashed lines) and average inference time (solid lines).

5.1.7 Scheduling of smart meter data access in electricity distribution grids

Nowadays, smart meters play a crucial role in the operation of modern power distribution networks. Although their functionality is mainly targeted for billing purposes, they have the potential to enable smart grid capabilities, such as forecasting of household consumption, photo-voltaic load matching and detection of outages and flickers in low voltage (LV) grids [232].

Measurements of active power, current or voltage data are made at specific time intervals and rely on bidirectional communication to transmit information to a control center, utility or retail company.

These agents are coordinated through a centralized head-end system [233]. Due to bandwidth restrictions in communicating, the head-end accesses multiple meters sequentially, through data concentrators. The data is then transmitted in clusters, from the concentrators to the head-end system, but due to resource-constrained communication networks, it is challenging to obtain the relevant measurements in near real-time.

Figure 5.12 depicts a high-level architecture of an Advanced Meter Infrastructure (AMI), consisting of Smart Meters (SMs), Data Concentrators (DC), Head-End station (HE) and Distribution System Operator (DSO) control center. The low bandwidth AMI communication networks between DC and smart meters, result in high latency and infrequent updates, which strongly affects data quality in real-time data-demanding applications.

Classically, the information validity is quantified by so called *age* [234], that is the time from measurement until the data is being utilized. A deeper link from the information *age* to the signal dynamics leads to another common metric, called *mismatch probability* [235] (mmPr), which in previous works [236] has shown to be a useful parameter for describing information quality.

Since the timings at which SMs are accessed directly affects the age of the information, altering the scheduling order can determine the quality of the data information.

Several works addressed the scheduling problem under the assumpion of such metrics and concepts. In addition to [237] and [238], which face the problem without a specific policy but with brute-force analysis, [239], [240] and [241] use joint routing and TDM-based scheduling in wireless mesh networks. These works optimized the scheduling algorithm by taking into account communication network constraints incurred due to wireless interference.

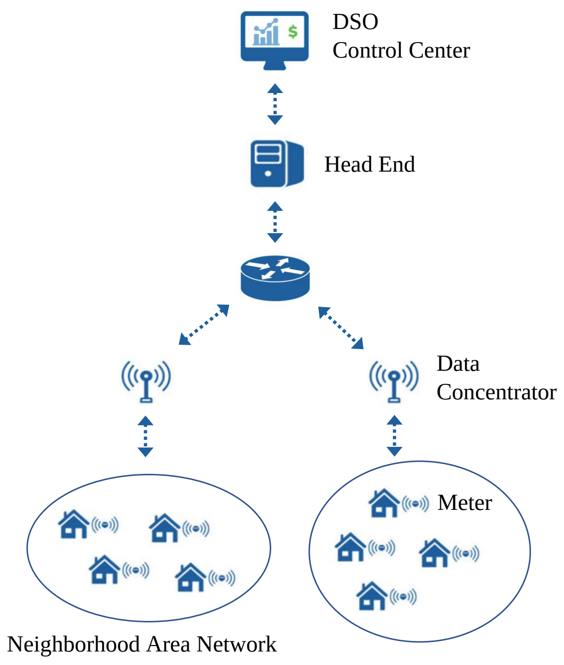


Figure 5.12: Typical Advanced Metering Infrastructure (AMI)

Recently, Farooq et al. [242] experiment with the use of the Hungarian algorithm, to find the best sequence for accessing smart meter data in case of low-performance networks.

System description

A time diagram showing how smart meters are accessed during an example reading cycle, is shown in fig. 5.13. The DC employs a reactive approach, which means that it submits a request for access to meter data, and the meter replies with the needed response. This information is forwarded by DC to the head-end (HE), which accumulates it and only at the end of the cycle forwards it to the controller or any other grid monitoring software. Every access to meters is associated by a specific access delay, which takes into account both communication stack delays and any potential cache or other systemic optimization measures that may have been used.

All of the smart meters' data is gathered within one collection cycle. When the previous data is transmitted to the controller, the subsequent cycle starts. After all the readings are taken, there is an idle time t_i that can be used for

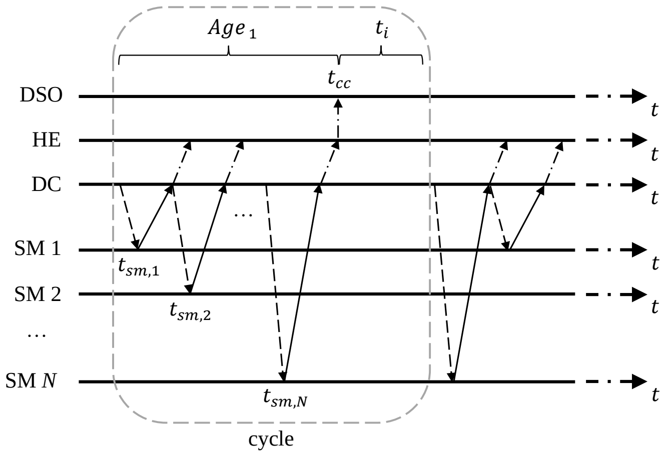


Figure 5.13: Example of SM data access time diagram. Dashed arrows represent data requests from DC, while solid arrows represent responses from SM. Dash-dot arrows mark data feedback from DC to DSO through HE.

features like updating the firmware or sending alerts.

Due to their position in the timetable, each meter experiences different ageing. For instance, on the highlighted cycle of fig. 5.13, meter N (smN) is accessed shortly before data transmission to the DSO system, since it is placed in the last position of the read queue; as a result, sm_N will have the shortest access delay and in a similar vein, sm_1 will experience the longest access delay.

Information quality metrics

In this work we considered the following performance metrics in relation to the access strategies:

- Information Age: the period of time between when data are acquired by the smart meter and when they are finally available to the DSO (for simplicity, the feedback time from HE to DSO is neglected). For example, the age of the n-th meter is expressed by:

$$Age_n = t_{cc} - t_{sm,n} \tag{5.13}$$

where t_{cc} and $t_{sm,n}$ are taken from the same read cycle (see fig. 5.13).

- Mismatch probability ($mmPr$): the probability that any of the N values of the information elements that are used at the requester does not match the current true value at the remote location, within a sensitivity threshold ε:

$$mmPr(n) = P\{I_n(t_{cc}) - I_n(t_{cc} - Age_n) > \varepsilon\} \tag{5.14}$$

where $I_n(\cdot)$ indicates the information message i.e. power, current or voltage readings, of n-th smart meter.

Dataset

To perform the simulations we used the free-access dataset published by the Indian Council on Energy, Environment and Water, as part of the study *"What Smart Meters Can Tell Us, Insights on Electricity Supply and Use in Mathura and Bareilly Households"* [243].

The data was collected using smart meters installed in nearly 100 urban households in Mathura and Bareilly districts of Uttar Pradesh, India. These smart meters recorded electricity consumption patterns and power supply information at three-minute intervals, from May 2019 to October 2021. The data also provides information on the situation of power supply (including hours timestamp, voltage, current withdrawn and other related variables). Figure 5.14 shows an example of the data series contained in the dataset; the consumption profile recorded from two households, over a 24-hours range.

GNN Assignment solver

To asses the impact of the access scheduling, we performed an analysis based on the system described above. We considered coverage areas with 4, 8, 16 and 32 smart meters and we made the following assumptions about the system dynamics:

- Meters are queried at fixed intervals of 15 minutes. This is currently the most frequent time resolution of meters readings, as stated in Refs. [237] and [242]. In addition, this assumption allows for a high level of abstraction that does not take into account the type of connection between each meter and the control center, whether it is through a slow or unreliable line, thus with high latency times, or through a reliable, wide-band

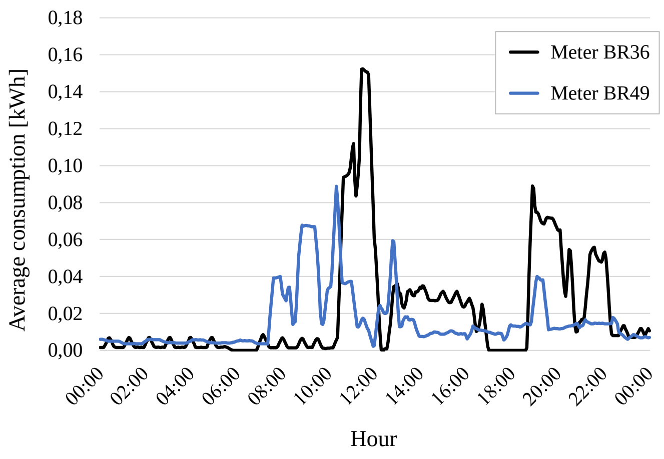


Figure 5.14: Consumption profile of two different households over 24 hours.

network. We assume that within the established time interval, data communication has occurred correctly.

- All meters are polled every cycle. This is a simplification since in real-world scenarios some SMs may be queried at a lower rate and thus excluded from one or more cycles. However, we bypass this option, which can be the subject of a future parametric study.

- Based on simulation studies, and relying on previous works [237, 242] we set the tolerance threshold for the mmPr parameter (ε) at 10 % of the mean power value.

The first step in the analysis is to determine the mmPr profiles for each of the meters, simulating every possible position in the read queue. As mentioned earlier, we considered four different scenarios for the total number of SMs inside the domain of a data concentrator: 4, 8, 16, and 32. We estimated the mmPr values over a 24-hour time frame, as we believe that the non-stationarity of the consumption data requires recalculation of the reading order after this time interval, to maintain the optimality of the scheduling criterion.

Figure 5.15 shows the trend of calculated mmPr values during a 24-hour time interval, for 8 smart meters from the dataset. Both the observation day and the smart meters were chosen with a random criterion, only for illustrative purposes.

Once the mmPr values for a single day were obtained, the problem of de-

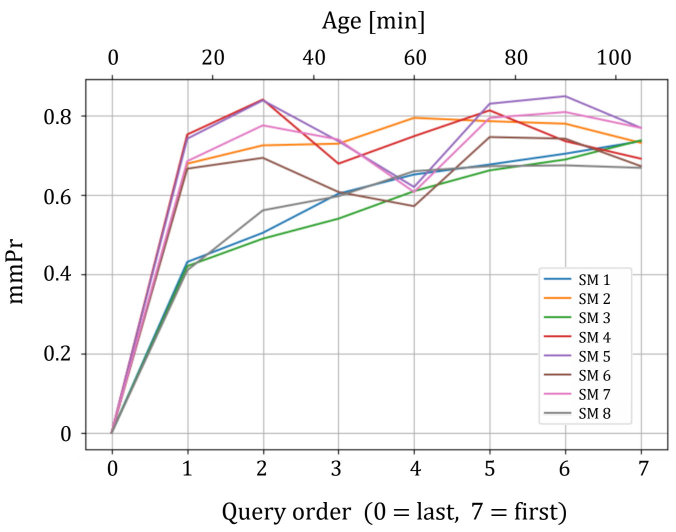


Figure 5.15: Calculated mmPr profiles, in relation to information age and position of SM in the reading queue.

termining the reading order was easily modeled as a linear sum assignment problem (LSAP), because a one-to-one assignment must be determined between the meters and the positions in the read queue, in order to identify the lowest "cost" that is the overall mmPr of a schedule.

The cost matrix was then defined by vectorizing the mmPr curves and overlaying them in an $N \times N$ grid, as shown on figure 5.16. The range of cost values lies in $[0, 1]$, so we could use the same training strategy as defined in chapter 4.

Experimental findings

In the following, two methods were applied to solve the assignment problem, the Hungarian algorithm, as defined in [194], and the GNN data-driven assignment method, which is sub-optimal but has lower complexity. The results are presented in the following tables.

Moreover, the fastest way to deal with the assignment problem is to disregard cost optimization and make random assignments. This obviously eliminates the computation time but drastically lowers the level of accuracy, which, for a problem of size N, can be calculated on a statistical basis as $1/N!$ being $N!$ the possible number of valid assignments.

Table 5.6 shows the total cost obtained from each of the two allocation meth-

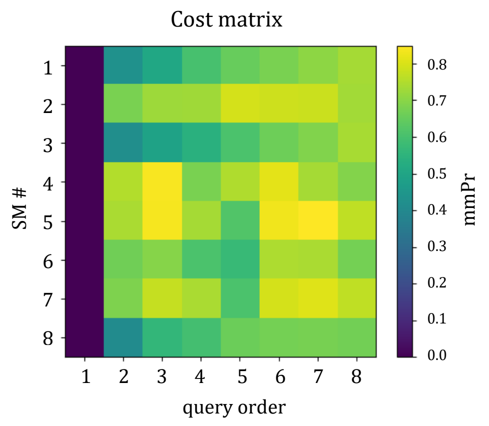


Figure 5.16: Example of the 8 × 8 cost matrix, related to the mmPr curves of figure 5.15.

Table 5.6: Assignment cost, e.g. mmPr values obtained by the Hungarian algorithm (Min), the worst possible solution (Max) and the GNN method.

Problem size	Min	Max	GNN	Relative error
$N = 4$	0.4134	0.5309	0.4153	1.61 %
$N = 8$	0.5032	0.6559	0.5071	2.55 %
$N = 16$	0.5661	0.7195	0.5699	2.48 %
$N = 32$	0.6318	0.8057	0.6353	2.01 %

ods, i.e. the minimum attainable mean mmPr. This value cannot be zero, since even with a few meters within each read cycle, mismatch of information is virtually inevitable. The minimum value of mmPr, given in the second column, also corresponds to that obtained by the Hungarian algorithm, since it is optimal. In contrast, the GNN approach fails to always determine the correct assignment, leading to sub-optimal scheduling. The column labeled with “Max” reports the upper bound of mmPr values, achieved with the worst scheduling. The rightmost column of table 5.6 reports the percentage deviation between the GNN prediction and the minimum value, within the spanning range.

Table 5.7 shows the accuracy in terms of matching assignments, between the optimal solution and the GNN solver.

Finally, table 5.8 reports the time results in the execution of the two algorithms, which is the main motivation in favor of the GNN approach.

As pointed out earlier, the size of the assignment problem greatly affects the

Table 5.7: Accuracy levels obtained by the Hungarian algorithm and the GNN method.

Problem size	Hungarian	GNN
$N = 4$	100 %	85.0 %
$N = 8$	100 %	75.0 %
$N = 16$	100 %	66.5 %
$N = 32$	100 %	65.0 %

Table 5.8: Mean execution time for a single-batch LSAP instance. Values are in μs.

Problem size	Hungarian	GNN
$N = 4$	88.2	353.8
$N = 8$	518.6	446.7
$N = 16$	4197.5	813.3
$N = 32$	34064.8	1209.0

execution time of the Hungarian algorithm, although it manages to reduce its complexity to $\mathcal{O}(N^3)$, which is much lower than the brute-force solution. Table 5.8 shows that GNN produces faster assignments, by about 1.1×, for $N = 8$, 5.1× for $N = 16$ and 28.2× for $N = 32$. The trend suggests that the advantage would grow for larger N. This is significant in real-world scenarios, as some smart grids may hold up to a hundred meters within the same subsystem, as stated in [237].

5.1.8 Final remarks

In this work we proposed a novel learning framework by adapting a data-driven approach of the linear sum assignment problem, and then compared the performances with two existing DNN-based strategies.

We demonstrated experimentally that the proposed approach has competitive performance, compared to previous MLP and CNN based approaches, regarding small assignment problems. For larger problems it is able to outperform significantly these approaches, while requiring significantly fewer computational resources.

Subsequently, this study introduced the typical components of an AMI, Advanced Meter Infrastructure of a smart grid, explaining how a proper scheduling of the smart meters query order can minimize information obsolescence during the data collection process. Smart meter power readings from different grid spans are employed, as experimental settings. Although simulation results show lower accuracy of the GNN approach compared to the optimal solution, there is a notable gain in computational demand and execution time, the more so as the number of counters in the neighborhood area network is larger.

Furthermore, we observed that the incorrect assignments of the GNN network actually minimally affect the error in terms of total cost. This indicates that although the network has been trained to minimize the assignments on a binary cross-entropy basis, it also succeeds to exploit the cost values on its decision mechanism, and tends to produce assignments which while not optimal (lower accuracy) have total assignment cost close to the optimal solution.

Future work will attempt to explore new application areas for the assignment problem, such as integrating the proposed GNN solver into a speaker-independent multi-talker speech separation model [244–246] in order to increase performance or speed up training time. In such systems the network produces multiple outputs in an unpredictable order, which must be assigned to the corresponding target signals to perform properly supervised training. Assignment is made based on a metric that quantifies the similarity between each target signal and the possible candidate output.

Another potential application we might consider is the use of the GNN LSAP solver for Multi-Object Tracking and Segmentation (MOTS). This involves not only detecting and segmenting objects in a sequence of video frames, but also assigning consistent IDs to each visible instance of the same object. Typically, this is formulated as consecutive assignment problems involving a few dozen objects and must be carried out under tight constraints, to be in sync with a real-time video stream [247]. In such context, the use of the proposed sub-optimal GNN solver could bring considerable advantage.

Chapter 6

Conclusions

In this dissertation we addressed the issues of sound signals analysis, and their most suitable representations for use within deep neural networks, and subsequently investigated sound sequence restoration techniques, using neural networks, as these having become the de-facto standard for most speech enhancement tasks. An overview was given of both restoration approaches proposed in the scientific literature and those currently employed in modern communication systems.

Chapter 2 aimed to provide the reader with the basic concepts necessary to understand the topics developed in the following chapters, and in particular it set out an overview of the most popular feature representation of audio signals, currently used in digital signal processing, explained the concepts of sound event detection and classification tasks, provided the founding principles of deep learning techniques on graph-structured data, and finally illustrated the concept of generative artificial intelligence, setting out the working principle of the most successful paradigms.

In chapter 3 we delved into the use of graph neural networks for audio processing applications, first briefly listing existing work that, in different ways, exploits graph structures for modeling the aspect of the task at hand. Then the topic of multilabel classification has been further explored, by also analyzing domains other than audio processing, where graph structures have been exploited to enrich the learning process by modeling the relationships between classes of objects or events. The chapter then continued with the exposition of the two papers produced during the doctoral activity, the first of which sees the proposal of an innovative method for the representation in graph form of features, derived from the sound spectrogram of a recording, through the application of geometric concepts, and the subsequent application for a sound event classification (SEC) task. The resulting compact representation proved effective at extracting complementary informative features from the audio spectrogram, bringing noticeable improvements when used in a hybrid CNN-GNN architecture for SEC. The second paper presented in chapter 3 dealt with the semantic representation of event co-occurrences in a SEC dataset, through a graph structure, which was then integrated into a classical pipeline for sound event classification, demonstrating its effectiveness in a weakly labeled data context.

Chapter 4 presented the topic of repairing audio sequences, corrupted as a result of packet losses, due to networks affected by various types of imperfections such as noise, limited bandwidth, outdated hardware, or traffic congestion. The chapter introduced the problems of packet loss concealment (PLC) and audio inpainting, defined the various models adopted for representing the problem, devised evaluation criteria for judging audio quality and outlined the techniques currently in the literature for its solution, both algorithmic, statistical, and DNN-based approaches, the latter being the most recent and best performing. Then the three works that address the packet loss concealment problem through generative neural networks were presented. Specifically, the generative adversarial network (GAN) paradigm was employed, first adapting a framework, already existing in the literature and successfully used for many image-to-image translation tasks, pix2pix, for inpainting damaged spectrograms. Subsequent work borrows the same framework and adapts it to the task of repairing signals containing musical sequences, introducing an additional comparison criterion based on the musical score, to increase performance and the ability to reconstruct large gaps, up to 750ms. Finally, the last work presented in chapter 4 dealt with the PLC problem by inpainting complex-valued spectrograms. This allowed the signal phase to be considered in the reconstruction process as well, an aspect that is typically neglected by the proposed approaches. Furthermore, the method was developed to have "adaptive" behavior, in the sense that its operation can be tuned at runtime to find a tradeoff between reconstruction quality and computational latency. This is of utmost importance in practical applications that have to run on resource-constrained devices. The three proposed methods presented state-of-the-art performances in comparison with similar recent methodologies based on deep neural networks, including the best performing ones based on generative neural networks.

Finally, in chapter 5, a contribution was presented that is somewhat separate from the audio processing topic, but still falls within the study of graph neural networks. This is the modeling in graph form of the Linear Sum assignment problem, known in combinatorial optimization, which frequently appears in various disciplines, as part of the solving process of specific tasks. The proposed GNN-based solver is sub-optimal and has been shown to be superior to other currently proposed DNN-based approaches, being at the same time competitive in terms of scalability.

Future perspectives

At the time this thesis is being written, research activities on the topic of reconstruction of damaged audio signals, are currently ongoing, driven by the growing interest of the research community in this topic and the proliferation of generative networks strategies. Two strands of research out of all possibly offer the most interesting perspectives: first, the field of denoising diffusion probabilistic models (DDPM) [248], which originated for image synthesis tasks and are rooted on non-equilibrium thermodynamics, are successfully forming the basis of many speech enhancement approaches, and although they still have downsides, related mainly to the difficulty of parallelizing computation, the quality of the samples they produce is astounding. Second, a promising approach to PLC should be complementing conditional-GAN networks, with a context characterization model, based on the transformer architecture [249]. The latter has seen considerable success in recent years being the basis of GPT models that, in the natural language processing field, produced the “AI revolution” known to the general public. In fact, some preliminary experiments we conducted confirmed the initial intuitions that conditioning the GAN with a signal drawn from the latent dimension of the transformer, succeeds in transferring to the generator a wide overview of the semantic content of the words contained in the recording. This allows the reconstruction of the missing gaps in a very selective manner, and leaves the generator with the sole task of modeling the speech pattern.

Bibliography

[1] Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. Fsd50k: an open dataset of human-labeled sound events. *arXiv preprint arXiv:2010.00475*, 2020.

[2] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203, 2020.

[3] Richard Lyon. Machine hearing: An emerging field. *Signal Processing Magazine, IEEE*, 27:131 – 139, 2010.

[4] Michel Vacher, Jean-François Serignat, and Stephane Chaillol. Sound classification in a smart room environment: an approach using gmm and hmm methods. In *IEEE Conference on Speech Technology and Human-Computer Dialogue*, volume 1, pages 135–146, 2007.

[5] Regunathan Radhakrishnan, Ajay Divakaran, and A Smaragdis. Audio analysis for surveillance applications. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 158–161, 2005.

[6] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley. Acoustic scene classification: Classifying environments from the sounds they produce. *IEEE Signal Processing Magazine*, 32:16–34, 2015.

[7] Sharath Adavanne, Pasi Pertilä, and Tuomas Virtanen. Sound event detection using spatial features and convolutional recurrent neural network. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, page 771–775. IEEE Press, 2017.

[8] Justin Salamon, Duncan MacConnell, Mark Cartwright, Peter Li, and Juan Pablo Bello. Scaper: A library for soundscape synthesis and augmentation. In *2017 IEEE Workshop on Applications of Signal Processing*

to Audio and Acoustics, WASPAA 2017, pages 344–348. Institute of Electrical and Electronics Engineers Inc., 2017.

[9] Haomin Zhang, Ian McLoughlin, and Yan Song. Robust sound event recognition using convolutional neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 559–563. IEEE, 2015.

[10] K. J. Piczak. Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2015.

[11] Sharath Adavanne, Konstantinos Drossos, Emre Çakir, and Tuomas Virtanen. Stacked convolutional and recurrent neural networks for bird audio detection. In *EUSIPCO*, pages 1729–1733, 2017.

[12] Tuomas Virtanen, Annamaria Mesaros, Toni Heittola, Mark D. Plumbley, Peter Foster, Emmanouil Benetos, and Mathieu Lagrange. *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016).* Tampere University of Technology. Department of Signal Processing, 2016.

[13] Burak Uzkent, Buket Barkana, and Hakan Cevikalp. Non-speech environmental sound classification using svms with a new set of features. *International Journal of Innovative Computing, Information and Control*, 8, 05 2012.

[14] Annamaria Mesaros, Toni Heittola, Antti Eronen, and Tuomas Virtanen. Acoustic event detection in real life recordings. In *2010 18th European Signal Processing Conference*, pages 1267–1271, 2010.

[15] S. Chu, S. Narayanan, and C. . J. Kuo. Environmental sound recognition with time–frequency audio features. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(6):1142–1158, 2009.

[16] Xavier Valero and Francesc Alías. Gammatone cepstral coefficients: Biologically inspired features for non-speech audio classification. *Multimedia, IEEE Transactions on*, 14:1684–1689, 2012.

[17] J. T. Geiger and K. Helwani. Improving event detection for audio surveillance using gabor filterbank features. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 714–718, 2015.

[18] Justin Salamon and Juan Pablo Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal processing letters*, 24(3):279–283, 2017.

[19] Sajjad Abdoli, Patrick Cardinal, and Alessandro Lameiras Koerich. End-to-end environmental sound classification using a 1d convolutional neural network. *Expert Systems with Applications*, 136:252–263, 2019.

[20] Emre Cakır, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1291–1303, 2017.

[21] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 131–135, 2017.

[22] Anurag Kumar and Bhiksha Raj. Audio event detection using weakly labeled data. *Proceedings of the 24th ACM international conference on Multimedia*, 2016.

[23] Anurag Kumar and Bhiksha Raj. Deep cnn framework for audio event recognition using weakly labeled web data. *ArXiv*, abs/1707.02530, 2017.

[24] Ting-Wei Su, Jen-Yu Liu, and Yi-Hsuan Yang. Weakly-supervised audio event detection using event-specific gaussian filters and fully convolutional networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 791–795, 2017.

[25] Anurag Kumar, Maksim Khadkevich, and Christian Fügen. Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 326–330, 2018.

[26] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

[27] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[28] M Gori, G Monfardini, and F Scarselli. A new model for learning in graph domains. In *International Joint Conference on Neural Networks*, volume 2, pages 729–734. IEEE, 2005.

[29] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[30] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007.

[31] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

[32] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[33] Amir Hosein Khas Ahmadi, Kaveh Hassani, Parsa Moradi, Leo Lee, and Quaid Morris. Memory-based graph networks. *CoRR*, abs/2002.09518, 2020.

[34] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

[35] Hao Yuan and Shuiwang Ji. Structpool: Structured graph pooling via conditional random fields. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

[36] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[37] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[38] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[39] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv*, 2017.

[40] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[41] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[42] Amir Shirian and Tanaya Guha. Compact graph architecture for speech emotion recognition. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6284–6288, 2021.

[43] Taichi Ishiwatari, Yuki Yasuda, Taro Miyazaki, and Jun Goto. Relation-aware graph attention networks with relational position encodings for emotion recognition in conversations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7360–7370, 01 2020.

[44] Panagiotis Tzirakis, Anurag Kumar, and Jacob Donley. Multi-channel speech enhancement using graph neural networks. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3415–3419, 2021.

[45] Weizhi Nie, Minjie Ren, Jie Nie, and Sicheng Zhao. C-gcn: Correlation based graph convolutional network for audio-video emotion recognition. *IEEE Transactions on Multimedia*, 23:3793–3804, 2021.

[46] Shuyun Tang, Zhaojie Luo, Guoshun Nan, Jun Baba, Yuichiro Yoshikawa, and Hiroshi Ishiguro. Fusion with hierarchical graphs for multimodal emotion recognition. In *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1288–1296, 2022.

[47] Yiwei Sun and Shabnam Ghaffarzadegan. An ontology-aware framework for audio event classification. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 321–325. IEEE, 2020.

[48] Helin Wang, Yuexian Zou, Dading Chong, and Wenwu Wang. Modeling label dependencies for audio tagging with graph convolutional network. *IEEE Signal Processing Letters*, 27:1560–1564, 2020.

[49] Shilei Zhang, Yong Qin, Kewei Sun, and Yonghua Lin. Few-shot audio classification with attentional graph neural networks. In *Proc. Interspeech 2019*, pages 3649–3653, 2019.

[50] Harsh Shrivastava, Yfang Yin, Rajiv Ratn Shah, and Roger Zimmermann. Mt-gcn for multi-label audio-tagging with noisy labels. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 136–140, 2020.

[51] Cyrus Vahidi, Shubhr Singh, Emmanouil Benetos, Huy Phan, Dan Stowell, György Fazekas, and Mathieu Lagrange. Perceptual musical similarity metric learning with graph neural networks. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2023)*, New Paltz, NY, United States, October 2023.

[52] Eric Grinstein, Mike Brookes, and Patrick A Naylor. Graph neural networks for sound source localization on distributed microphone networks. In *2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[53] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2285–2294, 06 2016.

[54] Qiang Li, Maoying Qiao, Wei Bian, and Dacheng Tao. Conditional graphical lasso for multi-label image classification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2977–2986, 2016.

[55] Xin Li, Feipeng Zhao, and Y. Guo. Multi-label image classification with a probabilistic label enhancement model. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI) 2014*, pages 430–439, 01 2014.

[56] Feng Zhu, Hongsheng Li, Wanli Ouyang, Nenghai Yu, and Xiaogang Wang. Learning spatial regularization with image-level supervisions for multi-label image classification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2027–2036, 2017.

[57] Zhouxia Wang, Tianshui Chen, Guanbin Li, Ruijia Xu, and Liang Lin. Multi-label image recognition by recurrently discovering attentional regions. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 464–472, 2017.

[58] Deepak Saini, Arnav Kumar Jain, Kushal Dave, Jian Jiao, Amit Singh, Ruofei Zhang, and Manik Varma. Galaxc: Graph neural networks with labelwise attention for extreme classification. *Proceedings of the Web Conference 2021*, 2021.

[59] Min Shi, Yufei Tang, Xingquan Zhu, and Jianxun Liu. Multi-label graph convolutional network representation learning. *IEEE Transactions on Big Data*, 8(5):1169–1181, 2022.

[60] Jack Lanchantin, Arshdeep Sekhon, and Yanjun Qi. Neural message passing for multi-label classification. In *Machine Learning and Knowledge Discovery in Databases*, page 138–163. Springer-Verlag, 2019.

[61] Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. Multi-label image recognition with graph convolutional networks. In *IEEE/CVF conference on computer vision and pattern recognition*, pages 5177–5186, 2019.

[62] Ankit Pal, Selvakumar Murugan, and Malaikannan Sankarasubbu. Magnet: Multi-label text classification using attention-based graph neural network. *ArXiv*, pages 494–505, 01 2020.

[63] Daoming Zong and Shiliang Sun. Bgnn-xml: Bilateral graph neural networks for extreme multi-label text classification. *IEEE Transactions on Knowledge &; Data Engineering*, 35(07):6698–6709, jul 2023.

[64] Qianwen Ma, Chunyuan Yuan, Wei Zhou, and Songlin Hu. Label-specific dual graph neural network for multi-label text classification. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 3855–3864. Association for Computational Linguistics, 01 2021.

[65] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *ACM International Conference on Multimedia*, page 1041–1044, 2014.

[66] Carlo Aironi, Samuele Cornell, Emanuele Principi, and Stefano Squartini. Graph-based representation of audio signals for sound event classification. In *EUSIPCO*, pages 566–570. IEEE, 2021.

[67] Kaan Ersahin, Ian G. Cumming, and Rabab Kreidieh Ward. Segmentation and classification of polarimetric sar data using spectral graph partitioning. *IEEE Transactions on Geoscience and Remote Sensing*, 48(1):164–174, 2010.

[68] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[69] Nikolaos D. Katopodes. *Free-Surface Flow: computational methods.* Elsevier, 2019.

[70] Karol J. Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM International Conference on Multimedia*, page 1015–1018, 2015.

[71] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017.

[72] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *CoRR*, abs/1704.02901, 2017.

[73] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *ArXiv*, abs/1903.02428, 2019.

[74] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[75] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1923, 1998.

[76] Carlo Aironi, Samuele Cornell, Emanuele Principi, and Stefano Squartini. Graph node embeddings for ontology-aware sound event classification: an evaluation study. In *2022 30th European Signal Processing Conference (EUSIPCO)*, pages 414–418, 2022.

[77] Yun Wang, Leonardo Neves, and Florian Metze. Audio-based multimedia event detection using deep recurrent neural networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2742–2746, 2016.

[78] Toan H Vu and Jia-Ching Wang. Acoustic scene and event recognition using recurrent neural networks. *Detection and Classification of Acoustic Scenes and Events*, 2016:1–3, 2016.

[79] Ayşegül Özkaya Eren and Mustafa Sert. Audio captioning using gated recurrent units. *arXiv preprint arXiv:2006.03391*, 2020.

[80] Yong Xu, Qiuqiang Kong, Wenwu Wang, and Mark D Plumbley. Large-scale weakly supervised audio classification using gated convolutional neural network. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 121–125, 2018.

[81] Miroslav Malik, Sharath Adavanne, Konstantinos Drossos, Tuomas Virtanen, Dasa Ticha, and Roman Jarina. Stacked convolutional and recurrent neural networks for music emotion recognition. *arXiv preprint arXiv:1706.02292*, 2017.

[82] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *IEEE conference on computer vision and pattern recognition*, pages 6857–6866, 2018.

[83] Yimin Zhou, Yiwei Sun, and Vasant Honavar. Improving image captioning by leveraging knowledge graphs. In *IEEE winter conference on applications of computer vision*, pages 283–293, 2019.

[84] Bingzhi Chen, Jinxing Li, Guangming Lu, Hongbing Yu, and David Zhang. Label co-occurrence learning with graph convolutional networks for multi-label chest x-ray image classification. *IEEE journal of biomedical and health informatics*, 24(8):2292–2302, 2020.

[85] Mehdi Allahyari, Krys J Kochut, and Maciej Janik. Ontology-based text classification into dynamically defined topics. In *IEEE international conference on semantic computing*, pages 273–278, 2014.

[86] Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Conference on empirical methods in natural language processing*, pages 349–357, 2018.

[87] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[88] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[89] Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

[90] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, pages 2014–2023. PMLR, 2016.

[91] Claudio Gallicchio and Alessio Micheli. Graph echo state networks. In *2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.

[92] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[93] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[94] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017.

[95] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[96] Carlos Gouvea and Carlos Pedroso. Mac-layer packet loss models for wi-fi networks: A survey. *IEEE Access*, 7:1–21, 12 2019.

[97] Dhwani R. Bhadra, Charmi A. Joshi, Priya R. Soni, Nikita P. Vyas, and Rutvij H. Jhaveri. Packet loss probability in wireless networks: A survey. In *2015 International Conference on Communications and Signal Processing (ICCSP)*, pages 1348–1354, 2015.

[98] Mostafa M Mohamed and Björn W Schuller. Concealnet: An end-to-end neural network for packet loss concealment in deep speech emotion recognition. *arXiv preprint arXiv:2005.07777*, 2020.

[99] Bong-Ki Lee and Joon-Hyuk Chang. Packet loss concealment based on deep neural networks for digital speech transmission. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(2):378–387, 2015.

[100] Jie Wang, Yuansheng Guan, Chengshi Zheng, Renhua Peng, and Xiaodong Li. A temporal-spectral generative adversarial network based end-to-end packet loss concealment for wideband speech transmission. *The Journal of the Acoustical Society of America*, 150(4):2577–2588, 2021.

[101] Santiago Pascual, Joan Serrà, and Jordi Pons. Adversarial auto-encoding for packet loss concealment. In *2021 IEEE Workshop on Applications*

of Signal Processing to Audio and Acoustics (WASPAA), pages 71–75. IEEE, 2021.

[102] Amir Adler, Valentin Emiya, Maria G Jafari, Michael Elad, Rémi Gribonval, and Mark D Plumbley. Audio inpainting. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(3):922–932, 2011.

[103] Pirmin P Ebner and Amr Eltelt. Audio inpainting with generative adversarial network. *arXiv preprint arXiv:2003.07704*, 2020.

[104] Andrés Marafioti, Nathanaël Perraudin, Nicki Holighaus, and Piotr Majdak. A context encoder for audio inpainting. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):2362–2372, 2019.

[105] Ondřej Mokrỳ and Pavel Rajmic. Audio inpainting: Revisited and reweighted. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2906–2918, 2020.

[106] Ya-Liang Chang, Kuan-Ying Lee, Po-Yu Wu, Hung-yi Lee, and Winston Hsu. Deep long audio inpainting. *arXiv preprint arXiv:1911.06476*, 2019.

[107] Jérémie Lecomte, Tommy Vaillancourt, Stefan Bruhn, Hosang Sung, Ke Peng, Kei Kikuiri, Bin Wang, Shaminda Subasingha, and Julien Faure. Packet-loss concealment technology advances in evs. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5708–5712, 2015.

[108] Jean-Marc Valin, Gregory Maxwell, Timothy B Terriberry, and Koen Vos. High-quality, low-delay music coding in the opus codec. In *135th AES Convention, October 17–20 2013, New York, USA*, 2013.

[109] 3rd generation partnership project, “adaptive multi-rate (amr) speech codec. error concealment of lost frames, 2004.

[110] Masahiro Toyoshima and Tetsuya Shimamura. Packet loss concealment for voip based on pitch waveform replication and linear predictive coding. In *2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pages 89–92. IEEE, 2014.

[111] C.A. Rodbro, M.N. Murthi, S.V. Andersen, and S.H. Jensen. Hidden markov model-based packet loss concealment for voice over ip. *IEEE*

Transactions on Audio, Speech, and Language Processing, 14(5):1609–1623, 2006.

[112] Bengt J. Borgström, Per H. Borgström, and Abeer Alwan. Efficient HMM-based estimation of missing features, with applications to packet loss concealment. In *Proc. Interspeech 2010*, pages 2394–2397, 2010.

[113] Georg Tauböck, Shristi Rajbamshi, and Peter Balazs. Dictionary learning for sparse audio inpainting. *IEEE Journal of Selected Topics in Signal Processing*, 15(1):104–119, 2020.

[114] Florian Lieb and Hans-Georg Stark. Audio inpainting: Evaluation of time-frequency representations and structured sparsity approaches. *Signal Processing*, 153:291–299, 2018.

[115] Fatiha Merazka. Packet loss concealment by interpolation for speech over ip network services. In *2013 Constantinides International Workshop on Signal Processing (CIWSP 2013)*, pages 1–4, 2013.

[116] Nathanaël Perraudin, Nicki Holighaus, Piotr Majdak, and Peter Balazs. Inpainting of long audio segments with similarity graphs. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(6):1083–1094, 2018.

[117] Yao Zhou, Changchun Bao, Jinwei Huang, and Yunhao Zhao. A neural vocoder based packet loss concealment algorithm. In *2022 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pages 1–5, 2022.

[118] Reza Lotfidereshgi and Philippe Gournay. Speech prediction using an adaptive recurrent neural network with application to packet loss concealment. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5394–5398. IEEE, 2018.

[119] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[120] Andrés Marafioti, Nicki Holighaus, Piotr Majdak, and Nathanaël Perraudin. Audio inpainting of music by means of neural networks. In *Audio Engineering Society Convention, 146*, 2019.

[121] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI*, pages 234–241, 2015.

[122] Mikolaj Kegler, Pierre Beckmann, and Milos Cernak. Deep speech inpainting of time-frequency masks. In *Proc. Interspeech 2020*, pages 3276–3280, 2020.

[123] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[124] Yupeng Shi, Nengheng Zheng, Yuyong Kang, and Weicong Rong. Speech loss compensation by generative adversarial networks. In *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 347–351, 2019.

[125] Santiago Pascual, Antonio Bonafonte, and Joan Serrà. Segan: Speech enhancement generative adversarial network. In *Proc. Interspeech 2017*, pages 3642–3646, 08 2017.

[126] Daniel Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on acoustics, speech, and signal processing*, 32(2):236–243, 1984.

[127] Zdenek Průša and Peter L Søndergaard. Real-time spectrogram inversion using phase gradient heap integration. In *Proceedings of International Conference on Digital Audio Effects (DAFx-16)*, pages 17–21, 2016.

[128] Philipos C Loizou. *Speech enhancement: theory and practice, 2nd edition.* CRC press, 2013.

[129] A. Bayya and M. Vis. Objective measures for speech quality assessment in wireless communications. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference (ICASSP)*, volume 1, pages 495–498 vol. 1, 1996.

[130] E. Vincent, R. Gribonval, and C. Fevotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1462–1469, 2006.

[131] Cees H Taal, Richard C Hendriks, Richard Heusdens, and Jesper Jensen. A short-time objective intelligibility measure for time-frequency weighted noisy speech. In *2010 IEEE international conference on acoustics, speech and signal processing*, pages 4214–4217. IEEE, 2010.

[132] Chandan K A Reddy, Vishak Gopal, and Ross Cutler. Dnsmos: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6493–6497, 2021.

[133] Lorenz Diener, Marju Purin, Sten Sootla, Ando Saabas, Robert Aichner, and Ross Cutler. PLCMOS – A Data-driven Non-intrusive Metric for The Evaluation of Packet Loss Concealment Algorithms. In *Proc. INTERSPEECH 2023*, pages 2533–2537, 2023.

[134] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[135] Carlo Aironi, Samuele Cornell, Luca Serafini, and Stefano Squartini. A time-frequency generative adversarial based method for audio packet loss concealment. *arXiv preprint 2307.15611*, 2023.

[136] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[137] Chae Young Lee, Anoop Toffy, Gue Jun Jung, and Woo-Jin Han. Conditional wavegan. *arXiv preprint arXiv:1809.10636*, 2018.

[138] Kundan Kumar, Rithesh Kumar, Thibault De Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in neural information processing systems*, 32, 2019.

[139] Jinhyeok Yang, Junmo Lee, Youngik Kim, Hoon-Young Cho, and Injung Kim. VocGAN: A High-Fidelity Real-Time Vocoder with a Hierarchically-Nested Adversarial Network. In *Proc. Interspeech 2020*, pages 200–204, 2020.

[140] Kasperi Palkama, Lauri Juvela, and Alexander Ilin. Conditional Spoken Digit Generation with StyleGAN. In *Proc. Interspeech 2020*, pages 3166–3170, 2020.

[141] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[142] Takeru Miyato and Masanori Koyama. cGANs with projection discriminator. In *International Conference on Learning Representations*, 2018.

[143] Nathanaël Perraudin, Peter Balazs, and Peter L. Søndergaard. A fast griffin-lim algorithm. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4, 2013.

[144] Sercan Ö Arık, Heewoo Jun, and Gregory Diamos. Fast spectrogram inversion using multi-head convolutional neural networks. *IEEE Signal Processing Letters*, 26(1):94–98, 2018.

[145] Junichi Yamagishi, Christophe Veaux, and Kirsten MacDonald. CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit (version 0.92), 2019.

[146] Niklas Blum, Serge Lachapelle, and Harald Alvestrand. Webrtc: Real-time communication for the open web platform. *Commun. ACM*, page 50–54, jul 2021.

[147] Baiyun Liu, Qi Song, Mingxue Yang, Wuwen Yuan, and Tianbao Wang. Plcnet: Real-time packet loss concealment with semi-supervised generative adversarial network. *Proc. Interspeech 2022*, pages 575–579, 2022.

[148] ITU-T Recommendation. Perceptual evaluation of speech quality (pesq): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. *Rec. ITU-T P. 862*, 2001.

[149] Miao Wang, Christoph Boeddeker, Rafael G. Dantas, and Amanda Seelan. Pesq (perceptual evaluation of speech quality) wrapper for python users, May 2022.

[150] Manuel Pariente. Pystoi, python implementation of stoi, 2018.

[151] Kaiyang Liu, Wendong Gan, and Chenchen Yuan. Maid: A conditional diffusion model for long music audio inpainting. In *2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[152] Carlo Aironi, Samuele Cornell, Leonardo Gabrielli, and Stefano Squartini. A score-aware generative approach for music signals inpainting. In *2023 4th International Symposium on the Internet of Sounds*, pages 1–7, 2023.

[153] Judith C Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[154] Eric Humphrey and Juan Bello. Rethinking automatic chord recognition with convolutional neural networks. In *Proc. ICML 2012*, volume 2, pages 357–362, 12 2012.

[155] Eric Humphrey and Juan Bello. From music audio to chord tablature: Teaching deep convolutional networks toplay guitar. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6974–6978, 05 2014.

[156] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. *International Society of Music Information Retrieval*, 2016.

[157] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 24(5):927–939, may 2016.

[158] Beat Gfeller, Christian Frank, Dominik Roblek, Matt Sharifi, Marco Tagliasacchi, and Mihajlo Velimirović. Spice: Self-supervised pitch estimation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1118–1128, 2020.

[159] Kalyanmoy Deb, Karthik Sindhya, and Jussi Hakanen. Multi-objective optimization. In *Decision sciences*, pages 161–200. CRC Press, 2016.

[160] Christof Weiß and Geoffroy Peeters. Comparing deep models and evaluation strategies for multi-pitch estimation in music recordings. *IEEE/ACM*

Transactions on Audio, Speech, and Language Processing, 30:2814–2827, 2022.

[161] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*, 2019.

[162] Frank Baumgarte and Christof Faller. Binaural cue coding - part i: Psychoacoustic fundamentals and design principles. *Speech and Audio Processing, IEEE Transactions on*, 11:509 – 519, 12 2003.

[163] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[164] Stephen C. Welch. Perceptual coding in python. `https://github.com/stephencwelch/Perceptual-Coding-In-Python`, 2015.

[165] Peter Kabal et al. An examination and interpretation of itu-r bs. 1387: Perceptual evaluation of audio quality. *TSP Lab Technical Report, Dept. Electrical & Computer Engineering, McGill University*, pages 1–89, 2002.

[166] Leyuan Sheng, Dong-Yan Huang, and Evgeniy N Pavlovskiy. High-quality speech synthesis using super-resolution mel-spectrogram. *arXiv preprint arXiv:1912.01167*, 2019.

[167] Yuzhou Liu and DeLiang Wang. Divide and conquer: A deep casa approach to talker-independent monaural speaker separation. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 27(12):2092–2102, dec 2019.

[168] Zhong-Qiu Wang and DeLiang Wang. Deep learning based target cancellation for speech dereverberation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:941–950, 2020.

[169] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, Los Alamitos, CA, USA, jul 2017. IEEE Computer Society.

[170] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[171] Jingshu Zhang, Mark D. Plumbley, and Wenwu Wang. Weighted magnitude-phase loss for speech dereverberation. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5794–5798, 2021.

[172] Juan Manuel Martin-Doñas, Angel Manuel Gomez, Jose A. Gonzalez, and Antonio M. Peinado. A deep learning loss function based on the perceptual evaluation of the speech quality. *IEEE Signal Processing Letters*, 25(11):1680–1684, 2018.

[173] Lorenz Diener, Sten Sootla, Solomiya Branets, Ando Saabas, Robert Aichner, and Ross Cutler. Interspeech 2022 audio deep packet loss concealment challenge. In *Proc. Interspeech 2022*, pages 580–584, 09 2022.

[174] Chandan K A Reddy, Vishak Gopal, and Ross Cutler. Dnsmos p.835: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors. In *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 886–890, 2022.

[175] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR, 2023.

[176] Bong-Ki Lee and Joon-Hyuk Chang. Packet loss concealment based on deep neural networks for digital speech transmission. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(2):378–387, 2016.

[177] Ju Lin, Yun Wang, Kaustubh Kalgaonkar, Gil Keren, Didi Zhang, and Christian Fuegen. A time-domain convolutional recurrent network for packet loss concealment. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7148–7152, 2021.

[178] Mohamed Nabih Ali, Alessio Brutti, and Daniele Falavigna. Speech enhancement using dilated wave-u-net: an experimental analysis. In *2020*

27th Conference of Open Innovations Association (FRUCT), pages 3–9, 2020.

[179] Jean Marc Valin, Ahmed Mustafa, Christopher Montgomery, Timothy B. Terriberry, Michael Klingbeil, Paris Smaragdis, and Arvindh Krishnaswamy. Real-time packet loss concealment with mixed generative and predictive model. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2022-September:570–574, 2022.

[180] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2410–2419. PMLR, 10–15 Jul 2018.

[181] Rainer Ernst Burkard and Eranda Dragoti-Cela. *Linear assignment problems and extensions*. Kluwer Academic Publishers, 1st edition, 1999.

[182] Wei Lian and Lei Zhang. A concave optimization algorithm for matching partially overlapping point sets. *Pattern Recognition*, 103:107322, 2020.

[183] Nina ST Hirata and Frank D Julca-Aguilar. Matching based ground-truth annotation for online handwritten mathematical expressions. *Pattern Recognition*, 48(3):837–848, 2015.

[184] Kevin Smith, Daniel Gatica-Perez, Jean-Marc Odobez, and Sileye Ba. Evaluating multi-object tracking. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops*, pages 36–36. IEEE, 2005.

[185] Robert M Haralick and Linda G Shapiro. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985.

[186] Guanding Yu, Lukai Xu, Daquan Feng, Rui Yin, Geoffrey Ye Li, and Yuhuan Jiang. Joint mode selection and resource allocation for device-to-device communications. *IEEE transactions on communications*, 62(11):3814–3824, 2014.

[187] Xiaofeng Lu, Qiang Ni, Wenna Li, and Hailin Zhang. Dynamic user grouping and joint resource allocation with multi-cell cooperation for uplink virtual mimo systems. *IEEE Transactions on Wireless Communications*, 16(6):3854–3869, 2017.

[188] Hao Song, Xuming Fang, and Yuguang Fang. Unlicensed spectra fusion and interference coordination for lte systems. *IEEE Transactions on Mobile Computing*, 15(12):3171–3184, 2016.

[189] Dong Yu, Morten Kolbæk, Zheng-Hua Tan, and Jesper Jensen. Permutation invariant training of deep models for speaker-independent multi-talker speech separation. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 241–245. IEEE, 2017.

[190] Yusuke Fujita, Naoyuki Kanda, Shota Horiguchi, Yawen Xue, Kenji Nagamatsu, and Shinji Watanabe. End-to-end neural speaker diarization with self-attention. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 296–303. IEEE, 2019.

[191] Desh Raj, Leibny Paola Garcia-Perera, Zili Huang, Shinji Watanabe, Daniel Povey, Andreas Stolcke, and Sanjeev Khudanpur. Dover-lap: A method for combining overlap-aware diarization outputs. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 881–888. IEEE, 2021.

[192] Shinji Watanabe, Michael Mandel, Jon Barker, and Emmanuel Vincent. Chime-6 challenge: Tackling multispeaker speech recognition for unsegmented recordings. In *CHiME 2020 - 6th International Workshop on Speech Processing in Everyday Environments*, Barcelona / Virtual, Spain, May 2020.

[193] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[194] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[195] Dimitri P Bertsekas. The auction algorithm: A distributed relaxation method for the assignment problem. *Annals of operations research*, 14(1):105–123, 1988.

[196] Roy Jonker and Ton Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. In *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR*, pages 622–622. Springer, 1988.

[197] NK Karmarkar and KG Ramakrishnan. Computational results of an interior point algorithm for large scale linear programming. *Mathematical Programming*, 52(1):555–586, 1991.

[198] KG Ramakrishnan, Narendra Karmarkar, and Anil P Kamath. An approximate dual projective algorithm for solving assignment problems. In *Network flows and matching*, pages 431–451. American Mathematical Society, 1991.

[199] Mustafa Akgül and Oya Ekin. A dual feasible forest algorithm for the linear assignment problem. *RAIRO-Operations Research*, 25(4):403–411, 1991.

[200] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. Society for Industrial and Applied Mathematics, 1984.

[201] Michael A Trick. A linear relaxation heuristic for the generalized assignment problem. *Naval Research Logistics (NRL)*, 39(2):137–151, 1992.

[202] Robert A Murphey, Panos M Pardalos, and Leonidas S Pitsoulis. A greedy randomized adaptive search procedure for the multitarget multisensor tracking problem. *Network design: Connectivity and facilities location*, 40:277–302, 1997.

[203] Amgad Naiem, Mohammed El-Beltagy, and P Ab. Deep greedy switching: A fast and simple approach for linear assignment problems. In *7th International Conference of Numerical Analysis and Applied Mathematics*, page 9999. AIP, 2009.

[204] Amgad Naiem and Mohammed El-Beltagy. On the optimality and speed of the deep greedy switching algorithm for linear assignment problems. In *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*, pages 1828–1837. IEEE, 2013.

[205] Haoran Sun, Xiangyi Chen, Qingjiang Shi, Mingyi Hong, Xiao Fu, and Nikos D Sidiropoulos. Learning to optimize: Training deep neural networks for wireless resource management. In *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–6. IEEE, 2017.

[206] Paul de Kerret, David Gesbert, and Maurizio Filippone. Team deep neural networks for interference channels. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2018.

[207] Haoran Sun, Xiangyi Chen, Qingjiang Shi, Mingyi Hong, Xiao Fu, and Nicholas D Sidiropoulos. Learning to optimize: Training deep neural networks for interference management. *IEEE Transactions on Signal Processing*, 66(20):5438–5453, 2018.

[208] Qingshan Liu and Yan Zhao. A one-layer projection neural network for linear assignment problem. In *2015 34th Chinese Control Conference (CCC)*, pages 3548–3552. IEEE, 2015.

[209] Mengyuan Lee, Yuanhao Xiong, Guanding Yu, and Geoffrey Ye Li. Deep neural networks for linear sum assignment problems. *IEEE Wireless Communications Letters*, 7(6):962–965, 2018.

[210] Nguyen Minh-Tuan and Yong-Hwa Kim. Bidirectional long short-term memory neural networks for linear sum assignment problems. *Applied Sciences*, 9(17):3470, 2019.

[211] Yihong Xu, Aljosa Osep, Yutong Ban, Radu Horaud, Laura Leal-Taixé, and Xavier Alameda-Pineda. How to train your deep multi-object tracker. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6787–6796. IEEE, 2020.

[212] Carlo Aironi, Samuele Cornell, and Stefano Squartini. Tackling the linear sum assignment problem with graph neural networks. In *Second Inter-*

national Conference, AII 2022, Reggio Calabria, Italy, September 1–3, 2022, pages 90–101. Springer Nature, 2022.

[213] Carlo Aironi, Samuele Cornell, and Stefano Squartini. A graph-based neural approach to linear sum assignment problems. *International Journal of Neural Systems*, 34(03):2450011, 2024. PMID: 38231046.

[214] Alex Grinman. The Hungarian algorithm for weighted bipartite graphs. Massachusetts Institute of Technology, 2015.

[215] Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4348–4355. International Joint Conferences on Artificial Intelligence Organization, 2021.

[216] Marcelo Prates, Pedro HC Avelar, Henrique Lemos, Luis C Lamb, and Moshe Y Vardi. Learning to solve np-complete problems: A graph neural network for decision tsp. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4731–4738. AAAI Press, 2019.

[217] Henrique Lemos, Marcelo Prates, Pedro Avelar, and Luis Lamb. Graph colouring meets deep learning: Effective graph neural network models for combinatorial problems. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 879–885. IEEE, 2019.

[218] K. Joshi Chaitanya, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. *ArXiv*, abs/1906.01227, 2019.

[219] Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

[220] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, page 13. Curran Associates, Inc., 2018.

[221] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR ’17. PMLR, 2017.

[222] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3835–3845. PMLR, 09–15 Jun 2019.

[223] M. Fey, J. E. Lenssen, C. Morris, J. Masci, and N. M. Kriege. Deep graph matching consensus. In *International Conference on Learning Representations (ICLR)*, pages 1220–1243. PMLR, 2020.

[224] Yunsheng Bai, Hao Ding, Ken Gu, Yizhou Sun, and Wei Wang. Learning-based efficient graph similarity computation via multi-scale convolutional set matching. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3219–3226, Apr. 2020.

[225] Alex Nowak, Soledad Villar, Afonso S Bandeira, and Joan Bruna. A note on learning algorithms for quadratic assignment with graph neural networks. *stat*, 1050:22, 2017.

[226] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5):12, oct 2019.

[227] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[228] Brian M. Clapper. Munkres implementation for python. `https://github.com/bmc/munkres`. Accessed: 2022-05-10.

[229] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv: Learning*, abs/1905.10947, 2019.

[230] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of*

the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'18/IAAI'18/EAAI'18, pages 3538–3545. AAAI Press, 2018.

[231] Hoang NT and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *ArXiv*, abs/1905.09550, 2019.

[232] Benjamin Völker, Andreas Reinhardt, Anthony Faustine, and Lucas Pereira. Watt's up at home? smart meter data analytics from a consumer-centric perspective. *Energies*, 14:719, 2021.

[233] François Lemercier, Guillaume Habault, Georgios Z. Papadopoulos, Patrick Maillé, Nicolas Montavont, and Periklis Chatzimisios. *Communication Architectures and Technologies for Advanced Smart Grid Services: Communication Networks and Services.* Wiley, 12 2018.

[234] Roy D Yates. The age of information in networks: Moments, distributions, and sampling. *IEEE Transactions on Information Theory*, 66(9):5712–5728, 2020.

[235] Martin Bøgsted, Rasmus L Olsen, and Hans-Peter Schwefel. Probabilistic models for access strategies to dynamic information elements. *Performance Evaluation*, 67(1):43–60, 2010.

[236] Mohammed S Kemal, Rasmus L Olsen, and Hans-Peter Schwefel. Information quality aware data collection for adaptive monitoring of distribution grids. In *The 1st EAI International Conference on Smart Grid Assisted Internet of Things*, pages 11–20. EAI, 2017.

[237] Mohammed S Kemal, Rasmus L Olsen, and Hans-Peter Schwefel. Optimized scheduling of smart meter data access: A parametric study. In *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–6. IEEE, 2018.

[238] Ahmed Shawky, R Olsen, Jens Pedersen, and H Schwefel. Network aware dynamic context subscription management. *Computer Networks*, 58:239–253, 2014.

[239] Kamal Jain, Jitendra Padhye, Venkata N Padmanabhan, and Lili Qiu. Impact of interference on multi-hop wireless network performance. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 66–80. Springer, 2003.

[240] Rene L Cruz and Arvind V Santhanam. Optimal routing, link scheduling and power control in multihop wireless networks. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, volume 1, pages 702–711. IEEE, 2003.

[241] Jihui Zhang, Haitao Wu, Qian Zhang, and Bo Li. Joint routing and scheduling in multi-radio multi-channel multi-hop wireless networks. In *2nd International Conference on Broadband Networks, 2005.*, pages 631–640. IEEE, 2005.

[242] Asma Farooq, Kamal Shahid, and Rasmus Lϕvenstein Olsen. Scheduling of smart meter data access using hungarian algorithm. In *2022 25th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pages 351–356. IEEE, 2022.

[243] Shalu Agrawal, Sunil Mani, Karthik Ganesan, and Abhishek Jain. *What Smart Meters Can Tell Us: Insights on Electricity Supply and Use in Mathura and Bareilly Households: Report.* Council on Energy, Environment and Water, 2020.

[244] Thilo von Neumann, Christoph Boeddeker, Keisuke Kinoshita, Marc Delcroix, and Reinhold Haeb-Umbach. Speeding up permutation invariant training for source separation. In *ITG Conference on Speech Communication*, pages 99–103. IEEE, 2021.

[245] Midia Yousefi, Soheil Khorram, and John Hansen. Probabilistic permutation invariant training for speech separation. In *Proceedings Interspeech 2021*, pages 4604–4608. IEEE, 09 2019.

[246] Shaked Dovrat, Eliya Nachmani, and Lior Wolf. Many-speakers single channel speech separation with optimal permutation training. In *Proceedings Interspeech 2021*, pages 3890–3894. IEEE, 2021.

[247] Anwesa Choudhuri, Girish Chowdhary, and Alexander G. Schwing. Assignment-space-based multi-object tracking and segmentation. In *2021*

IEEE/CVF International Conference on Computer Vision (ICCV), pages 13578–13587, 2021.

[248] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[249] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv*, 2017.

List of Publications

1. C. Aironi, S. Cornell, E. Principi and S. Squartini, “Graph-based Representation of Audio signals for Sound Event Classification”, 2021 29th European Signal Processing Conference (EUSIPCO), Dublin, Ireland, 2021, pp. 566-570, doi: 10.23919/EUSIPCO54536.2021.9616143.

2. C. Aironi, S. Cornell, E. Principi and S. Squartini, “Graph Node Embeddings for ontology-aware Sound Event Classification: an evaluation study”, 2022 30th European Signal Processing Conference (EUSIPCO), Belgrade, Serbia, 2022, pp. 414-418, doi: 10.23919/EUSIPCO55093.2022.9909608.

3. C. Aironi, S. Cornell and S. Squartini, “Tackling the linear sum assignment problem with graph neural networks”, Second International Conference, AII 2022, Reggio Calabria, Italy, September 1–3, 2022, (Springer Nature, 2022), pp. 90–101., doi: 10.1007/978-3-031-24801-6_7

4. C. Aironi, S. Cornell, L. Serafini and S. Squartini, “A Time-Frequency Generative Adversarial Based Method for Audio Packet Loss Concealment”, 2023 31st European Signal Processing Conference (EUSIPCO), Helsinki, Finland, 2023, pp. 121-125, doi: 10.23919/EUSIPCO58844.2023.10290027.

5. C. Aironi, S. Cornell, L. Gabrielli and S. Squartini, “A Score-aware Generative Approach for Music Signals Inpainting”, 2023 4th International Symposium on the Internet of Sounds, Pisa, Italy, 2023, pp. 1-7, doi: 10.1109/IEEECONF59510.2023.10335274.

6. C. Aironi, S. Cornell, and S. Squartini, “A graph-based neural approach to linear sum assignment problems”, International Journal of Neural Systems, volume 34, 2024, doi: 10.1142/S0129065724500114.

7. C. Aironi, L. Gabrielli, S. Cornell, and S. Squartini, “Complex-bin2bin: A Latency-Flexible Generative Neural Model for Audio Packet Loss Con-

cealment", IEEE Transactions on Audio, Speech, and Language Processing, (manuscript submitted for publication).