Computing Surveys

A Survey of Deep Learning Approaches for Soundscape Generation

Journal:	*Computing Surveys*
Manuscript ID	CSUR-2023-0601
Paper:	Long Survey Paper
Date Submitted by the Author:	26-Jul-2023
Complete List of Authors:	Duarte, Márcio; University of Porto Faculty of Engineering, Bernardes, Gilberto; University of Porto Faculty of Engineering Reis, Luís Paulo; University of Porto Faculty of Engineering
Computing Classification Systems:	Applied computing > Arts and humanities > Sound and music computing, Computing methodologies > Machine learning > Machine learning approaches > Neural networks
Note: The following files were submitted by the author for peer review, but cannot be converted to PDF. You must view these files (e.g. movies) online.	
ACM - An Overview of Deep Learning Approaches for Soundscapes Generation (Copy).zip	

A Survey of Deep Learning Approaches for Soundscape Generation

MÁRCIO DUARTE, Faculty of Engineering of the University of Porto, Portugal
GILBERTO BERNARDES, Faculty of Engineering of the University of Porto, Portugal
LUÍS PAULO REIS, Faculty of Engineering of the University of Porto, Portugal

Soundscape generation is the creation of realistic environments from a wide variety of audio sources. Creating soundscapes manually is tedious and requires expertise. Deep learning automates it using neural networks trained on soundscape data. This review examines recent advances, focusing on data and models. It analyses the strengths and limitations of the datasets. It examines deep learning architectures in use, including GANs, VAEs, text-to-sound and vocoders. Open problems include improving the authenticity and variety of generated soundscapes, integrating user preferences, and novel applications. Future research directions include user feedback, preferences, and new domains.

CCS Concepts: • **Applied computing** → *Sound and music computing*; • **Computing methodologies** → *Neural networks.*

Additional Key Words and Phrases: Soundscapes, Deep learning, Vocoders, Text-to-sound, Data augmentation, Automated feature extraction, Generative AI

ACM Reference Format:
Márcio Duarte, Gilberto Bernardes, and Luís Paulo Reis. 2023. A Survey of Deep Learning Approaches for Soundscape Generation. *ACM Comput. Surv.* 1, 1 (July 2023), 32 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Soundscapes refer to an acoustic environment that includes natural and human-made sounds, as perceived, experienced, and understood by individuals, in context [33, 68]. In other words, a soundscape encompasses the auditory milieu characterized by a collection of naturally occurring and human-generated sounds as perceived, encountered, and comprehended within a contextual framework by individuals. It is paramount in audio content creation, augmenting the user experience across media applications by infusing emotional engagement, a greater sense of immersion, and attention [9].

Traditionally, sound designers rely on manual labor to create soundscapes, which involve recording and editing real-world sounds, mixing, and adding sound effects [73]. Creating high-quality soundscapes is challenging, costly and time-consuming, requiring specialized skills and resources.

Authors' addresses: Márcio Duarte, Faculty of Engineering of the University of Porto, Department of Informatics Engineering, Porto, Portugal, up201909936@edu.fe.up.pt; Gilberto Bernardes, Faculty of Engineering of the University of Porto, Department of Informatics Engineering, Porto, Portugal, gba@fe.up.pt; Luís Paulo Reis, Faculty of Engineering of the University of Porto, Department of Informatics Engineering, Porto, Portugal, lpreis@fe.up.pt.

0360-0300/2023/7-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Hence, it engenders a notable impediment to the creation of soundscapes at scale [6, 74], namely in light of its growing popularity and consumption within podcasts, movies, and video games.[1]

To overcome the aforementioned limitations, algorithmic soundscape generation has emerged as a promising solution that streamlines soundscape creation. Preceding 2018, prevailing models for soundscape generation primarily revolved around statistical methods, featuring prominent employment of machine learning techniques with feature engineering. For a comprehensive overview of techniques employed before the era of deep learning, reference can be made to the review papers by Alias et al.[4] and Kalonaris et al.[36]. Noteworthy efforts at the feature engineering level are exemplified by Fernandez et al. [17], who represent sounds as high-level features, such as musical sheets, as an approach to generating musical sounds.

State-of-the-art models for soundscape generation feature deep learning techniques, greatly improving the quality, diversity, and scale of generative soundscape systems. These models are end-to-end systems that can transform textual soundscape descriptions into audible signals, generating realistic and high-quality soundscapes that can be customized according to the user's preferences and needs. Furthermore, these models can also adapt to the user's feedback and behavior, providing a more personalized and interactive experience.

To pursue these models, it becomes imperative to establish a robust groundwork in generative algorithms on media, particularly audio, while concurrently fostering a comprehensive understanding of natural language processing (NLP). As a specialized field within artificial intelligence, NLP facilitates the analysis and generation of natural language, enabling AI models to grasp the semantic and pragmatic nuances inherent in textual inputs and generate sounds accordingly. This approach holds considerable potential in reducing the time and effort required to create soundscapes, thus rendering it more accessible for content creators of varying scales to integrate superior audio components into their ventures.

1.1 Soundscape Generation: Feature Engineering Methods

Feature engineering methods for soundscape generation typically adopt a threefold strategy to resynthesize (and extend) a short soundscape recording provided by the user: 1) segmentation, 2) feature extraction and modeling, and 3) resynthesis of a given environmental sound. Statistical models adopting stochastic processes or pattern recognition methods were commonly applied to model and recreate a given soundscape recording with a degree of variation while maintaining its structure. Generated soundscapes relied on the similarity among audio segments to create smooth transitions [29].

Salamon's Scaper [67] was the most widely referenced work on soundscape synthesis in the literature. Scaper is a modular software library that facilitates the creation of synthetic sound environments using basic sound-generating objects, or "sound sources". These sources can represent simple sounds, such as bird songs, human speech, or car horns, and complex sounds, such as those produced by crowds or construction sites. Users can define the attributes of each sound source, such as its location, volume, and duration, and manipulate these parameters dynamically to create soundscapes that vary over time.

One of the main features of Scaper is its capability to generate synthetic soundscapes that are diverse and statistically representative of real-world environments. The library employs various sound-generating algorithms that can produce randomized yet realistic sounds to achieve this. For

[1]Consumer data reported in 2021 showed compelling evidence regarding the listening habits of individuals within the United States of America. The findings indicate a substantial growth in podcast listenership over the past decade, with 41% of Americans aged 12 or older having engaged with podcasts in the preceding month and 28% within the last week. Moreover, at the beginning of the same year, a notable 68% of Americans aged 12 and above had indulged in online audio consumption within the previous month, while 62% had done so within the preceding week [61]

example, it can generate sounds that resemble real-world sources but with variations in volume, pitch, and timbre to avoid repetition and increase diversity.

Additionally, Finney and Janer propose a different approach to conventional soundscape synthesis [18], which uses concatenative synthesis to construct a sound environment using sonic material provided by online communities. Concatenative synthesis is a technique that combines segments of audio samples based on their acoustic similarity and temporal continuity, as discussed in Section 3.1.6.

These examples suggest that conventional soundscape synthesis methods involved modifying existing soundscapes, similar to sound augmentation techniques used in data augmentation within deep learning methods (as discussed in Section 3.1). These systems presented many limitations. Notably, the confined processing of simple (static or redundant) sound textures barely addresses environmental sounds with complex changes over time. While this approach provides good results for highly redundant audio content, it does not provide an optimal answer to the problem, especially when processing complex with high temporal dependencies, such as moving vehicles and storms. Defining optimal features adapted to each process's soundscape is time-consuming and complex, requiring adaptations to the models according to the input soundscape.

1.2 Soundscape Generation: Feature Learning Methods

Deep learning generative models have become the state-of-the-art approach in algorithmic soundscape generation. These models learn sound features through latent spaces and use this knowledge representation to generate novel samples by inferring distinctive sets of features. This topic has retained remarkable attention within the research community. The field is characterized by rapid and dynamic evolution, encompassing specialized generative soundscape models and broader advancements in deep learning. Consequently, review articles serve as valuable resources for establishing the foundational knowledge within this study area.

Prior review articles exploring deep learning models and feature learning to provide a global and critical perspective of existing technologies for soundscape generation [34, 37, 32]. Several components of such systems have been isolated and addressed in dedicated surveys, such as soundscape datasets, audio augmentation techniques, feature learning, embeddings, and system evaluation.

Access to large-quality data is fundamental to the success of deep learning methods. In this context, many surveys in the literature have been dedicated to reviewing existing soundscape datasets on state-of-the-art models [69, 80, 5, 48]. The conclusions underscore the imperative for augmenting the existing datasets in this field, thereby necessitating greater diversity and representation. To maximize the existing audio datasets and their effectiveness in existing models, audio augmentation techniques have been explored in the literature [63, 48, 32]. However, there is still a need for more research on the effectiveness and generalizability of these techniques.

The feature learning and embeddings from the soundscape data are pivotal to these methods as they drive the generation of new soundscapes as reviewed in [8, 45, 5]. Evaluation methods of these models in light of the existing datasets are key to establishing a comparison across models and identifying areas for future endeavour [27, 37, 32]. The need for more standardized evaluation metrics and benchmarks in this field is commonly highlighted.

1.3 Problem Definition and Contribution

This article comprehensively overviews the current state-of-the-art soundscapes generation utilizing deep learning methods. The existing approaches to generating soundscapes using deep learning methods can be split into three major components, which we survey at length:

(1) Data: This component covers the soundscape datasets used to train and evaluate the deep learning models, the audio representations used to encode and decode the sound signals, and the data augmentation techniques used to enhance and diversify the audio data.
(2) Model: This component covers the deep learning model architectures used to learn and generate audio. It also covers the latent spaces and feature embeddings used to represent and manipulate the audio.
(3) Generation: This component covers the sampling methods for generating new audio samples from the learned feature distributions.

This article reviews existing soundscape datasets and sound augmentation methods to cover the above components of a generative deep learning system for soundscape creation. It also discusses the digital audio representations that can be encoded into compact and meaningful feature vectors using stochastic methods based on deep learning techniques.

A review of general deep learning architectures for generative purposes is conducted, detailing Generative Adversarial Networks (GANs) [24] and Variational Autoencoders (VAEs) [41]. Moreover, a comprehensive evaluation of general deep learning architectures utilized for generative purposes is undertaken, focusing on three prominent approaches: unsupervised, vocoder-based, and end-to-end. Within the unsupervised domain, notable examples such as WaveGAN [14] and SoundStream [82] are examined, which employ techniques to learn the latent distribution of unlabeled data and generate soundscapes without direct input. In vocoder-based methodologies, prominent models such as HiFi-GAN [42] and GANSynth [16] are considered, wherein spectrograms serve as input representations and sound signals are generated as output. The end-to-end paradigm is also explored, where systems such as AudioGen [43] and Riffusion [21] are regarded, which leverage natural language descriptions and other intricate forms of information as input representations, enabling the generation of sound signals as output.

An important focus of this review article, and its main novelty to related review articles, is the narrow domain of soundscape generation within the general domain of music and audio signal generative deep learning models, with a particular emphasis on the recent cutting-edge end-to-end generative soundscape models conditioned by textual inputs. In other words, soundscape generative models from textual descriptions across a timeline. By synthesizing and evaluating the current state-of-the-art soundscape generation with deep learning, we identify key challenges for future work within this domain.

The remainder of this article is structured as follows. Section 2 provides an overview of digital sound representation and the datasets of interest for soundscape generation. In Section 3, the most commonly employed techniques for audio and text augmentation are displayed. In Section 4, various generative deep learning architectures, including GANs and VAEs, will be explored. Section 5 will delve into different sound generation models, including WaveGAN, WaveNet, and MelGAN. Finally, in Section 6, the conclusions will be presented, highlighting potential avenues for future research in this field.

2 DIGITAL SOUND REPRESENTATION AND DATASETS

Two key representations of sound are commonly used in soundscape generation. One is the time-domain array representation, where sound is represented as a one-dimensional array of amplitude values sampled at a given frequency over time. The other is the frequency-domain representation, which can be obtained from the time-domain array using transforms like the discrete Fourier Transform (DFT) or the constant-Q transform. A common frequency-domain representation is the spectrogram, which shows how the frequency content of a signal changes over time. Since sound contents change over time, a short-time Fourier transform (STFT) is often used to compute the DFT over successive time frames and generate the spectrogram [71].

Understanding existing sound representations is of utmost importance in audio content generation. However, using deep learning techniques necessitates the availability of extensive datasets. This requirement is particularly pronounced in the domain of soundscape generation, where the utilization of generative deep-learning models remains largely unexplored. Consequently, significant efforts must be invested in accessing high-quality, large-scale datasets to advance research in this area.

2.1 Datasets

This section comprehensively reviews the most pertinent datasets applicable to generating soundscapes using textual prompts. These datasets are selected based on specific criteria, including datasets comprising more than 1000 sound samples. This criterion is of particular significance due to dataset size's essential role in facilitating the optimal functioning of deep learning algorithms. The survey scope covers datasets containing mostly diverse natural sounds as the most representative of a soundscape. In adherence to this work's scope, datasets specific to music or speech have been excluded from the review.

Two types of datasets are listed. Their main difference relies on the type of label (i.e., the textual description) adopted. We distinguish two labels: *categorical* and *descriptive*. Datasets featuring categorical labels are composed of sounds associated with a unique label, such as "music", "piano", or "singing." Datasets featuring descriptive labels include natural language sentences or descriptions of the soundscape, such as "boy singing while playing the piano." In the context of deep generative learning, descriptive labeled datasets are more suitable. However, datasets with discrete labels can also be beneficial when augmented. To our knowledge, Table 1 lists all soundscape datasets to date that fulfill our criteria.

The availability and quality of these datasets pose a significant challenge in soundscape generation. Unlike other domains, such as computer vision or natural language processing, where datasets can comprise hundreds of millions of entries, sound datasets are relatively scarce and limited in size. Even the largest dataset listed in Table 1, AudioSet, consists of only approximately 2.1 million sound segments, and it is not specifically tailored for soundscape generation. Moreover, most datasets in this context contain categorical labels, requiring additional engineering efforts to transform them into descriptive labels suitable for generating soundscapes. This transformation process may introduce noise or ambiguity in the labels, potentially compromising the effectiveness and diversity of the generated models. Consequently, there exists a pressing need for more extensive and diverse datasets that feature descriptive labels tailored explicitly for the generation of soundscapes.

3 DATA AUGMENTATION

Data-driven deep learning models often require large amounts of data to perform well. However, collecting and annotating data can be expensive and time-consuming, especially for specialized domains like soundscapes. Data augmentation refers to techniques for artificially expanding training sets by creating modified versions of existing data samples.

In this section, we discuss two main types of data augmentation for text-to-soundscape generation: acoustic and linguistic. Data augmentation can reduce overfitting and improve generalization by introducing more variations into the training set [57].

Table 1. Comparison of datasets for soundscapes

Name	Type	Description	# Samples	Duration	Labels
Acoustic Event Dataset [76]	Categorical labeled	The classes have rather specific names, such as "hammer" or "mouse_click".	5223	Average 8.8s	One of 28 labels
AudioSet [22]	Categorical labeled	The dataset is extensive, featuring 10-second audio clips from diverse YouTube content, including speech, animals, instruments, and environments. Sounds are annotated using a hierarchical ontology.	2084320	Average 10s	One or more of 527 labels
Clotho [15]	Descriptive labeled	A practical example is a sound described as "a car honks from the midst of a rainstorm" and similar variations.	4981	15 to 30s	8 to 20 words/caption. 5 captions/audio
FSDKaggle2018 [20]	Categorical labeled	Audio dataset from Freesound, following AudioSet's ontology.	11073	From 300ms to 30s	One or more of 41 labels
AudioCaps [39]	Descriptive labeled	Crowdsourced human-written text pairs on AudioSet dataset.	39597	10s each	9 words/caption
UrbanSound8K [66]	Categorical labeled	Urban sound dataset with short 4-second audio clips from various cities, captured in different seasons and weather conditions. Clips sourced from YouTube and Freesound, annotated with single class labels for urban environment description.	8732	Less or equal to 4s	One of 10 labels
YouTube-8M Segments [2]	Categorical labeled	High-quality video dataset for video research, with human-verified segment annotations that temporally localize entities. Includes time-localized frame-level features for audio analysis from YouTube videos.	237000	5s	One or more of 1000 labels

3.1 Acoustic Data Augmentation

Data augmentation is imperative for boosting the performance of sound-based machine learning models as it enlarges the training dataset and fortifies the models' robustness. According to Abayomi-Alli et al. [1], commonly leveraged data augmentation techniques for audio tasks include additive noise injection, time-shifting, pitch shifting, GAN-based methods (see Section 4.3), time stretching and concatenation. While sound overlapping is not a prevalent technique, it facilitates the generation of verisimilar soundscapes with multiple concurrent sources. Investigating this technique, alongside the methods mentioned above, can thus augment training datasets and furnish the models with diverse input data. The subsequent subsections provide an exhaustive analysis of these techniques and their impact on the performance of sound-based machine learning models.

3.1.1 Addition of Noise. Data augmentation for audio through the addition of noise is a widely used technique that involves the generation of new and diverse audio samples by introducing random noise to the original signal [49]. Adding noise to audio signals consists of several steps, such as selecting a noise source, specifying the noise level, combining the noise and audio signals element-wise, and normalizing the output to prevent clipping.

The selection of noise sources should be based on the relevance and realism of the problem at hand. They can be any digitally generated or recorded noise from a real-world source, such as white noise, babble noise, static noise, factory noise, jet cockpit, shouting, and background noise [49]. For example, if the problem is recognizing speech in noisy environments, the noise level can be determined by multiplying a certain percentage of white noise with the original sound signal. This can be expressed mathematically as $y' = y + 0.05 \times Wn$, where y represents the original sound, y' the augmented sound, and Wn some white noise [49].

The resulting audio signal with added noise is then normalized[2] to prevent clipping or overloading. These steps can be repeated with different noise sources and noise levels to generate multiple and diverse audio samples that can be used for data augmentation purposes [49].

3.1.2 Time Shifting. The time-shifting technique, also called time warping, involves the manipulation of the temporal structure of an audio signal for data augmentation purposes. Specifically, time shifting entails the complete displacement of an audio signal by a designated time interval, either forward or backward, through adding or removing audio samples or repositioning existing samples within the signal.

One potential implementation of time shifting involves truncating the audio signal's length, followed by using the trimmed segments to create novel and varied audio samples. For instance, an audio signal with 150 samples may be truncated to 125, thereby allowing for the generation of up to 25 new audio samples by shifting the trimmed sections. These new samples may be labeled identically to the original audio signal.

The application of time shifting through the trimming and shifting of audio signal segments can significantly influence the performance of machine learning models that rely on sound-based data. By presenting the model with various time-shifted iterations of the audio signal, this technique can facilitate the model's acquisition of sound patterns that remain invariant to temporal changes, such as the presence of a specific sound event or the spoken words in an audio recording. This can result in superior generalization performance on the novel unseen data and enhanced overall model performance.

3.1.3 Pitch Shifting. Pitch shifting is an audio data augmentation technique that modifies an audio signal's frequency. This technique is implemented by altering the audio signal to a higher or lower pitch, resulting in audio signals with different pitch characteristics. Pitch shifting is beneficial for training machine learning models to identify sound patterns invariant to pitch changes.

3.1.4 GAN Based Methods. Utilizing Generative Adversarial Networks (GANs) for data augmentation in audio signals can be a powerful and efficient method, albeit comparatively slower than

[2]Normalization is the process of scaling the audio signal's amplitude (typically an amplitude peak) to the $[-1, 1]$ range. This can be done by dividing each audio signal sample by the maximum absolute value of the signal. Mathematically, this can be expressed as $z = \frac{y'}{\max(|y'|)}$, where z represents the normalized sound, y' the augmented sound, and $\max(|y'|)$ the maximum absolute value of the augmented sound.

alternative techniques. This approach involves training a GAN network on available audio data to discern the underlying patterns and distributions present within the data. The network subsequently generates new, synthetic audio signals that resemble the input data [58].

The efficacy of GAN-based data augmentation is heavily reliant on the quality of the GAN training, as well as the diversity of the input data. When the GAN is well-trained and the input data is diverse, the generated data is of high quality.

Recognizing the significant computational resources and training time required by GANs in comparison to other data augmentation techniques is imperative. However, it is noteworthy that GAN-based augmentation can yield highly effective and precise results, thereby considered a valuable addition to the data augmentation toolkit for audio-based machine learning tasks. Despite the computational demands, the outcomes achieved through GAN-based augmentation justify its inclusion in the repertoire of techniques employed to enhance the performance and robustness of audio-based machine learning models.

3.1.5 Time Stretching. Time stretching is a data augmentation technique for audio signals that involves changing the duration of the signals, usually by increasing or decreasing their time axis. Time stretching aims to generate new audio samples from the original signals with varying durations.

A common approach to implementing time stretching is using a stretching factor. For instance, a stretching factor of 1.2 increases the duration of the audio signal, i.e., the time axis of the audio signal, by 20%. One method involves using a simple algorithm that duplicates some of the samples in the audio signal based on the stretching factor. However, this basic approach may result in unwanted artifacts, such as pitch changes, if the stretching factor is not an integer.

More advanced techniques, such as phase vocoder-based time stretching[3], can produce high-quality time stretching with minimal artifacts. These techniques use time and frequency domain processing to stretch the audio signal while preserving its spectral content and temporal structure. The resulting audio signal has a different time duration while maintaining the original pitch.

3.1.6 Sound Concatenation. Sound concatenation, also known as mixing up sounds, is a data augmentation method for audio signals that involves merging multiple audio signals to produce a novel and mixed audio signal. This technique can be implemented by extracting fragments from numerous audio signals and randomly concatenating them or using cross-fade techniques to ensure a smooth transition between the different audio fragments.

Sound concatenation can be particularly useful in a soundscape generation context where one seeks to train a network to recognize a prompt such as "dog barking followed by car honking". It is important to note that when applying sound concatenation, the label for the resulting audio signal will also change and must be appropriately updated.

3.1.7 Sound Overlapping. Sound overlapping, also called sound mixing or audio blending, is a data augmentation technique for audio signals that involves the fusion of two or more audio signals to create a novel composite audio signal. This approach can aid in recognizing sound patterns in the presence of multiple simultaneous sounds, a common occurrence in real-world applications.

[3]The phase vocoder is a digital signal processing algorithm in the class of analysis-synthesis methods. It operates on an input sound signal and produces an output sound signal that is either identical or altered in some aspect. The alteration can involve the temporal or spectral characteristics of the sound, such as duration, pitch, or timbre. The phase vocoder works as follows: it segments the input sound signal into short overlapping frames using a windowing function; then, it applies a discrete Fourier transform (DFT) to each frame to obtain a complex-valued representation of the amplitude and phase of each frequency bin; next, it modifies the amplitude or phase of each frequency bin according to the desired transformation; after that, it applies an inverse discrete Fourier transform (IDFT) to each frame to reconstruct the time-domain signal; and finally, it combines the output frames using an overlap-add technique to form the output sound signal [19].

The process of sound overlapping can be executed via the following four steps:

- Selection of multiple audio signals: Choose two or more audio signals that require a combination. These signals may be from the same or different sources, depending on the desired outcome and the problem.
- Adjustment of the amplitude of each audio signal: Balancing the amplitude levels of the audio signals combined is crucial to avoid one signal overpowering the others. This can be achieved by normalizing the amplitude of each signal or scaling them based on a predetermined factor.
- Mixing of the audio signals: Combine the chosen audio signals by adding them element-wise. This process generates a new, composite audio signal containing overlapping sounds from the original signals.
- Normalization of the output: Normalize the resulting composite audio signal to prevent clipping or overloading.

Repeating these steps with different combinations of audio signals and amplitude adjustments can generate diverse composite audio samples for data augmentation.

It is crucial to note that when utilizing sound overlapping as a data augmentation technique, the labels associated with the original audio signals must also be considered. In some cases, the labels may need to be combined or modified to represent the new composite audio signal accurately.

3.2 Linguistic Data Augmentation

Linguistic data transfiguration involves metamorphosing textual data to augment its diversity and quantity. This can ameliorate the performance and robustness of natural language processing models that rely on text data.

For sonic milieu generation, linguistic data transfiguration provides an efficacious approach to creating more varied and verisimilitudinous soundscape descriptions from text. Nonetheless, various existing soundscape datasets discussed in Section 2.1 contain categorical labels or tags instead of descriptive annotations.

It is imperative to note that some linguistic data augmentation techniques only function when the original text is in natural language, while others can still be applied to categorical labels.

Variations in input text can facilitate the generation of soundscapes with more variability and legitimacy. The following sections cover specific linguistic data augmentation techniques and their applications for soundscape generation.

While linguistic data augmentation has several advantages, it poses some issues and challenges [70].

The main benefits of textual augmentation are as follows:

(1) It can reduce the costs of collecting and annotating textual data.
(2) It can improve the accuracy of models by increasing the training data size, alleviating data scarcity, mitigating overfitting, and creating variability in the data.
(3) It can boost the generalizability of models by exposing them to different linguistic patterns and styles.
(4) It can increase the robustness of our models by making them resilient against adversarial attacks that attempt to deceive them through expert alterations of the input sequences.

However, there are also some potential downsides:

(1) It can introduce noise or errors into the data that may impact the quality and readability of the augmented text.
(2) It can change or lose the original text's meaning, style, or complexity, mainly if the transformation is inappropriate or irrelevant to the task or domain.

(3) It can be computationally expensive or time-consuming to generate high-quality and diverse augmented text, especially if it involves using external resources or models such as dictionaries, corpora, word embeddings, generative models, or translation models.

Various frameworks have been developed for augmenting text data linguistically. These can be broadly grouped into symbolic and neural augmentation models.

3.2.1 Symbolic Augmentation Models. These methods employ rule-based transformations operating directly on the surface form of text via predetermined heuristics. They include:

- Rule-based augmentation replaces, inserts, or deletes tokens according to specified rules. An example is replacing named entities with alternatives [78].
- Graph-based augmentation uses graph structures to perturb the text, *e.g.* swapping adjacent adjectives and nouns [3].
- MixUp combines existing examples via interpolation to synthesize augmented instances, *e.g.* combining "The cat sat on the mat" and "The dog lay on the rug" to generate "The cat lay on the mat" [26].
- Feature-based augmentation applies transformations to word embeddings, *e.g.* adding noise to the embedding space [10].

Despite their interpretability, symbolic methods struggle with complex transformations.

3.2.2 Neural Augmentation Models. These techniques leverage deep neural networks and large language models. They encompass:

- Back-translation, which translates text into another language and back, producing paraphrases, *e.g.* translating "The book was interesting." to French and back to English, yielding "The book was fascinating" [55].
- Generative augmentation employs generative language models to synthesize novel text, *e.g.* using an LLM such as GPT to rephrase sentences or simply fine-tune them.

While more complex, neural methods can generate diverse and realistic augmented instances.

Both symbolic and neural augmentation aim to expose models to more variability during training, helping combat overfitting and improve performance. However, symbolic methods offer interpretability, while neural methods provide more flexibility and variation.

Table 2. A taxonomy of text augmentation methods for transformer language models according to their algorithmic properties and underlying approaches.

	Interpretability Transparency	**Algorithmic Complexity**	**Capacity to Leverage Labels**	**Paradigmatic**
Rule-based	High	Relatively low	Incapable	Lexical substitution
Graph-based	High	Moderate	Incapable	Knowledge graph-based
Sample Combination	Medium	Moderate	Capable	Embeddings summation
Feature-based	Medium	Moderate	Incapable	Noise injection
Generative	Low	High	Capable	Text auto-generation
Back-translation	Medium	High	Capable	Inverse translation

Table 2 categorizes several common text augmentation strategies according to their transparency, complexity, dependence on labels, and paradigm.

Regarding interpretability, rule-based and graph-based methods exhibit high transparency since they employ explicit symbolic transformations. In contrast, the stochastic nature of generative models and back-translation compromises their interpretability.

Computational complexity also differs. Rule-based and graph-based augmentation are relatively efficient since they apply explicit symbolic transformations. In comparison, training neural networks for generative modeling and back-translation requires more computational resources.

For leveraging labels, given that some datasets mentioned in Section 2.1 contain categorical labels, and we desire descriptive annotations, it is pertinent to assess whether these techniques can transform categorical labels into text descriptions. Employing generative models or back-translation potentially enables this since the models attempt to make sense of the input and transform it into a sentence.

4 GENERATIVE DEEP LEARNING ARCHITECTURES

Generative deep learning architectures establish blueprints for developing deep learning networks that synthesize diverse and novel data samples according to a learned distribution. These architectures entail creating latent data constructs and learning to emulate the fundamental statistical patterns found in observed data. In order to learn such latent spaces, the training phase of a generative model mandates the selection of optimized parameters minimizing a chosen measure of distance, loss, or error between the model and the actual distribution.

Generative deep neural models have been applied to tasks comprising image synthesis, text generation, and audio synthesis. Their popularity has recently surged owing to their remarkable ability to generate high-quality data and effectively model complex distributions. In the following sections, we outline the most ordinarily used generative deep neural architectures, presented chronologically, as summarized in Table 3.

4.1 Deep Autoregressive Network (DARN)

DARN is an architecture for generative models that uses an autoregressive (AR) method to generate data [25]. AR models generate novel data by making predictions of the subsequent term in a sequence, relying on the information provided by the preceding terms. The idea was proposed in 2013 to model the data's complex, high-dimensional probability distribution by breaking it down into simple, conditionally independent distributions. This is accomplished by training a deep neural network that maps inputs to outputs through a series of hidden layers. This allows the network to build a complex representation of the data distribution over time, capturing complex patterns in the data and ultimately making more precise predictions. Fig. 1 shows an example of a DARN architecture, where each neuron in a layer is conditioned on the output of two neurons from the previous layer.

In greater detail, AR models define a tractable density model by decomposing the probability distribution over n time steps via the chain rule of probability, such that:

$$p(Y) = \prod_{i=1}^{K} p(y_k|y_1, ..., y_{i-1}) \tag{1}$$

Equation 1 represents the decomposition of the probability distribution over K time steps in the autoregressive approach. It assumes a canonical sequential order — the current term in the sequence (y_k) is only conditioned by a window of previous terms. Future terms are not taken into account. Ultimately, this implies that each data point solely depends on the preceding ones, and

Table 3. Comparison of Generative Deep Learning Architectures

Model	Year	Type	Key Characteristics	Inference
DARN	2013	Autoregressive	Uses a single model to predict the probability distribution of each output token conditioned on the previous tokens	Sequential
VAE	2013	Variational Autoencoder	Learns a latent representation of the input data and generates new samples by sampling from the learned latent space	Parallel
GAN	2014	Generative Adversarial Network	Consists of a generator and a discriminator that compete in a two-player minimax game to generate realistic samples	Parallel
Normalizing Flows	2015	Flow-based models	Transforms a simple probability distribution into a complex one by applying a sequence of invertible transformations	Parallel
Diffusion	2015	Flow-based models	Uses a diffusion process to model the probability distribution of the data	Parallel
Transformers	2017	Attention-based models	Uses self-attention to capture global dependencies and generate sequences	Sequential
VQ-VAE	2018	Vector Quantized Variational Autoencoder	Discretizes the continuous latent space by mapping each latent vector to the closest codebook vector	Parallel

the model acquires the ability to predict the subsequent term based solely on the immediately preceding context. The rationale behind this technique is similar to the one that recurrent neural networks (RNNs)[4] employ. Indeed, an RNN can be cast as an AR model that compresses the prior terms into a hidden state instead of providing them explicitly as input to a layer [31].

While DARNs can model complex distributions and capture sophisticated patterns in data, they also suffer from some limitations. First, DARNs have difficulties in capturing longer-term features due to the limited receptive field at each time step. Since each prediction is only conditioned on a window of previous steps, the network can lose information about distant dependencies in the data. This issue is known as the long-range dependency problem.

Second, DARNs generate data inherently sequentially, producing one step at a time based on previous predictions. This sequential generation process can be slow, especially for models with long sequences. The generation speed depends on the size of the receptive field and the number of time steps, limiting the scalability of DARNs to very high-dimensional data.

4.2 Variational Autoencoder (VAE)

Kingma and Welling introduced the Variational Autoencoders (VAEs) in 2013 [41]. VAEs differ from traditional autoencoders by utilizing a variational inference method to model complex distributions in high-dimensional spaces, generating new, unseen data similar to the initial training.

[4]Recurrent neural networks (RNNs) are a neural network architecture that can process sequential data, such as text, speech, or audio. RNNs have a recurrent structure that allows them to maintain a hidden state that encodes the information from previous terms. At each time step, the RNN takes the current term and the previous hidden state as inputs, updates the hidden state, and produces an output. RNNs can learn long-term dependencies and capture complex patterns in sequential data. However, they also suffer from drawbacks, such as vanishing or exploding gradients, difficulty in parallelization, and limited memory capacity [23].

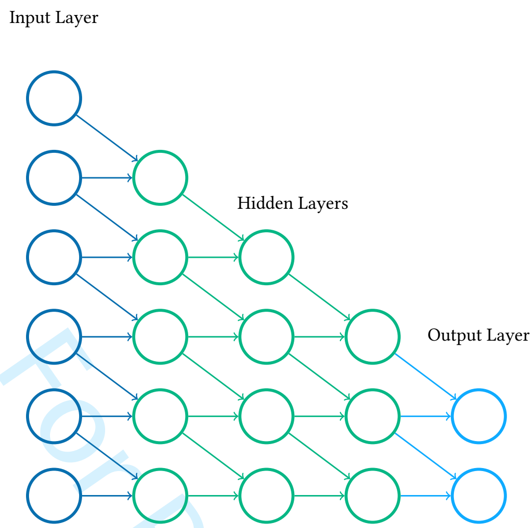

Fig. 1. **Deep autoregressive network (DARN)** — This network consists of five input neurons, three hidden layers, and two output neurons. The network is autoregressive, meaning that the input of each neuron in a layer depends on the output of a subset of neurons from the previous layer. In this case, each neuron is conditioned on the output of two neurons from the previous layer, as shown by the arrows in the figure.

In traditional autoencoders, the input undergoes a mapping process by the encoder, and subsequently, the decoder reconstructs it into a sample that closely emulates the original input. Conversely, Variational Autoencoders (VAEs) employ a distinct methodology by encoding the input data as a probability distribution across a latent space. This latent space encapsulates a continuum of potential feature variations pertaining to the input data.

During the training phase, the model acquires knowledge of this lower-dimensional representation. Subsequently, the decoder draws samples from this distribution to generate output, thereby reconstructing the input data based on the sampled latent vector. The decoder can generate fresh and varied samples within the continuous latent space by relinquishing the encoder.

An illustrative depiction of a VAE architecture can be observed in Fig. 2, where the encoder produces two vectors representing the mean and standard deviation of a normal distribution. The decoder then samples a latent vector from this distribution and reconstructs the input data accordingly.

VAEs generate smooth transitions between data points, as distributions close to each other generate similar outputs. An objective function trains these networks to minimize the loss between input and output and ensure that the learned distribution is similar to a prior distribution, such as a Gaussian.

However, VAEs can suffer from posterior collapse, where the encoder fails to capture the variability in the input data, and the decoder generates outputs that are not diverse. This can transpire for multiple reasons, such as a high reconstruction loss weight or a small latent space size. Posterior collapse can severely impact the performance of the VAE and result in poor-quality generated samples.

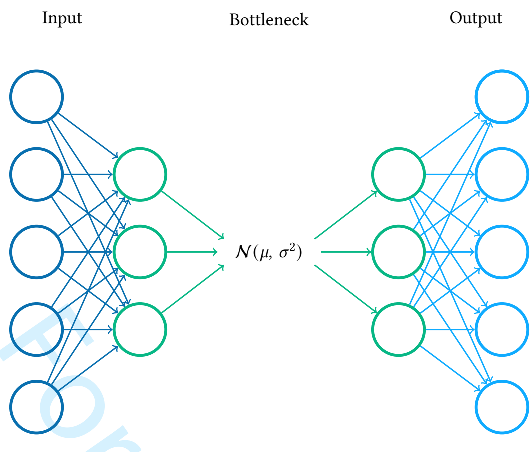

Fig. 2. **Variational autoencoder (VAE)** — Very similar to the traditional autoencoder, but unlike it, which uses a deterministic mapping from the input to the latent space, a VAE uses a probabilistic mapping that produces a distribution over the latent space. In this case, the encoder outputs two vectors: one for the mean and one for the standard deviation of a normal distribution. The decoder then samples a latent vector from this distribution and reconstructs its input data. This way, the VAE can generate new data by sampling from the latent space.

4.3 Generative Adversarial Network (GAN)

The Generative Adversarial Network (GAN) was first introduced in 2014 by Goodfellow et al. [24]. It is a novel approach to generative modeling using deep neural networks. GANs train two neural networks simultaneously: a generator network G and a discriminator network D.

The generator network G transforms a random noise vector z into a target distribution from a data space $\hat{X}$. The discriminator network D distinguishes between synthetic and real data by mapping the input data, be it X or $\hat{X}$, to a categorical label according to whether it thinks the input came from the actual data distribution $p(X)$ or the model distribution $p(z)$. An example of a GAN architecture is shown in Fig. 3, where the generator takes a random noise vector as input and produces a synthetic sample. The discriminator takes the synthetic and real samples as input and tries to classify them as fake or real.

The networks are trained using a minimax optimization framework, where G creates a synthetic sample $\hat{X}$ passed in conjunction with a real one X to D. D is trained to maximize the probability of distinguishing the real from the synthesized data, while G is trained to minimize $\log(1 - D(G(z)))$, attempting to fool D. The models are trained until a Nash equilibrium is reached, i.e., when G produces synthetic data indistinguishable from real data.

After training, the generator is used to sample from the learned distribution of the actual data, mapping random vectors to data samples in the target domain. While GANs have seen success in producing high-resolution images, they still need improvement in the audio domain. Training two different networks can also make the GAN framework unstable, leading to sub-optimal Nash equilibria and mode collapse.

Due to the challenges faced by DARNs in capturing longer-term features and their inherent slowness in generating waveforms sequentially, the GAN architecture emerges as a compelling

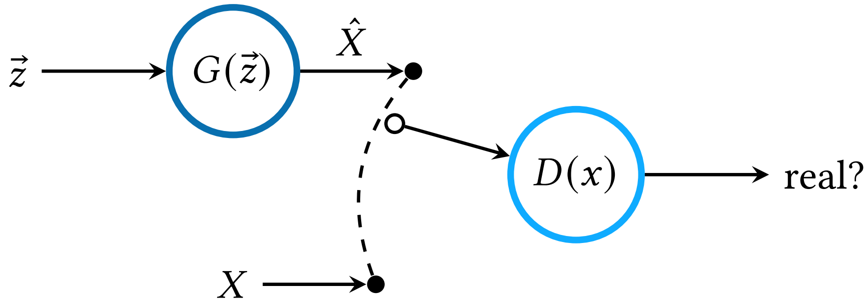

Fig. 3. **Generative adversarial network (GAN)** — A random noise vector $\vec{z}$ is passed through the generator in $G(\vec{z})$ to create the synthetic sample $\hat{X}$. Both this and the real sample X are passed to the discriminator D that predicts which of the samples is real. The two networks are trained in an adversarial manner, where the generator tries to fool the discriminator by generating realistic samples, and the discriminator tries to identify them correctly.

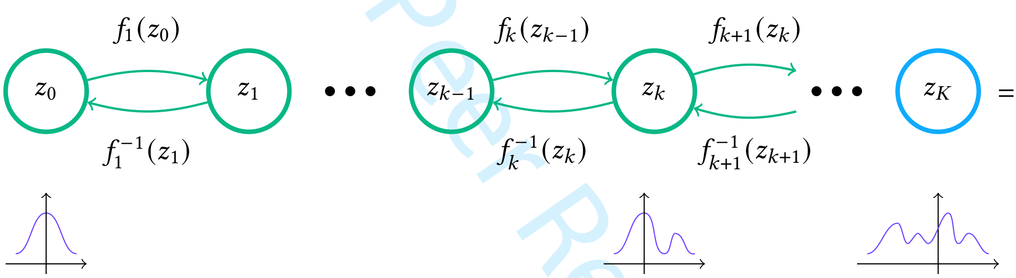

Fig. 4. **Normalizing flows network** — This illustration was based on [79] and shows the application of multiple invertible functions f_k composed one after the other in order to build the complex output $z_K = Y$ from a simple Gaussian distribution.

alternative. GANs possess the capability to model global latent structure and generate audio at a faster pace, thereby exhibiting promising prospects for audio generation.

4.4 Normalizing Flow Models

Normalizing flow models provide a flexible and robust framework for generative modeling and were first introduced in 2015 [62].

The key idea is to map simple distributions to more complex ones using the change of variables in the probability distributions technique. This technique involves applying a transformation to a distribution that transforms it into another, more complex, distribution. The whole concept starts with a simple distribution (e.g., Gaussian) for a set of latent variables z. The aim is to transform this distribution to represent an output Y. A single transformation is given by a smooth and invertible function f that can map between z and Y, such that $Y = f(z)$ and $z = f^{-1}(Y)$. Depending on the complexity of Y, one of these transformations may not yield an optimal distribution. Therefore, multiple invertible transformations are composed one after the other, constructing a "flow". Neural network layers parameterize each mapping function in the flow [31]. This process can be seen in Fig. 4.

Accurately, let z_0 be a multivariate random variable with a distribution $p_0(z_0)$ where p_0 is, for example, a Gaussian distribution. Then, for $k = 1, ..., K$ where K is the number of flow operations, let $z_k = f_i(z_{k-1})$ be a sequence of random multivariate variables. f_k^{-1} should exist for training to occur. The final output z_K models the target distribution.

Normalizing flow models are flexible, meaning they can model various distributions by stacking multiple normalizing flows to form a deep network. This allows it to capture complex relationships between variables in the data.

4.5 Diffusion Models

As introduced by Sohl-Dickstein et al. in 2015 [72], diffusion models simplify the generation process by breaking it down into smaller, more manageable steps. These models define a *Markov chain*[5] of diffusion steps, gradually adding random noise to data and then learning to reverse the diffusion process to construct desired data samples from the noise.

In practice, diffusion models use a Markov chain to gradually transform one distribution into another, starting from a simple known distribution (e.g., a Gaussian) into a target distribution using a diffusion process. Learning in this framework involves estimating small perturbations to a diffusion process, which can be accomplished using a network such as the U-Net [65]. Estimating small perturbations is more tractable than explicitly describing the entire distribution with a single, non-analytically-normalizable potential function.

The ultimate goal is to define a forward (or inference) diffusion process that can convert any complex data distribution into a simple, tractable distribution and then learn a finite-time reversal of this diffusion process that defines the generative model distribution. A major challenge is determining the appropriate noise level to increment per iteration. For example, training a network to directly denoise a full Gaussian to a real image is akin to training a GAN generator. It is more feasible to remove a small amount of noise per iteration.

To train these networks, one provides pairs of the original data sample X, a data sample at a random timestamp X_k, and the random step k, where $X_k = X + N(k)$ and N is a noising function. The network learns to extract the noise from the data given a timestamp, predicting $N(k)$ using image segmentation. However, since this prediction is imperfect, the network learns to predict $\tilde{N}(k)$ instead. Theoretically, applying $X_k - \tilde{N}(k)$ should yield $\tilde{X}$, which should be as close as possible to X. However, this process is challenging for large timestamps such as $k = 50$ since most data is Gaussian noise. On the other hand, applying the process for $k = 1$ is relatively straightforward.

During inference, the network receives noisy data X_k and a timestamp k, and returns $\tilde{N}(k)$. By doing $\tilde{X} = X_k - \tilde{N}(k)$, a bad data sample is generated. The algorithm then takes $\tilde{X}$ and applies $N(k-1)$, resulting in another noisy data sample with less noise. This process is repeated until $k = 0$, at which point a new data sample is generated.

4.6 Transformers

Deep learning has revolutionized audio generation in recent years. In 2017, the introduction of the "Attention is All You Need" [77] paper marked a significant milestone in deep learning. Although initially introduced for natural language processing, the transformer architecture has proven helpful in various data generation tasks, including audio synthesis. This marked a paradigm shift from

[5]A *Markov chain* [38] is a mathematical concept that describes a stochastic process. It exhibits the property of memorylessness, where the future state of the process solely relies on the present state and not on its history. In other words, a Markov chain satisfies the Markov property, indicating that the probability of transitioning to a future state is solely determined by the current state and not influenced by any past states. Consequently, from any given state, there exists a fixed probability of transitioning to different states in the subsequent step.

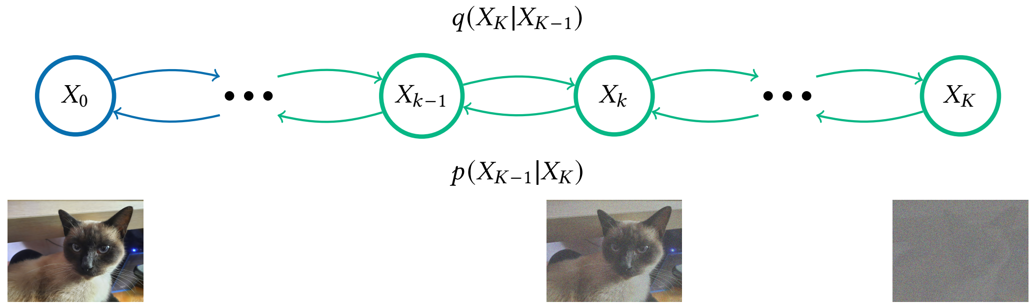

Fig. 5. **Diffusion model** — This illustration was based on [28] and shows the process of applying Gaussian noise to an image sample through multiple steps $q(X_k|X_{k-1})$. The model will then learn the operation p that transforms X_k into X_{k-1} with $p(X_{k-1}|X_k)$ and so on until X_0. At this point, the model has generated a new data sample.

the conventional Recurrent Neural Network (RNN)-based models, which were earlier widely used, with some incorporating a rudimentary form of the attention mechanism.

The attention mechanism enables the model to assign varying levels of importance to different input components. In the case of audio synthesis, the model can focus on specific aspects of the data, such as the pitch, rhythm, or timbre, to create more realistic sounds. The transformer architecture completely eschews the recurrent architecture and relies solely on attention, utilizing self-attention. Self-attention is a mechanism that enables the model to calculate the relevance of each input element concerning all other elements in the input sequence. This allows the model to dynamically focus on the most pertinent information at each calculation step, resulting in more accurate and realistic audio synthesis.

The paradigm shift introduced by this architectural transformation facilitates accelerated training and inference processes, allowing for the utilization of comprehensive datasets and substantial advancements in the quality of generated outcomes. As a result, it has brought about a revolutionary impact on audio generation. Transformers, known for their ability to process sequential input and output effectively, have emerged as formidable instruments across diverse realms of data generation, particularly in audio synthesis, as exemplified by Audiogen 5.4.7.

4.7 Vector Quantised Variational AutoEncoder (VQ-VAE)

The VQ-VAE, introduced in 2018, model distinguishes itself from traditional VAEs in two main aspects: the encoder network outputs discrete codes instead of continuous ones, and the prior is learned rather than static. While continuous feature learning has been the focus of many previous works, this model, introduced by [52], concentrates on discrete representations, a natural fit for complex reasoning, planning, and predictive learning.

The VQ-VAE model combines the VAE framework with discrete latent representations through a parameterization of the posterior distribution of (discrete) latents given an observation. Based on vector quantization, this model is simple to train, does not suffer from significant variance, and avoids the "posterior collapse". As illustrated in Fig 6, the VQ-VAE architecture consists of an encoder, a discrete latent space, and a decoder.

The VQ-VAE defines a latent embedding space $e \in R^{N \times D}$, where N is the size of the discrete latent space (i.e., a N-way categorical), and D is the dimensionality of each latent embedding vector e_n. There are N embedding vectors $e_n \in R^D, n \in 1, 2, ..., N$. The model takes an input X, passed through an encoder producing output $z_e(X)$. The discrete latent variables z are then calculated by

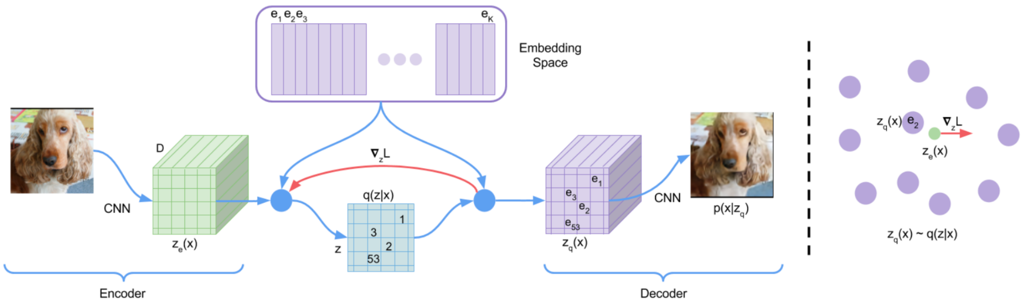

Fig. 6. **VQ-VAE** — Taken from the original paper, this figure presents two distinct illustrations. On the left side, a detailed diagram of the VQ-VAE architecture is provided, showcasing the flow of information through the encoder, the discrete latent space, and the decoder. On the right side, a visualization of the embedding space is displayed, where the encoder output $z(X)$ is mapped to its nearest embedding point e_2. The red arrow represents the gradient $\nabla_z L$, influencing the encoder's output adjustment. This adjustment may result in a different configuration during the subsequent forward pass, highlighting the dynamic nature of the learning process within the VQ-VAE model.

the nearest neighbor look-up using the shared embedding space e. The input to the decoder is the corresponding embedding vector e_n. This forward computation pipeline is a regular autoencoder with a non-linearity that maps the latents to 1-of-N embedding vectors.

The posterior categorical distribution $q(z|X)$ probabilities are defined as one-hot (Eq. 2):

$$q(z = n|X) = \begin{cases} 1 & \text{for } n = \text{argmin}_j ||z_e(X) - e_j||_2, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

where $z = e_n$ is the closest embedding vector to the encoder output $z_e(X)$. During forward computation, the nearest embedding $z_q(X)$ is passed to the decoder, and during the backward pass, the gradient $\nabla_z L$ is passed unaltered to the encoder. The overall loss function has three components to train different parts of the VQ-VAE: the reconstruction loss, the VQ objective, and the commitment loss. The total training objective becomes:

$$L = \log p(X|z_q(X)) \tag{3}$$
$$+ ||\text{sg}[z_e(X)] - e||_2^2 \tag{4}$$
$$+ \beta||z_e(X) - \text{sg}[e]||_2^2 \tag{5}$$

This equation combines the three following terms:

(1) **Reconstruction loss** (Equation 3): This term represents the log probability of the input data X given the latent variable $z_q(X)$. It measures how well the model can reconstruct the input data using $z_q(X)$ as a representation. Maximizing this term would lead to a better reconstruction of the input data.
(2) **VQ objective** (Equation 4): The second term measures the difference between the stop-gradient of the encoder output $z_e(X)$ and the embedding vector e. The stop-gradient operator, denoted as sg, acts as the identity during the forward pass but has zero partial derivatives during the backward pass. This term encourages the model to use the embeddings effectively by minimizing the distance between the encoder output and the closest embedding vector.

(3) **Commitment loss** (Equation 5): This term acts as a regularization term that measures the difference between the encoder output $z_e(X)$ and the stop-gradient of the embedding vector e. The β parameter controls the strength of this regularization. Minimizing this term would make $z_e(X)$ closer to the straight-through estimator of e.

VQ-VAE has emerged as a vital component in generative artificial intelligence, spanning domains such as image [59] and sound generation [81]. Our article recognizes VQ-VAEs as a cutting-edge deep learning architecture utilized in generating soundscapes, which we elaborate upon further in the models section.

5 GENERATIVE SOUNDSCAPES MODELS

The objective of deep learning models for soundscape generation is to create high-fidelity audio signals by learning from pre-existing sound data. It is important to notice that while numerous deep learning models are available for generative audio, there is a scarcity of models specifically developed for generating soundscapes.

Typically, these models comprise three primary components:

- Conversion of the sound signal into a compressed representation;
- Generation of a novel representation from the prior data;
- Conversion of the novel representation back into an audio signal.

The first component of the model entails transforming the original sound signal into a spectrogram (presented in Section 2) or another suitable representation, which is more compact to process than the raw audio signal.

The second component involves generating new low-resolution representations from the input, such as feature vectors, using deep generative architectures, as explained in Section 5.1. These models are trained on existing sound data to learn the target representation distribution and generate new, high-quality representations.

The third component of the model translates these newly-generated representations into an audio signal. These algorithms, known as *vocoders*, aim to produce high-fidelity audio signals that closely resemble the original sound data used to train the model.

5.1 Data Embedding

Data embedding is a crucial aspect of soundscape generation that involves converting data into numerical representations that capture its essential features and attributes. In the context of soundscape generation conditioned by textual prompts, both audio and text embedding are fundamental. Audio embedding, however, poses several challenges, including handling high-dimensional and sequential data, preserving temporal and spectral information, and assuring robustness and interpretability. To address these challenges, feature-based and learning-based methods can be applied.

While sound embedding methods have been developed, the authors could not find specialized sound embedding methods for soundscapes. Hence, this article focuses on sound embedding models that are not specific to soundscapes but rather for sound in general.

Feature-based embedding methods extract predefined features from raw audio data, such as spectral, temporal, or perceptual features [54]. These features are input into generative models or further processed to obtain lower-dimensional embeddings. One example of a feature-based embedding method is the application of the Short-Time Fourier Transform (STFT) to build a spectrogram. While feature-based embedding methods are simple and interpretable, they may lose some information or introduce noise during the feature extraction.

On the other hand, learning-based embedding methods learn embeddings directly from raw audio data using neural networks or other machine learning techniques. These methods can automatically discover relevant features from the data without relying on predefined criteria. Learning-based embedding methods are flexible and adaptive but may require more computational resources or suffer from overfitting or underfitting issues.

Text embedding has been a solved problem since the days of Word2Vec [47]. This autoencoder model is trained to capture a word's meaning by considering the words with which it appears. This model enables the representation of a word through a vector of latent factors.

However, for the problem under study, simply embedding the words is insufficient. When a user inputs a text prompt, the entire textual input must be embedded. A naive approach would be to average the latent factors for each input. However, this is now mainly solved with transformers (see Section 4.6). The transformer's encoder effectively processes an entire data string and produces a vector representation, thereby meeting the specific requirements of the task at hand.

The transformer model BERT, introduced in 2018, gained particular attention [12]. It introduced conditioning on the whole input for each word instead of previous vanilla transformer models that considered preceding words only.

While there are many techniques for data embedding, recent advancements in the field have led to the development of specialized models for various modalities, such as MuLan for audio. Instead of embedding the text and media separately and dealing with them afterward, media and text are embedded in the same space, meaning that a textual segment and a media sample representing the same textual segment should have similar latent factors.

5.1.1 MuLan. MuLan is a cutting-edge music audio embedding model introduced in 2022 [30] that aims to directly link music audio to unconstrained natural language music descriptions. It employs a two-tower parallel encoder architecture of two independent neural architectures using a contrastive loss objective to elicit a shared embedding space between music audio and text.

Each MuLan model comprises two separate embedding networks for the audio and text input modalities. These networks have no shared weights but terminate in 2-normalized embedding spaces with the same dimensionality. The contrastive loss objective minimizes the distance between matching audio-text pairs while maximizing the distance between mismatched pairs. This approach enables MuLan to learn a joint representation of music audio and text that captures their semantic relationships.

MuLan is trained using 44 million music recordings (equivalent to 370K hours) and weakly-associated, free-form text annotations. The resulting audio-text representation subsumes existing ontologies while graduating to true zero-shot functionalities. MuLan demonstrates versatility in transfer learning, zero-shot music tagging, language understanding in the music domain, and cross-modal retrieval applications.

5.2 Unsupervised Sound Generation

This section focuses on models that tackle unsupervised (or self-supervised) training, which involves acquiring knowledge of sound features and their distribution. This approach facilitates the generation of novel samples and enables latent feature representation. The following discussion covers notable models in this area, even if they were not specifically tailored for soundscape generation.

The selection of models in this section is based on their suitability for audio generation. Although these models were not explicitly designed for soundscape generation, they can still be utilized for feature extraction and can be adapted for soundscapes.

5.2.1 WaveGAN. Generative Adversarial Networks (GANs) have been highly influential in generating images that exhibit local and global coherence. In 2019, *WaveGAN* [14] has been proposed as a GAN-based model for the unsupervised synthesis of raw-waveform audio. The model modifies the transposed convolution operation used in Deep Convolutional GANs (DCGAN) to capture the structure of audio signals across various timescales. This modification includes using longer one-dimensional filters of length 25 instead of two-dimensional filters of size 5×5 and upsampling by a factor of 4 instead of 2 at each layer. Despite these changes, WaveGAN has the same number of parameters, numerical operations, and output dimensionality as DCGAN.

Experiments conducted on WaveGAN demonstrate that it can synthesize one-second slices of audio waveforms with global coherence, making it suitable for sound effect generation. The model also learns to produce intelligible words when trained on a small-vocabulary speech dataset without labels. The success of WaveGAN in generating coherent audio signals highlights the potential of GANs in generating high-quality sounds. This work opens up new possibilities for unsupervised synthesis of raw-waveform audio, such as music and speech. Additionally, it suggests that GANs can learn to capture the structure of signals across various timescales, which is a crucial factor in generating realistic audio.

5.2.2 SoundStream. SoundStream is a neural audio codec proposed in 2021 [82] that can efficiently compress speech, music, and general audio. A codec is software or hardware that compresses and decompresses audio signals. The model architecture consists of a fully convolutional encoder/decoder network and a residual vector quantizer, trained jointly end-to-end using both reconstruction and adversarial losses.

The fully convolutional encoder receives a time-domain waveform as input. It produces a sequence of embeddings at a lower sampling rate, which is then quantized by the residual vector quantizer (RVQ). The fully convolutional decoder then receives the quantized embeddings and reconstructs an approximation of the original waveform. The encoder and decoder use only causal convolutions, so the overall architectural latency of the model is determined solely by the temporal resampling ratio between the original time-domain waveform and the embeddings.

While there are similarities between SoundStream and a standard autoencoder in terms of the encoder-decoder architecture, SoundStream includes additional components such as the RVQ and the use of structured dropout for variable bitrate compression. A RVQ is a vector quantization method, e.g., a variant of the traditional vector quantization method in VQ-VAE. In an RVQ, the input data is first transformed into a lower-dimensional space using a neural network encoder. The resulting embeddings are then quantized using a codebook of fixed-size vectors, where each input embedding is assigned to the nearest codebook vector. However, instead of encoding the input embedding directly as the index of the assigned codebook vector, an RVQ computes the difference between the input embedding and the assigned codebook vector, known as the residual. The residual is then quantized using a second codebook, and the indices of both codebook vectors are transmitted as the compressed representation.

Using residual vectors in RVQ allows for better compression performance than traditional vector quantization methods. It captures the fine details of the input data that may be lost during quantization. In SoundStream, the RVQ is used to quantify the embeddings produced by the fully convolutional encoder, enabling efficient audio compression at low bitrates while maintaining high audio quality.

5.3 Vocoders

Generative models for audio synthesis have witnessed significant advancements in recent years. Among these models, vocoders are generative models that aim to synthesize raw audio waveforms based on a high-level representation, such as spectrograms. By utilizing vocoders, generating sounds with high sample rates that exhibit meaningful temporal dynamics and spectral characteristics becomes possible.

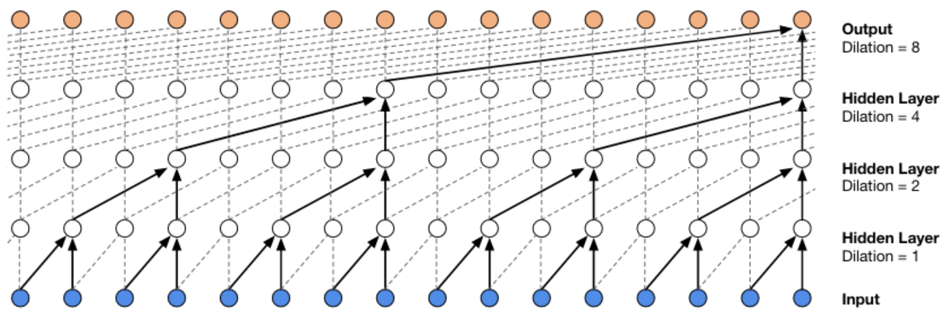

Fig. 7. **WaveNet** — This illustration was taken from [50]. It shows the idea behind WaveNet, applying dilated convolutions to autoregressive models.

Earlier research primarily focused on autoregressive approaches using neural networks, as discussed in Section 5.3.1. However, current investigations concentrate on the utilization of GANs, as evidenced by Sections 5.3.3, 5.3.4, and 5.3.5. GAN-based vocoders can generate high-quality soundscapes requiring less data and computational resources than previous autoregressive models.

5.3.1 WaveNet. *WaveNet* is a generative neural network developed by DeepMind in 2016. It uses a unique architecture based on dilated causal convolutions to generate raw audio waveforms [50]. It implements the PixelCNN [51] model for sound and follows an autoregressive architecture with the predictive distribution for each audio sample being conditioned on a window of previous ones.

WaveNet's structure allows it to process input sequences in parallel, enabling it to model long context dependencies, even with thousands of timesteps. It uses a series of dilated convolutional layers, where the dilation rate is increased with each layer, which effectively increases the receptive field of the network without increasing the number of parameters.

This structure enables WaveNet to capture long-range dependencies in the input sequence, vital for generating high-quality audio and text. If an RNN sees only one input sample at each time step, WaveNet has direct access to multiple input samples. For example, in speech generation, WaveNet can use its sizeable receptive field to model the relationship between a word spoken early in a sentence and its pronunciation later in the sentence.

WaveNet uses a softmax activation function at each output node to produce a probability distribution over the possible values at each time step. During training, the network is fed sequences of input data and their corresponding ground truth values. The model's parameters are adjusted so that its outputs match the ground truth as closely as possible.

WaveNet can use its trained parameters to generate new sequences by sampling from its output probability distribution during generation. This allows it to generate diverse and high-quality outputs, such as realistic human speech or written text, by combining its learned representations of the underlying data distribution with a small amount of randomness. The input of WaveNet is usually a mel-spectrogram (or other representations), and the output is a sound signal. WaveNet can be conditioned on, for instance, text for TTS settings by feeding extra information about the text itself (*e.g.* embeddings). If a model is not conditioned on text, it generates random sounds without any global structure behind it.

Even though this model is good at learning the characteristics of sounds over brief periods, it struggles with global latent structure. They are also very slow for training and inferring [75].

5.3.2 WaveNet Variants. The WaveNet model has proven to be a powerful tool for generating high-quality audio waveforms, especially for speech and music applications. However, its architecture, which utilizes dilated convolutions and deep residual networks, can be computationally demanding and challenging to train. Several WaveNet variants have been proposed in recent years to overcome these limitations to reduce the model's complexity while maintaining its effectiveness.

One such variant is WaveRNN [35], which employs a single RNN to approximate the dilated convolutions in WaveNet. This approach significantly speeds up training time while maintaining the quality of the generated audio. Another variant, FloWaveNet [40], employs a flow-based generative model that allows for efficient training with only one training stage while producing high-quality audio. Additionally, Fast WaveNet [53] employs a caching mechanism to reduce the computational cost of the model while maintaining an autoregressive structure.

These WaveNet variants have unique architectures and training procedures but share the goal of making audio generation more efficient and accessible. While these models are primarily focused on speech and music generation, they can be adapted to other types of audio data. Ongoing research in this area may explore further optimization of these models, integration with other models, and application to new domains.

5.3.3 MelGAN. The 2019's *MelGAN* paper [44] proposes that generating coherent raw audio waveforms with GANs is challenging but possible with some architectural changes. Previous works in this area have struggled to generate high-quality, coherent waveforms with GANs. However, the authors of MelGAN demonstrate that it is feasible to train GANs reliably to generate high-quality and coherent waveforms.

The generator in MelGAN is a fully convolutional feed-forward network that takes a mel-spectrogram as input and generates a raw waveform as output. This approach enables efficient and parallelized processing of audio data. The decoder determines whether the generated waveform is a realistic sound. It is not a single neural network but a multi-scale architecture with three discriminators (D1, D2, D3). These discriminators have identical network structures but operate on different audio scales. D1 operates on the scale of raw audio, while D2 and D3 operate on raw audio downsampled by a factor of 2 and 4, respectively. The use of multiple discriminators at different scales is motivated by the fact that audio has structure at different levels. MelGAN is significantly faster than other architectures, such as WaveNet, with comparable results (for inference, roughly thirty-six thousand times faster than WaveNet), given its reduced number of parameters.

5.3.4 GANSynth. *GanSynth*, presented in 2019, [16] is a GAN that generates coherent waveforms using log-magnitude spectrograms and phases. Compared to directly generating waveforms with stridden convolutions, the use of spectrograms and phases has been shown to produce better results.

The model begins by sampling a random vector z from a spherical Gaussian distribution. This vector is passed through a stack of transposed convolutions, which upsample and generate output data $x = G(z)$. The generated data is then fed into a discriminator network, which employs downsampling convolutions to estimate a divergence measure between the real and generated distributions.

The architecture of the discriminator network mirrors that of the generator, which allows for a more efficient training process. Optimizing the divergence measure enables the generator to produce spectrograms and phases that more closely resemble actual musical notes. Results show that GANs outperform WaveNet baselines on automated and human evaluation metrics and can efficiently generate several audio orders of magnitude faster than their autoregressive counterparts.

5.3.5 HiFi-GAN. Proposed in 2020, *HiFi-GAN* [42] is a GAN model that combines efficiency and high-fidelity speech synthesis by leveraging the periodic patterns inherent in speech audio. The model includes a generator and two discriminators, trained adversarially, and two additional losses for improving training stability and model performance.

The generator is a fully convolutional neural network that takes mel-spectrograms as input and upsamples them through transposed convolutions, matching the temporal resolution of raw waveforms. The discriminators are the multi-scale discriminator (MSD) and a multi-period discriminator (MPD). MSD evaluates the audio sequence on different scales using a mixture of three convolutional sub-discriminators with different average pools. At the same time, MPD consists of small sub-discriminators that capture different implicit structures of input audio by looking at different parts, accepting only equally spaced samples of input audio with different periods.

HiFi-GAN achieves high-quality synthesis efficiently, generating 22.05 kHz high-fidelity audio 167.9 times faster than real-time on a single V100 GPU, demonstrating superior computational efficiency compared to autoregressive and flow-based models. Additionally, a small-footprint version of HiFi-GAN generates samples 13.4 times faster than real-time on CPU with comparable quality to an autoregressive counterpart.

5.4 End to End Models

Audio synthesis is the task of producing artificial audio signals from text or other modalities. Conventional audio synthesis systems consist of multiple modules, such as a data analysis frontend, a sound model, and an audio synthesis backend. These modules require substantial domain knowledge and may entail suboptimal design decisions. Moreover, these modules are often trained independently on different objectives and datasets, which can result in errors and inconsistencies in the synthesized audio.

To overcome these challenges, end-to-end models that directly learn the mapping between text (or other modalities) and audio waveform using deep neural networks have been developed. These models obviate the need for intermediate stages and enable a more seamless integration of diverse modalities. Some examples of end-to-end models for audio synthesis are AudioGen 5.4.7, which synthesizes audio from text using discrete representations and a Transformer language model with augmentation and multi-stream techniques, and Riffusion 5.4.6, which synthesizes audio clips from text prompts using spectrogram images and Stable Diffusion with text and image conditioning.

5.4.1 SampleRNN. SampleRNN is a neural audio generation model proposed in 2017 that can produce high-quality audio samples from scratch [46]. It uses a hierarchical structure of RNNs to model the probability distribution of audio waveforms at different temporal resolutions. The lowest RNN operates on individual samples, while higher RNNs capture longer-term dependencies and structure. SampleRNN can learn from any audio data without any prior knowledge or labels.

The higher RNNs capture longer-term dependencies by receiving inputs from lower RNNs at a lower sampling rate, allowing them to process longer audio sequences. The higher RNNs also use skip connections to directly access the outputs of lower RNNs, which helps to avoid vanishing gradients and preserve information across different levels of abstraction. Each cell is an RNN variant that takes as input a frame of audio samples from a lower RNN and outputs a hidden state vector that encodes the long-term context of the audio. This output is passed upwards in the hierarchy to other RNNs that take it. Multiple layers are possible, each operating at a different temporal resolution. All the outputs are then inputted in the final level RNN, whose output is the next audio sample based on the combined information from all hierarchy levels.

5.4.2 Char2Wav. The Char2Wav model, proposed in 2017 [60], serves as a speech synthesis model comprising two distinct components: a reader and a neural vocoder. The reader accepts textual inputs and produces acoustic features as outputs. The neural vocoder then utilizes these acoustic features to generate raw waveform samples. The reader component is an attention-based recurrent sequence generator, a neural network capable of generating output sequences based on input sequences. In this context, the input sequences correspond to the text, while the output sequences represent acoustic features. The generator employs a bidirectional RNN as an encoder and an RNN with attention as a decoder. The attention mechanism enables the model to concentrate on various parts of

5.4.3 Jukebox. Jukebox, a generative model for music that produces music with singing in the raw audio domain, was introduced by Dhariwal et al. in 2020 [13]. The model addresses the challenge of dealing with the long context of raw audio by utilizing a multiscale VQ-VAE, which compresses the raw audio into discrete codes, and autoregressive Transformers to model those codes.

Once the VQ-VAE is trained, a prior $p(z)$ over the compressed space is learned to generate samples. The prior model is decomposed as $p(z) = p(z_{top})p(z_{middle}|z_{top})p(z_{bottom}|z_{middle}, z_{top})$, and separate models are trained for the top-level prior $p(z_{top})$, and upsamplers $p(z_{middle}|z_{top})$ and $p(z_{bottom}|z_{middle}, z_{top})$. Autoregressive Transformers with sparse attention are utilized for modeling in the discrete token space generated by the VQ-VAE.

Jukebox can generate high-quality and varied songs with coherence for multiple minutes. The model can be conditioned on the artist and genre to control the musical and vocal style, as well as on unaligned lyrics to enhance the controllability of the singing. The model's release includes thousands of non-cherry-picked samples, model weights, and code.

5.4.4 AudioLM. AudioLM, proposed in 2022 [7], introduces a sophisticated framework that leverages discrete audio representations and language modeling techniques to synthesize audio of exceptional quality while preserving long-term coherence.

The fundamental premise of AudioLM consists of transforming the input audio waveform into a sequence of discrete tokens, which can subsequently be manipulated by a Transformer-based model [77], which is a type of neural network that can learn to encode and decode sequences of data using parallel computations and positional embeddings. AudioLM exploits the remarkable aptitude of Transformer-based models to capture extensive dependencies and generate natural and consistent continuations based on short prompts by formulating audio synthesis as a language modeling task within this discrete representation space.

Nevertheless, pursuing a suitable discrete audio representation demands considerable time and effort. On the one hand, the representation should preserve the high fidelity of the audio waveform, necessitating a high bitrate and an extensive token sequence. On the other hand, the representation should be concise while encompassing the semantic content and long-term structure of the audio. AudioLM relies on a hybrid tokenization scheme that combines acoustic and semantic tokens to reconcile these conflicting requirements. Acoustic tokens encapsulate the intricate details of the audio waveform, while semantic tokens capture the higher-level meaning and structure. By conditioning the generation of acoustic tokens on semantic tokens, AudioLM achieves high-quality and coherent audio synthesis.

The tokenization and detokenization models facilitate the conversion of the input audio waveform into a sequence of discrete tokens and the reverse process, respectively. These models are pre-trained and fixed before training the language model, thereby decoupling them from the language modeling objective and simplifying the training procedure. In the case of AudioLM, SoundStream (see Section 5.2.2) is employed for computing acoustic tokens, while w2v-BERT [11] is utilized for computing semantic tokens.

The language model adopted by AudioLM is a decoder-only Transformer that operates on the discrete tokens generated by the tokenization model. Its training objective maximizes the likelihood of generating semantic and acoustic tokens given an input prompt. Specifically, AudioLM employs a hierarchical approach by first modeling the semantic tokens for the entire sequence and subsequently using them as conditioning to predict the acoustic tokens. This hierarchical strategy enables AudioLM to leverage both types of tokens in generating coherent and high-quality audio continuations.

The input prompt encompasses a sequence of semantic and acoustic tokens, each embedded discretely. The semantic embeddings undergo processing by a stack of Transformer decoder layers, thereby engendering a sequence of hidden states that encode both the input prompt and its extrapolation. The acoustic embeddings are concatenated with these hidden states and further processed by another stack of Transformer decoder layers incorporating causal self-attention and cross-attention over the hidden semantic states. The terminal layer produces logits over the acoustic token vocabulary, which are employed to sample novel acoustic tokens. The loss function is the cross-entropy between the predicted logits and the target acoustic tokens.

5.4.5 DiffSound. *DiffSound* [81] is a text-to-sound generation framework presented in 2022 that utilizes a text encoder, a VQ-VAE, a decoder, and a vocoder to synthesize audio corresponding to the input text. It focuses on designing a suitable decoder, which is a critical component of the framework. DiffSound is a diffusion decoder based on the discrete diffusion model. It predicts all mel-spectrogram tokens in one step and then refines the predicted tokens in the next step, resulting in better-predicted results after several steps. DiffSound outperforms the autoregressive (AR) decoder regarding text-to-sound generation quality and speed, with a generation speed five times faster than an AR decoder.

The framework operates as follows: first, the text is encoded into embeddings using a transformer or similar model. Second, this representation conditions the generation of spectrogram embeddings using diffusion (the DiffSound model). Third, these embeddings are passed through a pre-trained VQ-VAE decoder to generate the spectrogram. Finally, the spectrogram is processed through a vocoder, such as MelGAN, to generate the waveform. The complete process is illustrated in Fig. 8.

5.4.6 Riffusion. Riffusion [21] is an open-source model presented in 2022 that generates audio clips from text prompts. The model is based on Stable Diffusion and fine-tunes it to generate images of spectrograms, which can be converted to audio clips. The model computes spectrograms from audio using STFT and approximates the phase using the Griffin-Lim algorithm when reconstructing the audio clip. The model was conditioned on text prompts and other images using diffusion models, allowing sound transformations while preserving the structure of an original clip. The denoising strength parameter controls how much the generated clip deviates from the original clip toward the new prompt.

The model takes a text prompt as input during inference, which is then encoded into a latent representation using a text encoder. The model generates a spectrogram image from the latent representation using a modified version of Stable Diffusion [64] that is fine-tuned for spectrograms. Finally, the generated spectrogram image is converted into an audio clip using the Griffin-Lim algorithm.

5.4.7 AudioGen. In 2023, Kreuk et al. [43] proposed AudioGen, an auto-regressive generative model that generates audio samples conditioned on text inputs. The model comprises two primary stages: (i) learning a discrete representation of the raw audio using an auto-encoding method and (ii) training a Transformer language model over the learned codes obtained from the audio encoder, conditioned on textual features. During inference, the model samples from the language model generate a new set of audio tokens given text features, which can then be decoded into the waveform domain using the decoder component.

To address the challenge of text-to-audio generation, the authors propose an augmentation technique that mixes different audio samples to train the model to separate multiple sources

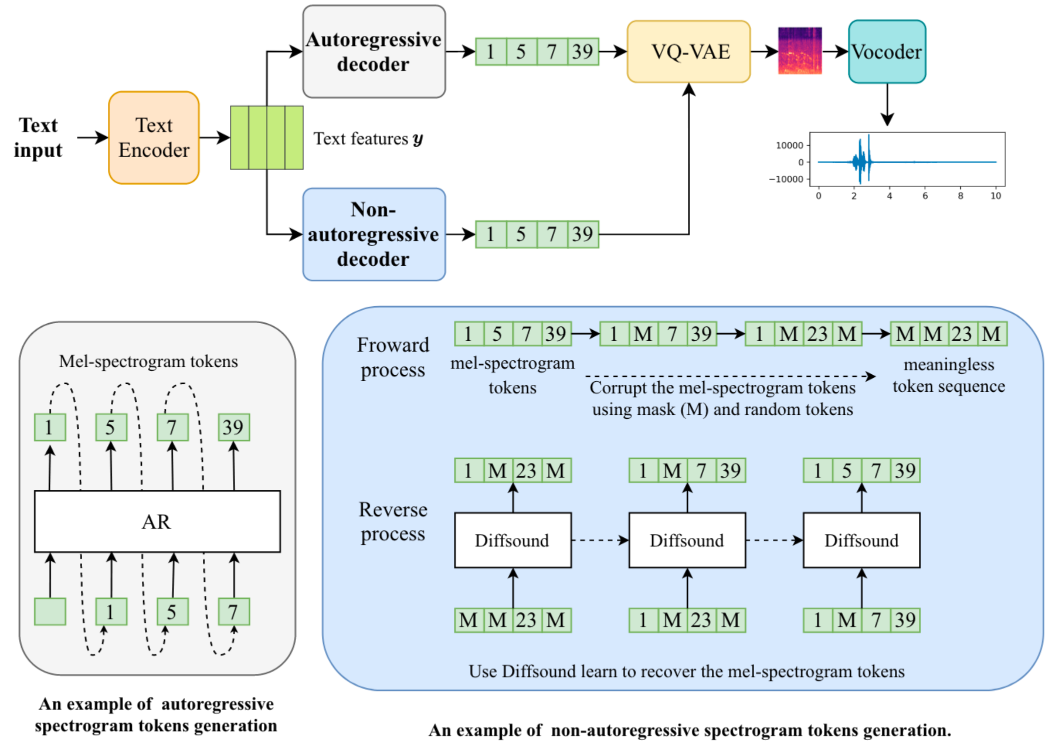

Fig. 8. **DiffSound framework** — This illustration was taken from the original paper. At the top, the general framework is present. Two decoders are present, but only one of them is used. The decoder results in a set of latent features. These features are passed to the decoder of the VQ-VAE that generates a Mel-Spectrogram (the square with red and blue tones) that, through a vocoder, generates a sound. The two bottom images represent the two decoders. One is a DARN, the other works with diffusion.

internally. Furthermore, the authors explore the use of multi-stream modeling for faster inference, allowing the use of shorter sequences while maintaining a similar bitrate and perceptual quality. The proposed method outperforms evaluated baselines over both objective and subjective metrics. Additionally, the authors extend the proposed method to conditional and unconditional audio continuation, demonstrating its ability to generate complex audio compositions.

6 DISCUSSION AND FUTURE DIRECTIONS

This article provides a comprehensive and current overview of deep learning techniques for sound generation, focusing on soundscapes driven by text prompts. It has reviewed and summarized the state-of-the-art concerning datasets, data augmentation, data generation frameworks, and sound generation models. It has also critically analyzed and compared the existing approaches and datasets for sound generation and highlighted their advantages and drawbacks.

The advent of deep learning techniques has brought about a new and demanding research area in sound generation, specifically soundscapes, which holds significant potential for a multitude of domains and industries. Soundscapes have emerged as a promising tool for musical orchestration, acoustic design, and urban planning, among other applications [56]. The versatility and utility of soundscapes have made them a valuable asset to various fields, aiming to leverage the power of sound to enhance user experience and quality of life.

The current state of research on deep learning techniques for synthesizing soundscapes has limitations that reveal growth opportunities. The field would benefit from improved foundational resources like datasets, data augmentation methods, generative models, and vocoders.

Compared to datasets in other AI domains, existing datasets for soundscape generation are limited in size, diversity, and relevant annotations. The largest available dataset, AudioSet, lacks specialized labels and fails to meet the scale needed to train complex neural networks. As a result, researchers rely on techniques like concatenating and overlaying sounds to augment data. However, more targeted augmentation methods, inspired by natural language processing techniques, could prove helpful in exploiting text annotations.

Recent advances in generative models, including transformers, diffusion models, and VQ-VAEs, have pushed the field forward. Models mapping audio and text to shared embeddings, like MuLAN, have enabled some success in embedding-based generation. Meanwhile, vocoders like HiFi-GAN can synthesize high-fidelity audio from embeddings. Although GANs are still commonly used for vocoders, their stagnation in other tasks suggests that new vocoder architectures may be needed. Models like AudioGen and DiffSound have demonstrated the potential of end-to-end text-based soundscape generation using transformers and diffusion, respectively. However, other approaches relying on adapting image generation models to operate on spectrograms, like Riffusion, highlight the nascent state of current techniques.

In this context, twofold directions for future research in the soundscape generation area are identified. First, there is a need to construct more realistic and diverse datasets for sound generation specifically focused on soundscapes. Descriptive labels and additional metadata are indispensable for facilitating the training of models to generate authentic and realistic soundscapes. These datasets are currently limited by their scarcity, noise, or homogeneity. Thus, forthcoming datasets should encompass a more comprehensive array of sounds from diverse soundscape environments. This would augment the authenticity and realism of synthesized soundscapes.

Second, creating a dedicated system focused on soundscapes, capable of transforming textual prompts into audio samples, is an imperative need. While there are existing end-to-end systems for other modalities, there is a requirement for a specialized system explicitly tailored for soundscapes. This system could be composed of pre-existing embedding, generation, and vocoder techniques, or it could be the outcome of research and development efforts aimed at refining and enhancing these models. Such system would not only advance the field of soundscape generation but also open up new possibilities for creative expression and exploration in the realm of audio synthesis.

Advancements in this domain may facilitate applications ranging from immersive virtual environments to auditory data augmentation. Overall, this is a promising area of research that deserves greater focus within the audio artificial intelligence communities.

ACKNOWLEDGMENTS

We want to express our most sincere gratitude to our institution, the Faculty of Engineering of the University of Porto, and to the Artificial Intelligence and Computer Science Laboratory, for providing us with the opportunity to pursue our academic goals and for the resources and facilities essential for completing this work.

REFERENCES

[1] Olusola O. Abayomi-Alli, Robertas Damaševičius, Atika Qazi, Mariam Adedoyin-Olowe, and Sanjay Misra. 2022. Data Augmentation and Deep Learning Methods in Sound Classification: A Systematic Review. en. *Electronics*, 11, 22, (Jan. 2022), 3795. Number: 22 Publisher: Multidisciplinary Digital Publishing Institute. DOI: 10.3390/electronics11223795.

[2] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. 2016. YouTube-8M: A Large-Scale Video Classification Benchmark. arXiv:1609.08675 [cs]. (Sept. 2016). DOI: 10.48550/arXiv.1609.08675.

[3] Hadeer Ahmed, Issa Traore, Mohammad Mamun, and Sherif Saad. 2023. Text augmentation using a graph-based approach and clonal selection algorithm. en. *Machine Learning with Applications*, 11, (Mar. 2023), 100452. DOI: 10.1016/j.mlwa.2023.100452.

[4] Francesc Alías, Joan Claudi Socoró, and Xavier Sevillano. 2016. A Review of Physical and Perceptual Feature Extraction Techniques for Speech, Music and Environmental Sounds. en. *Applied Sciences*, 6, 5, (May 2016), 143. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute. DOI: 10.3390/app6050143.

[5] Anam Bansal and Naresh Kumar Garg. 2022. Environmental Sound Classification: A descriptive review of the literature. en. *Intelligent Systems with Applications*, 16, (Nov. 2022), 200115. DOI: 10.1016/j.iswa.2022.200115.

[6] Gilberto Bernardes, Luis Aly, and Matthew Davies. 2016. Seed: Resynthesizing environmental sounds from examples. In *Proceedings of the Sound and Music Computing Conference*. (Sept. 2016), 55–62. DOI: 10.5281/zenodo.851183.

[7] Zalán Borsos et al. 2022. AudioLM: a Language Modeling Approach to Audio Generation. arXiv:2209.03143 [cs, eess]. (Sept. 2022). DOI: 10.48550/arXiv.2209.03143.

[8] S. Chandrakala and S. L. Jayalakshmi. 2019. Environmental Audio Scene and Sound Event Recognition for Autonomous Surveillance: A Survey and Comparative Studies. *ACM Computing Surveys*, 52, 3, (June 2019), 63:1–63:34. DOI: 10.1145/3322240.

[9] Tilanka Chandrasekera, So-Yeon Yoon, and Newton D'Souza. 2015. Virtual environments with soundscapes: a study on immersion and effects of spatial abilities. en. *Environment and Planning B: Planning and Design*, 42, 6, (Nov. 2015), 1003–1019. Publisher: SAGE Publications Ltd STM. DOI: 10.1068/b130087p.

[10] Tsz-Him Cheung and Dit-Yan Yeung. 2021. {MODALS}: Modality-agnostic Automated Data Augmentation in the Latent Space. In *International Conference on Learning Representations*. https://openreview.net/forum?id=XjYgR6gbCEc.

[11] Yu-An Chung, Yu Zhang, Wei Han, Chung-Cheng Chiu, James Qin, Ruoming Pang, and Yonghui Wu. 2021. W2v-BERT: Combining Contrastive Learning and Masked Language Modeling for Self-Supervised Speech Pre-Training. arXiv:2108.06209 [cs, eess]. (Sept. 2021). Retrieved Apr. 10, 2023 from http://arxiv.org/abs/2108.06209.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805. arXiv: 1810.04805. http://arxiv.org/abs/1810.04805.

[13] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. 2020. Jukebox: A Generative Model for Music. arXiv:2005.00341 [cs, eess, stat]. (Apr. 2020). DOI: 10.48550/arXiv.2005.00341.

[14] Chris Donahue, Julian McAuley, and Miller Puckette. 2019. Adversarial Audio Synthesis. arXiv:1802.04208 [cs]. (Feb. 2019). DOI: 10.48550/arXiv.1802.04208.

[15] Konstantinos Drossos, Samuel Lipping, and Tuomas Virtanen. 2019. Clotho: An Audio Captioning Dataset. *CoRR*, abs/1910.09387. arXiv: 1910.09387. http://arxiv.org/abs/1910.09387.

[16] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. 2019. GANSynth: Adversarial Neural Audio Synthesis. arXiv:1902.08710 [cs, eess, stat]. (Apr. 2019). DOI: 10.48550/arXiv.1902.08710.

[17] J. D. Fernandez and F. Vico. 2013. AI Methods in Algorithmic Composition: A Comprehensive Survey. en. *Journal of Artificial Intelligence Research*, 48, (Nov. 2013), 513–582. DOI: 10.1613/jair.3908.

[18] Nathaniel Finney and Jordi Janer. 2010. Soundscape generation for virtual environments using community-provided audio databases. In *W3C Workshop: Augmented Reality on the Web*. Barcelona Spain.

[19] J. L. Flanagan and R. M. Golden. 1966. Phase Vocoder. en. *Bell System Technical Journal*, 45, 9, 1493–1509. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.1538-7305.1966.tb01706.x. DOI: 10.1002/j.1538-7305.1966.tb01706.x.

[20] Eduardo Fonseca, Manoj Plakal, Frederic Font, Daniel P. W. Ellis, Xavier Favory, Jordi Pons, and Xavier Serra. 2018. General-purpose Tagging of Freesound Audio with AudioSet Labels: Task Description, Dataset, and Baseline. (2018). DOI: 10.48550/ARXIV.1807.09902.

[21] Seth* Forsgren and Hayk* Martiros. 2022. Riffusion - Stable diffusion for real-time music generation. (2022). https://riffusion.com/about.

[22] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. 2017. Audio Set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*. New Orleans, LA.

[23] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press, Cambridge, MA, USA.

[24] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. arXiv:1406.2661 [cs, stat]. (June 2014). DOI: 10.48550/arXiv.1406.2661.

[25] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. 2014. Deep AutoRegressive Networks. arXiv:1310.8499 [cs, stat]. (May 2014). DOI: 10.48550/arXiv.1310.8499.

[26] Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019. Augmenting Data with Mixup for Sentence Classification: An Empirical Study. arXiv:1905.08941 [cs]. (May 2019). DOI: 10.48550/arXiv.1905.08941.

[27] Noushin Hajarolasvadi, Miguel Arjona Ramírez, Wesley Beccaro, and Hasan Demirel. 2020. Generative Adversarial Networks in Human Emotion Synthesis: A Review. *IEEE Access*, 8, 218499–218529. Conference Name: IEEE Access. DOI: 10.1109/ACCESS.2020.3042328.

[28] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. arXiv:2006.11239 [cs, stat]. (Dec. 2020). DOI: 10.48550/arXiv.2006.11239.

[29] Reynald Hoskinson and Dinesh K. Pai. 2001. Manipulation and Resynthesis with Natural Grains. In *Proceedings of the 2001 International Computer Music Conference, ICMC 2001, Havana, Cuba, September 17-22, 2001.* Michigan Publishing. https://hdl.handle.net/2027/spo.bbp2372.2001.057.

[30] Qingqing Huang, Aren Jansen, Joonseok Lee, Ravi Ganti, Judith Yue Li, and Daniel P. W. Ellis. 2022. MuLan: A Joint Embedding of Music Audio and Natural Language. arXiv:2208.12415 [cs, eess, stat]. (Aug. 2022). Retrieved Apr. 10, 2023 from http://arxiv.org/abs/2208.12415.

[31] Muhammad Huzaifah and Lonce Wyse. 2021. Deep Generative Models for Musical Audio Synthesis. In *Handbook of Artificial Intelligence for Music: Foundations, Advanced Approaches, and Developments for Creativity.* Eduardo Reck Miranda, (Ed.) Springer International Publishing, Cham, 639–678. ISBN: 978-3-030-72116-9. DOI: 10.1007/978-3-030-72116-9_22.

[32] Guillermo Iglesias, Edgar Talavera, Angel González-Prieto, Alberto Mozo, and Sandra Gómez-Canaval. 2023. Data Augmentation techniques in time series domain: a survey and taxonomy. en. *Neural Computing and Applications*, (Mar. 2023). DOI: 10.1007/s00521-023-08459-3.

[33] 2014. ISO 12913-1:2014(en), Acoustics — Soundscape — Part 1: Definition and conceptual framework. (2014). Retrieved Apr. 26, 2023 from https://www.iso.org/obp/ui/#iso:std:iso:12913:-1:ed-1:v1:en.

[34] Shulei Ji, Jing Luo, and Xinyu Yang. 2020. A Comprehensive Survey on Deep Music Generation: Multi-level Representations, Algorithms, Evaluations, and Future Directions. arXiv:2011.06801 [cs, eess]. (Nov. 2020). DOI: 10.48550/arXiv.2011.06801.

[35] Nal Kalchbrenner et al. 2018. Efficient Neural Audio Synthesis. arXiv:1802.08435 [cs, eess]. (June 2018). DOI: 10.48550/arXiv.1802.08435.

[36] Stefano Kalonaris and Anna Jordanous. 2018. Computational Music Aesthetics: a survey and some thoughts. en. In Accepted: 2018-07-09. Dublin, Ireland, (Aug. 2018). Retrieved Apr. 13, 2023 from http://galapagos.ucd.ie/wiki/pub/OpenAccess/CSMC/Kalonaris.pdf.

[37] Navdeep Kaur and Parminder Singh. 2022. Conventional and contemporary approaches used in text to speech synthesis: a review. en. *Artificial Intelligence Review*, (Nov. 2022). DOI: 10.1007/s10462-022-10315-0.

[38] John G Kemeny, James Laurie Snell, et al. 1960. *Finite markov chains.* Vol. 356. van Nostrand Princeton, NJ.

[39] Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. 2019. AudioCaps: Generating Captions for Audios in The Wild. In *NAACL-HLT.*

[40] Sungwon Kim, Sang-gil Lee, Jongyoon Song, and Sungroh Yoon. 2018. FloWaveNet : A Generative Flow for Raw Audio. *CoRR*, abs/1811.02155. arXiv: 1811.02155. http://arxiv.org/abs/1811.02155.

[41] Diederik P. Kingma and Max Welling. 2022. Auto-Encoding Variational Bayes. arXiv:1312.6114 [cs, stat]. (Dec. 2022). DOI: 10.48550/arXiv.1312.6114.

[42] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. 2020. HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis. In *Advances in Neural Information Processing Systems.* H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, (Eds.) Vol. 33. Curran Associates, Inc., 17022–17033. https://proceedings.neurips.cc/paper/2020/file/c5d736809766d46260d816d8dbc9eb44-Paper.pdf.

[43] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. 2023. AudioGen: Textually Guided Audio Generation. arXiv:2209.15352 [cs, eess]. (Mar. 2023). DOI: 10.48550/arXiv.2209.15352.

[44] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C Courville. 2019. MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis. In *Advances in Neural Information Processing Systems.* H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, (Eds.) Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/6804c9bca0a615bdb9374d00a9fcba59-Paper.pdf.

[45] Shuo Liu, Adria Mallol-Ragolta, Emilia Parada-Cabaleiro, Kun Qian, Xin Jing, Alexander Kathan, Bin Hu, and Björn W. Schuller. 2022. Audio self-supervised learning: A survey. en. *Patterns*, 3, 12, (Dec. 2022), 100616. DOI: 10.1016/j.patter.2022.100616.

[46] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. 2017. SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. arXiv:1612.07837 [cs]. (Feb. 2017). DOI: 10.48550/arXiv.1612.07837.

[47] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. (2013). DOI: 10.48550/ARXIV.1301.3781.

[48] Ana Filipa Rodrigues Nogueira, Hugo S. Oliveira, José J. M. Machado, and João Manuel R. S. Tavares. 2022. Sound Classification and Processing of Urban Environments: A Systematic Literature Review. en. *Sensors*, 22, 22, (Nov. 2022), 8608. DOI: 10.3390/s22228608.

[49] Ondřej Novotný, Oldřich Plchot, Ondřej Glembek, Jan Honza Cernocký, and Lukáš Burget. 2019. Analysis of DNN Speech Signal Enhancement for Robust Speaker Recognition. en. *Computer Speech & Language*, 58, (Nov. 2019), 403–421. DOI: 10.1016/j.csl.2019.06.004.

[50] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. arXiv:1609.03499 [cs]. (Sept. 2016). DOI: 10.48550/arXiv.1609.03499.

[51] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. 2016. Conditional Image Generation with PixelCNN Decoders. arXiv:1606.05328 [cs]. (June 2016). DOI: 10.48550/arXiv.1606.05328.

[52] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2018. Neural Discrete Representation Learning. arXiv:1711.00937 [cs]. (May 2018). DOI: 10.48550/arXiv.1711.00937.

[53] Tom Le Paine, Pooya Khorrami, Shiyu Chang, Yang Zhang, Prajit Ramachandran, Mark A. Hasegawa-Johnson, and Thomas S. Huang. 2016. Fast Wavenet Generation Algorithm. arXiv:1611.09482 [cs]. (Nov. 2016). DOI: 10.48550/arXiv.1611.09482.

[54] Geoffroy Peeters, Bruno L. Giordano, Patrick Susini, Nicolas Misdariis, and Stephen McAdams. 2011. The Timbre Toolbox: Extracting audio descriptors from musical signals. en. *The Journal of the Acoustical Society of America*, 130, 5, (Nov. 2011), 2902–2916. DOI: 10.1121/1.3642604.

[55] Hieu Pham, Xinyi Wang, Yiming Yang, and Graham Neubig. 2021. Meta Back-Translation. In *International Conference on Learning Representations*. https://openreview.net/forum?id=3jjmdp7Hha.

[56] Bryan C. Pijanowski, Luis J. Villanueva-Rivera, Sarah L. Dumyahn, Almo Farina, Bernie L. Krause, Brian M. Napoletano, Stuart H. Gage, and Nadia Pieretti. 2011. Soundscape Ecology: The Science of Sound in the Landscape. *BioScience*, 61, 3, (Mar. 2011), 203–216. _eprint: https://academic.oup.com/bioscience/article-pdf/61/3/203/19404645/61-3-203.pdf. DOI: 10.1525/bio.2011.61.3.6.

[57] Domagoj Pluščec and Jan Šnajder. 2023. Data Augmentation for Neural NLP. arXiv:2302.11412 [cs]. (Feb. 2023). DOI: 10.48550/arXiv.2302.11412.

[58] Yanmin Qian, Hu Hu, and Tian Tan. 2019. Data augmentation using generative adversarial networks for robust speech recognition. en. *Speech Communication*, 114, (Nov. 2019), 1–9. DOI: 10.1016/j.specom.2019.08.006.

[59] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-Shot Text-to-Image Generation. arXiv:2102.12092 [cs]. (Feb. 2021). DOI: 10.48550/arXiv.2102.12092.

[60] Kanishka Rao, Fuchun Peng, Hasim Sak, and Francoise Beaufays. 2015. Grapheme-to-phoneme conversion using Long Short-Term Memory recurrent neural networks. en. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, South Brisbane, Queensland, Australia, (Apr. 2015), 4225–4229. ISBN: 978-1-4673-6997-8. DOI: 10.1109/ICASSP.2015.7178767.

[61] Edison Research. 2021. The Infinite Dial 2021. en-US. Section: Featured. (Mar. 2021). Retrieved Apr. 26, 2023 from https://www.edisonresearch.com/the-infinite-dial-2021-2/.

[62] Danilo Jimenez Rezende and Shakir Mohamed. 2016. Variational Inference with Normalizing Flows. arXiv:1505.05770 [cs, stat]. (June 2016). DOI: 10.48550/arXiv.1505.05770.

[63] Vincent Roger, Jérôme Farinas, and Julien Pinquier. 2022. Deep neural networks for automatic speech processing: a survey from large corpora to limited data. en. *EURASIP Journal on Audio, Speech, and Music Processing*, 2022, 1, (Aug. 2022), 19. DOI: 10.1186/s13636-022-00251-w.

[64] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2021. High-Resolution Image Synthesis with Latent Diffusion Models. *CoRR*, abs/2112.10752. arXiv: 2112.10752. https://arxiv.org/abs/2112.10752.

[65] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:1505.04597 [cs]. (May 2015). DOI: 10.48550/arXiv.1505.04597.

[66] J. Salamon, C. Jacoby, and J. P. Bello. 2014. A Dataset and Taxonomy for Urban Sound Research. In *22nd ACM International Conference on Multimedia (ACM-MM'14)*. Orlando, FL, USA, (Nov. 2014), 1041–1044.

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

Acoustics (WASPAA). IEEE, New Paltz, NY, (Oct. 2017), 344–348. ISBN: 978-1-5386-1632-1. DOI: 10.1109/WASPAA.2017.8170052.

[68] R. Murray Schafer. 1977. *The Tuning of the World*. en. Google-Books-ID: SIufAAAAMAAJ. Knopf. ISBN: 978-0-394-40966-5.

[69] Zhaofeng Shi. 2021. A Survey on Audio Synthesis and Audio-Visual Multimodal Processing. arXiv:2108.00443 [cs, eess]. (Aug. 2021). DOI: 10.48550/arXiv.2108.00443.

[70] Connor Shorten, Taghi M. Khoshgoftaar, and Borko Furht. 2021. Text Data Augmentation for Deep Learning. *Journal of Big Data*, 8, 1, (July 2021), 101. DOI: 10.1186/s40537-021-00492-0.

[71] Julius O. Smith. 2007. *Mathematics of the Discrete Fourier Transform (DFT)*. W3K Publishing, http://www.w3k.org/books/http://www.w3k.org/books/. ISBN: 978-0-9745607-4-8.

[72] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. arXiv:1503.03585 [cond-mat, q-bio, stat]. (Nov. 2015). DOI: 10.48550/arXiv.1503.03585.

[73] David Sonnenschein. 2001. *Sound Design: The Expressive Power of Music, Voice, and Sound Effects in Cinema*. en. Michael Wiese Productions. ISBN: 978-0-941188-26-5.

[74] Gerda Strobl, Gerhard Eckel, and Davide Rocchesso. 2006. Sound Texture Modeling: A Survey, en. In *Proceedings of the Sound and Music Computing Conference*, 61–65.

[75] Koray Tahiroğlu, Miranda Kastemaa, and Oskar Koli. 2020. Al-terity: Non-Rigid Musical Instrument with Artificial Intelligence Applied to Real-Time Audio Synthesis. English. In *Proceedings of the International Conference on New Interfaces for Musical Expression* (Proceedings of the International Conference on New Interfaces for Musical Expression). International Conference on New Interfaces for Musical Expression, (July 2020), 337–342.

[76] Naoya Takahashi, Michael Gygli, Beat Pfister, and Luc Van Gool. 2016. Deep Convolutional Neural Networks and Data Augmentation for Acoustic Event Recognition. In *Proc. Interspeech 2016*, 2982–2986. DOI: 10.21437/Interspeech.2016-805.

[77] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. arXiv:1706.03762 [cs]. (Dec. 2017). DOI: 10.48550/arXiv.1706.03762.

[78] Jason Wei and Kai Zou. 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, (Nov. 2019), 6382–6388. DOI: 10.18653/v1/D19-1670.

[79] Lilian Weng. 2018. Flow-based Deep Generative Models. *lilianweng.github.io*. https://lilianweng.github.io/posts/2018-10-13-flow-models/.

[80] Xuenan Xu, Mengyue Wu, and Kai Yu. 2022. A Comprehensive Survey of Automated Audio Captioning. arXiv:2205.05357 [cs, eess]. (May 2022). DOI: 10.48550/arXiv.2205.05357.

[81] Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. 2022. Diffsound: Discrete Diffusion Model for Text-to-sound Generation. arXiv:2207.09983 [cs, eess]. (July 2022). DOI: 10.48550/arXiv.2207.09983.

[82] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2021. SoundStream: An End-to-End Neural Audio Codec. arXiv:2107.03312 [cs, eess]. (July 2021). Retrieved Apr. 10, 2023 from http://arxiv.org/abs/2107.03312.