npj | acoustics

Review

https://doi.org/10.1038/s44384-025-00021-w

Machine Learning in Acoustics: A Review and Open-source Repository

Check for updates

Ryan A. McCarthy[1] ✉, You Zhang[2], Samuel A. Verburg[3], William F. Jenkins[1] & Peter Gerstoft[1,3]

Acoustic data provide scientific and engineering insights in fields ranging from bioacoustics and communications to ocean and earth sciences. In this review, we survey recent advances and the transformative potential of machine learning (ML) in acoustics including deep learning (DL). Using the Python high-level programming language, we demonstrate a broad collection of ML techniques to detect and find patterns for classification, regression, and generation in acoustics data automatically. We have ML examples including acoustic data classification, generative modeling for spatial audio, and physics-informed neural networks. This work includes **AcousticsML**, a set of practical Jupyter notebook examples on GitHub demonstrating ML benefits and encouraging researchers and practitioners to apply reproducible data-driven approaches to acoustic challenges.

Machine learning (ML)[1–4] has become a valuable tool for processing and analyzing large acoustic datasets and improving the interpretability of acoustic data[5–12]. A search of publications containing "machine learning" and "acoustics" as keywords reveals that interest in applying ML solutions to problems in acoustics has grown exponentially in the past 15 years, as illustrated by Fig. 1. This interest is partly driven by technological advances in acoustics, which produce increasingly large datasets. These datasets present new challenges for evaluation and interpretation, as the time and resources required to analyze such data manually are becoming prohibitively expensive. ML algorithms offer solutions to some of these challenges with their ability to identify patterns and trends through statistical and nonlinear approaches, offering fast, efficient, and reproducible approaches that scale to the growing size and complexity of acoustic datasets.

Although ML provides powerful tools for acoustic data processing and analysis, applying these techniques to acoustics remains difficult without guides that address the field's unique data challenges. Many fields (e.g., seismology, econometrics, meteorology) have tutorials demonstrating domain-specific ML applications, but establishing direct parallels with acoustic applications is often not straightforward. This work addresses these challenges by providing an open-source GitHub repository, designated as AcousticsML, featuring Jupyter Notebooks with diverse ML applications in acoustics, including time series analysis, physics-based modeling, classification, clustering, and techniques for managing large datasets. Additionally, we describe several structured workflows for applying machine learning to acoustic data, with each pipeline tailored to specific applications while offering a replicable framework. Although ML libraries are available in many programming languages and software suites, we rely on Python since it provides researchers with a mature ecosystem for ML development, offering comprehensive libraries, extensive documentation, and readable syntax.

Using ML is an inherently data-intensive task. The curation of publicly available data sets that can be used to evaluate the suitability of ML models for certain types of data has aided ML development. However, the field of acoustics has fewer defined baseline datasets with which to train and evaluate models. In this work we highlight several publicly available acoustic datasets for speech [13,14], ambient noise and sound classification[15–19], room impulse responses[20–22], and head-related transfer functions (HRTFs)[23–26]; additional data sets can be found in online repositories such as Kaggle or Zenodo. Additionally, we show models developed with smaller or simulated datasets that are nevertheless able to provide adequate information on which the models can train. Though simulations offer cost-efficient training alternatives when real data are limited, potential biases and realism limitations must be carefully addressed, a challenge that presents both constraints and opportunities for innovation in acoustic ML.

Recent advances in ML techniques

In ref. 5, several ML algorithms were highlighted to provide relevant details about acoustic applications. In particular, ML principles, supervised and unsupervised, and deep learning (DL) theory were discussed in detail to demonstrate key advancements in ML and how they can be applied to acoustic datasets. Models are trained from measurements and are formulated as

$$y = f(x) + \epsilon, \tag{1}$$

where y is the output, x is a single input (observation) with N features, $f(x)$ is the model that maps input features to the output, and ϵ is the uncertainty in

[1]Scripps Institution of Oceanography, UC San Diego, La Jolla, CA, 92093, USA. [2]Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY, 14627, USA. [3]Department of Electrical and Photonics Engineering, Technical University of Denmark, 2800 Kgs., Lyngby, Denmark.
✉e-mail: r1mccarthy@ucsd.edu

model prediction. The trained model algorithms can vary from linear to non-linear estimators that best fit any application. Although many ML techniques have been covered in ref. 5, this paper briefly reviews recent ML techniques, emphasizing acoustic applications through open-source code.

Generative models

Generative models have become essential tools in acoustic ML. As DL technology progresses, three main types of generative models have emerged at the forefront: Generative Adversarial Networks (GANs)[27], Variational Autoencoders (VAEs)[28–30], and Diffusion Models[31,32].

Variational autoencoder. VAEs[33], illustrated at the top of Fig. 2, are generative models based on an encoder-decoder architecture, generalized from deterministic autoencoders, which are mainly used for dimensionality reduction (see Section "Implicit neural representation"). The encoder (E) maps the input data $\mathbf{x}$ to a probabilistic latent space, defined by a distribution over low-dimensional latent variables $\mathbf{z}$, while the decoder (D) reconstructs the data from samples of this distribution. VAEs minimize the reconstruction error while ensuring that the latent space adheres to a prior distribution, typically a standard normal distribution.

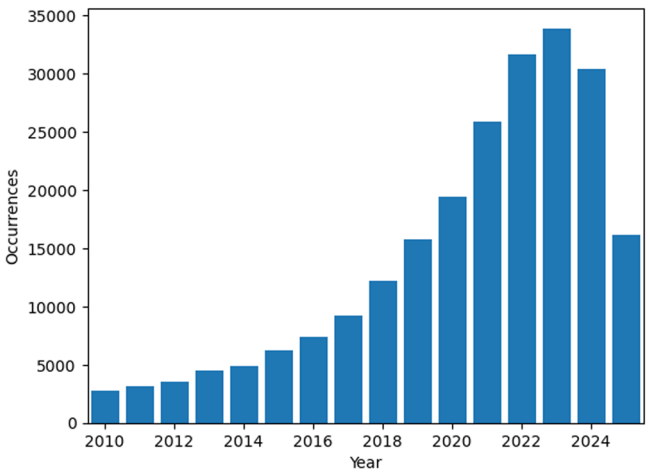

Fig. 1 | Publications Containing Machine Learning + Acoustics (July 2025). Number of publications containing the keywords "machine learning" and "acoustics", using the script in ref. 149.

The training of VAEs involves two objectives: reconstruction and regularization of the latent space. By minimizing the loss of reconstruction, the model aims to recreate the original data from its compressed version as accurately as possible. This ensures that the VAE learns an accurate representation of the data that can be used to generate new samples. Additionally, VAEs ensure that the latent space follows a simple and well-organized structure, typically similar to random noise. This step enables the model to generate diverse and realistic new samples, thereby preventing it from overfitting the training data.

Mathematically, this optimization is grounded in maximizing the variational lower bound, which can be thought of as a means to strike the best balance between fitting the model to the data and maintaining the model's simplicity by Kullback-Leibler (KL) divergence regularization. The model learns two distributions: the encoder distribution $p(\mathbf{x}|\mathbf{z})$ and the decoder distribution $q(\mathbf{z}|\mathbf{x})$.

$$\log p(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right] - D_{KL}\left(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right), \tag{2}$$

where $p(\mathbf{x}|\mathbf{z})$ is the likelihood of the data given the latent variables. $p(\mathbf{z})$ is the prior distribution on the latent variables, often chosen to be a standard Gaussian $\mathcal{N}(0, I)$. $q(\mathbf{z}|\mathbf{x})$ is the approximate posterior parameterized by a neural network.

The first term encourages the decoder to produce data reconstruction close to the input, while the second term, KL divergence, penalizes the model if its latent representations deviate too much from a predefined, simple distribution (often a Gaussian), ensuring that the latent space does not overfit specific data points in the datasets.

The combination of these two objectives strikes a balance between accurate data reconstruction and a smooth, well-structured, and meaningful latent space, enabling VAEs to generate meaningful new data. For example, once trained, a VAE can generate entirely new audio samples that are similar to the training data but not identical, making it particularly useful for applications such as speech synthesis[34] or music generation[35], where diversity and smooth latent interpolations are crucial.

Generative adversarial networks. GANs[36] are a class of ML models[37] that consist of two neural networks that work together: a generator (G) and a discriminator (D). These networks are trained through an adversarial process, where the generator attempts to create synthetic data

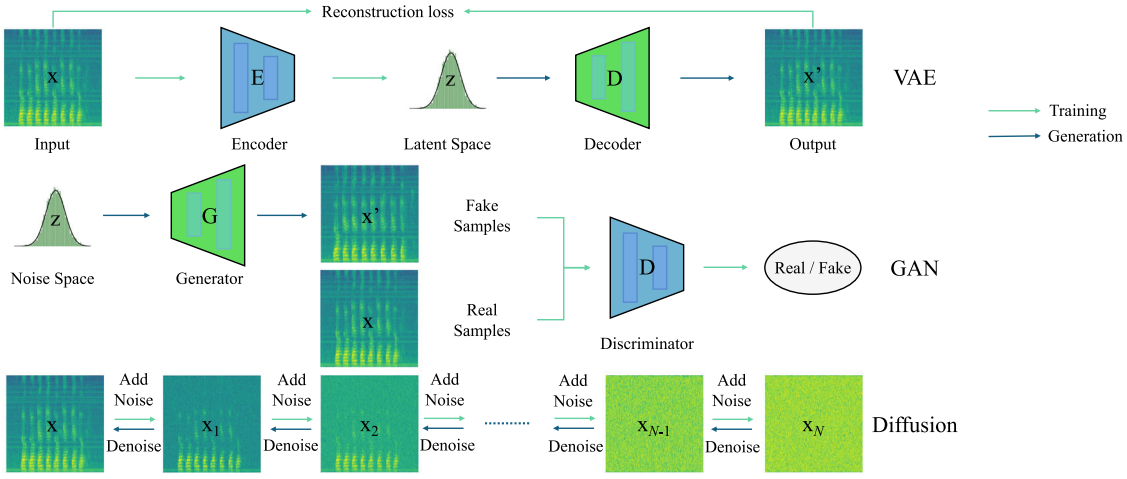

Fig. 2 | Illustration of different types of generative models with generating audio spectrograms as an example. 1) Variational Autoencoder (VAE): Trained to reconstruct the spectrogram while regularizing the latent space to a Gaussian distribution. The decoder (D) generates spectrograms by decoding Gaussian noise. 2) Generative Adversarial Networks (GANs): The generator (G) synthesizes spectrograms from random noise, while the discriminator (D) distinguishes real from generated samples in an adversarial training setup. 3) Diffusion Models: Trained by gradually adding noise to the spectrogram, then learning to reverse the process using neural networks. After training, the model generates spectrograms by reversing the noise process from Gaussian noise.

samples that follow real data distribution, and the discriminator's task is to distinguish between real and generated samples. This process is illustrated in Fig. 2 middle.

The generator takes a random noise vector $\mathbf{z}$ as input and learns to map it to the input data space, generating samples that resemble the real data. On the other hand, the discriminator attempts to correctly classify real and generated data, outputting the probability that a given sample is real (rather than fake). The discriminator tries to minimize the objective

$$\min_D \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))], \tag{3}$$

while the generator's objective is to maximize the probability of the discriminator making an error in distinguishing real from fake data.

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))], \tag{4}$$

where x represents real data, z is the random noise input, and $G(z)$ is the synthetic sample generated by the generator.

The training proceeds in a minimax game, where the generator and discriminator continuously improve, with the generator trying to produce increasingly realistic samples and the discriminator learning to distinguish between real and fake data. Through this adversarial process, GANs can generate highly realistic data in various domains, such as images, audio, and video, making them a cornerstone of modern generative modeling.

This adversarial process has been successfully adapted to generate audio data for tasks like sound synthesis[38]. A prominent application is speech generation from melspectrograms, where models like HiFi-GAN[39] achieve high-quality speech generation by reconstructing waveforms with exceptional fidelity. GAN has also been applied to generating room impulse responses,[27,40]. GANs have also shown effectiveness in the recent speech dialogue foundation model[41].

Anomaly detection leverages GANs to model the data distribution and identify out-of-distribution samples in the test set. Common approaches involve incorporating the learning of inverse mapping in contrast to the generators, which map data back to its latent representation, and using the reconstruction error as an anomaly score. Applications of GAN-based anomaly detection in acoustics include anomalous machine sound detection[42,43] and deepfake audio detection[44].

Additionally, adversarial training has been extended to tasks beyond synthesis, where the generator and discriminator adopt roles tailored to specific acoustic challenges. For instance, the discriminator can be trained as a classifier, while the generator acts as an encoder to extract latent features. In scenarios like channel-agnostic speaker embedding extraction,[45] the discriminator classifies channel information while the generator learns speaker representations. By adversarially maximizing the discriminator's classification error, the generator successfully encodes features invariant to channel variations, outperforming traditional data augmentation techniques. Similar ideas have been applied to speaker-invariant emotion recognition[46] and audio anti-spoofing[47].

GANs have also inspired novel designs for specific acoustic tasks, such as the design of acoustics metamaterials[48] and underwater noise modeling[49]. MetricGAN[50] focuses on optimizing perceptual metrics for speech enhancement, while other GAN variants tackle super-resolution in audio reconstruction[51]. These innovations demonstrate how GANs can be tailored and refined for specific applications in acoustics.

Diffusion models. Diffusion models[52,53] are a class of generative models inspired by thermodynamic processes. They generate samples by a denoising process when they learn to reverse a step-by-step noising process applied to the data. The overall framework, illustrated in the bottom subfigure of Fig. 2, consists of two main stages: a forward diffusion process that adds noise gradually and a reverse denoising process that reconstructs the data. We take a standard diffusion model named denoising diffusion probabilistic models (DDPM) to explain the process in detail. Due to the complexity of the mathematical foundation, we refer interested readers to ref. 54 for additional information on other diffusion models and a more generalized formulation through stochastic differential equations.

In the forward process ($\mathbf{x}_{t-1}$ to $\mathbf{x}_t$), Gaussian noise is added to the input data in a series of stages, gradually corrupting it until it is pure random noise. The scale of the noise varies at each step. Mathematically, each forward step can be expressed as

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t I), \tag{5}$$

where $\mathbf{x}_t$ represents the noisy data at step t, and β_t controls the amount of noise added at each step. The coefficients $\sqrt{1-\beta_t}$ and $\sqrt{\beta_t}$ encourage the distribution of the t step to be closer to a unit distribution compared to the $t-1$ step. The forward diffusion process can be viewed as equivalent to the encoding step of VAE models, which uses latent variables $\mathbf{z}_t$ to estimate the probability distribution.

$$\mathbf{z}_t = \sqrt{1-\beta_t}\mathbf{z}_{t-1} + \sqrt{\beta_t}\epsilon_t, \tag{6}$$

where $\epsilon_t \sim \mathcal{N}(0, I)$ is a Gaussian noise.

The reverse procedure ($\mathbf{x}_t$ back to $\mathbf{x}_{t-1}$) seeks to restore the original data from the corrupted version by gradually denoising. This is achieved through a neural network that learns to predict and reverse the noise step by step. The denoiser aims to minimize the difference between the predicted clean data and the true data across each step.

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_\theta(t)I) \tag{7}$$

where $\mu_\theta(\mathbf{x}_t, t)$ is the model's prediction for the clean data at step $t-1$ based on the noisy input at step t, and $\sigma_\theta(t)$ controls the noise removal process.

Latent diffusion models (LDM)[55] have been proposed as a more efficient approach. LDMs learn the denoising process in a lower-dimensional space, rather than directly on the raw data, allowing many irrelevant details in the data to be abstracted away. Consistency models[56] enforce consistency in the generated samples at any step t, reducing the number of required steps during sampling and leading to more efficient generation while maintaining high-quality outputs.

In acoustics, diffusion models have been applied in sound field synthesis[57], text-to-audio generation[58,59], and spatial audio generation[60].

Discussions on generative models. While VAEs and GANs utilize the same underlying principles, there are some important differences to note. VAEs stand out with their ability to model complex data distributions with a continuous latent space, offering both high-quality generation and interpretable representations. Their architecture includes encoder and decoder neural networks trained in tandem to enhance representation learning and reconstruction accuracy. GANs are well-known for producing high-quality outputs, where they can create realistic samples from random noise. However, they often require a delicate balance during training, and issues like mode collapse[61] can occur.

In contrast to VAEs and GANs, diffusion models adopt a distinct approach centered around optimizing the reverse diffusion process. This methodology emphasizes learning to predict and remove noise effectively, enabling the generation of high-quality samples that closely resemble the training data without the issues commonly faced by GANs, such as mode collapse or instabilities during adversarial training. Due to their stable training dynamics and flexibility, strong theoretical foundation, and ability to handle complex data distributions, diffusion models are valuable tools for generating realistic data in a variety of domains. The drawbacks of diffusion models include the high dimensionality of the noise (latent variables), which is the same as the original data, and the slow inference speed resulting from the large number of steps involved in the sampling process.

Implicit neural representation

Traditional data representations often take the form of high-dimensional matrices. For example, an image is typically stored as a matrix of pixel values, where the dimensions correspond to its width and height. However, this representation poses limitations, particularly when merging datasets of the same category but with varying resolutions and dimensions. Such structural inconsistencies can complicate downstream processing and integration tasks and hinder the generalizability of models trained on data with fixed dimensions.

Implicit neural representations (INRs),[62] also referred to as neural fields, offer a continuous and differentiable method for representing discrete data using neural networks. Instead of explicitly storing data in a matrix, INR uses a neural network to map input coordinates to corresponding output values, creating a continuous data representation. For example, in the case of images, a small neural network is trained to represent a single image. The network takes spatial coordinates as input and outputs the corresponding RGB pixel values at those coordinates. This approach inherently supports interpolation and enables the seamless merging of datasets with different resolutions.

The mathematical formulation is

$$f_\theta(\mathbf{x}) = \mathbf{y}, \tag{8}$$

where f_θ is the neural network parameterized by θ, $\mathbf{x}$ is the input coordinate (e.g., spatial coordinates), $\mathbf{y}$ is the output value (e.g., amplitude, color, distance, or any other property).

This representation can potentially model the distribution of a specific data type across varying dimensions or resolutions. It also relates directly to meta-learning, enabling generalization across multiple INRs. This approach contrasts with traditional neural network methods, which are typically trained on large datasets with fixed input-output structures. In the conventional case, the neural network is trained to produce an entire output of fixed dimensionality given some conditional input.

INRs have primarily evolved within computer graphics and computer vision over the years, but they have recently been adopted in acoustics, particularly in spatial audio, due to the location-dependent nature of the data and its requirement for consecutive resolution.

Creating immersive audio experiences requires accurate modeling of sound propagation from the source to the listener through space. A key challenge in room acoustics is modeling room impulse responses. However, real-world measurements are limited by the microphone-loudspeaker setup, which can only capture a restricted number of source-listener pairs. Traditional methods use mathematical interpolation to calculate the impulse response at new locations, while INRs provide a novel solution. Neural Acoustic Fields (NAFs)[63] were proposed to leverage neural fields to represent impulse responses in the time-frequency domain. Implicit Neural Representation for Audio Scenes (INRAS)[64] further refined this approach, introducing a decoupled module model to represent the scatter-bounce-gather process in audio propagation. This research has recently been extended to audio-visual novel-view acoustic synthesis,[65] where camera angle information from visual cues is incorporated as input to predict the impulse response at specific audio-visual scenes, making it particularly useful for audio-visual navigation.

The binaural audio through headphones or VR headsets is rendered not only with room acoustics but also incorporates the propagation to the ears, described by head-related transfer functions (HRTFs), and faces a similar modeling challenge due to limited measurements. HRTFs are continuous functions that take spatial directions as input and output the spectrum across all frequency bins and thus INRs are well-suited for this task. This idea was applied to binaural audio[66], where personalized HRTFs were implicitly modeled by estimating transformation functions for binaural synthesis using neural networks. The measurement directions do not constrain this method and directly predict binaural audio, with HRTFs as intermediate outputs and no ground truth. HRTF Field[67] directly applies INRs to model HRTFs across datasets and reveals another benefit of mixed database training for interpolation tasks, alleviating differences in spatial sampling schemes. This approach was further extended[68], where the model estimates the coefficients of cascaded infinite impulse response (IIR) filters rather than the HRTF magnitude directly, enabling a more compact representation that better captures the resonant characteristics of HRTFs with fewer parameters.

INRs are memory-efficient due to their simple architecture and ability to represent infinite resolution with the same set of parameter weights. However, optimizing INRs is challenging because they rely on continuous representations. INRs are prone to overfitting if training data is insufficient or lack diversity, which requires proper regularization to generalize well, such as mitigating differences between the HRTF databases[69]. INRs may struggle to accurately capture highly detailed or sharp features, particularly in data with high-frequency content.

Physics-informed machine learning

Physics-informed machine learning (PIML) integrates physical principles with ML to solve scientific and engineering problems[7]. Many physical systems are governed by physical laws that are (partially) understood through centuries of scientific progress. It is only natural to leverage well-established scientific knowledge to improve current ML workflows. At the same time, ML can be used for scientific discovery[70,71], helping us understand physical aspects that are currently poorly understood or too complex to model with traditional methods.

Incorporating physical knowledge into ML models can improve their accuracy, efficiency, interpretability, robustness, and generalization capabilities. Physical priors guide models toward learning physically plausible solutions, making them more accurate than ML models that rely purely on data. By adding physical constraints, the space of possible models considered by the learning algorithm is narrowed. Consequently, PIML models tend to be data-efficient, making them particularly useful in scenarios where data is scarce or expensive. In contrast, traditional ML models typically require large amounts of training data.

Physically motivated constraints also act as effective regularizers, improving model robustness. Models that incorporate physical laws generalize better and can extrapolate to regions where data is sparse or unavailable. On the other hand, traditional ML models often perform poorly outside the range of the training data and are more prone to overfitting.

The interpretability of ML models—i.e., the ability to understand and explain how the models make predictions and decisions—is crucial for building trustworthy ML systems. PIML models are generally more interpretable because they adhere to physical laws, leading to more reliable predictions and a better understanding of the model's behavior. In contrast, traditional ML, especially deep learning models, are often considered 'black boxes' with limited interpretability. Additionally, PIML models often require fewer parameters and less complex architectures compared to traditional ML models. These simpler models are often more transparent and easier to interpret.

There are different strategies to incorporate physics into ML workflows. One way is to embed physical principles directly into the network architecture design. An example of this is neural ordinary differential equations[72], which link residual neural networks to numerical time integrators. Following this idea, a very active field of research involves the design of custom network architectures that can predict the time evolution of dynamical systems robustly and efficiently[73]. Another way of incorporating physics into ML is to include physical constraints in the loss function, as described in the following section.

Physics-informed neural networks

This section provides an introduction to *physics-informed neural networks* (PINNs), one of the most popular modes of PIML. PINNs are neural networks that integrate physical constraints into their loss function. Physical systems are often expressed as partial differential equations (PDEs). These can be linear, such as the acoustic wave or Helmholtz equations, nonlinear,

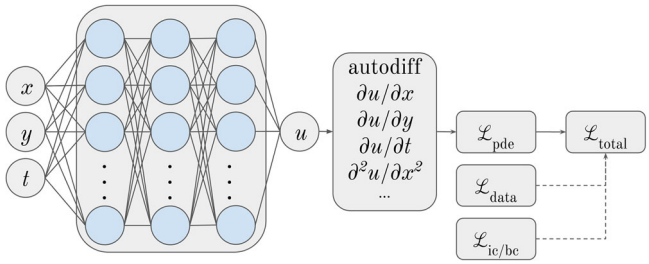

Fig. 3 | Diagram of a PINN. The neural network inputs are the coordinates x, y, t, and the output is the physical quantity of interest $u(x, y, t)$. Automatic differentiation is used to compute the partial derivatives of the output with respect to the inputs. A physics loss term, $\mathcal{L}_{pde}$, that contains the underlying PDE is formed, and added to loss terms that account for initial/boundary conditions, $\mathcal{L}_{ic/bc}$, and/or observed data, $\mathcal{L}_{data}$, to compute the total loss, $\mathcal{L}_{total}$.

such as the Burgers' equation, or a system of coupled PDEs. PINNs approximate the solution of a PDE by incorporating a residual term into its loss function that contains the PDE. Fig. 3 shows an example of a PINN. During training, the PDE residual is minimized, along with other terms that account for initial/boundary conditions and observed data. A key aspect of PINNs is the use of automatic differentiation to compute the differential equations that encode the physics into the loss function. Automatic differentiation, the backbone of modern ML, makes it possible to easily compute partial derivatives by breaking functions into elementary operations and applying the chain rule systematically.

The term PINNs was popularized around 2019[74]. Since then, PINNs have been applied in many fields of science and engineering[7,75], including fluid dynamics[6], climate modeling[76], and biomedicine[77], to name a few. In acoustics, PINNs have been applied in ocean acoustics[78], atmospheric sound propagation[79], room acoustics[80,81], spatial audio and sound field control[82], acoustic holography[83], ultrasound imaging[84,85], nonlinear propagation[86], and spatial inverse problems[87].

As with other PIML approaches, PINNs achieve better generalization and require less data than purely data-driven neural networks while being expressive enough to approximate complex PDE solutions. Unlike conventional numerical methods, PINNs are gridless, i.e., they can make predictions at any resolution without the need to be retrained. Since the PDE is enforced over the full domain, PINNs can make zero-shot predictions, i.e., the solution can be predicted at points where there are no data. In addition, PINNs are highly flexible, allowing them to solve both forward and inverse problems. For instance, a forward problem involves computing a wavefield given initial and boundary conditions, while an inverse problem aims to estimate PDE parameters (e.g., wave speed profile) from observed data. AcousticsML includes notebooks for forward and inverse problems involving the wave equation.

However, PINNs have limitations and challenges. Training a PINN for solving a forward problem is significantly slower than using conventional numerical solvers such as finite differences or finite elements. This is due to the need to extend the computational graph to compute the partial derivatives that constitute the PDE residual. Further, training PINNs can be difficult due to competing terms in the loss function and gradient stiffness[88–90]. Moreover, like other deep neural networks, PINNs suffer from spectral bias, struggling to capture the high-frequency content of the PDE solution[91].

Several extensions have been proposed to address these challenges, making PINNs a very active field of research. Actively selecting training points can achieve faster convergence[92]. Annealing algorithms that automatically scale different terms in the loss function have been proposed to alleviate gradient stiffness issues[88]. The use of Fourier features[91] and subdomain partitioning[93,94] has been proposed to address spectral bias and multiscale problems. Some of these extensions are covered in the AcousticsML notebooks.

Hyperparameter optimization

Most ML algorithms are controlled by tunable parameters that the user sets. Such parameters are often referred to as hyperparameters, as they are distinct from the parameters—or model weights—that are learned during the training process. For example, neural network weights are learned during training, while the learning rate, number of layers, number of neurons per layer, and other settings are hyperparameters set by the user. Other examples of hyperparameters include the number of trees in a random forest, the number of clusters in a clustering algorithm, and the number of neighbors in a nearest neighbors algorithm. More generally, hyperparameters are any parameters not learned during the training process and must be set by the user. This can include the choice of algorithm, loss function, or preprocessing steps, although in practice, such design choices are typically informed by domain knowledge.

Hyperparameter optimization is critical to selecting the model that best explains the dataset according to some criterion. The choice of criterion typically includes some measure of model performance, such as accuracy, precision, recall, or error, and may also incorporate computational cost. Consider a ML model F_θ with trainable parameters θ that maps data from an input space $\mathcal{X}$ to predictions in an output space $\mathcal{Y}$:

$$F_\theta : \mathcal{X} \to \mathcal{Y}. \tag{9}$$

The objective of hyperparameter optimization is to find the best set of hyperparameters ϕ^* that minimizes a loss function $\mathcal{L}$ over the hyperparameter space:

$$\phi^* = \arg\min_{\phi} \left[\mathcal{L}(F_\theta)\right]_\phi. \tag{10}$$

Eq. (10) can be solved in several systematic ways. A common approach is grid search, in which each hyperparameter is assigned a set of possible values, and the model is trained and evaluated for each combination of hyperparameters. This approach is simple and intuitive, but can be computationally expensive or even prohibitive, especially for models with many hyperparameters. Random or quasi-random searches can also be used, but may still require many models to be trained and evaluated[95]. Recent advances in Bayesian optimization have shown promise in addressing these challenges by constructing a probabilistic surrogate model of Eq. (10) and using the surrogate model to select the next set of hyperparameters to evaluate. Bayesian optimization incorporates the results of previous evaluations to inform the selection of future hyperparameters, allowing for more efficient exploration of the hyperparameter space[96,97]. Grid and random search are readily implemented in many ML libraries, while Bayesian optimization is available in specialized ML frameworks[98–100].

Uncertainty quantification

In physical systems, obtaining the parameter uncertainty, i.e., the degree to which the parameters are unknown, is nearly as important as obtaining the parameter estimates. Yet, most ML in acoustics neglects this. ML is grounded in the training-testing paradigm, in which the model parameters are estimated to minimize a loss function and then validated with test data. There is a consensus that ML models are more accurate than simple statistical models when making predictions due to their flexible nature, as nicely advocated in "Statistical modeling: the two cultures"[101]. The advent of new loss functions tailored to estimate probability distributions, combined with the progress in ML, leads to ML models that can estimate predictive uncertainty more accurately than simpler statistical models.

Uncertainty can be reducible (epistemic or statistical uncertainty) and irreducible (aleatoric or systematic uncertainty). The reducible uncertainty can often be reduced by, e.g., collecting more data and averaging, while the irreducible uncertainty can be mitigated by replacing the data model with a more robust alternative. Both uncertainties can be reduced by careful design, and an indication of whether this is successful can be obtained by analyzing the resulting uncertainty.

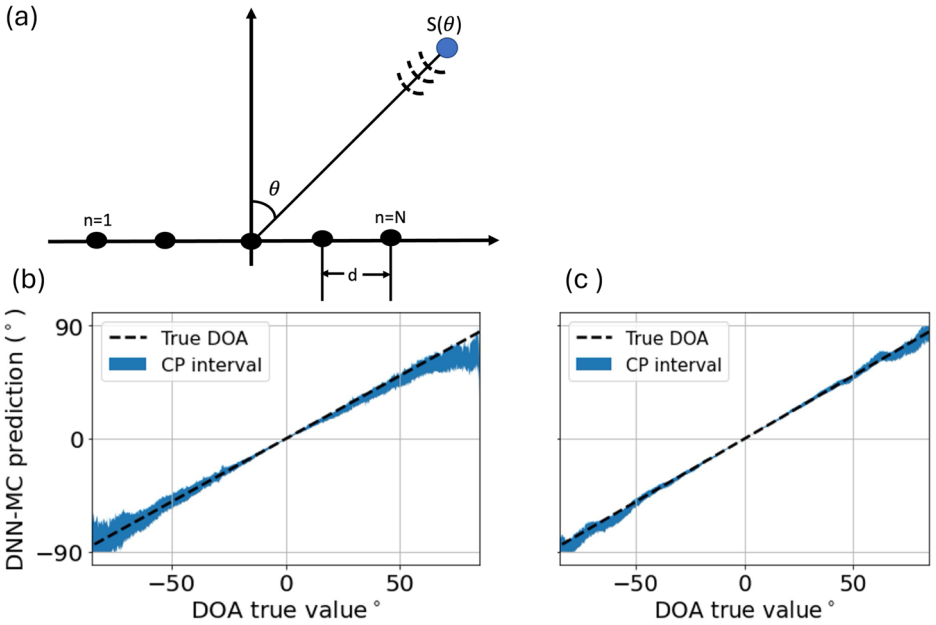

Fig. 4 | Using a deep neural network (DNN) for obtaining a prediction interval in beamforming. **a** Array setup with N sensor and one source impinging at direction \theta. **b** Conformal prediction interval (blue) versus true direction of arrival (DOA) for SNR of 0 dB. **c** As (**b**), but for a higher SNR of 20 dB. The true DOA is dashed.

Uncertainty quantification (UQ) involves identifying sources of uncertainty, assessing their impact on model outputs, and providing a measure of confidence in the model's predictions. UQ methods can generally be divided into Bayesian and frequentist methods.

Bayesian methods use a prior distribution, which describes our prior knowledge about the parameters, and a likelihood distribution, which describes the probability of the observed data given the parameters, to obtain a posterior distribution via Bayes' theorem[102,103]. The uncertainty intervals are then obtained from the posterior distribution, called credible intervals (region of the posterior distribution containing $1-\alpha$ of the probability). Credible intervals are used in Bayesian statistics to characterize the uncertainty of an unobserved parameter. For example, a 95% credible interval means there is 95% probability that the parameter lies within this range, given the data and the prior information.

Frequentists assume that the true unknown estimate is fixed and the uncertainty is quantified in terms of confidence intervals. Confidence intervals are derived only from sampled data, without a prior distribution. For a chosen confidence level $1-\alpha$, after running N tests with N confidence intervals, $1-\alpha$ of the confidence intervals are expected to include the true value. For example, a 95% confidence interval means that if the experiment were repeated many times, 95% of the intervals from those experiments would contain the true value. Methods for calculating confidence intervals include bootstrapping (repeated resampling)[104] or direct interval estimation by assuming an output distribution.

The credible intervals obtained using Bayesian methods represent the level of uncertainty associated with a random variable. Bayesian methods make assumptions for the prior distribution, which might be restrictive. This and non-linear forward models make a closed-form expression for the posterior distribution difficult to obtain. Sampling techniques can approximate the posterior distribution but lead to computational overhead.

Bayesian sampling has a rich history in acoustics[102,103,105–107]. These can provide an accurate sampling of the probability distributions, though often with some bias due to the choice of sampling parameters. They are computationally demanding, as accurate sampling requires many forward model runs. A simpler strategy is to perform UQ, providing a measure of uncertainty for the parameter estimate or observations. Although many UQ methods are available, we focus on interval estimation methods through the recently introduced prediction intervals with conformal prediction (CP)[108,109].

CP computes the prediction intervals in a few steps[110,111] as indicated in the example in Fig. 4. CP utilizes a parameter estimate plus a heuristic measure of uncertainty (scalar) to define a conformal mapping between this uncertainty scalar and the end points of a prediction interval that contains the true estimate with probability $1-\alpha$ based on *just one observation*. The conformal mapping refers to an unknown angle-preserving nonlinear mapping between these quantities, as the mapping is unknown and thus has to be learned via training data.

We demonstrate the approach with a simple example[109]. Consider estimating the direction of arrival (DOA) y from observations on an array of observations $\mathbf{x}$,

$$y = f(\mathbf{x}), \tag{11}$$

where the f could be any beamformer estimating a DOA, such as conventional beamforming or the DOA output of a neural network. We first train a neural network to give estimates of the DOAs. For a single observation i with input $\boldsymbol{x}_i$ received, a neural network estimates the mean DOA μ_i and the estimated variance σ_i^2 obtained by running the trained neural network with dropout, where the dropout is used for estimating a heuristic variance σ_i^2. This uncertainty estimate σ_i^2 is not guaranteed to satisfy the statistical coverage. CP can remedy this issue by calibrating the estimate using training data and conformal mapping.

The prediction interval for a single test point $\mathbf{x}_i$ with estimated μ_i and variance σ_i^2 is,

$$\mathcal{C}(\mathbf{x}_i, \alpha) = [\mu_i^k - \sigma_i q_\alpha,\ \mu_i^k + \sigma_i q_\alpha]. \tag{12}$$

Each DOA direction obtains the calibration factor q_α. To obtain q_α^k, we generate L realizations of $\boldsymbol{x}$ for random DOA $y^{\text{true}} \in [-90°, 90°]$ with random noise added at a given signal to noise ratio and estimate the μ^l and $(\sigma^2)^l$.

Defining a score function as $|\mu^l - y^{\text{true}}|/\sigma^l$, then we rank the L scores and pick the $1-\alpha$ ratio so that

$$\text{Prob}\left[\frac{|\mu^l - y^{\text{true}}|}{\sigma^l} \le q_\alpha\right] \ge 1-\alpha. \tag{13}$$

This determines the q_α for the whole dataset.

Figure 4 demonstrates the CP prediction on a simple DOA estimation for a 20-element linear array with half-wavelength spacing for one source. The data $\boldsymbol{x}$ are generated knowing the true DOA and adding noise to this sample. CP gives an uncertainty interval for just one observation, as seen for

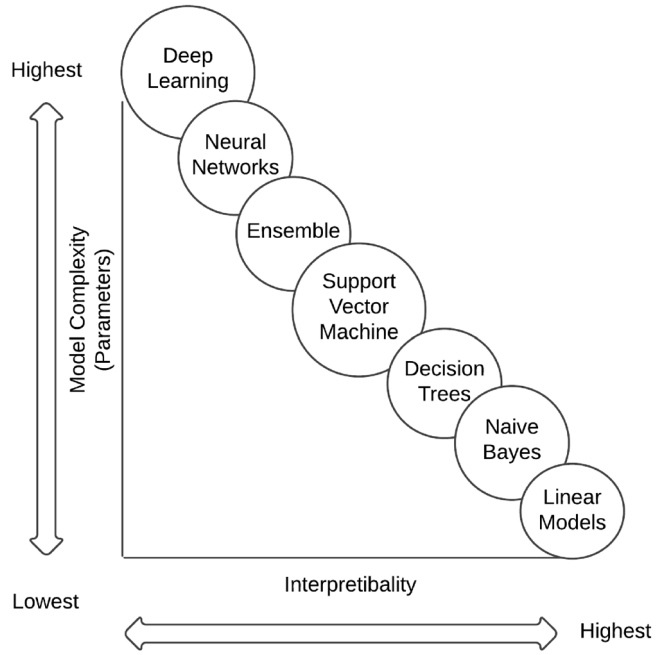

Fig. 5 | Model complexity vs. model interpretability. The larger the number of neurons, nodes, or higher-order fits, the more difficult it is to interpret the results.

varying across the observed direction of arrival, see Fig. 4(b) and (c). The uncertainty interval increases for lower SNR ratio, see (b) vs. (c).

Explainable AI

Larger and more complex models have recently become popular for many applications due to their higher accuracy than traditional ML models, automatic feature engineering, and ability to learn complex features. Understanding how models make predictions becomes crucial for building trust and enabling human oversight as these models grow in complexity. Models are often described as "black box" diagrams because of their complexity and lack of transparency in their predictions. Examples of the model complexity are shown in Fig. 5, where models are plotted as a function of complexity (i.e., number of hyperparameters) vs. the level of interpretability. Interpreting how these ML models achieve their success can be non-trivial. Statistical performance measures such as accuracy do not provide enough key information for model interpretation or reliability. Recently, AI has sparked increasing research interest in explainability and explainable AI[112,113], which can be broken down into a few different forms: 1) data explainability, 2) model explainability, 3) feature-based explainability, and 4) example-based explainability. We briefly discuss these varieties and a few explainable AI techniques.

Data explainability. Data explainability focuses on the data used to train and input into a model. This includes identifying biases and under-represented samples. One approach is data visualization to identify patterns, trends, and insights. Visualizing the interconnectivity between desired outputs and inputs is difficult for larger datasets thus a popular approach is to use dimensionality reduction techniques such as principal component analysis (PCA), t-distributed stochastic neighbors (t-SNE), uniform manifold approximation and projection (UMAP)[114], TriMAP[115], dictionary learning, autoencoders, or pairwise controlled manifold approximation (PacMAP)[116]. Algorithms provide unique dimensionality reduction techniques focusing on finding relations in the data mapped to a new dimensionality that reduces the variance. It should be cautioned that they may not represent the higher dimensionality correctly and could result in nonexistent relations based on the number of newly mapped dimensions. Further details can be found in Chapter 6 of the AcousticsML repository.

Model explainability. An ML model's capacity and complexity are due to the number of learnable parameters. As the number of parameters increases, it becomes difficult to interpret a model's prediction (see Fig. 5). Although larger models can produce highly accurate results, understanding model predictions can be ambiguous, thus referring to these models as "black box" models. Examples of black box models include deep neural networks, convolutional neural networks (CNN), and large gradient boosting models. Alternatively, smaller, more transparent, and interpretable models for prediction are referred to as "white box" models. White box models include linear models, Gaussian mixture models (GMM), Naive Bayes models, and decision trees. These models offer rule-based decisions and simple equations learned from training data. The trade-off for higher interpretability and transparency is prediction accuracy in these models. The choice of model complexity depends on the application and the desired outcome.

Feature-based explainability. Features are measurable properties that are input to a system. These inputs are related to a particular data sample that an ML model can interpret to make predictions. Features are typically independent and provide specific details about the sample. In acoustics, features can be 1D representations (e.g. intensity, energy, timbre, etc.) or 2D representations (e.g. spectrograms) for a given sample. Descriptions of features and feature selection are in Chapter 2 of the AcousticsML repository.

Feature selection can reduce complexity and improve ML model accuracy. Providing vague or correlated information can lead to misleading results, such as passing vague information about energy in frequency bands for classification. To improve model accuracy, we can consider the relative importance of each input for the given outputs, known as feature importance. One approach is to use prior statistical tests (e.g., Chi-squared test), or correlation tests to eliminate features before training a model to reduce the complexity. These approaches can help identify features with little variability that do not contribute significantly to the output. Feature importance can be learned after training a model using random permutations, feature nulling, or Recursive Feature Elimination. These techniques compare the accuracy of the model predictions before and after altering the inputs. Similarly, feature weights (i.e., coefficients for linear regression or number of occurrences for decision trees) may be beneficial in determining feature importance. For further information on feature selection approaches, see ref. 12.

Example-based explainability. Example-based explainability aims to identify how models make predictions by looking at global or local predictions. Global predictions provide a broad analysis of inputs, while local predictions focus on how smaller sample sets are predicted from the given inputs. Global techniques for identifying how models make predictions include random permutations[117], accumulated local effects[118], or partial dependence-based feature importance [ref. 4, Section 18.6.2]. Local techniques include Local Interpretable Model-Agnostic Explanations (LIME)[119], Shapley Additive explanations (SHAP)[120], or anchors[121]. Some of these techniques are covered in the tutorial notebooks for unsupervised, supervised, and DL models in Chapter 6 of the AcousticsML repository.

Despite its growing potential, explainable AI has several limitations and challenges. One of the most fundamental issues is the trade-off between model accuracy and interpretability: complex models such as DNNs, which outperform simpler ones but operate as "black boxes," offer little insight into how decisions are made. Although *post hoc* explanation techniques such as SHAP or LIME attempt to provide interpretability, these methods can produce inconsistent or misleading interpretations that do not reflect the internal logic of the model. The interpretability of model predictions often depends on the specific application, and the choice of explanation method is typically guided by the user's particular needs and objectives. Another key challenge is integrating domain-specific knowledge to enhance both model performance and interpretability without introducing bias or overfitting.

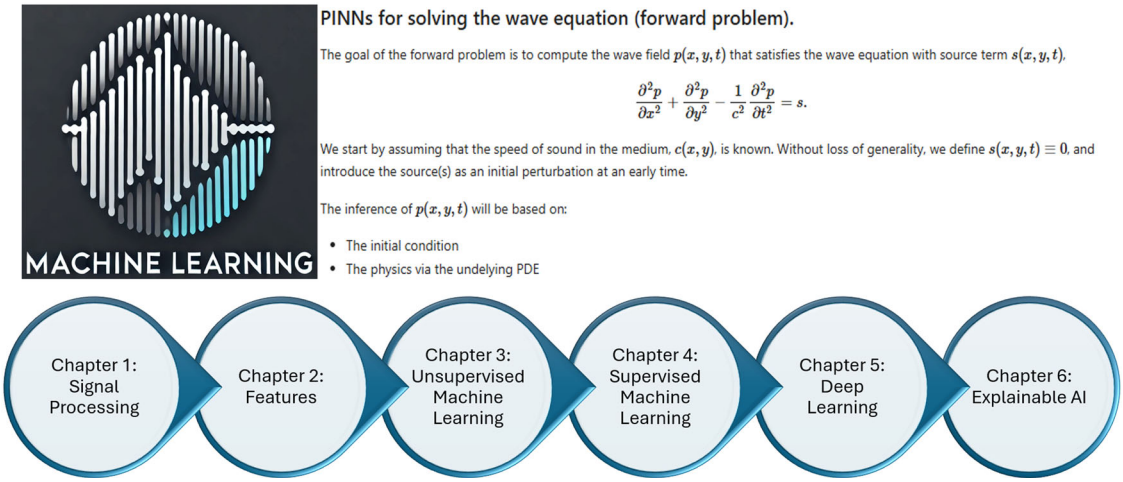

Fig. 6 | Overview of AcousticsML. AcousticsML includes ML examples for Acoustic Applications and is split into chapters (bottom), which include descriptions of the application and ML model used (top).

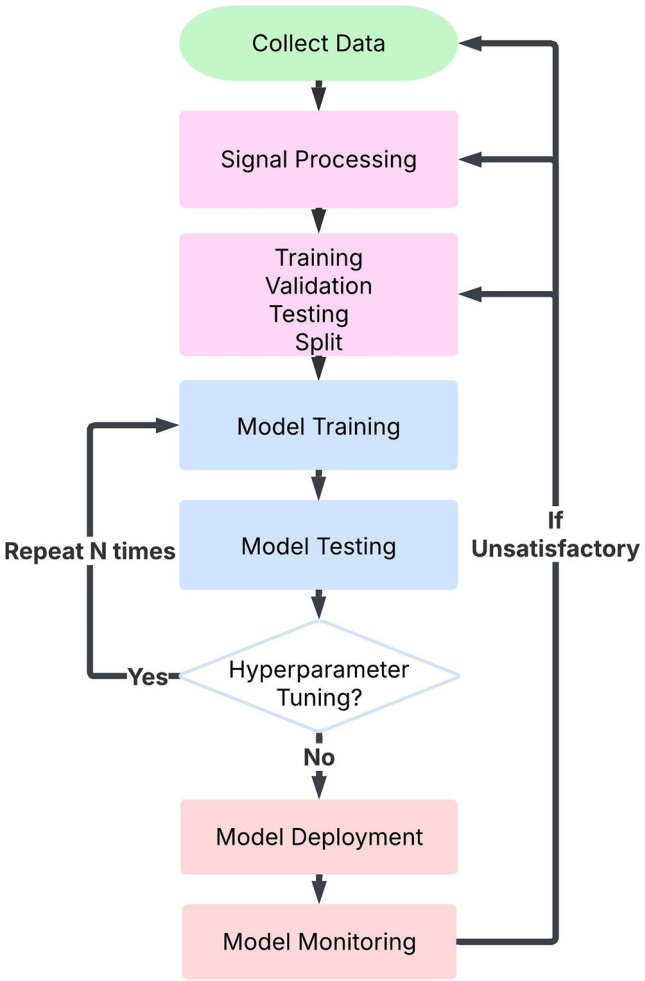

Fig. 7 | Workflow of ML models. Models are trained through a series of steps and iterated based on a performance metric to improve the model's capability.

The current lack of standard evaluation metrics for explanations complicates efforts to compare or validate explainable AI methods. In certain acoustic applications, this becomes particularly problematic, as flawed or opaque explanations can undermine trust, impede accountability, and lead to invalid predictions. The evolving nature of AI necessitates explanations that are adaptive and context-aware, capable of keeping pace with changing models, data distributions, and evolving ethical standards.

The AcousticsML repository

Due to the substantial amount of acoustic applications and ML models, the AcousticsML repository addresses particular topics that can be applied to broader applications. The AcousticsML repository provides an overview of the topics covered at the top of the page, followed by a brief discussion of the model used, how models are initiated and trained, and further references for further information (Fig. 6). Notebooks in the AcousticsML repository are grouped into six chapters that introduce ML techniques and applications that can be extended to other applications.

Chapter 1) Short introduction to signal processing techniques. Signal processing enables models to learn from data efficiently and effectively. Though the notebook does not provide all information on the theory of waves, ray propagation, or acoustic modes, it gives a brief background, additional links, and learning resources.

Chapter 2) Feature extraction and selection from acoustic data. The notebooks briefly discuss 1D statistical measures and 2D spectral features that can be input into a model.

Chapter 3) Unsupervised ML algorithm applications. These algorithms learn from the data and do not require prior labels, revealing patterns and relations in the data that may be difficult to recognize.

Chapter 4) Supervised ML algorithm applications. These models learn from labeled data to predict desired outcomes.

Chapter 5) Deep learning model applications. Notebooks emphasize DL through PyTorch and demonstrate CNNs, GANs, and PINNs.

Chapter 6) Explainable AI techniques for unsupervised, supervised, and DL models. Explainable AI techniques are significant to today's acoustic applications, providing insights into model prediction and human interpretation of data.

The AcousticsML repository follows a typical ML workflow illustrated in Fig. 7. First, data are selected and preprocessed using signal processing techniques. This process provides quality assurance and quality control (QA/QC) to remove random noise in the data, select relevant data, or improve model performance. The features are then extracted from the data using several quantitative and qualitative techniques to train the ML model algorithms.

Several ML model architectures are available including unsupervised, supervised, and DL. The choice of model and implementation depends on the application, the amount of labeled data available, and the desired implementation for prediction (e.g., regression or classification). Notebooks do not cover which model architecture to use but provide examples of some

available models to get started. Trained models are tested with additional available data to ensure they operate efficiently and successfully. This step is vital in determining whether a model was trained effectively and is generalizable. Training and testing models can occur repeatedly, adjusting hyperparameters to test performance. An additional post-processing step can be included to improve model prediction, but this is left out in this example.

Once a trained model has satisfactory performance, it can be deployed. Deployed models can be monitored with newly collected data to observe biases and prediction errors. If performance deteriorates over time, it may be due to noisy data, new unseen observations (i.e., observations not present in the training dataset), or the limited ability of the model. In any case, training a new model with the newly collected data, different signal processing techniques, or a new model architecture may be advantageous. As an important note, there is no one way to approach a problem with ML; instead, sets of techniques and model architectures can be applied effectively.

Application example workflows

We highlight several acoustic examples demonstrating ML pipelines. Pipelines describe the procedure of preprocessing, data input, and output from an ML model, as shown in Fig. 7. The AcousticsML repository examples provide initial workflows and procedures to learn and apply to other applications. Four applications and unique machine-learning models are selected to discuss their procedures and applications. The code for each application is available as a Jupyter Notebook in the AcousticsML repository.

Acoustic classification

In acoustic classification tasks, acoustic data are assigned to predefined or learned classes according to the features of each data sample. Classification can be applied to distinguish different kinds of animal calls[122,123], identify musical instruments[124], classify environmental sounds (e.g., anthropogenic noises or bioacoustics)[125], or monitor for malfunctions in machinery [126–128]. Classification leverages existing datasets to predict or classify new or unseen datasets into distinct classes based on similarity or probability. Deep learning has become a powerful tool for sound classification, enabling models to automatically learn complex features from raw audio data without manual feature engineering. CNNs, recurrent neural networks (RNNs), and, more recently, transformers are commonly used architectures for this task, leveraging large datasets and identifying non-linear patterns between inputs to make predictions. Choosing which model to implement depends on several factors, including the complexity of the problem, the amount of data available for training, and the choice of features to be used for prediction.

A challenge in acoustic classification is ensuring that training data is available to represent each class adequately. A model cannot classify something on which it is not trained; hence, the more diverse a classification problem (e.g., predicting a bird species from birdsong), the more training examples from each class are required to train a model. When little data is available to train a model, anomaly detection can first be used to identify sounds of interest, and unsupervised or manual labeling can be used to group similar sounds in subsequent analysis.

The choice of features used to classify acoustic data may also impact the performance of a model. For example, models may have difficulty differentiating between two animal species with similar vocalizations (i.e., similar frequency upsweeps, duration, loudness, etc.). Acoustic classification, therefore, depends on thoughtful feature engineering, where the representation of sound data directly impacts models' discriminative power. For instance, time-frequency representations like spectrograms may outperform simple frequency spectra by preserving temporal dynamics that reveal distinctive patterns in acoustic events. Feature engineering requires systematic experimentation to identify which acoustic features or transformations most effectively capture the discriminative characteristics of the target sounds.

Here, we describe an example of an acoustic classification approach and demonstrate how the model choice, amount of data used for training, and feature representation impact performance. Chapter 4 in the AcousticsML repository includes several Jupyter Notebooks demonstrating the application of classification to labeled acoustic data.

Dataset. Audio classification is demonstrated on data from the Audio Modified National Institute of Standards and Technology (AudioMNIST)[19]. This dataset consists of 30,000 recordings of spoken numbers in English, with 50 repetitions from 60 speakers of different nationalities. The duration of audio clips ranges from 0.3–1 s, with 3, 000 examples recorded for each spoken number. All audio clips are sampled at 22.05 kHz, and two recording locations are used.

Feature extraction. Two feature extraction techniques were evaluated to identify key differences between each spoken number. Both techniques transform the raw audio data into 1 by 1026 feature vectors. The first technique, a Fast Fourier Transform (FFT) approach, extracts features from frequency components in each audio clip from a spectrogram. Specifically, a window size of 1024 samples with 50% overlap is used to generate each spectrogram; then, the mean and standard deviation are taken from each frequency bin to produce a feature vector. Similarly, the second feature extraction technique uses Mel-frequency cepstral coefficients (MFCCs) to transform the spectrogram into a Mel scale before extracting features from each frequency bin using the mean and standard deviation. Examples of FFT and MFCC spectrograms are shown in the middle and bottom rows of Fig. 8.

Machine learning model. Two supervised ML models, a decision tree and a random forest [ref. 4, Section 18], are trained and evaluated using the AudioMNIST dataset. These models are prone to overfitting but provide simple and explainable decision-based predictions to classify audio samples. Decision trees (DTs) divide the input data by evaluating features, such as frequency, amplitude, or other acoustic characteristics, at each node, creating increasingly specific subsets of the data. This process continues until the data is grouped to minimize variation in the target at the leaf nodes. A random forest (RF) consists of multiple decision trees, each trained on different random subsets of the acoustic data, with each tree making an independent prediction. The final output of an RF is determined by combining the predictions of all trees, often through averaging or voting. This ensemble approach enhances accuracy and reduces the risk of overfitting to particular sound characteristics or noise patterns. At each split in a decision tree, features are selected based on their ability to maximize information gain, typically using measures like entropy, which helps ensure the tree captures the most meaningful acoustic distinctions between sound classes or patterns.

Training a model. Audio clips for each spoken number are divided into even subsets, with 80% of samples used for training and 20% for testing. Clips from each speaker are selected at random. Features are extracted from each clip using the two methods described previously. Training and testing data are normalized and standardized using the feature-wise mean and standard deviation from the training data. Hyperparameters for the decision tree and random forest are chosen using Bayesian Optimization provided in Optuna[99]. The DT hyperparameter space includes a maximum depth ranging from 4 to 100 branches, metrics for split quality (e.g., Gini index, entropy, log loss), and strategies to choose the split at each node (i.e., best or random). The RF hyperparameter space additionally includes the number of estimators ranging from 2 to 50. Models are trained with 3-fold cross-validation on the training data [ref. 4, Section 4.5]. The model with the highest validation accuracy is chosen as the best model. This process is repeated for 20 instances with varying hyperparameters for each model and feature choice.

Model evaluation. Overall, the DT and RF model performance is shown in Table 1. The MFCC feature representation has considerably greater

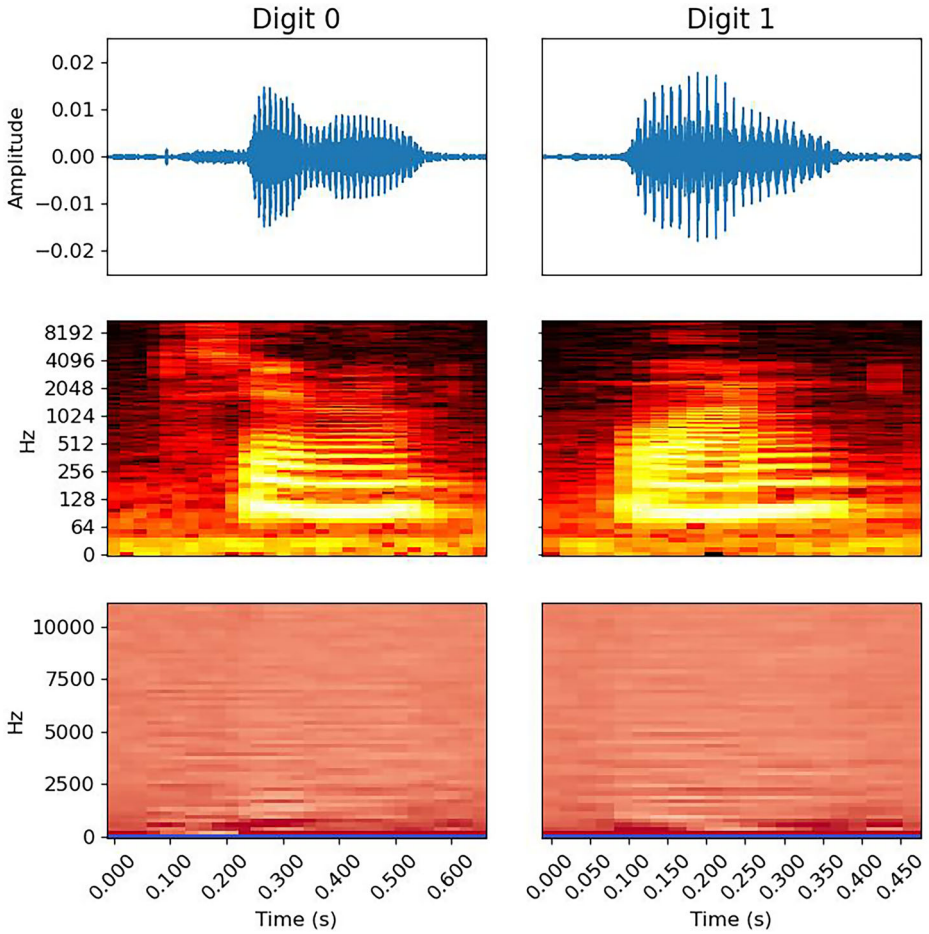

Fig. 8 | Spectra features from audio examples. Examples of audio data (top row) from the AudioMNIST dataset, with features extracted by FFT (middle row) and Mel-frequency cepstral coefficients (bottom row).

performance than the FFT feature representation, and RF models outperformed DT models with both feature representations. The RF model with MFCC feature representation had the highest overall performance.

Classification performance is visualized using a confusion matrix, as shown in Tables 2 and 3. Diagonals and off-diagonals represent the frequency of correct and incorrect classifications, respectively. The sum of each row indicates the true number of samples for the class, while the sum of each column indicates the number of times a class label is predicted. Confusion matrices are useful for identifying where a model may have prediction biases. For example, the RF trained with FFT features (Table 2) has many off-diagonal values when predicting class "one". In this instance, the model predicts numbers "zero", "two", "four", or "nine" as number "one". Analyzing rows of the confusion matrix shows the model has difficulty with the spoken digits "zero" and "two". Conversely, many off-diagonal values are nearly zero for the RF trained with MFCC features, indicating that the model has near-perfect accuracy, precision, and recall. Although the confusion matrix identifies inaccuracies in model prediction, it does not provide specific examples or a deeper look into why these errors occur. To address such performance issues, detailed inspection of data examples from classes with low accuracies is necessary to understand why a model struggles.

Acoustic data exploration with unsupervised ML

In cases involving large acoustic data sets—for instance, those generated by continuously recording sensor arrays—the sheer volume of data can make manual analysis impractical. Given time and resource constraints, scrutinizing every audio snippet for meaningful insights may be unfeasible. To overcome this challenge, unsupervised ML techniques can be employed to analyze and categorize the data systematically. Unsupervised ML involves algorithms and models that learn patterns and structures from data without explicit supervision or labeled target outputs[1,4]. Specifically, clustering algorithms seek to discover similar examples within the data and are used for data mining and exploratory data analysis[1,4]. Clustering results can then guide further analysis, such as targeted review of specific segments or features of interest, or removal of unwanted data types.

Many clustering algorithms perform more effectively when working with lower-dimensional data[129]. This is particularly relevant for acoustic datasets, which typically contain thousands or millions of features when represented as time series, spectrograms, scalograms, or energy envelopes. The high dimensionality of these representations—whether discrete samples in a time series or time-frequency bins in a spectrogram—presents computational challenges for standard clustering approaches. To address this limitation, we present a workflow that combines autoencoders (deep neural networks specialized in dimensionality reduction) with clustering algorithms. A code implementation of this workflow can be found in Chapter 3.5 in the AcousticsML repository. The dimensionality reduction of autoencoders has been paired with both supervised and unsupervised ML workflows in a variety of acoustic[130–137] and seismic[28,138–140] settings.

Dimensionality reduction with autoencoders. A sample from an acoustic data set can be represented as a vector $\mathbf{x} = [x_1, \ldots, x_N]^\mathsf{T} \in \mathbb{R}^{N\times 1}$, where each feature corresponds to an element x_n in the vector $\mathbf{x}$ which describes a point in N-dimensional space. Directly clustering high-dimensional data is vulnerable to the "curse of dimensionality":[4,129] as the dimensionality of the input data increases linearly, the number of data points required to maintain sufficient sampling density increases exponentially. Additionally, clustering algorithms can give less meaningful results as dimensionality increases, making clustering in high dimensions challenging and unreliable[129].

Table 1 | Model performances for each feature representation

Metrics	FFT		MFCC	
	DT	RF	DT	RF
Accuracy	0.75	0.93	0.87	**0.99**
Precision	0.75	0.93	0.87	**0.99**
Recall	0.75	0.93	0.87	**0.99**
F1 Score	0.75	0.93	0.87	**0.99**

Table 2 | Confusion matrices for the random forest trained with FFT features

	Predicted Label									
True Label	0	1	2	3	4	5	6	7	8	9
0	516	5	21	7	4	2	0	31	4	6
1	1	569	0	0	5	2	0	0	0	22
2	26	12	536	17	5	0	0	2	7	0
3	6	0	6	551	0	2	2	0	20	2
4	3	21	0	0	574	5	0	2	0	2
5	1	4	1	1	6	574	0	0	0	16
6	0	0	0	3	0	2	600	1	14	0
7	25	0	3	4	0	1	2	548	1	5
8	0	0	4	16	0	0	3	2	557	1
9	2	45	1	0	1	8	0	0	0	552

Table 3 | Confusion matrices for the random forest trained with Mel-frequency cepstral coefficients (MFCC) features

	Predicted Label									
True Label	0	1	2	3	4	5	6	7	8	9
0	579	3	7	4	0	0	0	2	0	1
1	2	590	0	0	0	0	0	0	0	7
2	3	0	596	2	0	0	0	1	2	1
3	7	0	4	574	0	0	0	0	4	0
4	2	2	0	0	602	0	0	1	0	0
5	0	1	0	0	4	597	0	1	0	0
6	0	0	0	0	0	0	617	3	0	0
7	0	0	1	0	0	1	0	586	0	1
8	1	0	0	2	0	1	1	1	577	0
9	0	9	0	0	0	0	0	1	0	599

A popular approach is principal component analysis (PCA), which projects higher-dimensional data into a lower-dimensional space [ref. 4, Section 20.1]. However, PCA is a linear method and may not be effective for data with complex, nonlinear structures. An alternative model that can capture nonlinear relations is an autoencoder, a neural network that learns to encode data into a latent, lower-dimensional representation[4]. A typical autoencoder architecture is shown in Fig. 9 and consists of three components: an *encoder*, a *bottleneck*, and a *decoder*[4]. First, the encoder maps input data, like spectrograms, from a data space X into a latent feature space Z by f_θ: $X \to Z$, where θ are the neural network parameters. Next, the decoder attempts to reconstruct X from Z by $g_\theta : Z \to X'$. An entire

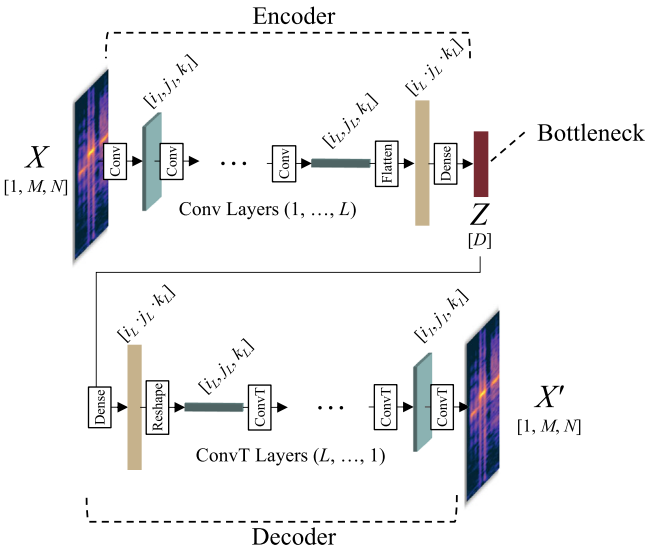

Fig. 9 | Architecture of an autoencoder network. The encoder maps input data X to a latent representation Z, and the decoder reconstructs the input from the latent representation. Embeddings in Z are a reduced dimensionality representation of the input data and can be used for clustering or other tasks.

forward pass through the autoencoder is represented as

$$F_\theta : X \to Z \to X', \quad F_\theta = g_\theta \circ f_\theta. \tag{14}$$

The autoencoder is trained by iteratively updating θ through back-propagation (Ref. 3, Section 6.5) to minimize a loss function defined as the reconstruction error between X and X', e.g., the mean squared error MSE($X - X'$). In minimizing the error, the autoencoder learns the salient features of X and accurately embeds them in Z, enabling subsequent tasks like clustering to be performed in the lower-dimensional latent space. A successor to autoencoders, the variational autoencoder (VAE), enables data generation from the latent space and is discussed in a previous section, "Variational autoencoder".

An example of an autoencoder applied to a spectrogram is shown in Fig. 10. The spectrogram contains 129 time bins and 129 frequency bins, producing 16,641 features. The encoder maps the spectrogram to a latent representation with 32 features, and the decoder reconstructs the spectrogram from the latent representation back to the original 16,641 features.

Clustering. After reducing the data dimensionality with the autoencoder, clustering algorithms can be applied to the latent space to group similar data points. One of the most common clustering algorithms is k-means, which partitions data into K clusters by minimizing the sum of squared distances between data points and their cluster centroids [ref. 4, Section 21.3]. However, k-means makes assumptions about the data, such as isotropic clusters and balanced cluster populations, which may not hold in practice. A more general approach is to model clusters as a mixture of K multivariate Gaussian distributions. GMMs can capture anisotropic and imbalanced clusters and yield probabilities that each point belongs to a particular cluster, enabling a more in-depth analysis of the clustering results [ref. 4, Section 21.4].

Determining the optimal number of clusters, K, is a challenging problem in unsupervised machine learning. Furthermore, when autoencoders are used in conjunction with clustering in the latent space, care must be taken to ensure that clustering results map to meaningful features in the original data space. The choice of K significantly affects the clustering results, and selecting an inappropriate number of clusters can lead to suboptimal or misleading results. The choice for K should be evaluated through both qualitative inspection and quantitative metrics. Clustering results are qualitatively

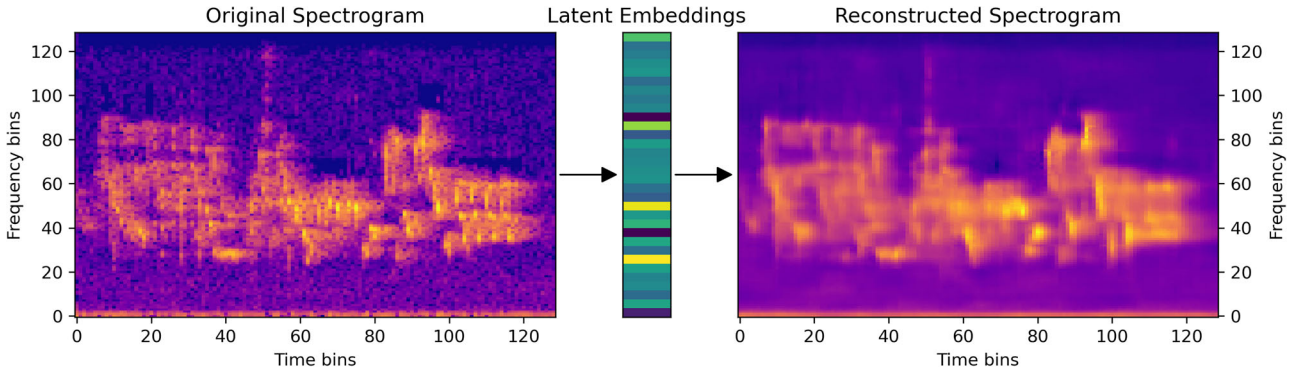

Fig. 10 | Spectrogram latent embedding and reconstruction example. A spectrogram (left) is provided to a convolutional autoencoder trained on birdsong. The 32-dimensional latent embeddings (center) contain the salient features required to produce the reconstructed spectrogram (right).

evaluated for similarity of data points to their respective cluster centroids and to data points within a cluster, and should be done in both the latent and data spaces. Quantitative evaluation can be performed using metrics such as the gap statistic[141] and silhouette scores [ref. 4, Section 21.3.7.3], which measure the compactness and separation of clusters.

Generative modeling for spatial audio

Generative models aim to model the distribution of data and use ML models to generate more synthetic data that seems to be from the same distribution. A typical ML method for generative modeling is GMM, assuming the data is a mixture of a Gaussian distribution and can sample from each distribution to generate a new data sample.

Generative adversarial networks for room acoustics. Chapter 5.2 presents a Jupyter Notebook using a generative adversarial network to generate room impulse responses.

Generating Room Impulse Responses (RIRs) is an important research topic due to their role in capturing the acoustic characteristics of an environment. Reverberant sound, which reflects the layout and materials of the environment, is crucial in human spatial awareness. However, in applications such as automatic speech recognition, this reverberation can act as noise, degrading system performance by masking speech clarity or introducing distortions. RIRs serve as a fundamental representation of sound propagation in a room, assuming the system behaves as linear and time-invariant. Despite their significance, RIRs are typically challenging to measure directly, as they require specialized setups, including loudspeakers to emit a sine sweep and microphones to record the response at various locations. Post-processing is necessary to extract the RIR, adding complexity to their practical use.

Generating RIRs through ML methods offers an attractive solution to overcome these challenges and improve the robustness of acoustic models in real-world environments.

In this example, we use the BUT ReverbDB dataset[20], which contains a dataset of real room impulse responses (RIRs). Following IR-GAN architecture[40], the generator comprises 5 layers of 1D deconvolution, and the discriminator consists of 5 layers of 1D convolution. Based on a standard GAN training setup, the model can generate plausible room impulse responses.

Personalized HRTF modeling with implicit neural representations. Chapter 5.3 of the AcousticsML repository presents a Jupyter Notebook on using implicit neural representations to model HRTFs across datasets.

Spatial audio plays a pivotal role in creating immersive experiences through headphones or VR headsets, allowing users to perceive the direction and distance of the sound. By incorporating individual uniqueness into auditory perception, spatial audio rendering can significantly enhance the sense of immersion. A key aspect of this task is to predict personalized head-related transfer functions (HRTFs), which describe the spatial filtering effects of human geometry for accurate sound localization.

Due to the resource-intensive nature of HRTF measurements, existing databases have a limited number of subjects, which poses challenges for data-intensive machine learning models. HRTFs are inherently high-dimensional, encompassing numerous spatial locations and frequency bins per subject.

Human geometry input can be in several formats, including anthropometric measurements, ear images, or a scanned head mesh, arranged from lower dimension to higher complexity. Usually, these data are fed into an encoder to map to a latent space, and then the HRTF is decoded from the latent space as the output. Typical ML models include autoencoders, variational autoencoders, and generative adversarial networks (GAN)[142].

Personalized HRTF modeling involves two tasks: interpolation and personalization. As measurement is time-consuming, the number of locations is usually limited. With generative modeling, getting the HRTF at arbitrary locations would be ideal. These include generating the HRTF, given the existing measured HRTFs as conditions, or modeling the whole distribution of the HRTFs for various subjects.

In the Jupyter Notebook, we present the use of INRs to interpolate the HRTFs[67]. The HRTF of each person at azimuth θ and elevation angle ϕ is modeled with the output of the generator $G(\theta, \phi, z)$, where the latent vector z represents the personalized HRTF of each person. The training and generation process is illustrated in Fig. 11.

We use the HUTUBS dataset[23] and build the INR with a 2-layer multilayer perceptron with 2048 nodes in each layer. The model is trained in an autodecoder fashion, where the latent vector z is first assumed at the origin and then updated with the negative gradient.

$$\mathbf{z} = \mathbf{z}_0 - \nabla_{\mathbf{z}_0} \mathcal{L}_{\mathrm{MSE}}\big(\mathbf{x}, G(\cdot, \cdot, \mathbf{z}_0)\big). \tag{15}$$

With the new z, the generator G is updated with the ℓ_2 distance between the generated HRTF and the ground truth.

$$\mathcal{L} = \mathcal{L}_{\mathrm{MSE}}(\mathbf{x}, G(\cdot, \cdot, \mathbf{z})). \tag{16}$$

Physics-informed neural networks (PINN)

Chapters 5.4 and 5.5 include two Jupyter Notebooks to solve forward and inverse problems in acoustics using PINNs. These notebooks use a finite difference solver (included in the repository) to generate data and ground truth solutions.

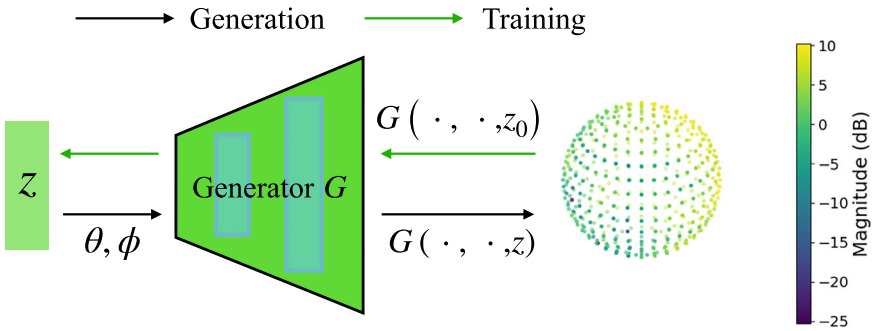

Fig. 11 | HRTF modeling with implicit neural representation. The colormap corresponds to the magnitude in the logarithmic (dB) domain at the 2 kHz frequency bin.

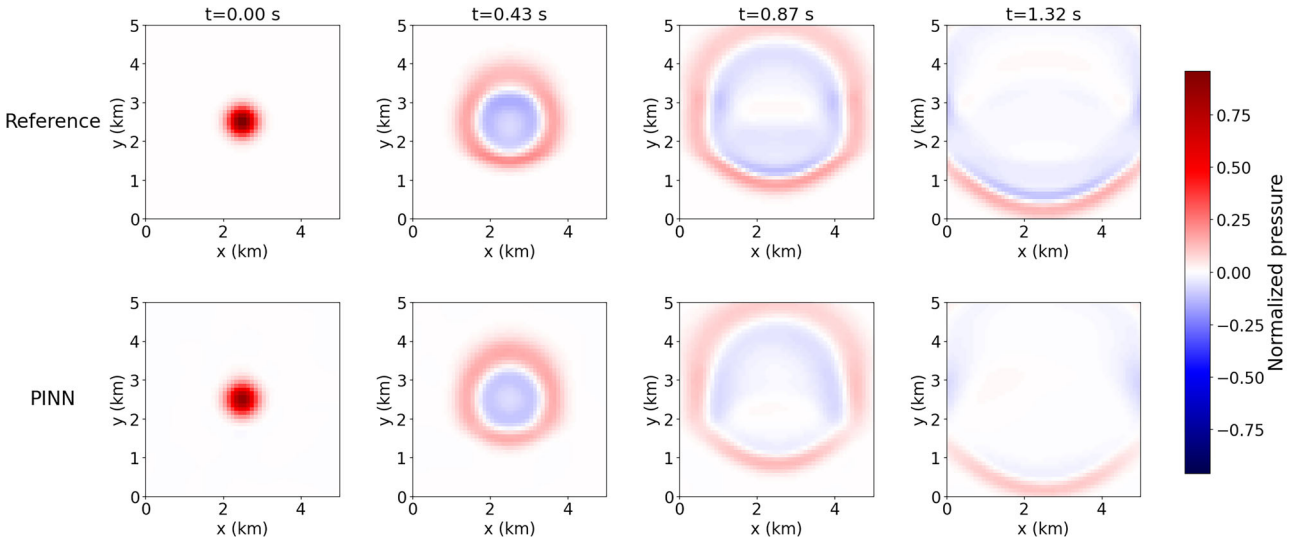

Fig. 12 | Forward solution of the wave equation in a 2D domain with a stratified wave speed and a Gaussian pulse as the initial condition. Top row: reference solution. Bottom row: PINN estimation.

Problem formulation. In Chapter 5.4, the goal is to solve the time-domain wave equation,

$$\nabla^2 p(\mathbf{r}, t) - \frac{1}{c^2(\mathbf{r})} \frac{\partial^2 p(\mathbf{r}, t)}{\partial t^2} = 0, \tag{17}$$

given a known wave speed, $c(\mathbf{r})$. It is assumed that the wavefield at some initial time steps,

$$p_0 = p(\mathbf{r}, t) \quad \text{for} \quad t \le t_0, \tag{18}$$

where t_0 is an early time for which the wave has not yet propagated far, is known. These snapshots contain the source position, shape, and the early wave propagation. The domain boundaries are considered absorptive so that there are no reflected waves. With this information, the goal is to compute the wave field $p(\mathbf{r}, t)$.

Loss function. The neural network used to represent the wave field is denoted $\hat{p}(\mathbf{r}, t; \boldsymbol{\theta}_p)$. The network's input is the spatiotemporal coordinates, $(\mathbf{r}, t)$, and its output is the computed wavefield. The network parameters $\boldsymbol{\theta}_p$ are tuned during training to minimize a loss function composed by the weighted sum of two terms,

$$\mathcal{L} = \lambda_{\text{pde}} \mathcal{L}_{\text{pde}} + \lambda_{\text{ic}} \mathcal{L}_{\text{ic}}. \tag{19}$$

The first one is the physics term, which constrains the network's output to satisfy the wave equation

$$\mathcal{L}_{\text{pde}} = \frac{1}{n_{\text{pde}}} \sum_{i=1}^{n_{\text{pde}}} \left\| \nabla^2 \hat{p}(\mathbf{r}_i, t_i; \boldsymbol{\theta}_p) - \frac{1}{c^2} \frac{\partial^2 \hat{p}(\mathbf{r}_i, t_i; \boldsymbol{\theta}_p)}{\partial t^2} \right\|^2, \tag{20}$$

where $(\mathbf{r}_i, t_i)$, $i = 1, \ldots, n_{\text{pde}}$ are stochastically sampled over the spatio-temporal domain during training, with n_{pde} user-chosen. The second term imposes the initial condition,

$$\mathcal{L}_{\text{ic}} = \frac{1}{n_{\text{ic}}} \sum_{j=1}^{n_{\text{ic}}} \left\| \hat{p}(\mathbf{r}_j, t_j; \boldsymbol{\theta}_p) - p_0(\mathbf{r}_j, t_j) \right\| \tag{21}$$

where $t_j \le t_0$ and $(\mathbf{r}_i, t_j)$, $j = 1, \ldots, n_{\text{ic}}$ are points sampled at early times.

This is a 'soft-constrained' formulation, meaning that the constraints guide the training but are not enforced in a hard way. Such soft-constrained PINNs are easy to formulate, but the learned function might not satisfy the

conditions exactly. Constraints can also be imposed using the neural network as part of a solution ansatz.

Figure 12 presents an example of using PINNs for solving the wave equation in a medium with a stratified wave speed. The initial condition is a Gaussian pulse in the center of the domain. A PINN is trained to minimize the loss of Eq. (19). After training, the output of the PINN approximates the wavefield, $p(\mathbf{r}, t)$. This example corresponds to Chapter 5.4.

Balancing the loss function. The weights λ_{pde} and λ_{ic} balance the two terms in the loss function. The choice of the weights is delicate, and manually finding weights that properly balance the loss terms can be difficult. In the notebooks, we implement an annealing algorithm[88] to automatically choose λ_{pde} and λ_{ic}. The weights are chosen based on the gradient of each loss term with respect to the network parameters to prevent an imbalance of the back-propagated gradients during training.

Network architecture. For the PINN architecture, a fully connected network with hyperbolic tangent activation functions, three layers, and 64 units per layer is chosen. This type of simple architecture is often used for PINNs, and it is convenient for this example.

Fourier features. Neural networks (not only PINNs) often suffer from *spectral bias*, i.e., they struggle to learn the high-frequency content of functions. High frequencies can be associated, for example, with function discontinuities or jumps, such as edges in an image. This is particularly relevant in the case of PINNs for acoustics because acoustic sources tend to contain a broad spectrum of frequencies. The use of Fourier features has been proposed as a way of alleviating the spectral bias of PINNs in multiscale problems[91]. A Fourier mapping of a network with inputs $\mathbf{r} \in \mathbb{R}^{n_{\text{in}}}$ can be computed as

$$\boldsymbol{\gamma}(\mathbf{r}) = \begin{bmatrix} \cos(\mathbf{B}\mathbf{r}) \\ \sin(\mathbf{B}\mathbf{r}) \end{bmatrix} \tag{22}$$

where $\mathbf{B} \in \mathbb{R}^{n_{\text{ff}} \times n_{\text{in}}}$, and n_{ff} is the number of Fourier features in the mapping. The entries of $\mathbf{B}$ are sampled from a normal distribution with variance σ^2. It has been shown, however, that the choice of σ^2 is not straightforward since it depends on the frequency content of the function to be approximated[91], and different σ^2 should be chosen for the spatial coordinates and the temporal ones. In the notebook, we change the formulation slightly and compute the Fourier mapping as a cosine transform with an offset,

$$\boldsymbol{\gamma}(\mathbf{r}) = \cos(\mathbf{B}\mathbf{r} + \mathbf{b}), \tag{23}$$

where $\mathbf{B} \in \mathbb{R}^{n_{\text{ff}} \times n_{\text{in}}}$ and $\mathbf{b} \in \mathbb{R}^{n_{\text{ff}}}$ are treated as network parameters that are learned during training. This $\boldsymbol{\gamma}$ is then input to the first layer of the neural network.

Causal training. A limitation of the original PINNs formulation is their inability to respect the spatio-temporal causality of the modeled physical system[143]. PINNs modeling transient phenomena are often biased toward minimizing the residual at later times before learning the initial condition. This causes the PINN to get stuck in a local minimum and learn an erroneous solution.

Different strategies have been proposed to enforce causality[143,144]. In the notebook, we implement a simple curriculum learning scheme where the PINN is trained over successively increasing time segments. In other words, we control the collocation points in Eq. (19) during training so that we only feed t_{i+1} to the PINN after minimizing the loss at t.

Inverse estimation problem. The second notebook, Chapter 5.5, presents an example of PINNs solving an inverse estimation problem. In this case, the wave speed $c(\mathbf{r})$ and initial condition p_0 are unknown, and the information available is discrete observations of the wave field at several locations, $p_{\text{obs}} = p(\mathbf{r}_j, t_j)$, $j = 1, \ldots, n_{\text{obs}}$. The inverse problem aims to estimate the wave speed $c(\mathbf{r})$ from the observations.

For solving the inverse problem we train two neural networks: $\hat{p}(\mathbf{r}, t; \boldsymbol{\theta}_p)$, which approximates the wave field, and $\hat{c}(\mathbf{r}; \boldsymbol{\theta}_c)$, which approximates the wave speed. Both networks are trained simultaneously with a single loss function:

$$\mathcal{L} = \lambda_{\text{pde}}\mathcal{L}_{\text{pde}}(\boldsymbol{\theta}_p, \boldsymbol{\theta}_c) + \lambda_{\text{obs}}\mathcal{L}_{\text{obs}}(\boldsymbol{\theta}_p), \tag{24}$$

where

$$\mathcal{L}_{\text{pde}} = \frac{1}{n_{\text{pde}}} \sum_{i=1}^{n_{\text{pde}}} \left\| \nabla^2 \hat{p}(\mathbf{r}_i, t_i; \boldsymbol{\theta}_p) - \frac{1}{\hat{c}(\mathbf{r}_i; \boldsymbol{\theta}_c)^2} \frac{\partial^2 \hat{p}(\mathbf{r}_i, t_i; \boldsymbol{\theta}_p)}{\partial t^2} \right\|^2, \tag{25}$$

and

$$\mathcal{L}_{\text{obs}} = \frac{1}{n_{\text{obs}}} \sum_{j=1}^{n_{\text{obs}}} \left\| \hat{p}(\mathbf{r}_j, t_j; \boldsymbol{\theta}_p) - p_{\text{obs}}(\mathbf{r}_j, t_j) \right\|. \tag{26}$$

The PDE loss term links the two networks, as the wave equation contains both $p(\mathbf{r}, t)$ and $c(\mathbf{r})$. Once trained, $\hat{c}(\mathbf{r})$ can be queried at any point $\mathbf{r}$ to obtain an estimate of the wave speed.

Conclusion and future perspectives

We present an ML review that provides an in-depth discussion of applications on our open-source GitHub page. We demonstrate typical approaches for applying ML to acoustic research, focusing on applications such as sound classification, generative modeling for synthetic data, and physics-informed ML. The notebooks and the paper focus on a few relevant topics, emphasizing the benefits of applying ML and evaluating performance. Central to ML, trained models must generalize well on unobserved data—the test data.

ML methods can find structure in the data and learn low-dimensional representations of complex physical phenomena, like wave propagation. A current trend, likely to continue, is the learning of tractable surrogate models to accelerate simulations, processing, and estimation tasks.

The integration of physical constraints into ML models has improved their accuracy, efficiency, interpretability, and generalizability. We foresee a closer integration of physics and domain-specific knowledge into ML models, for example, by designing custom architectures that inherently comply with the physics of the problem[72], or the integration of PDE numerical solvers that allow for gradients to flow in the training process[145].

Explainability and interpretability has a central role in the current development of AI. ML methods for scientific discovery, where interpretable mathematical expressions are learned from data, could have an impact on fundamental acoustic research. These include symbolic regression methods[146], and the use of small but efficient neural networks like Kolmogorov-Arnold networks[147].

Data availability

Code for the manuscript can be found at the AcousticsML repository at https://github.com/RAMshades/AcousticsML.

Received: 24 April 2025; Accepted: 27 June 2025;
Published online: 09 September 2025

References

1. Bishop, C. *Pattern Recognition and Machine Learning*. Information Science and Statistics (Springer-Verlag New York, 2006), 1 edn.
2. Theodoridis, S. *Machine learning: a Bayesian and optimization perspective* (Academic press, 2015).
3. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, Cambridge, MA, 2016).

4. Murphy, K. P. *Probabilistic Machine Learning: An Introduction* (MIT Press, Cambridge, MA, 2022).
5. Bianco, M. J. et al. Machine learning in acoustics: Theory and applications. *J. Acoust. Soc. Am.* **146**, 3590–3628 (2019).
6. Cai, S., Mao, Z., Wang, Z., Yin, M. & Karniadakis, G. E. Physics-informed neural networks (PINNs) for fluid mechanics: a review. *Acta Mechanica Sin.* **37**, 1727–1738 (2021).
7. Karniadakis, G. E. et al. Physics-informed machine learning. *Nat. Rev. Phys.* **3**, 422–440 (2021).
8. Michalopoulou, Z.-H., Gerstoft, P., Kostek, B. & Roch, M. A. Introduction to the special issue on machine learning in acoustics. *J. Acoust. Soc. Am.* **150**, 3204–3210 (2021).
9. Grumiaux, P.-A., Kitić, S., Girin, L. & Guérin, A. A survey of sound source localization with deep learning methods. *J. Acoust. Soc. Am.* **152**, 107–151 (2022).
10. Cunha, B. Z., Droz, C., Zine, A.-M., Foulard, S. & Ichchou, M. A review of machine learning methods applied to structural dynamics and vibroacoustic. *Mech. Syst. Signal Process.* **200**, 110535 (2023).
11. Niu, H., Li, X., Zhang, Y. & Xu, J. Advances and applications of machine learning in underwater acoustics. *Intell. Mar. Technol. Syst.* **1**, 1–8 (2023).
12. Stańczyk, U. & Jain, L. C. *Feature selection for data and pattern recognition: An introduction* (Springer, 2014).
13. Wichern, G. et al. WHAM!: Extending speech separation to noisy environments. In *Proc. Interspeech* (2019).
14. Maciejewski, M., Wichern, G. & Le Roux, J. WHAMR!: Noisy and reverberant single-channel speech separation. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*. (2020).
15. Stowell, D. & Plumbley, M. D. An open dataset for research on audio field recording archives: freefield1010. *ArXiv*, abs/1309.5275, 1–10 https://api.semanticscholar.org/CorpusID:1396809 (2013).
16. Xiao, Y. & Das, R. K. WildDESED: An LLM-powered dataset for wild domestic environment sound event detection system. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events (DCASE2024)*, 196–200 (2024).
17. Sayigh, L. et al. The Watkins marine mammal sound database: An online, freely accessible resource. *Proc. Meet. Acoust.* **27**, 040013 (2017).
18. Gemmeke, J. F. et al. Audio set: An ontology and human-labeled dataset for audio events. In *IEEE Int. Conf. Acoust., Speech, and Signal Process*., 776–780 (2017).
19. Becker, S. et al. AudioMNIST: exploring explainable artificial intelligence for audio analysis on a simple benchmark. *J. Frankl. Inst.* **361**, 418–428 (2024).
20. Szöke, I., Skácel, M., Mošner, L., Paliesek, J. & Černocky`, J. Building and evaluation of a real room impulse response dataset. *IEEE J. Sel. Top. Signal Process.* **13**, 863–876 (2019).
21. Koyama, S. et al. MeshRIR: A dataset of room impulse responses on meshed grid points for evaluating sound field analysis and synthesis methods. In *Proc. IEEE Int. Workshop Appl. Signal Process. Audio Acoust. (WASPAA)* (2021).
22. Verburg, S. A., Karakonstantis, X. & Fernandez Grande, E. Room impulse response dataset - ACT, DTU Electro (019). https://doi.org/10.11583/DTU.25867705.v1 Technical University of Denmark. Dataset.
23. Brinkmann, F. et al. A cross-evaluated database of measured and simulated HRTFs including 3D head meshes, anthropometric features, and headphone impulse responses. *J. Audio Eng. Soc.* **67**, 705–718 (2019).
24. Ghorbal, S., Bonjour, X. & Séguier, R. Computed HRIRs and ears database for acoustic research. In *Proc. Aud. Eng. Soc. Conven. 148* (Audio Engineering Society, 2020).
25. Guezenoc, C. & Seguier, R. A wide dataset of ear shapes and pinna-related transfer functions generated by random ear drawings. *J. Acoust. Soc. Am.* **147**, 4087–4096 (2020).
26. Engel, I. et al. The SONICOM HRTF dataset. *J. Aud. Eng. Soc.* **71**, 241–253 (2023).
27. Fernandez-Grande, E., Karakonstantis, X., Caviedes-Nozal, D. & Gerstoft, P. Generative models for sound field reconstruction. *J. Acoust. Soc. Am.* **153**, 1179–1190 (2023).
28. Jenkins, W. F., Gerstoft, P., Bianco, M. J. & Bromirski, P. D. Unsupervised Deep Clustering of Seismic Data: Monitoring the Ross Ice Shelf, Antarctica. *J. Geophys. Res.: Solid Earth* **126** (2021).
29. Bianco, M. J., Gannot, S., Fernandez-Grande, E. & Gerstoft, P. Semi-supervised source localization in reverberant environments with deep generative modeling. *IEEE Access* **9**, 84956–84970 (2021).
30. Saha, P. et al. Leveraging sound speed dynamics and generative deep learning for ray-based ocean acoustic tomography. *JASA Express Letters* **5** (2025).
31. Yang, D. et al. Diffsound: Discrete diffusion model for text-to-sound generation. *IEEE/ACM Trans. Audio, Speech, Lang. Proc.* **31**, 1720–1733 (2023).
32. Li, S., Cheng, L., Li, J., Wang, Z. & Li, J. Learning data distribution of three-dimensional ocean sound speed fields via diffusion models. *J. Acoustical Soc. Am.* **155**, 3410–3425 (2024).
33. Kingma, D. P. & Welling, M. Auto-encoding variational bayes. In *Proc. Int. Conf. Learn. Represent*. (2014).
34. Akuzawa, K., Iwasawa, Y. & Matsuo, Y. Expressive speech synthesis via modeling expressions with variational autoencoder. In *Proc. Interspeech*, 3067–3071 (2018).
35. Roberts, A., Engel, J., Raffel, C., Hawthorne, C. & Eck, D. A hierarchical latent vector model for learning long-term structure in music. In *Proc. Int. Conf. Mach. Learn*., 4364–4373 (PMLR, 2018).
36. Goodfellow, I. et al. Generative adversarial nets. In *Adv. Neural Info. Process. Sys*. 2672–2680 (2014).
37. Goodfellow, I. et al. Generative adversarial networks. *Commun. Acm.* **63**, 139–144 (2020).
38. Donahue, C., McAuley, J. & Puckette, M. Adversarial audio synthesis. In *Proc. Int. Conf. Learn. Represent*. (2019).
39. Kong, J., Kim, J. & Bae, J. HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis. In *Proc. Adv. Neural Inf. Process. Syst*., vol. 33, 17022–17033 (2020).
40. Ratnarajah, A., Tang, Z. & Manocha, D. IR-GAN: Room impulse response generator for far-field speech recognition. In *Proc. Interspeech*, 286–290 (2021).
41. Défossez, A. et al. Moshi: a speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037* (2024).
42. Tagawa, Y., Maskeliūnas, R. & Damaševičius, R. Acoustic anomaly detection of mechanical failures in noisy real-life factory environments. *Electronics* **10**, 2329 (2021).
43. Jiang, A., Zhang, W.-Q., Deng, Y., Fan, P. & Liu, J. Unsupervised anomaly detection and localization of machine audio: A GAN-based approach. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*. (2023).
44. Song, D., Lee, N., Kim, J. & Choi, E. Anomaly detection of deepfake audio based on real audio using generative adversarial network model. *IEEE Access* (2024).
45. Chen, Z., Wang, S., Qian, Y. & Yu, K. Channel invariant speaker embedding learning with joint multi-task and adversarial training. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*., 6574–6578 (IEEE, 2020).
46. Li, H., Tu, M., Huang, J., Narayanan, S. & Georgiou, P. Speaker-invariant affective representation learning via adversarial training. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*., 7144–7148 (IEEE, 2020).
47. Zhang, Y., Zhu, G., Jiang, F. & Duan, Z. An empirical study on channel effects for synthetic voice spoofing countermeasure systems. In *Proc. Interspeech*, 4309–4313 (2021).
48. Gurbuz, C. et al. Generative adversarial networks for the design of acoustic metamaterials. *J. Acoust. Soc. Am.* **149**, 1162–1174 (2021).

49. Zhou, M. et al. On generative-adversarial-network-based underwater acoustic noise modeling. *IEEE Trans. Veh. Technol.* **70**, 9555–9559 (2021).
50. Fu, S.-W. et al. MetricGAN+: An improved version of metricgan for speech enhancement. In *Proc. Interspeech*, 201–205 (2021).
51. Eskimez, S. E., Koishida, K. & Duan, Z. Adversarial training for speech super-resolution. *IEEE J. Sel. Top. Signal Process.* **13**, 347–358 (2019).
52. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N. & Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proc. Int. Conf. Mach. Learn*., 2256–2265 (2015).
53. Ho, J., Jain, A. & Abbeel, P. Denoising diffusion probabilistic models. In *Proc. Adv. Neural Inf. Process. Syst*., vol. 33, 6840–6851 (2020).
54. Croitoru, F.-A., Hondru, V., Ionescu, R. T. & Shah, M. Diffusion models in vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **45**, 10850–10869 (2023).
55. Rombach, R., Blattmann, A., Lorenz, D., Esser, P. & Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit*., 10684–10695 (2022).
56. Song, Y., Dhariwal, P., Chen, M. & Sutskever, I. Consistency models. In *Proc. Int. Conf. Mach. Learn*., 32211–32252 (2023).
57. Miotello, F. et al. Reconstruction of sound field through diffusion models. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*., 1476–1480 (IEEE, 2024).
58. Liu, H. et al. AudioLDM: text-to-audio generation with latent diffusion models. In *Proc. Int. Conf. Mach. Learn*. (2023).
59. Bai, Y., Dang, T., Tran, D., Koishida, K. & Sojoudi, S. ConsistencyTTA: Accelerating diffusion-based text-to-audio generation with consistency distillation. In *Proc. Interspeech*, 3285–3289 (2024).
60. Heydari, M., Souden, M., Conejo, B. & Atkins, J. Immersediffusion: A generative spatial audio latent diffusion model. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*. (2025).
61. Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U. & Sutton, C. VEEGAN: Reducing mode collapse in GANs using implicit variational learning. In *Proc. Adv. Neural Inf. Process. Syst*., vol. 30 (2017).
62. Sitzmann, V., Martel, J., Bergman, A., Lindell, D. & Wetzstein, G. Implicit neural representations with periodic activation functions. In *Proc. Adv. Neural Inf. Process. Syst*., vol. 33, 7462–7473 (2020).
63. Luo, A. et al. Learning neural acoustic fields. In *Proc. Adv. Neural Inf. Process. Syst*., vol. 35, 3165–3177 (2022).
64. Su, K., Chen, M. & Shlizerman, E. INRAS: Implicit neural representation for audio scenes. In *Proc. Adv. Neural Inf. Process. Syst*., vol. 35, 8144–8158 (2022).
65. Liang, S., Huang, C., Tian, Y., Kumar, A. & Xu, C. AV-NeRF: Learning neural fields for real-world audio-visual scene synthesis. In *Proc. Adv. Neural Inf. Process. Syst*. (2023).
66. Gebru, I. D. et al. Implicit HRTF modeling using temporal convolutional networks. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*., 3385–3389 (2021).
67. Zhang, Y., Wang, Y. & Duan, Z. HRTF field: Unifying measured HRTF magnitude representation with neural fields. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*. (2023).
68. Masuyama, Y. et al. NIIRF: Neural IIR filter field for HRTF upsampling and personalization. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*. (2024).
69. Wen, Y., Zhang, Y. & Duan, Z. Mitigating cross-database differences for learning unified HRTF representation. In *Proc. IEEE Int. Workshop Appl. Signal Process. Audio Acoust. (WASPAA)*, 1–5 (2023).
70. Rudy, S. H., Brunton, S. L., Proctor, J. L. & Kutz, J. N. Data-driven discovery of partial differential equations. *Sci. Adv.* **3**, e1602614 (2017).
71. Li, K. & Chitre, M. Data-aided underwater acoustic ray propagation modeling. *IEEE J. Ocean. Eng.* **48**, 1127–1148 (2023).
72. Chen, R. T., Rubanova, Y., Bettencourt, J. & Duvenaud, D. K. Neural ordinary differential equations. *Adv. Neural Inf. Process. Syst.* **31**, 6572–6583 (2018).
73. Zhai, W., Tao, D. & Bao, Y. Parameter estimation and modeling of nonlinear dynamical systems based on runge–kutta physics-informed neural network. *Nonlinear Dyn.* **111**, 21117–21130 (2023).
74. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Computational Phys.* **378**, 686–707 (2019).
75. Cuomo, S. et al. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *J. Sci. Comput.* **92**, 88 (2022).
76. Kashinath, K. et al. Physics-informed machine learning: case studies for weather and climate modelling. *Philos. Trans. R. Soc. A* **379**, 20200093 (2021).
77. Kissas, G. et al. Machine learning in cardiovascular flows modeling: predicting arterial blood pressure from non-invasive 4d flow MRI data using physics-informed neural networks. *Computer Methods Appl. Mech. Eng.* **358**, 112623 (2020).
78. Yoon, S., Park, Y., Gerstoft, P. & Seong, W. Predicting ocean pressure field with a physics-informed neural network. *J. Acoust. Soc. Am.* **155**, 2037–2049 (2024).
79. Pettit, C. L. & Wilson, D. K. A physics-informed neural network for sound propagation in the atmospheric boundary layer. In *Proceedings of Meetings on Acoustics*, vol. 42 (AIP Publishing, 2020).
80. Borrel-Jensen, N., Engsig-Karup, A. P. & Jeong, C.-H. Physics-informed neural networks for one-dimensional sound field predictions with parameterized sources and impedance boundaries. *JASA Express Letters***1** (2021).
81. Karakonstantis, X., Caviedes-Nozal, D., Richard, A. & Fernandez-Grande, E. Room impulse response reconstruction with physics-informed deep learning. *J. Acoust. Soc. Am.* **155**, 1048–1059 (2024).
82. Koyama, S., Ribeiro, J. G., Nakamura, T., Ueno, N. & Pezzoli, M. Physics-informed machine learning for sound field estimation: Fundamentals, state of the art, and challenges. *IEEE Signal Process. Mag.* **41**, 60–71 (2025).
83. Olivieri, M., Pezzoli, M., Antonacci, F. & Sarti, A. A physics-informed neural network approach for nearfield acoustic holography. *Sensors* **21**, 7834 (2021).
84. Shukla, K., Di Leoni, P. C., Blackshire, J., Sparkman, D. & Karniadakis, G. E. Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks. *J. Nondestructive Evaluation* **39**, 1–20 (2020).
85. Wang, H. et al. On acoustic fields of complex scatters based on physics-informed neural networks. *Ultrasonics* **128**, 106872 (2023).
86. Savović, S., Ivanović, M. & Min, R. A comparative study of the explicit finite difference method and physics-informed neural networks for solving the Burgers' equation. *Axioms* **12**, 982 (2023).
87. Liu, R. & Gerstoft, P. Spatial acoustic properties recovery with deep learning. *J. Acoust. Soc. Am.* **155**, 3690–3701 (2024).
88. Wang, S., Teng, Y. & Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM J. Sci. Comput.* **43**, A3055–A3081 (2021).
89. Wang, S., Yu, X. & Perdikaris, P. When and why PINNs fail to train: A neural tangent kernel perspective. *J. Computational Phys.* **449**, 110768 (2022).
90. Rathore, P., Lei, W., Frangella, Z., Lu, L. & Udell, M. Challenges in training PINNs: A loss landscape perspective. In *Proc. 41st Int. Conf. Mach. Learn*., 42159–42191 (2024).
91. Wang, S., Wang, H. & Perdikaris, P. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. *Computer Methods Appl. Mech. Eng.* **384**, 113938 (2021).

92. Nabian, M. A., Gladstone, R. J. & Meidani, H. Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civ. Infrastruct. Eng.* **36**, 962–977 (2021).
93. Jagtap, A. D. & Karniadakis, G. E. Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics***28** (2020).
94. Moseley, B., Markham, A. & Nissen-Meyer, T. Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations. *Adv. Computational Math.* **49**, 62 (2023).
95. Bergstra, J. & Bengio, Y. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* **13**, 281–305 (2012).
96. Shahriari, B., Swersky, K., Wang, Z., Adams, R. P. & de Freitas, N. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* **104**, 148–175 (2016).
97. Jenkins, W. F., Gerstoft, P. & Park, Y. Geoacoustic inversion using Bayesian optimization with a Gaussian process surrogate model. *J. Acoust. Soc. Am.* **156**, 812–822 (2024).
98. Martinez-Cantin, R. BayesOpt: A Bayesian Optimization Library for Nonlinear Optimization, Experimental Design and Bandits. *J. Mach. Learn. Res.* **15**, 3915–3919 (2014).
99. Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min*., 2623–2631 (2019).
100. Balandat, M. et al. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Adv. Neural Inf. Process. Syst*., vol. 33 (2020).
101. Breiman, L. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Stat. Sci.* **16**, 199–231 (2001).
102. Gerstoft, P. & Mecklenbräuker, C. F. Ocean acoustic inversion with estimation of a posteriori probability distributions. *J. Acoust. Soc. Am.* **104**, 808–819 (1998).
103. Bonnel, J., Dosso, S. E. & Chapman, N. R. Bayesian geoacoustic inversion of single hydrophone light bulb data using warping dispersion analysis. *J. Acoust. Soc. Am.* **134**, 120–130 (2013).
104. Tibshirani, R. J. & Efron, B. An introduction to the bootstrap. *Monographs on Statistics and Applied Probability*.**57** (1993).
105. Dosso, S. E. Quantifying uncertainty in geoacoustic inversion. i. a fast gibbs sampler approach. *J. Acoustical Soc. Am.* **111**, 129–142 (2002).
106. Xiang, N. Model-based bayesian analysis in acoustics?a tutorial. *J. Acoust. Soc. Am.* **148**, 1101–1120 (2020).
107. Vardi, A. et al. Estimation of the spatial variability of the new england mud patch geoacoustic properties using a distributed array of hydrophones and deep learning. *J. Acoustical Soc. Am.* **156**, 4229–4241 (2024).
108. Khurjekar, I. D. & Gerstoft, P. Uncertainty quantification for direction-of-arrival estimation with conformal prediction. *J. Acoust. Soc. Am.* **154**, 979–990 (2023).
109. Khurjekar, I. D. & Gerstoft, P. Multi-source doa estimation with statistical coverage guarantees. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process*., 5310–5314 (IEEE, 2024).
110. Shafer, G. & Vovk, V. A tutorial on conformal prediction. *Journal of Machine Learning Research* **9** (2008).
111. Angelopoulos, A. N. & Bates, S. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511* (2021).
112. Dwivedi, R. et al. Explainable AI (XAI): Core ideas, techniques, and solutions. *ACM Comput. Surv.* **55**, 1–33 (2023).
113. Angelov, P. P., Soares, E. A., Jiang, R., Arnold, N. I. & Atkinson, P. M. Explainable artificial intelligence: an analytical review. *Wiley Interdiscip. Rev.: Data Min. Knowl. Discov.* **11**, e1424 (2021).
114. McInnes, L., Healy, J., Saul, N. & Großberger, L. Umap: Uniform manifold approximation and projection. *J. Open Source Softw.* **3**, 861 (2018).
115. Amid, E. & Warmuth, M. K. TriMap: Large-scale Dimensionality Reduction Using Triplets. *arXiv preprint arXiv:1910.00204* (2019).
116. Wang, Y., Huang, H., Rudin, C. & Shaposhnik, Y. Understanding how dimension reduction tools work: an empirical approach to deciphering t-sne, umap, trimap, and pacmap for data visualization. *J. Mach. Learn. Res.* **22**, 1–73 (2021).
117. Fisher, A., Rudin, C. & Dominici, F. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *J. Mach. Learn. Res.* **20**, 1–81 (2019).
118. Apley, D. W. & Zhu, J. Visualizing the effects of predictor variables in black box supervised learning models. *J. R. Stat. Soc. Ser. B: Stat. Methodol.* **82**, 1059–1086 (2020).
119. Ribeiro, M. T., Singh, S. & Guestrin, C. "Why Should I Trust You?": Explaining the predictions of any classifier. In *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min*., 1135–1144 (2016).
120. Lundberg, S. M. & Lee, S.-I. A unified approach to interpreting model predictions. In *Proc. 31st Int. Conf. Neural Inf. Process. Syst*., 4768–4777 (2017).
121. Ribeiro, M. T., Singh, S. & Guestrin, C. Anchors: High-precision model-agnostic explanations. In *Proc. AAAI Conf. Artif. Intell*., vol. 32 (2018).
122. Kahl, S., Wood, C. M., Eibl, M. & Klinck, H. BirdNET: A deep learning solution for avian diversity monitoring. *Ecol. Inform.* **61**, 101236 (2021).
123. Kahl, S. et al. Large-scale bird sound classification using convolutional neural networks. *CLEF (working notes)***1866** (2017).
124. Blaszke, M. & Kostek, B. Musical instrument identification using deep learning approach. *Sensors* **22**, 3033 (2022).
125. Bermant, P. C., Bronstein, M. M., Wood, R. J., Gero, S. & Gruber, D. F. Deep machine learning techniques for the detection and classification of sperm whale bioacoustics. *Sci. Rep.* **9**, 12588 (2019).
126. Tama, B. A., Vania, M., Lee, S. & Lim, S. Recent advances in the application of deep learning for fault diagnosis of rotating machinery using vibration signals. *Artif. Intell. Rev.* **56**, 4667–4709 (2023).
127. Li, X., Zhang, W., Ding, Q. & Sun, J.-Q. Intelligent rotating machinery fault diagnosis based on deep learning using data augmentation. *J. intell., manuf.* **31**, 433–452 (2020).
128. Saufi, S. R., Ahmad, Z. A. B., Leong, M. S. & Lim, M. H. Challenges and opportunities of deep learning models for machinery fault detection and diagnosis: A review. *IEEE Access* **7**, 122644–122662 (2019).
129. Aggarwal, C. C. & Reddy, C. K. (eds.) *Data Clustering: Algorithms and Applications* (CRC Press, Taylor & Francis Group, Boca Raton, FL, 2014).
130. Ozanich, E., Thode, A., Gerstoft, P., Freeman, L. A. & Freeman, S. Deep embedded clustering of coral reef bioacoustics. *J. Acoust. Soc. Am*. 2587–2601 (2021).
131. Linhardt, T. & Gupta, A. S. Cost–Benefit Analysis of Metavariable Variation in Convolutional Autoencoders Applied to Acoustic Backscattering Data from Small Underwater Targets. In *OCEANS 2022*, 1–4 (2022).
132. De Salvio, D., Bianco, M. J., Gerstoft, P., D'Orazio, D. & Garai, M. Blind source separation by long-term monitoring: A variational autoencoder to validate the clustering analysis. *J. Acoust. Soc. Am.* **153**, 738–750 (2023).
133. Guerrero, M. J., Bedoya, C. L., López, J. D., Daza, J. M. & Isaza, C. Acoustic animal identification using unsupervised learning. *Methods Ecol. Evol.* **14**, 1500–1514 (2023).
134. Jedrusiak, M. D. et al. Towards an interdisciplinary formalization of soundscapes. *J. Acoust. Soc. Am.* **155**, 2549–2560 (2024).

135. Gibb, K., Eldridge, A., Sandom, C. & Simpson, I. Towards interpretable learned representations for ecoacoustics using variational auto-encoding. *Ecol. Inform.* **80**, 102449 (2024).
136. Liu, Y., Gao, W., Chen, D. & Xu, L. Mode-informed complex-valued neural processes for matched field processing. *J. Acoust. Soc. Am.* **157**, 493–508 (2025).
137. Zhang, C. et al. Exploring the directivities of whistle in the Indo-Pacific humpback dolphin (*Sousa Chinensis*) and their dependency on the whistles' frequency contour. *J. Acoust. Soc. Am.* **157**, 669–680 (2025).
138. Mousavi, S. M., Zhu, W., Ellsworth, W. & Beroza, G. Unsupervised clustering of seismic signals using deep convolutional autoencoders. *IEEE Geosci. Remote Sens. Lett.* **16**, 1693–1697 (2019).
139. Snover, D., Johnson, C. W., Bianco, M. J. & Gerstoft, P. Deep clustering to identify sources of urban seismic noise in Long Beach, California. *Seismol. Res. Lett.* **92**, 1011–1022 (2021).
140. Chien, C.-C., Jenkins, W. F., Gerstoft, P., Zumberge, M. & Mellors, R. Automatic classification with an autoencoder of seismic signals on a distributed acoustic sensing cable. *Computers Geotech.* **155**, 105233 (2023).
141. Tibshirani, R., Walther, G. & Hastie, T. Estimating the number of clusters in a data set via the gap statistic. *J. R. Stat. Soc. B* **63**, 411–423 (2001).
142. Fantini, D., Geronazzo, M., Avanzini, F. & Ntalampiras, S. A survey on machine learning techniques for head-related transfer function individualization. *IEEE Open J. Signal Process.* **6**, 30–56 (2025).
143. Wang, S., Sankaran, S. & Perdikaris, P. Respecting causality for training physics-informed neural networks. *Computer Methods Appl. Mech. Eng.* **421**, 116813 (2024).
144. Mattey, R. & Ghosh, S. A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations. *Computer Methods Appl. Mech. Eng.* **390**, 114474 (2022).
145. Zhu, W., Xu, K., Darve, E. & Beroza, G. C. A general approach to seismic inversion with automatic differentiation. *Computers Geosci.* **151**, 104751 (2021).
146. Wang, H. et al. Scientific discovery in the age of artificial intelligence. *Nature* **620**, 47–60 (2023).
147. Liu, Z. et al. KAN: Kolmogorov-Arnold networks. *Proc. Int. Conf. Learn. Represent.* (2025).
148. Gerstoft, P., Bianco, M. J., McCarthy, R. A. & Zhang, N. Tutorial on machine learning for acoustics. *J. Acoust. Soc. Am.* **156**, A78–A78 (2024).
149. Strobel, V. Pold87/academic-keyword-occurrence: First release 2018. Available online: https://zenodo.org/records/1218409#.ZDVFeOxBy3I. Accessed: 23-04-2025.

Acknowledgements

Part of the content of this manuscript was presented at the tutorial session of the ASA 187th Meeting, held Online, Nov 20, 2024[148]. P.G. thanks the support from the Office of Naval Research, N000142412016. W.F.J. thanks the support from the Office of Naval Research, N00014-24-1-2401.

Author contributions

R.A.M. prepared Figures 5-8. Y.Z. prepared Figures 2 and 11. S.A.V. prepared Figures 1, 3, and 12. W.F.J. prepared Figures 9 and 10. P.G. prepared Figure 4. All authors contributed equally to the conception, code development, original draft writing, review, and approval of the submitted manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Ryan A. McCarthy.

Reprints and permissions information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

© The Author(s) 2025