Machine learning in acoustics: Theory and applications

Michael J. Bianco,1,a) Peter Gerstoft,1 James Traer,2 Emma Ozanich,1 Marie A. Roch,3 Sharon Gannot,4 and Charles-Alban Deledalle5
1Scripps Institution of Oceanography, University of California San Diego, La Jolla, California 92093, USA
2Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA
3Department of Computer Science, San Diego State University, San Diego, California 92182, USA
4Faculty of Engineering, Bar-Ilan University, Ramat-Gan 5290002, Israel
5Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, California 92093, USA

(Received 9 May 2019; revised 23 September 2019; accepted 14 October 2019; published online 27 November 2019)

Acoustic data provide scientific and engineering insights in fields ranging from biology and communications to ocean and Earth science. We survey the recent advances and transformative potential of machine learning (ML), including deep learning, in the field of acoustics. ML is a broad family of techniques, which are often based in statistics, for automatically detecting and utilizing patterns in data. Relative to conventional acoustics and signal processing, ML is data-driven. Given sufficient training data, ML can discover complex relationships between features and desired labels or actions, or between features themselves. With large volumes of training data, ML can discover models describing complex acoustic phenomena such as human speech and reverberation. ML in acoustics is rapidly developing with compelling results and significant future promise. We first introduce ML, then highlight ML developments in four acoustics research areas: source localization in speech processing, source localization in ocean acoustics, bioacoustics, and environmental sounds in everyday scenes. © 2019 Acoustical Society of America.

https://doi.org/10.1121/1.5133944

I. INTRODUCTION

Acoustic data provide scientific and engineering insights in a very broad range of fields including machine interpretation of human speech1,2 and animal vocalizations,3 ocean source localization,4,5 and imaging geophysical structures in the ocean.6,7 In all these fields, data analysis is complicated by a number of challenges, including data corruption, missing or sparse measurements, reverberation, and large data volumes. For example, multiple acoustic arrivals of a single event or utterance make source localization and speech interpretation a difficult task for machines.2,8 In many cases, such as acoustic tomography and bioacoustics, large volumes of data can be collected. The amount of human effort required to manually identify acoustic features and events rapidly becomes limiting as the size of the datasets increase. Further, patterns may exist in the data that are not easily recognized by human cognition.

Machine learning (ML) techniques9,10 have enabled broad advances in automated data processing and pattern recognition capabilities across many fields, including computer vision, image processing, speech processing, and (geo)physical science.11,12 ML in acoustics is a rapidly developing field, with many compelling solutions to the aforementioned acoustics challenges. The potential impact of ML-based techniques in the field of acoustics, and the recent attention they have received, motivates this review.

Broadly defined, ML is a family of techniques for automatically detecting and utilizing patterns in data. In ML, the patterns are used, for example, to estimate data labels based on measured attributes, such as the species of an animal or their location based on recordings from acoustic arrays. These measurements and their labels are often uncertain; thus, statistical methods are often involved. In this way, ML provides a means for machines to gain knowledge, or to “learn.”13,14 ML methods are often divided into two major categories: supervised and unsupervised learning. There is also a third category called reinforcement learning, though it is not discussed in this review. In supervised learning, the goal is to learn a predictive mapping from inputs to outputs given labeled input and output pairs. The labels can be categorical or real-valued scalars for classification and regression, respectively. In unsupervised learning, no labels are given, and the task is to discover interesting or useful structure within the data. An example of unsupervised learning is clustering analysis (e.g., K-means). Supervised and unsupervised modes can also be combined. Namely, semi and weakly supervised learning methods can be used when the labels only give partial or contextual information.

Research in acoustics has traditionally focused on developing high-level physical models and using these models for inferring properties of the environment and objects in the environment. The complexity of physical principal-based...
models is indicated by the x axis in Fig. 1. With increasing amounts of data, data-driven approaches have made enormous success. The volume of available data is indicated by the y axis in Fig. 1. It is expected that as more data become available in physical sciences that we will be able to better combine advanced acoustic models with ML.

In ML, it is preferred to learn representation models of the data, which provide useful patterns in the data for the ML task at hand, directly from the data rather than by using specific domain knowledge to engineer representations. ML can build upon physical models and domain knowledge, improving interpretation by finding representations (e.g., transformations of the features) that are “optimal” for a given task. Representations in ML are patterns the input features, which are particular attributes of the data. Features include spectral characteristics of human speech, or morphological features of a physical environment. Feature inputs to an ML pipeline can be raw measurements of a signal (data) or transformations of the data, e.g., obtained by the classic principal components analysis (PCA) approach. More flexible representations, including Gaussian mixture models (GMMs) are obtained using the expectation-maximization (EM). The fundamental concepts of ML are by no means new. For example, linear discriminant analysis (LDA), a fundamental classification model, was developed as early as the 1930s. The K-means clustering algorithm and the perceptron algorithm, which was a precursor to modern neural networks (NNs), were developed in the 1960s. Shortly after the perceptron algorithm was published, interest in NNs waned until the 1980s when the backpropagation algorithm was developed. Currently we are in the midst of a “third-wave” of interest in ML and AI principles.

ML in acoustics has made significant progress in recent years. ML-based methods can provide superior performance relative to conventional signal processing methods. However, a clear limitation of ML-based methods is that they are data-driven and thus require large amounts of data for testing and training. Conventional methods also have the benefit of being more interpretable than many ML models. Particularly in deep learning, ML models can be considered “black-boxes”—meaning that the intervening operations, between the inputs and outputs of the ML system, are not necessarily physically intuitive. Further, due to the no free-lunch theorem, models optimized for one task will likely perform worse at others. The intention of this review is to indicate that, despite these challenges, ML has considerable potential in acoustics.

**FIG. 1.** (Color online) Acoustic insight can be improved by leveraging the strengths of both physical and ML-based, data-driven models. Analytic physical models (lower left) give basic insights about physical systems. More sophisticated models, reliant on computational methods (lower right), can model more complex phenomena. Whereas physical models are reliant on rules, which are updated by physical evidence (data), ML is purely data-driven (upper left). By augmenting ML methods with physical models to obtain hybrid models (upper right), a synergy of the strengths of physical intuition and data-driven insights can be obtained.
This review focuses on the significant advances ML has already provided in the field of acoustics. We first introduce ML theory, including deep learning (DL). Then we discuss applications and advances in the theory in five acoustics research areas. In Secs. II–IV, basic ML concepts are introduced, and some fundamental algorithms are developed. In Sec. V, the field of DL is introduced, and applications to acoustics are discussed. Next, we discuss applications of ML theory to the following fields: speaker localization in reverberant environments (Sec. VI), source localization in ocean acoustics (Sec. VII), bioacoustics (Sec. VIII), and reverberation and environmental sounds in everyday scenes (Sec. IX). While the list of fields we cover and the treatment of ML theory is not exhaustive, we hope this article can serve as inspiration for future ML research in acoustics. For further reference, we refer readers to several excellent ML and signal processing textbooks, which are useful supplements to the material presented here: Refs. 2, 13, 14, 16, and 21–25.

II. MACHINE LEARNING PRINCIPLES

ML is data-driven and can model potentially more complex patterns in the data than conventional methods. Classic signal processing techniques for modeling and predicting data are based on provable performance guarantees. These methods use simplifying assumptions, such as Gaussian independent and identically distributed (iid) variables, and second order statistics (covariance). However, ML methods, and recently DL methods in particular, have shown improved performance in a number of tasks compared to conventional methods. But, the increased flexibility of the ML models comes with certain difficulties.

Often the complexity of ML models and their training algorithms make guaranteeing their performance difficult and can hinder model interpretability. Further, ML models can require significant amounts of training data, though we note that “vast” quantities of training data are not required to take advantage of ML techniques. Due to the no free lunch theorem, models whose performance is maximized for one task will likely perform worse at others. Provided high-performance is desired only for a specific task, and there is enough training data, the benefits of ML may outweigh these issues.

A. Inputs and outputs

In acoustics and signal processing, measurement models explain sets of observations using a set of models. The model explaining the observations is typically called the “forward” model. To find the best model parameters, the forward model is “inverted.” However, ML measurement models are articulated in terms of models relating inputs and outputs, both of which are observed,

\[ y = f(x) + \epsilon. \tag{1} \]

Here, \( x \in \mathbb{R}^N \) are \( N \) inputs and \( y \in \mathbb{R}^P \) are \( P \) outputs to the model \( f(x) \). \( f(x) \) can be a linear or non-linear mapping from input to output. \( \epsilon \) is the uncertainty in the estimate \( f(x) \) which is due to model limitations and uncertainty in the measurements. Thus, the ML measurement model (1) has similarities with the “inverse” of the typical “forward” model.

Per Eq. (1), \( x \) is a single observation of \( N \) inputs, called features, from which we would like to estimate a single set of outputs \( y \). For example, in a simple feed-forward NN (Sec. III C and Sec. V), the input layer \( x \) has dimension \( N \) and the output layer \( y \) has dimension \( P \). The NN then constitutes a non-linear function \( f(x) \) relating the inputs to the outputs. To train the NN [learn \( f(x) \)] requires many samples of input/output pairs. We define \( X = [x_1, \ldots, x_M]^T \in \mathbb{R}^{M \times N} \) and \( Y = [y_1, \ldots, y_M] \in \mathbb{R}^{P \times M} \) the corresponding \( P \) outputs for \( M \) samples of the input/output pairs. We here note that there are many ML scenarios where the number of input samples and output samples are different (e.g., recurrent NNs have more input samples than output samples).

The use of ML to obtain output \( y \) from features \( x \), as described above, is called supervised learning (Sec. III). Often, we wish to discover interesting or useful patterns in the data without explicitly specifying output. This is called unsupervised learning (Sec. IV). In unsupervised learning, the goal is to learn interesting or useful patterns in the data. In many cases in unsupervised learning, the input and desired output is the features themselves.

B. Supervised and unsupervised learning

ML methods generally can be categorized as either supervised or unsupervised learning tasks. In supervised learning, the task is to learn a predictive mapping from inputs to outputs given labeled input and output pairs. Supervised learning is the most widely used ML category and includes familiar methods such as linear regression (also called ridge regression) and nearest-neighbor classifiers, as well as more sophisticated support vector machine (SVM) and neural network (NN) models—sometimes referred to as artificial NNs, due to their weak relationship to neural structure in the biological brain. In unsupervised learning, no labels are given, and the task is to discover interesting or useful structure within the data. This has many useful applications, which include data visualization, exploratory data analysis, anomaly detection, and feature learning. Unsupervised methods such as PCA, K-means, and Gaussian mixture models (GMMs) have been used for decades. Newer methods include t-SNE, dictionary learning, and deep representations (e.g., autoencoders). An important point is that the results of unsupervised methods can be used either directly, such as for discovery of latent factors or data visualization, or as part of a supervised learning framework, where they supply transformed versions of the features to improve supervised learning performance.

C. Generalization: Train and test data

Central to ML is the requirement that learned models must perform well on unobserved data as well as observed data. The ability of the model to predict unseen data well is called generalization. We first discuss relevant terminology,
then discuss how generalization of an ML model can be assessed.

Often, the term *complexity* is used to denote the level of sophistication of the data relationships or ML task. The ability of a particular ML model to well approximate data relationships (e.g., between features and labels) of a particular complexity is the *capacity*. These terms are not strictly defined, but efforts have been made to mathematically formalize these concepts. For example, the Vapnik-Chervonenkis (VC) dimension provides a means of quantifying model capacity in the case of binary classifiers. Data complexity can be interpreted as the number of dimensions in which useful relationships exist between features. Higher complexity implies higher-dimensional relationships. We note that the capacity of the ML model can be limited by the quantity of training data.

In general, ML models perform best when their capacity is suited to the complexity of the data provided and the task. For mismatched model-data/task complexities, two situations can arise. If a high-capacity model is used for a low-complexity task, the model will overfit, or learn the noise or idiosyncrasies of the training set. In the opposite scenario, a low-capacity model trained on a high-complexity task will tend to underfit the data, or not learn enough details of the underlying physics, for example. Both overfitting and underfitting degrade ML model generalization. The behavior of the ML model on training and test observations relative to the model parameters can be used to determine the appropriate model complexity. We next discuss how this can be done. We note that underfitting and overfitting can be quantified using the bias and variance of the ML model. The bias is the difference between the mean of our estimated targets \( \hat{y} \) and the true mean, and the variance is the expected squared deviation of the estimated targets around the estimated mean value.

To estimate the performance of ML models on unseen observations, and thereby assess their generalization, a set of test data drawn from the full training set can be excluded from the model training and used to estimate generalization given the current parameters. In many cases, the data used in developing the ML model are split repeatedly into different sets of training and test data using cross validation techniques (Sec. II D)\(^2^9\). The test data is used to adjust the model hyperparameters (e.g., regularization, priors, number of NN units/layers) to optimize generalization. The hyperparameters are model dependent, but generally govern the model’s capacity.

In Fig. 2, we illustrate the effect of model capacity on train and test error using polynomial regression. Train and test data (10 and 100 points) were generated from a sinusoid \( y = \sin 2\pi x \) (left) with additive Gaussian noise. Polynomial models of orders 0 to 9 were fit to the training data, and the RMSE of the test and train data predictions are compared.

\[
RMSE = \sqrt{\frac{1}{M} \sum_m (y_m - \hat{y}_m)^2}, \quad \text{with} \quad M \text{ the number of samples (test or train) and } \hat{y}_m \text{ the estimate of } y_m.
\]

Increasing model capacity (complexity) decreases the training error, up to degree 9 where the degree plus intercept matches the number of training points (degrees of freedom). While increasing the complexity initially decreases the RMSE of the test data prediction, errors do not significantly decrease for polynomial degrees greater than 3, and increase for degrees greater than 5. Thus, we would prefer to use a model of degree 3, though the smallest test error was obtained for degree 5. In ML applications on real data, the test/train error curves are generated using cross-validation to improve the robustness of the model selection.

Alternatively, the model can be trained, tuned, and evaluated by dividing the data into three distinct sets: training, validation, and test. In this case the model is fit on the training data, and its performance on the validation data is used to tune the hyperparameters. Only after the hyperparameters are fully tuned on the training and validation data is the model performance evaluated on the test data. Here the test data is kept in a “vault,” i.e., it should never influence the model parameters.

**D. Cross-validation**

In many cases, we do not have enough samples to divide the data into three fully representative subsets (train, validation, and test). Thus, we prefer to use the tools of cross-validation with only two subsets of data: training and test. Cross-validation evaluates the model generalization by creating multiple training and test sets from the data (without replacement). The model parameters in this case are tuned using the “test” data.

One popular cross-validation technique, called K-fold cross validation,\(^2^1\) assesses model generalization by dividing training data into \( K \) roughly equal-sized subgroups of the data, called *folds*. One fold is excluded from the model...
training and the error is calculated on the excluded fold. This procedure is executed $K$ times, with the $i$th fold used as the test data and the remaining $K-1$ folds used for model training. With target values divided into folds by $Y = [Y_1, \ldots, Y_K]$ and inputs $X = [X_1^T, \ldots, X_K^T]^T$, the cross validation error $CV_{err}$ is

$$CV_{err}(f, \theta) = \frac{1}{K} \sum_{i=1}^{K} L(Y_i, f^{-i}(X_i^T, \theta)),$$  \hspace{1cm} (2)$$

with $f^{-i}$ the model learned using all folds except $i$, $\theta$ the hyperparameters, and $L$ a loss function. $CV_{err}(f, \theta)$ gives a curve describing the cross-validation (test) error as a function of the hyperparameters.

Some issues arise when using cross-validation. First, it requires as many training runs as subdivisions of the data. Further, tuning multiple hyperparameters with cross-validation can require a number of training runs that is exponential in the number of parameters. Some alternatives to the aforementioned test/train paradigms penalize the model complexity directly in the optimization. Such constraints include the well known Akaike information criterion (AIC) and Bayesian information criterion (BIC). However, AIC and BIC do not account for parameter uncertainty and often favor overly simple models. In fully Bayesian approaches (as described in Sec. II F), parameter uncertainty and model complexity are both well modeled.

E. Curse of dimensionality

The often high-dimensionality of data also presents a challenge in ML, referred to as the “curse of dimensionality.” Considering features $x$ are uniformly distributed in $N$ dimensions (see Fig. 3) with $x_n = l$ the normalized feature value, then $l$ (for example, describing a neighborhood as a hypercube) constitutes a decreasing fraction of the features space volume. The fraction of the volume, $l^N$, is given by $f_v = l^N$, with $f_v$ and $l$ the volume and length fractions, respectively. Similarly, data tend to become more sparsely distributed in high-dimensional space. The curse of dimensionality most strongly affects methods that depend on distance measures in feature space, such as K-means, since neighborhoods are no longer “local.” Another result of the curse of dimensionality is the increased number of possible configurations, which may lead to ML models requiring increased training data to learn representations.

With prior assumptions on the data, enforced as model constraints (e.g., total variation\textsuperscript{10} or $\ell_2$ regularization), training with smaller datasets is possible.\textsuperscript{16} This is related to the concept of learning a manifold, or a lower-dimensional embedding of the salient features. While the manifold assumption is not always correct, it is at least approximately correct for processes involving images and sound [for more discussion, see Ref. 16 (pp. 156–159)].

F. Bayesian machine learning

A theoretically principled way to implement ML methods is to use the tools of probability, which have been a critical force in the development of modern science and engineering. Bayesian statistics provide a framework for integrating prior knowledge and uncertainty about physical systems into ML models. It also provides convenient analysis of estimated parameter uncertainty. Naturally, Bayes’ rule plays a fundamental rule in many acoustic applications, especially in methods for estimating the parameters of model-based inverse methods. In the wider ML community, there are also attempts to expand ML to be Bayesian model-based, for a review see Ref. 31. We here discuss the basic rules of probability, as they relate to Bayesian analysis, and show how Bayes’ rule can be used to estimate ML model parameters.

Two simple rules for probability are of fundamental importance for Bayesian ML.\textsuperscript{13} They are the sum rule

$$p(x) = \sum_{y \in \Omega} p(x, y),$$  \hspace{1cm} (3)$$

FIG. 3. (Color online) Illustration of curse of dimensionality. 10 uniformly distributed data points on the interval (0 1) can be quite close in 1D (top, squares), but as the number of dimensions, $N$, increases, the distance between the points increases rapidly. This is shown for points in 2D (top, circles), and 3D (bottom). The increasing volume $l^N$, with $l$ the normalized feature value scale, presents two issues. (1) local methods (like K-means) break-down with increasing dimension, since small neighborhoods in lower-dimensional space cover an increasingly small volume as the dimension increases. (2) Assuming discrete values, the number of possible data configurations, and thereby the minimum number of training examples, increase with dimension $O(l^N)$ (Refs. 16, 21).
and the product rule
\[
p(x, y) = p(x|y)p(y).
\] (4)

Here, the ML model inputs \( x \) and outputs \( y \) are uncertain quantities. The sum rule (3) states that the marginal distribution \( p(x) \) is obtained by summing the joint distribution \( p(x, y) \) over all values of \( y \). The product rule (4) states that \( p(x, y) \) is obtained as a product of the conditional distribution, \( p(y|x) \), and \( p(x) \).

Bayes’ rule is obtained from the sum and product rules by
\[
p(y|x) = \frac{p(x,y)}{\sum_{y\in\mathcal{Y}} p(x,y)} = \frac{p(x|y)p(y)}{p(x)},
\] (5)

which gives the model output \( y \) conditioned on the input \( x \) as the joint distribution \( p(x,y) \) divided by the marginal \( p(x) \).

In ML, we need to choose an appropriate model \( f(x) \) (1) and estimate the model parameters \( \theta \) to best give the desired output \( y \) from inputs \( x \). This is the inverse problem. The model parameters conditioned on the data is expressed as \( p(\theta|x,y) \). From Bayes’ rule (5) we have
\[
p(\theta|x,y) = \frac{p(y|x,\theta)p(\theta|x)}{p(y|x)} \propto p(y|x,\theta)p(\theta).
\] (6)

\( p(\theta) \) is the prior distribution on the parameters, \( p(y|x,\theta) \) called the likelihood, and \( p(\theta|x,y) \) the posterior. The quantity \( p(y|x) \) is the distribution of the data, also called the evidence or type II likelihood. Often it can be neglected [e.g., Eq. (7)] as for given data \( p(y|x) \) is constant and does not affect the target, \( \theta \).

A Bayesian estimate of the parameters \( \theta \) is obtained using Eq. (6). Assuming a scalar linear model \( y = f(x) + \epsilon \), with \( f(x) = x^T w \), where the parameters \( \theta = w \in \mathbb{R}^N \) are the weights (see Sec. III A for more details). A simple solution to the parameter estimate is obtained if we assume the prior \( p(w) \) is Gaussian, \( N(\mu, \Gamma) \), with \( \mu \) mean and covariance \( \Gamma \). Often, we also assume a Gaussian likelihood \( p(x,y|\theta) \), \( N(x^T w, \sigma_r) \) with mean \( x^T w \) and covariance \( \Sigma_r \). We get, see Ref. 13 (p. 93),
\[
p(w|x,y) = N(w_p, \Sigma_p),
\] (8)
\[
w_p = \Sigma_p \left( \frac{1}{\sigma_r} \mathbf{x}_y + \Gamma^{-1} \mu \right),
\] (9)
\[
\Sigma_p = \left( \frac{1}{\sigma_r} \mathbf{x}_x^T + \Gamma^{-1} \right)^{-1}.
\] (10)

The formulas are very efficient for sequential estimation as the prior is conjugated, i.e., it is of the same form as the posterior. In acoustics, this framework has been used for range estimation32 and for sparse estimation via the sparse Bayesian learning approach.33,34 In the latter, the sparsity is controlled by diagonal prior covariance matrix, where entries with zero prior variance will force the posterior variance and mean to be zero.

With prior knowledge and assumptions about the data, Bayesian approaches to parameter estimation can prevent overfitting. Further, Bayesian approaches provide the probability distribution of target estimates \( \hat{y} \). Figure 4 shows a Bayesian estimate of polynomial curve-fit developed in Fig. 2. The mean and standard deviation of the predictions from the model are given. The Bayesian curve fitting is here performed assuming prior knowledge of the noise standard deviation (\( \sigma_r = 0.2 \)) and with a Gaussian prior on the weights (\( \sigma_w = 10 \)). The hyperparameters can be estimated from the data using empirical Bayes.35 This is counterpoint to the test-train error analysis (Fig. 2), where fewer assumptions are made about the data, and the noise is unknown. We note that it is not always practical to formally implement Bayesian parameter estimation due to the increased computational cost of estimating the posterior distribution versus optimization. Where practical, Bayesian models well characterize ML results because they explicitly provide uncertainty in the model parameter estimates with the posterior distribution, and also permit explicit specification of prior knowledge of the parameter distributions (the prior) and data uncertainty.

### III. SUPERVISED LEARNING

The goal of supervised learning is to learn a mapping from a set of inputs to desired outputs given labeled input and output pairs (1). For discussion, we here focus on real-valued features and labels. The \( N \) features in \( x \) can be real, complex, or categorical (binary or integer). Based on the type of desired output \( y \), supervised learning can be divided into two subcategories: regression and classification. When \( y \) is real or complex valued, the task is regression. When \( y \) is categorical, the task is called classification.

The methods of finding the function \( f \) are the core of ML methods and the subject of this section. Generally, we

![FIG. 4. (Color online) Bayesian estimate of polynomial regression model parameters for sinusoidal data from Fig. 2. Given prior knowledge and assumptions about the data, Bayesian parameter estimation can help prevent overfitting. It also provides statistics about the predictions. The mean of the prediction (blue line) is compared with the true signal (red) and the training data (blue dots, same as Fig. 2). The standard deviation of the prediction (STD, light blue) is also given by the Bayesian estimate. The estimate uses prior knowledge about the noise level \( \sigma_r = 0.2 \) and a Gaussian prior on the model weights \( \sigma_w = 10 \).](image-url)
prefer to use the tools of probability to find \( f \), if practical. We can state the supervised ML task as the task of maximizing the conditional distribution \( p(y|x) \). One example is the maximum a posteriori (MAP) estimator
\[
\hat{y} = f(x) = \arg\max_y p(y|x),
\]
which gives the most probable value of \( y \), corresponding to the mode of the distribution conditioned on the observed evidence \( p(y|x) \). While the MAP can be considered Bayesian, it is really only a step toward Bayesian treatment (see Sec. II F) since MAP returns a point estimate rather than the posterior distribution.

In the following, we further describe regression and classification methods and give some illustrative applications.

### A. Linear regression, classification

We illustrate supervised ML with a simple method: linear regression. We develop a MAP formulation of linear regression in the context of direction-of-arrival (DOA) estimation in beamforming. In seismic and acoustic beamforming, waveforms are recorded on an array of receivers with the goal of finding their DOA. The features are the Fourier-transformed measurements from \( M \) receivers, \( x \in \mathbb{C}^M \), and the output \( y \) is the DOA azimuth angle [see Eq. (1)]. The relationship between DOA and array power is non-linear, thus, the MAP estimate (a.k.a. damped least squares, or ridge regression). \( ^{13,36} \) Equation (17) has the analytic solution
\[
\hat{w} = (A^T A + \lambda_2 I)^{-1} A^T x.
\]

Although the \( \ell_2 \) regularization in Eq. (17) is often convenient, it is sensitive to outliers in the data \( x \). In the presence of outliers, or if the true weights \( w \) are sparse (e.g., few non-zero weights), a better prior is the Laplacian, which gives
\[
\hat{w} = \arg\min_w \| x - Aw \|^2_2 + \lambda_1 \| w \|^1_1,
\]
where \( \lambda_2 = \sigma_e / h_w \) a regularization parameter, and \( h_w \) a scaling parameter for the Laplacian distribution. \( ^{14} \) Equation (19) is called the \( \ell_1 \) regularized least-squares estimator of \( w \).

While the problem is convex, it is not analytic, though there are many practical algorithms for its solution. \( ^{24,25,37} \) In sparse modeling, the \( \ell_1 \)-regularization is considered a convex relaxation of \( \ell_0 \) pseudo-norm, and under certain conditions, provides a good approximation to the \( \ell_0 \)-norm. For a more detailed discussion, please see Refs. 24 and 25. The solution to Eq. (19) is also known as the LASSO, \( ^{38} \) and forms the cornerstone of the field of compressive sensing (CS). \( ^{39,40} \)

Whereas in the estimate \( \hat{w} \) obtained from Eq. (17) many of the coefficients are small, the estimate from Eq. (19) has only few non-zero coefficients. Sparsity is a desirable property in many applications, including array processing\( ^{41,42} \) and image processing.\( ^{25} \) We give an example of \( \ell_1 \) (in CS) and \( \ell_2 \) regularization in the estimation of DOAs on a line array, Fig. 5.

Linear regression can be extended to the binary classification problem. Here for binary classification, we have a single desired output \( (N = 1) y_m \) for each input \( x_m \), and the labels are either 0 or 1. The desired labels for \( M \) observations are \( y \in \{0, 1\}^{1 \times M} \) (row vector),
\[
y = Xw.
\]

Here \( w \in \mathbb{R}^N \) is the weights vector. Following the derivation of Eq. (17), the MAP estimate of the weights is given by
\[
\hat{w} = (X^T X + \lambda_1 I)^{-1} X^T y,
\]
with \( w \) the ridge regression estimate of the weights.

This ridge regression classifier is demonstrated for binary classification (\( C = 2 \) in Fig. 7 (top). The cyan class is 0 and red is 1, thus, the decision boundary (black line) is
SVMs are based on maximum margin classifiers,\textsuperscript{14} and use a concept called the kernel trick to use potentially infinite-dimensional feature mappings with reasonable computational cost.\textsuperscript{13} This uses kernel functions, relating the transforms of two features as $\kappa(x_i,x_j) = \phi(x_i)\phi(x_j)$. They can be interpreted as similarity measures of linear or non-linear transformations of the feature vectors $x_i, x_j$. Kernel functions can take many forms [see Ref. 13 (pp. 291–232)], but for this review we illustrate SVMs with the Gaussian radial basis function (RBF) kernel

$$\kappa(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2).$$ \hfill (22)

$\gamma$ controls the length scale of the kernel. RBF can also be used for regression. The RBF is one example of kernelization of an infinite dimensional feature transform.

SVMs can be easily formulated to take advantage of such kernel transformations. Below, we derive the maximum margin classifier of SVM, following the arguments of Ref. 13, and show how kernels can be used to enhance classification.

Initially, we assume linearly separable features $X$ (see Fig. 6) with classes $s_m \in \{1, -1\}$. The class of the objects corresponding to the features is determined by

$$y = Xw + w_0,$$ \hfill (23)

with $w$ and $w_0$ the weights and biases. A decision hyperplane satisfying $Xw + w_0 = 0$ is used to separate the classes. If $y_m$ is above the hyperplane ($y_m > 0$), the estimated class label is $\hat{s}_m = 1$, whereas if $y_m$ is below ($y_m < 0$), $\hat{s}_m = -1$. This gives the condition $s_my_m > 0 \forall m$. The margin $d_M$ is defined as the distance between the nearest features (Fig. 6) with different labels, $x_-, s = -1$ and $x_+, s = +1$. These points correspond to the equations $w^T x_- + w_0 = -1$ and $w^T x_+ + w_0 = 1$.

**FIG. 6.** (Color online) Support vector machine (SVM) binary classification with separable classes (2D, $N = 2$). The hyperplane is estimated by a SVM which maximizes the margin $d_M$ subject to the constraint that none of the data are misclassified [see Eq. (25)]. When there are only two support vectors (as shown here), the hyperplane is orthogonal to the difference of the support vectors.
The difference between these equations, normalized by the weights $\|w\|_2$, yields an expression for the margin

$$\frac{w^T}{\|w\|_2} (x_+ - x_-) = \frac{2}{\|w\|_2}, \tag{24}$$

The expression says the projection of the difference of $x_+$ and $x_-$ on $w^T/\|w\|_2$ (unit vector perpendicular to the hyperplane) is $2/\|w\|_2$. Hence, $d_M = 2/\|w\|_2$.

The weights $w$ and $w_0$ are estimated by maximizing the margin $2/\|w\|_2$, subject to the constraint that the points $x_m$ are correctly classified. Observing that $2/\|w\|_2$ is equivalent to $\min \frac{1}{2} \|w\|_2^2$, the optimization is a quadratic program

$$\min_{w,w_0} \frac{1}{2} \|w\|_2^2,$$

subject to $s_m (w^T x_m + w_0) \geq 1 \forall m. \tag{25}$$

If the data are linearly non-separable (class overlapping), slack variables $\xi_m \geq 0$ allows some of the training points to be misclassified. This gives

$$\min_{w,w_0} \frac{1}{2} \|w\|_2^2 + C \sum_{m=1}^{M} \xi_m,$$

subject to $s_m (w^T x_m + w_0) \geq 1 - \xi_m \forall m. \tag{26}$$

The parameter $C > 0$ controls the trade-off between the slack variable penalty and the margin.

For the non-linear classification problems, the quadratic program (26) can be kernelized to make the data linearly separable in a non-linear space defined by feature vectors $\phi(x_m)$. The kernel is formed from the feature vectors by $\kappa(x_m, x'_m) = \phi(x_m)^T \phi(x'_m)$. Equation (26) can be rewritten using the Lagrangian dual

$$L(a) = \sum_{i=1}^{M} a_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} a_i a_j \xi_i \xi_j \kappa(x_i, x_j),$$

subject to $0 \leq a_i \leq C,$

$$\sum_{i=1}^{M} a_i \xi_i = 0. \tag{27}$$

Equation (27) is solved as a quadratic programming problem. From the Karush-Kuhn-Tucker conditions, either $a_i = 0$ or $s_m y_m = 1$. Points with $a_i = 0$ are not considered in the solution to Eq. (27). Thus, only points within the specified slack distance $\xi_m$ from the margin, $s_m y_m = 1 - \xi_m$, participate in the prediction. These points are called support vectors.

In Fig. 7 we use SVM with the RBF kernel (22) to classify points where the true decision boundary is either linear or circular. The SVM result is compared with linear regression (Sec. III A) and NNs (Sec. III C). Where linear regression fails on the circular decision boundary, SVM with RBF well separates the two classes. The SVM example was implemented in Python using sklearn.43

We here note that the SVM does not provide probabilistic output, since it gives hard labels of data points and not distributions. Its label uncertainties can be quantified heuristically.14

Because the SVM is a two-class model, multi-class SVM with $K$ classes requires training $K(K-1)/2$ models on all possible pairs of classes. The points that are assigned to the same class most frequently are considered to comprise a single class, and so on until all points are assigned a class from 1 to $K$. This approach is known as the “one-versus-rest” scheme, although slight modifications have been introduced to reduce computational complexity.13,14

SVMs have been used for acoustic target classification,44 underwater source localization,5 and classifying animal calls45,46 to name a few examples. For large datasets, SVMs suffer from high computational cost. Further, kernel machines with generic kernels do not generalize well. Recent developments in deep learning were designed to overcome these limitations, as evidenced by neural networks (NNs) outperforming RBF kernel SVMs on the MNIST dataset.6,47

C. Neural networks: Multi-layer perceptron

Neural networks (NNs) can overcome the limitations of linear models (linear regression, SVM) by learning a non-linear
mapping of the inputs $\phi(x_m)$ from the data over their network structure. Linear models are appealing because they can be fit efficiently and reliably, with solutions obtained in closed form or with convex optimization. However, they are limited to modeling linear functions. As we saw previously, linear models can use non-linear features by prescribing basis functions (DOA estimation) or by mapping the features into a more useful space using kernels (SVM). Yet these prescribed feature mappings are limited since kernel mappings are generic and based on the principle of local smoothness. Such general functions perform well for many tasks, but better performance can be obtained for specific tasks by training on specific data. NNs (and also dictionary learning, see Sec. IV) provide the algorithmic machinery to learn representations directly from data.$^{10, 16}$

The purpose of feed-forward NNs, also referred to as deep NNs (DNNs) or multi-layer perceptrons (MLPs), is to approximate functions. These models are called feed-forward because information flows only from the inputs (features) to the outputs (labels), through the intermediate calculations. When feedback connections are included in the network, the network is referred to as a recurrent NN (RNN) (for more details see Sec. V).

NNs are called networks because they are composed of a series of functions associated by a directed graph. Each set of functions in the NN is referred to as a layer. The number of layers in the network (see Fig. 8), called the NN depth, typically is the number of hidden layers plus one (the output layer). The NN depth is one of the parameters that affect the capacity of NNs. The term deep learning refers to NNs with many layers.$^{16}$

In Fig. 8, an example three layer fully connected NN is illustrated. The first layer, called the input layer, is the features $x_m \in \mathbb{R}^N$. The last layer, called the output layer, is the target values, or labels $y_m \in \mathbb{R}^P$. The intervening layers of the NN, called hidden layers since the training data does not explicitly define their output, are $z^{(1)} \in \mathbb{R}^Q$ and $z^{(2)} \in \mathbb{R}^R$. The circles in the network (see Fig. 8) represent network units.

The output of the network units in the hidden and output layers is a non-linear transformation of the inputs, called the activation. Common activation functions include softmax, sigmoid, hyperbolic tangent, and rectified linear units (ReLU). Activation functions are further discussed in Sec. V. Before the activation, a linear transformation is applied to the inputs

$$a_q = \sum_{n=1}^{N} w^{(1)}_{nq} x_n + w^{(1)}_{q0},$$

with $a_q$ the input to the $q$th unit of the first hidden layer, and $w^{(1)}_{nq}$ and $w^{(1)}_{q0}$ the weights and biases, which are to be learned. The output of the hidden unit $z^{(1)}_q = g_1(a_q)$, with $g_1$ the activation function. Similarly,

$$a_r = \sum_{q=1}^{Q} w^{(2)}_{rq} z^{(1)}_q + w^{(2)}_{r0},$$

$$a_p = \sum_{r=1}^{R} w^{(3)}_{rp} z^{(2)}_r + w^{(3)}_{p0},$$

and $z^{(2)}_r = g_2(a_r)$, $y_p = g_3(a_p)$. The NN architecture, combined with the series of small operations by the activation functions, make the NN a general function approximator. In fact, a NN with a single hidden layer can approximate any continuous function arbitrarily well with a sufficient number of hidden units.$^{28}$ We here illustrate a NN with two hidden layers. Deeper NN architectures are discussed in Sec. V.

NN training is analogous to the methods we have previously discussed (e.g., linear regression and SVM models): a loss function is constructed and gradients of the cost function are used to train the model. For NNs, a typical loss function, $L$, for classification is cross-entropy.$^{16}$ Given the target values (labels) $S = [s_1, \ldots, s_m] \in \mathbb{R}^{P \times M}$ and input features $X$, the average cross-entropy $L$ and weight estimate are given by

$$L(w) = -\frac{1}{P} \sum_{p=1}^{P} \sum_{m=1}^{M} s_{pm} \ln y_{pm},$$

$$\bar{w} = \arg \min_w L(w),$$

with $w$ the matrix of the weights and $\bar{w}$ its estimate. The gradient of the objective (30), $\nabla L(w)$, is obtained via backpropagation.$^{20}$ Backpropagation uses the derivative chain rule to find the gradient of the cost with respect to the weights at each NN layer. With backpropagation, any of the numerous variants of gradient descent can be used to optimize the weights at all layers.

The gradient information from backpropagation is used to find the optimal weights. The simplest weight update is obtained by taking a small step in the direction of the negative gradient

$$w_{\text{new}} = w_{\text{old}} - \eta \nabla L(w_{\text{old}}),$$

with $\eta$ called the learning rate, which controls the step size. Popular NN training algorithms are stochastic gradient descent$^{16}$ and Adam (adaptive moment estimation).$^{39}$

The choice of activation functions for the hidden and output layers are determined by 4 important NN
phrases.56,57 Aside from compression, such methods can be used based methods have been developed for travel time tomography. The data, represented by the data model parameters \( \theta \), give probabilistic unsupervised learning models of the form \( p(Y|X; \theta) \). This is in contrast to supervised models that predict the probability of labels or regression values given the data and model: \( p(Y|X; \theta) \) (see Sec. III). We note that the distinction between unsupervised and supervised learning methods is not always clear. Generally, a learning problem can be considered unsupervised if there are no annotated examples or prediction targets provided.

The structures discovered in unsupervised learning serve many purposes. The models learned can, for example, indicate how features are grouped or define latent representations of the data such as the subspace or manifold which the data occupies in higher-dimensional space. Unsupervised learning methods for grouping features include clustering algorithms such as K-means\(^ {18} \) and Gaussian mixture models (GMMs). Unsupervised methods for discovering latent models include principal components analysis (PCA), matrix factorization methods such as non-negative matrix factorization (NMF),\(^ {50} \) independent component analysis (ICA),\(^ {51} \) and dictionary learning.\(^ {24,25,28,52} \) Neural network models, called autoencoders, are also used for learning latent models.\(^ {16} \) Autoencoders can be understood as a non-linear generalization of PCA and, in the case of sparse regularization (see Sec. III), dictionary learning.

The aforementioned models of unsupervised learning have many practical uses. Often, they are used to find the “best” representation of the data given a desired task. A special class of K-means based techniques, called vector quantization,\(^ {53} \) was developed for lossy compression. In sparse modeling, dictionary learning seeks to learn the “best” sparsifying dictionary of basis functions for a given class of data. In ocean acoustics, PCA (a.k.a. empirical orthogonal functions) have been used to constrain estimates of ocean sounds speed profiles (SSPs), though methods based on sparse modeling and dictionary learning have given an alternative representation.\(^ {54,55} \) Recently, dictionary-learning based methods have been developed for travel tomography.\(^ {56,57} \) Aside from compression, such methods can be used for data restoration tasks such as denoising and inpainting.

Methods developed for denoising and inpainting can also be extended to inverse problems more generally. In the following, we illustrate unsupervised ML, highlighting PCA, EM with GMMs, K-means, dictionary learning, and autoencoders.

### A. Principal components analysis

For data visualization and compression, we are often interested in finding a subspace of the feature space which contains the most important feature correlations. This can be a subspace which contains the majority of the feature variance. PCA finds such a subspace by learning an orthogonal, linear transformation of the data. The principal components of the features are obtained as the right singular vector of the design matrix \( X \) (or eigenvector of \( X^T X \)) with

\[
X^T X = \Sigma \Sigma^T.
\]

\( \Sigma = [p_1, \ldots, p_N] \in \mathbb{R}^{N \times N} \) are principal components (eigenvectors) and \( \Sigma^2 = \text{diag}(\sigma_1^2, \ldots, \sigma_N^2) \in \mathbb{R}^{N \times N} \) are the total variances of the data along the principal directions defined by principal components \( p_n \), with \( \sigma_1^2 \geq \ldots \geq \sigma_N^2 \). This matrix factorization can be obtained using, for example, singular value decomposition.\(^ {21} \)

In the coordinate system defined by \( \Sigma \), with axes \( p_n \), the first coordinate accounts for the highest portion of the overall variance in the data and subsequent axes have equal or smaller contributions. Thus, truncating the resulting coordinate space results in a lower dimensional representation that often captures a large portion of the data variance. This has benefits both for visualization of data and modeling as it can reduce the aforementioned curse of dimensionality (see Sec. II E). Formally, the projection of the original features \( X \) onto the principal components \( \Sigma \) is

\[
B^T = XQ,
\]

with \( Q \in \mathbb{R}^{N \times P} \) the first \( P \) eigenvectors and \( B = [b_1, \ldots, b_M] \in \mathbb{R}^{P \times M} \) the lower-dimensional projection of the data. \( X \) can be approximated by

\[
\hat{X}^T \approx QB,
\]

which give a compressed version data \( \hat{X} \) with less information than the original data \( X \) (lossy compression).

PCA is a simple example of representation learning that attempts to disentangle the unknown factors generating the data variance. The principal variances quantify the importance of the features, and the principal components are a coordinate system under which the features are uncorrelated. While correlation is an important feature dependency, we often are interested in learning representations that can disentangle more complicated, perhaps correlated, dependencies.

### B. Expectation maximization and Gaussian mixture models

Often, we would like to model the dependency between observed features. An efficient way of doing this is to
assume that the observed variables are correlated because they are generated by a hidden or latent model. Such models can be challenging to fit but offer advantages, including a compressed representation of the data. A popular latent modeling technique called Gaussian mixture models (GMMs) models arbitrary probability distributions as a linear superposition of K Gaussian densities. The latent parameters of GMMs (and other mixture models) can be obtained using a non-linear optimization procedure called the expectation-maximization (EM) algorithm. EM is an iterative technique which alternates between (1) finding the expected value of the latent factors given data and initialized parameters, and (2) optimizing parameter updates based on the latent factors from (1). We here derive EM in the context of GMMs and later show how it relates to other popular algorithms, like K-means.

For features \( x_m \), the GMM is

\[
p(x_m) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_m | \mu_k, \Sigma_k),
\]

with \( \pi_k \) the weights of the Gaussians in the mixture, and \( \mu_k \) and \( \Sigma_k \) the mean and covariance of the \( k \)th Gaussian. The weights \( \pi_k \) define the marginal distribution of a binary random vector \( z_m \in \{0, 1\}^K \), which give membership of data vector \( x_m \) to the \( k \)th Gaussian (\( z_{km} = 1 \) and \( z_{jm} = 0 \forall j \neq k \)).

The features \( x_m \) are related to the latent vector \( z_m \) and the parameters \( \theta = \{ \pi_k, \mu_k, \Sigma_k \} \) via conditional and joint distributions. The conditional distribution \( p(x_m | z_m) \) is obtained using the sum rule (3),

\[
p(x_m | \theta) = \sum_{z_m} p(x_m | z_m, \theta) p(z_m | \theta) = \sum_{z_m} p(x_m, z_m | \theta).
\]

To find the parameters, the log-likelihood or \( p(x_m | \theta) \) is maximized over observations \( X = [x_1, \ldots, x_M] \).

\[
\ln p(X | \theta) = \sum_{m=1}^{M} \ln \left\{ \sum_{z_m} p(x_m, z_m | \theta) \right\}.
\]

Equation (37) is challenging to optimize because the logarithm cannot be pushed inside the summation over \( z_m \).

In EM, a complete data log likelihood

\[
L(\theta) = \sum_{m=1}^{M} \ln p(x_m, z_m | \theta)
\]

is used to define an auxiliary function, \( Q(\theta, \theta^{old}) = E[L(\theta) | \theta^{old}] \), which is the expectation of the likelihood evaluated assuming some knowledge of the parameters. The knowledge of the parameters is based on the previous or “old” values, \( \theta^{old} \). The EM algorithm is derived using the auxiliary function. For more details, please see Ref. 14 (pp. 350–354). Helpful discussion is also presented in Ref. 13 (pp. 430–443).

The first step of EM, called the E-step (for expectation), estimates the responsibility \( r_{km} \) of the \( k \)th Gaussian in reconstructing the \( m \)th data density \( p(x_m) \) given the current parameters \( \theta \). From Bayes’ rule, the E-step is

\[
r_{km} = \frac{\pi_k \mathcal{N}(x_m | \mu_k^{old}, \Sigma_k^{old})}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_m | \mu_j^{old}, \Sigma_j^{old})}.
\]

The second step of EM, called the M-step, updates the parameters by maximizing the auxiliary function, \( \theta^{new} = \arg \max_{\theta} Q(\theta, \theta^{old}) \), with the responsibilities \( r_{km} \) from the E-step (39). The M-step estimates of \( \pi \) (using also \( \Sigma_k^{old} = \frac{1}{M} \sum_{m=1}^{M} x_m x_m \), \( \mu_k^{new} = \frac{1}{r_k} \sum_{m=1}^{M} r_{km} x_m \), \( \Sigma_k^{new} = \frac{1}{r_k} \sum_{m=1}^{M} r_{km} (x_m - \mu_k^{new})(x_m - \mu_k^{new})^T \), with \( r_k = \sum_{m=1}^{M} r_{km} \) the weighted number of points with membership to centroid \( k \). The EM algorithm is run until an acceptable error has been obtained. The error can be obtained for example by evaluating the log likelihood (37) with the estimated parameters (40).

We note that singularities can arise in the maximum likelihood approach to EM, presented here. If only one data point is assigned to a Gaussian (and there is more than one Gaussian), the log likelihood function (37) goes to infinity as the variance of the Gaussian component with a single data point goes to zero. This does not occur in a Bayesian formulation.

In EM, the objective function is not convex and solutions often can get caught in local minima. These issues can be corrected, in part, using multiple parameter initializations and choosing the results with the smallest residual. In ML, local minima are a common challenge as optimization objectives are rarely convex. This is an especially large issue in DL and has driven significant development in DL algorithms (see Sec. V).

GMMs (EM) have been used extensively in acoustics. A few of the applications include source localization, separation, and speech enhancement. These applications are further discussed in Sec. VI. GMMs have also been used in animal vocalization classification.

C. K-means

The K-means algorithm is a method for discovering clusters of features in unlabeled data. The goal of doing this can be to estimate the number of clusters or for data compression (e.g., vector quantization). Like EM, K-means solves Eq. (37). Except, unlike EM, \( \pi_k = \frac{1}{K} \) and \( \Sigma_k = \sigma^2 I \) are fixed. Rather than responsibility \( r_{km} \) describing the posterior distribution of \( z_m \) [per Eq. (39)], in K-means the
membership is a “hard” assignment (in the limit $\sigma \to 0$, please see Ref. 13 for more details),
\[
r_{km} = \begin{cases} 
1 & \text{if } \hat{k} = \arg\min_k ||x_m - \mu_k^{\text{old}}||_2, \\
0 & \text{otherwise.}
\end{cases}
\]

Thus in K-means, each feature vector $x_m$ is assigned to the nearest centroid $\mu_k$. The distance measure is the Euclidean distance [defined by the $\ell_2$-norm, Eq. (41)]. Based on the centroid membership of the features, the centroids are updated using the mean of the feature vectors in the cluster
\[
\mu_k^{\text{new}} = \frac{1}{r_k} \sum_{r_{km} = 1} x_r.
\]

Sometimes the variances are also calculated. Thus, K-means is a two-step iterative algorithm which alternates between categorizing the features and updating the centroids. Like EM, K-means must be initialized, which can be done with random initial assignments. The number of clusters can be estimated using, for example, the gap statistic.\(^{2,1}\)

D. Dictionary learning

In this section we introduce dictionary learning and discuss one classic dictionary learning method: the K-SVD algorithm.\(^{62}\) An important task in sparse modeling (see Sec. III) is obtaining a dictionary which can well model a given class of signals. There are a number of methods for dictionary design, which can be divided roughly into two classes: analytic and synthetic. Analytic dictionaries have columns, called atoms, which are derived from analytic functions such as wavelets or the discrete cosine transform (DCT).\(^{24,63}\) Such dictionaries have useful properties, which allow them to obtain acceptable sparse representation performance for a broad range of data. However, if enough training examples of a specific class of data are available, a dictionary can be synthesized or learned directly from the data. Learned dictionaries, which are designed from specific instances of data using dictionary learning algorithms, often achieve greater reconstruction accuracy over analytic, generic dictionaries. Many dictionary learning algorithms are available.\(^{25}\)

As discussed in Sec. III, sparse modeling assumes that a few (sparse) atoms from a dictionary $D \in \mathbb{R}^{N \times K}$ can adequately construct a given feature $x_m$. With coefficients $\beta_m \in \mathbb{R}^K$, this is articulated as $x_m \approx D \beta_m$. The coefficients can be solved by
\[
\hat{\beta}_m = \arg\min_{\beta} \left\| D \beta - x_m \right\|_2^2 \text{ subject to } \| \beta_m \|_0 = T,
\]
Equation (43) can be modified to also solve for the dictionary,\(^{24}\)
\[
\hat{D}, \hat{\beta} = \arg\min_D \left\{ \arg\min_{\beta} \left\| D \beta - x_m \right\|_2^2 \right\}
\]
\[
\text{subject to } \| \beta_m \|_0 = T \forall m, \tag{44}
\]
\[\text{with } B = [\beta_1, \ldots, \beta_M] \text{ the coefficients for all examples. Equation (44) is a bi-linear optimization problem for which no general practical algorithm exists.}\(^{24}\) However, it can be solved well using methods related to K-means. Clustering-based dictionary learning methods\(^{28}\) are based on the alternating optimization concept introduced in K-means and EM. The operations of a dictionary learning algorithm are (1) sparse coding given dictionary $D$, and (2) dictionary update based coefficients $B$.

This assumes an initial dictionary (the columns of which can be Gaussian noise). Sparse coding can be accomplished by OMP or other greedy methods. The dictionary update stage can be approached in a number of ways. We next briefly describe the class K-SVD dictionary learning algorithm\(^ {24,62}\) to illustrate basic dictionary learning concepts. Like K-means, K-SVD learns $K$ prototypes of the data (in dictionary learning these are called atoms, where in K-means they are called centroids) but, instead of learning them as the means of the data “clusters,” they are found using the SVD since there may be more than one atom used per data point. In the K-SVD algorithm, dictionary atoms are learned based on the SVD of the reconstruction error caused by excluding the atoms from the sparse reconstruction. For more details please see Ref. 24.

Expressing the dictionary coefficients as row vectors $\beta_T \in \mathbb{R}^N$ and $\beta_T \in \mathbb{R}^N$, which relate all examples $X$ to $d_n$ and $d_j$, respectively, the $\ell_2$-penalty from Eq. (44) is rewritten as
\[
\left\| X^T - D B \right\|_F^2 = \left\| X^T - \sum_{k=1}^K d_k \beta_T^k \right\|_F^2 = \| E_j - d_j \beta_T^j \|_F^2, \tag{45}\]
Equation (43) exactly is often impractical. However, many fast-approximate solution methods exist, including orthogonal matching pursuit (OMP)\(^{24}\) and sparse Bayesian learning (SBL).\(^{64}\)

where $E_j = (X^T - \sum_{k \neq j} d_k \beta_T^k), \tag{46}$
and $\| \cdot \|_F$ is the Frobenius norm.

An update to the dictionary entry $d_j$ and coefficients $\beta_T^j$ which minimizes Eq. (45) is found by taking the SVD of $E_j$. However, many of the entries in $\beta_T^j$ are zero (corresponding to examples which do not use $d_j$). To properly update $d_j$ and $\beta_T^j$ with SVD, Eq. (45) must be restricted to examples $x_m$ which use $d_j$,
\[
\left\| E_j^R - d_j \beta_T^j \right\|_F^2, \tag{47}\]
where $E_j^R$ and $\beta_T^j$ are entries in $E_j$ and $\beta_T^j$, respectively, corresponding to examples which use $d_j$. Thus for each K-SVD iteration, the dictionary entries and coefficients are
sequentially updated as the SVD of \( E_j^T = USV^T \). The dictionary entry \( d_j \) is updated with the first column in \( U \) and the coefficient vector \( \beta_j \) is updated as the product of the first singular value \( S(1, 1) \) with the first column of \( V \).

For the case when \( T = 1 \), the results of K-SVD reduces to the K-means based model called gain-shape vector quantization.\(^{24, 53}\) When \( T = 1 \), the \( \ell_2 \)-norm in Eq. (44) is minimized by the dictionary entry \( d_j \) that has the largest inner product with example \( x_m \).\(^{24}\) Thus for \( T = 1 \), \([d_1, \ldots, d_N]\) define radial partitions of \( \mathbb{R}^d \). These partitions are shown in Fig. 9(b) for a hypothetical 2D \((K = 2)\) random dataset.

Other clustering-based dictionary learning methods are the method of optimal directions\(^{65}\) and the iterative thresholding and signed K-means algorithm.\(^{66}\) Alternative methods include online dictionary learning.\(^{67}\)

Dictionary learning has been applied in a number of applications in acoustics. The applications include acoustic signal denoising,\(^{68}\) geophysical parameter compression (ocean acoustics),\(^{55}\) seismic tomography,\(^{56, 69}\) and damage detection.\(^{70}\)

E. Autoencoder networks

Autoencoder networks are a special case of NNs (Sec. III), in which the desired output is an approximation of the input. Because they are designed to only approximate their input, autoencoders prioritize which aspects of the input should be copied. This allows them to learn useful properties of the data. Autoencoder NNs are used for dimensionality reduction and feature learning, and they are a critical component of modern generative modeling.\(^{16}\) They can also be used as a pretraining step for DNNs (see Sec. V B). They can be viewed as a non-linear generalization of PCA and dictionary learning. Because of the non-linear encoder and decoder functions, autoencoders potentially learn more powerful feature representations than PCA or dictionary learning.

Like feed-forward NNs (Sec. III C), activation functions are used on the output of the hidden layers (Fig. 8). In the case of an autoencoder with a single hidden layer, the input to the hidden layer is \( z_1 = g_1(a_p(x)) \) and the output is \( \hat{x} = g_2(a_p(z_1)) \), with \( P = M \) (see Fig. 8). The first half of the NN, which maps the inputs to the hidden units is called the encoder. The second half, which maps the output of the hidden units to the output layer (with same dimension \( N \) of input features) is called the decoder. The features learned in this single layer network are the weights of the first layer.

If the code dimension is less than the input dimension, the autoencoder is called undercomplete. In having the code dimension less than the input, undercomplete networks are well suited to extract salient features since the representation of the inputs is “compressed,” like in PCA. However, if too much capacity is permitted in the encoder or decoder, undercomplete autoencoders will still fail to learn useful features.\(^{16}\)

Depending on the task, code dimension equal to or greater than the inputs is desirable. Autoencoders with code dimension greater than the input dimension are called overcomplete and these codes exhibit redundancy similar to overcomplete dictionaries and CNNs. This can be useful for learning shift invariant features. However, without regularization, such autoencoder architectures will fail to learn useful features. Sparsity regularization, similar to dictionary learning, can be used to train overcomplete autoencoder networks.\(^{16}\) For more details and discussion, please see Sec. V.

Like other unsupervised methods, autoencoders can be used to find transformations of the parameters for data interpretation and visualization. They can also be used for feature extraction in conjunction with other ML methods. Applications of autoencoders in acoustics include speech enhancement\(^{71}\) and acoustic novelty detection.\(^{72}\)

V. DEEP LEARNING

Deep learning (DL) refers to ML techniques that are based on a cascade of non-linear feature transforms trained during a learning step.\(^{73}\) In several scientific fields, decades of research and engineering have led to elegant ways to model data. Nevertheless, the DL community argues that these models often do not have enough capacity to capture the subtleties of the phenomena underlying data and are perhaps too specialized. And often it is beneficial to learn the representation directly from a large collection of examples using high-capacity ML models. DL leverages a
fundamental concept shared by many successful handcrafted features: all analyze the data by applying filter banks at different scales. These multi-scale representations include Mel frequency cepstrum used in speech processing, multi-scale wavelets,\textsuperscript{74} and scale invariant feature transform (SIFT)\textsuperscript{75} used in image processing. DL mimics these processes by learning a cascade of features capturing information at different levels of abstraction. Non-linearities between these features allow deep NNs (DNNs) to learn complicated manifolds. Findings in neuroscience also suggest that mammal brains process information in a similar way.

In short, a NN-based ML pipeline is considered DL if it satisfies:\textsuperscript{73} (i) features are not handcrafted but learned, (ii) features are organized in a hierarchical manner from low- to high-level abstraction, (iii) there are at least two layers of non-linear feature transformations. As an example, applying DL on a large corpus of conversational text must uncover meanings behind words, sentences and paragraphs (low-level) to further extract concepts such as lexical field, genre, and writing style (high-level).

To comprehend DL, it is useful to look at what it is not. MLPs with one hidden layer (a.k.a., shallow NNs) are not deep as they only learn one level of feature extraction. Similarly, non-linear SVMs are analogous to shallow NNs. Multi-scale wavelet representations\textsuperscript{76} are a hierarchy of features (sub-bands) but the relationships between features are linear. When a NN classifier is trained on (hand-engineered) transformed data, the architecture can be deep, but it is not DL as the first transformation is not learned.

Most DL architectures are based on DNNs, such as MLPs, and their early development can be traced to the 1970–1980s. Three decades after this early development, only a few deep architectures emerged. And these architectures were limited to process data of no more than a few hundred dimensions. Successful examples developed over this intervening period are the two handwritten digit classifiers: Neocognitron\textsuperscript{77} and LeNets.\textsuperscript{78} Yet the success of DL started at the end of the 2000s on what is called the third wave of artificial NNs. This success is attributed to the large increase in available data and computation power, including parallel architectures and GPUs. Nevertheless, several open-source DL toolboxes\textsuperscript{79–82} have helped the community in introducing a multitude of new strategies. These aim at fighting the limitations of back-propagation: its slowness and tendency to get trapped in poor stationary points (local optima or saddle points). The following describes some of these strategies, see Ref. 16 for an exhaustive review.

A. Activation functions and rectifiers

The earliest multi-layer NN used logistic sigmoids (Sec. III C) or hyperbolic tangent for the non-linear activation function $g$.

$$ z'_l = g(a'_l), \text{ where } a'_l = W_l z_{l-1} + b'_l, $$ (48)

where $z'_l$ is the vector of features at layer $l$ and $a'_l$ are the vector of potentials (the affine combination of the features from the previous layer). For the sigmoid activation function in Fig. 10(a), the derivative is significantly non-zero for only $a$ near 0. With such functions, in a randomly initialized NN, half of the hidden units are expected to activate $[f(a) > 0]$ for a given training example, but only a few will influence the gradient, as $a \gg 0$. In fact, many hidden units will have near-zero gradient for all training samples, and the parameters responsible for those units will be slowly updated. This is called the \textit{vanishing gradient problem}. A naive repair to the problem is to increase the learning rate. However, parameter updates will become too large for small $a$. Due to this, the overall training procedure might be unstable: this is the \textit{gradient exploding problem}. Figure 10(b) indicates of these two problems. Shallow NNs are not necessarily susceptible to these problems, but they become harmful in DNNs. Back-propagation with the aforementioned activation functions in DNNs is slow, unstable, and leads to poor solutions.

Alternative activations have been developed to address these issues. One important class is rectifier units. Rectifiers are activation functions that are zero-valued for negative-valued inputs and linear for positive-valued inputs. Currently, the most popular is the rectifier linear unit (ReLU),\textsuperscript{83} defined as (see Fig. 10)

$$ g(a) = \text{ReLU}(a) \triangleq \max(a, 0). $$ (49)

While the derivative is zero for negative potentials $a$, the derivative is one for $a > 0$ (though non-differentiable at 0, ReLU is continuous and then back-propagation is a sub-gradient descent). Thus, in a randomly initialized NN, half of the hidden units fire and influence the gradient, and half do not fire (and do not influence the gradient). If the weights are randomly initialized with zero-mean and variance that preserves the range of variations of all potentials across all NN layers, most units get significant gradients from at least half of the training samples, and all parameters in the NN are expected to be equally updated at each epoch.\textsuperscript{84,85} In practice, the use of rectifiers leads to tremendous improvement in convergence. Regarding exploding gradients, an efficient solution called gradient clipping\textsuperscript{86} simply consists in thresholding the gradient.
B. End-to-end training

While important for successful DL models, only addressing vanishing or exploding gradient problems is not alone enough for back-propagation. It is also important to avoid poor stationary points in DNNs. Pioneering methods for avoiding these stationary points included training DNNs by successively training shallow architectures in an unsupervised way.47,87 Because the individual layers in this case are initially trained sequentially, using the output of preceding layers without optimizing jointly the weights of the preceding layer, these approaches are termed as greedy layer-wise unsupervised pretraining.

However, the benefits of unsupervised pretraining are not always clear. Many modern DL approaches prefer to train networks end-to-end, training all the network layers jointly from initialization instead of first training the individual layers.16 They rely on variants of gradient descent that aim at fighting poor stationary solutions. These approaches include stochastic gradient descent, adaptive learning rates,88 and momentum techniques.86 Among these concepts, two main notions emerged: (i) annealing by randomly exploring configurations first and exploiting them next and (ii) momentum which forms a moving average of the negative gradient called velocity. This tends to give faster learning, especially for noisy gradients or high-curvature loss functions.

Adam49 is based on adaptive learning rate and moment estimation. It is currently the most popular optimization approach for DNNs. Adam updates each weight \( w_{ij} \) at each step \( t \) as follows:

\[
\begin{align*}
    w_{ij}^{(t+1)} &= w_{ij}^{(t)} - \frac{\eta}{\sqrt{\hat{v}_{ij}^{(t)}} + \epsilon} \hat{m}_{ij}^{(t)}, \\
    \hat{m}_{ij}^{(t)} &= m_{ij}^{(t)} / (1 - \beta_1^t), \\
    \hat{v}_{ij}^{(t)} &= v_{ij}^{(t)} / (1 - \beta_2^t), \\
    m_{ij}^{(t)} &= \beta_1 m_{ij}^{(t-1)} + (1 - \beta_1) \frac{\partial L(W^{(t)})}{\partial w_{ij}^{(t)}}, \\
    v_{ij}^{(t)} &= \beta_2 v_{ij}^{(t-1)} + (1 - \beta_2) \left( \frac{\partial L(W^{(t)})}{\partial w_{ij}^{(t)}} \right)^2.
\end{align*}
\]

(50)

with \( \eta > 0 \) the learning rate, \( \epsilon > 0 \) a smoothing term, and \( \hat{m}_{ij}^{(t)} \) and \( \hat{v}_{ij}^{(t)} \) the first and second moment of the velocity estimated, for \( 0 < \beta_1 < 1 \) and \( 0 < \beta_2 < 1 \), as

\[
\begin{align*}
    m_{ij}^{(t)} &= \beta_1 m_{ij}^{(t-1)} + (1 - \beta_1) \frac{\partial L(W^{(t)})}{\partial w_{ij}^{(t)}}, \\
    v_{ij}^{(t)} &= \beta_2 v_{ij}^{(t-1)} + (1 - \beta_2) \left( \frac{\partial L(W^{(t)})}{\partial w_{ij}^{(t)}} \right)^2.
\end{align*}
\]

(51)

Gradient descent methods can fall into the local minima near the parameter initialization, which leads to underfitting. On the contrary, stochastic gradient descent and variants are expected to find solutions with lower loss and are more prone to overfitting. Overfitting occurs when learning a model with many degrees of freedom compared to the number of training samples. The curse of dimensionality (Sec. II E) claims that, without assumptions on the data, the number of training data should grow exponentially with the number of free parameters. In classical NNs, an output feature is influenced by all input features, a layer is fully connected (FC). Given an input of size \( N \) and a feature vector of size \( P \), a FC layer is then composed of \( N \times (P + 1) \) weights (including a bias term, see Sec. III C). Given that the signal size \( N \) can be large, FC NNs are prone to overfitting. Thus, special care should be taken for initializing the weights, and specific strategies must be employed to have some regularization, such as dropout90 and batch-normalization.91

With dropout, at each epoch during training, different units for each sample are dropped randomly with probability \( 1 - p \), \( 0 < p \leq 1 \). This encourages NN units to specialize in detecting particular patterns, and subsequently features to be sparse. In practice, this also makes the optimization faster. During testing, all units are used and the predictions are multiplied by \( p \) (such that all units behave as if trained without dropout).

With batch-normalization, the outputs of units are normalized for the given mini-batch. After normalization into standardized features (zero mean with unit variance), the features are shifted and rescaled to a range of variation that is learned by backpropagation. This prevents units having to constantly adapt to large changes in the distribution of their inputs (a problem known as internal covariate shift). Batch-normalization has a slight regularization effect, allowing for a higher learning rate and faster optimization.

C. Convolutional neural networks

Convolutional NNs (CNNs)77,78 are an alternative to conventional, fully connected NNs for temporally or spatially correlated signals. They limit dramatically the number of parameters of the model and memory requirements by relying on two main concepts: local receptive fields and shared weights. In fully connected NNs, for each layer, every output interacts with every input. This results in an excessive number of weights for large input dimension [number of weights is \( O(N \times P) \)]. In CNNs, each output unit is connected only with subsets of inputs corresponding to given filter (and filter position). These subsets constitute the local receptive field. This significantly reduces the number of NN multiplication operations on the forward pass of a convolutional layer for a single filter to \( O(N \times K) \), with \( K \) typically a factor 100 smaller than \( N \) and \( P \). Further, for a given filter, the same \( K \) weights are used for all receptive fields. Thus the number of parameters for each layer and weight is reduced from \( O(N \times P) \) to \( O(K) \).

Weight sharing in CNNs gives another important property called shift invariance. Since for a given filter, the weights are the same for all receptive fields, the filter must model well signal content that is shifted in space or time. The response to the same stimuli is unchanged whenever the stimuli occurs within overlapping receptive fields. Experiments in neuroscience reveal the existence of such a behavior (denoted self-similar receptive fields) in simple cells of the mammal visual cortex.92 This principle leads CNNs to consider convolution layers with linear filter banks on their inputs.

Figure 11 provides an illustration of one convolution layer. The convolution layer applies three filters to an input
signal $x$ to produce three feature maps. Denoting the $q$th input feature map at layer $l$ as $z_p^{(l-1)}$ and the $p$th output feature map at layer $l$ as $\bar{z}_p^{(l)}$, a convolution layer at layer $l$ produces $C_{out}$ new feature maps from $C_{in}$ input feature maps as follows:

$$\bar{z}_p^{(l)} = g \left( \sum_{q=1}^{C_{in}} w_{pq}^{(l)} * z_q^{(l-1)} + b_p^{(l)} \right) \quad \text{for} \quad p = 1, \ldots, C_{out},$$

(54)

where * is the discrete convolution, $w_{pq}^{(l)}$ are $C_{out} \times C_{in}$ learned linear filters, $b_p^{(l)}$ are $C_{out}$ learned scalar bias, $p$ is an output channel index, and $q$ an input channel index. Stacking all feature maps $\bar{z}_p^{(l)}$ together, the set of hidden features is represented as a tensor $\bar{z}^{(l)}$ where each channel corresponds to a given feature map.

For example, a spectrogram is represented by a $N \times C$ tensor where $N$ is the signal length and the number of channels $C$ is the number of frequency sub-bands. Convolution layers preserve the spatial or temporal resolution of the input tensor, but usually increase the number of channels: $C_{out} \geq C_{in}$. This produces a redundant representation which allows for sparsity in the feature tensor. Only a few units should fire for a given stimulus: a concept that has also been influenced by vision research experiments. Using tensors is a common practice allowing us to represent CNN architectures in a condensed way, see Fig. 12.

Local receptive fields impose that an output feature is influenced by only a small temporal or spatial region of the input feature tensor. This implies that each convolution is restricted to a small sliding centered kernel window of odd size $K$, for example, $K = 3 \times 3 = 9$ is a common practice for images. The number of parameters to learn for that layer is then $C_{out} \times (C_{in} \times K + 1)$ and is independent on the input signal size $N$. In practice $C_{in}$, $C_{out}$ and $K$ are chosen so small that it is robust against overfitting. Typically, $C_{in}$ and $C_{out}$ are less than a few hundreds. A byproduct is that processing becomes much faster for both learning and testing.

Applying $D$ convolution layers of support size $K$ increases the region of influence (called effective receptive field) to a $D(K-1) + 1$ window. With only convolution layers, such an architecture must be very deep to capture long-range dependencies. For instance, using filters of size $K = 3$, a 10 deep architecture will process inputs in sliding windows of only size 21.

To capture larger-scale dependencies, CNNs introduce a third concept: pooling. While convolution layers preserve the spatial or temporal resolution, pooling preserves the number of channels but reduces the signal resolution. Pooling is applied independently on each feature map as

$$\bar{z}_p^{(l)} = \text{pooling}(z_p^{(l)}), \quad \text{for} \quad p = 1, \ldots, C_{out}$$

(55)

and such that $z_p^{(l)}$ has a smaller resolution than $\bar{z}_p^{(l)}$. Max-pooling of size 2 is commonly employed by replacing in all directions two successive values by their maximum. By alternating $D$ convolution and pooling layers, the effective receptive field becomes of size $2^{D-1}(K+1) - 1$. Using filters of size $K = 3$, a 10 deep architecture will have an effective receptive field of size 2047 and can thus capture long-range dependencies.

**FIG. 11.** The first layer of a traditional CNN. For this illustration we chose a first hidden layer extracting three feature maps. The filters have the size $K = 3 \times 3$. 
Pooling is also grounded on neuroscientific findings about the mammalian visual cortex. Neural cells in the visual cortex condense the information to gain invariance and robustness against small distortions of the same stimuli. Deeper tensors become more elongated with more channels and smaller signal resolution. Hence, the deeper the CNN architecture, the more robust the CNN becomes relative to the exact locations of stimuli in the receptive field. Eventually the tensor becomes flat, meaning that it is reduced to a vector. Features in that tensor are no longer temporally or spatially related and they can serve as input feature vectors for a classifier. The output tensor is not always exactly flat, but then the tensor is mapped into a vector. In general, a MLP with two hidden FC layers is employed and the architecture is trained end-to-end by back-propagation. In particular, a MLP with two hidden FC layers is used as input feature vectors for a classifier. The output tensor is mapped into a vector. In general, a MLP with two hidden FC layers is employed and the architecture is trained end-to-end by back-propagation.

This type of architecture is typical of modern image classification NNs such as AlexNet and ZFnet, but was already employed in Neocognitron and LeNet. The main difference is that modern architectures can deal with data of much higher dimensions as they employ the aforementioned strategies (such as rectifiers, Adam, dropout, batch-normalization). A trend in DL is to make such CNNs as deep as possible with the least number of parameters by employing specific architectures such as inception modules, depth-wise separable convolutions, skip connections, and dense architectures.

Since 2012, such architectures have led to state of the art classification in computer vision, even rivaling human performances on the ImageNet challenge. Regarding acoustic applications, this architecture has been employed for broadband DOA estimation where each class corresponds to a given time frame.

**D. Transfer learning**

Training deep classifiers from scratch requires using large labeled datasets. In many applications, these are not available. An alternative is using transfer learning. Transfer learning reuses parts of a network that were trained on a large and potentially unrelated dataset for a given ML task. The key idea in transfer learning is that early stages of a deep network learn generic features that may be applicable to other tasks. Once a network has learned such a task, it is often possible to remove the feed forward layers at the end of the network that are tailored exclusively to the trained task. These are then replaced with new classification or regression layers, and the learning process finds the appropriate weights of these final layers on the new task. If the previous representation captured information relevant to the new task, they can be learned with a much smaller dataset. In this vein, deep autoencoders (see Sec. IV E) can be used to learn features from a large unlabeled dataset. The learned encoder is next used as a feature extractor after which a classifier can be trained on a small labeled dataset (see Fig. 13). Eventually, after the classifier has been trained, all the layers will be slightly adjusted by performing a few backpropagation steps end-to-end (referred to as fine tuning). Many modern DL techniques rely on this principle.

**E. Specialized architectures**

Beyond classification, there exists myriad NN and CNN architectures. Fully convolutional and U-net architectures, which are enhanced CNNs, are widely used for regression problems such as signal enhancement, segmentation, or object localization. Recurrent NNs (RNNs) are an alternative to classical feed-forward NNs to process or produce sequences of variable length. In particular, long short term memory networks (called LSTMs) are a specific type of RNN that have produced remarkable results in several applications where temporal correlations in the data is significant. Applications include speech processing and natural language processing. Recently, NNs have gained much attention in unsupervised learning tasks. One key example is data generation with variational autoencoders and generative adversarial networks (GANs). The later relies on an original idea grounded on game theory. It performs a two player game between a generative network and a discriminative one. The generator learns the distribution of the data such that it can produce fake data from random seeds. Concurrently, the discriminator learns the boundary between real and fake data such that it can distinguish the fake data from the ones of the training set. Both NNs compete against each other. The generator tries to fool the discriminator such that the
fake data cannot be distinguished from the ones of the training set.

**F. Applications in acoustics**

DL has yielded promising advances in acoustics. The data-driven DL approaches provide good results relative to conventional or hand-engineered signal processing methods in their respective fields. Aside from improvements in performance, DL (and also ML generally) can provide a general framework for performing acoustics tasks. This is an alternative paradigm to developing highly specialized algorithms in the individual subfields. However, an important challenge across all fields is obtaining sufficient training data. To properly train DNNs in audio processing tasks, hours of representative audio data may be required. Since large amounts of training data might not be available, DL is not always practical. Though scarcity of training data can be addressed partly by using synthetic training data or data augmentation. In the following we highlight recent advances in the application of DL in acoustics.

Two tasks in acoustics and audio signal processing that have benefited from DL are sound event detection and source localization. These methods replace physics-based acoustic propagation models or hand-engineered detectors with deep-learning architectures. In Ref. 105, convolutional recurrent NNs achieve state-of-the-art results in the sound event detection task in the 2017 Detection and Classification of Acoustic Scenes and Events (DCASE) challenge. In Ref. 96, CNNs are developed for broadband DOA estimation which use only the phase component of the STFT. The CNNs obtain competitive results with steered response power phase transform (SRP-PHAT) beamforming. The CNN was trained using synthetically generated noise and it generalized well to speech signals. In Ref. 107 the event detection and DOA estimation tasks are combined into a signal DNN architecture based on convolutional RNNs. The proposed system is used with synthetic and real-world, reverberant and anechoic data, and the DOA performance is competitive with MUSIC. In Ref. 104, DL is used to localize ocean sources in a shallow ocean waveguide using a single hydrophone, as shown in Fig. 14. Two deep residual NNs (50-layers each, ResNet50) are trained to localize the range and depth of a source using millions of synthetic acoustic fields. The ResNet50 DL model achieves competitive source range and depth prediction error when compared to popular genetic algorithm-based inversion methods. The source (range or depth) prediction error defined here is the percentage of predictions with maximum error below a given value, with given values for range and depth defined along the x axis in Fig. 14.

DL has also been applied in speech modeling, source separation, and enhancement. In Ref. 110 a deep clustering approach is proposed, based on spectral clustering, which uses a DNN to find embedding features for each time-frequency region of a spectrogram. This is applied to the problem of separating two speakers of the same gender, but can be applied to problems where multiple sources of the same class are active. In Ref. 111 DNNs are used to remove reverberation from speech recordings using a single
microphone. The system works with the STFT of the speech signals. Two different U-net architectures, as well as adversarial training with GANs are implemented. The dereverberation performance of the proposed DL architectures outperform competing methods in most cases.

Much like in acoustics, seismic exploration research has traditionally focused on advanced signal processing algorithms, with only occasional applications of pattern recognition techniques. ML and especially DL methods have recently seen significant increases in seismic exploration applications. One area of the field has obtained many benefits from DL models is the interpretation of geological structure elements. Classification and interpretation of these structures, such as salt domes, channels, faults and folds, from seismic images faces several challenges, including handling extremely large volumes of 3D seismic data and sparse and uncertainty manual image annotations from geologists. Many benefits are achieved by automating these procedures. Several recently developed ML techniques construct attributes adapted to specific data via ML algorithms instead of hand-engineering them.

DL has been applied to the seismic interpretation of faults, channels, salt domes, as well as seismic facies classification using 3D CNNs and GANs. In Ref. 121, a 3D U-net was applied to detect or segment faults from a 3D seismic images. In Ref. 124, a semi-supervised facies classifier based on 3D CNNs with GANs was developed to handle large volumes of data from new exploration fields which might have few labels. There have also been interesting developments in seismic data post-processing, including automated seismic facies classification.

VI. SPEAKER LOCALIZATION IN REVERBERANT ENVIRONMENTS

Speech enhancement is a core problem in audio signal processing, with commercial applications in devices as diverse as mobile phones, hands-free systems, human-car communication, smart homes, or hearing aids. An essential component in the design of speech enhancement algorithms is acoustic source localization. Speaker localization is also directly applicable to many other audio related tasks, e.g., automated camera steering, teleconferencing systems, and robot audition.

Driven by its large number of applications, the localization problem has attracted significant research attention, resulting in a plethora of localization methods proposed during the last two decades. Nevertheless, robust localization in adverse conditions, namely in the presence of background noise and reverberation, still remains a major challenge.

A recent challenge on acoustic source localization and tracking (LOCATA), endorsed by the IEEE Audio and Acoustic Signal Processing technical committee, has established a database to encourage research teams to test their algorithms. The challenge dataset consists of acoustic recordings from real-life scenarios. With this data, the performance of source localization algorithms in real-life scenarios can be assessed.

There is a growing interest in supervised-learning for audio source localization using NNs. In the recent issue on “Acoustic Source Localization and Tracking in Dynamic Real-Life Scenes” in the IEEE Journal on Selected topics in Signal Processing, three papers used variants of NNs for source localization. We expect this trend to continue, with an emphasis on methods that do not require a large set of labeled data. Such labeled data is very difficult to obtain in the localization problem. For example, in Ref. 129, a weakly labeled ML paradigm is presented. The approach used few labeled samples with known positions along with larger set of unlabeled samples, for which only their relative physical ordering is known.

In this short survey, we explore two families of learning-based approaches. The first is an unsupervised method based on GMM classification. The second is a semi-supervised method based on manifold learning.

Despite the progress that has been made in the recent years in the manifold-learning approach for localization, some major challenges remain to be solved, e.g., robustness to changes in array constellation and the acoustic environment, and the multiple concurrent speakers case.

A. Localization and tracking based on the expectation-maximization procedure

In this section, we review an unsupervised learning methodology for speaker localization and tracking of an unknown number of concurrent speakers in noisy and reverberant enclosures, using a spatially distributed microphone array. We cast the localization problem as a classification problem in which the measurements (or features extracted thereof) can be associated with a grid of candidate positions, $\mathcal{P} = \{p_1, \ldots, p_M\}$, where $M = |\mathcal{P}|$ is the number of candidates. The actual number of speakers is always significantly lower than $M$.

The speech signals, together with an additive noise, are captured by an array of microphones ($N > 1$). The binaural case ($N = 2$) was presented in Ref. 130. We assume a simple sound propagation model with a dominant direct-path and potentially a spatially diffuse reverberation tail. The $n$th microphone signal in the STFT domain is given by

$$z_n(t, k) = \sum_{m=1}^{M} d_{m}(t, k) g_{m,n}(k) s_m(t, k) + v_n(t, k), \quad (56)$$

where $t = 0, \ldots, T - 1$ is the time index, $k = 0, \ldots, K - 1$ is the frequency index, $g_{m,n}(k)$ is the direct-path transfer function from the speaker at the $m$-th position to the $n$-th microphone,

$$g_{m,n}(k) = \frac{1}{\|p_m - p_n\|} \exp\left(-\frac{2\pi k \tau_{m,n}}{K T_s}\right), \quad (57)$$

where $T_s$ is the sampling period, and $\tau_{m,n} = \|p_m - p_n\|/c$ is the TDOA between candidate position $p_m$ and microphone position $p_n$ and $c$ the sound velocity. This TDOA can be
where \( z(t,k) \) is the speech signal uttered by a speaker at grid point \( m \) and \( \mathbf{v}(t,k) \) is either an ambient noise or a spatially diffused reverberation tail. The indicator signal \( d_m(t,k) \) indicates whether speaker \( m \) is active in the \((t,k)\)-th STFT bin,

\[
d_m(t,k) = \begin{cases} 
1, & \text{if speaker } m \text{ is active in STFT bin } (t,k), \\
0, & \text{otherwise}. 
\end{cases}
\]  

(58)

Note that, according to the sparsity assumption\(^{131}\) the vector \( \mathbf{d}(t,k) = \text{vec}_m \{ d_m(t,k) \} \in \{ e_1, \ldots, e_M \} \), where \( \text{vec}_m \{ \} \) is a concatenation of the elements along the \( m \)th index and is a “one-hot” vector (equals ‘1’ in its \( m \)th entry, and zero elsewhere). The \( N \) microphone signals are concatenated in a vector form

\[
\mathbf{z}(t,k) = \sum_{m=1}^{M} d_m(t,k) \mathbf{g}_m(k) s_m(t,k) + \mathbf{v}(t,k),
\]  

(59)

where \( \mathbf{z}(t,k), \mathbf{g}_m(k), \) and \( \mathbf{v}(t,k) \) are the respective concatenated vectors.

We will discuss several alternative feature vector selections from the raw data. Based on the W-disjoint orthogonality property of the speech signal,\(^{131,132}\) these features can be attributed a GMM (35), with each Gaussians associated with a candidate position in the enclosure on the predefined grid.

An alternative is to organize the microphones in dual-microphone nodes and to extract the pair-wise relative phase ratio (PRP)

\[
\phi_n(t,k) = \frac{z_n^1(t,k)}{z_n^2(t,k)},
\]  

(60)

with \( n \) the node index (number of microphones in this case is \( 2N \)), the superscript is the microphone-pair index (either 1 or 2) within the pair \( n \). Under the assumptions that (1) the inter-microphone distance is small compared with the distance of grid points from the node center, and 2) the reverberation level is low, the PRP of a signal impinging the microphones located at \( \mathbf{p}_n^1 \) and \( \mathbf{p}_n^2 \) from a grid point \( \mathbf{p}_m \) can be approximated by

\[
\tilde{\phi}_m^k(\mathbf{p}_m) \approx \exp \left( - \frac{2\pi k - ||\mathbf{p}_m - \mathbf{p}_n^1|| - ||\mathbf{p}_m - \mathbf{p}_n^2||}{c \cdot T_s} \right).
\]  

(61)

Since this approximation is often violated, we use \( \tilde{\phi}_m^k(\mathbf{p}_m) \) as the centroid of a Gaussian that describes the PRP. For multiple speakers in unknown positions we can use the W-disjoint orthogonality to express the distribution of the PRP as a GMM,

\[
f(\phi) = \prod_{t,k} \prod_{m=1}^{M} \pi_m \prod_{n} \mathcal{CN}(\phi_n(t,k); \tilde{\phi}_m^k(\mathbf{p}_m), \sigma^2).
\]  

(62)

We will also assume for simplicity that \( \sigma^2 \) is set in advance.

Using the GMM, the localization task can be formulated as a maximum likelihood parameter estimation problem. The number of active speakers in the scene and their position will be indirectly determined by examining the GMM weights, \( \pi_m, m = 1, \ldots, M \), and selecting their peak values. As explained above, the ML parameter estimation problem cannot be solved in closed-form. Instead, we will resort to the expectation-maximization (EM) procedure.\(^9\) The E-step results in the estimate of the indicator signal (here the hidden data),

\[
\hat{d}^{(r-1)}(t,k,m) \triangleq \mathbb{E} \left\{ d(t,k,m) | \phi(t,k); \hat{\pi}^{(r-1)} \right\}
\]

\[
= \frac{\hat{\pi}_m^{(r-1)} \prod_{n} \mathcal{CN}(\phi_n(t,k); \tilde{\phi}_m^k(\mathbf{p}_m), \sigma^2)}{\sum_{m=1}^{M} \hat{\pi}_m^{(r-1)} \prod_{n} \mathcal{CN}(\phi_n(t,k); \tilde{\phi}_m^k(\mathbf{p}_m), \sigma^2)}.
\]  

(63)

In the M-step the GMM weights are estimated,

\[
\hat{\pi}_m^{(r)} = \frac{\sum_{t,k} \hat{d}^{(r-1)}(t,k,m)}{T \cdot K}.
\]  

(64)

The procedure is repeated until a number of predefined iterations \( \ell = L \) is reached. We refer to this procedure as batch EM, as opposed to the recursive and distributed variants that will be later introduced. In Fig. 15 a comparison between the classical SRP-PHAT and the batch EM is depicted. It is evident that the EM algorithm (which maximizes the ML criterion) achieves much higher resolution.

In Ref. 133, a distributed version of this algorithm was presented, suitable for wireless acoustic sensor networks (WASN) with known microphone positions. WASNs are characterized by low computational resources in each node and by a limited connectivity between the nodes. A bi-directional tree-based distributed EM (DEM) algorithm that circumvents the inherent network limitations was proposed, by substituting the standard EM iterations by iterations across nodes. Furthermore, a recursive distributed EM (RDEM) variant, which is better suited for online applications, is proposed.

In Ref. 134, an improved, bio-inspired, acoustic front-end that enhances the direct-path, and consequently increasing the robustness of the proposed schemes to high reverberation, is presented. An alternative method for enhancing the direct-path is presented in Ref. 135, where the multi-path propagation model of the sound is taken into account, by the so-called convolutive transfer function (CTF) model.\(^{136}\)

In another variant of the classification paradigm, the GMM is substituted by a mixture of von Mises\(^{137}\) which is a suitable distribution for the periodic phase of the microphone signals.

Here we will elaborate on another alternative feature, namely the raw microphone signals (in the STFT domain). According to our measurement model [Eqs. (56), (59)] the raw data can also be described by a GMM.\(^{138–40}\)
where the covariance matrix of each Gaussian is given by
\[
\Phi_{\theta,m}(t,k) = g_m(k)g_m^H(t,k) + \Phi_{\gamma}(k).
\] (66)

Here, we assumed that the noise is stationary and its PSD known. In this case (frequency index \(k\) is omitted for brevity), the E-step simplifies to\(^{141}\)
\[
\hat{d}_m^{(t-1)}(t) = \frac{\hat{\pi}_m^{(t-1)} T_m(t)}{\sum_m \hat{\pi}_m^{(t-1)} T_m(t)},
\] (67)
with the likelihood ratio test (LRT),
\[
\text{LRT}_m(t) = \frac{1}{\text{SNR}_{\text{post}}^m(t)} \exp(\text{SNR}_{\text{post}}^m(t) - 1),
\] (68)
where \(\text{SNR}_{\text{post}}^m(t) = \hat{\delta}_{m,MVDR}(t) / \phi_{\theta,m}^2\) is the posterior SNR of a signal from the \(m\)th candidate position. \(\hat{\delta}_{m,MVDR}(t) \equiv w_m^H z(t)\) is an estimate of the speech using the minimum variance distortionless response beamforming (MVDR-BF), \(w_m = \Phi_{\gamma}^{-1} g_m / \Phi_{\gamma} g_m^H g_m\), which constitutes a sufficient statistic for estimating the speech PSD \(\phi_{\theta,m}(t)\) given the observations \(z(t)\), and \(\phi_{\gamma,m} \equiv 1 / \Phi_{\gamma}^{-1} g_m^H g_m\) is the PSD of the residual noise at the output of the MVDR-BF, directed towards the \(m\)th position candidate.

Two recursive EM (REM) variants can be found in literature, see Cappé and Moulines\(^{142}\) and Titterington\(^{143,144}\). The former is based on recursive calculation of the auxiliary function, and the latter utilizes a Newton-based recursion for the maximization, with the Hessian substituted by the Fisher information matrix (FIM). Recursive EM algorithms for source localization were analyzed and developed in Refs. 145 and 146. Titterington’s method was extended to deal with constrained maximization, encountered in the problem at hand in Ref. 147. Applying these procedures to both data models in Eqs. (62) and (65) results in Ref. 147,
\[
\hat{\pi}_m^R(t) = \hat{\pi}_m^R(t - 1) + \gamma_t (\hat{\pi}_m(t) - \hat{\pi}_m^R(t - 1)),
\] (69)
where \(\hat{\pi}_m(t) = \sum_k \hat{d}(t,k,m) / K\) is the instantaneous estimate of the indicator and \(\hat{\pi}_m^R(t)\) is the recursive estimator. The tracking capabilities of the algorithm in Ref. 141 in simulated data with low noise level and two speakers in reverberation time \(T_{60} \approx 300\) ms is depicted in Fig. 16. For this experiment we used an eight-microphone linear array, hence only DOA estimation capabilities were examined. For the DOA candidates we used a grid of possible azimuth angles between \(-90^\circ\) and \(90^\circ\), with a resolution of \(2^\circ\). The proposed algorithm provides speaker DOA probability distributions as a function of time, as depicted in Fig. 16, and not directly the DOA estimates. To estimate the actual trajectory of the speakers, the speaker locations at each time are found by the probability maxima. In another variant that uses the same features (68), the tracking problem is recast as an hidden Markov model (HMM) and the time-varying DOAs are inferred by the forward-backward algorithm\(^{148}\) rather than the recursive EM in (69).

**B. Speaker localization and tracking using manifold learning**

Until recently, a main paradigm in localization research was based on certain statistical and physical assumptions

![FIG. 15. (Color online) Two targets localization. 10 × 10 cm grid, 12 nodes, inter-microphone distance per node 50 cm, \(T_{60} = 300\) ms. (a) Steered response power phase transform (SRP-PHAT) beamformer (Ref. 106). (b) Batch expectation maximization (EM) (Refs. 133, 147).](image)

![FIG. 16. (Color online) Two speaker tracking using recursive expectation-maximization (Ref. 141). True DOAs are indicated with dashed lines.](image)
regarding the propagation of sound sources,\textsuperscript{149,150} and mainly focused on robust methods to extract the direct-path. However, valuable information on the source location can also be extracted from the reflection pattern in the enclosure.

The main claim here is that the intrinsic reflection patterns of the sound source on the room facets and the objects in the enclosure define a fingerprint, uniquely characterizing the source location, and that meaningful location information can be inferred from the data by harnessing the principles of manifold learning.\textsuperscript{151,152} Yet, the intrinsic degrees of freedom in the acoustic responses have a limited number. Hence, we can conclude that the variability of the acoustic responses in specific enclosures depends only on a small number of parameters. This calls upon manifold learning approaches to improve localization abilities. We first consider recordings from two microphones

\begin{align}
y_1(n) &= a_1(n) \ast s(n) + u_1(n), \\
y_2(n) &= a_2(n) \ast s(n) + u_2(n),
\end{align}

\tag{70}

with \(s(n)\) the source signal, \(a_i(n), i = \{1, 2\}\) the acoustic impulse responses (AIRs) relating the source and each of the microphones, and \(v_i(n)\) noise signals which are independent of the source. Define the acoustic transfer functions (ATFs) \(A_i(k)\) as the Fourier transform of the AIRs \(a_i(n)\), respectively. Then, the relative transfer function (RTF) is defined as\textsuperscript{153}

\[ H(k) = \frac{A_2(k)}{A_1(k)}. \]

\tag{71}

The RTF represents the acoustic path, encompassing all sound reflection paths. As such, it can be viewed as a generalization of the PRP centroid (61). A plethora of blind RTF estimation procedure exists.\textsuperscript{154} Finally, we define the RTF vector by concatenating several values of \(H(k)\) in the relevant frequency band (where the speech power is significant),

\[ \mathbf{h} = \begin{bmatrix} H(k_1) & H(k_2) & \cdots & H(k_D) \end{bmatrix}^T, \]

\tag{72}

with \(k_1\) and \(k_D\) are the lower and upper frequencies of the significant frequency band. Note that the RTF is independent of the source signal, hence can serve as an acoustic feature, as required in the following method.

Our goal is to find a representation of RTF vectors, as defined in Eq. (72). This representation should reflect the intrinsic degrees-of-freedom that control the variability of a set of RTF. To this end, we collect a set of \(N\) RTF vectors from the examined environment: \(\mathbf{h}_i, i = 1, 2, \ldots, N\). We then construct a graph that empirically represents the acoustic manifold. The RTFs are used as the graph nodes (not to be confused with the microphone constellation as defined above), and the edges are defined using an RBF kernel

\[ k(\mathbf{h}_i, \mathbf{h}_j) = \exp \left\{-\|\mathbf{h}_i - \mathbf{h}_j\|_2^2 / \epsilon \right\} \]

to connect two RTF vectors, \(\mathbf{h}_i, \mathbf{h}_j\). Define the \(N \times N\) kernel matrix \(K_{ij} = k(\mathbf{h}_i, \mathbf{h}_j)\). Let \(\mathbf{D}\) be a diagonal matrix whose diagonal elements are the sums of rows of \(\mathbf{K}\). Define the row stochastic \(\mathbf{P} = \mathbf{D}^{-1} \mathbf{K}\), a non-symmetric transition matrix with elements defining a Markov process on the graph \(\mathbf{P} = \mathbf{P}(\mathbf{h}_i, \mathbf{h}_j)\), which is a discretization of a diffusion process on the manifold.\textsuperscript{155} Since \(\mathbf{P}\) is non-symmetric but similar to a symmetric matrix, we can define the left- and right-eigenvectors of the matrix with shared non-negative eigenvalues: \(\mathbf{P}\phi^{(i)} = \lambda_i \phi^{(i)}\) and \(\psi^{(i)}\mathbf{P} = \lambda_i \psi^{(i)}\). In these definitions, \(\phi^{(i)}\) is the left (row) eigenvector, \(\psi^{(i)}\) is the right (column) eigenvector, \(\psi^{(i)}\) is the left (row) eigenvector and \(\lambda_i\) is the corresponding eigenvalue.

This decomposition induces a nonlinear mapping of the RTF into a low-dimensional Euclidean space,

\[ \Phi_d : \mathbf{h}_i \mapsto [\lambda_1 \phi^{(i)}_1, \ldots, \lambda_d \phi^{(i)}_d]^T. \]

\tag{73}

The nonlinear operator, defined in Eq. (73), is referred to as the diffusion mapping.\textsuperscript{155} It maps the \(D\)-dimensional RTF vector \(\mathbf{h}_i\) in the original space to a lower \(d\)-dimensional Euclidian space, constructed as the \(i\)th component of the most significant \(d\) eigenvectors (multiplied by the corresponding eigenvalue). Note, that the first eigenvector \(\phi_0\) is an all-ones trivial vector since the rows of \(\mathbf{P}\) sum to 1.

The diffusion distance reflects the flow between two RTFs on the manifold, which is related to the geodesic distance on the manifold, namely, two RTFs are close to each other if their associated nodes of the graph are well-connected. It can be proven that the diffusion distance

\[ D_{\text{Diff}}(\mathbf{h}_i, \mathbf{h}_j) = \sum_{r=1}^{N} \left( p(\mathbf{h}_i, \mathbf{h}_r) - p(\mathbf{h}_j, \mathbf{h}_r) \right)^2 / \psi_0^{(r)} \]

\tag{74}

is equal to the Euclidean distance in the diffusion maps space when using all \(N\) eigenvectors, and that it can be well-approximated by using only the first few \(d\) eigenvectors,

\[ D_{\text{Diff}}(\mathbf{h}_i, \mathbf{h}_j) \cong ||\Phi_d(\mathbf{h}_i) - \Phi_d(\mathbf{h}_j)||. \]

\tag{75}

This constitutes the basis of the embedding from the high-dimensional RTFs with their intricate geodesic distance, to the simple Euclidean distance in the low-dimension space. Thus, distances and ordering in the low-dimensional space can be easily measured. As we will next demonstrate, the low-dimensional representation inferred from this mapping has a one-to-one correspondence with physical quantities, here the location of the source.

To demonstrate the ability of this nonlinear diffusion mapping to capture the controlling parameters of the acoustic manifold, the following scenario was simulated.\textsuperscript{156} Two microphones were positioned at \([3, 3, 1]\) m and \([3.2, 3, 1]\) m in \(6 \times 6.2 \times 3\) m room with reverberation time of \(T_{60} = 500\) ms and \(\text{SNR} = 20\) dB. The source position was confined to a circle around the microphone pair with \(2\) m radius. It is evident from Fig. 17 that the dominant eigenvector indeed corresponds to the angle of arrival of the source signal in the range \(10^\circ - 60^\circ\). This forms the basis for a semi-supervised localization method.\textsuperscript{157}

It is acknowledged that collecting labeled data in reverberant environment is a cumbersome task; however, measuring RTFs in the enclosure is relatively easy. It is therefore proposed to collect a large number of RTFs in the room where localization is required. These unlabeled RTFs can be collected whenever a speaker is active in the environment.
E.g., the RMSE of this method in reverberation level may improve significantly the localization accuracy, with a large number of training points. Using this semi-supervised regularizer, the optimization has two regularization terms, a Tikhonov regularizer with \( k \) and a manifold-regularization \( k \) with \( \gamma_L \) and \( \gamma_M \). Smoothness along the inferred manifold.

These RTFs will be used to infer the structure of the manifold. A small number of labelled RTFs, i.e., RTFs with an associated accurate position label, will also be collected. These points will be used to anchor the inferred manifold to the physical world, thereby facilitating the position estimation of an untrained accurate position label, will also be collected. These RTFs will be used to infer the structure of the manifold. A small number of labelled RTFs, i.e., RTFs with an associated accurate position label, will also be collected. These points will be used to anchor the inferred manifold to the physical world, thereby facilitating the position estimation of an unknown RTF at test time. In this method, a mapping from an RTF to a position \( p = g(h) \) (here we define a mapping from the RTF to each coordinate, hence a vector to scalar mapping) is inferred by the following optimization problem:

\[
\hat{g} = \arg\min_{g \in \mathcal{H}^L} \frac{1}{n_L} \sum_{i=1}^{n_L} (\hat{p}_i - g(h_i))^2 + \gamma_k \|g\|^2_{\mathcal{H}_L} + \gamma_M \|g\|^2_{\mathcal{M}},
\]

with \( n_L \) labeled pairs \( \{h_i, \hat{p}_i\} \) and \( n_U \gg n_L \) unlabeled RTFs. The optimization has two regularization terms, a Tikhonov regularizer \( \|g\|^2_{\mathcal{H}_L} \) that controls the smoothness of the mapping and a manifold-regularization \( \|g\|^2_{\mathcal{M}} \) that controls the smoothness along the inferred manifold.

The minimizer of the regularized optimization problem can be found by optimization in a reproducing kernel Hilbert space (RKHS) \( \mathcal{H}^L \)

\[
g(h) = \sum_{i=1}^{n_D} a_i k(h_i, h),
\]

where \( k : \mathcal{M} \times \mathcal{M} \to \mathbb{R} \) is the reproducing kernel of \( \mathcal{H}_L \), with \( k(h_i, h_j) \) as defined above, and \( n_D = n_L + n_U \) the total number of training points. Using this semi-supervised method may improve significantly the localization accuracy, e.g., the RMSE of this method in reverberation level of \( T_{60} = 600 \text{ ms} \) and \( \text{SNR} = 5 \text{ dB} \) is \( \approx 3^\circ \), while the classical generalized cross-correlation (GCC) method \( \text{GCC} \) achieves an RMSE of \( 18^\circ \) at the same acoustic conditions.

An extension to the multiple microphone case is presented in Ref. 160. In this case, it is necessary to fuse the viewpoints of all nodes into one mapping from a set of RTFs to a single position estimate \( g : \mathbb{U}_{M=1}^{M} \mathcal{M}_m \to \mathbb{R} \).

Using a Bayesian perspective of the RKHS optimization, \( g = g(h) \) is modelled as a Gaussian process, it is easy to extend the single-node problem to a multiple-node problem, by using an average Gaussian process \( g = (1/M)(g^1 + g^2 + \cdots + g^M) \sim \mathcal{GP}(0, \bar{k}) \). The covariance of this Gaussian process can be calculated from the training data

\[
\text{cov}(g(h_i), g(h_j)) = \bar{k}(h_i, h_j) = \frac{1}{M^2} \sum_{q=1}^{M} \sum_{r=1}^{M} k_q(h_i^q, h_j^q)k_w(h_i^w, h_j^w).
\]

See Fig. 18 for a schematic depiction of the multi-manifold fusion paradigm.

The multi-manifold localization scheme \( \text{ML} \) was evaluated using real signals recorded at the Bar-Ilan acoustic lab, see Fig. 19. A 6 × 6 × 2 m room is covered by two-sided panels allowing to control the reverberation level. In the reported experiment reverberation time was set to \( T_{60} = 620 \text{ ms} \). The source position was confined to a 2.8 × 2.1 m area. Three microphone pairs with inter-distance of 0.2 m were used. For the algorithm training 20 labelled samples with 0.7 m resolution and 50 unlabeled samples were used. The algorithm was tested with two noise types: air-conditioner noise and babble noise. As an example, for SNR = 15 dB the SRP-PHAT \( \text{SRP-PHAT} \) achieves an RMSE of 58 cm (averaged over 25 samples in the designated area) while the multi-manifold algorithm achieves 47 cm. Finally, in dynamic scenarios, recursive versions using the Kalman filter and its extensions, with the

\[
\text{FIG. 17. (Color online) Diffusion mapping. A set of } N \text{ RTF vectors } h_i, \text{ } i = 1, \ldots, N \text{ is considered. Using diffusion mapping each RTF } D\text{-dimensional vector } h_i \text{ is mapped into the } i\text{th component the first non-trivial eigenvector } \phi_1(i) \text{ (the eigenvalue is shared by all components and hence ignored here). By mapping the entire set we get } N \text{ embedded values. These values constitute the } y \text{ axis of the graph. In the } x \text{ axis we draw the known angle of arrival of associated with the RTF vector } h_i \text{. A clear correspondence is demonstrated, proving that the diffusion mapping indeed blindly extracts the intrinsic degree-of-freedom of the RTF set, and hence can be utilized for data-driven localization (Ref. 156).}
\end{equation*}

\[
\text{FIG. 18. (Color online) A multi-view perspective of the acoustic scene with each manifold defining a mapping from the RTFs to position estimates (Ref. 160).}
\end{equation*}
covariance matrices of the propagation and measurement processes inferred from the manifold structure. Simulation results for $T_{60} = 300$ ms and a sinusoidal movement, 5 s long and approximate velocity of 1 m/s is depicted in Fig. 20. Very good tracking capabilities are demonstrated. In the simulations, the room size was set to $5.2 \times 6.2 \times 3$ m, the number of microphone pairs was $M = 4$ with 0.2 m inter-distance between microphone pairs. The training comprised 36 samples with 0.4 m resolution.

VII. SOURCE LOCALIZATION IN OCEAN ACOUSTICS

Underwater source localization methodologies in ocean acoustics have conventionally relied on physics-based propagation models of the known environment. Unlike conventional methods, ML methods may be considered “model-free” as they do not rely on physics-based forward-modeling to predict source location. ML instead infers patterns from acoustic data which allows for a purely data-driven approach to source localization. However, in lieu of sufficient data, model simulations can also be incorporated with experimental data for training, in which case ML may not be fully model-free. The application of ML to underwater source localization is a relatively new research area with the potential to leverage recent advancements in computing for accurate, real-time prediction.

Matched-field processing (MFP) has been applied to ocean source localization for decades with reasonable success. Recent MFP modifications incorporate compressive sensing since there are only a few source locations. However, MFP is prone to model mismatch. Model mismatch has been alleviated by data-replica MFP where closely matched data is available.

The earliest ML-based approach to underwater source localization was implemented by a NN trained on modeled data to learn a forward model consisting of weighted transformations. Early ML methods were also applied to seabed inversion with limited success and to seafloor classification using both supervised and unsupervised learning. The models were linear in the weight space due to lack of widespread knowledge about efficient nonlinear inference algorithms. Also at this time, NN performance was hindered by computational limitations.

Due to these early computational limitations, alternative methods replaced NNs in the state-of-the-art. These methods included Bayesian inference with physical forward models and model-free localization methods, including the waveguide and array invariant methods, which are effective in well-studied waveguide environments. The field of ML once-again gained momentum with the growth of computational efficiency, the advent of open-source software and, notably, improved learning algorithms for deep, nonlinear inference.

More recent developments in ML for underwater acoustics concerned target classification. Studies of ocean source localization using ML appeared soon thereafter and include applications to experimental data for broadband ship localization, target characterization, and post-processing of time difference of arrival estimates. Recently, studies have examined underwater source localization with CNNs and DL, taking advantage of 2D data structure, shared weighting, and huge model-generated datasets. Other recent applications of ML in ocean acoustics include geoacoustic inversion.

In Niu et al., the sample covariance matrix was used in a feed-forward neural network (NN) classifier to predict source range. The NN performed well on simulated data.
and localized cargo ships from Noise09 and Santa Barbara Channel experiments\textsuperscript{186} (Fig. 21). While NNs achieved high accuracy, MFP was challenged by solution ambiguity (Fig. 22). Huang \textit{et al.}\textsuperscript{191} used the eigenvalues of the sample covariance matrix in a deep time-delay neural network (TDNN) regression, which they trained on simulated data from many environments. For a shallow, sloping ocean environment, the TDNN was trained at multiple ocean depths to avoid model mismatch. It tracked the ships location accurately, whereas MFP always overestimated the ship range (Fig. 23). Recently, Niu \textit{et al.}\textsuperscript{104} input the acoustic amplitude on a single hydrophone into a deep residual CNN (ResNet)\textsuperscript{109} to predict source range and depth (Fig. 14). The deep model was trained with tens of millions of samples from numerous environmental configurations. The deep Res-Net had lower range prediction error and competitive depth error.
compared to the seismo-acoustic genetic algorithm (SAGA) inversion method.

Future source localization research will benefit from combining the developments in propagation modeling, parallel and cloud computation tools, and big data storage for long-term or large-scale acoustic recordings. Powerful new ML methods utilizing these techniques will achieve real-time, accurate ocean source localization.

VIII. BIOACOUSTICS

Bioacoustics is the study of sound production and perception, including the role of sound in communication and the effects of natural and anthropogenic sounds on living organisms. ML has the potential to address many questions in this field. In some cases, ML is directly applied to answer specific questions: When are animals present and vocalizing?\textsuperscript{194,195} Which animal is vocalizing?\textsuperscript{196,197} What species produced a vocalization?\textsuperscript{198} What call or song was produced and how do these sounds relate to one another?\textsuperscript{199,200} Among these questions, species detection and identification is a primary driver of many bioacoustics studies due to the reasonably direct implications for conservation and mitigation.

Information mined from these direct acoustic measurements and can be used to answer specific biological, ecological, and management questions. Examples of this include: What is the density of animals in an area?\textsuperscript{201} and how is the density changing over time?\textsuperscript{202} How do lunar patterns affect foraging behavior?\textsuperscript{203} Many of the issues presented throughout this section are also relevant to soundscape ecology, which is the study of all sounds within an environment.\textsuperscript{204}

Although many recent works are starting to use learned features, such as those produced by autoencoder NNs or other dimensionality reduction techniques mentioned in the Introduction, much of the bioacoustics literature uses hand-selected features. These are either applied across the spectrum, such as cepstral representations of spectral\textsuperscript{205} which capture the shape of the spectral envelope of a short segment of the signal\textsuperscript{206,207} in a low number of dimensions (Fig. 24) or engineered towards specific calls. Many of the features designed for specific calls tend to concentrate on statistics of acoustic parameters such as mean or center frequency, bandwidth, time-bandwidth products, number of inflections in tonal calls, etc. It is fairly common to use psychoacoustic scales such as the Melodic (Mel) scale which recognizes that humans (and most other animals) have an acoustic fovea, a frequency range where they can most accurately perceive frequency differences. However, it is important to remember that this varies between species and the standard Mel scale is weighted towards humans whose hearing characteristics may vary from the target species.

Learned features attempt to determine the feature set from the data and include any type of manifold learner such as principal component analysis or autoencoders. In most
cases, the feature learners are given standard features such as cepstral coefficients or statistics of a call (in which case they are simply learning a manifold of the features) or attempt to learn from relatively unprocessed data such as time-frequency representations. Stowell and Plumbley\textsuperscript{208} provide an example of using a spherical K-means learner to construct features from Mel-filtered spectra. Spherical K-means normalizes the input vectors and uses a cosine distance as its distortion metric. Other feature learners that have been used in bioacoustics include sparse autoencoders,\textsuperscript{209} and CNNs that learn weights associated with features of interest.\textsuperscript{210}

There are many examples of template-based methods that can work well when calls are highly stereotyped. The simplest type of template method is the time-domain matched filter, but in bioacoustics matched filters are typically implemented in time-frequency space.\textsuperscript{194} More complex matched filters permit non-linear compression or elongation of the filter with dynamic time warping,\textsuperscript{211} which has been used for both delphinid whistles\textsuperscript{212} and bird calls.\textsuperscript{205} However, even these so-called “stereotyped” calls have, in many species, been shown to drift over time. It has been shown that the tonal frequency of blue whale calls have decreased.\textsuperscript{213} These types of changes can cause matched-template methods to require recalibration.

Supervised learning is the primary learning paradigm that has been used in ML bioacoustics research and can be traced back to the use of linear discriminant analysis. An early example of this was the work of Steiner\textsuperscript{198} that examined classifying delphinid whistles by species. GMMs have been used to capture statistical variation of spectral parameters of the calls of toothed whales\textsuperscript{61} and sequence information has been exploited with hidden Markov models for classifying bird song by species.\textsuperscript{207,214} Multi-level perceptron NNs also have a rich history of being applied in bioacoustics, with varied uses such as bat species identification, bowhead whale (\textit{Balaena mysticetus}) call detection, and recognizing killer whale (\textit{Orcinus orca}) dialects.\textsuperscript{215-217} Decision tree methods have been used, with early approaches using classification and regression trees for species identification.\textsuperscript{218} SVM-based methods have also had considerable success, examples include classifying the calls of birds and anurans to species.\textsuperscript{25,66}

Ensemble learning is a well-known method of combining classifiers to improve results by reducing the variance of the classification decision through the use of multiple classifiers that learn from different data, with well-known examples such as random forest\textsuperscript{219} and adaptive boosting.\textsuperscript{220} These techniques have been leveraged by the bioacoustics community, such as the work by Gradišek \textit{et al.}\textsuperscript{221} that used random forests to distinguish bumble bee species based on characteristics of their buzz.

One of the most recent trends in bioacoustic pattern recognizers is the use of DNNs that have reduced classification error rates in many fields\textsuperscript{10} and have reduced many of the issues of overfitting NNs seen in earlier artificial NNs through a variety of methods such as increased training data, architectural changes, and improved regularization techniques. An early use of this in bioacoustics can be seen in the work of Halkias \textit{et al.}\textsuperscript{209} that demonstrated the ability of deep Boltzmann machines to distinguish mysticete species. Deep CNNs and RNNs have been used for bat species identification,\textsuperscript{222} whale species identification,\textsuperscript{223} detecting and characterizing sperm whale echolocation clicks,\textsuperscript{197} and have become one of the dominant types of recognizers for bird species identification since the successful introduction of CNNs in the LifeCLEF bird identification task.\textsuperscript{224}

Unsupervised ML has not been used as extensively in bioacoustics, but has several noteworthy applications and large potential. Much of the work has been to cluster calls into distinct types, with the goal of using objective methods that are repeatable and do not suffer from perceptual bias. Examples of this include the K-means clustering,\textsuperscript{225,226} adaptive resonance theory clustering (Deecke and Janik\textsuperscript{215}), self-organizing maps,\textsuperscript{227} and clustering graph nodes based on modularity.\textsuperscript{228} Clustering sounds to species is also of interest and has been used to investigate toothed whale echolocation clicks in data deficient areas where not all species’ sounds have been well described\textsuperscript{229} using Biemann’s graph clustering algorithm\textsuperscript{230} that shares similarities with bottom up clustering approaches.

There are several repositories for bioacoustic data. The Macaulay Library at the Cornell Lab of Ornithology\textsuperscript{231} maintains an extensive database of acoustic media with a combination of curated and citizen-scientist recordings. Portions of the Xeno-Canto collection\textsuperscript{232} of bird sounds has been used extensively as a competition dataset in the CLEF series of conferences. The marine mammal bioacoustics community maintains the Moby Sound database of marine mammal sounds,\textsuperscript{233} which includes many of the datasets used in the Detection, Classification, Localization, and Density Estimation for marine mammals series of workshops. In addition, there are government databases such as the British Library’s sounds library\textsuperscript{234} which includes animal calls and soundscape recordings. Many organizations are trying to come to terms with the large amounts of data generated by passive acoustic recordings and some governments are conducting trials of long-term repositories for passive acoustic data such as the United States’ National Center for Environmental Information’s data archiving pilot program.\textsuperscript{235}

In Fig. 25 we illustrate the effect of recording equipment and sampling site location mismatch on cross validation results in marine mammal call classification. For some problems, changes across equipment or environments can cause severe degradation of performance. When these types of issues are not considered, performance in the field can vary significantly from what was expected based on laboratory experiments. Each case (acoustic encounter, preamplifier, preamplifier group, and site) specifies a grouping criterion for training/test folds. The acoustic encounter case are sets of calls from a group of animals when they are within detection range of the data logger. Calls from each encounter are entirely in training or test data. The preamplifier case adds further restrictions that encounters recorded on the same preamplifier are never split across the training and testing. The preamplifier group is case stricter yet; clicks from preamplifiers with similar characteristics cannot be split. The final group indicates that acoustic encounters from the same recording site cannot be split.
Finally, an ongoing challenge for the use of ML in bio-acoustics is managing detection data generated from long-term datasets. Some recent efforts are beginning to organize and store data products resulting from passive acoustic monitoring. While peripheral to the performance of ML algorithms, the ability to store the scores and decisions of ML algorithms along with descriptions of the algorithms and the parameters used is critical to comparing results and analyzing long-term trends.

IX. REVERBERATION AND ENVIRONMENTAL SOUNDS IN EVERYDAY SCENES

Humans encounter complex acoustic scenes in their daily life. Sounds are created by a wide range of sources (e.g., speech, music, impacts, scrapes, fluids, animals, machinery), each with its own structure and each highly variable in its own right. Moreover, the sound from these sources reverberate in the environment, which profoundly distorts the original source waveform. Thus the signal that reaches a listener usually contains a mixture of highly variable unknown sources, each distorted by the environment in an unknown fashion.

This variability of sounds in everyday scenes poses a great challenge for acoustic classification and inference. Classification algorithms must be sensitive to inter-class variation, robust to intra-class variation, and robust to reverberation—all of which are context dependent. Robust identification of sounds in natural scenes often requires both large training datasets, to capture the requisite acoustic variability, and domain specific knowledge about which acoustic features are diagnostic for specific tasks.

Overcoming these challenges will enable a range of novel technologies. These technologies include, for example, hearing aids which can extract speech from background noise and reverberation, or self-driving cars which can locate a fire-truck siren amidst a noisy street. Some applications which have already been investigated include: inspection of tile properties from impact sounds; classification of aircraft from takeoff sounds; and cough sound recognition in pig farms. More examples are given in Table 2 of Sharan and Moir. These are all tasks which must deal with the complexities of natural acoustic scenes. Because environmental sounds are so variable and occur in so many different contexts—the very fact which makes them difficult to model and to parse—any ML system that can overcome these challenges will likely yield a broad set of technological innovations. As such, the analysis and understanding of sound scenes and events is an active field of research.

There is another reason that algorithms which parse natural acoustic scenes are of special interest. By definition, such algorithms attempt the same challenge that biological hearing systems have evolved to solve—organisms, as well as engineers, desire to make sense of sound and thereby infer the state of the world. This convergence of goals means that engineers can take inspiration from auditory perception research. It also raises the possibility that ML algorithms may help us understand the mechanisms of auditory perception in both humans and animals, which remain the most successful systems in existence for acoustical inference in natural scenes.

In the following, we will consider two key challenges of applying ML algorithms to acoustic inference in natural scenes: (1) robustness to reverberation and (2) classification of a large range of diverse environmental sounds.

A. Reverberation

Acoustic reverberation is ubiquitous in natural scenes, and profoundly distorts sounds as they propagate from a source to a listener (Fig. 26). Thus any recorded sound is a product of both the source and environment. This presents a challenge to source recognition algorithms, as a classifier trained in one environment, may not work when presented with sounds from a different space, or even sounds presented from different locations within the same space. However,
reverberation also provides a source of information about the environment, and the source-listener distance. Humans can robustly identify sources, source locations, and properties of the environment from reverberant sounds. This suggests that the human auditory system can, from a single sound, separately infer multiple causal factors. The process by which this is done is poorly understood, and has yet to be replicated via algorithms.

The effect of reverberation can be described by filtering with the environment impulse response (IR),

\[
    r_j(t) = s(t) * h_j(t),
\]

where \( r(t) \) is the reverberant sound, \( s(t) \) the source signal, and \( h(t) \) the impulse response; the subscript \( j \) indexes across microphones in a multi-sensor array. An algorithm that seeks to identify the source (or IR), must either be robust to variations introduced by natural IRs (or sources), or it must be able to separate the signal into its constituents [i.e. \( s(t) \) and \( h(t) \)]. The challenge is that, in general, both \( s(t) \) and \( h(t) \) are unknown and such a separation is an ill-posed problem.

Presumably, to make sense of reverberant sounds, an algorithm must leverage knowledge about the acoustical structure of sources, IRs, or both. Natural scenes, despite highly diverse environments, display statistical regularities in their IRs, such as consistent frequency-dependent variation in decay rates (Fig. 26). This regularity partially enables human comprehension of reverberant sounds. If such regularities exist, ML algorithms can in principle learn them, if they receive appropriate training.

One way to address the variability introduced by reverberation is to incorporate reverberant sounds in the training dataset. This has been used to improve performance of a deep neural networks (DNNs) trained on speech recognition. Though effective in principle, this may require exceptionally large datasets to generalize to a wide range of environments.

A number of datasets with labelled sound sources in a range of reverberant environments have been prepared (REVERB challenge, ASpIRE challenge). Some datasets have focused instead on estimation of room acoustic parameters (ACE challenge). The proceedings of these challenges provide a thorough overview of state-of-the-art systems.

Given that the physical process underlying reverberation is well understood and can be simulated, the statistics of reverberation can also be assessed by simulating a large number of rooms. This has been used to train DNNs to reconstruct a spectrogram of dry (i.e., anechoic) speech from a spectrogram of reverberant speech.

Another approach to addressing reverberation is to derive algorithms which model the effects of reverberation on sound signals. For example, reverberant IRs smooth signals in the time domain, decreasing the signal kurtosis (which is largest for signals containing sparse high-amplitude peaks). Assuming the source signal is sparse, a dereverberation filter can be learned which maximizes the kurtosis of the output signal and returns an estimate of the source. More recent speech dereverberation methods, also employing machine learning methodologies, can be found in Refs. 111 and 250.

Another feature used is the spatial covariance of a microphone array. The direct-arriving (i.e., non-reverberant) sound is strongly correlated across two spatially separated microphones, as the signal detected at each channel is the same signal with different time delays. The reverberation, which consists of a summation of many signals incident from different directions is much less correlated across channels. This can be exploited to yield a dereverberation algorithm, and to estimate signal direction-of-arrival. There are also spectral-subtraction based methods to dereverberation, which for example estimate and subtract the late reverberant speech component. For a comprehensive review of speech dereverberation methods, please see Ref. 268.
In addition to estimating the source signal, it is often desirable to infer properties of the IR from the reverberant signal, and thereby infer characteristics of the environment.\(^{269}\) The most common such property to be inferred is the reverberation time (RT), which is the time taken for reverberant energy to decay some amount. RT can be estimated from histograms of decay rates measured from short windows of the signal.\(^{270}\)

The techniques described above have all shown some success in estimating sources or environments from reverberant audio. However, in most cases either the sound sources or the IRs were drawn from a constrained set (i.e. only speech, or a small number of rooms). It remains to be seen how well these approaches will generalize to the relative cacophony of everyday scenes.

### B. Environmental sounds

There are many challenges to identifying sources in natural scenes. First, there is the tremendous range of different sound sources. Natural scenes are filled with speech, music, animal calls, traffic sounds, machinery, fluid sounds, electronic devices, and a range of clattering, clanking, scraping and squeaking of everyday objects colliding. Second, there is tremendous variability within each class of sound. The sound of a plate dropped on a floor varies dramatically with the plate, the floor, the height of the drop, and the angle of impact. Third, natural scenes often contain many simultaneous sound sources which overlap and interfere. To recognize acoustic scenes, or the sources therein, an algorithm must simultaneously be sensitive to the differences between different sources and robust to the variation within each source.

The most obvious solution to overcoming the complexity of natural scenes is to train classifiers on large and varied sets of labelled recordings. To this end, a number of public datasets and have been introduced for both source recognition in natural scenes (DCASE challenges),\(^{103}\) ESC;\(^{271}\) TUT;\(^{72}\) Audio set;\(^{273}\) UrbanSound;\(^{274}\) and scene classification (DCASE; TUT). Thorough overviews are given for state-of-the-art algorithms in proceedings of these challenges, in Virtanen et al.,\(^{283}\) Sharan and Moir,\(^{245}\) for sound recognition, and Barchiesi et al.\(^{275}\) for scene recognition.

Recently, massive troves of online videos have proven a useful source of sounds for training and testing. One approach is to use meta-data tags in such videos as “weak labels.”\(^{276}\) Even though the labels are noisy and are not time-synced to the actual noise event—which may be sparse throughout the video—this can be mitigated by the sheer size of the training corpus, as millions of such videos can be obtained and used for training and testing.\(^{277}\)

Another approach to audiovisual training is to use state-of-the-art image processing algorithms to provide object and scene labels to each frame of the video. These can then be used as labels for sections of audio allowing conventional training of a classifier to recognize sound events from the audio waveform.\(^{278}\) Similarly, a network can be trained to map image statistics to audio statistics and thereby generate a plausible sound for a given image, or image sub-patch.\(^{279}\)

The synchronicity between object motion (rendered in pixels) and audio events can be leveraged to extract individual audio sources from video. Classifiers which receive inputs from both audio and video channels can be trained to differentiate videos with veridical audio, from videos with the wrong audio or temporally misaligned audio. Such algorithms learn “audiovisual features” and can then infer audio structure from pixel patterns alone. This enables audio source separation with video of multiple musicians or speakers,\(^ {280}\) or identification of where in an image a source is emanating.\(^ {281,282}\)

Whether trained by video features or by traditional labels, a sound source classifier must learn a set of acoustic features diagnostic of relevant sources. In principle, the features can be learned directly on the audio waveform. Some algorithms do this,\(^ {283}\) but in practice, most state-of-the-art algorithms use pre-processing to map a sound to a lower-dimensional representation from which features are learned. Classifiers are frequently trained upon short-time-Fourier transform (STFT) domains, and many variations thereupon with non-linear frequency spacings and spacings (mel-spaced, Gammatone, ERB, etc.). These decompositions (sometimes termed cochleagrams if the frequency spacing is designed to mimic the sensitivity of the cochlea within the ear) all favor finer spectral resolution at lower frequencies than higher frequencies, which both mirrors the sensitivity of biological audition and may be optimal for recognition of natural sounds.\(^ {284}\) Beyond the Spectro-temporal domain, algorithms have been presented which learn features upon a wide range of transformations of acoustical data (summarized by Sharan and Moir,\(^ {242}\) Li et al.,\(^ {285}\) and Waldekar and Saha\(^ {266}\)).

Sparse decomposition provides a framework to optimally decompose a waveform into a set of features from which the original sound can be approximately reconstructed. This has been put to use to optimize source recognition algorithms,\(^ {287}\) and, particularly in the form of non-negative matrix factorization (NMF), provides a learned set of features for sound recognition,\(^ {285}\) scene recognition,\(^ {289}\) source separation,\(^ {290}\) or denoising.\(^ {291}\)

Another approach to choosing acoustic features for classification, is to consider the generative processes by which environmental sounds are created. In many cases, such as impacts of rigid-body objects, the physical processes by which sound is created are well characterized and can be simulated from physical models.\(^ {292}\) Although full physical simulations are impractically slow for inference by a generative model, such models allow impact audio to be simulated rather than recorded.\(^ {293,294}\) (Fig. 27). This allows the creation of arbitrarily large datasets over which classification algorithms can be trained. The 20K audio-visual dataset contains orders of magnitude more labelled impact sounds (with associated videos) than earlier datasets.

Such physical synthesis models allow the training of classifiers which may move beyond recognizing broad sound classes and be able to judge fine-grained physical features such as material, shape or size of colliding objects. Humans can readily make such distinctions,\(^ {295,296}\) though how they do so is not known. In principle, detailed and flexible judgments
can be made via a generative model which explicitly encodes the relevant causal factors (i.e., the physical parameters we hope to infer, such as material, shape, size, mass, etc.). Such generative models have been used to infer objects and surfaces from images, vocal tract motion from speech, simple sounds from simulated scenes, and the motion of objects from the impact sounds made as they bounced and scraped across surfaces. However, as high-resolution physical sound synthesis is computationally expensive and slow, it is not yet clear how to apply such approaches to more realistic environmental scenes.

Given that the structure of natural sounds are determined by the physical properties of moving objects, audio classification can be aided by video information. Video provides, in addition to class labels as described above, information about the materials present in a scene, and the manner in which objects are moving. Owens et al. recorded a large set of videos of a drum stick striking objects in everyday scenes. The sounds produced by collision were projected into a low-dimensional feature space where they served as “labels” for the video dataset. A neural network was then trained to associate video frames with sound features, and could subsequently synthesize plausible sounding impacts for silent video of colliding objects.

C. Towards human-level interpretation of environmental sounds and scenes

As we have described above, recent developments in ML have enabled significant progress in algorithms that can recognize sounds from everyday scenes. These have already enabled novel technologies and will no doubt continue to do so. However, current state-of-the-art systems still do not match up to human perception in many inference tasks.

Consider, for example, the sound of an object (e.g., a coin, a pencil, a wine glass, etc.) dropped on a hard surface. From this sound alone, humans can identify the source, make guesses about how far and how fast it moved, estimate the distance and location of both the initial impact and the location of settling, distinguish objects of different material or size, and judge the nature of the scene from reverberation. In contrast, current state-of-the-art systems are considered successful if they can distinguish the sound of a basketball bouncing from a door slammed shut or the bark of a dog. They identify but do not interpret the sound the way that humans do. Interpreting natural sounds at this level of detail remains an unsolved engineering problem, and it is not known how humans do this intuitively. It is possible that developments in ML hearing of natural scenes and studies of biological hearing will proceed together, each informing and inspiring the other, to yet make a machine that “hears the world” like a human to parse and interpret the rich environmental sounds present in everyday scenes.

X. CONCLUSION

In this review, we have introduced ML theory, including deep learning (DL), and discussed a range of applications of ML theory in acoustics research areas. While our coverage of the advances of ML in the field of acoustics is not exhaustive, it is apparent that ML has enabled many recent advances. We hope this article can serve as inspiration for future ML research in acoustics. It is observed that large, publicly available datasets (e.g., Refs. 103, 250–252, 272, 302, and 303) have encouraged innovation across the acoustics field. ML in acoustics has enormous transformative potential, and its benefits can increase with open data.

Despite their limitations, ML-based methods provide good performance relative to conventional processing in many scenarios. However, ML-based methods are data-driven and require large amounts of representative training data to obtain reasonable performance. This can be seen as an expense of accurately modeling complex phenomena, as ML models often have very high capacity. In contrast, standard processing methods often have lower capacity, but are based on training-free statistical and mathematical models.
Based on this review, we foresee a transformation of acoustic processing from hand-engineering, basic-intuition-driven modeling to a more data-driven ML paradigm. The benefits of ML in acoustics cannot be fully realized without building-upon the indispensable physical intuition and theoretical developments within well-established sub-fields, such as array processing. Thus, development of ML theory in acoustics should be done without forgetting the physical principles describing our environments.

ACKNOWLEDGMENTS

This work was supported by the Office of Naval Research, Grant No. N00014-18-1-2118.


