1 Machine Learning and Communications: An Introduction

Deniz Gündüz, Yonina C. Eldar, Andrea Goldsmith, and H. Vincent Poor

1.1 Introduction

Wireless communications is one of the most prominent and impactful technologies of the last few decades. Its transformative impact on our lives and society is immeasurable. The fifth generation (5G) of mobile technology has recently been standardised and is now being rolled out. In addition to its reduced latency and higher data rates compared to previous generations, one of the promises of 5G is to connect billions of heterogeneous devices to the network, supporting new applications and verticals under the banner of the Internet of Things (IoT). The number of connected IoT devices is expected to increase exponentially in the coming years, enabling new applications including mobile healthcare, virtual and augmented reality, and self-driving cars as well as smart buildings, factories, and infrastructures.

In parallel with recent advances in wireless communications, modern machine learning (ML) techniques have led to significant breakthroughs in all areas of science and technology. New ML techniques continue to emerge, paving the way for new research directions and applications, ranging from autonomous driving and finance to marketing and healthcare, just to name a few. It is only natural to expect that a tool as powerful as ML should have a transformative impact on wireless communication systems, similarly to other technology areas. On the other hand, wireless communication system design has traditionally followed a model-based approach, with great success. Indeed, in communication systems, we typically have a good understanding of the channel and network models, which follow fundamental physical laws, and some of the current systems and solutions already approach fundamental information theoretical limits. Moreover, communication devices typically follow a highly standardized set of rules; that is, standardization dictates what type of signals to transmit, as well as when and how to transmit them, with tight coordination across devices and even different networks. As a result, existing solutions are products of research and engineering design efforts that have been optimized over many decades and generations of standards, which are based upon theoretical foundations and extensive experiments and measurements. From this perspective, compared to other areas in which ML has made significant advances in recent years, communication systems can be considered more amenable to model-based solutions rather than generic ML approaches. Therefore, one can question what potential benefits ML can provide to wireless communications [1–3].

The chapters in the first part of this book provide many answers to this question. First, despite the impressive advances in coding and communication techniques over the last few decades, we are still far from approaching the theoretical limits in complex networks of heterogeneous devices, or even understanding and characterizing such limits. Current design approaches are built upon engineering heuristics: we try to avoid or minimize interference through various access technologies, which allows us to reduce a complex interconnected network into many parallel point-to-point links. We then divide the design of the point-to-point communication systems into many different blocks, such as channel estimation, feedback, equalization, modulation, and coding, which are easier to model and may even lend themselves to optimal solutions. Although this modular design is highly attractive from an engineering perspective, it is generally suboptimal.

With 5G, we aim to connect billions of IoT devices, which have significantly different requirements in terms of latency and energy efficiency compared to conventional mobile handsets. The networks will have to sustain a much more diverse set of devices and traffic types, each with different requirements and constraints. This will require significantly more flexibility and more adaptive and efficient use of the available resources. Therefore, the performance loss due to the aforementioned modular design and separate optimization of the individual blocks is becoming increasingly limiting. Cross-layer design as an alternative has been studied extensively, but this type of design still depends on a model-based approach that is often significantly more complex than the modular design. This complexity has led to ad-hoc solutions for each separate cross-layer design problem and scenario, with performance gains that were often limited or nonexistent relative to more traditional modular design. Data-driven ML techniques can provide an alternative solution, with potentially larger gains in performance and reasonable complexity in implementation.

Second, in current systems, even with the modular approach, the optimal solution may be known theoretically, yet remain prohibitive to attain computationally. For example, we can write down the maximum likelihood detection rule in a multiuser scenario, but the complexity of its solution grows exponentially with the number of users [4]. Similarly, in a communication network with a multi-antenna transmitter and multiple single-antenna receivers, the optimal beamforming vectors can be written as the solution of a well-defined optimization problem, albeit with no known polynomial complexity solution algorithm [5]. As a result, we either provide low-complexity yet suboptimal solution techniques to the problem [6] or resort to simple solutions that can be solved in closed-form [7]. Recent results have shown that machine learning techniques can provide more attractive complexity-performance trade-offs [8, 9].

There has already been a significant amount of research proposing data-driven ML solutions for different components of a communication system, such as channel estimation [10–12], channel state information compression and feedback [13, 14], channel equalization [15], channel decoding [16], and more, many of which will be discussed in detail in the later chapters of this book. These solutions are shown to outperform conventional designs in many scenarios, particularly when we do not have a simple model of the communication system (e.g., when the channel coefficients do not follow a distribution with a known covariance matrix). However, another potential benefit of the ML approaches is to go beyond the aforementioned modular design and learn the best communication scheme in an end-to-end fashion rather than targeting each module separately [17, 18]. This end-to-end ML approach facilitates attacking much more complex problems with solutions that were deemed elusive with conventional approaches. A good example is the joint source-channel coding problem, which is typically divided into the compression and channel coding subproblems. Despite the known suboptimality of this approach, and decades of efforts in designing joint source-channel coding schemes, these often resulted in ad-hoc solutions for different source and channel combinations with formidable complexity. However, recent results have shown that ML can provide significant improvements in the end-to-end performance of joint source-channel coding [19–22].

An additional benefit of ML is that is does not require knowledge of channel parameters or statistics. In particular, many known communication methods rely on knowledge of the channel parameters or statistics or the ability to estimate these statistics with minimal resources. In modern wireless networks, this is no longer the case. Thus, ML can offer efficient techniques to compensate for unknown and varying channel knowledge [23, 24]. In addition, hardware limitations, such as the use of low bit rate quantizers or nonlinear power amplifiers, can significantly increase the complexity of the underlying channel model. This renders a design based on exact channel knowledge difficult to implement [25].

Another dimension of the potential synergies between communications and ML is the design of communication networks to enable ML applications [26]. While a significant amount of information is collected and consumed by mobile devices, most of the learning is still carried out centrally at cloud servers. Such a centralized approach to learning at the edge has limitations. First, offloading all the data collected by edge devices to the cloud for processing is not scalable. For example, an autonomous car is expected to collect terrabytes of data every day. As the number of vehicles increases, the cellular network infrastructure cannot support such a huge rise in data traffic solely from autonomous cars. A similar surge in traffic is expected from other connected devices. The sheer volume of the collected data makes such centralized approaches highly unsustainable. This is particularly problematic when the collected data has large dimensions and has low *information density*, which refers to the information within the data relevant for the underlying task. For example, a surveillance camera may collect and offload hours of recording of a still background, which has little use for the task of detecting intruders. Therefore, it is essential to enable edge devices with some level of intelligence to extract and convey only the relevant information for the underlying learning task [27–29].

Having all the intelligence centralized also causes significant privacy and security concerns [30–32]. Most of the data collected from edge devices, such as smart meters [33, 34], autonomous cars [35], health sensors [36], or entertainment devices [37], collect highly sensitive information that reveals significant personal information about our lives and daily habits. Sharing this data in bulk with other parties is a growing concern for consumers, which can potentially hinder the adoption of some of these services. The alternative can be to carry out local learning at each individual device with the available data; however, the locally available data can be limited in quantity and variety, resulting in overfitting.

Fully centralized intelligence also introduces latency. In many applications that involve edge devices, inference and action need to be fast. For example, autonomous vehicles must detect and avoid pedestrians or other obstacles rapidly. Even though some of the inference tasks can be carried out on board, devices may not have the necessary processing capability, and some of the inference tasks may require fusing information distributed across multiple edge devices. For example, an autonomous car may benefit from camera or LIDAR data from other nearby cars or terrain information available at a nearby base station to make more accurate decisions about its trajectory and speed. In such scenarios, it is essential to enable distributed inference algorithms that can rapidly gather the most relevant information and make the most accurate decisions [28].

In light of these observations, an important objective in merging communications and ML is to bring intelligence to the network edge, rather than offloading the data to the cloud and relying on centralized ML algorithms. The two core ingredients behind the recent success of ML techniques are massive datasets and tremendous memory and processing power to efficiently and rapidly process the available data to train huge models, such as deep neural networks (DNNs). Both of these ingredients are plentiful at the network edge, yet in a highly distributed manner. The second part of this book is dedicated to distributed ML algorithms, particularly focusing on learning across wireless networks, addressing these challenges and highlighting some of the important research achievements and remaining challenges.

Before revieweing the individual chapters in the book, we provide a brief overview of basic ML techniques, particularly focusing on their potential applications in communication problems.

1.2 Taxonomy of Machine Learning Problems

The type of problems to which machine learning algorithms are applied can be grouped into three categories: *supervised learning*, *unsupervised learning*, and *reinforcement learning (RL)*. All three types of problems have found applications in wireless communications. Here we provide a general overview of these different categories and what types of problems they are used to solve and then highlight their applications in wireless communications.

1.2.1 Supervised Learning

In supervised learning the goal is to teach an algorithm the input-output relation of a function. This can be applied to a wide variety of functions and input/output types. Consider, for example, the input vector denoted by $\mathbf{x} \in \mathcal{X}$ and the associated vector of target variables $\mathbf{c} \in \mathcal{C}$. Supervised learning problems are classified into two groups: if the input is mapped to one of a finite number of discrete classes (i.e., $\mathcal{C}$ is a discrete set), then the learning task is called *classification*. If, instead, the output can take continuous values (i.e., $\mathcal{C}$ is a continuous set), then the learning task is called *regression*. The often cited toy problem of classifying images into dog and cat labels is a classical example of a classification problem. Most classification tasks have more than two classes, and other common examples that are often used to compare and benchmark supervised learning algorithms are classification of handwritten digits and spam detection.

In communications, the design of a receiver for a fixed transmission scheme is a classification task. For example, consider a simple modulation scheme mapping input bits to constellation points. The receiver trying to map each received noisy symbol to one of the constellation points can be considered a classification problem. This can also be extended to decoding of coded messages transmitted over a noisy channel, where the decoder function tries to map a vector of received symbols to a codeword [16, 38].

Another application of classification in communications is the detection of the type of wireless signals in the air, which is often required for military applications, or for cognitive radios [39]. This includes the detection of the transmitting device by identifying the particular hardware impairments of each individual transmitter [40], detection of the modulation type used by the device [41–43], or detection of the wireless technology employed by the transmitter [44]. A dataset of synthetic simulated channel effects on wireless modulated signals of 11 different modulation types has been released in [45], which has significantly helped the research community to test and compare proposed techniques on a benchmark dataset.

A well-known example of a regression problem encountered in wireless communication systems is channel estimation [10, 11, 46], where the goal is to estimate the channel coefficients from noisy received versions of known pilot signals. While traditional channel estimation methods assume a known channel model, and try to estimate the parameters of this model through least squared or minimum mean-squared error estimation techniques, a data-driven channel estimator does not make any assumptions on the channel model. It instead relies on training data generated from the underlying channel. In the context of wireless communications, while a data-driven approach is attractive when an accurate channel model is not available, the requirement of a large training dataset can mean that training needs to be done off-line.

Supervised learning can also be used as an alternative method to rapidly obtain reasonable suboptimal solutions to complex optimization problems. In wireless networks, we often face highly complex nonconvex or combinatorial optimization problems, for example, in scheduling or deciding transmit powers in a network of interfering transmitters. Many of these problems do not have low-complexity optimal solutions; however, in practice, we need a fast solution to be implemented within the coherence time of the channels. Therefore, we typically resort to some low-complexity suboptimal solution. An alternative would be to train a neural network using the optimal solution for supervision. This can result in a reasonable performance that can be rapidly obtained once the network is trained. Such methods have been extensively applied to wireless network optimization problems with promising results in terms of the performance-complexity trade-off [8, 47].

1.2.2 Unsupervised Learning

In unsupervised learning problems, we have training data without any output values, and the goal is to learn functions that describe the input data. Such functions may be useful for more efficient supervised learning, as they can be seen as a method for feature extraction. They may also be used to make the input data more amenable to human understanding and interpretation. Unsupervised learning is not as well-defined as supervised learning, since it is not clear what type of description of the data we are looking for. Moreover, it is often the case that the measure to use to compare different descriptions is not obvious and may highly depend on the type of data and application we have in mind. Common unsupervised learning problems are *clustering*, *dimension reduction*, and *density estimation*, all of which have been used extensively in communication systems.

Indeed, clustering is nothing but source compression or quantization, where the goal is to identify a small number of representatives that can adequately represent all possible input vectors. In density estimation, the goal is to determine the distribution of data as accurately as possible from a limited number of samples. Parameterized density estimation is often used in wireless communications when estimating the channel from a limited number of pilot signals. Dimensionality reduction is similar to clustering in the sense that we want a more efficient representation of data, but rather than limiting this representation to a finite number of clusters, we limit its dimension. Projecting a large-dimensional input data to two or three dimensions is used for visualization. A common technique for dimensionality reduction is principle component analysis (PCA), which is also known as the Karhunen-Loeve transform and is used for lossy data compression. More recently neural networks in the form of autoencoders have been employed for dimensionality reduction. Autoencoders play an important role in the data-driven design of compression and communication schemes.

In machine learning, some approaches first try to learn the distribution of data (or the data as well as the output in the context of supervised learning). These are called *generative models*, because once the underlying distribution is learned, one can generate new samples from this distribution. With the advances in deep learning, DNNs have also been used in *deep generative models*, which have shown remarkable performance in modeling complex distributions. Two popular architectures for deep generative modeling are variational autoencoders (VAEs) [48] and generative adversarial networks (GANs) [49]. Generative models have recently been employed in [50, 51] to model a communication channel from data; such models can be used for training other communication components when no channel model is available.

1.2.3 Reinforcement Learning (RL)

RL is another class of machine learning problems, where the goal is to learn how to interact in a random unknown environment based on feedback received in the form of costs or rewards following each action. In contrast to supervised learning, where the outputs corresponding to a dataset of input samples are given, in RL these need to be learned through interactions with the environment. The environment has a state, and the agent interacts with the environment by taking actions. The goal is to learn the right action to take at each state in order to maximize (minimize) the long-term reward (cost). The cost/reward acquired depends both on the state and the action taken and is typically random with a stationary distribution. Notable examples of RL algorithms are game playing agents that can beat human masters in chess and Go, or more recently in Atari games or more advanced multiplayer video games.

A fundamental aspect of RL is the trade-off between *exploration* and *exploitation*. Exploration refers to taking new unexplored actions to gather more information about the environment to potentially discover actions with higher rewards (lower costs). Exploitation, on the other hand, refers to exploiting the actions that are more likely to provide higher rewards (lower costs) based on past observations. A conservative agent that is more likely to exploit its current knowledge risks losing out on high reward actions that it has never tried. On the other hand, an agent that keeps exploring without considering its past experiences ends up with a low reward. Hence, the goal is to find the right balance between exploration and exploitation.

RL has found applications in wireless networks as early as in the 1990s [52, 53], including power optimization in the physical layer for energy efficient operation [54]. Similarly to other machine learning tools, RL algorithms can be used for two types of problems requiring interactions with an environment, which can model the wireless network environment a device is operating in: the device might have an accurate model of the environment, but the solution of an optimal operation policy for the device may be elusive. In such a case, RL techniques can be used as a numerical solution technique to characterize the optimal (or near optimal) strategy for the device. Note that this is similar to the application of RL in games such as chess or Go, which have well-understood rules but are still highly complex to study methodically. This type of problem typically appears in networking, where multiple devices are scheduled to share the limited spectral resources.

Alternatively, RL methods can be used in wireless networking problems for which the environment is known to be stationary, yet we do not have a good model to characterize its statistical behaviour. This might be the case, for example, when operating over unlicensed bands, where it is difficult or impossible to model the statistics of device activations, spectrum activity, traffic arrivals, and channel variations. In such scenarios, RL tools can be used to learn the best operation strategy through direct interactions with the environment in an online fashion. An example of the application of RL techniques to such a problem is channel access for cognitive users, where the probability of availability of each channel is unknown to the user. This problem is formulated in [55] as a multiarmed bandit (MAB) problem, which is a special case of RL with a single state. The name MAB is motivated by bandit machines in casinos: a gambler wants to play the arm that has the highest chance of winning, but this is unknown in advance. Hence, the gambler has to try as many different arms as possible to discover the best one, while also trying to exploit the estimated best arm as much as possible, since trying each arm has a cost. In [56], scheduling of multiple energy-harvesting wireless transmitters is considered. Due to the presence of finite-capacity batteries, this problem is formulated as a restless MAB, which refers to the fact that the rewards of the arms are governed by an underlying Markov process, and the state of the arms may change even if they are not played.

1.3 ML Tools Used in Communication System Design

In the previous section we classified ML problems into three main groups. There are many different formulations and approaches within each of these categories. Moreover, many different techniques are available to solve the same problem, with each method providing a different complexity-performance trade-off depending on the implementation constraints, availability of data, and uncertainty in the system parameters. While providing background on all existing tools is beyond the scope of this chapter, we briefly review here three fundamental tools that are commonly used in the design of communication networks (and many other engineering problems), which also appear in most of the later chapters of the book. In particular, we provide a brief introduction to neural networks and how they are trained, particularly for supervised learning tasks, then present the concept of an autoencoder for unsupervised learning, and finally discuss the main challenges of RL along with common solution approaches.

1.3.1 Deep Neural Networks (DNNs)

An artificial neural network is a popular ML model consisting of several layers of “neurons,” which are processing units loosely inspired by biological neurons. In a *feedforward neural network*, neurons are organized into layers, and there are no feedback connections that would feed the output of any layer back into the model itself. Neural networks that incorporate feedback connections are called *recurrent neural networks*. Please see Fig. 1.1 for an illustration of a neural network architecture.

The first layer takes as input an affine function of the values of the input signal, while the following layers take as input affine functions of the outputs of the neurons in the previous layer. The left-most layer is typically considered the *input layer* and the right-most layer is the *output layer*, while the remaining layers in between are called the *hidden layers*; see Fig. 1.1. The number of hidden layers provides the *depth* of the network, which led to the terminology of “deep learning.” Each hidden layer typically consists of many neurons, the number of which determines the *width* of the network. Let d_k denote the width of the kth layer, where d_0 corresponds to the input dimension (e.g., the number of pixels in the input image). If we denote the output of neuron i in layer k by $x_i^{(k)}$, and its input by $y_i^{(k)}$, we have

$$y_i^{(k)} = w_{0,i}^{(k)} + \sum_{n=1}^{d_{k-1}} w_{n,i}^{(k)} x_n^{(k-1)}, \tag{1.1}$$

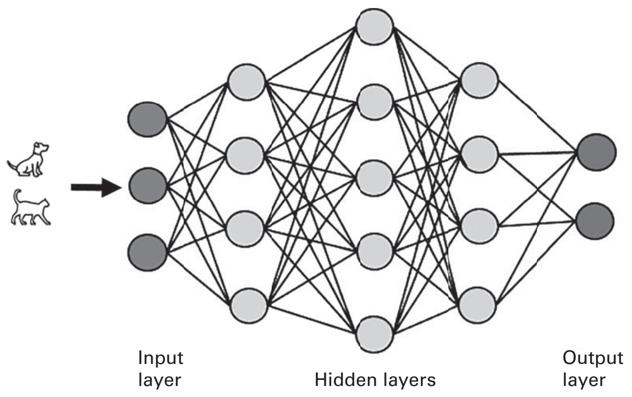

Figure 1.1 Neural network architecture.

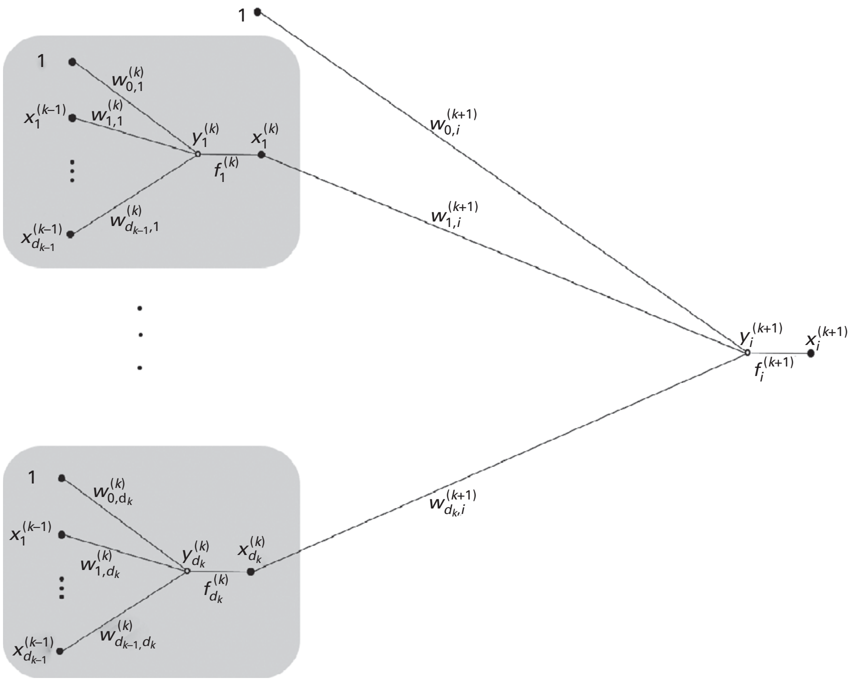

Figure 1.2 One layer of a neural network, and the forward propagation of the neuron activations.

where $w_{n,i}^{(k)}$ denotes the weight of the connection from the output of the nth neuron in layer $k-1$ to the input of the ith neuron in layer k. Here, $w_{0,i}^{(k)}$ corresponds to the bias term for the ith neuron at layer k. Each neuron then applies a nonlinear "activation function" on its input, denoted by $f_i^{(k)}$ for the ith neuron at layer k, resulting in

$$x_i^{(k)} = f_i^{(k)}(y_i^{(k)}), \tag{1.2}$$

as illustrated in Fig. 1.2.

The evaluation of the output values of the neurons in a neural network by recursively computing Eqs. (1.1) and (1.2) is called *forward propagation*, as the information flows from the input layer toward the output layer.

Different activation functions can be used in Eq. (1.2). The most common activation functions are the identity function, step function, a rectified linear unit (ReLU), sigmoid function, hyperbolic tangent, and softmax. Softmax is a generalization of the sigmoid function, which represents a binary variable. Note that the model in Eq. (1.2) allows for employing a different activation function for each neuron in the network. Softmax is often used as the activation function in the output layer of a classifier network to represent a probability distribution over n classes. On the other hand, ReLU is typically used as the activation function for hidden layers, although it is not possible to say which activation function will perform the best. This is often determined through trial and error.

Another aspect of a neural network that we must design is its *architecture*, which refers to the depth of the network and the width of each layer. Although a network with a single layer would be sufficient to fit the training dataset, deeper networks have been shown to generalize better to the test set. Deeper networks may also use less parameters in total; however, they are harder to train. Therefore, the network architecture is often determined through trial and error, taking into account any constraints on the training time and complexity.

Neural networks without an activation function, or an identity activation function, can only learn linear models; nonlinear activation functions are required to learn more complex nonlinear functions from the inputs to the outputs. Even though the choice of the activation function and the network architecture are important design parameters in practice, a network with a single hidden layer with a sigmoid activation function is sufficient to approximate (with arbitrary precision) any continuous function between any two Euclidean spaces [57, 58]. Note, however, that this is an existence result, and we still need to identify the right parameters that would provide the desired approximation. Identification of these parameters corresponds to the *training* stage of the neural network.

For a given neural network architecture, the output corresponding to each input data sample is determined by the weights, $w_{n,i}^{(k)}$. Therefore, the goal of the training process is to determine the weight values that will result in the best performance. Here, the performance measure will depend on the underlying problem. It can be the accuracy of detecting the correct label in a classification problem or determining the correct value in a regression problem. For example, in a multiclass classification problem with C classes, the width of the output layer is chosen as C, and the normalized values of the output layer are interpreted as the likelihoods of the corresponding classes. Let $\mathbf{s}_1, \ldots, \mathbf{s}_N$ denote the data points in the training dataset, with corresponding labels $\mathbf{l}_1, \ldots, \mathbf{l}_N$ represented as one-hot encoded vectors of size C, whose elements are all 0 except for the entry corresponding to the correct class, which is set to 1. That is, if data sample $\mathbf{s}_n$ in the training dataset belongs to class c, then we have $l_{n,c} = 1$, and $l_{n,m} = 0$ for $m \neq c$. In this case, the loss function will be given by

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{c=1}^{C} l_{n,c} \ln x_c^{(K)}(\mathbf{s}_n, \mathbf{w}), \tag{1.3}$$

where K is the number of layers in the network, $\mathbf{w}$ represents all the weights (including the bias terms) in the network, and we have written $x_k^{(K)}(\mathbf{s}_n, \mathbf{w})$ to represent the outputs of the neurons in the output layer to highlight the dependence of their values on the network weights as well as the input data. This error function is called the *cross entropy*, or the *log loss*, and is widely used as the objective/loss function when training parameterized classification algorithms.

The error function in neural networks is highly nonlinear and nonconvex, therefore it is difficult to find the globally optimal weight parameters that minimize Eq. (1.4). Instead, we resort to numerical approaches with the hope of finding reasonable, potentially locally optimal, solutions. A common numerical approach to solve such optimization problems is the *gradient descent* method, where we start from an initial value of the weight vector $\mathbf{w}$, and iteratively update it along the negative gradient direction:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \nabla E(\mathbf{w}(t)), \tag{1.4}$$

where $\eta > 0$ is the *learning rate*. Since the error function is defined over the whole training dataset, the gradient at each iteration must be computed for all the data points. Alternatively, we can use an unbiased estimate of the gradient at each iteration by computing it only at a random data sample. This is called *stochastic gradient descent*, and it results in faster convergence in training DNNs.

In a neural network, the error function is a complex function of the weights, and how to compute the gradient with respect to the weight vector is not obvious at all. Fortunately, the recursive structure of the network allows for computing the gradients in a systematic and efficient manner using the chain rule of differentiation. This is known as the *backpropagation algorithm*, also referred to as *backprop*. There are excellent textbooks explaining the technical details of the backpropagation algorithm and other techniques to speed up neural network training. We refer the readers to [59, 60] for further details.

1.3.2 Autoencoders

Autoencoders play an important role in the application of neural networks in wireless communications. An autoencoder is a pair of neural networks, called the *encoder* and *decoder* networks, trained together in an unsupervised manner in order to recover the input signal at the output of the decoder network with minimal distortion (see Fig. 1.3). The output of the encoder network is called the *bottleneck layer*, which typically has lower dimension compared to the input signal. After training, this bottleneck layer recovers a low-dimensional representation of the input signal that still allows the decoder network to recover the input signal within some distortion. Equivalently, the encoder acts as a data-driven dimensionality reduction technique,

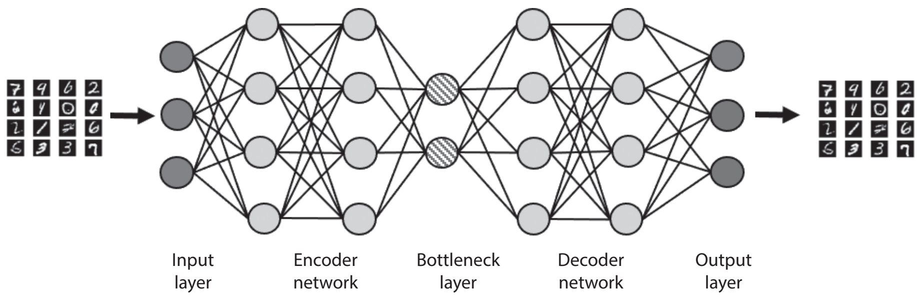

Figure 1.3 Autoencoder neural network architecture.

similarly to principal component analysis (PCA). Autoencoders were introduced in the 1980s [61], but they achieved competitive or superior performance compared to other unsupervised learning techniques only recently with the advances in computational capabilities that allowed fast training of sufficiently complex autoencoder models on large datasets [62].

While autoencoders have been widely used for preprocessing in supervised learning and visualization, their potential for data compression was also recognized in the 1990s [63, 64]. In data compression, the low-dimensional representation generated by the encoder network must be converted into a bit sequence that can then be used by the decoder network to recover the input signal. Since we are typically limited by a finite bit budget, the output of the encoder network needs to be quantized, which introduces additional distortion at the reconstruction; hence, the autoencoder must be trained together with the quantization layer. This creates some difficulty as the quantization operation is not differentiable, which prevents backpropagation during training. Several methods have been introduced in the literature to deal with this, such as straight through estimation, which ignores the quantization operation in the backpropagation [65], or approximating the quantization operation with a differentiable function [66].

Denoising autoencoders were introduced in [67] in order to provide robustness in the generated low-dimensional representation to partial corruptions in the input signal. In a denoising autoencoder, some components of the input signal are randomly erased before being fed into the encoder network, and the autoencoder is trained to recover the uncorrupted input signal with the highest fidelity. This way, the denoising autoencoder learns not only to represent the input signal efficiently, but also to recover the random perturbation introduced on the input signal. If we treat the basic autoencoder as applying source compression, we can consider the setting in the denoising autoencoder as remote source coding [68].

Inspired by the denoising autoencoder, communication over a noisy channel is treated in [69], where the channel introduces random noise on the signal generated by the encoder output before it is fed into the decoder network. To the best of our knowledge, this was the first work to treat the design of the encoder and decoder in a channel communication problem as training of an autoencoder. In this model, the input signal is a sequence of equally likely bits, representing the input message, and the goal is to transmit these inputs bit sequence over the noisy channel as reliably as possible. Note that the input message has maximum entropy, so there is no structure to be learned; instead, the goal of the autoencoder in this model is to learn the set of input codewords that can best mitigate the impairments introduced by the channel. Here, we can remove the low-dimensionality requirement on the bottleneck layer, as we typically introduce additional redundancy against channel noise. The dimension of the bottleneck layer determines the code rate.

A major challenge in the design of channel coding schemes using autoencoders is the sheer number of input signals that must be recognized at the decoder. In particular, for a message size of n bits, we have 2^n possible input messages; equivalently, the number of classes that must be recognized by the decoder network grows exponentially with n. This makes the training of such an autoencoder highly challenging for moderate to large blocklengths.

1.3.3 Deep Reinforcement Learning (DRL)

DRL refers to the particular family of RL techniques where DNNs are employed to approximate the agent's strategy or utility function. In RL techniques, actions are chosen at each state based on the expected rewards they will provide. The expected reward from each action is updated continuously based on the past experiences of the agent. In general, it is difficult to map these experiences to an accurate estimate of the reward function. In many complex scenarios, the agent will not be able to explore all possible state-action pairs a sufficient number of times to make accurate estimates. For such problems, DNNs are used to estimate the average reward for each state-action pair or directly predict the best action to take (more precisely, a distribution over actions). DNN parameters are updated as in neural network training, but based on the rewards accrued instead of the labels used in supervised training of DNNs. Thanks to the generalization power of DNNs, DRLs can solve highly complex RL problems even with continuous state and action spaces. However, similarly to supervised learning problems, to achieve a good performance level, DRLs require significant training data and training time. Hence, in the wireless context, it may be more appropriate for stationary scenarios where the training can be carried out off-line using the available data or the model of the underlying system, and the trained network is used afterward. They can also be effective when the variations in the system statistics are relatively slow, so that it is possible to track the optimal policy with limited additional training.

1.4 Overview of the Book

This book is aimed at postgraduate students, Ph.D. students, researchers, and engineers working on communication systems, as well as researchers in machine learning seeking an understanding of the potential applications of ML in wireless communications. To read and fully understand the content it is assumed that the reader has some background in communication systems and basic ML. The book consists of 17 technical chapters written by leading experts in their areas, organised into two parts. The first part, consisting of 10 chapters, considers the application of machine learning techniques in solving a variety of wireless communication problems. The remaining seven chapters of the book comprise the second part, which focuses on the design and optimization of wireless network architectures to carry out machine learning tasks in an efficient manner.

Each chapter is self-contained and the chapters are independent of each other. Therefore, there is flexibility in selecting material both for university courses and short seminars. A brief summary of each chapter is given next.

1.4.1 Part I: Machine Learning for Wireless Networks

In Chapter 2, the authors treat the problem of information transmission over a wireless channel in an end-to-end manner, considering jointly the source compression and channel coding problems. This chapter explores joint source-channel coding schemes based on an autoencoder architecture, where a pair of jointly trained DNNs act as the encoder and the decoder. The chapter focuses on image and text transmission problems and shows that a neural network aided design not only outperforms state of the art digital transmission schemes, but also provides graceful degradation with variations in channel quality.

In Chapter 3, the authors extend the joint source-channel coding problem treated in the previous chapter to the lossy transmission of correlated sources over a network of orthogonal links. Joint source-network coding is known to be an NP-hard problem, and no practical solution exists for general sources (e.g., images). This chapter explores neural network aided design of such codes for arbitrary network topologies.

In Chapter 4, the author focuses explicitly on the design of channel codes using deep learning. The chapter highlights that deep learning based design of channel coding results in a broader range of code structures compared to traditional code designs, which can then be optimized efficiently through gradient descent. However, identifying the right network architectures and efficiently training the network parameters are challenges that still need to be overcome. The chapter provides a range of neural network aided code designs and highlights many different scenarios where such designs outperform conventional codes.

Chapter 5 focuses on the applications of deep learning to the problems of channel estimation, feedback, and signal detection, particularly for orthogonal frequency division multiplexing (OFDM) and millimeter-wave (mmWave) systems. Convolutional neural networks (CNNs) are employed for these tasks and are shown to learn channel features and estimate the channel successfully, after being trained on a large dataset generated using the channel model. For the signal detection task, “deep unfolding” is employed to benefit from the structure of an iterative decoder to improve the speed and accuracy of the training process.

In Chapter 6, the authors contrast the traditional model-based design approach to communication systems with the data-driven approach based on ML. They demonstrate through various examples that a combination of the two approaches, as model-based ML, can benefit from the best of both worlds. In particular, such designs use ML to learn unknown aspects of the system model and reduce complexity, while benefiting from the lower training requirements and near-optimal performance of model-based approaches.

Chapter 7 tackles the challenging problem of distributed optimization in wireless networks, where the nodes have to make decisions based on their local view of the network, while their actions collaboratively decides the global reward or cost function. This chapter considers the framework in which the nodes can exchange limited amounts of information through backhaul links in order to coordinate their actions. This highly complicated nonconvex optimization problem is solved by employing DNNs trained in an unsupervised manner.

Chapter 8 also deals with the radio resource allocation problem in wireless networks employing neural networks, which are trained in an off-line manner and are employed for inference assuming the radio environment remains sufficiently stationary with respect to the training setting. This chapter focuses on the global energy efficiency of the network, defined as the ratio of the total throughput achieved over the network to the total power consumption. The authors show that neural networks can be trained to operate close to the optimal performance, while having much lower inference complexity, albeit at the expense of additional training cost.

Chapter 9 focuses on the applications of RL in physical layer wireless network problems. After a brief overview of Markov decision processes (MDPs), partially observable MDPs, and multiarmed bandits, the authors present some of the most widely used algorithms, in particular Q-learning, SARSA, and deep RL. They then provide a number of examples of how these techniques can be applied in a variety of wireless networking problems from power management and cache content optimization to channel access and real-world spectrum sharing problems.

In Chapter 10, the authors present scalable learning frameworks that are capable of processing and storing large amounts of data generated in a large network. This requires parallel learning algorithms that can decompose the global learning problem into smaller tasks that can be carried out locally. The authors propose Bayesian nonparametric learning and RL as potential tools for scalable learning in the wireless context. Finally, they present several practical use cases for the presented tools. The authors also touch upon the model interpretability and adaptivity aspects of these solutions.

Chapter 11 deals with the fundamental limits of communication over noisy channels. Shannon's channel coding theorem identifies the fundamental limit of communication over a memoryless noisy channel as a single-letter mutual information expression, which is maximized over input distributions. For certain channels with feedback, capacity can be formulated as a directed information. This chapter introduces RL tools for the identification and optimization of the capacity for both known and unknown channel models.

1.4.2 Part II: Wireless Networks for Machine Learning

Chapter 12 presents a general introduction to collaborative learning and how it can be implemented over wireless nodes. The authors first provide an overview of distributed learning algorithms, including federated and fully decentralized learning. They also present methods to reduce the communication load of these distributed learning algorithms, which measures the amount of information that must be exchanged among the devices that participate in the learning process. Then, resource allocation and scheduling problems are presented when nodes within physical proximity of each other carry out distributed learning over a shared wireless medium.

Chapter 13 focuses on federated learning at the wireless network edge. After a comprehensive introduction to the federated learning framework and existing algorithms, the authors focus on the implementation of federated learning with limited communication resources. The adaptation to available channel resources is achieved by adjusting the number of local iterations.

Chapter 14 also deals with federated learning, focusing on the communication bottleneck. The authors present a quantization theoretic framework to reduce the communication load from the devices to the parameter server in federated learning. In particular, scalar quantization, subtractive dithering, and universal vector quantization techniques are considered to reduce the communication load in federated learning.

While Chapter 15 also considers federated learning over wireless networks, the main focus of this chapter is to highlight how the signal superposition property of the wireless medium can be exploited to increase the speed and accuracy of the learning process. In this chapter, unlike in conventional digital communication techniques, the devices participating in the federated learning process are allowed to transmit their model updates in an uncoded/uncompressed fashion simultaneously over the same channel resources. With this approach the wireless medium becomes a computation tool instead of solely enabling the transmission of information from the transmitter to a receiver.

Chapter 16 presents federated distillation as a communication-efficient distributed learning framework. The authors first introduce the concepts of knowledge distillation and codistillation and then explain how these techniques can be exploited in the federated learning setting to reduce the communication load. The chapter concludes with the application of federated distillation to two selected applications.

Chapter 17 addresses the privacy aspects of federated learning in the wireless context. It is known that released model parameters in federated learning may leak sensitive information about the underlying datasets. Differential privacy has been introduced as a method to address this privacy leakage. This chapter studies how differential privacy can be adopted when federated learning is carried out across wireless nodes.

Chapter 18 focuses on the inference aspect of machine learning at the wireless network edge. The authors first overview techniques such as network splitting, joint source-channel coding, and inference-aware scheduling to improve the speed and efficiency of inference over wireless networks. Then, they provide a detailed analysis of pruning-based dynamic neural network splitting and dynamic compression ratio selection methods.

In conclusion, we would like thank all the authors for their contributions to this book and for their hard work in presenting the material in a unified and accessible fashion. We hope that you, the reader, will find these expositions to be useful.

References

[1] D. Gündüz et al., “Machine learning in the air,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2184–2199, 2019.

[2] N. Farsad et al., “Data-driven symbol detection via model-based machine learning,” *Communications in Information and Systems*, 2020.

[3] M. Chen et al., “Communication efficient federated learning,” in *Proc. National Academy of Sciences (PNAS)*, doi:10.1073/pnas.2024789118, April 2021.

[4] S. Verdu, *Multiuser Detection*, 1st ed. Cambridge University Press, 1998.

[5] Y. Liu, Y. Dai, and Z. Luo, “Coordinated beamforming for MISO interference channel: Complexity analysis and efficient algorithms,” *IEEE Trans. on Signal Processing*, vol. 59, no. 3, pp. 1142–1157, 2011.

[6] Q. Shi et al., “An iteratively weighted MMSE approach to distributed sum-utility maximization for a MIMO interfering broadcast channel,” *IEEE Trans. on Signal Processing*, vol. 59, no. 9, pp. 4331–4340, 2011.

[7] M. Joham, W. Utschick, and J. A. Nossek, “Linear transmit processing in MIMO communications systems,” *IEEE Trans. on Signal Processing*, vol. 53, no. 8, pp. 2700–2712, 2005.

[8] H. Sun et al., “Learning to optimize: Training deep neural networks for interference management,” *IEEE Trans. on Signal Processing*, vol. 66, no. 20, pp. 5438–5453, 2018.

[9] H. Huang et al., “Unsupervised learning-based fast beamforming design for downlink MIMO,” *IEEE Access*, vol. 7, pp. 7599–7605, 2019.

[10] H. Ye, G. Y. Li, and B. Juang, “Power of deep learning for channel estimation and signal detection in OFDM systems,” *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 114–117, 2018.

[11] M. B. Mashhadi and D. Gündüz, “Pruning the Pilots: Deep Learning-Based Pilot Design and Channel Estimation for MIMO-OFDM Systems,” in *IEEE Transactions on Wireless Communications*, vol. 20, no. 10, pp. 6315–6328, Oct. 2021.

[12] M. Boloursaz Mashhadi and D. Gündüz, “Deep learning for massive MIMO channel state acquisition and feedback,” *Journal of Indian Institute of Sciences*, no. 100, p. 369–382, 2020.

[13] C. Wen, W. Shih, and S. Jin, “Deep learning for massive MIMO CSI feedback,” *IEEE Wireless Communications Letters*, vol. 7, no. 5, pp. 748–751, 2018.

[14] M. B. Mashhadi, Q. Yang, and D. Gündüz, “Distributed deep convolutional compression for massive MIMO CSI feedback,” in *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2621–2633, Apr. 2021.

[15] W. Xu et al., “Joint neural network equalizer and decoder,” in *15th Int. Symp. on Wireless Communication Systems (ISWCS)*, pp. 1–5, 2018.

[16] T. Gruber et al., “On deep learning-based channel decoding,” in *51st Annual Conf. on Information Sciences and Systems (CISS)*, pp. 1–6, 2017.

[17] S. Dorner et al., “On deep learning-based communication over the air,” in *51st Asilomar Conf. on Signals, Systems, and Computers*, pp. 1791–1795, 2017.

[18] A. Felix et al., “OFDM-autoencoder for end-to-end learning of communications systems,” in *IEEE Int. Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, 2018.

[19] E. Bourtsoulatze, D. Burth Kurka, and D. Gündüz, “Deep joint source-channel coding for wireless image transmission,” *IEEE Trans. on Cognitive Communications and Networking*, vol. 5, no. 3, pp. 567–579, 2019.

[20] N. Farsad, M. Rao, and A. Goldsmith, “Deep learning for joint source-channel coding of text,” in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2326–2330, 2018.

[21] D. B. Kurka and D. Gündüz, “DeepJSCC-f: Deep joint source-channel coding of images with feedback,” *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 178–193, 2020.

[22] D. B. Kurka and D. Gündüz, “Bandwidth-Agile Image Transmission With Deep Joint Source-Channel Coding,” in *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 8081–8095, Dec. 2021.

[23] N. Shlezinger, R. Fu, and Y. C. Eldar, “Deepsic: Deep soft interference cancellation for multiuser MIMO detection,” *IEEE Trans. on Wireless Communications*, pp. 1–1, 2020.

[24] N. Shlezinger et al., “ViterbiNet: A deep learning based Viterbi algorithm for symbol detection,” *IEEE Trans. on Wireless Communications*, vol. 19, no. 5, pp. 3319–3331, 2020.

[25] N. Shlezinger and Y. C. Eldar, “Deep Task-Based Quantization”, Entropy 2021, 23, 104, January 2021.

[26] D. Gündüz et al., “Communicate to learn at the edge,” *IEEE Communications Magazine*, vol. 58, no. 12, pp. 14–19, Dec. 2020.

[27] N. Shlezinger, Y. C. Eldar, and M. R. D. Rodrigues, “Hardware-limited task-based quantization,” *IEEE Trans. on Signal Processing*, vol. 67, no. 20, pp. 5223–5238, 2019.

[28] M. Jankowski, D. Gunduz, and K. Mikolajczyk, “Wireless image retrieval at the edge,” *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 89–100, Jan. 2021.

[29] P. Neuhaus et al., “Task-based analog-to-digital converters,” *IEEE Trans. on Signal Processing*, vol. 69, pp. 5403–5418, 2021.

[30] C. Ma et al., “On safeguarding privacy and security in the framework of federated learning,” *IEEE Network*, vol. 34, no. 4, pp. 242–248, 2020.

[31] K. Wei et al., “Federated learning with differential privacy: Algorithms and performance analysis,” *IEEE Trans. on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[32] B. Hasırcıoğlu and D. Gündüz, “Private Wireless Federated Learning with Anonymous Over-the-Air Computation,” ICASSP 2021 – 2021 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5195–5199.

[33] O. Tan, J. Gómez-Vilardebó, and D. Gündüz, “Privacy-cost trade-offs in demand-side management with storage,” *IEEE Trans. on Information Forensics and Security*, vol. 12, no. 6, pp. 1458–1469, 2017.

[34] G. Giaconi, D. Gunduz, and H. V. Poor, “Privacy-aware smart metering: Progress and challenges,” *IEEE Signal Processing Magazine*, vol. 35, no. 6, pp. 59–78, 2018.

[35] S. Karnouskos and F. Kerschbaum, “Privacy and integrity considerations in hyperconnected autonomous vehicles,” *Proc. IEEE*, vol. 106, no. 1, pp. 160–170, 2018.

[36] M. Malekzadeh et al., “Protecting sensory data against sensitive inferences,” in *Proc. 1st Workshop on Privacy by Design in Distributed Systems*, 2018.

[37] V. Sivaraman et al., “Smart iot devices in the home: Security and privacy implications,” *IEEE Technology and Society Magazine*, vol. 37, no. 2, pp. 71–79, 2018.

[38] E. Nachmani et al., “Deep learning methods for improved decoding of linear codes,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 119–131, 2018.

[39] O. A. Dobre et al., “Survey of automatic modulation classification techniques: classical approaches and new trends,” *IET Communications*, vol. 1, no. 2, pp. 137–156, 2007.

[40] L. J. Wong, W. C. Headley, and A. J. Michaels, “Specific emitter identification using convolutional neural network-based IQ imbalance estimators,” *IEEE Access*, vol. 7, pp. 33 544–33 555, 2019.

[41] T. J. O’Shea, J. Corgan, and T. C. Clancy, “Convolutional radio modulation recognition networks,” in *Engineering Applications of Neural Networks*, C. Jayne and L. Iliadis, eds. Springer International Publishing, pp. 213–226, 2016.

[42] T. J. O’Shea, T. Roy, and T. C. Clancy, “Over-the-air deep learning based radio signal classification,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, 2018.

[43] P. Triantaris et al., “Automatic modulation classification in the presence of interference,” in *European Conf. on Networks and Communications (EuCNC)*, pp. 549–553, 2019.

[44] M. Kulin et al., “End-to-end learning from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications,” *IEEE Access*, vol. 6, pp. 18484–18501 2018.

[45] T. O’Shea and N. West, “Radio machine learning dataset generation with gnu radio,” *Proc. GNU Radio Conf.*, vol. 1, no. 1, 2016.

[46] P. Dong et al., “Deep cnn-based channel estimation for mmwave massive MIMO systems,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 5, pp. 989–1000, 2019.

[47] W. Cui, K. Shen, and W. Yu, “Spatial deep learning for wireless scheduling,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1248–1261, 2019.

[48] C. Doersch, “Tutorial on variational autoencoders,” *arXiv preprint*, arXiv stat.ML.1606.05908, 2016.

[49] I. Goodfellow et al., “Generative adversarial nets,” in *Advances in Neural Information Processing Systems*, vol. 27, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, eds. Curran Associates, Inc., 2014, pp. 2672–2680.

[50] H. Ye et al., “Channel agnostic end-to-end learning based communication systems with conditional gan,” in *IEEE Globecom Workshops (GC Wkshps)*, pp. 1–5, 2018.

[51] T. J. O’Shea, T. Roy, and N. West, “Approximating the void: Learning stochastic channel models from observation with variational generative adversarial networks,” in *Int. Conf. on Computing, Networking and Communications (ICNC)*, 2019, pp. 681–686.

[52] J. A. Boyan and M. L. Littman, “Packet routing in dynamically changing networks: A reinforcement learning approach,” in *Proc. 6th Int. Conf. on Neural Information Processing Systems*, 1993, p. 671–678.

[53] S. Singh and D. Bertsekas, “Reinforcement learning for dynamic channel allocation in cellular telephone systems,” in *Advances in Neural Information Processing Systems*, vol. 9, M. C. Mozer, M. Jordan, and T. Petsche, eds. MIT Press, pp. 974–980, 1997.

[54] T. X. Brown, “Low power wireless communication via reinforcement learning,” in *Proc. 12th Int. Conf. on Neural Information Processing Systems*, pp. 893–899, 1999.

[55] L. Lai et al., "Cognitive medium access: Exploration, exploitation, and competition," *IEEE Trans. on Mobile Computing*, vol. 10, no. 2, pp. 239–253, 2011.

[56] P. Blasco and D. Gündüz, "Multi-access communications with energy harvesting: A multi-armed bandit model and the optimality of the myopic policy," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 3, pp. 585–597, 2015.

[57] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signal Systems 2*, pp. 303–314, 1989.

[58] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[59] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

[60] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[61] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Internal Representations by Error Propagation*. MIT Press, pp. 318–362, 1986.

[62] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[63] Y. Yaginuma, T. Kimoto, and H. Yamakawa, "Multi-sensor fusion model for constructing internal representation using autoencoder neural networks," in *Proc. Int. Conf. on Neural Networks (ICNN'96)*, vol. 3, pp. 1646–1651, 1996.

[64] A. Steudel, S. Ortmann, and M. Glesner, "Medical image compression with neural nets," in *Proc. 3rd Int. Symp. on Uncertainty Modeling and Analysis and Annual Conf. of the North American Fuzzy Information Processing Society*, 1995, pp. 571–576.

[65] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint*, arXiv cs.LG.1308.3432, 2013.

[66] R. Gong et al., "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2019, pp. 4851–4860.

[67] P. Vincent et al., "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. on Machine Learning*, pp. 1096–1103, 2008.

[68] S. I. Gelfand and M. S. Pinsker, "Coding of sources on the basis of observations with incomplete information," *Problems of Information Transmission*, vol. 15, pp. 115–125, 1979.

[69] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Unsupervised representation learning of structured radio communication signals," in *1st Int. Workshop on Sensing, Processing and Learning for Intelligent Machines (SPLINE)*, pp. 1–5, 2016.