Recent Advances in Deep Learning for Channel Coding: A Survey

Toshiki Matsumine, *Member, IEEE*, and Hideki Ochiai, *Fellow, IEEE*

Abstract—This paper provides a comprehensive survey on recent advances in deep learning (DL) techniques for the channel coding problems. Inspired by the recent successes of DL in a variety of research domains, its applications to the physical layer technologies have been extensively studied in recent years, and are expected to be a potential breakthrough in supporting the emerging use cases of the next generation wireless communication systems such as 6G. In this paper, we focus exclusively on the channel coding problems and review existing approaches that incorporate advanced DL techniques into code design and channel decoding. After briefly introducing the background of recent DL techniques, we categorize and summarize a variety of approaches, including model-free and mode-based DL, for the design and decoding of modern error-correcting codes, such as low-density parity check (LDPC) codes and polar codes, to highlight their potential advantages and challenges. Finally, the paper concludes with a discussion of open issues and future research directions in channel coding.

Index Terms—Channel coding, deep learning (DL), low-density parity check (LDPC) codes, machine learning (ML), neural network, polar codes, turbo codes.

I. INTRODUCTION

Channel coding is a well-established area of research with a long history dating back to the Shannon's theory [1] where he introduced the *Shannon limit* as the maximum rate at which information can be transmitted over a given communication channel. Subsequently, researchers have made tremendous efforts to develop a practical coding scheme that approaches the Shannon limit at a realistic implementation cost [2]. The notable successes in coding theory include the invention of modern capacity-approaching codes such as turbo codes [3], low-density parity check (LDPC) codes [4], and polar codes [5]. These coding techniques have contributed significantly to various communication systems, such as wired and wireless communications, as well as storage systems, from the viewpoint of improving reliability and energy efficiency.

Due to the emerging wireless applications, including extended reality (XR) for telemedicine, tactile Internet, vehicle-to-everything (V2X), and wireless data centers, the next generation wireless communication systems impose unprecedentedly diverse and stringent requirements for, e.g., ultra-high data rate, ultra-low latency, and high energy efficiency [6]–[35]. In 5G, the main use cases are enhanced mobile broadband (eMBB), ultra-reliable low-latency communication (URLLC), and massive machine type communication (mMTC), and for

each case, the system requirement is specified in terms of a single key performance indicator (KPI), such as throughput, latency, reliability, and energy efficiency. On the other hand, due to the diversity of applications, many use cases in the next wireless communications such as 6G will require trade-offs among different KPIs, which poses a new challenge for the design of physical layer techniques [33], [36], [37].

Traditionally, the design of coding schemes has been based on mathematical models and expert knowledge, such as coding theory and information theory. Although this approach has contributed significantly to the recent progress in practical channel coding, it also has limitations. Specifically, it relies on mathematical models that do not fully capture real-world environments, and thus there is always a mismatch between the model for which we design systems and the actual environment to which they are applied. In addition, in the next generation communications, the system design problem will become increasingly complex due to demanding requirements and therefore will not be mathematically tractable in most cases. To address these issues, *data-driven* approaches to communication system design based on machine learning (ML) techniques have emerged as a new paradigm that supports or replaces the conventional system design based on mathematical models.

In particular, inspired by the recent successes of deep learning (DL) technologies in broad research areas, their applications in communication systems have been extensively studied. This DL trend has been accelerated by the development of dedicated DL frameworks, such as Tensorflow [38] and Pytorch [39], which makes it easier for researchers to implementate their DL algorithms. Furthermore, most of them are built with graphics processing unit (GPU) acceleration provided by the NVIDIA compute unified device architecture (CUDA) Deep Neural Network library (cuDNN), which significantly speeds up DL training due to its ability to perform parallel computations and high memory bandwidth. In addition, various DL processors such as field programmable gate arrays (FPGAs) and tensor processing unit (TPU) have been explored in the literature [40], [41], enabling efficient hardware implementations of DL-based communications and networking in beyond 5G and 6G.

A. DL Applications to The Physical Layer

In recent years, DL has been successfully applied to the physical layer of communication systems [1] [31], [42], [44], [46]–[50], [58]–[73]. One of the seminal works is [58], where

T. Matsumine is with the Institute of Advanced Sciences, Yokohama National University, Yokohama, Japan (e-mail: matsumine-toshiki-mh@ynu.ac.jp)

H. Ochiai is with the Graduate School of Engineering, Osaka University, Osaka, Japan (e-mail: ochiai@comm.eng.osaka-u.ac.jp)

[1]We note that ML techniques for the physical layer have been studied sporadically for many years, e.g., in [55]–[57]. However, the applications of DL are rather new, which started to flourish around 2016.

TABLE I
Survey papers related to our work.

Year	Reference	Contents related to channel coding.
2017	Wang et al. [42]	Review of early works on DL-based decoding.
2019	Zhang et al. [43]	Brief introduction of DL-based channel decoding.
	Gunduz et al. [44]	Brief review of DL-aided decoding.
	Balatsoukas et al. [45]	Review of deep unfolding for channel decoding.
2020	Samad et al. [46]	Addressing DL-based channel decoding.
	Zhang et al. [47]	Review of DL-based decoding and code construction.
2021	Ly et al. [48]	Review of DL applications for LDPC code identification, decoding.
2023	Mao et al. [49]	Briefly addressing DL-aided decoder and genetic algorithm for code construction.
	Akrout et al. [50]	Domain generalization [51], [52] in the channel decoding problem.
2024	Ye et al. [53]	Review of DL-based decoding for turbo, LDPC, and polar codes.
	Rowshan et al. [54]	Comprehensive survey on channel coding with brief introduction of DL for channel decoding.

the authors introduced the concept of end-to-end learning for communication systems, which is considered as autoencoder (AE). The authors also introduced other DL applications, such as modulation classification and radio transformer networks. Later on, many papers discussed potential applications of DL to communication problems, such as channel decoding, signal detection, channel modeling, and multiple-input multiple-output (MIMO) signal detection [42], [44], [59]–[63].

One of the notable advances in DL for the physical layer is the development of an open source Python library, called *Sionna*, which was released by NVIDIA in 2022 [74]. It supports link-level simulation of multi-user (MU)-MIMO systems with 5G-compliant channel codes, the third generation partnership project (3GPP) channel models, channel estimation, and so forth. Each building block is implemented using TensorFlow and allows for gradient-based optimization via backpropagation. Sionna alo has a native NVIDIA GPU support.

As the number of publications related to DL-based approaches for physical layer technologies has been increasing almost exponentially, it is important to classify and summarize them to highlight the current state and challenges. However, despite its importance, only a handful of survey papers have been dedicated to the channel coding problems so far.

B. Our Scope and Related Works

Existing papers on DL for channel coding may be classified into the following categories:

1) DL-based code design,
2) DL-based channel decoding,
3) End-to-end learning for communication systems.

In DL-based code design, the optimization of code parameters, such as the degree distribution of LDPC codes and the locations of frozen bits in polar codes, is performed using DL techniques. Channel decoding is a popular DL application as the decoding problem is essentially the classification problem which is what DL is good at. This approach utilizes a deep neural network (DNN) to replace or augment a conventional channel decoder with the purpose of improving the error correction performance or reducing the complexity and/or latency. End-to-end learning of communication systems is another popular application, where a transmitter-receiver pair is often completely replaced by "black-box" DNNs and trained over a differential channel model in an end-to-end fashion. In

this approach, not only a channel encoder-decoder pair can be trained, but also other physical layer components such as source encoder-decoder and symbol mapper-demapper.

Although end-to-end learning has been extensively studied in the literature, and this approach would be particularly promising for a new paradigm of semantic communication [68], [70], [72], [75]–[83], the complete replacement of transceivers by DNNs poses several challenges in practice and does not seem feasible at this time. Therefore, this paper focuses on DL-based approaches that may be applicable to existing systems with appropriate modifications. In particular, we consider code design using offline DL techniques and DL-based channel decoding to replace or support conventional decoders[2].

In Table I, we summarize existing survey papers related to our scope in this work. We emphasize that most of the existing survey papers cover a wide range of DL applications in the physical layer, rather than dealing with DL-assisted channel coding in a comprehensive manner. Nevertheless, the paper most closely related to our work would be [47], in which the authors discussed a wide range of DL applications in the physical layer, including the channel coding problems such as channel decoding and code construction. However, since its publication in 2020, a significant number of new techniques have been proposed. By focusing solely on the channel coding problems, we attempt to provide a comprehensive survey including the state-of-the-art techniques.

The overall organization of this paper is visualized in Fig. 1. In Section II, we briefly introduce the basics of DL techniques and the state-of-the-art models to facilitate the understanding of our survey. In Section III, we review DL-based design of LDPC and polar codes. Then, in Section IV, we consider various DL approaches to the channel decoding problem. Finally, we conclude this paper by discussing the challenges and future directions in Section V to stimulate further research. We provide a list of abbreviations that we use after Section V.

II. BRIEF INTRODUCTION OF DEEP LEARNING

In this section, we briefly review the basics of DL technologies, starting with neural networks and their optimization. We

---

[2]Although we mainly focus on *classical* error-correcting codes in this paper, DL-based code design and decoding has also been extensively studied in the realm of *quantum* error-correcting codes, e.g., in [84]–[90].
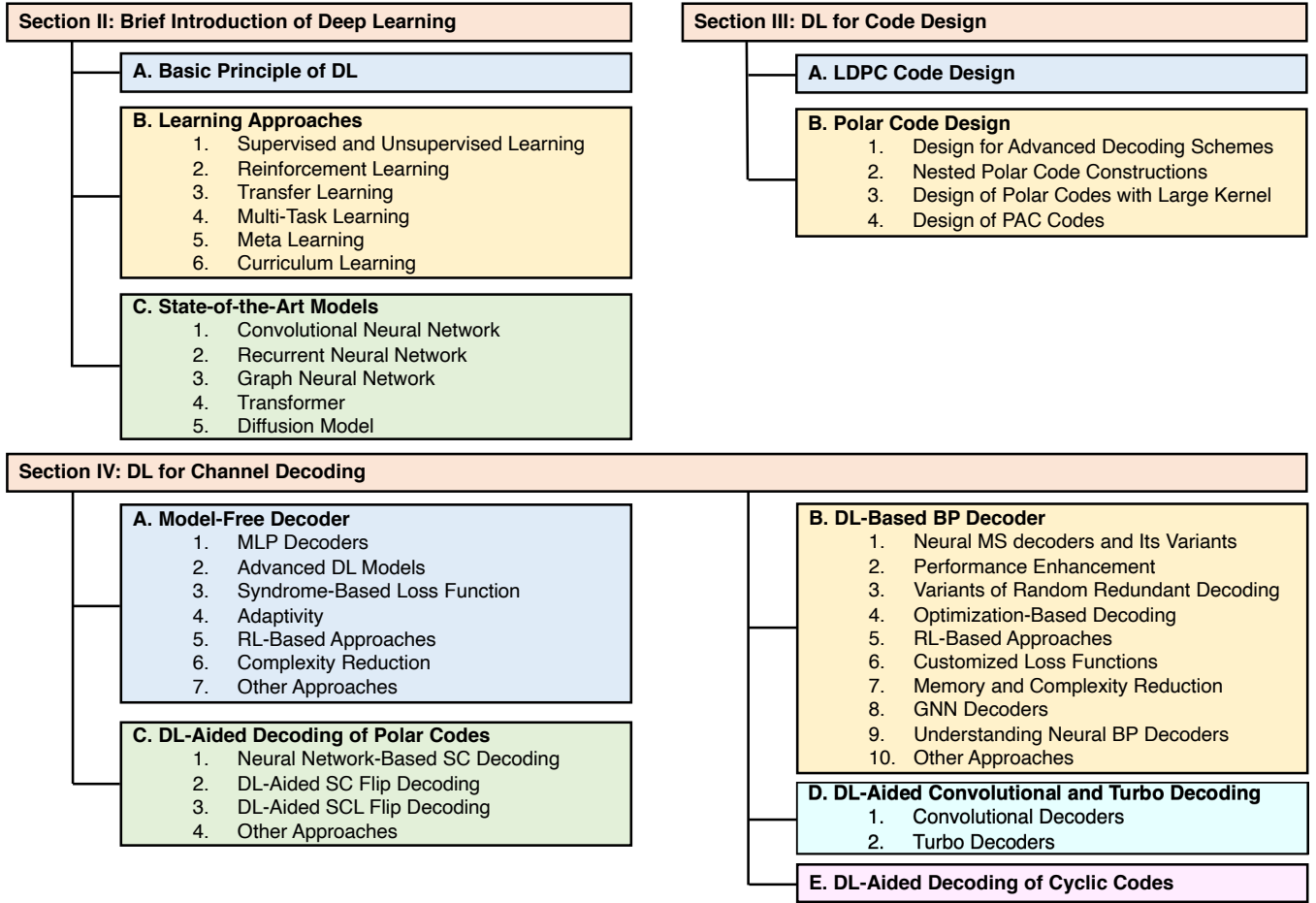
Section II: Brief Introduction of Deep Learning

A. Basic Principle of DL

B. Learning Approaches
1. Supervised and Unsupervised Learning
2. Reinforcement Learning
3. Transfer Learning
4. Multi-Task Learning
5. Meta Learning
6. Curriculum Learning

C. State-of-the-Art Models
1. Convolutional Neural Network
2. Recurrent Neural Network
3. Graph Neural Network
4. Transformer
5. Diffusion Model

Section III: DL for Code Design

A. LDPC Code Design

B. Polar Code Design
1. Design for Advanced Decoding Schemes
2. Nested Polar Code Constructions
3. Design of Polar Codes with Large Kernel
4. Design of PAC Codes

Section IV: DL for Channel Decoding

A. Model-Free Decoder
1. MLP Decoders
2. Advanced DL Models
3. Syndrome-Based Loss Function
4. Adaptivity
5. RL-Based Approaches
6. Complexity Reduction
7. Other Approaches

C. DL-Aided Decoding of Polar Codes
1. Neural Network-Based SC Decoding
2. DL-Aided SC Flip Decoding
3. DL-Aided SCL Flip Decoding
4. Other Approaches

B. DL-Based BP Decoder
1. Neural MS decoders and Its Variants
2. Performance Enhancement
3. Variants of Random Redundant Decoding
4. Optimization-Based Decoding
5. RL-Based Approaches
6. Customized Loss Functions
7. Memory and Complexity Reduction
8. GNN Decoders
9. Understanding Neural BP Decoders
10. Other Approaches

D. DL-Aided Convolutional and Turbo Decoding
1. Convolutional Decoders
2. Turbo Decoders

E. DL-Aided Decoding of Cyclic Codes

Fig. 1. Organization of this survey paper.

then introduce the state-of-the-art training and DL models. For details on the theory of general DL techniques, please refer, e.g., to [91]–[100].

A. Basic Principle of DL

DL is a subfield of ML that uses DNNs with multiple hidden layers between input and output layers. Among various neural network structures, multi-layer perceptron (MLP) is a class of fully connected feedforward neural networks that consist of at least one hidden layer in addition to input and output layers [101].

An example of a single hidden layer MLP is shown in Fig. 2, where the input vector $\mathbf{x} \in \mathbb{R}^3$ is mapped to the output vector $\mathbf{y} \in \mathbb{R}^2$ by applying a series of affine transformations and nonlinear activation functions as

$$\mathbf{y} = \phi_1(\mathbf{W}_1\mathbf{h} + \mathbf{b}_1) \tag{1}$$
$$= \phi_1(\mathbf{W}_1\phi_0(\mathbf{W}_0\mathbf{x} + \mathbf{b}_0) + \mathbf{b}_1), \tag{2}$$

where $\mathbf{W}_0 \in \mathbb{R}^{3\times3}$ and $\mathbf{W}_1 \in \mathbb{R}^{3\times2}$ are weight matrices, $\mathbf{b}_0 \in \mathbb{R}^3$ and $\mathbf{b}_1 \in \mathbb{R}^2$ are bias terms, and $\phi_i(\cdot)$ with $i \in \{0, 1\}$ denotes the element-wise application of a nonlinear activation function.

The nonlinear activation function allows the neural network to approximate highly complex functions, and the choice of
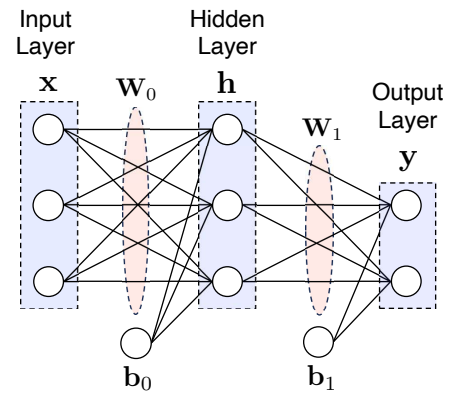
Fig. 2. An example of MLP with single hidden layer.

activation functions has a significant impact on the resulting performance. Although there are a number of activation functions [102], one of the most widely used modern activation functions is rectified linear unit (ReLU) [103] and its variants such as Leaky ReLU [104] and parametric ReLU [105].

As for the optimization of the DNN parameters, i.e., $\Theta \triangleq \{\mathbf{W}_0, \mathbf{W}_1, \mathbf{b}_0, \mathbf{b}_1\}$ in our example, the most common approach is gradient descent, which is a first-order iterative

algorithm for finding a local minimum of a differentiable function. The basic idea of gradient descent is to update the parameters in the opposite direction of the gradient of the differentiable loss function that we wish to minimize. Letting $f(\Theta)$ denote the loss function that is differentiable with respect to the parameter set Θ, in the i-th iteration of gradient descent, the trainable parameters are updated as

$$\Theta_{i+1} = \Theta_i - \eta \nabla f(\Theta_i), \quad (3)$$

where $\eta \in \mathbb{R}_+$ is a learning rate that determines the size of the steps taken in the direction of steepest descent.

Gradient descent algorithms can be classified according to the amount of data used to compute the gradient, namely, batch gradient descent (BGD), stochastic gradient descent (SGD), and mini-batch stochastic gradient descent (MBSGD). BGD calculates the gradients for the entire training dataset, while SGD performs a parameter update (and thus gradient calculation) for each training data sample. Meanwhile, MBSGD substitutes small data batches for single samples in SGD, thereby reducing the variance of the parameter updates, which can lead to more stable convergence. Furthermore, the gradient computation in MBSGD can be efficiently performed by DL libraries. For these reasons, MBSGD is one of the most popular stochastic optimization methods for training DNNs.

However, the standard MBSGD does not necessarily guarantee good convergence, and many improvements have been proposed that adaptively control the learning rate during training. These approaches include Momentum [106], Adagrad [107], Adadelta [108], RMSprop [3], Nadam [109], and Adam [110]. Implementations of these optimizers are available in DL frameworks such as Tensorflow [38] and Pytorch [39]. For more details on SGD algorithms, see [111].

B. Learning Approaches

There are many training methods for ML techniques. In the following, we introduce some of the major approaches that are often applied to the design of communication systems.

1) Supervised and Unsupervised Learning: Supervised learning trains algorithms based on labeled datasets consisting of pairs of inputs and corresponding correct outputs, i.e., ground truth. The goal is to analyze patterns from a large dataset and predict outcomes for new data. Supervised learning is commonly used for tasks such as classification and regression. Since the channel decoding problem can be seen as a type of classification, the simplest approach would be to train a DL-based channel decoder, where a DNN is trained to estimate a transmitted codeword by minimizing the error between the correct and estimated codewords.

Unsupervised learning is another type of ML algorithm that learns patterns from data without human supervision. Self-supervised learning, often used to train large language models (LLMs), can also be considered unsupervised learning in the sense that it uses the data itself to generate supervising signals, rather than relying on human supervision. There is also an approach called semi-supervised learning, which

---

³Originally proposed in http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.
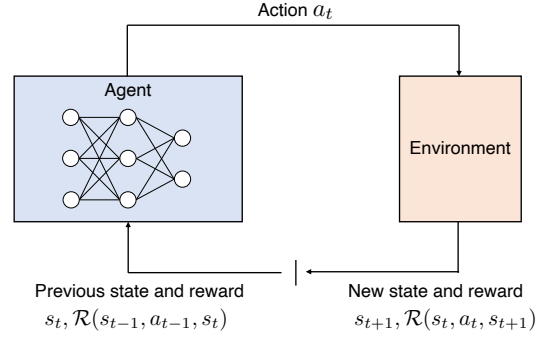

Fig. 3. A typical reinforcement learning framework.

combines both supervised learning and unsupervised learning, i.e., it uses both labeled and unlabeled data. Although these approaches are particularly suitable for the practical scenario where a sufficiently large labeled dataset is not available, their applications to channel decoding are rather new topics to be investigated.

2) Reinforcement Learning: Reinforcement learning (RL) is an experience-driven autonomous learning framework where an intelligent agent learns to take actions in a dynamic environment in order to maximize the cumulative reward [112]. RL is typically modeled as Markov decision process (MDP) which consists of

- a set of environment and agent states $\mathcal{S}$
- a set of actions $\mathcal{A}$
- the transition probability from state s_t to state s_{t+1} upon action a_t at time t, denoted by $\mathcal{P}(s_{t+1}|s_t, a_t)$, and the corresponding reward function $\mathcal{R}(s_t, a_t, s_{t+1})$.

The process is shown in Fig. 3. At each time step t, the agent observes a state $s_t \in \mathcal{S}$ and takes an action $a_t \in \mathcal{A}$, following a policy $\pi(a_t|s_t)$. Then the agent receives a scalar reward r_t, and transitions to the next state s_{t+1}, according to the reward function $\mathcal{R}(s_t, a_t, s_{t+1})$ and state transition probability $\mathcal{P}(s_{t+1}|s_t, a_t)$, respectively. This process continues until the agent reaches a terminal state. The return is the discounted, accumulated reward with the discount factor $\gamma \in (0, 1]$. The agent aims to maximize the expectation of such long term return from each state.

In many practical problems, the states of the MDP are high-dimensional and difficult to solve with traditional RL algorithms. On the other hand, thanks to their powerful function approximation properties, the use of DNNs to approximate the optimal policy and/or the optimal value functions in RL provides an efficient way to overcome these problems. This approach, called deep reinforcement learning (DRL), has achieved remarkable results in a variety of research areas. In particular, deep Q-network (DQN) [113], where a DNN model is built to approximate the Q function (the value of an action in a given state), showed impressive results in Atari [113]. More details on DRL can be found, for example, in [114]–[117].

3) Transfer Learning: Transfer learning is a technique for transferring knowledge learned from a task in a source domain to imp performance on a related task in a target domain [118]–[122]. Transfer learning addresses the problem of insufficient
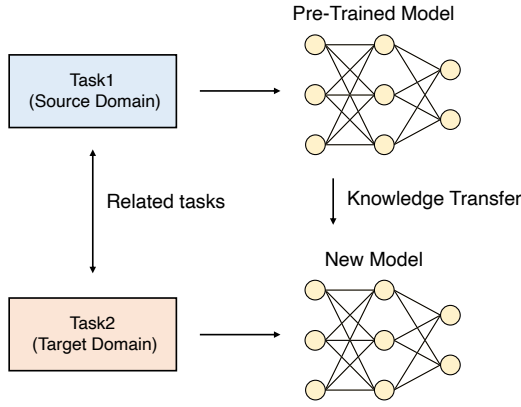
Fig. 4. The concept of transfer learning.

labeled training data by transferring the knowledge across task domains. This concept is illustrated in Fig. 4. Note, however, that the transferred knowledge may be worthless if there is little or even nothing in common between the source and target domains.

In the channel coding problems, transfer learning can be useful to adapt DL-based code design and decoding trained for a certain channel model and code parameters to new channel models or parameters. For example, we usually train a DNN for design or decoding assuming a certain code rate, and adapting to a new code rate requires re-training of the DNN, which is time consuming and computationally expensive. Since codes derived from a single mother code by rate matching, i.e., puncturing and shortening, may have many similarities, transfer learning could be used to significantly reduce the computational burden of re-training or to improve the performance with the new parameter.

4) Multi-Task Learning: In multi-task learning, a set of multiple tasks is solved jointly, sharing an inductive bias among them [123], [124]. Multi-task learning is inherently a multi-objective problem because different tasks may conflict, requiring a trade-off. A common compromise is to optimize a proxy objective that minimizes a weighted linear combination of per-task losses. Since this joint representation must capture useful features across all tasks, multi-task learning can hinder individual task performance if the different tasks seek conflicting representations, i.e., the gradients of different tasks point in opposing directions or differ significantly in magnitude. This phenomenon is commonly known as negative transfer.

There are some similarities between transfer learning and multi-task learning. Both aim to improve learners' performance by knowledge transfer. On the other hand, the main difference is that the former transfers the knowledge contained in the related domains, while the latter transfers the knowledge by learning some related tasks simultaneously. In other words, multi-task learning pays equal attention to each task, while transfer learning pays more attention to the target task than to the source task.

Similar to transfer learning, multi-task learning could be used to efficiently support multiple different code parameters. Furthermore, multi-task learning can be used for multi-

objective optimization, which is common in many communication system design problems. Application examples of multi-task learning include AE-based constellation design that attempts to jointly minimize bit error rate (BER) and peak-to-average power ratio (PAPR) for orthogonal frequency division multiplexing (OFDM) systems [125].

5) Meta-Learning: Unlike traditional learning approaches that attempt to solve tasks from scratch with a fixed algorithm, meta-learning aims to learn the learning algorithm itself by learning from previous experience or tasks [126], [127]. This *learning-to-learn* framework can lead to several benefits, such as improved data and computational efficiency.

In a meta-learning framework, there are two types of data, a larger data set of examples from related tasks (meta-training data) and a small training data set for a new task (meta-testing data). Standard meta-learning consists of two phases, 1) meta-training where a set of hyperparameters is optimized given the meta-training data set, and 2) meta-testing where model parameters, which are initialized with the meta-trained hyperparameters, are optimized using the meta-testing data. Thus, the meta-training phase aims to optimize hyperparameters that allow efficient training on a new, *a priori* unknown, target task in the meta-testing phase.

Meta-learning could naturally be applied to adaptive decoder design, where the decoder parameters are initialized by meta-training and then optimized based on meta-testing to adapt to a new channel. In addition, the concept can be applied to a wide range of problems in the physical layer, such as signal demodulation, joint transmitter and receiver optimization via end-to-end learning, channel prediction, and so forth, as reviewed in [128].

6) Curriculum Learning: Curriculum learning, originally proposed in [129], is a training strategy that trains a machine learning model from easier data to harder data, imitating human learning [130], [131]. The basic idea is to "start small" [132], i.e., to train the ML model with easier data subsets, and then gradually increase the difficulty level of the data until the entire training dataset is used. Curriculum learning can be seen as a special form of the continuation method which is a general strategy for global optimization of non-convex functions [129].

As the idea of curriculum learning serves as a general training strategy, it has been exploited in a considerable range of applications. In contrast to the standard training approach based on random data shuffling, curriculum learning can provide performance improvements with faster training convergence speed. In curriculum learning, the design of a curriculum strategy, i.e., a sequence of training criteria, plays a key role.

Curriculum learning can be beneficial for designing communication systems including channel coding schemes. For example, designing and decoding long codes is generally a more difficult task than designing and decoding short codes. As such, instead of learning to design or decode long codes directly, curriculum learning strategies that start from training with short codes and then gradually increase the code length have the potential to achieve not only faster training convergence, but also better performance.
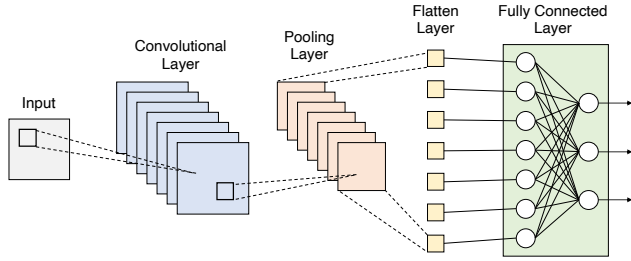
Fig. 5. A simple CNN architecture.

C. State-of-the-Art Models

Finding an appropriate architecture is important when applying DL to the problems in communication systems. In the following, we briefly review the representative DL models that are useful for such problems.

1) Convolutional Neural Network: Convolutional neural networks (CNNs) have made tremendous success in a vast research field including computer vision and natural language processing (NLP) [133]–[135]. A CNN is a type of feed-forward neural network with a convolutional layer that learns features by applying filters (or kernels) to data. A simple example of a CNN architecture is illustrated in Fig. 5. The CNN has fewer connections and parameters than the MLP since each neuron in a convolutional layer receives input only from a restricted area of the previous layer. This restricted area is called the receptive field of the neuron, and in the case of a fully connected layer, the receptive field corresponds to the entire previous layer.

CNNs have been particularly successful in image recognition tasks, achieving state-of-the-art results on several benchmarks such as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Representative CNN models include AlexNet [136], VGGnet [137], Inception (GoogLeNet) [138], ResNet [139], and DenseNet [140]. Their success is due to their ability to capture spatial features and patterns by using a hierarchical architecture of layers that perform convolution operations and extract features at different levels of abstraction [141].

As for their applications in channel coding, they are often used for decoding, and have been demonstrated to be more efficient than standard MLP. Other popular applications of CNN in the physical layer include channel estimation in time-frequency domain for OFDM systems [142] and fully CNN receiver that replaces the conventional channel estimator, equalizer, and demapper [143].

2) Recurrent Neural Network: Unlike unidirectional feed-forward neural networks, a reccurent neural network (RNN) is a bi-directional artificial neural network that is capable of learning long-term dependencies from sequential data [92], [144]. Due to their ability to use internal state (memory) to process arbitrary input sequences, they are particularly suited for processing time-series data such as speech recognition. On the other hand, due to the recurrent connections, classical RNNs have the gradient vanishing and exploding problems, i.e., the long-term gradients may not converge and approach zero or infinity during backpropagation.
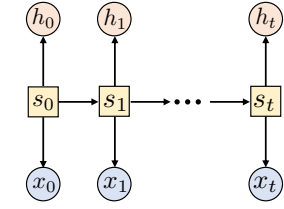
(a) RNN.
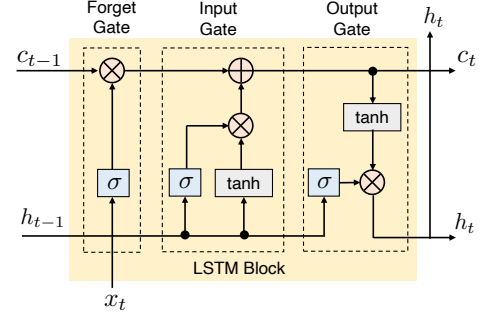
(b) LSTM with a forget gate.
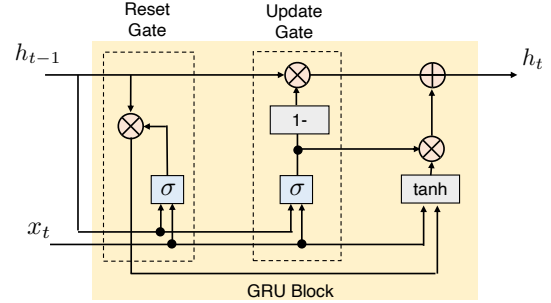
(c) GRU.

Fig. 6. Comparison of RNN, LSTM, and GRU architectures [148].

Long short term memory (LSTM) is one of the most popular RNN models that can reduce the effects of vanishing and exploding gradients [145] [146] by introducing a gating mechanism to input or forget certain features. Another popular model is a gated recurrent unit (GRU) [147], in which recurrent units adaptively capture dependencies of different time scales to accommodate the higher memory requirements of LSTM. A comparison of these architectures is shown in Fig. 6. For more details of RNN, in particular LSTM, please refer to [148]–[150].

The LSTM and GRU models have been widely applied to communication systems. In particular, due to the analogous structure to convolutional codes, RNNs may be well suited for decoding of convolutional codes. Similarly, RNNs have been successfully applied to signal detection for channels with memory [151].

3) Graph Neural Network: While ML effectively captures hidden patterns in Euclidean data, a common assumption of existing ML algorithms is that instances are independent of each other. This assumption no longer holds for graph data, where every node is related to others. Extending DNN models to non-Euclidean domains, which is generally referred to as

geometric DL, has been an emerging area of research. In particular, a graph neural network (GNN) that operates on the graph domain has recently become a popular graph analysis method [152]–[155].

Let $G \in (\mathcal{V}, \mathcal{E})$ be a graph, where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the edge set. Let $\mathcal{N}_u$ be the neighborhood of some node $u \in \mathcal{V}$. Additionally, let $\mathbf{x}_u$ be the properties of node $u \in \mathcal{V}$. GNN implements a permutation-equivalent layer, called a GNN layer, which maps a representation of a graph into an updated representation of the same graph. Although the design of GNN layers is one of the active research areas, one popular approach is message passing neural network (MPNN) layers (other popular approaches include graph convolutional networks [156] and graph attention networks [157]). In an MPNN layer in a generic GNN, nodes update their representations by aggregating the messages received from their immediate neighbors [158] and the output of the layer (node representations $\mathbf{h}_u$ for each $u \in \mathcal{V}$) is expressed as

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right), \qquad (4)$$

where ϕ and ψ are typically trainable differential functions, whereas $\bigoplus$ is a nonparametric permutation invariant aggregation operator that can take an arbitrary number of inputs. In particular, ϕ and ψ are referred to as update and message functions, respectively.

Due to its close relationship to a Tanner graph, a GNN is particularly useful for designing and decoding codes over graph. One of the major advantages of GNN is their scalability, i.e., a GNN trained for a small code length will generalize to any code length, while this usually requires additional training.

4) Transformer: Transformer is a DL architecture based on the multi-head attention mechanism [159]. As depicted in Fig. 7, it consists of an encoder and a decoder, each of which has several Transformer blocks having the same architecture. Each Transformer block consists of a multi-head attention layer, a feed-forward neural network, a shortcut connection, and a layer normalization. Given a sequence of elements, the self-attention mechanism explicitly models the dependencies among all entities of a sequence.

It has no recurrent units, and thus requires less training time than previous RNN architectures, such as LSTM, and its later variant has been widely adopted for training most of representative LLMs, such as Open AI's generative pretrained Transformer (GPT) series [160], Meta's Large Language Model Meta AI (LLaMA) [161], Googles Pathways Language Model (PaLM) [162], and Gemini [163] are based on the Transformer model. More details and the applications of the Transformer can be found in [164]–[166].

Although Transformer was originally proposed for NLP tasks [159], it has been successfully adopted for a variety of tasks such as computer vision [167], audio applications [168]. For the physical layer technologies, the use of Transformer is rather new [169]. Due to its excellent performance, Transformer has the potential to improve the performance of existing DL methods for communication engineering problems.
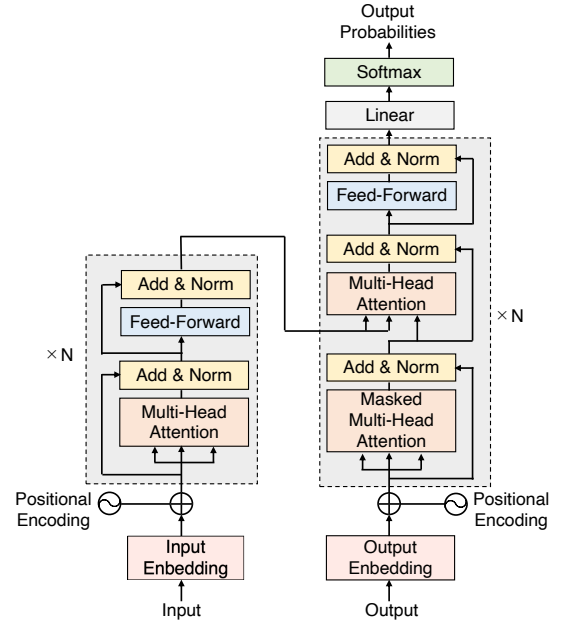

Fig. 7. The Transformer architecture [159].

5) Diffusion Model: Diffusion models (DMs) are a class of probabilistic generative models that progressively corrupt data by injecting noise, and then learn to reverse this process for sample generation. The training procedure consists of two phases: the forward diffusion process and the backward denoising process [170]. In the forward process, typically Gaussian noise is injected into the training data until it becomes pure Gaussian. In the backward process, the noise is sequentially removed to reconstruct the original image. The noise subtracted at each step is estimated by a neural network. Among different formulations [171], [172], denoising diffusion probabilistic model (DDPM) [173] is a representative DM inspired by the theory of non-equilibrium thermodynamics.

Due to its high generative quality and versatility, DM could be applied to many problems in communication systems, such as channel estimation [174], signal detection [175], AE [176]–[178], and network optimization problems [179], [180]. However, the application of DMs to the physical layer is a relatively new area of research [181]–[183].

III. DL FOR CODE DESIGN

Modern capacity-approaching codes such as LDPC and polar codes are usually designed based on well-established analytical tools, such as density evolution (DE) [193], [194] and extrinsic information transfer (EXIT) chart [195]. However, these techniques rely on assumptions that do not hold in practice. For example, for the design of LDPC codes, DE and EXIT chart analyses assume simple channel models, such as binary-input additive white Gaussian noise (BI-AWGN) channels, infinite code length, and unlimited belief propagation (BP) decoding iterations. These techniques are also used for polar code design, but they are again limited to simple channel models and decoding schemes such as successive cancellation (SC) decoding. For more realistic channel models

TABLE II
SUMMARY OF DL-BASED POLAR CODE DESIGN.

Category	Reference	Main Contributions
Advanced Decoding Schemes	Ebada et al. [184]	Design for BP decoding with finite iteration count.
	Huang et al. [185]	RL-based design of polar codes for SCL decoding.
	Leonardon et al. [186]	Design that minimizes BLER under SCL decoding via projected gradient descent.
	Liao et al. [187]	GNN-based polar code construction for CA-SCL decoding.
	Miloslavskaya et al. [188]	Optimization of polar codes with dynamic frozen bits under SCL decoding.
Nested Polar Codes	Huang et al. [185]	Construction of nested polar codes via advantage actor-critic algorithms.
	Li et al. [189]	Stochastic policy optimization by a customized network.
	Ankireddy et al. [190]	Nested polar code construction based on sequence modeling and Transformer.
Polar Codes with Large Kernel	Hebbar et al. [191]	Polar codes via large nonlinear neural network-based kernels.
PAC Codes	Mishra et al. [192]	RL-based algorithm for rate-profile construction of PAC codes.

and advanced decoding schemes, DL could replace or support existing code design techniques. We have summarized the DL-based approaches to polar code design in Table II.

A. LDPC Code Design

Irregular LDPC codes are characterized by a variable degree distribution $\lambda(x)$ and a check degree distribution $\rho(x)$, which are expressed as

$$\lambda(x) = \sum_{i=2}^{d_\mathrm{v}} \lambda_i x^{i-1}, \ \rho(x) = \sum_{i=2}^{d_\mathrm{c}} \rho_i x^{i-1}, \qquad (5)$$

where λ_i and ρ_i represent the fraction of edges emanating from variable nodes (VNs) and check nodes (CNs) of degree i and $\lambda(1) = \rho(1) = 1$. The maximum variable degree and check degree are denoted by d_v and d_c, respectively. The degree distribution is often optimized to maximize the iterative decoding threshold, which is defined as the lowest channel signal-to-noise ratio (SNR) at which the message distribution in BP evolves in such a way that its associated error probability converges to zero as the number of iterations tends to infinity. The method of identifying a threshold by tracking the evolution of the message distribution is termed DE [193].

The code design problem belongs to the class of nonlinear constraint satisfaction problems with continuous space parameters, where we first explore the space of degree distributions to find degree distribution pairs, traditionally solved by differential evolution [193], and then evaluate the BP threshold of the selected pairs via DE. In [196], the authors modeled the code design process as a supervised learning problem by mapping the recursive update equation of DE to an RNN architecture, which they refer to as neural density evolution (NDE). They also proposed a multi-objective loss function for NDE that ensures its high configurability, i.e., various code rates and maximum degrees. Their simulations show that the proposed designs achieve the performance of state-of-the-art designs in asymptotic settings for a variety of codeword lengths and channels.

B. Polar Code Design

An encoder of polar codes of length N is represented by the generator matrix $\mathbf{G}_N = \left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix} \right)^{\otimes n} \in \mathbb{F}_2^{N \times N}$, where $n = \log_2 N$ is a positive integer and $\mathbf{A}^{\otimes n} = \mathbf{A} \otimes \mathbf{A}^{\otimes(n-1)}$ is the nth Kronecker power of the matrix $\mathbf{A}$ [5]. Let $\mathcal{I} \subset$

$\{0, 1, \ldots, N-1\}$ denote a set of information bit indices with its cardinality $K = |\mathcal{I}|$, and $\mathcal{F} = \{0, 1, \ldots, N-1\} \backslash \mathcal{I}$ denote the complement of $\mathcal{I}$ with its cardinality $|\mathcal{F}| = N - K$. Letting $\mathbf{u} = (u_0, u_1, \ldots, u_{N-1}) \in \mathbb{F}_2^N$ be an input vector to the polar encoder, the bits u_i with $i \in \mathcal{I}$ are chosen to carry information, whereas those with $i \in \mathcal{F}$ are frozen (i.e., fixed to a predetermined bit value known by encoder and decoder). The code rate of the polar code is thus $R = K/N$. Polar code design or construction is equivalent to identifying an appropriate index set $\mathcal{I}$ for a given channel model and decoding scheme.

In his original paper, Arıkan suggested using Monte-Carlo simulations to estimate the reliabilities of bit channels [5]. Subsequently, DE [194] and its improved version [197] were proposed to accurately estimate the reliabilities at the cost of high complexity. This complexity was alleviated by Gaussian approximation (GA) of DE [198], improved GA [199], and reciprocal channel approximation (RCA) [200].

1) Design for Advanced Decoding Schemes: Although polar codes were originally proposed with SC decoding [5], their finite-length performance is unsatisfactory. In order for polar codes to achieve performance comparable to other capacity-approaching codes, advanced decoding schemes, such as successive cancellation list (SCL) or cyclic redundancy check-aided successive cancellation list (CA-SCL) decoding [201], are required. However, the above-mentioned polar code construction schemes assume SC decoding and there is no explicit approach to designing polar codes for SCL and CA-SCL decoding.

In [184], the authors proposed to design polar codes for BP decoding with limited number of iterations over both additive white Gaussian noise (AWGN) and Rayleigh fading channels. By representing the frozen and non-frozen bit vectors by soft-valued vectors, which can be considered as training weights of a neural network, training is performed to minimize the cross entropy loss between transmitted and estimated codewords while satisfying the target code rate requirement and training convergence. The simulations showed that the learned polar code outperforms the performance of the 5G polar code under Arıkan's conventional BP decoder.

On the other hand, the authors in [185], [202] proposed a genetic algorithm and RL-based design of polar codes for SCL decoding. As a reward function in RL, the authors computed an SNR required for a code to achieve a target block error rate (BLER) via Monte-Carlo simulations. In the same line

of research, the authors in [203] proposed a tabular RL-based construction of polar codes for SCL decoding. Instead of evaluating the BLER based on the Monte-Carlo method as in [185], [202], they designed the reward function that sends a negative immediate reward (penalty) to the agent when the selected action causes a frame error in genie-aided SCL decoding[4]. The proposed method achieved comparable or slightly better performance with a lower computational complexity in training than the method in [185].

As another approach, the authors in [186] proposed a two-step optimization method. More specifically, they first trained MLPs to predict BLER under SCL decoding from input frozen bit sequences, and then the code that minimizes the BLER was constructed via projected gradient descent (PGD) [204] which has been widely studied in the realm of adversarial attacks on neural networks. Simulations demonstrated that the proposed construction successfully improves the performance of the codes on the dataset used for predicting BLER.

More recently, the authors in [187] proposed a GNN-based polar code construction algorithm for CA-SCL decoding. More specifically, a polar code is first mapped onto a unique heterogeneous graph called the polar-code-construction message-passing (PCCMP) graph, and then a heterogeneous GNN-based iterative message-passing algorithm is proposed which aims to find a PCCMP graph corresponding to the polar code with minimum BLER under CA-SCL decoding. The proposed GNN-based iterative message-passing method has a salient property that a single trained model can be directly applied to constructions for different design SNRs and different block lengths without any additional training. Numerical experiments showed that the proposed constructions outperform classical constructions in [197] under CA-SCL decoding.

In [188], the authors proposed neural network-based adaptive polar coding scheme that adapts to various channel conditions and quality of service requirements. Specifically, the authors developed an MLP-based performance prediction framework for polar codes with dynamic frozen bits under SCL decoding. Then the authors presented a new class of polar codes with dynamic frozen bits parameterized by a single integer parameter, and used the performance prediction framework to optimize the parameter for a given target BLER, list size, code length and rate. The simulation results show that the proposed codes outperform 5G polar codes under CA-SCL decoding with various list sizes. Although a neural network is not used due to the training difficulty, the same authors also proposed an RL-based method to design dynamic frozen bits of polar codes that minimize the BLER under SCL decoding in [205].

2) Nested Polar Code Construction: In general, on-the-fly design of polar codes that adaptively select frozen bits for a given channel may be too complex to implement in practical systems. On the other hand, the authors in [206]–[209] studied the design of polar codes based on a universal reliability order of bit channels that is independent of channel conditions. As such, it is preferable in practice to impose a nested property
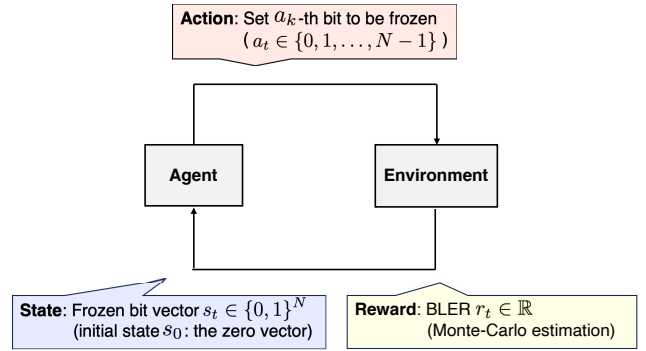

Fig. 8. RL-based construction of a rate-K/N nested polar code at time $t \in \{1, \ldots, K\}$ in [185], [202]. The goal is to minimize the expected cumulative BLER, i.e., $\sum_{t=1}^{K} r_k$. The ones in the frozen bit vector s_t indicate locations of frozen bits.

on polar codes with different rates, so that all polar codes can be derived from the same mother code based on the universal reliability sequence [210].

The authors in [185], [202] proposed constructing nested polar codes via advantage actor-critic (A2C) algorithms [211]. The paper regarded code construction as a multi-step MDP, where for a given (N, K) polar code (current state), a new action is taken to construct $(N, K+1)$ polar code (an updated state). This MDP is illustrated in Fig. 8. The reliability ordered sequence is then constructed by sequentially appending the actions to the end of initial polar code construction. The proposed code design was shown to outperform the conventional DE construction under SCL decoding.

Meanwhile, the authors in [189] first transformed the problem of nested polar code construction into a stochastic policy optimization problem for sequential decision, and then represented the policy by a customized neural network. Furthermore, the authors proposed a gradient-based algorithm to minimize the average loss of the policy. Simulation results demonstrate that the proposed construction achieves better performance than the state-of-the-art nested polar codes for SCL decoding in [185], [202]. A similar construction of nested polar codes has also been proposed in [190], where the authors parameterized the policy network by a Transformer encoder-only model, which can directly predict the next information bit in the nested sequence. It was shown that the proposed Transformer-based construction can achieve better error rate performance than the approach proposed in [189].

3) Design of Polar Codes with Large Kernel: Another way to improve finite-length performance of binary polar codes is to increase the size of the polarization kernel. In fact, channel polarization holds for all kernels provided that they are not unitary and not upper triangular under any column permutation [212]. However, binary polar codes with large kernels exhibit poor performance for practical short-to-medium block lengths and face an exponential increase in computational complexity with kernel size.

In [191], the authors proposed polar codes via large non-linear neural network-based kernels, termed as DEEPPOLAR, and its decoder based on a generalization of SC decoding. They also developed a principled curriculum-based training

---

[4]In genie-aided SCL decoding, the decoder can always output the correct codeword if it is in the list.

methodology that allows DEEPPOLAR to generalize well to high SNR scenarios, characterized by rare error events. It was shown that the DEEPPOLAR outperforms the classical polar codes with SC decoding.

4) Design of Polarization Adjusted Convolutional Codes: In the Shannon Lecture at the 2019 International Symposium on Information Theory (ISIT), Arıkan introduced a new class of codes, called polarization adjusted convolutional (PAC) codes, that concatenate convolutional precoding with the polar transform [213]. PAC codes significantly improve the performance of polar codes at short-to-moderate block lengths, where channel polarization occurs relatively slowly.

The first step in encoding of PAC codes is rate profiling. For PAC codes of rate-K/N, this step inserts K information bits into a vector of length N, which is subsequently input to the convolutional precoder. The selection of K bit indices out of N possible indices is called rate profile construction and its design significantly affects the performance of PAC codes.

In [192], the authors proposed an RL-based algorithm for rate-profile construction of PAC codes. Specifically, by mapping the rate profile construction problem to MDP, the authors proposed Q-learning with a set of customized reward and update strategies. Simulation results showed that the proposed rate-profile construction provides better error rate performance compared to the Monte-Carlo-based rate profiling design in [214].

IV. DL FOR CHANNEL DECODING

DL methods for channel decoding have been an active area of research and have been extensively studied as a means to replace or assist conventional decoding algorithms. In what follows, we first review model-free decoders that do not assume specific code structure and thus are applicable to any codes. We then review model-based DL BP decoding methods take take into account specific factor graphs. Subsequently, we focus on DL methods for decoding polar codes, convolutional/turbo codes, and cyclic codes.

A. Model-Free Decoders

A model-free decoder employs neural networks that do not take into account any specific structure of the codes and thus can potentially benefit from the powerful architectures of advanced DL models. However, such decoders typically suffer from the curse of dimensionality, since the size of the training dataset grows exponentially with the number of information bits. On the other hand, a potential advantage over conventional non-DL-based decoding is a highly parallelizable structure, allowing *one-shot* decoding instead of iterative decoding. The model-free approaches that we will discuss below is summarized in Table III.

1) MLP Decoders: The paper [215] is one of the initial works on DL-based channel decoding where the authors investigated the direct application of MLP to decoding of random and polar codes. Their empirical results demonstrated that for structured codes, the DL decoder can generalize even for codewords not seen in the training phase, and the DL decoder can achieve maximum a posterior (MAP) decoding

performance for a very small code lengths such as 16 bits, but learning for longer codes is prohibitively complex due to the exponentially increasing training complexity.

Meanwhile, the authors in [216] investigated the impact of various configurations of an MLP decoder on BER performance, such as the number of hidden layers, the number of nodes for each layer, and activation functions. Similarly, the paper [217] empirically studied the impact of the number of hidden layers and nodes as well as a training SNR of the MLP decoder on its performance and investigated the minimum numbers required to achieve similar performance to the optimal maximum-likelihood decoder for short linear and nonlinear block codes.

In [218], [243], the authors investigated the application of small MLP decoders to low-energy and low-latency applications. In particular, the paper made comparisons between single-label and multi-label neural decoders, and demonstrated that the multi-label decoder generally requires more hidden layers and nodes to achieve similar performance to the single-label decoder.

2) Advanced DL Models: The performance of the above-mentioned MLP decoder can be enhanced by advanced DL models. For instance, in [219], the authors investigated different types of decoders based on MLP, CNN, and RNN (in particular, LSTM). Their empirical results demonstrated that the RNN and CNN decoders can actually achieve better BER performance than the MLP decoder at the cost of higher computation time. In [220], RNN (bidirectional gated recurrent unit (Bi-GRU)) was used for encoding/decoding of turbo codes as in [244].

To further improve the performances, the authors of [221], [245] introduced the concept of residual learning [139] to the MLP, CNN, and RNN-based decoders. Specifically, the paper introduced a denoiser network prior to the decoder that simply aims to remove noise induced at the channel, and proposed a training loss that considers both denoising and decoding performance. It was demonstrated that the proposed denoiser network can improve the BER performance at the cost of marginal increase in run time.

More recently, inspired by the success of the Transformer model in various applications [159], a novel Transformer architecture for decoding algebraic block codes, termed error correction code Transformer (ECCT) was proposed in [222]. The ECCT takes as input a concatenation of reliabilities of codeword bits (absolute values of received symbols in the case of BI-AWGN) and syndrome bits as its input where each element of which is embedded in a high-dimensional space with its own position-dependent embedding vector. Then, a self-attention mechanism is employed, where the interaction between bits specified by the code structure, i.e., parity check matrix (PCM), is incorporated as domain knowledge. Extensive simulation results demonstrated that the proposed Transformer-based decoder outperforms state-of-the-art neural decoders.

Although the ECCT in [222] employs a mask matrix that is derived from the PCM, there exist numerous PCMs for the same code which will lead to different decoding performances. Motivated by this fact, the authors in [246] addressed the

TABLE III
SUMMARY OF MODEL-FREE DECODERS.

Category	Reference	Main Contributions
MLP Decoders	Gruber et al. [215]	Initial work on MLP-based channel decoding.
	Seo et al. [216]	Investigation on the impact of various configurations of an MLP decoder.
	Leung et al. [217]	Empirical study on the impact of hyperparameters of the MLP decoder.
	Leung et al. [218]	Investigated small MLP for applications with low energy and latency.
Advanced DL Models	Lyu et al. [219]	Investigation on different types of DL decoders, namely, MLP, CNN, RNN.
	Sattiraju et al. [220]	Bi-GRU-based decoder.
	Zhu et al. [221]	Residual MLP, CNN, RNN-based decoders.
	Choukroun et al. [222]	Novel Transformer architecture for decoding block codes.
	Choukroun et al. [223]	DDPM for soft-decision decoding of linear codes.
Syndrome-Based Loss Function	Bennatan et al. [224]	Syndrome-based approach to soft-decision decoding of linear codes.
	Kamassury et al. [225]	Iterative algorithm, referred to as iterative error decimation.
	Artemasov et al. [226]	SISO decoder based on Stacked-GRU for turbo product codes.
Adaptability	Wang et al. [227]	Unified DL-based decoder for polar and LDPC codes.
	Jiang et al. [228]	A meta learning-based model independent neural decoder.
	Lee et al. [229]	Transfer learning for decoding a set of rate-compatible polar codes.
	Artemasov et al. [230]	A unified DL decoder for BCH and polar codes concatenated with CRC.
RL-Based Approach	Carpi et al. [231]	DQN for iterative bit-flipping decoding of binary linear codes.
	Gao et al. [232]	Q-learning-based bit-flipping decoding for polar codes.
Complexity Reduction	Kavvousanos et al. [233]–[236]	Magnitude-based pruning and quantization for parameter reduction.
	Cavarec et al. [237]	A DL-aided adaptation of the order parameter in OSD.
Other Approaches	Raviv et al. [238]	Data-driven framework for permutation selection in permutation decoding.
	Kurmukova et al. [239]	Friendly jamming for improving decoding performance.
	Tsvieli et al. [240]	Investigation on the problem of maximizing the margin of the decoder.
	Zhong et al. [241], [242]	DL-based decoders for spin-torque transfer magnetic random access memory.

problem of identifying the optimal PCM. In particular, the authors proposed a systematic mask matrix constructed from the systematic PCM which results in sparse self-attention map, and proposed a novel Transformer architecture called a double-masked ECCT that consists of two parallel masked self-attention blocks employing distinct mask matrices.

Meanwhile, in [223], the authors employed DDPM [173] for soft decoding of linear codes with arbitrary block lengths. Their framework models the transmission over the AWGN channel as a series of diffusion steps that can be iteratively reversed. The paper also proposed to condition the diffusion decoder on the number of parity check errors and to employ a line-search procedure to control the reverse diffusion step size.

Furthermore, in [247], the authors proposed a foundation model [248] for channel codes by extending the Transformer architecture. A foundation model refers to a model that is initially trained on a wide range of data, generally based on self-supervision, and then adapted (e.g., transferred or fine-tuned) to a wide range of downstream tasks. Thus, the proposed framework provided a universal decoder that is capable of adapting and generalizing to any (unseen) code of any length.

3) Syndrome-Based Loss Function: Syndrome decoding is a well-known approach for decoding algebraic codes. Several approaches have been proposed for training DL-based syndrome decoders that estimate the transmitted codeword from the syndrome. Syndrome-based training does not rely on the knowledge of the transmitted codeword, and is thus promising for online adaptation to changing channel conditions.

The paper [224] is one of the early works on syndrome-based DL decoding, which proposed to use the absolute values of the received symbols, i.e., reliabilities in the case of the BI-AWGN channel, and the syndrome of its hard decisions for
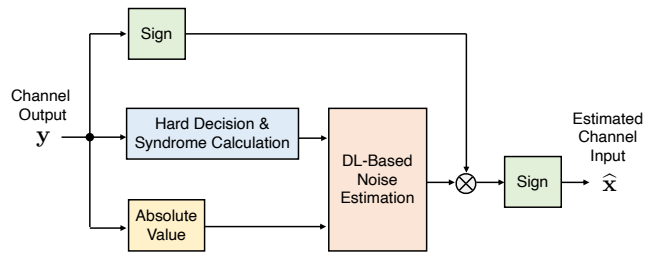

Fig. 9. The syndrome-based DL decoder proposed in [224]. The channel input sequence $\mathbf{x}$ consists of BPSK symbols, and $\mathbf{y}$ is the output of a BISO channel.

decoding, instead of directly using the received symbols as the input to the decoder. The proposed decoder is illustrated in Fig. 9. Furthermore, the authors introduced permutations from the code's automorphism group [249], [250] as a pre-processing. Permutations in this group have the property that the permuted version of any codeword is guaranteed to be also a valid codeword, i.e., the permuted input of the decoder is a noisy valid codeword. Simulations demonstrated that the proposed framework can achieve near maximum-likelihood performance for short Bose–Chaudhuri–Hocquenghem (BCH) codes.

The syndrome-based approach, which attempts to predict the error vector from its syndrome alone, can suffer from the potential presence of inconsistent training examples, called *disturbance* [251], i.e., training examples with the same syndromes but with different error vectors. To solve this problem, the authors in [225] proposed an iterative algorithm, referred to as iterative error decimation, which is robust against the superposition of error patterns. In each iteration, the DL decoder estimates the error vector and then decimates (subtracts) it from the received vector. The simulation results demonstrated

that the proposed scheme improves the performance of the scheme in [224].

Furthermore, the authors in [226] proposed a syndrome-based approach to soft-input soft-output (SISO) decoding of BCH component codes in turbo product codes [252] based on Stacked-GRU, which is an RNN architecture composed of GRU. They introduced a regularization term into a loss function and demonstrated that the proposed DL decoder outperforms the original chase decoder in [252].

4) Adaptability: Adaptive coding is a technique for adapting a code rate to a channel condition in wireless channels. For a DL-based decoder, supporting multiple code rates not only requires multiple training, which is computationally intensive, but also requires a large amount of memory to store the learned DL parameters.

To address this issue, a unified DL-based decoder for polar and LDPC codes was proposed in [227], which supports different codes, i.e., polar and LDPC codes, with a single DNN by utilizing a code indicator at the decoder input. The simulation results demonstrated the potential of the unified decoder for very short code lengths, e.g., 16 bits. A similar unified DL decoder using the code indicator was also proposed in [230] for BCH and CRC-concatenated polar codes. Their results demonstrated that, for a code length of 64, the proposed unified decoding scheme with code indicator achieves a small performance gap of less than 0.2 dB from the decoder trained solely for the single code.

Meanwhile, the paper [228] introduced the meta-learning-based neural decoder, termed as model independent neural decoder (MIND), which can adapt to a new channel with a small number of pilots and few gradient descent steps. Specifically, the proposed approach consists of meta-learning and meta-training steps, where the model learns good initial parameters in the meta-learning step, and then adapts the parameters to the observed channel in the meta-testing phase using minimal adaptation data and pilots. It has been demonstrated that the proposed scheme can adapt to a channel while achieving a performance close to that of a DL decoder designed solely for the particular channel. In the same line of research, the authors in [229] proposed transfer learning to efficiently train decoders for a set of rate-compatible polar codes that are expurgated from the same mother code as in 5G NR.

5) RL-Based Approaches: Although the majority of previous works on DL-based channel decoding are based on supervised learning, several RL-based approaches have been proposed for the cases where supervisory data (ground truth) is unavailable.

The paper [231] is one of the earliest works on RL-based channel decoding. Unlike these studies, the authors in [231] proposed an RL framework for iterative bit flipping (BF) decoding of binary linear codes. Specifically, they linked the BF decision at each step to MDP and applied RL to find good decision strategies. The authors also exploited the permutation automorphism group to improve the performance. The extensive simulations showed that the learned BF decoders with DQN can achieve near-optimal performance for short, high-rate codes.

Later, the authors in [232] applied RL-based BF decoding to polar codes. In contrast to the DQN proposed in [231], they used simple Q-learning and attempted to map channel observations directly onto estimated codewords. Simulation results showed that the proposed Q-learning achieves comparable performance to the learned BF decoding in [231] with lower complexity.

6) Complexity Reduction: In general, decoding longer codes requires a larger neural network size. Such a network not only requires a huge amount of computational resources in the training phase, but also imposes high computational and space complexities in the inference phase. In particular, the complexity in the inference phase is of practical importance since the training is usually performed offline.

The authors in [233] attempted to reduce the parameter size of a DL decoder, by introducing various simplified neural network structures with fewer parameters. In [234], the same authors further extended their work by introducing the magnitude-based pruning [253] and quantization of the parameters. The proposed decoder was demonstrated to achieve similar performance to the original system in [224] even with 80% reduction of the network parameters and 8-bit fixed-point representation. Furthermore, FPGA implementation of the proposed decoder has been presented in [235], [236].

Meanwhile, the authors in [237] considered DL-aided complexity reduction of ordered statistic decoding (OSD) [254], which is a soft-decision decoding algorithm of linear block codes that approaches the optimal maximum-likelihood decoding performance especially for short codes. Although increasing the order parameter of OSD leads to near-maximum-likelihood performance, it may waste computational resources when the received signal can be decoded with lower order. The paper [237] proposed a learning-based approach to adapt the required order parameter to the channel condition and demonstrated the effectiveness of the proposed scheme in terms of the performance-complexity trade-off through numerical simulations.

7) Other Approaches: It is known that multiple decoding attempts over different permutations of received codewords provide a performance gain [250], [255]. However, it remains unclear how to choose the permutation that yields the best performance. To address this, the authors in [238] presented a DL approach to selecting candidates from the code's automorphism group in permutation decoding. In this scheme, a trained network predicts the probability of successful decoding for each permutation, and decoding is performed only for permuted codewords with the highest probabilities. The proposed algorithm has been demonstrated by simulations to achieve remarkable performance gains over a random selection of permutations from the automorphism group.

In [239], the authors proposed a novel approach referred to as *friendly attack* for improving channel decoding performance, inspired by the concept of adversarial attacks. The proposed scheme introduces small perturbations into the modulated symbols before transmission. The perturbations are designed by a modified iterative fast gradient method [256] such that a loss function between the decoded codeword and the transmitted codeword is minimized. The performance

improvement by the proposed scheme has been demonstrated for various codes and decoders.

Although the use of DL for channel decoding has been experimentally validated, the theoretical justification for the developed algorithm in terms of, e.g. the generalization properties, remains challenging. The authors in [240] addressed the problem of maximizing the margin of the decoder for an additive noise channel whose noise distribution is unknown, as well as for a nonlinear channel with AWGN. They formulated a maximum margin optimization problem, which is common in support vector machines (SVMs), for the decoder learning problem, and they relaxed it to a regularized loss minimization (RLM) problem by several approximation steps. The paper then provided expected generalization error bounds for both models, under optimal choice of the regularization parameter. The paper also presented a theoretical guidance for choosing the training SNR based on the bound for the additive noise channel.
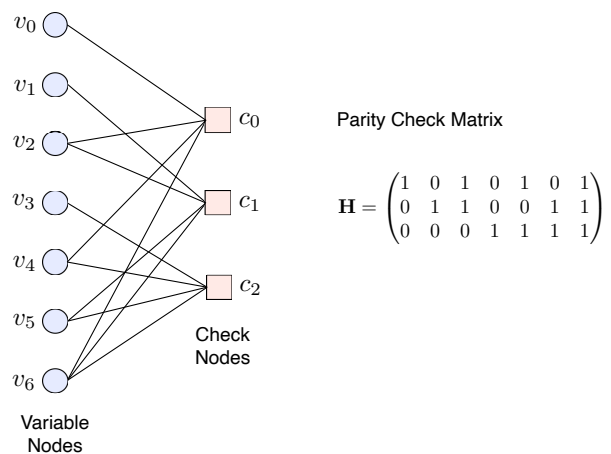
B. DL-Aided BP Decoding

BP decoding is an efficient iterative decoding algorithm that is commonly used for decoding LDPC codes. BP decoding is performed on Tanner graph consisting of CNs and VNs, which correspond to codeword bits and parity check equations, respectively. An example of PCM with the corresponding Tanner graph representation of a (7, 4) Hamming code is shown in Fig. 10a. In BP decoding, decoding messages are iteratively updated at CNs and VNs based on the Bayes' rule. In practice, the min-sum approximation [304] is applied to the CN updates to reduce complexity, and this decoding is referred to as min-sum (MS) decoding. The performance loss due to the min-sum approximation can be compensated for by normalized min-sum (NMS) and offset min-sum (OMS) decoders [305] at the cost of slightly increased complexity.

In [257], the authors proposed a DL-based implementation of BP decoding by treating BP decoding as a differentiable process, where the decoding messages are passed through unrolled iterations in a feed-forward fashion. Additionally, trainable weights were introduced at the edges, which are then optimized via SGD. An example of the unrolled BP trellis for a (7, 4) Hamming code is illustrated in Fig. 10b. For code length N, the number of edges E, and the number of iterations L, the unfolded trellis has N neurons at the input and output layers, and E neurons at the $2L$ hidden layers. The network architecture is a non-fully connected neural network. As we review below, the trainable BP decoder over the unfolded trellis has been extensively studied in the literature. We summarize these works in Table IV.

Note that the idea of unfolding an iterative algorithm into a structure analogous to a neural network, i.e., deep unfolding [306], is a common model-based DL approach [307] often considered in the design of communication systems. Besides channel decoding, the idea of deep unfolding has been successfully applied, for example, to MIMO signal detection and channel estimation [45], [62], [308].

1) Neural MS Decoders and Its Variants: As mentioned above, the paper [257] was the initial work that applied a

(a) PCM and corresponding Tanner graph.
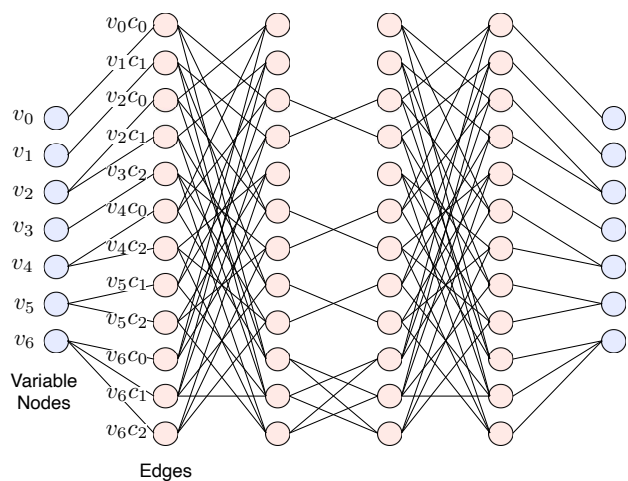
(b) Unrolled BP trellis (two iterations).

Fig. 10. An example of deep unfolded BP decoder for (7, 4) Hamming codes.

feedforward network to BP decoding, where trainable weights are assigned to the edges of the factor graph, which are then optimized via SGD over the unrolled iterative BP decoding. The proposed parameterized decoder can compensate for the effect of small cycles in the Tanner graph of the code by properly scaling the weights. The effectiveness of the proposed decoder has been demonstrated for short BCH codes, for which standard BP decoding does not work well due to many short cycles in the graph. Later, in [309], the same authors extended the work by introducing an RNN architecture and showed that this architecture leads to comparable performance to the feedforward architecture in [257] even with fewer parameters. In [258], the authors also proposed neural network-based NMS and OMS decoders for reducing the complexity of the BP decoder, which are the generalized versions of the standard NMS and OMS decoders [305]. Neural network-based OMS decoding has also been studied independently in [259]. Another approach to enhance the performance of MS decoding while preserving the low complexity property was considered in [263]; the authors proposed min-sum decoding

TABLE IV
SUMMARY OF DL-AIDED BP DECODING.

Category	Reference	Main Contribution
Neural MS Decoding and Its Variants	Nachmani et al. [257], [258]	DL-aided BP, NMS, and OMS decoding.
	Lugosch et al. [259]	NOMS decoding.
	Dai et al. [260]	Neural network-aided OMS and NOMS decoding.
	Yu et al. [261]	Neural AMS decoding.
	Hsu et al. [262]	Neural network-aided VWMS decoding.
	Wu et al. [263]	Neural MS decoding with linear approximation for PB-LDPC codes.
	Kim et al. [264]	Neural SCMS decoding.
Performance Enhancement	Teng et al. [265]–[267]	CNN-based learned BF for BP.
	Sun et al. [268]	LSTM-based learned BF for BP.
Variants of Random Redundant Decoding (RRD)	Nachmani et al. [258]	mRRD decoding with RNN-based BP decoders.
	Liu et al. [269]	Node-classified redundant decoding algorithm.
Optimization-Based Decoding	Wei et al. [270]	Trainable ADMM-penalized decoder.
	Wadayama et al. [271]	Trainable PGD decoder for LDPC codes.
	Wadayama et al. [272]	Proximal decoding for LDPC codes.
RL-Based Approach	Doan et al. [273]	RL-based selection of permutations on factor-graph.
	Habib et al. [274]–[276]	RL-based scheduling optimization for sequential BP decoding.
Customized Loss Function	Lugosch et al. [277]	Soft syndrome as a loss function for training a neural BP decoder.
	Teng et al. [278]	New syndrome losses for syndrome-based DL decoding of polar codes.
	Nachmani et al. [279]	New loss function based on sparse node and knowledge distillation losses.
Memory/Complexity Reduction	Teng et al. [280]	Weight quantization mechanism for an RNN polar decoder.
	Ibrahim et al. [281]	Quantization of an RNN polar decoder.
	Xiao et al. [282], [283]	Finite alphabet iterative decoders for LDPC codes via quantized RNN.
	Lyu et al. [284]	A joint optimization of message quantization and quantization thresholds.
	Lian et al. [285], [286]	Weight-sharing across edges based on scalar parameters.
	Wang et al. [287], [288]	A parameter sharing scheme within the same layer for a neural NMS decoder.
	Dai et al. [289]	A weight-sharing scheme for a neural MS decoder of protograph LDPC codes.
	Liang et al. [290], [291]	Tensor-train and tensor-ring decompositions for parameter size reduction.
	Cheng et al. [292]	A weight-sharing scheme for adapting to multiple code rates.
	Buchberger et al. [293]	A novel pruning-based neural BP decoder for short linear block codes.
	Buchberger et al. [294]	A neural BP with decimation.
GNN Decoders	Satorras et al. [295]	A hybrid inference model that combines BP and GNN.
	Cammerer et al. [296]	A fully GNN-based decoder.
	Tian et al. [297]	An edge-weighted GNN decoder.
Understanding Neural BP Decoders	Ankireddy et al. [298]	Empirical study on how the learned weights attenuate the effect of these cycles.
	Adiga et al. [299]	Theoretical study on the generalization capabilities of neural BP decoders.
Other Approaches	Clausius et al. [300]	GNN-based joint equalization and decoding.
	Wiesmayr et al. [301]	Deep-unfolded interleaved detection and decoding for MIMO wireless systems.
	Lee et al. [302]	Learning-aided multi-round BP decoding with impulsive perturbation.
	Wang et al. [303]	DL detection of decodable codewords for reducing the decoding delay.

with linear approximation (LAMS) for protograph-based low-density parity-check (PB-LDPC) codes, where the magnitudes of the check node updating and channel output are linearly adjusted by a small and shallow neural network. In contrast to the above-mentioned studies that considered the flooding schedule, the authors in [310] proposed a neural network-aided normalized offset min-sum (NOMS) decoding for the layered BP with application to 5G LDPC codes.

As another variant of MS decoding, the adjusted min-sum (AMS) decoding proposed by Qualcomm [311] has drawn attention, where it employs look-up tables (LUTs) to simplify nonlinear CN processing. In [261], the authors introduced a neural network based selection mechanism to AMS decoding that automatically selects the check node updating rule from either the MS rule or the BP rule and demonstrated that the proposed decoder outperforms the conventional neural NMS decoder. The single-minimum min-sum (SMMS) algorithm [312] is another variant of MS decoding that simplifies the CN update in the MS algorithm. Specifically, in SMMS decoding, only one minimum magnitude is calculated at each CN over all the CN inputs and a correction is applied to outgoing messages if required. The SMMS decoder can be further improved by the variable weight min-sum (VWMS) algorithm [313], which

introduces variable correction factors into the CN update that depend on the number of iterations. To efficiently learn the optimal correction factors in the VWMS algorithm, the authors in [262] proposed a neural network-aided approach, instead of an exhaustive search in the original work [313]. The effectiveness of the proposed scheme in terms of throughput was demonstrated experimentally using the 40 nm CMOS TSMC process. Unlike the above-mentioned methods, which simplify CN updates, self-corrected min-sum (SCMS) decoding [314] modifies the VN processing by deleting unreliable messages. More specifically, in SCMS decoding, any variable node message that changes sign between two consecutive iterations is discarded, i.e., set to zero. The authors in [264] introduced trainable normalization and offset weights to the SCMS decoder, which are trained by DL techniques. It was demonstrated that the error rate performance of the proposed neural SCMS decoder is close to that of the BP decoding.

Although the above-mentioned works applied neural BP decoding to LDPC codes, it has also been used to decode polar codes. In [315], similar to [257], a neural MS decoder was applied to factor graphs of polar codes. The proposed decoder was demonstrated by simulations to outperform conventional BP decoding with the same number of iterations. Also, the

authors presented an efficient hardware implementation of the basic computation block of the proposed decoder. In [316], a similar trainable BP decoding was applied to sparse graphs of polar codes [317], which was shown to achieve comparable performance to BP decoding even with a single trainable parameter. Furthermore, the authors in [260] proposed an NOMS decoder for polar codes that introduces both normalization (or scaling) factors and offsets, and demonstrated that the proposed decoder achieves better performance than the state-of-the-art schemes, including the decoder proposed in [315].

In order to enhance the performance of a standalone polar code and close the performance gap from CA-SCL decoding with lower latency, the authors in [318] proposed neural BP decoding for polar codes concatenated with a CRC code by exploiting the concatenated factor graph of the polar code and CRC, while the conventional BP decoding for concatenated CRC-polar codes is applied only to the factor graph of polar codes and the CRC is used only to verify the result of BP at each iteration. Furthermore, the authors in [319] considered concatenated polar and LDPC codes and proposed two-dimensional OMS decoding. They optimized the trainable parameters of the decoder by back propagation over the unfolded BP trellis and showed that the performance of the proposed decoder is comparable to the exact BP decoder.

2) Performance Enhancement: To enhance BP decoding of polar codes, the authors in [265]–[267] proposed to combine BP decoding with a CNN-assisted bit flipping mechanism, which performs the flipping bit selection in the BP-BF decoder [320] based on a CNN trained using the metadata of the BP decoding process. The authors demonstrated that the proposed scheme can achieve a lower BLER than SCL decoding. Meanwhile, in order to reduce the computational complexity of the CNN-based approach, the paper [268] proposed an LSTM network that predicts error-prone bits to be flipped based on the magnitude of log-likelihood ratio (LLR) after the original BP decoding.

The authors in [321] introduced a hypernetwork [322] that generates weights of a neural BP decoder to make the decoder more adaptive by letting the weights be a function of the node's input. The same authors also introduced hypernetworks for decoding short polar codes [323] and showed that the proposed decoder achieves similar BER as SCL decoding in the high SNR region. Furthermore, they proposed an autoregressive BP decoder that incorporates the estimated SNR and multiple autoregressive signals obtained from the intermediate output of the network [324].

Training data preparation is an essential part of training DNN-based decoders. In particular, the choice of training SNR plays an important role in training a DL-based channel decoder for generalization. A common approach is to train the decoder over varying SNR ranges [257]. Besides, the optimal choice of the training SNR has been studied either empirically [215], [244] or analytically [325]. To address the problem of choosing the optimal training SNR, the authors in [326] proposed *active deep decoding*, inspired by active learning [327]. Specifically, based on the observation that no optimal training SNR for all validation sets exists, the paper proposed to adaptively sample training data instead of passively generating examples during training. It was demonstrated that this active deep decoding scheme offers performance gain by effectively sampling the training data without increasing the inference (decoding) complexity.

In [328], inspired by ensemble models that are widely used to solve complex tasks by decomposing them into multiple simpler tasks each of which is solved locally by a single expert member of the ensemble [329], the authors introduced the ensemble of neural BP decoders. The proposed scheme consists of a single classical hard-decision decoder (HDD) and multiple trainable BP decoders, where the classical HDD is employed to assign a received codeword to a single expert BP decoder based on the number of the estimated codeword errors. It was demonstrated that this scheme achieves remarkable performance gains over the single neural BP. Furthermore, the data-driven ensemble scheme has been extended to BP polar decoders in [330], [331].

Meanwhile, many practical LDPC codes exhibit an error floor[5]. For applications such as ultra-reliable and low-latency communications that require extremely low BLER, it can be critical to mitigate the error floor. Since the error floor of LDPC codes is commonly attributed to the suboptimality of the iterative message passing decoding algorithms for factor graphs with cycles, the paper [332] proposed training methods for neural NMS decoders to eliminate the error floor of LDPC codes. Specifically, inspired by the boosting learning technique [333], the authors divided the decoder into two cascaded neural decoders and trained the first decoder to improve the waterfall performance, while the second decoder was trained to handle the residual errors that are not corrected by the first decoder.

3) Variants of Random Redundant Decoding: The random redundant decoding (RRD) algorithm [334] and multiple-bases belief-propagation (MBBP) decoder [335] are other approaches to soft-decision decoding of short block codes based on a redundant PCM. modified random redundant decoding (mRRD) [250] is an algorithm that attempts to benefit from both RRD and MBBP decoding, which make use of a permutation group (automorphism group) of codes and parallel iterative decoders, respectively.

In [269], the authors proposed a node-classified redundant decoding (NC-RD) algorithm for high-density parity-check (HDPC) codes in order to improve the performance of RRD decoding. The NC-RD algorithm introduces two preprocessing steps to the RRD decoding. More specifically, the algorithm first classifies the variable nodes of the parity-check matrix by the k-median algorithm based on the number of shortest cycles associated with each variable node, and then generates a list of permutations of bit positions from the automorphism group based on the permutation reliability metrics. The authors further proposed the neural network-based NC-RD algorithm by unfolding the NC-RD decoding process and introducing trainable weights.

In [258], the authors applied the concept of DL-based BP decoding to mRRD [250] by replacing the BP decoding blocks in the mRRD algorithm with the proposed RNN-based BP

---

[5]For modern iteratively decodable codes, such as LDPC codes and turbo codes, there is an SNR point after which the error rate decreases only slowly. This phenomenon is called *error floor*.
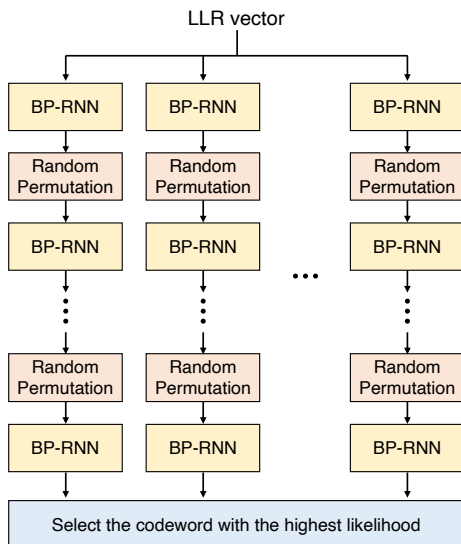
LLR vector

Fig. 11. The mRRD algorithm with RNN-based BP decoders in [258].

decoders. The resulting decoder structure is shown in Fig. 11. The proposed RNN-based mRRD decoder has been demonstrated to achieve near maximum-likelihood performance with reasonable computational complexity.

4) Optimization-Based Decoding: Optimization-based decoding is another research direction aimed at improving the performance of BP decoders. The origin of the optimization-based decoding goes back to the work by Feldman who introduced a linear programming (LP) formulation of decoding LDPC codes [336].

The LP decoder is based on the LP relaxation of the original maximum-likelihood decoding problem [337]. However, the LP decoder has higher computational complexity and worse error-correcting performance in the low SNR region compared with the BP decoder. In order to address the above drawbacks, the paper [270] proposed a trainable alternating direction method of multipliers (ADMM)-penalized decoder [338] by unfolding the ADMM iterations. It was demonstrated that the proposed decoder can outperform the conventional BP decoder in high SNR region with comparable execution time.

Afterwords, the authors in [271] introduced a trainable projected gradient decoder for LDPC codes by unfolding the PGD algorithm and optimizing the parameters via backpropagation. The proposed decoder alternately performs the gradient and projection steps, where the former moves in the direction of the negative gradient of the objective function, while the latter maps the search point into a feasible region that nearly satisfies the optimization constraint. Also, in [272], the same authors proposed proximal decoding of LDPC codes based on the proximal gradient method [339], which is used for solving non-differentiable convex optimization problems.

5) RL-Based Approaches: It is known that parallel BP decoders on independently permuted factor graphs can significantly improve the performance of single BP decoding for polar codes [340], [341]. In [273], the authors addressed the problem of selecting the permutations on the factor graph that lead to successful decoding given a channel observation.

Specifically, they viewed the selection of permutations as a multi-armed bandit problem and proposed an RL-based CRC-aided BP decoder that attempts to select the best set of permutations. The proposed scheme was shown to achieve better performance than other approaches such as cyclically-shifted and random factor-graph permutations [340], [342].

In [274]–[276], the authors proposed a novel RL-based sequential BP decoding scheme to optimize the scheduling of CN clusters for moderate length LDPC codes[6]. In the proposed scheme, m CNs are divided into sets of z CNs, called *cluster*, and the scheduling problem, i.e., cluster selection with $\lceil m/z \rceil$ possible actions, was optimized by Q-learning. Furthermore, they proposed novel meta-learning based sequential decoding schemes to quickly adapt to changing channel conditions due to fading in wireless scenarios. The RL-based scheduling of sequential BP decoding has also been proposed for generalized LDPC codes [343], where the authors showed that the proposed RL-based decoding scheme was shown to significantly outperform the standard BP flooding decoder, as well as a sequential decoder based on random scheduling with the smaller number of CN updates.

6) Customized Loss Function: As in Section IV-A3, a syndrome loss function has been used for DL-based BP decoders. In [277], the authors proposed a soft syndrome as a loss function for training a neural BP decoder. Unlike the paper [224], which utilizes the hard syndrome as an input to the decoder, this paper introduced the *soft* syndrome which is defined similarly to the CN update rule in MS decoding, in addition to the conventional cross entropy loss function.

While the application of [277] was limited to decoders that output a soft estimate of the codeword, this is not the case for polar decoders that do not use a PCM. To address this, the authors in [278] proposed two modified syndrome losses: frozen-bit syndrome loss and CRC-enabled syndrome loss. The authors also introduced a syndrome-enabled blind equalizer based on the proposed syndrome loss, which does not require the transmission of training sequences.

Unlike the above syndrome-based approaches, the authors in [279] considered a linear combination of sparse node loss and knowledge distillation loss, in addition to the conventional cross entropy loss. Knowledge distillation is a technique in DL where one uses a teacher network to guide the training of a smaller student neural network [344]. Sparse node loss imposes a sparse constraint on the node activations based on the L_p norm, whereas the knowledge distillation loss aims to mimic the teacher network, which was the standard MS decoder without trainable parameters, by transferring knowledge. It was shown that the proposed loss terms provide BER performance improvement of up to 1.1 dB without increasing the runtime complexity and the model size.

7) Memory and Complexity Reduction: A neural network-based BP decoder introduces different weights (or scaling factors) to different edges in the Tanner graph, which can significantly increase the computational and space complexities of standard BP decoding. In fact, this issue has been

[6]In contrast to the standard flooding scheduling where all CNs and VNs are updated simultaneously at each iteration, sequential BP decoding, or layered decoding, updates nodes or sets of nodes individually in sequence.

(a) Quantization of weights.

(b) Sharing weights (connections with the same color have the same weight).

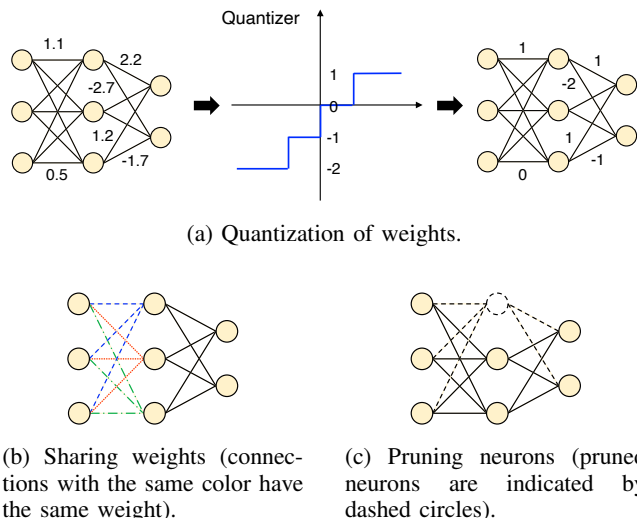(c) Pruning neurons (pruned neurons are indicated by dashed circles).

Fig. 12. Approaches to reducing computational and space complexities of a DL decoder.

studied extensively for generic DNNs, i.e., not limited to channel decoding applications, due to the large parameter sizes of modern DL models [345]. A popular approach is neural network pruning [346]–[350], which aims to remove redundant parameters of an original network while preserving the accuracy. Parameter shaping is also an effective way to reduce parameters by sharing parameters between different neurons. Parameter shaping is typically exploited in CNN, where all neurons in a particular feature map share the same weight. Another popular approach is parameter quantization [351], [352]. These major approaches to addressing the complexity problem are illustrated in Fig. 12.

(a) Quantization: In [280], the authors proposed a weight quantization mechanism for an RNN polar decoder. Specifically, they proposed a two-step approach, where floating-point weights are quantized into 2^q quantization levels and then they are further compressed into 2^c ($c < q$, $c, q \in \mathbb{N}$) quantization levels, which are the most commonly used among 2^q quantization levels. The quantization of the RNN polar decoder was also studied in [281], where the authors demonstrated that quantization after training leads to better performance compared to the case where quantization is applied after every epoch during training.

Instead of quantizing DNN decoder parameters, several papers have investigated quantizing decoding messages or LLR values computed from channel observations. For example, in [353], the authors trained a parameterized quantization of LLR values that maximizes the performance of BP decoding. Similarly, the authors in [354] investigated the design of quantizers in an LDPC decoder that are used for quantizing both LLRs and iterative decoding messages. On the other hand, the authors in [355] proposed to train neural BP decoder for the system with one-bit quantizer.

Unlike existing studies that consider the AWGN channel, the authors in [282], [283], [356] considered BSC and proposed finite precision decoders, called finite alphabet iterative decoder (FAID), for LDPC codes with recurrent quantized neural

network (RQNN). More specifically, they proposed the BER as the loss function to train the RQNNs over BSC by leveraging straight-through estimator (STE) [357] to overcome the issue of gradients vanishing caused by the low precision activations in the RQNN and quantization in the BER.

In [284], the authors proposed a joint optimization of quantized message alphabets and quantization thresholds. Specifically, the authors utilized the softmax distribution [358] to make the quantization thresholds trainable by softening the one-hot distribution of the quantization. The proposed decoder was shown to outperform the original non-surjective FAIDs [359] in terms of error rate performance.

(b) Weight Sharing: In [285], [286], the authors proposed simple-scaling models for weighted BP decoding in [257] that share weights across edges, using only three scalar parameters per iteration: message weight, channel weight, and damping factor. The authors showed that such simple scaling models are often sufficient to achieve gains similar to the fully parameterized decoder.

The authors in [287], [288] proposed a parameter sharing scheme for a neural NMS decoder that shares the same correction (normalization) factors in the same layer. In contrast, the authors in [360], [361] proposed a family of weight sharing schemes for a neural NMS decoder that uses the same weight for edges with the same check node degree and/or variable node degree. Similarly, the authors in [289] proposed a neural MS decoder for protograph LDPC codes where a bundle of edges derived from the same edge type share identical parameters. Due to the lifting structure of protograph LDPC codes, the same set of parameters can be employed for multiple codes derived from the same base code.

In [290], the authors applied the tensor-train (TT) decomposition [362] to a neural NMS decoder, where it decomposes a high-order tensor into several low-order tensors. This not only reduces the number of weight parameters, but also the number of multiplications required in the CN and VN updates. Furthermore, in [291], the same authors proposed tensor-ring (TR) decomposition [363] combined with weight sharing to further reduce the storage and computational complexity.

In [292], a weight sharing scheme was proposed for a neural BP or MS decoder to adapt to multiple code rates with a reasonable amount of parameters. Specifically, instead of training different decoders, they proposed to train a single rate-compatible decoder based on multi-task learning, where different parts of the parameters are activated to handle different code rates.

(c) Pruning: In [293], the authors proposed a novel pruning-based neural BP decoder for short linear block codes. The key idea was to prune unimportant CNs with small weights of an overcomplete PCM. Similarly, in [294], the same authors proposed a neural BP with decimation [364] for LDPC codes. In particular, they identified the least reliable VN with the aid of DL, i.e., the VN with the lowest absolute *a posteriori* LLR, and then decimated it to $\pm\infty$. It has been demonstrated that the proposed decoder with decimation can significantly outperform the conventional neural BP decoder.

8) GNN Decoders: In general, BP computes the optimal (posterior) marginal probability distributions only for a non-

loopy graphical model, and in practice it often computes a poor approximation of the true distribution. To tackle this limitation, the authors in [295] extended the standard GNN equations to factor graphs and presented a hybrid inference model that combines messages from BP and from GNN, where the GNN messages are learned to complement the BP messages.

Instead of the method in [295], which extends BP decoding by combining it with a GNN, the authors in [296] proposed a fully GNN-based decoder. In contrast to weighted BP decoding, they introduced two types of MPNN-based trainable message update functions: *the edge message update functions* and *the node update functions*. Independently, an edge-weighted GNN decoder has been proposed in [297]. In the proposed decoder, they applied an MPNN for updating messages and assigned a trainable weight to each edge message, which is optimized by a fully-connected feed-forward neural network, i.e., MLP. The major advantage of these GNN decoders over the standard neural BP decoder is that the number of trainable parameters is not affected by the code length. Therefore, after training, the trained decoder can be applied to codes with different rates and lengths without retraining.

9) Understanding Neural BP Decoders: In [298], the authors empirically showed how the learned weights mitigate the effect of short cycles in Tanner graphs to improve the reliability of the posterior LLRs and contribute to the robustness of the decoders across channels. The authors also introduced an analytical approach for finding the weights using GA and compared the neural MS decoders, showing that for complicated fading channels, the neural network-based weight optimization leads to better performance than the GA-based optimization.

The authors in [299] theoretically investigated the generalization capabilities [365] of neural BP decoders, i.e., the difference between empirical and expected BERs. The paper presented new theoretical results that bound the gap and showed its dependence on the decoder complexity, in terms of code parameters (such as message/code lengths, VN/CN degrees), decoding iterations, and the training dataset size. They empirically observed that the generalization gap increases with decoding iterations and code length, and decays with the training dataset size, supporting the theoretical results in their paper.

10) Other Approaches: To improve the decoding of short Raptor-like LDPC codes, the authors in [366] considered multi-round BP decoding with impulsive perturbation [367]. Perturbation is a process of making a small intentional change in the received signal, and this scheme iteratively performs conventional BP decoding and perturbation until a valid codeword is found. In [302], the authors proposed a neural network based perturbation symbol selection scheme where the symbols to be perturbed are selected from a pre-trained neural network and showed that the proposed scheme performs better than existing schemes such as [366] for Raptor-like LDPC codes.

The performance of a standalone neural BP decoder could be further enhanced by jointly optimizing signal detection and decoding. For instance, in [301], iterative signal detection and decoding via deep unfolding was proposed for MU-

MIMO-OFDM. In [300], the authors proposed GNN-based joint detection and decoding for inter-symbol interference (ISI) channels.

Besides the approaches introduced in Section IV-B7, the authors in [303] proposed another approach to alleviate decoding complexity and latency. Specifically, they proposed a DL approach for detecting the decodable codewords and predicting the iteration number from the received signal to reduce the decoding delay. This could potentially be useful for early feedback prediction in hybrid automatic repeat request (HARQ). Furthermore, in [368], the authors accelerated neural BP decoding through coded distributed computing [369]. In particular, they reformulated the neural BP decoding operations as matrix-vectors to facilitate distributed parallel decoding.

In addition to communication systems, DL-based decoders have also been studied for storage systems. In [242], the authors proposed DL-based decoder for spin-torque transfer magnetic random access memory (STT-MRAM) [370]. In order to adapt to the process variation and unknown offset of the resistance caused by the change in working temperature, the authors proposed an adaptive decoding scheme based on the three DNN decoders, i.e., BF, MS, and BP decoders, which share the same DNN architecture but have different weights. In [241], the same research group proposed a neural normalized offset reliability-based min-sum (RBMS) decoding algorithm for STT-MRAM by introducing trainable parameters to the RBMS algorithm [371]. It has been demonstrated that the proposed scheme can outperform the RBMS algorithm over the STT-MRAM channel, while maintaining similar decoder structure and time complexity of the standard RBMS decoder.

Neural network-based BP decoding has been studied not only for classical error-correcting codes, but also for quantum error-correcting codes. For example, neural BP decoding has been applied to quantum LDPC codes [372] for which standard BP decoding may be insufficient due to the error degeneracy feature of quantum error-correcting codes [373]. By designing the loss function to account for error degeneracy, the decoding accuracy was improved up to three orders of magnitude compared to the standard BP decoder without training. Neural BP decoding for quantum LDPC codes was also studied in [374], [375].

C. DL-Aided Decoding of Polar Codes

Model-free decoders in Section IV-A as well as neural BP decoders in Section IV-B are easily applicable to polar codes. In the following, we also review methods for designing model-free decoders that take the specific code structure into account. Furthermore, we focus on DL approaches that augment conventional SC or SCL decoders, instead of replacing them with a DNN. In Table V, we provide the summary of these methods.

1) Neural Network-Based SC Decoding: Although the straightforward application of DNN is a viable option for decoding polar codes as in [215], the major issue was the exponential growth of training complexity. In [376], the authors addressed this issue by introducing partitioned neural network (PNN) decoders. More specifically, inspired by the

TABLE V
SUMMARY OF DL-AIDED POLAR DECODERS.

Category	Reference	Main Contribution
Neural Network-Based SC Decoding	Cammerer et al. [376], [377]	Partitioned neural network decoders.
	Doan et al. [378]	Neural successive cancellation (NSC) decoding.
	Wodiany et al. [379]	Efficient implementation of an low-precision NSC decoder.
	Hebbar et al. [380]	Novel curriculum learning-based sequential neural decoder.
DL-Aided SCF Decoding	Wang et al. [381]	LSTM network that estimates the first erroneous bit.
	He et al. [382]	LSTM-based identification of erroneous bits for DSCF decoding.
	Doan et al. [383], [384]	Neural DSCF with trainable bit-flipping metric.
	Wang et al. [385]	Q-learning-assisted SCF decoding algorithm.
	Doan et al. [386]	RL-based bit-flipping strategy for fast SC decoding.
DL-Aided SCLF Decoding	Hashemi et al. [387]	Trainable bit-flipping metric for SCL decoding.
	Doan et al. [388]	FSCLF decoding algorithm.
	Chen et al. [389]	LSTM-assisted bit-flipping algorithm for a CA-SCL decoder.
	Tao et al. [390]	New flip algorithm based on DNC.
	Liang et al. [391]	Stacked LSTM to improve the accuracy of erroneous bit prediction.
	Li et al. [392]	Approximated bit-flipping metric for DSCLF decoding.
Other Approaches	Lu et al. [393]	DL-aided shifting metric for SCL decoding.
	Liu et al. [394]	CRC error-correction aided SCL decoding.

simplified successive cancellation algorithm in [395], which divides the decoding tree into single parity checks (SPCs) and repetition codes (RCs)), they replaced the SPC and RC subdecoders by neural networks. Simulations showed that the PNN decoder achieves similar BER performance to the SC and BP decoders for short lengths, such as 128 bits, with potentially much lower latency. A similar concept of partitioning a neural network-based decoder for polar codes was also investigated in [377]. Furthermore, in order to reduce the latency of PNN, neural successive cancellation (NSC) decoding of polar codes was proposed in [378], where multiple constituent neural network decoders are incorporated into SC decoding, and its efficient implementation based on a low-precision neural network decoder was studied in [379].

Recently, another approach to tackle the difficulty of learning to decode long polar codes was proposed in [380], where a novel curriculum learning-based sequential neural decoder for polar and PAC codes was proposed. The paper designed a novel curriculum to train RNN, where the problem of joint estimation of information bits is decomposed into a sequence of sub-problems of increasing difficulty. The proposed decoder was shown to achieve better BER performance than the conventional supervised training without curriculum and standard SC decoding.

Instead of completely replacing a conventional decoder with DNNs, DL-based approaches that support conventional decoders such as SC and SCL decoding have been extensively studied. Among them, DL-assisted successive cancellation flip (SCF) decoding [396] is one of the most popular approaches, which will be reviewed in the following.

2) DL-Aided SC Flip Decoding: Despite its low-complexity, the error correcting performance of SC decoding at finite block lengths is not comparable to other modern codes such as LDPC codes. In order to improve the finite block length performance, SCF decoding has been proposed in [396] inspired by the fact that the first erroneous bit decision in SC decoding has a detrimental impact on the resulting error rate. The SCF decoder first performs standard SC decoding to generate a first estimated codeword, and if the codeword passes the CRC, decoding is complete. If the CRC check fails,
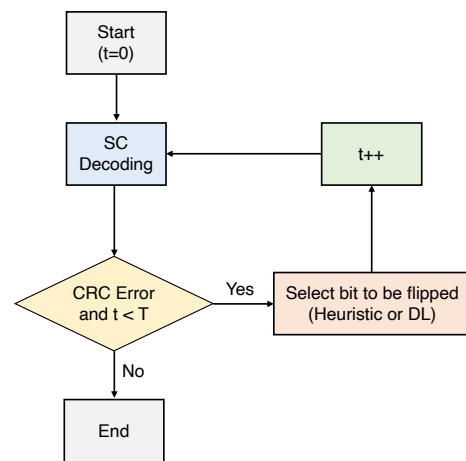

Fig. 13. Flowchart of an SCF decoding framework with the number of trials T.

the SCF decoding makes T additional attempts to identify the first error in the codeword. In each attempt, a single estimated codeword bit is flipped with respect to the initial decision. The algorithm terminates when a valid codeword has been found or when all T attempts have been made. The SCF decoding procedures is shown in Fig. 13. SCF decoding retains the $O(N)$ memory complexity of the original SC algorithm and has an average computational complexity that is practically $O(N \log N)$ at high SNR, while still providing a significant gain in terms of error correcting performance. While SCF decoding is limited to correcting a single erroneous bit in the codeword, dynamic successive cancellation flip (DSCF) decoding [397], [398] is a generalization of SCF-based decoding that is able to correct higher-order erroneous information bits by dynamically updating the set of flipping bit indices after each decoding attempt.

The common challenge in SCF and DSCF decoding is how to identify the first error bit that causes error propagation. In the original work [396], the estimated codeword bits with the smallest amplitudes of LLR are flipped, but they are not necessarily the first errors. In fact, the optimal bit flipping

strategy is still an open problem due to the lack of a rigorous mathematical characterization. Furthermore, DSCF decoding requires expensive exponential and logarithmic computations to compute the BF metric, which is used to determine the bit flipping position.

A popular DL-based solution is to train an LSTM network to estimate the first erroneous bit to be flipped. The authors in [381] have proposed an LSTM network for SCF decoding that takes an LLR sequence of the previous SC decoding attempt and outputs a vector where each element corresponds to the probability that a bit is the first error. Furthermore, the authors proposed a two-step training method that combines supervised learning with RL to train the LSTM to reverse previous incorrect flips. Similarly, the authors in [382] proposed an LSTM-based error bit identification for DSCF decoding where the network is trained to identify the first erroneous bit and additional erroneous bits by supervised learning and RL, respectively.

There are other learning approaches that do not rely on model-free DNNs. In [383], [384], the authors proposed neural DSCF decoding where they properly approximated and introduced trainable parameters to the BF metric and optimized its parameter by RMSProp, a variant of the SGD optimization technique. In [385], the authors proposed a Q-learning-assisted SCF decoding algorithm that selects the candidate flipping bits through the learned Q-table instead of metric sorting. It was demonstrated that the proposed decoding algorithm is particularly effective in reducing the decoding delay caused by sorting during the decoding process without sacrificing performance. Similarly, in [386], an RL-based BF strategy is also investigated for fast SC decoding of polar codes [399], where the authors developed a new parameterized BF model based on [387] and optimized the trainable parameters using the policy gradient method.

3) DL-Aided SCL Flip Decoding: The DL-aided BF mechanism can be applied not only to SC decoding but also to SCL decoding to further improve the performance.

In [387], the authors proposed the BF metric for SCL decoding, which is expressed by a trainable correlation matrix representing the likelihood of each decoded bit. They optimized the trainable matrix using the RMSprop optimizer and demonstrated that compared to the conventional metric in [397], the proposed BF metric significantly reduces computational complexity associated with the bit metric calculation while maintaining similar error rate performance.

In [388], the authors applied BF to fast successive cancellation list (FSCL) decoding [400], [401], referred to as the fast successive cancellation list clip (FSCLF) decoding algorithm, to address the high latency problem associated with the successive cancellation list flip (SCLF) decoding algorithm. Specifically, the authors introduced a BF strategy tailored to FSCL decoding that avoids tree-traversal in the binary tree representation of SCLF to reduce the latency of the decoding process, and then derived a path selection error metric with a trainable parameter. The proposed decoder was shown to significantly reduce the average decoding latency, average complexity, and memory consumption of the SCLF decoder at the cost of slight degradation in error rate performance.

In contrast to the above approaches that do not utilize a DNN, several papers have proposed a neural network-based selection of the flipping bit position. For instance, in [389], an LSTM-assisted BF algorithm has been proposed for a CA-SCL decoder. Furthermore, the authors have used the domain knowledge to reduce the complexity and memory requirements and computational complexity for efficient hardware implementation.

The authors in [390] proposed a new flip algorithm using differentiable neural computer (DNC) [402], which can be considered as an LSTM augmented with an external memory through attention-based soft read and write mechanisms. The proposed decoding algorithm is a two-phase decoding assisted by the two DNCs, i.e., flip DNC and flip-validate DNC. The former ranks flip positions for multi-bit flipping, while the latter is used to re-select flip positions when decoding fails. Simulation results show that the proposed DNC-aided SCLF achieves better error rate performance and reduction in the number of flipping attempts compared to the LSTM-based algorithms.

The authors in [391] proposed a stacked LSTM network to improve the accuracy of erroneous bit prediction. Specifically, they trained the three models separately: the first and second models predict the positions of the first and the second erroneous bits and the third model decides whether to continue flipping. Simulation results demonstrate that their proposed algorithms outperform existing SCLF decoding algorithms in terms of BLER performance and average number of decoding attempts.

In [392], the authors proposed an approximated error metric for dynamic successive cancellation list flip (DSCLF) decoding of polar codes to improve the performance while keeping the average complexity low. To compensate for the approximation error, they introduced learnable parameters into the metric and optimized it through the custom neural network model using the RMSprop optimizer.

4) Other Approaches to Enhancing SCL Decoding: Shifted-pruning (SP) is another approach to enhance the performance of SCL decoding for polar codes [403], which aims to prevent the correct path from being eliminated from the list. In [403], it was demonstrated that the proposed SP mechanism offers remarkable performance gains over the BF approach. However, the SCL decoder with SP generally suffers from high computational complexity, due to the re-decoding attempts and the computation of the shifting metric. To alleviate this issue, the authors in [393] proposed a DL-aided shifting metric that is free from transcendental functions and can be computed on-the-fly based on the path metrics.

Another approach to improve the performance of CA-SCL decoding was proposed in [394], where the authors take advantage of the inherent error correction capability of CRC, i.e., not just for error detection. The authors performed CRC-based error correction using an LSTM network, where the LSTM network estimates the error pattern from the LLR sequence and the CRC syndrome. The proposed CRC error-correction aided SCL decoding scheme was demonstrated to outperform the error rate of the conventional CRC error-detection aided SCL decoding scheme at the same list size.

D. DL-Aided Convolutional and Turbo Decoding

DL decoders have also been applied to convolutional and turbo codes. In particular, as we review below, several DL-aided turbo decoders have been proposed in recent years. We have listed these approaches in Table VI.

1) Convolutional Decoder: Convolutional codes encode the input stream by convolving with the generator polynomial, which can be efficiently implemented by shift registers. Convolutional codes can be represented by a time-invariant trellis, which allows efficient maximum-likelihood decoding based on the well-known Viterbi algorithm [408].

In order to improve the performance of iterative threshold decoding (ITD) [409], the authors in [404] proposed a DNN decoder for convolutional codes by unfolding a high-order recurrent neural network (HORNN) decoder [410]. The unfolded HORNN can be seen as a feedforward DNN whose parameters are trained by backpropagation with MBSGD. It was shown that with proper optimization of the parameters, the proposed decoder outperforms the conventional ITD and achieves performance close to maximum-likelihood decoding.

2) Turbo Decoder: Turbo codes, also known as parallel concatenated convolutional codes [3], consist of two (usually identical) recursive systematic convolutional (RSC) codes concatenated in parallel and a bit interleaver. Turbo codes are typically decoded by iterative decoding between two constituent SISO decoders where one constituent decoder computes posterior probabilities based on the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [411], and then passes them to the other decoder. The turbo decoder significantly improves the error-correction performance by iteratively exchanging extrinsic information between the two constituent SISO decoders.

In [244], the authors proposed a DL-based BCJR decoder for an RSC code based on bi-GRU, referred to as NEURAL-BCJR, and then extended it to a turbo decoder by replacing the component SISO decoder with the proposed NEURALBCJR decoder. Simulations demonstrated that the proposed scheme is particularly beneficial for non-Gaussian channels, such as t-distributed noise. Later, a DL-aided turbo decoder, termed DEEPTURBO, was introduced in [405]. They also used bi-GRUs to replace the conventional SISO decoders as in [244], but the authors in [405] trained different bi-GRU weights across different iterations, whereas the authors in [244] shared the same weight for all bi-GRU blocks. This enables a fully end-to-end training without imitating the BCJR algorithm. Furthermore, DEEPTURBO increased the number of posterior LLR values exchanged between the two decoders to expedite iterative decoding. Extensive simulations have demonstrated that DEEPTURBO exhibits an improved reliability, adaptivity, and lower error floor compared to NEURALBCJR.

The aforementioned model-free turbo decoders, i.e., NEU-RALBCJR [244] and DEEPTURBO [405], are illustrated in Fig. 14, where the constituent SISO decoders of the turbo decoder are replaced with Bi-GRUs without considering the specific trellis structure of the RSC encoders. In contrast, the authors in [406] proposed a novel model-driven decoder architecture, called TurboNet, which integrates DNN into the traditional max-log-MAP algorithm. Furthermore, they applied network pruning to TurboNet to effectively reduce the number

(a) Turbo encoder.

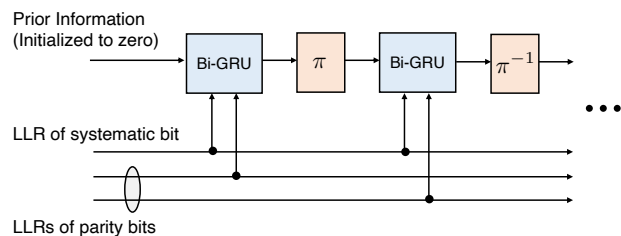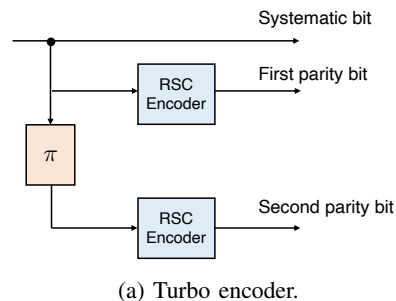(b) Model-free DL turbo decoders based on Bi-GRUs.

Fig. 14. An encoder and DL-based decoder of turbo codes. In the NEURAL-BCJR decoder [244], each Bi-GRU is pre-trained to imitate BCJR algorithm, followed by an end-to-end training. In the DEEPTURBO decoder [405], on the other hand, all Bi-GRUs are trained directly to optimize the end-to-end performance.

of parameters. The resulting TurboNet+ decoder was shown to achieve state-of-the-art performance and outperform existing DL turbo decoders even with lower computational complexity.

Subsequently, the authors in [407] proposed TINYTURBO which significantly reduces the trainable parameters of TurboNet+ by sharing the same weight across bit indices in the computation of the posterior LLR. In particular, for a block length of 40, it was demonstrated that TINYTURBO with 18 parameters outperforms TurboNet+ with 720 parameters over AWGN channels. Furthermore, the strong adaptability of TINYTURBO to other block lengths, rates, and trellises, as well as its robustness to channel variations, were demonstrated.

E. DL-Aided Decoding of Cyclic Codes

DL decoders exploiting algebraic properties of cyclic codes have also been studied. In [412], the authors proposed a neural network-based decoder for cyclic codes by exploiting their cyclically invariant property. More specifically, inspired by the fact that the maximum-likelihood decoder of any cyclic code is equivariant with respect to cyclic shifts, they imposed a shift-invariant structure on the weights of the neural decoder so that any cyclic shift of inputs results in the same cyclic shift of the outputs. Simulations of BCH codes and punctured Reed-Muller (RM) codes showed that the proposed decoder consistently outperforms the neural BP decoder proposed in [258]. Furthermore, they proposed a list decoding procedure that can significantly reduce the decoding error for BCH codes and punctured RM codes.

While the list decoding significantly improves the BLER, the major drawback was its relatively high BER. To improve the BER, the same authors proposed the improved version

TABLE VI
SUMMARY OF DL-AIDED CONVOLUTIONAL AND TURBO DECODERS.

Category	Reference	Main Contribution
Convolutional Decoders	Teich et al. [404]	A DNN decoder for convolutional codes.
Turbo Decoders	Kim et al. [244]	A neural network BCJR decoder, referred to as NEURALBCJR.
	Jiang et al. [405]	Deep turbo decoder (DEEPTURBO) trained in an end-to-end manner.
	He et al. [406]	A novel model-driven decoder, called TurboNet.
	Hebbar et al. [407]	TINYTURBO that reduces the parameters of TurboNet+.

of the list decoder in [413]. The new decoder achieved a significantly lower BER compared to the list decoder in [412] while maintaining the same BLER.

V. CONCLUSION

In this paper, we have provided a comprehensive survey on DL for the channel coding problems. In particular, we have focused on DL methods for the code design and channel decoding problems. In what follows, we summarize the potential advantages and challenges of these approaches.

A. Code Design Applications

The conventional code design algorithms such as EXIT chart and variants of DE require ideal assumptions about channel models and decoding schemes. In contrast, the major advantage of using data-driven DL for code design is that one can tailor codes to more realistic channels and decoding schemes, for which theoretical analysis is intractable.

In Section III, we saw that RL is a particularly popular approach to designing polar codes, among others. In this method, an agent learns to choose a new information bit position that minimizes the cumulative reward, which corresponds to the actual performance in terms of BLER. Calculating the reward requires Monte-Carlo simulations, which can be computationally intensive depending on the code length and target error rate. This complexity issue can hinder its application to scenarios with long code lengths and low error rates. Therefore, a design objective that can be efficiently computed during the training process may be desirable.

B. Channel Decoding Applications

As we have seen, channel decoding is a popular application of DL and a significant number of papers on this topic have become available. These approaches can be broadly classified into model-free and model-based approaches.

1) Model-Free Approaches: Model-free decoders employing a "black-box" neural network have the potential to outperform conventional decoding algorithms in terms of error rate performance and decoding complexity/latency. In particular, it has been demonstrated that model-free decoders can outperform existing decoding algorithms for short code lengths with highly parallelizable structures. This approach is thus potentially suitable for low-latency applications requiring short code lengths. Note that the performance is highly dependent on the DL model employed, and currently, the Transformer-based decoder achieves the state-of-the-art performance [222]. However, the performance could be potentially improved by the advanced DL techniques.

Due to the curse of dimensionality, the applications of model-free decoders have generally been limited to short codes. In general, a larger model size is required to decode a longer code, which not only entails high computational complexity, but also high space complexity in both training and inference phases. Another concern about this method is the robustness against adversarial attacks [204], as wireless networks are always vulnerable to radio jamming attacks [414], [415] due to the openness of wireless channels. In particular, DL-based communication systems may have a higher risk of being disrupted by jamming attacks than classical systems [416], [417].

Instead of completely replacing conventional decoders, employing a DNN to augment existing decoders is effective for arbitrary code lengths. For example, model-free training has been extensively studied for DNN-based selection of flipping bit indices in SCF decoding as we reviewed in Section IV-C.

2) Model-Based Approaches: In contrast to the model-free approach, the model-based approach realizes a scalable decoder by taking advantage of the knowledge of code structures and conventional decoding algorithms. One of the most promising approaches is deep unfolding, which unfolds an iterative algorithm [306] and introduces a set of trainable parameters. In particular, as we reviewed in Section IV-B, neural network-based BP decoding over an unfolded Tanner graph augmented with trainable parameters [257] has been extensively investigated. As the underlying BP decoding algorithm has already been adopted in a wide range of communication systems, this approach can be applied to these systems with much less modification compared to model-free decoders. Many existing works have demonstrated that, by introducing and optimizing trainable weights that mitigate the effect of short cycles in the Tanner graph, the DL BP decoder can achieve better a trade-off between decoding performance and latency, i.e., the number of decoding iterations, compared to the standard BP decoder. This means that the performance advantage of the DL BP decoder becomes more significant as the number of short cycles increases.

C. Challenges and Future Directions

Despite their excellent performance, DL-based channel coding schemes face challenges that need to be addressed. We conclude this survey by highlighting several future research directions in this regard.

1) Flexibility to Support Diverse Applications: Next generation communication systems such as 6G will support heterogeneous applications that employ the channel codes with various block lengths, reliability, and latency requirements [37]. Since there is no one-size-fits-all channel coding scheme,

multiple code parameters, i.e., rates and lengths, must be supported to meet these requirements. On the other hand, adapting a DL decoder to different channels and code parameters would require an enormous amount of different parameter sets. This issue could be alleviated, for example, by a parameter sharing scheme and scalable GNNs as discussed in Section IV-B. Furthermore, in order to support multiple code parameters, training must be performed multiple times, which is time consuming and computationally intensive. This complexity issue can be addressed by techniques such as transfer learning, meta-learning, and foundation models [248].

2) Explainable AI: One of the disadvantages of DL methods is their black-box nature, which can hinder physical insights into the phenomena. Thus, evaluating and enhancing the explainability of generic DL models, i.e., the ability to provide reasons for the outcomes of the system [418], remains an active field of research [419]–[425]. In general, the explainability of DL models tends to have an inverse relationship to their performance, e.g., prediction accuracy [426]. Thus, a recent advanced DL model with a large number of parameters is particularly difficult to interpret and explain.

In the next generation communications such as 6G, the concept of explainable AI (XAI) will become increasingly important especially for the emerging mission-critical services, such as autonomous driving and remote surgery [427]–[429]. Although the effectiveness of DL for the physical layer has been demonstrated in terms of its performance, its explainability has not been well studied. Thus, XAI-based channel coding that increases the transparency of DL models and explains the reasons for decisions will be of practical importance. The new insights gained from XAI will also help us to devise code design and decoding algorithms.

3) Efficient Training and Inference: Recent advances in DL technologies have been driven by the exponential growth of data and computational power, with a focus on performance rather than the economic and environmental costs. This research trend is often referred to as *Red AI* [430]. Indeed, the computations required for DL algorithms result in a surprisingly large carbon footprint [431]–[433]. In contrast to Red AI, which prioritizes achieving state-of-the-art results, *Green AI* aims to produce innovative results while taking into account computational costs [430], [434]. This paradigm shift toward energy and cost efficiency is inevitable for the long-term success, and it is therefore important to carefully select and preprocess data, reduce redundancy, and avoid overfitting so as to minimize the amounts of data and computational resources required [435].

Data-centric approaches are promising for reducing the energy consumption of DL algorithms [434], while their primary goal was to improve performance in terms of accuracy[7]. These approaches recognize that the quality of the training data has a significant impact on their performance, and thus prioritize data quality over model refinement [436]–[438]. These include active learning, knowledge transfer, dataset distillation, data augmentation, and curriculum learning. For channel decoding applications, some papers have employed these techniques for

---

[7]https://datacentricai.org/

enhancing error rate performance, but more emphasis should be placed on the data efficiency.

4) Quantum Machine Learning: Quantum computing has a great potential to solve the classical channel coding problems [439]–[444] more efficiently than digital computers. Especially in the current noisy intermediate-scale quantum (NISQ) era [445], where the fidelity of quantum gates is limited by noise and decoherence, hybrid quantum-classical algorithms such as variational quantum eigensolver (VQE) [446] and quantum approximate optimization algorithm (QAOA) [447] are promising [448], [449]. The potential of these algorithms for the classical channel decoding problems has been demonstrated in [440].

Furthermore, quantum machine learning (QML), which integrates quantum algorithms into ML, has received increasing attention [88], [450]–[455]. For example, it has been shown that well-designed quantum neural networks can achieve a higher capacity and faster training ability than comparable classical feedforward neural networks [456]. Furthermore, quantum counterparts to classical CNNs, autoencoders, and generative adversarial networks (GANs) have been studied in [457]–[460]. These methods have the potential to improve existing methods based on classical computers for a wide range of communication problems including channel coding.

LIST OF ABBREVIATIONS

3GPP	Third Generation Partnership Project
A2C	Advantage Actor-Critic
ADMM	Alternating Direction Method Of Multipliers
AE	Autoencoder
AMS	Adjusted Min-Sum
AWGN	Additive White Gaussian Noise
BCH	Bose–Chaudhuri–Hocquenghem
BCJR	Bahl-Cocke-Jelinek-Raviv
BER	Bit Error Rate
BF	Bit Flipping
BGD	Batch Gradient Descent
BI-AWGN	Binary-Input Additive White Gaussian Noise
Bi-GRU	Bidirectional Gated Recurrent Unit
BISO	Binary-Input Symmetric-Output
BLER	Block Error Rate
BP	Belief Propagation
BPSK	Binary Phase Shift Keying
CA-SCL	Cyclic Redundancy Check-Aided Successive Cancellation List
CN	Check Node
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
DDPM	Denoising Diffusion Probabilistic Model
DE	Density Evolution
DL	Deep Learning
DM	Diffusion Model
DNC	Differentiable Neural Computer
DNN	Deep Neural Network
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
DSCF	Dynamic Successive Cancellation Flip

DSCLF	Dynamic Successive Cancellation List Flip	PAPR	Peak-To-Average Power Ratio
ECCT	Error Correction Code Transformer	PB-LDPC	Protograph-Based Low-Density Parity-Check
eMBB	Enhanced Mobile Broadband	PCCMP	Polar-Code-Construction Message-Passing
EXIT	Extrinsic Information Transfer	PCM	Parity Check Matrix
FAID	Finite Alphabet Iterative Decoder	PGD	Projected Gradient Descent
FPGA	Field Programmable Gate Array	PNN	Partitioned Neural Network
FSCL	Fast Successive Cancellation List	QAOA	Quantum Approximate Optimization Algorithm
FSCLF	Fast Successive Cancellation List Clip	QML	Quantum Machine Learning
GA	Gaussian Approximation	RBMS	Reliability-Based Min-Sum
GAN	Generative Adversarial Network	RC	Repetition Code
GNN	Graph Neural Network	RCA	Reciprocal Channel Approximation
GPT	Generative Pre-Trained Transformer	ReLU	Rectified Linear Unit
GPU	Graphics Processing Unit	RL	Reinforcement Learning
GRU	Gated Recurrent Unit	RLM	Regularized Loss Minimization
HARQ	Hybrid Automatic Repeat Request	RM	Reed-Muller
HDD	Hard-Decision Decoder	RNN	Reccurent Neural Network
HDPC	High-Density Parity-Check	RQNN	Recurrent Quantized Neural Network
HORNN	High-Order Recurrent Neural Network	RRD	Random Redundant Decoding
ILSVRC	ImageNet Large Scale Visual Recognition Challenge	RSC	Recursive Systematic Convolutional
ISI	Inter-Symbol Interference	SC	Successive Cancellation
ISIT	International Symposium On Information Theory	SCF	Successive Cancellation Flip
ITD	Iterative Threshold Decoding	SCL	Successive Cancellation List
KPI	Key Performance Indicator	SCLF	Successive Cancellation List Flip
LAMS	Min-Sum Decoding With Linear Approximation	SCMS	Self-Corrected Min-Sum
LDPC	Low-Density Parity Check	SGD	Stochastic Gradient Descent
LLaMA	Large Language Model Meta AI	SISO	Soft-Input Soft-Output
LLM	Large Language Model	SMMS	Single-Minimum Min-Sum
LLR	Log-Likelihood Ratio	SNR	Signal-To-Noise Ratio
LP	Linear Programming	SP	Shifted-Pruning
LSTM	Long Short Term Memory	SPC	Single Parity Check
LUT	Look-Up Table	STE	Straight-Through Estimator
MAP	Maximum A Posterior	STT-MRAM	Spin-Torque Transfer Magnetic Random Access Memory
MBBP	Multiple-Bases Belief-Propagation	SVM	Support Vector Machine
MBSGD	Mini-Batch Stochastic Gradient Descent	TPU	Tensor Processing Unit
MDP	Markov Decision Process	TR	Tensor-Ring
MIMO	Multiple-Input Multiple-Output	TT	Tensor-Train
MIND	Model Independent Neural Decoder	URLLC	Ultra-Reliable Low-Latency Communication
ML	Machine Learning	V2X	Vehicle-To-Everything
MLP	Multi-Layer Perceptron	VN	Variable Node
mMTC	Massive Machine Type Communication	VQE	Variational Quantum Eigensolver
MPNN	Message Passing Neural Network	VWMS	Variable Weight Min-Sum
mRRD	Modified Random Redundant Decoding	XAI	Explainable AI
MS	Min-Sum	XR	Extended Reality
MU	Multi-User		
NC-RD	Node-Classified Redundant Decoding		
NISQ	Noisy Intermediate-Scale Quantum		
NLP	Natural Language Processing		
NMS	Normalized Min-Sum		
NOMS	Normalized Offset Min-Sum		
NSC	Neural Successive Cancellation		
OFDM	Orthogonal Frequency Division Multiplexing		
OMS	Offset Min-Sum		
OSD	Ordered Statistic Decoding		
PAC	Polarization Adjusted Convolutional		
PaLM	Pathways Language Model		

REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423 and 623–656, July and Oct. 1948.

[2] D. J. Costello and G. D. Forney, "Channel coding: The road to channel capacity," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1150–1177, 2007.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE International Conference on Communications (ICC)*, pp. 1064–1070, May 1993.

[4] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Info. Theory,*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[5] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.

[6] P. Yang, Y. Xiao, M. Xiao, and S. Li, "6G wireless communications: Vision and potential techniques," *IEEE Network*, vol. 33, no. 4, pp. 70–75, 2019.

[7] Z. Zhang, Y. Xiao, Z. Ma, M. Xiao, Z. Ding, X. Lei, G. K. Karagiannidis, and P. Fan, "6G wireless networks: Vision, requirements, architecture, and key technologies," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 28–41, 2019.

[8] T. Huang, W. Yang, J. Wu, J. Ma, X. Zhang, and D. Zhang, "A survey on green 6G network: Architecture and technologies," *IEEE Access*, vol. 7, pp. 175 758–175 768, 2019.

[9] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Network*, vol. 34, no. 3, pp. 134–142, 2019.

[10] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The roadmap to 6G: AI empowered wireless networks," *IEEE Commun. Mag.*, vol. 57, no. 8, pp. 84–90, 2019.

[11] M. Z. Chowdhury, M. Shahjalal, S. Ahmed, and Y. M. Jang, "6G wireless communication systems: Applications, requirements, technologies, challenges, and research directions," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 957–975, 2020.

[12] A. Dogra, R. K. Jha, and S. Jain, "A survey on beyond 5G network with the advent of 6G: Architecture and emerging technologies," *IEEE Access*, vol. 9, pp. 67 512–67 547, 2020.

[13] G. Gui, M. Liu, F. Tang, N. Kato, and F. Adachi, "6G: Opening new horizons for integration of comfort, security, and intelligence," *IEEE Wireless Communications*, vol. 27, no. 5, pp. 126–132, 2020.

[14] N. Kato, B. Mao, F. Tang, Y. Kawamoto, and J. Liu, "Ten challenges in advancing machine learning technologies toward 6G," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 96–103, 2020.

[15] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6G networks: Use cases and technologies," *IEEE Commun. Mag.*, vol. 58, no. 3, pp. 55–61, 2020.

[16] N. Rajatheva, I. Atzeni, E. Bjornson, A. Bourdoux, S. Buzzi, J.-B. Dore, S. Erkucuk, M. Fuentes, K. Guan, Y. Hu *et al.*, "White paper on broadband connectivity in 6G," *arXiv preprint arXiv:2004.14247*, 2020.

[17] S. Chen, Y.-C. Liang, S. Sun, S. Kang, W. Cheng, and M. Peng, "Vision, requirements, and technology trend of 6G: How to tackle the challenges of system coverage, capacity, user data-rate and movement speed," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 218–228, 2020.

[18] S. Dang, O. Amin, B. Shihada, and M.-S. Alouini, "What should 6G be?" *Nature Electronics*, vol. 3, no. 1, pp. 20–29, 2020.

[19] I. F. Akyildiz, A. Kak, and S. Nie, "6G and beyond: The future of wireless communications systems," *IEEE Access*, vol. 8, pp. 133 995–134 030, 2020.

[20] L. U. Khan, I. Yaqoob, M. Imran, Z. Han, and C. S. Hong, "6G wireless systems: A vision, architectural elements, and future directions," *IEEE Access*, vol. 8, pp. 147 029–147 044, 2020.

[21] E. Yaacoub and M.-S. Alouini, "A key 6G challenge and opportunity—connecting the base of the pyramid: A survey on rural connectivity," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 533–582, 2020.

[22] H. Viswanathan and P. E. Mogensen, "Communications in the 6G era," *IEEE Access*, vol. 8, pp. 57 063–57 074, 2020.

[23] F. Tariq, M. R. Khandaker, K.-K. Wong, M. A. Imran, M. Bennis, and M. Debbah, "A speculative study on 6G," *IEEE Wireless Communications*, vol. 27, no. 4, pp. 118–125, 2020.

[24] H. Tataria, M. Shafi, A. F. Molisch, M. Dohler, H. Sjöland, and F. Tufvesson, "6G wireless systems: Vision, requirements, challenges, insights, and opportunities," *Proceedings of the IEEE*, vol. 109, no. 7, pp. 1166–1199, 2021.

[25] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, D. Niyato, O. Dobre, and H. V. Poor, "6G internet of things: A comprehensive survey," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 359–383, 2021.

[26] W. Jiang, B. Han, M. A. Habibi, and H. D. Schotten, "The road towards 6G: A comprehensive survey," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 334–366, 2021.

[27] M. Alsabah, M. A. Naser, B. M. Mahmmod, S. H. Abdulhussain, M. R. Eissa, A. Al-Baidhani, N. K. Noordin, S. M. Sait, K. A. Al-Utaibi, and F. Hashim, "6G wireless communications networks: A comprehensive survey," *IEEE Access*, vol. 9, pp. 148 191–148 243, 2021.

[28] C. De Lima, D. Belot, R. Berkvens, A. Bourdoux, D. Dardari, M. Guillaud, M. Isomursu, E.-S. Lohan, Y. Miao, A. N. Barreto *et al.*, "Convergent communication, sensing and localization in 6G systems: An overview of technologies, opportunities and challenges," *IEEE Access*, vol. 9, pp. 26 902–26 925, 2021.

[29] C. De Alwis, A. Kalla, Q.-V. Pham, P. Kumar, K. Dev, W.-J. Hwang, and M. Liyanage, "Survey on 6G frontiers: Trends, applications, requirements, technologies and future research," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 836–886, 2021.

[30] F. Guo, F. R. Yu, H. Zhang, X. Li, H. Ji, and V. C. Leung, "Enabling massive iot toward 6G: A comprehensive survey," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 11 891–11 915, 2021.

[31] M. Matthaiou, O. Yurduseven, H. Q. Ngo, D. Morales-Jimenez, S. L. Cotton, and V. F. Fusco, "The road to 6G: Ten physical layer challenges for communications engineers," *IEEE Commun. Mag.*, vol. 59, no. 1, pp. 64–69, 2021.

[32] M. Noor-A-Rahim, Z. Liu, H. Lee, M. O. Khyam, J. He, D. Pesch, K. Moessner, W. Saad, and H. V. Poor, "6G for vehicle-to-everything (V2X) communications: Enabling technologies, challenges, and opportunities," *Proceedings of the IEEE*, vol. 110, no. 6, pp. 712–734, 2022.

[33] C.-X. Wang, X. You, X. Gao, X. Zhu, Z. Li, C. Zhang, H. Wang, Y. Huang, Y. Chen, H. Haas *et al.*, "On the road to 6G: Visions, requirements, key technologies and testbeds," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 905–974, 2023.

[34] M. Chafii, L. Bariah, S. Muhaidat, and M. Debbah, "Twelve scientific challenges for 6G: Rethinking the foundations of communications theory," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 868–904, 2023.

[35] C. You, Y. Cai, Y. Liu, M. Di Renzo, T. M. Duman, A. Yener, and A. L. Swindlehurst, "Next generation advanced transceiver technologies for 6G," *arXiv preprint arXiv:2403.16458*, 2024.

[36] M. Geiselhart, F. Krieg, J. Clausius, D. Tandler, and S. ten Brink, "6G: A welcome chance to unify channel coding?" *IEEE BITS the Information Theory Magazine*, vol. 3, no. 1, pp. 67–80, 2023.

[37] H. Zhang and W. Tong, "Channel coding for 6G extreme connectivity-requirements, capabilities and fundamental tradeoffs," *IEEE BITS the Information Theory Magazine*, vol. 3, no. 1, pp. 54–66, 2023.

[38] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 265–283, 2016.

[39] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A Matlab-like environment for machine learning," in *Proc. Advances in Neural Information Processing Systems (NeurIPS) Workshop on Big Learning*, 2011.

[40] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–12, 2017.

[41] M. Qasaimeh, K. Denolf, J. Lo, K. Vissers, J. Zambreno, and P. H. Jones, "Comparing energy efficiency of CPU, GPU and FPGA implementations for vision kernels," in *Proc. IEEE International Conference on Embedded Software and Systems (ICESS)*, pp. 1–8, 2019.

[42] T. Wang, C.-K. Wen, H. Wang, F. Gao, T. Jiang, and S. Jin, "Deep learning for wireless physical layer: Opportunities and challenges," *China Communications*, vol. 14, no. 11, pp. 92–111, 2017.

[43] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.

[44] D. Gündüz, P. de Kerret, N. D. Sidiropoulos, D. Gesbert, C. R. Murthy, and M. van der Schaar, "Machine learning in the air," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2184–2199, 2019.

[45] A. Balatsoukas-Stimming and C. Studer, "Deep unfolding for communications systems: A survey and some new directions," in *Proc. IEEE International Workshop on Signal Processing Systems (SiPS)*, pp. 266–271, 2019.

[46] A. Samad, W. Saad, R. Nandana, C. Kapseok, D. Steinbach, B. Sliwa, C. Wietfeld, K. Mei, S. Hamid, H.-J. Zepernick *et al.*, *White Paper on Machine Learning in 6G Wireless Communication Networks*. University of Oulu, 2020.

[47] C. Zhang, Y.-L. Ueng, C. Studer, and A. Burg, "Artificial intelligence for 5G and beyond 5G: Implementations, algorithms, and optimizations," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 10, no. 2, pp. 149–163, 2020.

[48] A. Ly and Y.-D. Yao, "A review of deep learning in 5G research: Channel coding, massive MIMO, multiple access, resource allocation, and network security," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 396–408, 2021.

[49] C. Mao, Z. Mu, Q. Liang, I. Schizas, and C. Pan, "Deep learning in physical layer communications: Evolution and prospects in 5G and 6G networks," *IET Communications*, vol. 17, no. 16, pp. 1863–1876, 2023.

[50] M. Akrout, A. Feriani, F. Bellili, A. Mezghani, and E. Hossain, "Domain generalization in machine learning models for wireless communications: Concepts, state-of-the-art, and open issues," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 3014–3037, 2023.

[51] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, "Domain generalization: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 4, pp. 4396–4415, 2022.

[52] J. Wang, C. Lan, C. Liu, Y. Ouyang, T. Qin, W. Lu, Y. Chen, W. Zeng, and P. Yu, "Generalizing to unseen domains: A survey on domain generalization," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 8, pp. 8052–8072, 2022.

[53] N. Ye, S. Miao, J. Pan, Q. Ouyang, X. Li, and X. Hou, "Artificial intelligence for wireless physical-layer technologies (AI4PHY): A comprehensive survey," *IEEE Trans. Cogn. Commun. Netw.*, 2024.

[54] M. Rowshan, M. Qiu, Y. Xie, X. Gu, and J. Yuan, "Channel coding towards 6G: Technical overview and outlook," *IEEE Open J. Commun. Soc.*, vol. 5, pp. 2585–2685, 2024.

[55] X.-A. Wang and S. B. Wicker, "An artificial neural net Viterbi decoder," *IEEE Trans. Commun.*, vol. 44, no. 2, pp. 165–171, 1996.

[56] A. Hamalainen and J. Henriksson, "A recurrent neural decoder for convolutional codes," in *Proc. IEEE International Conference on Communications (ICC)*, vol. 2, pp. 1305–1309, 1999.

[57] M. Ibnkahla, "Applications of neural networks to digital communications–A survey," *Signal Processing*, vol. 80, no. 7, pp. 1185–1215, 2000.

[58] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, 2017.

[59] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 4, pp. 648–664, 2018.

[60] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, "Deep learning based communication over the air," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 132–143, 2018.

[61] H. Huang, S. Guo, G. Gui, Z. Yang, J. Zhang, H. Sari, and F. Adachi, "Deep learning for physical-layer 5G wireless techniques: Opportunities, challenges and solutions," *IEEE Wireless Communications*, vol. 27, no. 1, pp. 214–222, 2019.

[62] H. He, S. Jin, C.-K. Wen, F. Gao, G. Y. Li, and Z. Xu, "Model-driven deep learning for physical layer communications," *IEEE Wireless Communications*, vol. 26, no. 5, pp. 77–83, 2019.

[63] M. Kim, W. Lee, J. Yoon, and O. Jo, "Toward the realization of encoder and decoder using deep neural networks," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 57–63, 2019.

[64] M. Varasteh, J. Hoydis, and B. Clerckx, "Learning to communicate and energize: Modulation, coding, and multiple access designs for wireless information-power transmission," *IEEE Trans. Commun.*, vol. 68, no. 11, pp. 6822–6839, 2020.

[65] F. Restuccia and T. Melodia, "Deep learning at the physical layer: System challenges and applications to 5G and beyond," *IEEE Commun. Mag.*, vol. 58, no. 10, pp. 58–64, 2020.

[66] S. Zhang, J. Liu, T. K. Rodrigues, and N. Kato, "Deep learning techniques for advancing 6G communications in the physical layer," *IEEE Wireless Communications*, vol. 28, no. 5, pp. 141–147, 2021.

[67] B. Ozpoyraz, A. T. Dogukan, Y. Gevez, U. Altun, and E. Basar, "Deep learning-aided 6G wireless networks: A comprehensive survey of revolutionary PHY architectures," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 1749–1809, 2022.

[68] Z. Lu, R. Li, K. Lu, X. Chen, E. Hossain, Z. Zhao, and H. Zhang, "Semantics-empowered communications: A tutorial-cum-survey," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 1, pp. 41–79, 2023.

[69] T. Ohtsuki, "Machine learning in 6G wireless communications," *IEICE Trans. Commun.*, vol. 106, no. 2, pp. 75–83, 2023.

[70] Y. Shi, L. Lian, Y. Shi, Z. Wang, Y. Zhou, L. Fu, L. Bai, J. Zhang, and W. Zhang, "Machine learning for large-scale optimization in 6G wireless networks," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2088–2132, 2023.

[71] N. Van Huynh, J. Wang, H. Du, D. T. Hoang, D. Niyato, D. N. Nguyen, D. I. Kim, and K. B. Letaief, "Generative AI for physical layer communications: A survey," *IEEE Trans. Cogn. Commun. Netw.*, 2024.

[72] N. Islam and S. Shin, "Deep learning in physical layer: Review on data driven end-to-end communication systems and their enabling semantic applications," *arXiv preprint arXiv:2401.12800*, 2024.

[73] T. M. Hoang, A. Vahid, H. D. Tuan, and L. Hanzo, "Physical layer authentication and security design in the machine learning era," *IEEE Communications Surveys & Tutorials*, 2024.

[74] J. Hoydis, S. Cammerer, F. A. Aoudia, A. Vem, N. Binder, G. Marcus, and A. Keller, "Sionna: An open-source library for next-generation physical layer research," *arXiv preprint arXiv:2203.11854*, 2022.

[75] H. Xie, Z. Qin, G. Y. Li, and B.-H. Juang, "Deep learning enabled semantic communication systems," *IEEE Trans. Signal Process.*, vol. 69, pp. 2663–2675, 2021.

[76] Z. Qin, X. Tao, J. Lu, W. Tong, and G. Y. Li, "Semantic communications: Principles and challenges," *arXiv preprint arXiv:2201.01389*, 2021.

[77] M. Xu, W. C. Ng, W. Y. B. Lim, J. Kang, Z. Xiong, D. Niyato, Q. Yang, X. S. Shen, and C. Miao, "A full dive into realizing the edge-enabled metaverse: Visions, enabling technologies, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 656–700, 2022.

[78] W. Yang, H. Du, Z. Q. Liew, W. Y. B. Lim, Z. Xiong, D. Niyato, X. Chi, X. S. Shen, and C. Miao, "Semantic communications for future internet: Fundamentals, applications, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 213–250, 2022.

[79] X. Luo, H.-H. Chen, and Q. Guo, "Semantic communications: Overview, open issues, and future research directions," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 210–219, 2022.

[80] D. Gündüz, Z. Qin, I. E. Aguerri, H. S. Dhillon, Z. Yang, A. Yener, K. K. Wong, and C.-B. Chae, "Beyond transmitting bits: Context, semantics, and task-oriented communications," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 1, pp. 5–41, 2022.

[81] T. M. Getu, G. Kaddoum, and M. Bennis, "Making sense of meaning: A survey on metrics for semantic and goal-oriented communication," *IEEE Access*, vol. 11, pp. 2169–3536, 2023.

[82] K. Li, B. P. L. Lau, X. Yuan, W. Ni, M. Guizani, and C. Yuen, "Towards ubiquitous semantic metaverse: Challenges, approaches, and opportunities," *IEEE Internet Things J.*, vol. 10, no. 24, pp. 21 855–21 872, 2023.

[83] D. Wheeler and B. Natarajan, "Engineering semantic communication: A survey," *IEEE Access*, vol. 11, pp. 13 965–13 995, 2023.

[84] G. Torlai and R. G. Melko, "Neural decoder for topological codes," *Physical Review Letters*, vol. 119, no. 3, p. 030501, 2017.

[85] S. Varsamopoulos, B. Criger, and K. Bertels, "Decoding small surface codes with feedforward neural networks," *Quantum Science and Technology*, vol. 3, no. 1, p. 015004, 2017.

[86] S. Varsamopoulos, K. Bertels, and C. G. Almudever, "Comparing neural network based decoders for the surface code," *IEEE Transactions on Computers*, vol. 69, no. 2, pp. 300–311, 2019.

[87] H. P. Nautrup, N. Delfosse, V. Dunjko, H. J. Briegel, and N. Friis, "Optimizing quantum error correction codes with reinforcement learning," *Quantum*, vol. 3, p. 215, 2019.

[88] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, "Machine learning and the physical sciences," *Reviews of Modern Physics*, vol. 91, no. 4, p. 045002, 2019.

[89] A. d. iOlius, P. Fuentes, R. Orús, P. M. Crespo, and J. E. Martinez, "Decoding algorithms for surface codes," *arXiv preprint arXiv:2307.14989*, 2023.

[90] M. Krenn, J. Landgraf, T. Foesel, and F. Marquardt, "Artificial intelligence and machine learning for quantum technologies," *Physical Review A*, vol. 107, no. 1, p. 010101, 2023.

[91] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[92] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[93] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[94] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[95] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[96] L. Deng, D. Yu *et al.*, "Deep learning: methods and applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[97] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–36, 2018.

[98] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. Awwal, and V. K. Asari, "A state-of-the-art survey on deep learning theory and architectures," *Electronics*, vol. 8, no. 3, p. 292, 2019.

[99] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, p. 100379, 2021.

[100] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A survey of deep learning and its applications: a new paradigm to machine learning," *Archives of Computational Methods in Engineering*, vol. 27, pp. 1071–1092, 2020.

[101] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, *Learning Internal Representations By Error Propagation*. MIT Press, Cambridge, MA, 1985.

[102] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, 2022.

[103] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. International Conference on Machine Learning (ICML)*, pp. 807–814, 2010.

[104] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. International Conference on Machine Learning (ICML)*, vol. 30, no. 1, p. 3, 2013.

[105] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015.

[106] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. International Conference on Machine Learning (ICML)*, pp. 1139–1147, 2013.

[107] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[108] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[109] T. Dozat, "Incorporating nesterov momentum into Adam," in *Proc. International Conference on Learning Representations (ICLR)*, pp. 2013–2016, 2016.

[110] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[111] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[112] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[113] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[114] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, 2017.

[115] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[116] S. E. Li, *Deep reinforcement learning*. Springer, 2023.

[117] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker *et al.*, "Model-based reinforcement learning: A survey," *Foundations and Trends® in Machine Learning*, vol. 16, no. 1, pp. 1–118, 2023.

[118] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2009.

[119] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, pp. 1–40, 2016.

[120] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Proc. International Conference on Artificial Neural Networks (ICANN)*, pp. 270–279, 2018.

[121] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[122] M. Wang, Y. Lin, Q. Tian, and G. Si, "Transfer learning promotes 6G wireless communications: Recent advances and future challenges," *IEEE Trans. Rel.*, vol. 70, no. 2, pp. 790–807, 2021.

[123] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," *arXiv preprint arXiv:2009.09796*, 2020.

[124] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 12, pp. 5586–5609, 2021.

[125] M. Kim, W. Lee, and D.-H. Cho, "A novel PAPR reduction scheme for OFDM system based on deep learning," *IEEE Commun. Lett.*, vol. 22, no. 3, pp. 510–513, 2017.

[126] J. Vanschoren, "Meta-learning: A survey," *arXiv preprint arXiv:1810.03548*, 2018.

[127] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5149–5169, 2021.

[128] L. Chen, S. T. Jose, I. Nikoloska, S. Park, T. Chen, O. Simeone *et al.*, "Learning with limited samples: Meta-learning and applications to communication systems," *Foundations and Trends® in Signal Processing*, vol. 17, no. 2, pp. 79–208, 2023.

[129] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. International Conference on Machine Learning (ICML)*, pp. 41–48, 2009.

[130] X. Wang, Y. Chen, and W. Zhu, "A survey on curriculum learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 4555–4576, 2021.

[131] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, "Curriculum learning: A survey," *International Journal of Computer Vision*, vol. 130, no. 6, pp. 1526–1565, 2022.

[132] J. L. Elman, "Learning and development in neural networks: The importance of starting small," *Cognition*, vol. 48, no. 1, pp. 71–99, 1993.

[133] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern recognition*, vol. 77, pp. 354–377, 2018.

[134] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, 2021.

[135] M. Krichen, "Convolutional neural networks: A survey," *Computers*, vol. 12, no. 8, p. 151, 2023.

[136] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1097–1105, 2012.

[137] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[138] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

[139] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[140] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 4700–4708, 2017.

[141] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 6232–6251, 2016.

[142] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh, "Deep learning-based channel estimation," *IEEE Commun. Lett.*, vol. 23, no. 4, pp. 652–655, 2019.

[143] M. Honkala, D. Korpi, and J. M. Huttunen, "DeepRx: Fully convolutional deep learning receiver," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3925–3940, 2021.

[144] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 3104–3112, 2014.

[145] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[146] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, pp. 2451–2471, 2000.

[147] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[148] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[149] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," *arXiv preprint arXiv:1801.01078*, 2017.

[150] G. Van Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model," *Artificial Intelligence Review*, vol. 53, pp. 5929–5955, 2020.

[151] N. Farsad and A. Goldsmith, "Neural network detection of data sequences in communication systems," *IEEE Trans. Signal Process.*, vol. 66, no. 21, pp. 5663–5678, 2018.

[152] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, 2008.

[153] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[154] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[155] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, 2020.

[156] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[157] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[158] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," *arXiv preprint arXiv:2104.13478*, 2021.

[159] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[160] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Proc. Advances in neural information processing systems (NeurIPS)*, vol. 33, pp. 1877–1901, 2020.

[161] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[162] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "PaLM: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[163] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.

[164] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *AI Open*, vol. 3, pp. 111–132, 2022.

[165] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM computing surveys (CSUR)*, vol. 54, no. 10s, pp. 1–41, 2022.

[166] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu *et al.*, "A survey on vision transformer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 87–110, 2022.

[167] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16×16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[168] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *Proc. International Conference on Machine Learning (ICML)*, pp. 28 492–28 518, 2023.

[169] Y. Wang, Z. Gao, D. Zheng, S. Chen, D. Gunduz, and H. V. Poor, "Transformer-empowered 6G intelligent networks: From massive MIMO processing to semantic communication," *IEEE Wireless Communications*, vol. 30, no. 6, pp. 127–135, 2022.

[170] Z. Chang, G. A. Koulieris, and H. P. Shum, "On the design fundamentals of diffusion models: A survey," *arXiv preprint arXiv:2306.04542*, 2023.

[171] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," *ACM Computing Surveys*, vol. 56, no. 4, pp. 1–39, 2023.

[172] H. Cao, C. Tan, Z. Gao, Y. Xu, G. Chen, P.-A. Heng, and S. Z. Li, "A survey on generative diffusion models," *IEEE Trans. Knowl. Data Eng.*, 2024.

[173] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 6840–6851, 2020.

[174] B. Fesl, M. Baur, F. Strasser, M. Joham, and W. Utschick, "Diffusion-based generative prior for low-complexity MIMO channel estimation," *arXiv preprint arXiv:2403.03545*, 2024.

[175] M. Letafati, S. Ali, and M. Latva-aho, "Denoising diffusion probabilistic models for hardware-impaired communications," *arXiv preprint arXiv:2309.08568*, 2023.

[176] ——, "Probabilistic constellation shaping with denoising diffusion probabilistic models: A novel approach," *arXiv preprint arXiv:2309.08688*, 2023.

[177] M. Kim, R. Fritschek, and R. F. Schaefer, "Learning end-to-end channel coding with diffusion models," in *Proc. 26th International ITG Workshop on Smart Antennas and 13th Conference on Systems, Communications, and Coding (WSA & SCC)*, pp. 1–6, 2023.

[178] T. Wu, Z. Chen, D. He, L. Qian, Y. Xu, M. Tao, and W. Zhang, "CDDM: Channel denoising diffusion models for wireless communications," *IEEE Trans. Wireless Commun.*, 2024.

[179] H. Du, R. Zhang, Y. Liu, J. Wang, Y. Lin, Z. Li, D. Niyato, J. Kang, Z. Xiong, S. Cui *et al.*, "Beyond deep reinforcement learning: A tutorial on generative diffusion models in network optimization," *arXiv preprint arXiv:2308.05384*, 2023.

[180] L. Bariah, Q. Zhao, H. Zou, Y. Tian, F. Bader, and M. Debbah, "Large generative AI models for telecom: The next big thing?" *IEEE Commun. Mag.*, 2024.

[181] M. Letafati, S. Ali, and M. Latva-aho, "WiGenAI: The symphony of wireless and generative AI via diffusion models," *arXiv preprint arXiv:2310.07312*, 2023.

[182] N. Van Huynh, J. Wang, H. Du, D. T. Hoang, D. Niyato, D. N. Nguyen, D. I. Kim, and K. B. Letaief, "Generative AI for physical layer communications: A survey," *arXiv preprint arXiv:2312.05594*, 2023.

[183] C. Zhao, H. Du, D. Niyato, J. Kang, Z. Xiong, D. I. Kim, K. B. Letaief *et al.*, "Generative AI for secure physical layer communications: A survey," *arXiv preprint arXiv:2402.13553*, 2024.

[184] M. Ebada, S. Cammerer, A. Elkelesh, and S. ten Brink, "Deep learning-based polar code design," in *Proc. 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 177–183, 2019.

[185] L. Huang, H. Zhang, R. Li, Y. Ge, and J. Wang, "AI coding: Learning to construct error correction codes," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 26–39, 2019.

[186] M. Leonardon and V. Gripon, "Using deep neural networks to predict and improve the performance of polar codes," in *Proc. 11th International Symposium on Topics in Coding (ISTC)*, pp. 1–5, 2021.

[187] Y. Liao, S. A. Hashemi, H. Yang, and J. M. Cioffi, "Scalable polar code construction for successive cancellation list decoding: A graph neural network-based approach," *IEEE Trans. Commun.*, vol. 71, no. 11, pp. 6231–6245, 2023.

[188] V. Miloslavskaya, Y. Li, and B. Vucetic, "Neural network based adaptive polar coding," *IEEE Trans. Commun.*, 2023.

[189] Y. Li, Z. Chen, G. Liu, Y.-C. Wu, and K.-K. Wong, "Learning to construct nested polar codes: An attention-based set-to-element model," *IEEE Commun. Lett.*, vol. 25, no. 12, pp. 3898–3902, 2021.

[190] S. K. Ankireddy, S. A. Hebbar, H. Wan, J. Cho, and C. Zhang, "Nested construction of polar codes via Transformers," *arXiv preprint arXiv:2401.17188*, 2024.

[191] S. A. Hebbar, S. K. Ankireddy, H. Kim, S. Oh, and P. Viswanath, "DeepPolar: Inventing nonlinear large-kernel polar codes via deep learning," *arXiv preprint arXiv:2402.08864*, 2024.

[192] S. K. Mishra, D. Katyal, and S. A. Ganapathi, "A modified Q-learning algorithm for rate-profiling of polarization adjusted convolutional (PAC) codes," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 2363–2368, 2022.

[193] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, 2001.

[194] R. Mori and T. Tanaka, "Performance of polar codes with the construction using density evolution," *IEEE Commun. Lett.*, vol. 13, no. 7, pp. 519–521, 2009.

[195] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[196] E. Nisioti and N. Thomos, "Design of capacity-approaching low-density parity-check codes using recurrent neural networks," *arXiv preprint arXiv:2001.01249*, 2020.

[197] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6562–6582, 2013.

[198] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Trans. Commun.*, vol. 60, no. 11, pp. 3221–3227, 2012.

[199] H. Ochiai, P. Mitran, and H. V. Poor, "Capacity-approaching polar codes with long codewords and successive cancellation decoding based on improved gaussian approximation," *IEEE Trans. Commun.*, vol. 69, no. 1, pp. 31–43, 2021.

[200] H. Ochiai, K. Ikeya, and P. Mitran, "A new polar code design based on reciprocal channel approximation," *IEEE Trans. Commun.*, vol. 71, no. 2, pp. 631–643, 2023.

[201] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.

[202] L. Huang, H. Zhang, R. Li, Y. Ge, and J. Wang, "Reinforcement learning for nested polar code construction," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2019.

[203] Y. Liao, S. A. Hashemi, J. M. Cioffi, and A. Goldsmith, "Construction of polar codes with reinforcement learning," *IEEE Trans. Commun.*, vol. 70, no. 1, pp. 185–198, 2021.

[204] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[205] V. Miloslavskaya, Y. Li, and B. Vucetic, "Design of compactly specified polar codes with dynamic frozen bits based on reinforcement learning," *IEEE Trans. Commun.*, vol. 72, no. 3, pp. 1257–1272, 2024.

[206] C. Schürch, "A partial order for the synthesized channels of a polar code," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 220–224, 2016.

[207] M. Bardet, V. Dragoi, A. Otmani, and J.-P. Tillich, "Algebraic properties of polar codes from a new polynomial formalism," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 230–234, 2016.

[208] G. He, J.-C. Belfiore, I. Land, G. Yang, X. Liu, Y. Chen, R. Li, J. Wang, Y. Ge, R. Zhang *et al.*, "Beta-expansion: A theoretical framework for fast and recursive construction of polar codes," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2017.

[209] M. Mondelli, S. H. Hassani, and R. Urbanke, "Construction of polar codes with sublinear complexity," *IEEE Trans. Inf. Theory*, 2018.

[210] V. Bioglio, C. Condo, and I. Land, "Design of polar codes in 5G new radio," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 29–40, 2020.

[211] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 12, 1999.

[212] S. B. Korada, E. Şaşoğlu, and R. Urbanke, "Polar codes: Characterization of exponent, bounds, and constructions," *IEEE Trans. Inf. Theory*, vol. 56, no. 12, pp. 6253–6264, 2010.

[213] E. Arıkan, "From sequential decoding to channel polarization and back again," *arXiv preprint arXiv:1908.09594*, 2019.

[214] M. Moradi and A. Mozammel, "A monte-carlo based construction of polarization-adjusted convolutional (PAC) codes," *arXiv preprint arXiv:2106.08118*, 2021.

[215] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, "On deep learning-based channel decoding," in *Proc. 51st Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, 2017.

[216] J. Seo, J. Lee, and K. Kim, "Decoding of polar code by using deep feed-forward neural networks," in *Proc. International Conference on Computing, Networking and Communications (ICNC)*, pp. 238–242, 2018.

[217] C. T. Leung, R. V. Bhat, and M. Motani, "Low-latency neural decoders for linear and non-linear block codes," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2019.

[218] ——, "Low latency energy-efficient neural decoders for block codes," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 2, pp. 680–691, 2023.

[219] W. Lyu, Z. Zhang, C. Jiao, K. Qin, and H. Zhang, "Performance evaluation of channel decoding with deep neural networks," in *Proc. IEEE International Conference on Communications (ICC)*, pp. 1–6, 2018.

[220] R. Sattiraju, A. Weinand, and H. D. Schotten, "Performance analysis of deep learning based on recurrent neural networks for channel coding," in *Proc. IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pp. 1–6, 2018.

[221] H. Zhu, Z. Cao, Y. Zhao, and D. Li, "Learning to denoise and decode: A novel residual neural network decoder for polar codes," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8725–8738, 2020.

[222] Y. Choukroun and L. Wolf, "Error correction code transformer," *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 38 695–38 705, 2022.

[223] ——, "Denoising diffusion error correction codes," *arXiv preprint arXiv:2209.13533*, 2022.

[224] A. Bennatan, Y. Choukroun, and P. Kisilev, "Deep learning for decoding of linear codes-a syndrome-based approach," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 1595–1599, 2018.

[225] J. K. S. Kamassury and D. Silva, "Iterative error decimation for syndrome-based neural network decoders," *arXiv preprint arXiv:2012.00089*, 2020.

[226] D. Artemasov, K. Andreev, P. Rybin, and A. Frolov, "Soft-output deep neural network-based decoding," *arXiv preprint arXiv:2304.08868*, 2023.

[227] Y. Wang, Z. Zhang, S. Zhang, S. Cao, and S. Xu, "A unified deep learning based polar-ldpc decoder for 5G communication systems," in *Proc. 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1–6, 2018.

[228] Y. Jiang, H. Kim, H. Asnani, and S. Kannan, "Mind: Model independent neural decoder," in *Proc. IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, 2019.

[229] H. Lee, E. Y. Seo, H. Ju, and S.-H. Kim, "On training neural network decoders of rate compatible polar codes via transfer learning," *Entropy*, vol. 22, no. 5, p. 496, 2020.

[230] D. Artemasov, K. Andreev, and A. Frolov, "On a unified deep neural network decoding architecture," in *Proc. IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, pp. 1–5, 2023.

[231] F. Carpi, C. Häger, M. Martalò, R. Raheli, and H. D. Pfister, "Reinforcement learning for channel coding: Learned bit-flipping decoding," in *Proc. 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 922–929, 2019.

[232] J. Gao and K. Niu, "A reinforcement learning based decoding method of short polar codes," in *Proc. IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 1–6, 2021.

[233] E. Kavvousanos, V. Paliouras, and I. Kouretas, "Simplified deep-learning-based decoders for linear block codes," in *Proc. IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 769–772, 2018.

[234] E. Kavvousanos and V. Paliouras, "Hardware implementation aspects of a syndrome-based neural network decoder for BCH codes," in *Proc. IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, pp. 1–6, 2019.

[235] ——, "Optimizing deep learning decoders for FPGA implementation," in *Proc. 31st International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 271–272, 2021.

[236] ——, "A low-latency syndrome-based deep learning decoder architecture and its FPGA implementation," in *Proc. 11th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pp. 1–4, 2022.

[237] B. Cavarec, H. B. Celebi, M. Bengtsson, and M. Skoglund, "A learning-based approach to address complexity-reliability tradeoff in OS decoders," in *Proc. 54th Asilomar Conference on Signals, Systems, and Computers*, pp. 689–692, 2020.

[238] N. Raviv, A. Caciularu, T. Raviv, J. Goldberger, and Y. Be'ery, "perm2vec: Graph permutation selection for decoding of error correction codes using self-attention," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 79–88, 2020.

[239] A. Kurmukova and D. Gunduz, "Friendly attacks to improve channel coding reliability," *arXiv preprint arXiv:2401.14184*, 2024.

[240] A. Tsvieli and N. Weinberger, "Learning maximum margin channel decoders," *IEEE Trans. Inf. Theory*, vol. 69, no. 6, pp. 3597–3626, 2023.

[241] X. Zhong, K. Cai, Z. Mei, and T. Q. Quek, "Deep learning-based decoding of linear block codes for spin-torque transfer magnetic random access memory," *IEEE Transactions on Magnetics*, vol. 57, no. 2, pp. 1–5, 2020.

[242] X. Zhong, K. Cai, P. Kang, G. Song, and B. Dai, "Deep learning-based adaptive error-correction decoding for spin-torque transfer magnetic random access memory (STT-MRAM)," *IEEE Transactions on Magnetics*, vol. 59, no. 11, 2023.

[243] C. T. Leung, R. V. Bhat, and M. Motani, "Multi-label neural decoders for block codes," in *Proc. IEEE International Conference on Communications (ICC)*, pp. 1–6, 2020.

[244] H. Kim, Y. Jiang, R. Rana, S. Kannan, S. Oh, and P. Viswanath, "Communication algorithms via deep learning," in *Proc. International Conference on Learning Representations (ICLR)*, pp. 1–17, 2018.

[245] Z. Cao, H. Zhu, Y. Zhao, and D. Li, "Learning to denoise and decode: A novel residual neural network decoder for polar codes," in *Proc.*

IEEE 92nd Vehicular Technology Conference (VTC2020-Fall), pp. 1–6, 2020.

[246] S.-J. Park, H.-Y. Kwak, S.-H. Kim, S. Kim, Y. Kim, and J.-S. No, "How to mask in error correction code Transformer: Systematic and double masking," *arXiv preprint arXiv:2308.08128*, 2023.

[247] Y. Choukroun and L. Wolf, "A foundation model for error correction codes," in *Proc. International Conference on Machine Learning (ICML)*, 2024.

[248] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[249] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Elsevier, 1977, vol. 16.

[250] I. Dimnik and Y. Be'ery, "Improved random redundant iterative HDPC decoding," *IEEE Trans. Commun.*, vol. 57, no. 7, pp. 1982–1985, 2009.

[251] L. Tallini and P. Cull, "Neural nets for decoding error-correcting codes," in *IEEE Technical applications conference and workshops*, p. 89, 1995.

[252] R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Trans. Commun.*, vol. 46, no. 8, pp. 1003–1010, 1998.

[253] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *arXiv preprint arXiv:1710.01878*, 2017.

[254] M. P. C. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Inf. Theory*, vol. 41, no. 5, pp. 1379–1396, Sep. 1995.

[255] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Belief propagation list decoding of polar codes," *IEEE Commun. Lett.*, vol. 22, no. 8, pp. 1536–1539, 2018.

[256] A. Kurakin, I. J. Goodfellow, and S. Bengio, *Adversarial examples in the physical world*. Chapman and Hall/CRC, 2018.

[257] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *Proc. 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 341–346, 2016.

[258] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119–131, 2018.

[259] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 1361–1365, 2017.

[260] B. Dai, R. Liu, and Z. Yan, "New min-sum decoders based on deep learning for polar codes," in *Proc. IEEE International Workshop on Signal Processing Systems (SiPS)*, pp. 252–257, 2018.

[261] H. Yu, M.-M. Zhao, M. Lei, and M.-J. Zhao, "Neural adjusted min-sum decoding for LDPC codes," in *Proc. IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, pp. 1–5, 2023.

[262] Y.-L. Hsu, L.-W. Liu, Y.-C. Liao, and H.-C. Chang, "GC-Like LDPC code construction and its NN-aided decoder implementation," *IEEE Open J. Circuits Syst.*, vol. 5, pp. 189–198, 2024.

[263] X. Wu, M. Jiang, and C. Zhao, "Decoding optimization for 5G LDPC codes by machine learning," *IEEE Access*, vol. 6, pp. 50 179–50 186, 2018.

[264] T. Kim and J. Park, "Neural self-corrected min-sum decoder for NR LDPC codes," *IEEE Commun. Lett.*, 2024.

[265] C.-F. Teng, K.-S. Ho, C.-H. Wu, S.-S. Wong, and A.-Y. Wu, "Convolutional neural network-aided bit-flipping for belief propagation decoding of polar codes," *arXiv preprint arXiv:1911.01704*, 2019.

[266] C.-F. Teng and A.-Y. A. Wu, "Convolutional neural network-aided tree-based bit-flipping framework for polar decoder using imitation learning," *IEEE Trans. Signal Process.*, vol. 69, pp. 300–313, 2020.

[267] C.-F. Teng, A. K.-S. Ho, C.-H. D. Wu, S.-S. Wong, and A.-Y. A. Wu, "Convolutional neural network-aided bit-flipping for belief propagation decoding of polar codes," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7898–7902, 2021.

[268] Y. Sun, Y. Shen, W. Song, Z. Gong, X. You, and C. Zhang, "LSTM network-assisted belief propagation flip polar decoder," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, pp. 979–983, 2020.

[269] B. Liu, Y. Xie, and J. Yuan, "A deep learning assisted node-classified redundant decoding algorithm for BCH codes," *IEEE Trans. Commun.*, pp. 1–1, 2020.

[270] Y. Wei, M.-M. Zhao, M.-J. Zhao, and M. Lei, "ADMM-based decoder for binary linear codes aided by deep learning," *IEEE Commun. Lett.*, vol. 24, no. 5, pp. 1028–1032, 2020.

[271] T. Wadayama and S. Takabe, "Deep learning-aided trainable projected gradient decoding for LDPC codes," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 2444–2448, 2019.

[272] ——, "Proximal decoding for LDPC codes," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 106, no. 3, pp. 359–367, 2023.

[273] N. Doan, S. A. Hashemi, and W. J. Gross, "Decoding polar codes with reinforcement learning," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2020.

[274] S. Habib, A. Beemer, and J. Kliewer, "Learned scheduling of LDPC decoders based on multi-armed bandits," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 2789–2794, 2020.

[275] ——, "Belief propagation decoding of short graph-based channel codes via reinforcement learning," *IEEE J. Sel. Inf. Theory*, vol. 2, no. 2, pp. 627–640, 2021.

[276] ——, "Reldec: Reinforcement learning-based decoding of moderate length LDPC codes," *IEEE Trans. Commun.*, vol. 71, no. 19, pp. 5661–5674, 2023.

[277] L. Lugosch and W. J. Gross, "Learning from the syndrome," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, pp. 594–598, 2018.

[278] C.-F. Teng and Y.-L. Chen, "Syndrome-enabled unsupervised learning for neural network-based polar decoder and jointly optimized blind equalizer," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 10, no. 2, pp. 177–188, 2020.

[279] E. Nachmani and Y. Be'ery, "Neural decoding with optimization of node activations," *IEEE Commun. Lett.*, vol. 26, no. 11, pp. 2527–2531, 2022.

[280] C.-F. Teng, C.-H. D. Wu, A. K.-S. Ho, and A.-Y. A. Wu, "Low-complexity recurrent neural network-based polar decoder with weight quantization mechanism," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1413–1417, 2019.

[281] Z. Ibrahim and Y. Fahmy, "Enhanced learning for recurrent neural network-based polar decoder," in *Proc. International Conference on Electrical Engineering (ICEENG)*, pp. 105–109, 2022.

[282] X. Xiao, B. Vasic, R. Tandon, and S. Lin, "Finite alphabet iterative decoding of LDPC codes with coarsely quantized neural networks," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2019.

[283] X. Xiao, B. Vasić, R. Tandon, and S. Lin, "Designing finite alphabet iterative decoders of LDPC codes via recurrent quantized neural networks," *IEEE Trans. Commun.*, vol. 68, no. 7, pp. 3963–3974, 2020.

[284] Y. Lyu, M. Jiang, Y. Zhang, C. Zhao, N. Hu, and X. Xu, "Optimized non-surjective FAIDs for 5G LDPC codes with learnable quantization," *IEEE Commun. Lett.*, vol. 28, no. 2, pp. 253–257, 2023.

[285] M. Lian, C. Häger, and H. D. Pfister, "What can machine learning teach us about communications?" in *Proc. IEEE Information Theory Workshop (ITW)*, pp. 1–5, 2018.

[286] M. Lian, F. Carpi, C. Häger, and H. D. Pfister, "Learned belief-propagation decoding with simple scaling and SNR adaptation," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 161–165, 2019.

[287] Q. Wang, S. Wang, H. Fang, L. Chen, L. Chen, and Y. Guo, "A model-driven deep learning method for normalized min-sum LDPC decoding," in *Proc. IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, 2020.

[288] Q. Wang, Q. Liu, S. Wang, L. Chen, H. Fang, L. Chen, Y. Guo, and Z. Wu, "Normalized min-sum neural network for LDPC decoding," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 1, pp. 70–81, 2022.

[289] J. Dai, K. Tan, Z. Si, K. Niu, M. Chen, H. V. Poor, and S. Cui, "Learning to decode protograph LDPC codes," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 1983–1999, 2021.

[290] Y. Liang, C.-T. Lam, and B. K. Ng, "A low-complexity neural normalized min-sum LDPC decoding algorithm using tensor-train decomposition," *IEEE Commun. Lett.*, vol. 26, no. 12, pp. 2914–2918, 2022.

[291] ——, "Joint-way compression for LDPC neural decoding algorithm with tensor-ring decomposition," *IEEE Access*, vol. 11, pp. 22 871–22 879, 2023.

[292] Y. Cheng, W. Chen, L. Li, and B. Ai, "Rate compatible LDPC neural decoding network: A multi-task learning approach," *IEEE Trans. Veh. Technol.*, vol. 73, no. 5, pp. 7374–7378, 2024.

[293] A. Buchberger, C. Häger, H. D. Pfister, L. Schmalen, and A. G. i Amat, "Pruning and quantizing neural belief propagation decoders," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 1957–1966, 2020.

[294] ——, "Learned decimation for neural belief propagation decoders," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8273–8277, 2021.

[295] V. G. Satorras and M. Welling, "Neural enhanced belief propagation on factor graphs," in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 685–693, 2021.

[296] S. Cammerer, J. Hoydis, F. A. Aoudia, and A. Keller, "Graph neural networks for channel decoding," in *Proc. IEEE Global Communications Conference Workshops (GC Wkshps)*, pp. 486–491, 2022.

[297] K. Tian, C. Yue, C. She, Y. Li, and B. Vucetic, "A scalable graph neural network decoder for short block codes," in *Proc. IEEE International Conference on Communications*, pp. 1268–1273, 2023.

[298] S. K. Ankireddy and H. Kim, "Interpreting neural min-sum decoders," in *Proc. IEEE International Conference on Communications (ICC)*, pp. 6645–6651, 2023.

[299] S. Adiga, X. Xiao, R. Tandon, B. Vasić, and T. Bose, "Generalization bounds for neural belief propagation decoders," *IEEE Trans. Inf. Theory*, vol. 70, no. 6, pp. 4280–4296, 2024.

[300] J. Clausius, M. Geiselhart, D. Tandler, and S. t. Brink, "Graph neural network-based joint equalization and decoding," *arXiv preprint arXiv:2401.16187*, 2024.

[301] R. Wiesmayr, C. Dick, J. Hoydis, and C. Studer, "DUIDD: Deep-unfolded interleaved detection and decoding for MIMO wireless systems," in *Proc. 56th Asilomar Conference on Signals, Systems, and Computers*, pp. 181–188, 2022.

[302] H. Lee, Y.-S. Kil, M. Y. Chung, and S.-H. Kim, "Neural network aided impulsive perturbation decoding for short raptor-like LDPC codes," *IEEE Commun. Lett.*, vol. 11, no. 2, pp. 268–272, 2021.

[303] Y. Wang, S. Zhang, C. Zhang, X. Chen, and S. Xu, "A low-complexity belief propagation based decoding scheme for polar codes-decodability detection and early stopping prediction," *IEEE Access*, vol. 7, pp. 159 808–159 820, 2019.

[304] M. P. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, 1999.

[305] J. Chen, A. Dholakia, E. Eleftheriou, M. P. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, 2005.

[306] John R. Hershey, Jonathan Le Roux, and Felix Weninger, "Deep unfolding: Model-based inspiration of novel deep architectures," *arXiv preprint arXiv:1409.2574*, 2014.

[307] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-based deep learning," *Proceedings of the IEEE*, vol. 11, no. 5, pp. 465–499, 2023.

[308] A. Jagannath, J. Jagannath, and T. Melodia, "Redefining wireless communication for 6G: Signal processing meets deep learning with deep unfolding," *IEEE Trans. Artif. Intell.*, vol. 2, no. 6, pp. 528–536, 2021.

[309] E. Nachmani, E. Marciano, D. Burshtein, and Y. Be'ery, "RNN decoding of linear block codes," *arXiv preprint arXiv:1702.07560*, 2017.

[310] N. Shah and Y. Vasavada, "Neural layered decoding of 5G LDPC codes," *IEEE Commun. Lett.*, vol. 25, no. 11, pp. 3590–3593, 2021.

[311] T. Richardson, S. Kudekar, and L. Vincent, "Adjusted mim-sum decoder," *Patent*, vol. 20, no. 109, 2018.

[312] A. Darabiha, A. C. Carusone, and F. R. Kschischang, "A bit-serial approximate min-sum LDPC decoder and FPGA implementation," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 4–pp, 2006.

[313] F. Angarita, J. Valls, V. Almenar, and V. Torres, "Reduced-complexity min-sum algorithm for decoding LDPC codes with low error-floor," *IEEE Trans. Circuits Syst. I*, vol. 61, no. 7, pp. 2150–2158, 2014.

[314] V. Savin, "Self-corrected min-sum decoding of LDPC codes," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 146–150, 2008.

[315] W. Xu, Z. Wu, Y.-L. Ueng, X. You, and C. Zhang, "Improved polar decoder based on deep learning," in *Proc. IEEE International workshop on signal processing systems (SiPS)*, pp. 1–6, 2017.

[316] W. Xu, X. You, C. Zhang, and Y. Be'ery, "Polar decoding on sparse graphs with deep learning," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, pp. 599–603, 2018.

[317] S. Cammerer, M. Ebada, A. Elkelesh, and S. ten Brink, "Sparse graphs for belief propagation decoding of polar codes," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 1465–1469, 2018.

[318] N. Doan, S. A. Hashemi, E. N. Mambou, T. Tonnellier, and W. J. Gross, "Neural belief propagation decoding of CRC-polar concatenated codes," in *Proc. IEEE International Conference on Communications (ICC)*, pp. 1–6, 2019.

[319] W. Xu, X. Tan, Y. Be'ery, Y.-L. Ueng, Y. Huang, X. You, and C. Zhang, "Deep learning-aided belief propagation decoder for polar codes," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 10, no. 2, pp. 189–203, 2020.

[320] Y. Yu, Z. Pan, N. Liu, and X. You, "Belief propagation bit-flip decoder for polar codes," *IEEE Access*, vol. 7, pp. 10 937–10 946, 2019.

[321] E. Nachmani and L. Wolf, "Hyper-graph-network decoders for block codes," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2329–2339, 2019.

[322] V. K. Chauhan, J. Zhou, P. Lu, S. Molaei, and D. A. Clifton, "A brief review of hypernetworks in deep learning," *arXiv preprint arXiv:2306.06955*, 2023.

[323] E. Nachmani and L. Wolf, "A gated hypernet decoder for polar codes," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5210–5214, 2020.

[324] ——, "Autoregressive belief propagation for decoding block codes," *arXiv preprint arXiv:2103.11780*, 2021.

[325] M. Benammar and P. Piantanida, "Optimal training channel statistics for neural-based decoders," in *Proc. 52nd Asilomar Conference on Signals, Systems, and Computers*, pp. 2157–2161, 2018.

[326] I. Be'Ery, N. Raviv, T. Raviv, and Y. Be'Ery, "Active deep decoding of linear codes," *IEEE Trans. Commun.*, vol. 68, no. 2, pp. 728–736, 2019.

[327] B. Settles, "Active learning literature survey," *Technical Report*, 2009.

[328] T. Raviv, N. Raviv, and Y. Be'ery, "Data-driven ensembles for deep and hard-decision hybrid decoding," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 321–326, 2020.

[329] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, pp. 1–39, 2010.

[330] T. Raviv, A. Goldmann, O. Vayner, Y. Be'ery, and N. Shlezinger, "CRC-aided learned ensembles of belief-propagation polar decoders," *arXiv preprint arXiv:2301.06060*, 2023.

[331] T. Raviv, A. Goldman, O. Vayner, Y. Be'ery, and N. Shlezinger, "CRC-aided learned ensembles of belief-propagation polar decoders," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8856–8860, 2024.

[332] H.-Y. Kwak, D.-Y. Yun, Y. Kim, S.-H. Kim, and J.-S. No, "Boosting learning for LDPC codes to improve the error-floor performance," *arXiv preprint arXiv:2310.07194*, 2023.

[333] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[334] T. R. Halford and K. M. Chugg, "Random redundant soft-in soft-out decoding of linear block codes," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 2230–2234, 2006.

[335] T. Hehn, J. B. Huber, S. Laendner, and O. Milenkovic, "Multiple-bases belief-propagation for decoding of short block codes," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 311–315, 2007.

[336] J. Feldman, "Decoding error-correcting codes via linear programming," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.

[337] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[338] S. Barman, X. Liu, S. C. Draper, and B. Recht, "Decomposition methods for large scale LP decoding," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 7870–7886, 2013.

[339] N. Parikh, S. Boyd *et al.*, "Proximal algorithms," *Foundations and trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[340] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Belief propagation decoding of polar codes on permuted factor graphs," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, 2018.

[341] N. Doan, S. A. Hashemi, M. Mondelli, and W. J. Gross, "On the decoding of polar codes on permuted factor graphs," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2018.

[342] N. Hussami, S. B. Korada, and R. Urbanke, "Performance of polar codes for channel and source coding," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 1488–1492, 2009.

[343] S. Habib and D. G. Mitchell, "Reinforcement learning for sequential decoding of generalized LDPC codes," in *Proc. 12th International Symposium on Topics in Coding (ISTC)*, pp. 1–5, 2023.

[344] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[345] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," *arXiv preprint arXiv:2007.05558*, 2020.

[346] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 2, pp. 239–242, 1990.

[347] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[348] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," *arXiv preprint arXiv:1810.05270*, 2018.

[349] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" *Proceedings of Machine Learning and Systems*, vol. 2, pp. 129–146, 2020.

[350] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks," *Journal of Machine Learning Research*, vol. 22, no. 241, pp. 1–124, 2021.

[351] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. Van Baalen, and T. Blankevoort, "A white paper on neural network quantization," *arXiv preprint arXiv:2106.08295*, 2021.

[352] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision*. Chapman and Hall/CRC, 2022, pp. 291–326.

[353] T. Wadayama and S. Takabe, "Joint quantizer optimization based on neural quantizer for sum-product decoder," *arXiv preprint arXiv:1804.06002*, 2018.

[354] M. Geiselhart, A. Elkelesh, J. Clausius, F. Liang, W. Xu, J. Liang, and S. Ten Brink, "Learning quantization in LDPC decoders," in *Proc. IEEE Global Communications Conference Workshops (GC Wkshps)*, pp. 467–472, 2022.

[355] J. Gao, K. Niu, and C. Dong, "Learning to decode polar codes with one-bit quantizer," *IEEE Access*, vol. 8, pp. 27 210–27 217, 2020.

[356] B. Vasić, X. Xiao, and S. Lin, "Learning to decode LDPC codes with finite-alphabet message passing," in *Proc. Information Theory and Applications Workshop (ITA)*, pp. 1–9, 2018.

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

[358] L. Wang, X. Dong, Y. Wang, L. Liu, W. An, and Y. Guo, "Learnable lookup table for neural network quantization," in *Proc. IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 12 423–12 433, 2022.

[359] T. T. Nguyen-Ly, K. Le, V. Savin, D. Declercq, F. Ghaffari, and O. Boncalo, "Non-surjective finite alphabet iterative decoders," in *Proc. IEEE International Conference on Communications (ICC)*, pp. 1–6, 2016.

[360] L. Wang, S. Chen, J. Nguyen, D. Dariush, and R. Wesel, "Neural-network-optimized degree-specific weights for LDPC minsum decoding," *arXiv preprint arXiv:2107.04221*, 2021.

[361] L. Wang, C. Terrill, D. Divsalar, and R. Wesel, "LDPC decoding with degree-specific neural message weights and RCQ decoding," *IEEE Trans. Commun.*, vol. 72, no. 4, pp. 1912–1924, 2024.

[362] I. V. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[363] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki, "Tensor ring decomposition," *arXiv preprint arXiv:1606.05535*, 2016.

[364] T. Filler and J. Fridrich, "Binary quantization using belief propagation with decimation over factor graphs of LDGM codes," *arXiv preprint arXiv:0710.0192*, 2007.

[365] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. MIT press, 2018.

[366] H. Lee, Y.-S. Kil, M. Jang, S.-H. Kim, O.-S. Park, and G. Park, "Multi-round belief propagation decoding with impulsive perturbation for short LDPC codes," *IEEE Wireless Commun. Lett.*, vol. 9, no. 9, pp. 1491–1494, 2020.

[367] D. Xiao, J. Lu, C. Lin, and B. Jiao, "A perturbation method for decoding LDPC concatenated with CRC," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 667–671, 2007.

[368] X. Han, R. Liu, Y. Li, C. Yi, J. He, and M. Wang, "Accelerating neural BP-based decoder using coded distributed computing," *IEEE Trans. Veh. Technol.*, 2024.

[369] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, 2017.

[370] E. Chen, D. Apalkov, Z. Diao, A. Driskill-Smith, D. Druist, D. Lottis, V. Nikitin, X. Tang, S. Watts, S. Wang *et al.*, "Advances and future prospects of spin-transfer torque random access memory," *IEEE Transactions on Magnetics*, vol. 46, no. 6, pp. 1873–1878, 2010.

[371] H. Chen, K. Zhang, X. Ma, and B. Bai, "Comparisons between reliability-based iterative min-sum and majority-logic decoding algorithms for LDPC codes," *IEEE Trans. Commun.*, vol. 59, no. 7, pp. 1766–1771, 2011.

[372] Y.-H. Liu and D. Poulin, "Neural belief-propagation decoders for quantum error-correcting codes," *Physical Review Letters*, vol. 122, no. 20, p. 200501, 2019.

[373] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes," *arXiv preprint arXiv:0801.1241*, 2008.

[374] A. Gong, S. Cammerer, and J. M. Renes, "Graph neural networkrs for enhanced decoding of quantum LDPC codes," *arXiv preprint arXiv:2310.17758*, 2023.

[375] S. Miao, A. Schnerring, H. Li, and L. Schmalen, "Neural belief propagation decoding of quantum LDPC codes using overcomplete check matrices," in *Proc. IEEE Information Theory Workshop (ITW)*, pp. 215–220, 2023.

[376] S. Cammerer, T. Gruber, J. Hoydis, and S. ten Brink, "Scaling deep learning-based decoding of polar codes via partitioning," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2017.

[377] J. Gao and R. Liu, "Neural network aided SC decoder for polar codes," in *Proc. IEEE 4th International Conference on Computer and Communications (ICCC)*, pp. 2153–2157, 2018.

[378] N. Doan, S. A. Hashemi, and W. J. Gross, "Neural successive cancellation decoding of polar codes," in *Proc. IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, 2018.

[379] I. Wodiany and A. Pop, "Low-precision neural network decoding of polar codes," in *Proc. IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, 2019.

[380] S. A. Hebbar, V. V. Nadkarni, A. V. Makkuva, S. Bhat, S. Oh, and P. Viswanath, "CRISP: Curriculum based sequential neural decoders for polar code family," in *Proc. International Conference on Machine Learning (ICML)*, pp. 12 823–12 845, 2023.

[381] X. Wang, H. Zhang, R. Li, L. Huang, S. Dai, Y. Huangfu, and J. Wang, "Learning to flip successive cancellation decoding of polar codes with lstm networks," in *Proc. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–5, 2019.

[382] B. He, S. Wu, Y. Deng, H. Yin, J. Jiao, and Q. Zhang, "A machine learning based multi-flips successive cancellation decoding scheme of polar codes," in *Proc. IEEE Vehicular Technology Conference (VTC2020-Spring)*, pp. 1–5, 2020.

[383] N. Doan, S. A. Hashemi, F. Ercan, T. Tonnellier, and W. J. Gross, "Neural dynamic successive cancellation flip decoding of polar codes," in *Proc. IEEE International Workshop on Signal Processing Systems (SiPS)*, pp. 272–277, 2019.

[384] ——, "Neural successive cancellation flip decoding of polar codes," *Journal of Signal Processing Systems*, vol. 93, pp. 631–642, 2021.

[385] X. Wang, J. He, J. Li, and L. Shan, "Reinforcement learning for bit-flipping decoding of polar codes," *Entropy*, vol. 23, no. 2, p. 171, 2021.

[386] N. Doan, S. A. Hashemi, F. Ercan, and W. J. Gross, "Fast SC-Flip decoding of polar codes with reinforcement learning," in *Proc. IEEE International Conference on Communications (ICC)*, pp. 1–6, 2021.

[387] S. A. Hashemi, N. Doan, T. Tonnellier, and W. J. Gross, "Deep-learning-aided successive-cancellation decoding of polar codes," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, pp. 532–536, 2019.

[388] N. Doan, S. A. Hashemi, and W. J. Gross, "Fast successive-cancellation list flip decoding of polar codes," *IEEE Access*, vol. 10, pp. 5568–5584, 2022.

[389] C.-H. Chen, C.-F. Teng, and A.-Y. A. Wu, "Low-complexity LSTM-assisted bit-flipping algorithm for successive cancellation list polar decoder," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1708–1712, 2020.

[390] Y. Tao and Z. Zhang, "DNC-aided SCL-flip decoding of polar codes," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2021.

[391] F.-S. Liang, S. Lu, and Y.-L. Ueng, "Deep-learning-aided successive cancellation list flip decoding for polar codes," *IEEE Trans. Cogn. Commun. Netw.*, vol. 10, no. 2, pp. 374–386, 2024.

[392] J. Li, L. Zhou, Z. Li, W. Gao, R. Ji, J. Zhu, and Z. Liu, "Deep learning-assisted adaptive dynamic-SCLF decoding of polar codes," *IEEE Trans. Cogn. Commun. Netw.*, 2024.

[393] Y. Lu, M. Zhao, M. Lei, C. Wang, and M. Zhao, "Deep learning aided SCL decoding of polar codes with shifted-pruning," *China Communications*, vol. 20, no. 1, pp. 153–170, 2023.

[394] X. Liu, S. Wu, Y. Wang, N. Zhang, J. Jiao, and Q. Zhang, "Exploiting error-correction-CRC for polar SCL decoding: A deep learning based approach," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 2, pp. 817–828, 2020.

[395] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast list decoders for polar codes," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 318–328, 2015.

[396] O. Afisiadis, A. Balatsoukas-Stimming, and A. Burg, "A low-complexity improved successive cancellation decoder for polar codes," in *Proc. 48th Asilomar Conf. Signals, Systems and Computers (ACSSC)*, pp. 2116–2120, 2014.

[397] L. Chandesris, V. Savin, and D. Declercq, "Dynamic-SCFlip decoding of polar codes," *IEEE Trans. Commun.*, vol. 66, no. 6, pp. 2333–2345, 2018.

[398] F. Ercan, T. Tonnellier, N. Doan, and W. J. Gross, "Practical dynamic SC-flip polar decoders: Algorithm and implementation," *IEEE Trans. Signal Process.*, vol. 68, pp. 5441–5456, 2020.

[399] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE Commun. Lett.*, vol. 15, no. 12, pp. 1378–1380, 2011.

[400] S. A. Hashemi, C. Condo, and W. J. Gross, "A fast polar code list decoder architecture based on sphere decoding," *IEEE Trans. Circuits Syst. I*, vol. 63, no. 12, pp. 2368–2380, 2016.

[401] ——, "Fast and flexible successive-cancellation list decoders for polar codes," *IEEE Trans. Signal Process.*, vol. 65, no. 21, pp. 5756–5769, 2017.

[402] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[403] M. Rowshan and E. Viterbo, "Improved list decoding of polar codes by shifted-pruning," in *Proc. IEEE Information Theory Workshop (ITW)*, pp. 1–5, 2019.

[404] W. G. Teich, R. Liu, and V. Belagiannis, "Deep learning versus high-order recurrent neural network based decoding for convolutional codes," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, 2020.

[405] Y. Jiang, S. Kannan, H. Kim, S. Oh, H. Asnani, and P. Viswanath, "DeepTurbo: Deep turbo decoder," in *Proc. IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, 2019.

[406] Y. He, J. Zhang, S. Jin, C.-K. Wen, and G. Y. Li, "Model-driven DNN decoder for turbo codes: Design, simulation, and experimental results," *IEEE Trans. Commun.*, vol. 68, no. 10, pp. 6127–6140, 2020.

[407] S. A. Hebbar, R. K. Mishra, S. K. Ankireddy, A. V. Makkuva, H. Kim, and P. Viswanath, "TinyTurbo: Efficient turbo decoders on edge," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 2797–2802, 2022.

[408] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[409] C. Cardinal, D. Haccoun, F. Gagnon, and N. Baatani, "Turbo decoding using convolutional self doubly orthogonal codes," in *Proc. IEEE International Conference on Communications (ICC)*, vol. 1, pp. 113–117, 1999.

[410] M. Mostafa, W. G. Teich, and J. Lindner, "Analysis of high order recurrent neural networks for analog decoding," in *Proc. International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, pp. 116–120, 2012.

[411] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.

[412] X. Chen and M. Ye, "Cyclically equivariant neural decoders for cyclic codes," in *Proc. International conference on machine learning (ICML)*, pp. 1771–1780, 2021.

[413] ——, "Improving the list decoding version of the cyclically equivariant neural decoder," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 2344–2349, 2022.

[414] H. Pirayesh and H. Zeng, "Jamming attacks and anti-jamming strategies in wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 767–809, 2022.

[415] D. Adesina, C.-C. Hsieh, Y. E. Sagduyu, and L. Qian, "Adversarial machine learning in wireless communications using RF data: A review," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 77–100, 2022.

[416] M. Sadeghi and E. G. Larsson, "Physical adversarial attacks against end-to-end autoencoder communication systems," *IEEE Commun. Lett.*, vol. 23, no. 5, pp. 847–850, 2019.

[417] J. Chen, J. Ge, S. Zheng, L. Ye, H. Zheng, W. Shen, K. Yue, and X. Yang, "AIR: Threats of adversarial attacks on deep learning-based information recovery," *IEEE Trans. Wireless Commun.*, 2024.

[418] P. J. Phillips, C. A. Hahn, P. C. Fontana, D. A. Broniatowski, and M. A. Przybocki, "Four principles of explainable artificial intelligence (draft)," *NIST Interagency/Internal Report (NISTIR)*, 2020.

[419] R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman, "Metrics for explainable AI: Challenges and prospects," *arXiv preprint arXiv:1812.04608*, 2018.

[420] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[421] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu, "Explainable AI: A brief survey on history, research areas, approaches and challenges," in *Proc. International Conference on Natural Language Processing and Chinese Computing*, pp. 563–574, 2019.

[422] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable AI: A review of machine learning interpretability methods," *Entropy*, vol. 23, no. 1, p. 18, 2020.

[423] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, "Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[424] D. Minh, H. X. Wang, Y. F. Li, and T. N. Nguyen, "Explainable artificial intelligence: A comprehensive review," *Artificial Intelligence Review*, pp. 1–66, 2022.

[425] R. Dwivedi, D. Dave, H. Naik, S. Singhal, R. Omer, P. Patel, B. Qian, Z. Wen, T. Shah, G. Morgan *et al.*, "Explainable AI (XAI): Core ideas, techniques, and solutions," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–33, 2023.

[426] D. Gunning and D. Aha, "DARPA's explainable artificial intelligence (XAI) program," *AI magazine*, vol. 40, no. 2, pp. 44–58, 2019.

[427] W. Guo, "Explainable artificial intelligence for 6G: Improving trust between human and machine," *IEEE Commun. Mag.*, vol. 58, no. 6, pp. 39–45, 2020.

[428] S. Wang, M. A. Qureshi, L. Miralles-Pechuan, T. Huynh-The, T. R. Gadekallu, and M. Liyanage, "Applications of explainable AI for 6G: Technical aspects, use cases, and research challenges," *arXiv preprint arXiv:2112.04698*, 2021.

[429] S. Wang, M. A. Qureshi, L. Miralles-Pechuán, T. Huynh-The, T. R. Gadekallu, and M. Liyanage, "Explainable AI for 6G use cases: Technical aspects and research challenges," *IEEE Open J. Commun. Soc.*, vol. 5, pp. 2490–2540, 2024.

[430] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[431] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," *arXiv preprint arXiv:1906.02243*, 2019.

[432] D. Patterson, J. Gonzalez, U. Hölzle, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. R. So, M. Texier, and J. Dean, "The carbon footprint of machine learning training will plateau, then shrink," *Computer*, vol. 55, no. 7, pp. 18–28, 2022.

[433] C.-J. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. Aga, J. Huang, C. Bai *et al.*, "Sustainable AI: Environmental implications, challenges and opportunities," *Proceedings of Machine Learning and Systems*, vol. 4, pp. 795–813, 2022.

[434] S. Salehi and A. Schmeink, "Data-centric green artificial intelligence: A survey," *IEEE Trans. Artif. Intell.*, vol. 5, no. 5, pp. 1973–1989, 2024.

[435] G. Menghani, "Efficient deep learning: A survey on making deep learning models smaller, faster, and better," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–37, 2023.

[436] M. H. Jarrahi, A. Memariani, and S. Guha, "The principles of data-centric AI (DCAI)," *arXiv preprint arXiv:2211.14611*, 2022.

[437] D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, and X. Hu, "Data-centric AI: Perspectives and challenges," in *Proc. SIAM International Conference on Data Mining (SDM)*, pp. 945–948, 2023.

[438] D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, Z. Jiang, S. Zhong, and X. Hu, "Data-centric artificial intelligence: A survey," *arXiv preprint arXiv:2303.10158*, 2023.

[439] P. Botsinis, D. Alanis, Z. Babar, H. Nguyen, D. Chandra, S. X. Ng, and L. Hanzo, "Quantum algorithms for wireless communications," *IEEE Communications Surveys & Tutorials*, 2018.

[440] T. Matsumine, T. Koike-Akino, and Y. Wang, "Channel decoding with quantum approximate optimization algorithm," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 2574–2578, 2019.

[441] J. Cui, Y. Xiong, S. X. Ng, and L. Hanzo, "Quantum approximate optimization algorithm based maximum likelihood detection," *IEEE Trans. Commun.*, vol. 70, no. 8, pp. 5386–5400, 2022.

[442] K. Yukiyoshi and N. Ishikawa, "Quantum search algorithm for binary constant weight codes," *arXiv preprint arXiv:2211.04637*, 2022.

[443] M. Kim, S. Kasi, P. A. Lott, D. Venturelli, J. Kaewell, and K. Jamieson, "Heuristic quantum optimization for 6G wireless communications," *IEEE Network*, vol. 35, no. 4, pp. 8–15, 2021.

[444] S. Kasi, J. Kaewelh, and K. Jamieson, "A quantum annealer-enabled decoder and hardware topology for nextg wireless polar codes," *IEEE Trans. Wireless Commun.*, vol. 23, no. 4, pp. 3780–3794, 2023.

[445] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[446] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, no. 1, p. 4213, 2014.

[447] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[448] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke *et al.*, "Noisy intermediate-scale quantum algorithms," *Reviews of Modern Physics*, vol. 94, no. 1, p. 015004, 2022.

[449] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio *et al.*, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[450] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, pp. 172–185, 2015.

[451] A. Perdomo-Ortiz, M. Benedetti, J. Realpe-Gómez, and R. Biswas, "Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers," *Quantum Science and Technology*, vol. 3, no. 3, p. 030502, 2018.

[452] S. J. Nawaz, S. K. Sharma, S. Wyne, M. N. Patwary, and M. Asaduzzaman, "Quantum machine learning for 6G communication networks: State-of-the-art and vision for the future," *IEEE Access*, vol. 7, pp. 46 317–46 350, 2019.

[453] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, "Power of data in quantum machine learning," *Nature Communications*, vol. 12, no. 1, p. 2631, 2021.

[454] O. Simeone *et al.*, "An introduction to quantum machine learning for engineers," *Foundations and Trends® in Signal Processing*, vol. 16, no. 1-2, pp. 1–223, 2022.

[455] B. Narottama, Z. Mohamed, and S. Aïssa, "Quantum machine learning for next-g wireless communications: Fundamentals and the path ahead," *IEEE Open J. Commun. Soc.*, vol. 4, pp. 2204–2224, 2023.

[456] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner, "The power of quantum neural networks," *Nature Computational Science*, vol. 1, no. 6, pp. 403–409, 2021.

[457] M. Henderson, S. Shakya, S. Pradhan, and T. Cook, "Quanvolutional neural networks: Powering image recognition with quantum circuits," *Quantum Machine Intelligence*, vol. 2, no. 1, p. 2, 2020.

[458] J. Romero, J. P. Olson, and A. Aspuru-Guzik, "Quantum autoencoders for efficient compression of quantum data," *Quantum Science and Technology*, vol. 2, no. 4, p. 045001, 2017.

[459] P.-L. Dallaire-Demers and N. Killoran, "Quantum generative adversarial networks," *Physical Review A*, vol. 98, no. 1, p. 012324, 2018.

[460] S. Lloyd and C. Weedbrook, "Quantum generative adversarial learning," *Physical Review Letters*, vol. 121, no. 4, p. 040502, 2018.