Machine Learning for Physical Layer of Communication Systems: A Survey

Touseef Ali[1] (1217395244), **Mohammed Aladsani**[1](121340446),
Hitesh Kumar[1](1217397623), **Swarnim Sinha**[2](1217390967)
[1]School of Electrical, Computer, and Energy Engineering
[2]School of Computing, Informatics, and Decision System Engineering
Arizona State University, Tempe, Az 85281
{tali4, maladsan, hkumar20, ssinha39}@asu.edu

Abstract

There has been a tremendous increase in the demand for high-speed as well as the number of users for wireless data communications in the past few years. The proliferation of these requirements will soon call for more robust techniques in communication systems. Recent developments in the field of Machine Learning (ML) have shown encouraging results in various applications including communications. The aim of this work is to present a survey on the applications of ML algorithms in communication systems. The focus of this work is the use of ML algorithms in the physical layer problems of communication systems which include beamforming, channel estimation, modulation/demodulation and detection. The dynamic nature of problems in physical layer requires the algorithms to be more adaptive. This work highlights the performance improvements in solving these problems using ML techniques as compared to conventional methods. Challenges and limitation of ML techniques in communications domain are also discussed.

1 Introduction

Physical layer act as an interface between the higher layers of communication system and the channel which is the logical connection between transmitter and receiver enabling exchange of information. Physical layer is designed to mitigate the effects of a channel. Channel modeling is essential to highly mobile communication systems to ensure high data rate transmissions [1]. Conventional techniques of channel estimation include statistical modeling whcih require tremendous amount of measurement data. The emerging communication standards like 5G and internet of things require efficient channel modeling methodologies. Deep learning techniques have shown some promising results in processing big data. In this work, we review some of the seminal papers that utilizes deep learning techniques in solving physical layer communication problems. In section 2, we discuss end-to-end communication system modeling [2,3]. A review of the work by O'Shea et al.[2], one of the highly cited papers on the subject matter, is presented. In section 3, we review paper related to beamforming using Reconfigurable Intelligent Surface (RIS) [4]. A review on Convolutional Neural Network based Automatic modulation classification [5] is presented in section 4. Finally in section 5, we discuss challenges in applying deep learning techniques to physical layer problems in communication systems and also enlist some other ML based frameworks towards this domain.

2 Channel and End-to-End Communication Modeling

Consider an end-to-end SISO communication system [2,3] in which the transmitter is sending one out of M messages to a receiver over a channel. Each message is selected from the set $\mathbb{M} = \{1, \cdots, M\}$

with uniform probability. The transmitter generates the signal $\mathbf{x} = f_{\boldsymbol{\theta}_T}(m)$ which is the channel and line encoding of the selected message $m \in \mathbb{M}$. The transmitted signal $\mathbf{x}$ after passing through the stochastic channel generates $\mathbf{y}$ where the distribution of $\mathbf{y}$ is conditional on $\mathbf{x}$ i.e. $\mathbf{y} \sim P(\mathbf{y}|\mathbf{x})$. The receiver applies channel and line decoding using the transformation $f_{\boldsymbol{\theta}_R}(\mathbf{y})$ to the received signal $\mathbf{y}$ to get the estimate $\hat{m}$ of the transmitted message m. Here, $\boldsymbol{\theta}_T$ and $\boldsymbol{\theta}_R$ denotes the parameters in the transmitter and receiver, respectively. Figure 1 shows the block diagram of a typical communication system.

Figure 1: Typical Communication System

O'Shea et al.[2] models the end-to-end communication system using NN based autoencoder approach. For simplicity, we consider $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. The goal of the autoencoder is to reconstruct the transmitted signal at the receiver with minimal probability of error. The NNs in this autoencoder learn robust line and channel encoding and decoding transformation functions that mitigate the effects of the channel layer. The input to the transmitter end of the autoencoder is modeled as M-dimensional standard basis column vector $\mathbf{e}_m$ with 1 at the mth position and zero at other positions. Channel effects are modeled as additive white Gaussian noise layer $\sim \mathcal{N}(0, \nu) = \mathcal{N}(0, (\frac{2n}{k}\frac{E_b}{N_0})^{-1})$, where $k = log_2(M)$ is the number of bits for each message and E_b/N_0 is the SNR per bit or energy per bit to noise power spectral density ratio. The complete architecture of the autoencoder is depicted in Table 1.

Layer	**Output Dimensions**
Transmitter:	-
Input	M
Dense+ReLU	M
Dense+Linear	n
Normalization Layer	n
Channel:	-
Noise	n
Receiver:	-
Dense+ReLU	M
Dense+softmax	M

Table 1: Layout of the Autoencoder for end-to-end communications

The output of the autoencoder at the receiver end is a probability vector $\mathbf{p}$ that gives the probability for each message in $\mathbb{M}$. We can get an estimate $\hat{m}$ of the message m from the $\mathbf{p}$. Training is done on the S training samples using SGD on all possible messages m to minimize the categorical cross entropy loss function:

$$L = \frac{1}{S}\sum_{k=1}^{S} l(\mathbf{p}^{(k)}, m^{(k)}) \tag{1}$$

where $l(\mathbf{p}^{(k)}, m^{(k)}) = -log(p_m^{(k)})$ is the categorical cross-entrophy for kth input message sample in the training set. Next, the idea is extended to MIMO communication systems. We consider the simple case in which there are two transmitters and two receivers using the same channel to communicate. Generalization to more than two transmitters and receivers is straightforward. Consider Transmitter i communicating message m_i to receiver i, where $i \in \{1, 2\}$. The transmitted signals $\mathbf{x}_i \in \mathbb{C}^n$after passing through the AWGN channel interferes with each other at the receivers. The received signals

at each receiver can be expressed as:

$$\begin{aligned} \mathbf{y}_1 &= \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{n}_1 \\ \mathbf{y}_2 &= \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{n}_2 \end{aligned} \tag{2}$$

Where $\mathbf{n}_1, \mathbf{n}_2 \sim \mathcal{CN}(0, (\frac{2n}{k}\frac{E_b}{N_0})^{-1}\mathbf{I}_n) \in \mathbb{C}^n$. Two coupled autoencoder layers are designed similar to SISO case where the only differences are received signals at each receiver are modeled as relationships above and each complex vector is modeled as a single long vector with real and imaginary parts stacked on top of each other. Hence, n in Table 1 are replaced with $2n$. For total S_t training samples in the minibatch t, the cross entropy loss functions for each transmitter receiver pair are given by:

$$\begin{aligned} \tilde{L_1} &= \frac{1}{S_t}\sum_{k\in t} l_1(\mathbf{p}_1^{(k)}, m_1^{(k)}) \\ \tilde{L_2} &= \frac{1}{S_t}\sum_{k\in t} l_2(\mathbf{p}_2^{(k)}, m_2^{(k)}) \end{aligned} \tag{3}$$

where $l_i(\mathbf{p}_i^{(k)}, m_i^{(k)})$ is the categorical cross entropy for ith transmitter-receiver pair while sending kth message in the minibatch t. To train the coupled autoencoders together, the weighted sum of both the loss functions are minimized, i.e. $L = \alpha L_1 + (1-\alpha)L_2$. The authors used dynamic weight, α_t for minibatch t, given by:

$$\alpha_{t+1} = \frac{\tilde{L}_1}{\tilde{L_1} + \tilde{L_2}} \tag{4}$$

Initially $\alpha_0 = 0.05$ is selected. The authors compared the Block Error Rate (BER) versus E_b/N_0 curves for autoencoder approach to coventional Hamming codes with binary hard-decision decoding or Maximum Likelihood decoding (MLD) for different parameters (n, k) i.e. for n channel uses to transmit k-bits for each message, where $k = log_2(M)$. The results for SISO case are reproduced in Figure 2(a) [1].

While O'Shea et al.[2] discovered a basis for using deep learning to learn a physical layer scheme for end to end communications systems, its biggest shortcoming while implementing it practically is the requirement of knowledge of the channel model or gradient of the instantaneous channel transfer function. Because of its dynamic nature, channel model is seldom known. Additionally, channel effects also include quantization of the received signal which are nondifferentiable. This also hinders the use of back propagation based gradient descent while training. To address this problem, Aoudia et al.[3] introduced reinforcement learning based algorithm for training end to end comm system which does not require knowledge of the channel model. The idea is to first train the receiver using supervised learning on observations only and then train the transmitter based on an estimated gradient loss using RL.

To train the receiver, the transmitter generates channel encoded complex signal $\mathbf{x}$ for each message $m \in \mathbb{M}$ in the minibatch t. The transmitted signal is altered by the channel effects to get $\mathbf{y}$ for each example which is the input to the receiver. The receiver generates a probability distribution over $\mathbb{M}$ for every example. Next, optimization is performed using SGD on the cross-entropy loss function:

$$L(\boldsymbol{\theta}_R) = \frac{1}{S}\sum_{k=1}^{S} l(f_{\boldsymbol{\theta}_R}(\mathbf{y}^{(k)}), m^{(k)}) \tag{5}$$

The transmitter is trained next using RL to come up with a channel encoding scheme that minimizes the scalar loss provided by the receiver. In this case, set $\mathbb{M}$ is the state space and $\mathbf{x} \in \mathbb{C}^n$ (input to the channel) is the action space. The policy π for this case is chosen to probability distribution over $\mathbf{x}$ conditioned over m. For each training example, the transmitter function generates channel encoded symbols $\mathbf{x}$. Next these symbols are sampled as per the distribution of policy to get $\mathbf{x}_p$. For Gaussian policy:

$$\begin{aligned} \mathbf{x}_p &= \sqrt{1-\sigma_\pi^2}\mathbf{x} + \mathbf{w} \\ &= \sqrt{1-\sigma_\pi^2} f_{\theta_T}(m) + \mathbf{w} \end{aligned} \tag{6}$$

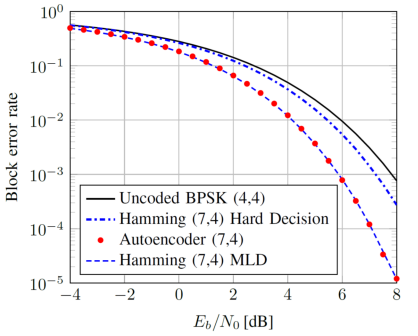
Where $\mathbf{w}$ is zero mean complex Gaussian vector i.e. $\mathbf{w} \sim \mathcal{CN}(\mathbf{0}, \sigma_\pi^2\mathbf{I})$. Hence,

$$\pi(\mathbf{x}_p | f_{\boldsymbol{\theta}_T}(m)) = \frac{1}{(\pi\sigma_\pi^2)^n} exp\Big(-\frac{\|\mathbf{x}_p - \sqrt{1-\sigma_\pi^2} f_{\boldsymbol{\theta}_T}(m)\|_2^2}{\sigma_\pi^2}\Big) \tag{7}$$

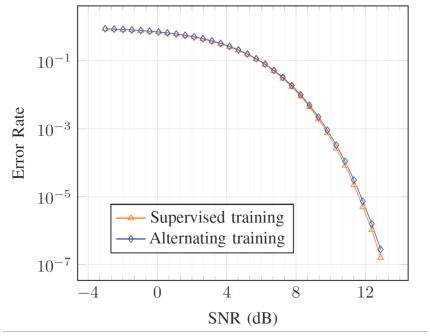
Next, $\mathbf{x}_p$ is sent over the channel to get $\mathbf{y}$ at the receiver. The receiver generates probability distribution over $\mathbb{M}$ and computes the per-example for each training example. The per-example losses are sent to the transmitter that performs optimization using SGD. The gradient of the loss function for the optimization step can be expressed as:

$$\nabla_{\boldsymbol{\theta}_T} L(\boldsymbol{\theta}_T) = \frac{1}{S} \sum_{k=1}^{S} l(f_{\boldsymbol{\theta}_R}(\mathbf{y}^{(k)}), m^{(k)}) \nabla_{\boldsymbol{\theta}_T} log(\pi(\mathbf{x}_p^{(k)} | f_{\boldsymbol{\theta}_T}(m^{(k)}))) \tag{8}$$

Aoudia et al.[3] showed that the algorithm performance is the same as the autoencoder proposed by O'Shea et al.[2]. The results from [3] are reproduced in Figure 2 (b).



(a) Comparison of Autoencoder with conventional techniques[2]



(b) Comparison of Supervised (Autoencoder) versus Alternating RL technique[3]

Figure 2: BER vs E_b/N_0 curves

3 Beamforming

A paramount task in communication systems is beamforming. The problem of determining the required parameters for the transmitter to steer the beam in the desired direction is highly nonlinear. Massive MIMO arrays are considered as enabling paradigm in communication standards like 5G. The computation complexity in such arrays is of the order of N^2 where N is the number of array elements [12]. Thus, it became an interesting research topic to employ deep learning in this kind of problem. One of the tools used in beamforming is Reconfigurable Intelligent Surface (RIS). RIS consist of a set of planar elements which are designed such that their reflective phase information exploit electromagnetic interaction to control the incident beam and then reflect it in the desired direction. Analytically the required phase information needed to steer a reflected beam can be given by:

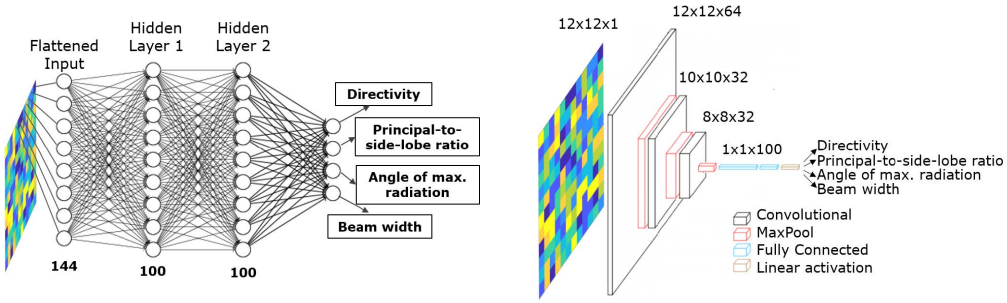
$$\Phi_{Reflected}(\theta,\phi) = \sum_{m=1}^{M}\sum_{n=1}^{N} \Phi_{Incident}(\theta,\phi) \cdot |\Gamma_{mn}| \cdot e^{j\phi_{mn}} \cdot e^{jkd_x(m-1)sin\theta cos\phi} \cdot e^{jkd_y(n-1)sin\theta sin\phi} \tag{9}$$

where $\Phi_{Reflected}(\theta,\phi)$ and $\Phi_{Incident}(\theta,\phi)$ denote the reflected and incident beam patterns, respectively, Γ_{mn} and ϕ_{mn} are the tunable parameters of RIS needed to control the beam. Because of the nonlinear relationship of the tunable parameters to the reflected beam pattern, tuning these parameters in polynomial time is a daunting task. In the paper [4], the authors applied Neural networks to determine the reflection beam characteristics of RIS given a set of tuning parameters of the RIS. One important aspect in applying ML in physical engineering problems is formatting the required input data which is usually acquired from simulations, equations, measurements, or any combination of these to be fit for the input of the desired ML approach. Sometimes the type of data could dictate what ML approach to be used. In the beamforming problem, the tuning parameters are complex in nature while ML algorithms like deep learning require real inputs to operate. Hence the data is to pre-formated. One popular approach to represent complex data is by using coding bits [13]. For

example, a complex value of e^{j0} is denoted as "0" bit and $e^{j\pi}$ is denoted as "1" bit and the higher the bit count, the more dense complex values can be accommodated. This transforms a complex data matrix into a binary matrix of data that can be fed into ML algorithm. Another concern in this problem is the data generation for training. In this paper, the authors opted to use data calculated from the aforementioned analytical equation since generating sufficient samples using simulation software packages such as commercial EM full wave solvers tend to be time consuming. Generating samples using physical measurements, which may also require pre and post calibrations, is even more tedious and expensive. Fortunately, the analytical equation predicts the beam characteristics of interest well enough. Moreover, the equation has an FFT structure that can be exploited to reduce computation time. The authors in [4] chose two Neural networks approaches: Multiple-layer perceptron neural network (MLPNN) and Convolutional neural network (CNN). MLPNN is simpler to implement but it requires further preformatting of the input data into a vector instead of a matrix, where the CNN takes the matrix itself an input which is a natural representation of the desired tuning parameter. The loss function used for training the MLPNN is given by

$$L(\mathbf{X}, \mathbf{y}) = \text{MSE}(\mathbf{X}, \mathbf{y}) + \lambda \sum_{i=0}^{N-1} w_i^2 \tag{10}$$

where $\mathbf{X}$ is the input image (target radiation pattern), $\mathbf{y}$ is the set of target metrics (directivity, principle-to-sidelobe ratio, angle of maximum radiation and beamwidth), λ is the regularization parameter and w_i are the weights of the MLPNN in all the layers. The schematics for both the approaches are depicted in Figure. 3. For both the approaches, the network parameters are chosen after an extensive user-driven search, since going throughout all the possible combinations is not computationally feasible. The authors[4] showed that both approaches produced similar predictions however CNN is favored due to reduced computation complexity especially when the elements of RIS is large.



(a) MLPNN Approach

(b) CNN Approach

Figure 3: ML based Beamforming approaches

4 Automatic Modulation Classification (AMC)

Automatic modulation classification/recognition (AMC) is the accurate identification of modulation scheme of a received signal in non-cooperative communication systems with insufficient or lack of prior knowledge. AMC is required to accurately analyze the information in a received signal. AMC has many civilian and military applications. Convential AMC techniques can be broadly classified into two categories: maximum likelihood based AMC (ML-AMC) and feature based AMC. The former being more optimal albeit computationally expensive than the later. Feature based AMC has three stages: data preprocessing , feature extraction and classification. The evolution of deep learning has led to the use of CNN for feature extraction in feature based AMC, called CNN-AMC [5], which not only makes an approximation of the optimal ML-AMC with minimal loss but also improvement in speed. This also makes feature based AMC more efficient in low signal-to-noise scenarios. The optimal ML-AMC can be considered as a benchmark of estimation performance for CNN-AMC.

Consider a system that receives N time samples $\mathbf{y} = [y_0, \cdots, y_{N-1}]^T \in \mathbb{C}^N$ of a signal with unknown modulation scheme. The complex envelope of this signal is given by

$$y(t) = x(t; \mathbf{u}_k) + n(t) \tag{11}$$

where $n(t) \sim \mathcal{CN}(0, N_0)$ is the system noise and $\mathbf{u}_k$ is a set of multidimensional parameters under kth modulation scheme, $k \in \{1, \cdots, C\}$. These parameters include signal variables like signal amplitude, propagation phase offset, constellation points under kth modulation scheme, symbol interval and residual carrier frequency, and channel variables. The channel is considered as LTI and frequency-flat i.e., no fading effects are assumed. This assumption leads to samples in $\mathbf{y}$ being I.I.D. This favours use of CNN as compared to MLPNN since the convolution kernels remain the same for signal segments in different time slots. The complex vector $\mathbf{y}$ is transformed into real matrix $\mathbf{Y} \in \mathbb{R}^{2\times N}$ with $\text{Re}(\mathbf{y})$ as first and $\text{Im}(\mathbf{y})$ as second row. Let H_k be the hypothesis that kth modulation is used in $\mathbf{y}$, then we define the likelihood vector $\boldsymbol{\xi}^N = [\xi_1^N, \cdots, \xi_C^N]^T$, where $\xi_k^N = P_{cc}(\mathbf{y}|H_k)$ is the probability of correctly classifying the modulation scheme and $\sum_{k=1}^{C} \xi_k^N = 1$.

The input to the learning model has two parts , one part is observation sequence $\mathbf{Y} \in \mathbb{R}^{2\times 1000}$ and another is an estimated SNR. First part is a matrix which needs to be convoluted to gain higher level information while the second part, SNR, is a scalar quantity. The architecture for CNN-AMC [5] is depicted in Figure 4. The categorical cross-entropy loss function, using batch size of N_b, for this architecture is given by

$$L(\theta) = -\frac{1}{N_b} \sum_{i=1}^{N_b} \boldsymbol{\xi}_i^N \log \hat{\boldsymbol{\xi}}_i^N + (1 - \boldsymbol{\xi}_i^N) \log(1 - \hat{\boldsymbol{\xi}}_i^N) \tag{12}$$

SGD is used to minimize the above loss function.

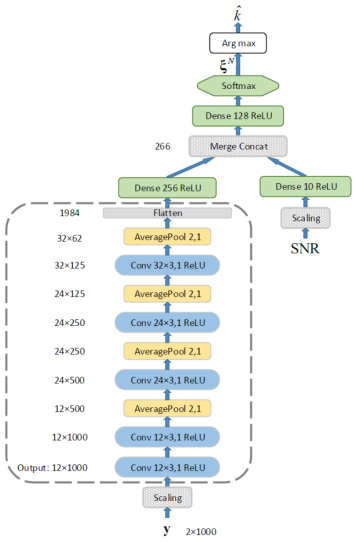


Figure 4: CNN-AMC Architecture [5]

To expedite the process of training, two step training approach is adopted. First, pretraining of the CNN-AMC and then fine-tuning the weights. In the pre-training phase of the training, first, training data is generated including Gaussian noise samples. Additional label is added for the Gaussian noise. Next, the CNN-AMC parameters are initialized with random values. Finally, CNN-AMC is trained and model parameters are stored with minimal validation loss. Fine-tuning phase of training is carried out next. First, training data is generated and corresponding target labels are produced using ML-AMC. Next, the top-layer of CNN-AMC is replaced with randomly initialized values. Stored model parameters from the first phase are loaded for the rest of the parameters of the CNN-AMC. Finally, CNN-AMC is fully trained and model parameters are stored for later use.

When the environment or the modulation set or scheme is changed, transfer learning can be applied instead of retraining the model completely. Even if the environment changes from coherent to incoherent, the first step of two step training phase can be skipped which significantly reduces the training time of the model. A comparison of CNN-AMC and ML-AMC is reproduced in Figure 5[5]. The simulation results show that CNN-AMC produced results that are very close to that of ML-AMC which is considered as optimum and is about 40x to 1700x faster than ML-AMC.

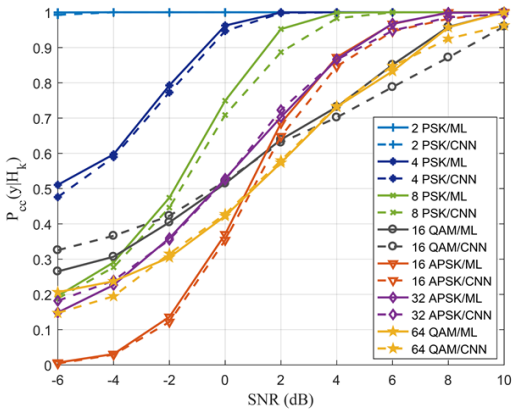


Figure 5: Using CNN-AMC and ML-AMC to correctly classify various modulation schemes [5]

5 Discussions and Conclusions

Development of Machine Learning (ML) frameworks for communications domain is in its infant stage. Some of the challenges in using deep learning for communications are listed below:

- Machine Learning algorithms are generally trained on open datasets and their performance is compared to common benchmarks. No such provisions exist for communication systems.
- Many signal processing tasks in communication systems involve complex numbers and related operations. Currently available deep learning libraries do not support complex numbers.
- Most of the problems in communication systems suffer from the "curse of dimensionality". For example in section 2, if possible messages $M = 2^{100}$ then the training complexity becomes ridiculously high.

Some other notable contributions, surely not exhaustive, are mentioned below:

- Zhang et al.[6] proposed Random Forest and K-Nearest Neighbor (KNN) based models for path loss of aeronautical channel.
- Shiva et al.[7] used deep learning based algorithms to estimate the virtual angular beam (i.e. angle-of-departure (AoD) of the dominant propagation paths) at the user end and towards the Base Station (BS) receiver of cellular network.
- Huang et al.[8] proposed deep learning architecture for estimation angle of arrival of a received signal.
- In [9] the authors estimate channel model using KPowerMeans clustering algorithm [10] to group multipath components based on delay and angular characteristics.
- Alkhateeb et al.[11] proposed deep learning based novel beamforming technique that overcomes several problems in mmWave mobile communications systems.

References

[1] Tse, David, and Pramod Viswanath. Fundamentals of wireless communication. Cambridge university press, 2005.

[2] O'Shea, Timothy, and Jakob Hoydis. "An introduction to deep learning for the physical layer." IEEE Transactions on Cognitive Communications and Networking 3.4 (2017): 563-575.

[3] Aoudia, Fayçal Ait, and Jakob Hoydis. "End-to-end learning of communications systems without a channel model." 2018 52nd Asilomar Conference on Signals, Systems, and Computers. IEEE, 2018.

[4] Taghvaee, Hamidreza, et al. "Radiation pattern prediction for Metasurfaces: A Neural Network based approach." arXiv preprint arXiv:2007.08035 (2020).

[5] Meng, Fan, et al. "Automatic modulation classification: A deep learning enabled approach." IEEE Transactions on Vehicular Technology 67.11 (2018): 10760-10772.

[6] Zhang, Yan, et al. "Air-to-air path loss prediction based on machine learning methods in urban environments." Wireless Communications and Mobile Computing 2018 (2018).

[7] Navabi, Shiva, et al. "Predicting wireless channel features using neural networks." 2018 IEEE international conference on communications (ICC). IEEE, 2018.

[8] Huang, Hongji, et al. "Deep learning for super-resolution channel estimation and DOA estimation based massive MIMO system." IEEE Transactions on Vehicular Technology 67.9 (2018): 8549-8560.

[9] Ko, Junghoon, et al. "Millimeter-wave channel measurements and analysis for statistical spatial channel model in in-building and urban environments at 28 GHz." IEEE Transactions on Wireless Communications 16.9 (2017): 5853-5868.

[10] Czink, Nicolai. The Random-Cluster Model; a stochastic MIMO channel model for broadband wireless communication systems of the 3rd generation and beyond. Diss. 2007.

[11] Alkhateeb, Ahmed, et al. "Deep learning coordinated beamforming for highly-mobile millimeter wave systems." IEEE Access 6 (2018): 37328-37348.

[12] Yang, Huanhuan, et al. "A programmable metasurface with dynamic polarization, scattering and focusing control." Scientific reports 6 (2016): 35692.

[13] Shan, Tao, et al. "Coding programmable metasurfaces based on deep learning techniques." IEEE Journal on Emerging and Selected Topics in Circuits and Systems 10.1 (2020): 114-125.