IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 5, 2021, accepted November 15, 2021, date of publication November 16, 2021,
date of current version December 21, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3128813

Parameter Estimation and Classification via Supervised Learning in the Wireless Physical Layer

KYLE W. MCCLINTICK[1], (Student Member, IEEE), GALAHAD M. WERNSING[1], (Student Member, IEEE), PAULO VICTOR R. FERREIRA[2], (Member, IEEE), AND ALEXANDER M. WYGLINSKI[1], (Senior Member, IEEE)
[1]Worcester Polytechnic Institute, Worcester, MA 01609, USA
[2]Cohu, Inc., Norwood, MA 02062, USA

Corresponding author: Kyle W. McClintick (kwmcclintick@wpi.edu)

ABSTRACT Emerging wireless networks possess the potential to achieve levels of connectivity and Quality-of-Service (QoS) that are orders of magnitude higher than today's networks. Realizing the potential of these networks will require flexible, low cost, and accurate Digital Signal Processing (DSP). Supervised Learning (SL) models employing unknown parameter estimation and classification techniques have experienced widespread use in physical (PHY) layer wireless communication systems since they can achieve low costs via inexpensive forward-pass computations, attain flexible operations due to trainable parameters, and yield accurate results based on the universal approximator attribute. In this survey and tutorial paper, we present a methodical explanation of how SL can be applied to unknown parameter estimation and classification across several different PHY layer components of a wireless communications system. Additionally, via a survey and comparison of popular methods, this paper provides insights on how to perform weight training, weight initialization, loss function regularization, data pre-processing, input feature design, ensemble training, and hyper-parameter validation. In our review of state-of-the-art works, we found significant use of SL algorithms in the following PHY layer applications: Dynamic Spectrum Access (DSA), channel corrections, Automatic Gain Control (AGC), Multiple Input Multiple Output (MIMO) control, Analog to Digital (ADC) conversion, and Automatic Coding and Modulation (ACM). The overarching goal of this survey and tutorial paper is to assist the reader in understanding the motivation and methodologies associated with various SL algorithms applied to PHY layer DSP operations, as well as to provide the reader with the necessary tools and techniques needed for addressing open challenges to be experienced by future wireless networks.

INDEX TERMS Digital signal processing, parameter estimation and classification, physical layer, machine learning, supervised learning, wireless communications.

I. INTRODUCTION

Parameter estimation and classification in the presence of noise is a ubiquitous problem in wireless signal processing. Machine Learning (ML) models have a successful performance history of predicting these unknown parameters in real-world wireless systems with few assumptions about data. However, only recently has the use of ML models become practical, due to faster hardware leveraging graphics processing units and cloud computing [1], a greater amount of data from increasingly permeating wireless networks collected by cheaper sensors [2], and improved algorithms that train faster [3] while performing more accurate predictions [4].

ML-based wireless signal processing solutions tout a number of advantages. These include the ability to implement real-world Cognitive Radio (CR) networks with dynamic spectrum access capabilities [6], Wireless Sensor Networks (WSN) with less re-programming and longer service life [7], Self Organizing Networks (SON) with robust routing protocols [9], and more secure Internet of Things (IoT) networks that fully utilize resources to stay online longer [11].

To provide a useful resource for the wireless communications community, we provide a comprehensive tutorial for the trial-and-error approach of designing SL models for the PHY layer used in unknown parameter estimation and

The associate editor coordinating the review of this manuscript and approving it for publication was Derek Abbott.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

TABLE 1. A comparison of surveys and tutorials discussing SL in wireless communications.

	References	Machine Learning Models							Machine Learning Concepts									Probabilistic Concepts			
		LSR	SR	SVM	CNN	RNN	DT	KNN	WU	WI	R	DC	DPP	DA	V	KM	IE	MLE	NLL	SGD	KKT
MAC layers and above	[5]		✓		✓	✓															
	[6]		✓	✓												✓			✓	✓	✓
	[7]		✓	✓			✓	✓					✓			✓					
	[8]		✓	✓			✓	✓								✓					
	[9]		✓	✓			✓	✓													
	[10]				✓	✓	✓						✓		✓						
	[11]		✓	✓	✓	✓							✓				✓		✓		
	[12]		✓		✓			✓											✓	✓	
	[13]	✓	✓	✓				✓					✓			✓		✓	✓	✓	✓
	[14]																				
	[15]	✓	✓	✓			✓	✓					✓	✓	✓						
	[16]		✓	✓	✓	✓			✓								✓			✓	
	[17]																				
PHY	[18]		✓		✓						✓	✓		✓			✓			✓	
	This work	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

classification. A great need exists for such a resource, as virtually every system and problem requires either an alteration to an existing SL model or the design of an entirely new one. Furthermore, these models almost never function at an acceptable level on their first iteration, and require iterative development stages driven by a combination of intuition and trial-and-error. Finally, the data used by each model is unique to each system and problem, such that data collection must be constantly performed as new wireless systems are created or more widely deployed.

In this work, we first present the history, derivations, and an assortment of definitions related to Neural Networks (NNs) employed in the PHY layer of a typical wireless communication system. A survey of PHY layer SL works is conducted by discussing the importance of each PHY layer process that has experienced significant SL research, how those processes are performed with and without the use of SL algorithms, as well as how those SL algorithms have advanced the current state-of-the art of those processes and their associated remaining open challenges. From each PHY layer process, one work is selected to serve as a subject for a collection of tutorials in which we describe the methodology behind these works, present the lessons learned, and provide a list of future research tasks for the wireless community. Finally, based on the analysis and lessons learned from this survey and tutorial, we present a collection of guidelines for reproducible, comprehensively designed SL models.

Several popular software libraries are used to create and train ML models. Most importantly, these libraries handle back propagation using numerical methods, which otherwise would have to be performed by analytically deriving each gradient using linear algebra. One such library is TensorFlow [19] by Google, which is capable of mobile device deployment as well as browser deployment using JavaScript, provides community support from commercial ML companies, and visualizes easily with TensorBoard. Another library is MXNet [20] by Intel, which supports a wide range of languages and is relatively fast. Keras [21] was developed by a Google employee and is a high level library meant to be used in conjunction with low level abstraction applications. Theano [22] originated from the Université de Montreal and is capable of supporting array processing libraries as well as easy to use with high-level abstraction libraries such as Keras. Facebook developed Pytorch [23] in order to support dynamic graphs as well as both high and low level abstraction, while a DeepMind employee developed Lasagne [24] to be used with Theano. Finally, Caffe [25] was developed by the Berkeley Vision and Learning Center, which builds and trains models with very little code, and provides support for validation and improving models. In our survey of state-of-the-art works, we have found that TensorFlow [19], Keras [21], and Pytorch [23] are the most popular libraries used in open source code. These libraries are easy to learn and powerful due to their large online communities, constant development, and support from both academia and industry.

A. CONTENTS & STRUCTURE

The review and analysis of existing ML models applied to unknown parameter estimation and classification problems provide valuable insights on why different models perform inconsistently for the same data sets, and vice versa. Additionally, a tutorial on ML would be a valuable resource that would enable researchers to reproduce prior works and develop new implementations. Depending on the ML model used, not every section will be of interest to every reader. The paper structure can be categorized by the following interests:

1) SL models and probabilistic concepts are presented as a resource for researchers looking to learn more about SL theory in Section II, which reviews the three paradigms of ML and discusses their appropriate usage. The Neural Network (NN) is derived and defined in Section II-A with an emphasis on Least Squares Regression (LSR), Most Likely Estimation (MLE), and Stochastic Gradient Descent (SGD). Section II-B presents an argument for SL-based unknown parameter estimation and classification, referencing the non-generative universal approximator proof. Finally, a variety of SL algorithms are derived, specifically the Convolutional Neural Network (CNN), with an emphasis on input-equivariance in Section II-C; the Recurrent Neural Network (RNN), with an emphasis on vanishing gradients in Section II-D; and the Support Vector Machine (SVM), with an emphasis on kernels and margins in Section II-E.
2) Section III-A surveys a history of seminal Supervised Learning (SL)-based unknown parameter estimation

and classification papers applied to PHY layer applications. Section III-B surveys the sequential tasks of a receiver's PHY layer in the wireless stack, common processing tasks, and a brief survey of recent works. This section is suggested for all readers, as it establishes the context and motivation that is leveraged throughout the rest of this tutorial.

3) Those interested in learning from recent works are recommended to read Section IV, which reviews a selection of popular works that use a variety of SL models in several PHY layer applications covered in Section III-B. Specifically, Section IV-A reviews Dynamic Spectrum Access (DSA), Section IV-B examines channel corrections such as equalization, Section IV-C explores Automatic Gain Control (AGC), Section IV-D covers Multiple Input Multiple Output (MIMO) coding and antenna control, Section IV-E investigates Analog-to-Digital Conversion (ADC) / Digital-to-Analog Conversion (DAC), and Section IV-F examines Automatic Coding and Modulation (ACM).
4) Researchers keen on designing or altering a model are recommended to read Section V, which serves as a survey and guide to the choices made when changing or designing a new model for research purposes. Included are guides on how to update the weights (Section V-A), initialize the model weights (Section V-B), guide the training process by regularization (Section V-C), decide on the type of data to collect (Section V-D), pre-process data (Section V-E), simulate the additional training data using data augmentation (Section V-F), and validate the hyper-parameter choices (Section V-G). A set of instructions for, and benefits of, ensemble learning are also included (Section V-H).

B. RELATED WORKS

Scientific publications that provide details on the implementation of ML applied to wireless communication systems (Table 1) are relatively difficult to find despite the widespread use of ML. A summary of ML employed in wireless communications is presented in Table 1, which includes the ML models used, the layers of the wireless communications stack affected, and the ML and probabilistic concepts employed. After this review was conducted, it was noticed that while most topics were presented by at least one work, there still remains a need for both a PHY layer survey paper, and for a comprehensive case study relating all key ML and probabilistic concepts using visual analyses and comparisons.

II. SL OVERVIEW

One frequently used ML approach is Supervised Learning (SL), where estimation or classification of test data is performed using a model trained on input data paired with the correct output values. SL algorithms can be used in a wide variety of wireless communications applications, including error control codes, network security, data compression, modulation classification, power management, bit detection, user localization, mobility prediction, and filtration [7]. Popular families of SL algorithms include the CNN [26], SVM [27], KNN, the DT, and RNN [28]. A summary of the models is presented in Table 2.

A. NN OVERVIEW

Intuitive use of SL algorithms requires an understanding of a wide array of statistical topics including regression, concavity, Bayes theorem, MLE, chain rule, and gradient descent.

NN models estimate or classify unknown parameters y by observing training data matrices $x \in \mathbb{R}^{n,p}$ with n samples and some dimensionality p. Each element $x_i^j,\ j = 1, \ldots, p$ of each sample $i = 1, \ldots, n$ is also called a feature. The set of all values data can take $x \in \Omega$ is called the state-space of the data.

Given the continuous nature of many real-life state-spaces, it is impossible to train a model with every value in Ω. When the state-space is discrete and large or continuous, a scientist may choose to parameterize a predictive model, or utilize a set of weights $w \in \mathbb{R}^{n,p}$ and scalar biases b so that a continuous trend-line $\hat{y}$ can be approximated from a finite training data set, used to choose model parameters that optimize an objective function that minimizes trend-line error. Such models allow generalization under the assumption that states observed are descriptive of the underlying data distribution, and allow test data predictions to be looked up quickly $\hat{y} = f(x, w)$.

In reality, few estimation problems can be effectively approached using a linear model. Non-linear, polynomial regression solutions present a possible solution, where an unknown parameter y may be estimated as:

$$\hat{y} = b + w_1^1 x_1^1 + \ldots + w_1^p x_1^p + \ldots + w_n^1 x_n^1 + \ldots + w_n^p x_n^p, \tag{1}$$

however these models assume that the inputs and outputs have a specific polynomial trendline. If the polynomial model is too low of an order, prediction error is high and the model is said to under-fit. Similarly, a polynomial model with too high an order is said to over-fit.

The logistic regression is an alternative algorithm to polynomial regression that has a greater resilience against both over- and under-fitting a model to a set of data [29]. Robustness to over- and under-fitting is achieved by ''boosting'' [29], through which a set of small, serialized, and parallel nonlinear regression models, or neurons (Figure 1), solve a union of small estimation problems to solve a high-dimension estimation problem. Neurons are grouped in sequentially connected sets called layers. A NN is said to be deep when it is constructed of three or more layers [30]–[35], while a NN with a single layer of neurons is defined as being shallow.

In logistic regression, each layer or parallel set of neurons performs two computations, including the linear logit:

$$z = x^T w + b, \tag{2}$$

TABLE 2. A description, summary of the building blocks, and relative advantages and disadvantages of each of the SL models surveyed in Section II.

SL Model	Description	Building Blocks	Advantages	Disadvantages
Logistic Regression	Unknown parameter estimation via error minimization of the union of serial and parallel non-linear regression models	Neurons	Simple to design	Slow to train deep models
Softmax Regression	Unknown parameter classification via error minimization of the union of serial and parallel non-linear regression models	Neurons	Simple to design	Slow to train deep models
CNN	Unknown parameter classification or estimation via logistic/softmax regression with filter banks of constrained weights	Neuron, pooling, zero padding, convolution	Computationally efficient deep models, strong spatial learning	Assumes input equivariance
RNN	Unknown parameter classification or estimation via logistic/softmax regression with feedback and memory states	Hidden state, memory, gates, neuron	Non-fixed input dimensions, strong temporal learning	Slow to train deep models
SVM	Unknown parameter classification via error minimization of support vector pairs constrained by box and linear bounds	Box constraint bounds, linear constraint bounds, sum of forces, kernel, weights, biases	Resistant to training data outliers	Inefficient for large number of classes, poor generalization, slow testing speed
DT	Unknown parameter classification via entropy maximization of the union of serial and parallel decision rules	Nodes, leafs, feature subsets, decision rules, descendants	Fastest training and testing speed	High training variation
KNN	Unknown parameter classification via weighted fingerprinting	Distance, number of neighbors	Lowest assumption on data of any SL model, no training phase	Poor generalization, slow testing speed

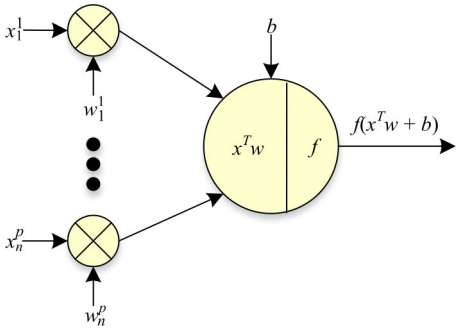


FIGURE 1. A neuron is comprised of a biased vector multiplication of inputs and weights. In this mathematical model inspired by the biological neuron, the neuron is changed from a linear regression model to a non-linear one by the arbitrary activation function f, which in logistic regression is given by the sigmoid function.

and the non-linear activation:

$$a = f(z). \tag{3}$$

The sigmoid activation function, which is defined as:

$$f(z) = \frac{1}{1 + e^{-z}}, \tag{4}$$

is chosen as the activation function to satisfy the non-constructive universal approximation proof [36]. Logistic regression networks are terminated with a linear activation function, defined as $f(z) = z$, instead of the sigmoid activation function, because the sigmoid function is bounded. When classifying a discrete label instead of a continuous one, the softmax [37] activation is implemented instead of the linear activation function. The softmax activation is defined as:

$$f(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \tag{5}$$

which computes the class score of each of the K classes, or the confidence that an input signal belongs to the i^{th} class. The softmax function must be used in place of the linear function for categorical classification, because it transforms model predictions from an ordinal space to a categorical space. The quality of final-layer NN activations is determined via a loss function [30]–[35], or any function that compares NN predictions $\hat{y}$ to true labels y, and returns smaller values when predictions and true labels are similar.

B. WHY USE SL?

There exists numerous scenarios where a wireless transceiver may possess one or more unknown parameters describing the wireless channel or transmitted signal, thus affecting the parameter classification implementation. This makes the problem of optimizing the system parametric, meaning one or more parameters are unknown. This use of the word parametric differs from the data scientist's definitions of parametric and non-parametric, which describe the presence or absence of weights in an estimator model. In the case of a parametric problem, statisticians generally take one of three approaches [38]. The first is to apply the Neyman-Pearson (NP) rule [39] with some significance level α to identify a uniformly most powerful or locally most powerful decision rule. The second is to determine a Bayesian decision rule [40] if priors and cost assignments exist. The third is to use the generalized likelihood ratio test [41], which can be implemented if the MLE for all unknown parameter(s) can be identified. All of these solutions [38] require a Probability Density Function (PDF) describing the likelihood of observations.

ML algorithms that utilize NNs provide powerful solutions when data is available; however their PDF is difficult

to model, as a single layer NN has been proven to be a universal function approximator [36]. This non-constructive proof asserts and proves that any bounded function $f(x)$ can be approximated as $\hat{f}(x)$ within a finite tolerance δ, or $|f(\hat{x}) - f(x)| < \delta$.

C. CNN OVERVIEW

CNNs [30]–[35] have received significant research attention due to their success in high-dimension and large input space applications. While logistic regression models allow for higher order predictions than LSR and lower assumption model designing than polynomial regression, their computational cost does not scale well with a large number of inputs and many layers of neurons. For example, a data set of spectrograms, where each of 100 time-steps is computed as a 512-bin Fast Fourier Transform (FFT) (*i.e.*, $x \in \mathbb{R}^{512,100}$), a single layer softmax regression model with 25 neurons would require 1,280,000 weights and 25 bias terms. The computation of the first layer's linear logit $z = x^T w + b$ would require 2.1×10^{18} computations given $\mathcal{O}(n^3)$ for matrix multiplication and $\mathcal{O}(n)$ for element-wise addition ($n = 1.28 \times 10^6$). The CNN was designed to address this computational load by constraining the softmax regression model to use the same set (filter) of weights for each input feature. Returning to our example, instead of employing 25 neurons, we use a set of 25 filter weights $w \in \mathbb{R}^{f_h, f_w}$ to compute the linear logits, where filter height $f_h = 5$ and width $f_w = 5$ span 5 frequency bins and 5 time steps. The weights are convolved over the input data; the r^{th} row c^{th} column of the linear logit is now computed as:

$$z_{r,c} = \sum_{i=-f_h/w}^{f_h/2} \sum_{j=-f_w/2}^{f_w/2} x_{r+i,c+j} w_{i,j}, \tag{6}$$

and the dimensions of the linear logit $z \in \mathbb{R}^{z_h, z_w}$ are:

$$\begin{aligned} z_h &= (x_h - f_h) + 1, \\ z_w &= (x_w - f_w) + 1. \end{aligned} \tag{7}$$

The resulting number of matrix multiplications (Figure 2) between the filter and input is presently $z_h \times z_w = 48768$, or 7.62×10^8 computations ($n = 1.56 \times 10^4$), which has a reduced number of computations by 10 orders of magnitude.

Convolution is a linear function, so if a CNN is to have the same non-linear capabilities as softmax regression, a similar activation function must be used after computing the linear convolutional logit z. In CNN, Rectified Linear Unit (ReLU) activations [42] can compute each activation $a \in \mathbb{R}^{z_h, z_w}$ from input element x as:

$$a(z(x)) = \max(0, z(x)). \tag{8}$$

Training a CNN to perform classification (Algorithm 1) is done by calculating the weights most likely to predict the correct label in the presence of noise. This is accomplished by minimizing categorical cross entropy loss. SGD, where gradients are calculated via the chain rule (as in logistic/softmax regression), is performed to minimize cross entropy loss. The test-stage protocol for CNNs is described in Algorithm 2.

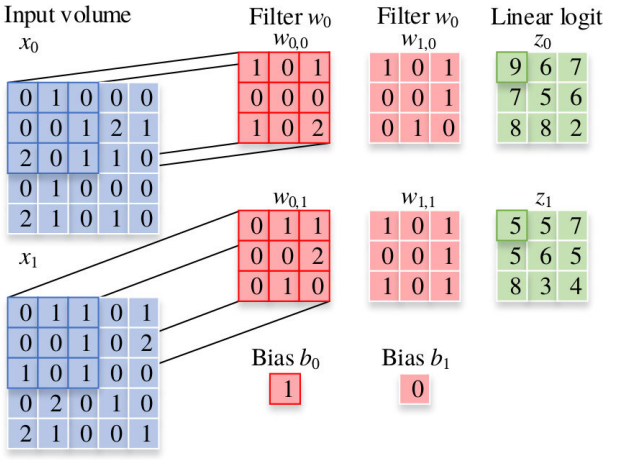


FIGURE 2. A convolutional layer computing outputs $z_{r,c}$ across two input dimensions for some arbitrary data. Representing data with more than one dimension in wireless communications applications is often used to correlate mathematical transforms on data to their labels (*i.e.*, real/imaginary components, absolute value, difference between features).

Algorithm 1 CNN Model Training Protocol [43]

procedure GIVEN TRAINING DATA X, LABELS Y, LEARNING RATE η, TRAINING ITERATIONS n_e
  initialize model parameters w, b
  for n_e **do**
    for each x in X, y in Y **do**
      for layer convolution layer l in L **do**
        for filter w in layer l **do**
          apply zero-padding if used
          compute convolution $z_l(x, w, b)$
          apply ReLU $a = \max(0, z)$
          compute max pooling $P = \text{pool}(a)$
        end for
      end for
      flatten sampled features, $p = \text{flatten}(P)$
      for each dense layer **do**
        compute linear logit, $z(p, w, b)$
        compute activation $a = \max(0, z)$
      end for
      compute softmax $\hat{y} = \arg\max(a(z))$
      compute loss $f_{CE}(y, \hat{y})$
      compute gradients $\frac{\delta}{\delta w} f_{CE}$, $\frac{\delta}{\delta b} f_{CE}$
      update model parameters w, b
    end for
  end for
end procedure

D. RNN OVERVIEW

One fundamental constraint of many SL algorithms is that all inputs have the same number of channels and features, while numerous applications (*i.e.*, time-domain signals) have inputs that vary in their number of features.

Algorithm 2 CNN Classification Prediction Protocol [43]

procedure GIVEN TRAINED MODEL PARAMETERS w, b, TEST DATA X
  for each x in X **do**
    for layer convolution layer l in L **do**
      for filter w in layer l **do**
        apply zero-padding if used
        compute convolution $z_l(x, w, b)$
        apply ReLU $a = \max(0, z)$
        compute max pooling $P = \text{pool}(a)$
      end for
    end for
    flatten down sampled features, $p = \text{flatten}(P)$
    for each dense layer **do**
      compute linear logit, $z(p, w, b)$
      compute activation function $a = \max(0, z)$
    end for
    compute softmax $\hat{y} = \arg\max(a(z))$
  end for
end procedure

Recurrent Neural Networks (RNNs) [30]–[35] (Figure 3) were developed to address this constraint. In a single neuron RNN, one neuron referred to as h (also called a hidden node) scans over a length of inputs and the memory of the hidden state at any time $h_t, t = 1, \ldots, T$ is a function of all previous data $h_{t-1}, \ldots, h_0$.

In a single neuron RNN the linear logit is computed as:

$$z_t = [Uh_{t-1} + Vx_t]. \tag{9}$$

RNNs function on the assumption of sparse data, or that some inputs x_t are not important towards the computation of outputs $\hat{y}$ while others are. Consequently, the activation function used to make the model nonlinear should not add or remove information from data by changing sign, making zero-valued inputs non-zero, or making non-zero inputs zero-valued.

Long sequences ($T \to \infty$) cause the hidden state $h_t = \max(0, z_t)$ to become numerically unstable due to the recurrent computation from equation (9), such that $Uh_{t-1} \to \infty$. Consequently, the tanh activation is used:

$$h_t(z_t) = \frac{e^{2z_t} - 1}{e^{2z_t} + 1}, \tag{10}$$

which is bounded, has the attribute $\tanh(0) = 0$, and does not change sign. The prediction at each step is finally calculated as:

$$\hat{y} = h_t^T w. \tag{11}$$

E. SVM OVERVIEW

While the RNN and CNN families of algorithms leverage the neuron as their core computational unit, the SVM [30]–[35] algorithm does not. Consequently, SVM algorithms have witnessed an increase in popularity during periods of disillusionment about the computational costs of NN algorithms, serving as an alternative approach to SL.

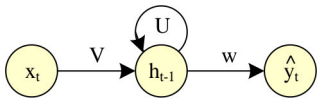


FIGURE 3. The hidden state h_t is computed using each element of the input $x_t, t = 1, \ldots, T$. At each step a prediction $\hat{y}$ is computed, which allows a high level of prediction granularity not seen in other models. This system can either classify each sample or use all the samples to form a final prediction.

SVM algorithms compute decision boundaries given training data, mapping ranges of values in the input space of the algorithm to specific discrete outputs. A decision boundary is a saddle point that separates two exclusive sets of input feature values where the same classification decision is made. Formally, this can defined as $\hat{y} = f(x, w) = C,\ x \in X_C$, where X_C is the set of all input features that result in an SL model classifying the input data x as belonging to class C. In contrast to this, NN-based algorithms are designed to find line fitting or regression solutions. Thus, the focus of SVM algorithms is on using learned weights to separate data by label rather than fitting a trendline to data.

To illustrate, a set of Binary Phase Shift Keying (BPSK) samples can be mapped to binary using the SVM algorithm. The aim is to classify the noisy signal $x = s + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ to hypothesis $s_0 = A\cos(2\pi ft)$ or $s_1 = A\cos(2\pi ft + \pi)$, corresponding to bipolar bits (in-phase amplitude) $y = -1$ and $y = 1$, respectively. A naïve decision rule may be that for mapping $f(x) = x$, values $f(x) > 0$ correspond to outcomes $\hat{y} = 1$, and resulting values $f(x) < 0$ correspond to outcome $\hat{y} = -1$. If some test data were to have their error computed (Figure 4), which error function makes the most sense for this decision rule? If least squared loss, defined by:

$$L_{ls}(f(x_i), y_i) = (y_i - f(x_i))^2, \tag{12}$$

is used to compute the error of predictions on test data, non-zero error will be computed for correct predictions (*i.e.*, for $|x_i| \to \infty, f_{ls} = (1 - \infty)^2 = \infty$). If the goal is to compute error rate, 0-1 loss:

$$L_{0-1}(f(x_i), y_i) = \begin{cases} 1 & f(x_i) \neq y_i \\ 0 & f(x_i) = y_i, \end{cases} \tag{13}$$

would seem like the appropriate error function to use. However, SGD is not possible, as this error function is not differentiable due to its discontinuity and zero-valued gradients.

This issue is resolved by hinge loss, an error function that is both differentiable and computes error rate:

$$L_{hinge}(f(x_i), y_i) = \max(0, 1 - y_i f(x_i)), \tag{14}$$

where error increases for correct classifications as they move towards the decision bound.

In SVM, the sum of the regularization loss and hinge loss are weighted by the hyper-parameter C (Figure 5) to achieve

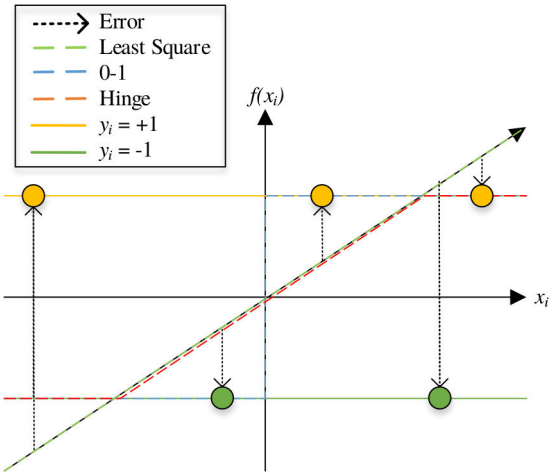


FIGURE 4. A set of noisy signals $f(x_i) = x_i$ are mapped to $\hat{y}_i = 1$ for $f(x_i) > 0$ and $\hat{y}_i = -1$ for $f(x_i) < 0$. The distance from the classifications $y_i = -1, 1$ to the hinge loss, least squared loss, and 0-1 loss is the computed error for that test point using that error function. It is incorrect to use least squared loss to minimize the SVM's error rate (the incorrect classification of BPSK data points $\hat{y}_i = y_i$) because the technique penalizes correct classifications $|x_i| \to \infty$ (*i.e.*, $f_{ls} = (1 - \infty)^2 = \infty$). 0-1 loss would seem ideal; however, it cannot be differentiated due to the discontinuity and flat regions, which is required for training nonlinear models using SGD. Hinge loss $L_{hinge}(f(x_i), y_i) = \max(0, 1 - y_i f(x_i))$ is a suitable error function because it is differentiable (unlike L_{0-1}) and does not penalize large signals like L_{ls}.

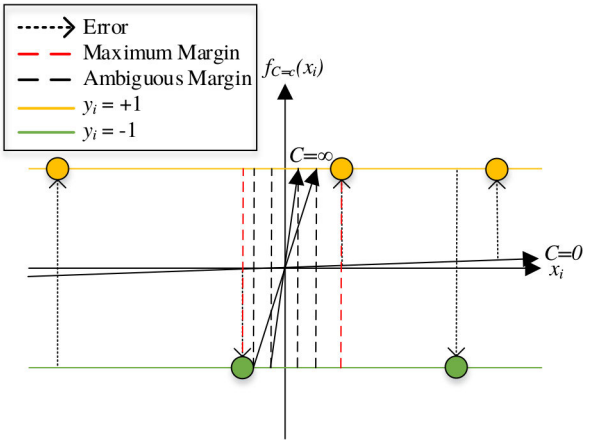


FIGURE 5. If there exists a set of weights $\hat{w}$ that gives minimum hinge loss L_{hinge}, then there exists an infinite range $\lambda\hat{w}$ for $\lambda \geq 1$ of weight sets that give the same hinge loss [43]. The regularization loss $L_{reg} = \frac{1}{2nC}||w||_2^2$ limits the optimization problem to a single weight $\hat{w}$ which produces the minimum multi-objective loss $L = L_{hinge} + L_{reg}$ and has the lowest slope (maximum margin). If C is too small, $L_{reg} \to \infty$ and the weights are forced to be zero-valued $\hat{w} = 0$. Therefore, C must be chosen carefully.

a multi-objective loss function wherein weights are chosen to both minimize hinge loss and make weights as small as possible.

The weighted sum of regularization and hinge loss is defined as:

$$L = \sum_{i=1}^{n} \left(\frac{1}{2nC}||w||_2^2 + \max(0, 1 - y_i f(x_i)\right), \tag{15}$$

where maximum likelihood weights w are computed by SGD.

The dual SVM training procedure is given by Algorithm 3, and the testing procedure by Algorithm 4, where $K(X, X)$ is the kernel definition, which is any similarity function relating two samples, and, uniquely, forward passes are not computed until the test phase due to the linear and box constrained training scheme. Forward passes are computed as:

$$\hat{y}_i = \sum_{k=1}^{n} a_k \circ y_k \circ K_k^T + b. \tag{16}$$

For data where straight or linear decision boundaries exist, linear kernels are used. On the other hand, if the data is expected to have a nonlinear decision boundary, the kernel or similarity function must be changed accordingly. Two popular nonlinear kernels include the polynomial kernel:

$$K(x_i, x_j) = (1 + x_i^T x_j)^d, \tag{17}$$

where d is the order of the polynomial, and the Gaussian or Radial Basis Function (RBF) kernel:

$$K(x_i, x_j) = e^{-||x_i - x_j||^2/2\sigma^2}, \tag{18}$$

where σ^2 is the variance of the modeled Gaussian random variable. Both kernels can be implemented by substituting the kernels into Algorithms (3-4).

III. SURVEY OF SL IN THE PHY LAYER

As discussed in Section I, NNs perform unknown parameter estimation and classification tasks using large amounts of data but few assumptions about the data. In most digital signal processing tasks, the Radio Frequency (RF) channel is treated as a black box due to the quantity and dynamic behavior of stochastic processes present. Due to the universal approximator proof, SL algorithms provide unprecedented flexibility and accuracy in a variety of digital signal processing tasks which must estimate or classify unknown parameters given data that has been altered by a black box set of stochastic processes.

This section provides a brief history of how SL-based parameter estimation and classification algorithms have been applied to the wireless radio system's PHY layer in Section III-A. Next, the wireless receiver's PHY layer will be surveyed in Section III-B, which explores areas of active research that apply SL-based methods and discusses their application-specific advantages and disadvantages.

A. EVOLUTION OF SL APPLIED TO THE WIRELESS RADIO PHY LAYER

Due to training and decision rule concerns raised during the "Artificial Intelligence (AI) winter" [82], ML in wireless communications did not begin to widely appear in published works until the late 1980s. New works at that time were published digital signal processing papers, which often made use of NNs as a non-linear estimator, primarily for equalization [83]. Previously, the Volterra series [84] and nonlinear auto-regressive moving average filters [85] had been used

Algorithm 3 Train Dual SVM [30]

procedure GIVEN TRAINING DATA $X \in \mathbb{R}^{n,p}$, KERNEL $K(X, X)$, TRAINING LABELS Y, MARGIN WEIGHT C, AND TRAINING ITERATIONS n_e
  initialize weights, bias term w, b
  for n_e **do**
    for $i = 1, \ldots, n$ **do**
      for $j = 1, \ldots, n$ **do**
        compute $L(a_i, a_j, C)$
        compute $H(a_i, a_j, C)$
        if $L = H$ **then** pass
        else
          compute $E_i(K(X, X), Y, a, b)$
          compute $E_j(K(X, X), Y, a, b)$
          compute $\eta(K(X, X))$
          compute $a_i^*(L, H, a_i, y_i, E_i, E_j, \eta)$
          compute $a_j^*(y_i, y_j, a_j, a_i^*)$
          update $a_i = a_i^*, a_j = a_j^*$
          compute $b(b_i, b_j, a_i^*, a_j^*)$
        end if
      end for
    end for
  end for
end procedure

Algorithm 4 Dual SVM Predict on Test Data Given Trained Model [30]

procedure GIVEN TEST DATA X, TEST KERNEL $K(X, X)$, TRAINING LABELS Y, TRAINED WEIGHTS a AND BIAS b
  for $i = 1, \ldots, n$ **do**
    Compute $\hat{y}_i$ using equation (16)
    if $\hat{y}_i < 0$ **then**
      $\hat{y}_i = -1$
    else
      $\hat{y}_i = 1$
    end if
  end for
end procedure

for non-linear estimation. This trend continued until the year 2000 when advances in computing, wireless communications, and ML allowed for the application of ML to other layers of the protocol stack besides the PHY layer. A timeline of the history of ML used in the PHY layer is presented in Table 3.

B. SURVEY OF SL APPLIED TO THE WIRELESS RADIO PHY LAYER

A significant number of wireless communications PHY layer blocks can be assisted or replaced by SL-based parameter classification or estimation algorithms. To provide context and a better understanding of the topics covered in Section IV, an illustration of the PHY layer, which highlights the contributions of SL to the various blocks, is shown in Figure 6.

TABLE 3. Timeline of survey papers exploring the use of SL models in PHY layer applications.

Year	Event
1986	← Tank [60] implements an ADC NN during what is called the "AI winter", a time when researchers were disillusioned by gradient descent issues.
1989	← AI winter ends, and many DSP works are published leveraging NNs. Bruck *et al.* [44] use a NN to perform MLE decoding of Hamming and Reed-Muller block codes.
1990	← Chen *et al.* [62] and others make use of NNs as a non-linear estimator for equalization.
1991	← Kunz [72] uses a softmax classifier to decide which channel to transmit on within a cellular network, using interference patterns and channel demand as determinants.
1992	← Aazhang *et al.* [51] use a NN to demodulate CDMA signals, additionally "assisting" training by learning with interference in the samples (on-line training). ← Fang *et al.* [86] use a variational auto-encoder NN to compress data, learning the latent space of data by minimizing reconstruction loss.
1995	← Ibnkahla *et al.* begin publishing works that use a NN to model the inverse non-liner function of traveling wave tubes, wireless channels, and amplifiers. They summarize three years of these works in their 1998 paper [66]. ← Iba [87] and others optimize the power usage of a power distribution system, a technique later used for low power wireless networks such as IoT. Genetic algorithms are used at this time, in favor over initial RNNs (Hopfield NN [88]) due to learning issues present in the latter. ← Southall *et al.* [89] use an RBF NN to perform beam steering towards far-field targets.
1998	← Nandi *et al.* [90] uses a NN to classify the modulation type of observed signals, and another to classify modulation order.
2000	← Hoppensteadt *et al.* [65] train a NN-based Phase-Locked-Loop (PLL) for phase synchronization of multiple signals.
2010	← Daniels *et al.* [91], among the rise of MIMO technologies, publish an ACM work with an SL model that leverages the estimated matrix utilized in MIMO systems.

References for SL-based works implemented by each block are annotated above their corresponding block. For a broader survey of ML applied to all layers of wireless communication systems, the reader is recommended to refer to the works in Table 1.

1) DYNAMIC SPECTRUM ACCESS

Licensed access to the wireless spectrum is auctioned off in blocks of frequency for specific applications such as cellular telephony. Channel access can be optimized by multiplexing several users by time, frequency, code space, physical space, user or service priority, or a combination of these methods, in either the down-link or up-link. It has long been known that there exists a substantial amount of frequency spectrum underutilized by primary users across all wireless networks [92]. Radios equipped with DSA capabilities can potentially reduce the spectrum scarcity by transmitting across licensed spectrum while respecting the rights of the licensed users.

DSA capabilities require computations in every layer of the wireless stack. However, the only task in the PHY layer is spectrum sensing, or detecting/predicting available spectrum from the perspective of the DSA radio, also called a secondary user. Since there are significant penalties for disrupting the activities of primary users, the spectrum sensing models must display a repeatable and high level of applied accuracy in

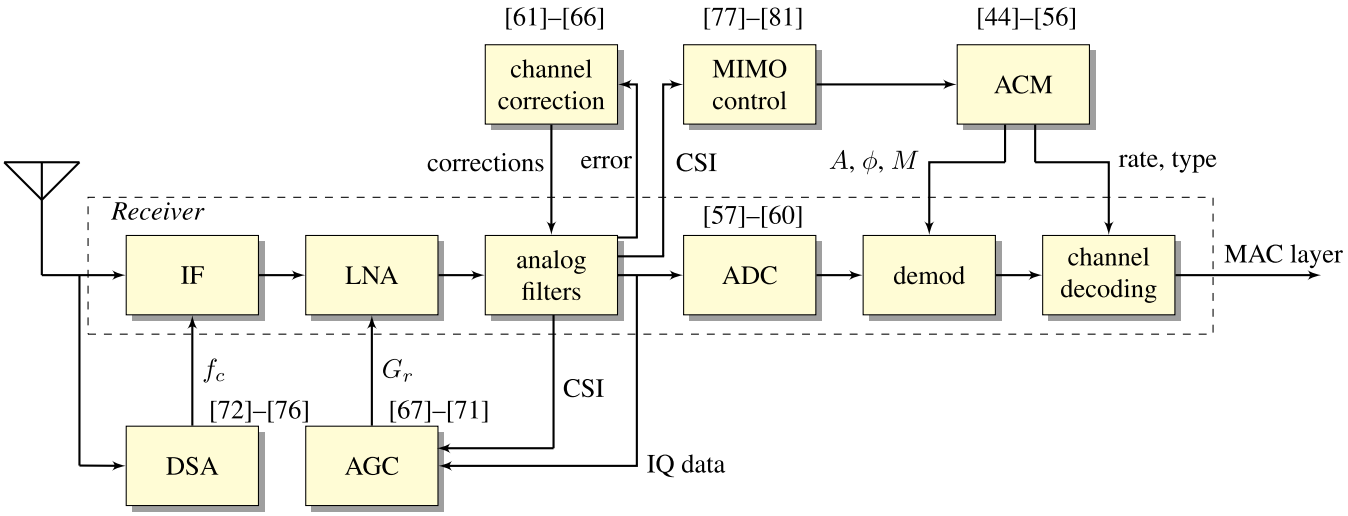


FIGURE 6. A survey of works that apply SL algorithms to various sequential stages of receiving information in the PHY layer. One representative work from each block that is annotated with references is analyzed in Section IV. The terms f_c, G_r, A, ϕ, and M represent chosen carrier frequency, receiver gain, and modulation amplitudes, phases, and order.

order to see commercial use. Spectrum sensing strategies can be categorized as energy detectors, matched filter detectors, and cyclostationary feature detectors. Additionally, spectrum sensing may be conducted via a fusion center utilizing a set of detectors in what is called cooperative spectrum sensing. In what follows, we review work on these topics, which we summarize and compare in Table 4.

Kunz [72] performed an end-to-end spectrum sensing simulation, where a NN was used to decide the frequency assignment of an entire network in both a homogeneous and real-world topography. Depending on the bandwidth and frequency at which the spectrum sensing is being performed, DSA can potentially be prohibitively expensive. Works like Tumuluru *et al.* [73] predicted that spectrum availability would reduce these costs. This prediction was made using simulated data, assuming that Poisson distributed primary user activity and on/off times following geometric distributions in both stationary and time-varying scenarios. Mahajan and Bagai [74] also derived a prediction approach by using a NN to predict how many time slots each spectrum band in a set would be unoccupied for by the primary user. Tang *et al.* [76] performed spectrum sensing with an emphasis on reliably detecting lower power signals, while Thilina *et al.* [75] performed spectrum prediction cooperatively, a process where input features to a variety of SL models are the concatenation of energy levels from the perspective of each radio in the network.

Several random distributions exist in literature (*i.e.*, Erlang [93], geometric [94], Poisson [95]) to describe the time-domain activity of populations of radios for simulation purposes. However, in applied settings, the above SL works demonstrate the ability to form non-linear trend-lines to predict the behavior of individual radios. Additionally, they show that online training allows adaptation in response to minor changes in the Radio Frequency (RF) and physical environment. Finally, SL building-blocks such as the softmax layer provide information about the trained ML model's confidence in decisions made, which allows the use of safety nets to avoid expensive Quality-of-Service (QoS) lapses.

One significant issue these SL-based spectrum sensing works do not address is that non-linear models do not possess bounds for guaranteed performance. Weights learned during SL training can always converge to a local, rather than a global, error minima. This means that a minimum QoS can be difficult to promise to users. Consequently, this issue is a significant disadvantage for commercial applications who might consider using SL-based DSA in an already high-risk service feature.

2) AUTOMATIC GAIN CONTROL

Gain control is an essential process in wireless communications, required for any system to function properly. Receiving radio power amplifiers are controlled through feedback loops to maximize SNR while avoiding clipping and non-linearities. Their counterparts, transmitting radio amplifiers, control gain to conserve power and mitigate interference to other radios.

Gain control approaches that maximize energy efficiency, spectral efficiency, and/or Weighted Sum Rate (WSR) across a network of radios are also called resource management approaches. The most common algorithm used for this, and as a result is most commonly benchmarked against SL-based approaches, is the iterative Weighted Minimum Mean Square Error (WMMSE) algorithm [96]. A frequently asked research question is whether the resource management optimization algorithm used arrives at a Nash equilibrium [97], or a steady state where no radio changes their gain value. In what follows, we review work on these topics, which we summarize and compare in Table 5.

Lee *et al.* [67] used channel information to train a NN in order to maximize the WSR of a network of device-to-device

TABLE 4. A summary of SL-based unknown parameter estimation or classification works on DSA.

Reference	Application	Model(s)	Inputs	Outputs	Key Contribution
[72]	Frequency assignment	NN	Traffic and interference	Channel to transmit on for each transmitter	Keystone paper
[73]	Spectrum prediction	NN	Channel activity history	Channel on/off at next time step	Energy efficiency
[74]	Spectrum prediction	NN	Channel activity history	Channel vacancy duration	Long-term forecasting
[75]	Spectrum sensing	SVM, KNN	Secondary user energy levels	Channel availability	Cooperative spectrum sensing
[76]	Spectrum prediction	NN	Energy and cyclic spectrum values	Presence of primary user	Low power signal detection

user equipment by adjusting transmit gain. Additionally, Lee *et al.* [68] used channel information to train a CNN to maximize the energy and spectral efficiency of a network by adjusting transmit gain. Zappone *et al.* [69] used a NN, trained by data containing User Equipment (UE) location and a path loss model, to maximize global energy efficiency over a radio network by adjusting transmit gain. Sun *et al.* [71] presented a NN trained to maximize the WSR by adjusting transmit gain with an emphasis on online training and scalability for larger networks. Liang *et al.* [70] trained a NN in two stages with channel information mapped to transmit gain, first with WMMSE generated targets, then with Unsupervised Learning (UL) generated targets, to maximize the WSR of a network of radios.

These works show that SL models are ideal for the multi-player, multi-objective optimization tasks seen in resource management works, especially in the presence of unknown physical and RF environments. They demonstrate that, once trained, SL models can compute forward passes relatively faster than their iterative counter-parts, making real-time optimization possible.

However, this computational gain makes the assumption that data and target distributions do not drift during the transition from the training stage to the testing stage. Another issue is that in resource management problems, optimal gain control is unknown, such that data targets must be generated using traditional models such as WMMSE. The above works show that SL weights trained this way cannot achieve resource management decisions more efficiently than the WMMSE algorithm that generated trained data, making the use of SL redundant unless computational gains are achieved.

3) CHANNEL CORRECTIONS

Channel corrections refer to any filtering or feedback controller that mitigates noise embedded in received signals by the wireless channel or radio equipment. These corrections include channel equalization, frequency correction, phase correction and resolving phase ambiguity, and distortion inversion.

Frequency correction of received signals is often applied in sequential coarse and fine stages [98], requiring the computation of a FFT. Phase corrections are computed and applied by the Phase-Locked-Loop (PLL) [99]. Equalization [100] is an inversion technique, where the inverse of a non-flat or multipath wireless channel is estimated and multiplied with a received signal in the frequency domain. Finally, techniques such as digital pre-distortion [101] invert the noise introduced by equipment such as power amplifiers, in a similar way to equalization. In what follows, we review work on these topics, which we summarize and compare in Table 6.

Chen *et al.* [62] performed adaptive, non-linear equalization of noisy binary observations with inter-symbol-interference over tap-based channels by using a NN classifier. Sebald and Bucklew [63] performed the same experiment using SVMs. Kechriotis *et al.* [64] equalized a variety of modulated signals using a RNN, where the input data originated from either linear, partial response, or non-linear wireless channel models. Hoppensteadt and Izhikevich [65] utilized a NN-based phased-locked-loop to synchronize the phase of a set of cyclostationary 6×10 pixel greyscale images. Ibnkahla *et al.* [66] used NNs to model the inverse transfer function of a traveling wave guide and solid state power amplifiers in order to remove their non-linearities. Indriyatmoko *et al.* [61] corrected the phase of differential GPS systems using a NN.

Much like in the case of resource management, these works show that channel corrections are computed at a lower cost when iterative and expensive feedback controllers are replaced with SL models. Trained models compute forward passes significantly faster than their traditional counterparts can form estimates through feedback loops and repeated domain transforms. Additionally, SL models presented in these works outperform their traditional counterparts, as the training targets are not generated by their traditional counterparts, but rather by using sequences of symbols or bits known *a priori* by all radios in the network.

However, higher performing SL-based channel correction algorithms come with new challenges. Since no information is contained in *a priori* symbols or bits, data throughput is reduced during the SL training cycles. Consequently, the models must be trained offline to generalize to all test scenarios, or be trained in cycles of minimum duration and frequency, such that a certain QoS is maintained.

4) MIMO CONTROL

Radio transceivers that perform up-link and down-link tasks with multiple antennas in order to increase data throughput are known as MIMO systems. Recently, massive MIMO systems with spatial multiplexing have gained significant attention [102] due to their potential for higher throughput,

TABLE 5. A summary of SL-based unknown parameter estimation or classification works on AGC.

Reference	Application	Model(s)	Inputs	Outputs	Key Contribution
[67]	WSR optimization	NN	CSI	Transmit gain	Low computational cost
[68]	Energy and spectral efficiency optimization	CNN	CSI	Transmit gain	Higher or similar efficiency as WMMSE
[69]	Energy efficiency optimization	NN	UE locations	Transmit gain	end-to-end parameter estimation
[71]	WSR optimization	NN	CSI	Transmit gain	Online training, scalability
[70]	WSR optimization	NN	CSI	Transmit gain	Training without WMMSE-generated targets

TABLE 6. A summary of SL-based unknown parameter estimation or classification works on channel corrections.

Reference	Application	Model(s)	Inputs	Outputs	Key Contribution
[62]	Equalization	NN	Received samples	Transmitted symbol	Keystone paper
[63]	Equalization	SVM	Received samples	Transmitted symbol	Model analysis, mitigating multicollinearity
[64]	Equalization	RNN	Received samples	Transmitted symbol	First use of RNN for equalization
[65]	Phase correction	NN	Unaligned cyclostationary signals	Phase corrections	NN-based PLL
[66]	Nonlinear channel with memory estimation	NN	Transmitted signal	Received signal	Amplifier and traveling wave tube distortion and non-linearity correction
[61]	Phase correction prediction	NN	History of phase corrections	Phase error at next time step	Continued DGPS operation when correction signals are lost

lower latency, lower power, and lower costs than traditional MIMO wireless systems.

A MIMO transmit-receive pair with n_r receiving antennas and n_t transmitting antennas has an upper bound on data rate (bps/Hz) given by [103]:

$$C = E\left[\log_2\left(\mathrm{I}_{n_r} + \frac{\mathrm{SNR}}{n_t} HH^T\right)\right], \tag{19}$$

where H can be modeled as a deterministic or random channel matrix, I_x represents a square identity matrix with x columns and rows, and Signal-to-Noise Ratio (SNR) is the ratio of received power of the transmitted signal and the noise power of the wireless channel. In what follows, we review work on these topics, which we summarize and compare in Table 7.

MIMO radio systems frequently use SL to estimate and classify a range of unknown parameters using estimated CSI and SNR. For example, Klautau *et al.* [77] trained several SL models to select the highest data throughput antenna pairs. These models were trained with two-dimensional maps of vehicle positions and sizes that were translated to either the optimal beam pair (determined via ray tracing) or the optimal azimuth and elevation angles. Gao *et al.* [78] leveraged SL as an optimizer to choose a digital pre-coding matrix and analog beam-forming matrix to maximize data rate and minimize power consumption. Alkhateeb [79] presented a dataset framework parameterized by $\mathcal{S}$ and channel $\mathcal{R}$ for performing beam selection similar to that performed by Klautau *et al.* [77]. He *et al.* [80] developed an approach that chose which antenna to transmit to, assuming H is known at both the transmitter and eavesdropper, to maximize secrecy probability. Samuel *et al.* [81] compared a deep detector to traditional detectors in both fixed and varying channels, while Lee *et al.* [104] computed antenna transmission and reflection properties using a NN based on a three-dimensional particle simulator assuming point transmitters and sphere receivers.

These references highlight how MIMO implementations can benefit from SL models in multi-objective optimization problems with unknown parameters that cannot be related analytically to noisy observations. Additionally, they show that SL models can classify or estimate antenna behavior better than traditional methods by learning the RF environment through online training. Finally, they show that trained SL models achieve lower computational costs than traditional alternatives.

Although SL has the potential to solve challenging problems, it does possess several implementation issues as well. For instance, training models offline that generalize well to online data is difficult. Alternatively, it is more expensive to perform online training of SL models relative to traditional unknown parameter estimation and classification techniques. Finally, real-world labeled data can be scarce in expensive, rare, or protected systems such as 5G massive MIMO base stations.

5) ANALOG TO DIGITAL CONVERSION

DACs and ADCs are fundamental for all digital radio operations, and significantly influence PHY layer data representation. ADC is performed by mapping each sample of a discretized analog signal to a unique bit sequence; this process is referred to as quantization. The mapping process from continuous to discrete amplitudes also introduces some distortion called quantization error, which is directly related to the resolution of the quantization process. Increasing the bandwidth and amplitude dynamics of a modern radio system imposes greater constraints and requirements on the ADC architecture; as a result it needs to be designed with flexibility and adaptability in the presence of manufacturing variation, timing errors, jitters, and parasitic capacitance [105].

TABLE 7. A summary of SL-based unknown parameter estimation or classification works on MIMO control.

Reference	Application	Model(s)	Inputs	Outputs	Key Contribution
[77]	Antenna selection or aiming	SVM, DT, NN	2D maps of vehicle size and location	Optimal beam pair or azimuth and elevation angle	Ray tracing assisted training
[78]	Power efficiency	Adaptive cross entropy NN	CSI, beam candidates	Pre-coding and beam-forming matrix	Low cost MIMO switch and inverter
[79]	Beam selection	NN	Uplink pilots	Beamforming vector for each base station	Beam selection data set generation framework
[80]	Secrecy probability optimization	SVM	CSI	Optimal transmit antenna	Low feedback overhead
[81]	MIMO detection	NN, LSTM	CSI	Presence of signal	Survey and comparison of methods
[104]	S-parameter antenna estimation	NN	Antenna size and spacing	Channel gain matrix	Molecular function approximation

The development of high speed processors has also affected the need for higher resolution and higher sampling rates to meet the increasing demands of wireless systems. In what follows, we review work on these topics, which we summarize and compare in Table 8.

Tank and Hopfield [60] minimized a custom loss function by training a set of conductance weights between amplifiers in a 4-neuron ''Hopfield'' ADC using analog signals. This work was expanded on in many ways [58], including through the addition of Complementary Metal-Oxide-Semiconductor (CMOS) memristors. Vidyasagar [59] attempted to regularize the Hopfield ADC to encourage convergence to a global loss minima, thus avoiding hysteresis. Danial *et al.* [57] implemented a memristor DAC with a focus on online training loops.

Research on SL-based ADCs has been incremental and largely centered around Hopfield's ADC design [60]. The sampling frequency f_s must be chosen carefully for these designs to avoid aliasing, minimize voltage fluctuations, minimize costs, and ensure the rate is below the maximum sampling rate allowed by the memristor design. The SL algorithm cyclically reads and writes, where the final results is sampled at the end of the reading period T_r. The writing cycle mitigates transient effects and latches using a negative edge trigger over a period T_w, upon which time the feedback circuit is activated, executing the learning algorithm, and computing the error between the observed output and the desired output. The sampling frequency is defined as:

$$f_s = \frac{1}{T_r + T_w} > 2f_m, \tag{20}$$

where f_m is the maximum frequency component of the signal being converted. Typically a LPF or anti-aliasing filter is used before the ADC to filter out frequencies above f_m, which can result in the loss of information if f_s is not properly chosen. To keep costs low, it is typically chosen to assign $f_s = 2f_m$. However, oversampling with a higher rate and then digitally filtering the signal to limit the signal band width can make it easier to realize anti-aliasing filters, improve bit depth, and reduce noise. Voltage division performed during Hopfield ADC writing cycles creates non-uniformity in the voltage cycles proportional to f_s, significantly reducing the learning rate of the converter, and increasing the error between the actual and desired output. A capacitor may be added to mitigate this non-uniformity. The capacitor's value is bounded by the sampling frequency of the converter as:

$$\frac{1}{f_s} \geq \frac{C_{\text{shock,max}}}{K(V_{\text{DD}} - 2V_T)}, \tag{21}$$

where V_{DD} is the power supply, K is the transistor conduction strength constant, and V_T is the trained synaptic voltage weight. Additionally, converters can only support a finite range of f_s, limited by:

$$f_{\max} = \frac{1}{2\pi R_{\text{OFF}} C_{\text{mem}}} \sqrt{\left(\frac{R_{\text{OFF}}}{R_{\text{ON}} \cdot 2^{N+1}}\right)^2 - 1}, \tag{22}$$

where R_{OFF} and R_{ON} are the memristor conductance when off or on, and C_{mem} is the capacitance of the memristor.

This collection of works show that the small architecture of NN-based ADCs allow for real-time training to satisfy these needs. Fault tolerant SL models additionally calibrate to modeled noise, where traditional ADCs attempt to quantize noise-free signals. End-to-end NN-based ADCs do not make use of power amplifiers, increasing energy efficiency with simple digital circuits.

Similarly to channel correction applications, the live training of SL models requires the transmission of *a priori* targets, thus reducing data throughput. These works show a trend of SL-based ADCs that do not classify bits directly from analog signals, but rather train optimal resistance values within the analog circuits. These hybrid analog/NN-based ADCs require circuit-specific learning techniques, such that they do not generalize well across different hardware.

6) AUTOMATIC CODING AND MODULATION

Jointly adjusting the modulation order and type alongside the error control code rate and type in order to maximize data throughput is essential to meet modern QoS demands [106]. Spectrum enforcement, the generation of coverage maps, and wireless localization require modern radios to be capable of learning the modulation and coding behavior of unknown signals from the ground up. This has led to many novel signal classification works, which we discuss in what follows.

Error control codes [107] are binary mappings that add redundancy to messages in order to make it easier to reconstruct those messages in the presence of noise. A control code

TABLE 8. A summary of SL-based unknown parameter estimation or classification works on ADC.

Reference	Application	Model(s)	Inputs	Outputs	Key Contribution
[60]	ADC	Hopfield ADC	Analog samples	Bits	Keystone paper
[58]	ADC	Hopfield ADC	Analog samples	Bits	Programmable memristors
[59]	ADC	Hopfield ADC	Analog samples	Bits	Hysteresis avoidance through regularization
[57]	ADC	Hopfield DAC	Bits	Analog samples	Inverse design

rate refers to the ratio of information bits to redundant bits in a control code type. Error control code types are categorized as either block or convolutional. Recent improvements to control codes include turbo codes [108] and Low-Density Parity-Check (LDPC) codes [109]. Other error reduction techniques such as interleaving [110] reduce errors without adding redundant information by shifting bits around in time to avoid bursty noise.

The modulation and demodulation of information is performed to map bits and symbols. The order of a modulation scheme M refers to how many unique symbols the mapping can take, related to the length of the binary codewords b by the equation $M = 2^b$. The modulation type is defined by how symbols are different from one another, be it by shifting phase, frequency, amplitude, or a combination of phase and amplitude. All modulation types are carefully designed to maximize the space between symbols while minimizing symbol amplitude, balancing error reduction with power consumption. In what follows, we review work on these topics, which we summarize and compare in Table 9.

Bruck and Blaum [44] trained small NNs to decode Hamming and Reed-Muller block codes, while Orturo *et al.* [45] used a NN to decode Bose-Chaudhuri-Hocquenghem (BCH) (7,4) block codes. Wang and Wicker [47] used a NN to perform Viterbi convolutional decoding. Aazhang *et al.* [51] assisted a matched filter with a NN in a Code Division Multiple Access (CDMA) detector to detect and demodulate signals, aided by online training. Kechriotis and Manolakos [52], [53] performed a similar experiment with a Hopfield NN, which is an early version of the RNN architecture. West and O'Shea [49], [50] used a LSTM RNN as well as a CNN to perform modulation classification on noisy IQ data given eight modulation types and orders ranging from $M = 4$ to $M = 64$. Rajendran *et al.* [56] performed similar research, and also employed the averaged magnitude FFT of noisy signals to train a model to classify one of the six possible wireless standards that have been observed. Park *et al.* [55] used an SVM to perform modulation classification on the continuous wavelet transform of noisy signals mapped to four different modulation types and orders ranging from $M = 2$ tot $M = 16$.

These works demonstrate that trained SL models are computationally cheap compared to many advanced coding techniques such as looping turbo codes. Additionally, the results from each of these publications demonstrate how control codes can be calibrated to account for specific noise behaviors, in a similar way to ADC applications. Finally, large libraries of signals can be used to train deep models to identify a variety of aspects of unknown signals, whereas traditional detectors are designed with specific signal standards or structures in mind.

As with several PHY layer applications discussed so far, the use of SL models in control codes require the transmission of *a priori* bits, which reduces data throughput during training. A real-world issue overlooked by many of these simulation-based works is that online training requires the transmission of weight updates or error values across the wireless channel, which results in noisy learning and loads the network layer with an additional task to coordinate between devices. There are also issues with signal classification, such as the inability to generate live training data from uncooperative transmitters. Training these models offline to be generalized without knowing the channel effects and transmission behaviors that will be experienced at the testing phase is often not feasible.

IV. ANALYSIS OF PHY LAYER SL WORKS

In this section, we analyze a subset of works surveyed in Section III-B by performing an in-depth examination of several publications applying SL models to each domain of the PHY layer. The intent of this section is to provide insight for other researchers on how to apply SL to their PHY layer research activities. The following analysis is organized by PHY layer application topic, with each section describing a chosen publication, along with its goal, solution, results, lessons learned, and directions for future work in the broader topic.

A. DYNAMIC SPECTRUM ACCESS

The DSA research presented by Thilina *et al.* [75] involved training several SL models, whose weights were stored in a central organizing radio that could both receive primary user activity information from all listening secondary users, and transmit commands to each secondary user on whether or not it was safe to transmit without disrupting the primary users. In this way, the SL model was used to perform cooperative spectrum sensing across the network of listening secondary users (Figure 7). The authors simulated a single channel network of primary users whose transmissions were used to compute a time-series of non-central chi-squared energy estimates at each secondary user. The authors' aim was to train a variety of SL algorithms to perform a binary classification at each time step to determine channel availability, given two secondary users for visualization purposes. Classification was performed using SVM models with both polynomial and linear kernels, as well as the K-Nearest Neighbor (KNN) [111] algorithm with both city block and Euclidean distances. Classification accuracy is significantly dependent on SNR, but the Euclidean KNN had the highest detection

TABLE 9. A summary of SL-based unknown parameter estimation or classification works on ACM.

Reference	Application	Model(s)	Inputs	Outputs	Key Contribution
[44]	Error control decoding	NN	Bits	Bits	Hamming and Reed-Muller block codes
[45]	Error control decoding	NN	Bits	Bits	BCH block codes
[47]	Error control decoding	NN	Bits	Bits	Viterbi convolutional codes
[51]	CDMA demodulation	NN	Baseband symbols	Bits	Online training
[52], [53]	CDMA demodulation	Hopfield NN	Baseband symbols	Bits	Low computational cost
[49], [50]	Modulation classification	LSTM, CNN	Baseband IQ symbols	Modulation class	Signal classification
[55]	Modulation classification	SVM	Continuous wavelet transform of baseband symbols	Modulation class	Input feature design
[56]	Modulation classification	RNN, LSTM, CNN	Baseband IQ symbols	Modulation class	Comparison of models

probability, also called precision or Positive Predictive Value (PPV):

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{23}$$

at low SNR values, as did the linear kernel SVM at medium SNR values. Additionally, all classifiers achieved a 100% detection probability with a high enough SNR. Many SL research efforts provide deeper classification performance detail through the use of confusion matrices [112], whose rows and columns describe the number of occurrences of predicted and true classes. In particular, the diagonal values of confusion matrices highlight the byclass PPV of the tested data set.

By training several SL models to perform cooperative spectrum sensing, this work highlighted several lessons regarding trade-offs with respect to choosing the appropriate algorithm for the problem.

For primary user transmit power values over 100 mW, the linear kernel SVM model had a higher probability of detection than any other detection model used by the central radio of the secondary users. The observed power from a single primary user was log-normal distributed from the perspective of a secondary user, and chi-squared distributed for a set of primary users. Consequently, similar to threshold-based NP detectors [113], the decision bounds could be separated by a scalar threshold for one secondary user, a linear boundary for two secondary users, and a sum of linear boundaries for three or more secondary users. The combination of a sum of linear bounds was also linear, such that the polynomial kernels used in this experiment over-fitted to the training data and did a less successful job of classifying the presence of the primary user.

The SVM linear kernel model also outperformed its benchmark model, the Fisher linear discriminant analysis classifier [114]. In a similar way to LSR and other non-SL-based linear models, the Fisher linear discriminant classifier assumes data homoscedasticity, or that the covariance of the data does not vary by sample or by class. This assumption does not hold, since the primary user active state power measurements were log-normal distributed for a single primary user and chi-squared for multiple primary users. However, the primary user inactive power levels describe the power distribution of the noise inherent in the channel, modeled by a squared Gaussian distribution. Since the data from each class were generated from different distributions, their covariance matrices cannot be equal and the data is heteroscedastic. The SVM model does not assume homoscedasticity.

For primary user transmit power values under 100 mW, the weighted-vote KNN model with an unreported K value had a $\sim$ 33% higher probability of detection by the fusion center of secondary users than all other models. Typically K is chosen by guess-and-check validation searches (see Section V-G). The authors cited this success as a consequence of the KNN leveraging information local to each test sample, which suggests that a smaller K value was chosen, and that the linear kernel SVM boundary is slightly biased in low transmit power scenarios. Furthermore, the KNN performing classifications using Euclidean distance outperformed the lower order city block distance model due to the ''curse of dimensionality'' [115], *i.e.*, a rule of thumb that states higher order data is classified more accurately using lower order distance metrics, and vice versa.

The reported model training duration and classification delays highlight the non-parametric nature of the KNN models, which do not train a set of weights, but instead compare each training-testing data pair, scaling poorly with training data size. The higher training time for the polynomial kernel (equation 17) SVM than its linear counterpart suggests either that the polynomial kernel trains on a larger number of support vectors, or that the weight updates take longer to converge due to a flatter loss gradient caused by many decision boundaries being equally accurate.

Thilina *et al.* [75] shows that spectrum sensing using SL computes decision boundaries that correctly fit the underlying linear distribution of primary user signal detections. Spectrum sensing and signal detection will always be an area of open research, requiring higher detection rates using less data at lower SNR values. This is especially important when detecting primary users in real-time within short time windows, so that spectral coexistence can be realized in frequency white spaces or even unlicensed frequencies.

Primary user receivers may also be passive devices, and thus their locations, frequency bands of operation, and receiving power levels can be potentially difficult or even impossible to estimate. Detecting or inferring any information about

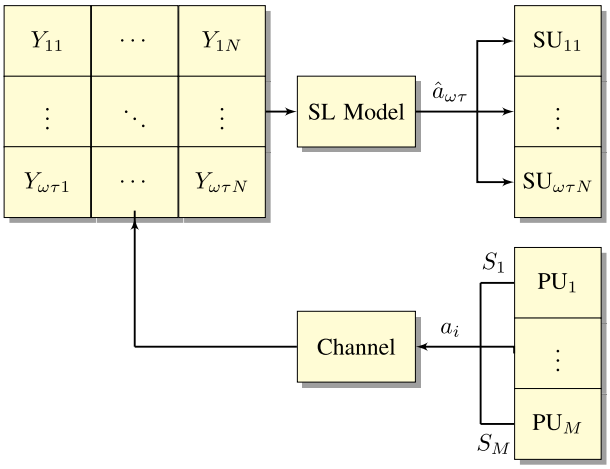


FIGURE 7. SL-based collaborative spectrum sensing was performed by Thilina *et al.* [75] via classifying channel availability a_i at time $i = 1, \ldots, \omega\tau$ and communicating that availability to the $n = 1, \ldots, N$ secondary users. All SL models used by the authors performed classification by observing time slices of the estimated total energy received from all M primary users on the channel at time i by the n^{th} secondary user.

primary user receivers to determine the availability of a channel is an area of open research that may benefit from SL-based signal detectors, localization systems, and eavesdroppers.

Another area of open research is the classification of primary users from secondary users from other cognitive networks. Since the penalty of interfering with the activity of secondary users from other networks is significantly less than that of interfering with a licensed primary user, it is valuable to know what spectrum is occupied by what type of user.

Finally, SL-based spectrum sensing has a few additional areas of open research, including the enforcement of a minimum guaranteed QoS for primary users. Relatively few spectrum sensing works establish protocols for monitoring SL classification accuracy and implementing safety measures when performance is negatively impacted. Another SL specific issue is training generalized models despite time-varying spectrum and PHY distributions. Most SL-based spectrum sensing papers implement a form of energy detection on stationary primary and secondary users, whose classification performance would decrease significantly in a mobile network.

B. CHANNEL CORRECTIONS

SL-based equalization has been extensively studied by the research community with numerous implementations having been proposed. For example, Sebald and Bucklew [63] trained a SVM model to jointly flatten and demodulate signals transmitted over a non-flat frequency channel (see Algorithm 3). The authors simulated signals that are altered by a non-linear channel with Inter-Symbol Interference (ISI), modeled by a Finite Impulse Response (FIR) filter and injected with AWGN. The SVM classifier was trained to classify delayed bipolar bits by observing a number of transmitted bits greater than the sum of the equalizer delay and number of ISI samples. The authors found that certain sequences of bits were indistinguishable, such that model predictions of the previous sample were needed to learn effective decision boundaries. Even after this, some bit sequences had equal constellation points requiring the implementation of a bank of SVMs that were equal in number to the amount of constellation points, which were trained and tested on a subset of data. This final step was found to properly create a data space with unique constellation points, which allowed there to be an optimal decision boundary.

The work of Sebald and Bucklew [63] provides an excellent lesson in combating multicollinearity by adding information to model inputs. The authors took two thoughtful steps, (Figure 8) first to ensure that input-output pairs were unique, then to decouple input features. The issue of overlapping constellation points extends to many other applications, such as ACM. There are also more hands-off approaches to mitigating multicollinearity, such as Principal Component Analysis (PCA) [116], which removes input features that are highly correlated.

A visualization of decision boundary behavior in regions with no data is also provided by Sebald and Bucklew [63]. In particular, ''ghost'' decision regions result from underfitting, where decision regions wrap around to the other side of a finite data space despite no training data being present there. Areas with no training data can also have winding, high variance curves that seem to be fitting to invisible data. Finally, ''spoons'', or decision regions enclosed by another decision region, can be encountered, where there exists a trade off between all of these behaviors and computational costs, achieved by decreasing C while increasing the number of support vectors.

SL-based equalization is a frequently explored application of SL used in the PHY layer of wireless systems. The success of the analyzed work regarding non-linear equalization models and those similar to it, as well as the continued success of linear equalization coefficient estimators [100], has resulted in SL-based equalization research reaching relative maturity. Phase and frequency-corrective algorithms have similarly reached a level of maturity, with many SL-based channel correction works over the past 20 years building upon the state-of-the-art. These works aim to leverage recent advances in feedback-based ML models [117] to take advantage of the temporal nature of data dispersed by non-linear channels. Other works introduce methods for computationally-efficient online training, of models with zero offline training rather than the more common approach of training offline and tuning weights with small sets of online data [118].

C. AUTOMATIC GAIN CONTROL

Lee *et al.* [67] trained an AGC framework by using a logistic regression model, simulated at a Base Station (BS), to maximize the WSR of Device-to-Device User Equipment (DUE) capable of sharing resources with Cellular User Equipment (CUE). This process is referred to as underlaid communications. The model inputs were simulated assuming a set of

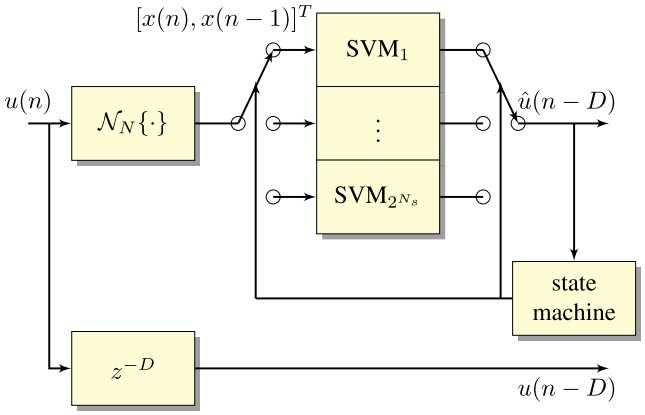


FIGURE 8. The SL-based equalizers designed by Sebald *et al.* [63] observed channel outputs $\boldsymbol{X}(n) = [x(n), \cdots, x(n-M+1)]^T$ from the non-linear channel with length N ISI to classify the delayed channel input $\hat{u}(n-D)$. Multicollinearity was reduced by concatenating two observed channel output arrays, and by implementing a feedback controller on a bank of 2^{N_S} trained SVM models, where $N_s = N + M - D - 2$.

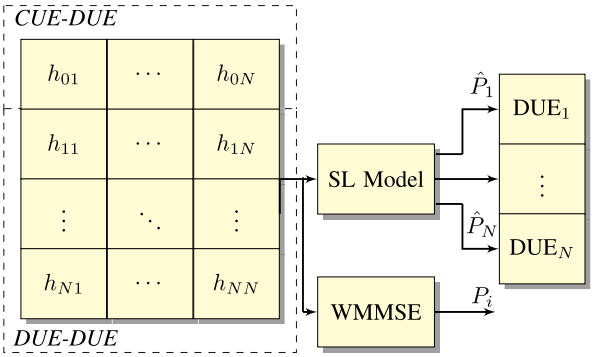


FIGURE 9. SL-based AGC optimization in underlaid communications as designed by Lee *et al.* [67] was performed by estimating each DUE's transmit gain as a multi-parameter estimation problem. The trained model made these regression estimates by observing the estimated interference that would be caused between the target CUE and all DUE, as well each DUE pair. The underlying information that was learned by the model was the location of each device, the path loss model of the channel, and the transmit power of the CUE.

N randomly placed DUE radios that were interfering with a single CUE attempting to connect to the BS. An $N \times N$ channel information matrix was computed assuming a path loss model with multi-fading, and that matrix was concatenated via early data integration with a length N CUE information array (Figure 9). Data targets were generated by minimizing the WSR using the WMMSE algorithm. The authors found that when trained with 50, 000 samples, their model's training WSR matched the WMMSE, and their model's testing WSR fell behind by about 1 Mbits/sec. However, after a 50 minute training period, their model predicted transmit gains in about 0.4 ms compared to the iterative WMMSE's 4.5 ms computation time.

While the order of magnitude improvement to the computation speed of test data is impressive, it is worth noting that the WMMSE elapsed time of 4.5 ms would be suitable for all but the most spatially and temporally dynamic scenarios. If the DUE channel matrix and CUE channel array were to be changed more frequently than once every 4.5 ms, WMMSE would not be fast enough, which would result in in the shallow NN becoming a viable alternative.

Since the deep and shallow NNs have near equal training and testing set WSR gaps, over-training caused by model depth would not be likely. Additionally, the gaps shrink considerably when the number of samples N_S is increased. On the other hand, over-training is likely when training on an insufficient number of samples. The authors' even data split was insufficient. The training and testing WSR gaps could likely be reduced by using a 80:20 or 70:30 split without needing to generate additional samples.

Regularization, especially in the form of creating filter banks of weights or switching to a CNN model, would also mitigate over-training issues. This would guide the model to relate elements of the gain matrix to reduce over-fitting, particularly by using vertical filters that relate CUE-DUE to DUE-CUE values. Since the simulated channel matrices were functions only of the radial distance between DUEs and CUEs, the assumption of input equivariance would readily hold without the need for substantial pooling or the addition of redundant filters. An example of data that would challenge CNN input equivariance would be a 3D city model, where path loss is a function of location.

This analyzed work and those surveyed in Section III-B that focus on SL-based AGC show that model accuracy is limited by the ability to generate learning targets. The transmit gain chosen by each radio is treated as an unknown parameter to be estimated by observing noisy samples of the channel information, such that SL algorithms must be taught by another estimator. Consequently, SL models can only provide computational gains to AGC for the purpose of improving spectral or energy efficiency. Although in many wireless applications the online training of SL models yields a flexibility gain, SL-based AGC models only provide computational gains when trained offline. Consequently, WMMSE estimators actually provide more flexibility than SL models in this application.

The broader field of SL-based AGC possesses several open challenges for the research community. For instance, low power, long-life WSNs can potentially experience frequent and long network disruptions due to radio interference, destruction of equipment, or loss of power. This requires energy efficiency optimization processes using limited information. While model outputs must be generated by an estimator and taught to a SL model, early integration of input data other than the channel state matrices may give SL models a redundancy that WMMSE estimators do not possess, as WMMSE estimators operate solely on CSI. Finally, there are few works in open literature that control the gain of receivers with SL models, which must be balanced to maximize SNR and simultaneously minimize clipping and amplifier non-linearities. Existing receiver AGCs control the gain to achieve a user-defined target output power, but shallow SL models may compute these values more quickly than they do in their transmitter counterparts.

D. MULTIPLE INPUT MULTIPLE OUTPUT CONTROL

MIMO research is being extensively conducted by the wireless community, and many works use SL to obtain viable solutions to challenging problems. One such work implemented by Klautau *et al.* [77], attempted to train a SL model to perform beam-selection in a vehicular network. The authors sought to address data collection challenges by using Remcom's Wireless InSite ray tracing simulator in conjunction with the Simulation of Urban Mobility (SUMO) [119] PHY traffic simulator to generate data sets. The data mapped vehicle location and dimensions to the lowest loss transmitting and receiving pair of antennas between a vehicle and Road Side Unit (RSU) in site-specific, randomly generated Vehicle-to-Infrastructure (V2I) environments. A channel matrix H was computed for each time instance of the episodes within the environment.

The authors trained a SVM with a linear kernel and a decision tree, along with their ensemble add-ons (adaboost, forest), and a deep softmax regression model to predict on testing data (Figure 10). Additionally, both shallow and deep logistic regression models were trained using continuous space azimuth and elevation angles for lowest loss receiving and transmitting antenna pairs in order to see if modeling the problem as a regression instead of classification problem would improve results.

While using the Adaboost [120] and random forest [121] ensemble methods improved classification accuracy, the deep softmax regression model performed the best on test data, especially non-line-of-sight data, at 63.8% and 38.1%, respectively. The regression models performed poorly, with $4.8^o/49.9^o$ Root Mean Square (RMS) error for transmitting elevation and azimuth, and $6.2^o/102.8^o$ for receiving elevation and azimuth. While no metric for direct comparison between the classification and regression models was presented, the authors commented that the regression accuracy was prohibitively low. The authors also noted that the deep softmax regression model achieved 100% classification data on training data samples.

The authors trained SL models to learn how to map between vehicle positions and dimensions in order to achieve the lowest loss beam pair without knowing the ray tracing model, its location-specific parameters, the MIMO pre- and post-coding code books, or how the channel information matrix and code books would classify the optimal beam pair. This showcases how universal approximators learn direct relationships between noisy observations and unknown classes with no analytical alternative.

This reference presents several lessons for researchers interested in this topic. The 61-class data targets outperformed the elevation/azimuth angle targets because the discretization process introduced additional information. By providing a set of known beam pairs, the authors eliminated many sub-optimal angles as possible model predictions, whereas the regression models had to learn those angles are sub-optimal. While this was not discussed by the authors, it is likely that the angle targets were investigated as an alternative because small errors in the predicted angle do not impact MIMO performance, while picking the wrong beam pair can result in more significant performance issues.

Ensemble training of the SVM and decision tree models using the Adaboost [120] and random forest [121] schemes gave large gains to the classification accuracy, which implies that the predictions of each trained model possessed a low covariance c between each other. The expected squared error of an ensemble of n models is $\frac{v}{n} + \frac{c(n-1)}{n}$ (see Section V-H), so this application may have been close to achieving to the ideal $\frac{1}{n}$ error reduction. With no information as to how many models were trained by each ensemble protocol, it is difficult to describe the exact relationship between the model count n and the error. Ensemble training of the deep softmax and logistic regression models were not presented, potentially due to the fact that a nominal gain was achieved via the training of a single instance of the model.

The deep softmax regression model outperformed the linear SVM model, especially when classifying beam pairs in the more complicated non-line-of-sight data samples. This is an example of under-fitting, where a linear trend line cannot separate each combination of the 61 classes with high accuracy. It would have been interesting to observe how the SVM model performed when using non-linear kernels such as the polynomial or RBF kernel, and if under-fitting could have been avoided by decreasing the margin.

While no information was given about the DT or random forest models used, it is known that DT models are especially unstable SL algorithms that benefit from ensemble learning techniques [120], [121]. Consequently, it is expected that the classification performance of a single DT will be worse relative to other SL classifiers, and that ensembles of DTs will also possess this poor performance if not composed of a sufficient number of models.

Lacking information about the deep softmax regression model, the over-fitting problem can potentially be avoided using model depth/width reduction (Section V-A), weight regularization (Section V-C), simulating additional training samples (Section V-F), or hyper-parameter validation (Section V-G).

There exists a large number of open research problems that focus on improving the generalization of trained beam-selection SL models, even for testing data simulated from the same distribution as training data. This challenge extends further to real, data-scarce MIMO systems, where experimental test data is generated using a different, unknown distribution instead of simulated training data.

In the broader field of SL models applied to MIMO classification and regression problems, there are also many open PHY layer challenges. Time Division Duplexing (TDD) of MIMO systems assumes reciprocity of the wireless channels and radio hardware, and SL-based calibration techniques may achieve more accurate reciprocity relative to traditional techniques. Pilot reuse in low coherence interval or high channel delay spread scenarios is vulnerable to pilot contamination, which may be mitigated by innovative usage of SL algorithms

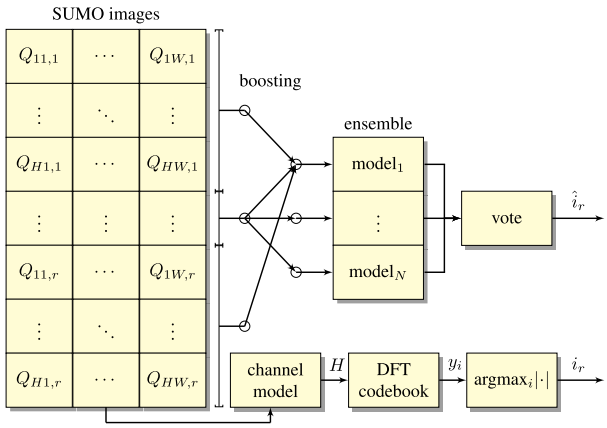


FIGURE 10. SL-based MIMO control as designed by Klautau *et al.* [77] was performed using beam selection. SUMO pixel maps $Q_{s,r} \in \{-1, 0, 1\}$ indicated if a pixel was occupied by a receiving vehicle, a non-receiving vehicle, or no vehicle. A subset of receivers r were selected by each SL model in the ensemble to maximize independence. The model is learning how the ground truth optimal beam pairs $\hat{i}$ are estimated via the channel model, pre- and post-coding MIMO codes, and a simple maximization function.

to enhance pilot allocation, improve estimates of the channel to remove interfering pilots, or refine the design of pilot sequences. Phase drift can also contaminate pilots, an issue traditionally mitigated by PLL, but SL-based phase error estimation and correction algorithms have the potential to exceed PLL performance. The upper bounds on MIMO performance assume a favorable propagation environment, meaning one with minimal fading and BSs that possess similar channel responses across all terminals. Large array orientation and antenna count planning, as well as coding schemes, can be optimized using SL models, despite unfavorable propagation environments. The large amounts of base-band data generated in MIMO systems require the rapid, distributed, and coherent online training of weights. Finally, carefully designed SL models achieve a lower cost and power consumption than expensive hardware such as ADCs, which drives down the costs of massive MIMO base stations.

E. ANALOG TO DIGITAL & DIGITAL TO ANALOG CONVERSION

SL-based DAC/ADC works surveyed in Section III-B build upon the Hopfield ADC [60]. The state-of-the-art Hopfield ADC was designed by Danial *et al.* [57], which proposed the first Hopfield DAC and a novel learning rate function. Using a logistic regression model, the authors trained the resistance values of a 4-bit memristor DAC to calibrate in the presence of noise and manufacturing errors (Figure 11). To train the model, a pulse width modulator circuit produced SL targets by periodically producing all 16 possible analog output voltages in a stair pattern. Model inputs were a 4-bit array, where the weight or resistance value of a memristor was only updated if that bit was on for that input sample. Additionally, the least significant bit was annealed due to frequent oscillations in training, while the most significant bit was updated by the full learning rate when active. The proposed DAC achieved 0.12 integral non-linear Least Significant Bits (LSB), 0.11 differential non-linear LSB, and 3.63 effective bits.

SL algorithms frequently have their universal approximator characteristic leveraged to the extent of offering end-to-end solutions. Instead of using a deep NN to map binary blocks of data directly to analog wave forms, this work gives an example of using just four weights in a shallow model to guide an analog circuit to produce those mappings. Additionally, the weights were constrained by a custom training method of the authors' design to achieve a lower voltage bound when each memristor was activated in order to ensure proper functionality of the DAC. Both of these domain-specific weight regularizations facilitated the training convergence, side-stepping the need for the model to learn the inverting amplifier, memristor architecture, or binary-weighted effective resistance distribution.

The analyzed work also presents a good example of developing a custom learning function. Weight updates and learning rate annealing functions (Section V-A) are well-explored areas of research, with novel learning rate functions typically being domain-specific and only useful for the problem they are targeting. By noticing that frequent changes in the Least Significant Bits (LSB) of training bits were causing oscillating training loss, the authors were able to determine the need for a custom learning rate function.

One of the more difficult challenges faced by the original Hopfield ADC was the programmability and re-programmability of resistance values, which was largely solved by the use of memristors. Current open challenges in designing SL-based DAC/ADC solutions and Hopfield converters include electromagnetically-accurate memristor programming. One such recent work identified that memristor voltage drops greater than 0.2 Volts cause the programmed resistance value to change over time without being re-programmed [122]. Any architecture that deviates from the Hopfield design would be significant in this research area, if an improvement to converter speed or accuracy can be achieved. Benchmark data sets have not been developed for this area of research, such that authors have not drawn direct comparisons between their improved models and the state-of-the-art. Consequently, there is no quantitative analysis available to determine the improvement brought by the use of CMOS memristors [58], custom regularization schemes [59], and the inverse DAC problem [57].

F. AUTOMATIC CODING AND MODULATION

PHY layer SL-based applications have recently received significant attention due to the innovative work of Deepsig and their demodulation [49], [50], [123], error control coding and decoding [124]–[126], channel estimation [127], and signal classification [48], [128]–[130] implementations, supplemented by published data sets. O'Shea and Hoydis [125] explored the use of a CNN demodulator and a Variational Auto-Encoder (VAE) transceiver with a focus on error control coding. While a VAE did optimize the weights during training

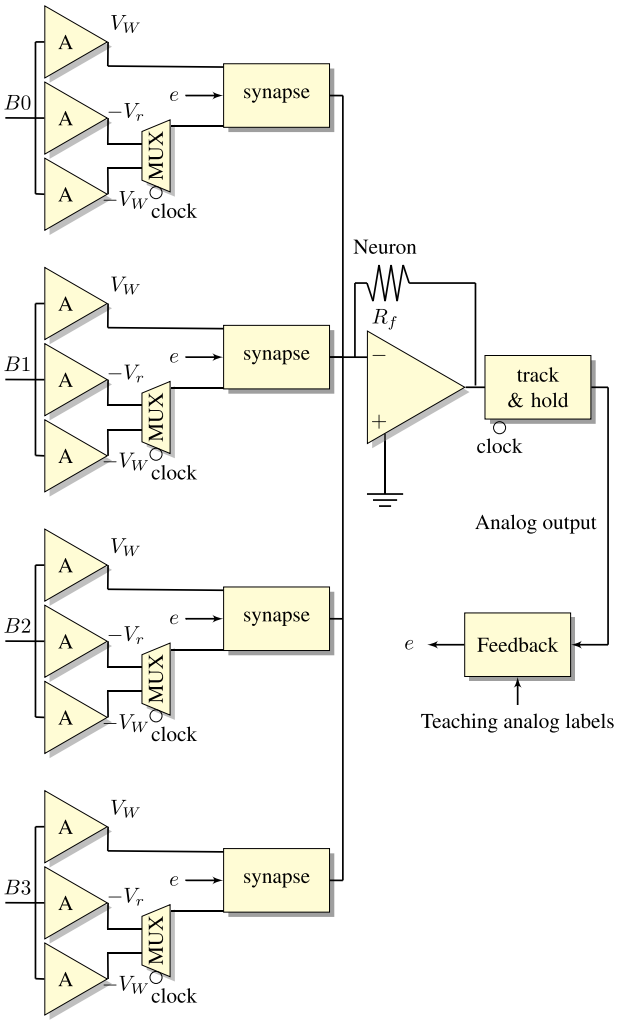


FIGURE 11. The Hopfield DAC designed by Danial *et al.* [57], which converted four-bit code sequences into analog wave forms. The target labels were generated by a periodic staircase signal, and each synapse was a small circuit with a main component that was a memristor with programmable resistance. The amplifier feedback circuit is analogous to an analog neuron because its input is a sum of synapses.

to minimize the reconstruction error, this family of models is still considered to be from the UL or semi-supervised paradigm. Consequently, our analysis of this work will focus on the proposed CNN demodulator, which was trained using IQ sequences of base-band data to classify one of ten possible modulation schemes used by an intercepted transmission. The authors found that a shallow, two layer CNN with three dense layers and a softmax layer achieved the best results. The trained model was evaluated using test signals across a range of SNRs, achieving 100% classification accuracy for seven of the classes when the SNR exceeded 10 dB, and about 50% for the other three. Classification accuracy for all 10 classes peaked at 88% at 10 dB SNR, and matched the guessing accuracy at -20 dB SNR.

While both our survey in this work and the O'Shea and Hoydis [125] acknowledged that modulation and many other PHY layer tasks can be viewed as regression or classification tasks to be solved using SL, this short survey presents several insightful lessons regarding the application of SL algorithms. To fully understand these lessons, a closer look into their data set generation software is needed, which uses GNU Radio's dynamic channel model package.

The use of a non-flat frequency channel in simulating the data was required for the authors to design the CNN in order to enforce both the input equivariance with pooling layers and the use of 172 filters, a high number for a relatively low-dimensional problem.

The issue of multicollinearity was experienced in this work, and was the potential cause of the implementation falling short of the 100% classification accuracy for high SNR signals possessing short 128-sample length observations. Ambiguities arising from the shared constellation points of two pairs of classes could have also contributed to this result. Unfortunately, this claim was not proven nor was a solution identified using material from Section IV-D.

The authors did report several lessons learned via their own experiences during this work, including the notion that SL models can be trained to perform most successfully by training them first with high SNR signals, then with gradually decreasing SNR signals. They also found that each PHY layer problem has a single optimal choice of data representation, which most strongly correlates noisy observations to the unknown parameter. Examples given include FFTs and spectrograms for carrier estimation and IQ data for modulation classification.

As higher order modulation schemes gain popularity given their ability to push larger data rates in modern wireless communication systems, online training approaches for SL-based ACM models have become challenging due to the large state-space of possible messages sent. Given the unpredictable and time-varying nature of wireless hardware and channels, there is a significant need for the implementation of data augmentation algorithms to allow for the online training of models using very few training samples.

Without the presence of cooperating transmitters exchanging training data with targets known *a priori*, these SL-based signal classifiers must instead be trained offline. Making weights generalizable is an active, multi-disciplinary, multi-domain challenge with several existing regularization and training analysis solutions.

While Deepsig has published a set of modulation classification data sets [126], [131], there still exists a significant need for accessible data sets that can be used across a range of various PHY layer applications. Such data sets allow for the evaluation and comparison of novel models, and drive innovation to solve unique problems presented by the behavior of the data. The peer review and circulation of such data sets take time, requiring the capture of real-world behavior without being too specific regarding radio hardware or wireless channels.

V. SL GUIDELINES

Implementing an SL algorithm requires the consideration of several design factors, including the depth and width of

the CNN and RNN, the ordering, type, and parameters of the layers in a CNN, the SVM kernel type and parameter choice, and the SGD parameters. This section covers the basic concepts and practical considerations for an SL implementation, with special attention drawn to the separate but related tasks of weight initialization (Section V-B) and updates (Section V-A).

A. WEIGHT UPDATES

The task of using SGD to compute the global loss minimum is non-trivial. Several versions of SGD have been developed [43], [132]–[135], each placing different emphases on how to decrease the magnitude of weight errors, perform fewer calculations, and lessen the sensitivity to initial parameter choices. In general, standard weight updates are defined as the weight matrix w changing after each gradient calculation:

$$w_{(i+1)} = w_{(i)} - \eta \nabla_{w_{(i)}}, \tag{24}$$

where the hyper-parameter η is the learning rate, and $\nabla_{w_{(i)}}$ is the current weight gradient.

Typically, weight updates are halted after training over a pre-determined number of epochs, or after the gradient becomes smaller than a convergence threshold. Additional stop conditions can be implemented, such as convergence patience, where the gradients must be below a convergence threshold for a sequential number of epochs before stopping. Weight values converging to local loss minima is avoided by computing gradients from small batch sizes or by using robust weight update techniques (*i.e.*, equation (25)). The state-of-the-art theory for understanding loss curves was described by Nakkiran *et al.* [136], who explain and expand on the ''classic'' bias-variance curve (Figure 12). The reference describes how previous loss curve behavior theories expect SL models to be initially ''classically under-trained'' as the model is trained over epochs. During this period, the loss curves indicate that the models under-fit (*i.e.*, training and validation error are relatively high) until an critical epoch is reached. After this critical point, all future training epochs will show loss curves that indicate the model is increasingly over-fitting, demonstrated by a growing difference between a small training error and relatively large validation error.

However, the authors found that, in practice, if model complexity and duration are large enough to achieve near zero training error, increasing the model complexity or training duration further will eventually decrease testing loss after reaching the ''interpolation threshold''. This phenomenon is called model-size double descent, or epoch-wise double descent, because a ''double descent'' shaped test error curve is observed. Finally, the authors show that, by varying the amount of training data available, the location of the interpolation threshold will shift, such that test error can increase if training is performed for the same number of epochs.

One innovation applied to the standard SGD is the development of ''momentum'' updates that attempt to achieve faster convergence [137]. The term momentum is used as a physical analogy for the optimization problem that is SGD, where the current weights of a model can be thought of as a ball rolling around in a mountain range. Standard SGD updates have been found to be slow to converge for non-spherical gradients, and unable to leave local loss minima ''valleys'' if the learning rate is too small. Momentum updates, which were inspired by position, velocity, and acceleration models from physics, were developed to solve both of these issues. See Table 10 for a survey of momentum-based SGD updates. In such a weight update scheme, the weights w are updated proportionally to their ''velocity'' as:

$$v_{(i)} = \mu v_{(i-1)} - \eta \nabla_{w_{(i)}}, \tag{25a}$$

$$w_{(i+1)} = w_{(i)} + v_{(i)}, \tag{25b}$$

where the velocity is initialized as $v_{-1} = 0$, and the momentum μ is a tunable hyper-parameter. The current velocity value is defined as $v_{(i)}$, and $v_{(i-1)}$ is the velocity of the weights from the previous epoch of training over epochs $i = 1, \ldots, n_e$.

At the start of training, it is common for the ''ball'' to descend a steep and tall ''mountain'', such that its velocity is so high that the weights ''overshoot'' the loss minima, and ascend the gradient of another loss ''mountain'' or circle the ''valley''. The Nesterov momentum [132] is one of many SGD momentum schemes developed to mitigate these stability issues. It does so by computing velocity that is proportional to the difference in the velocity of the i^{th} and $(i-1)^{th}$ training iteration, making it much harder for weight updates to become unstable (*i.e.*, $\mu v_{(i-1)} \gg \eta \nabla_{w_{(i)}}$). For the i^{th} SGD backward pass, the Nesterov momentum update on $w_{(i)}$ is defined as:

$$v_{(i)} = \mu v_{(i-1)} - \eta \nabla_{w_{(i)}}, \tag{26a}$$

$$w_{(i+1)} = w_{(i)} + (1+\mu) v_{(i)} - \mu v_{(i-1)}. \tag{26b}$$

Another category of SGD algorithms use higher order statistics to enhance the depth and speed of the convergence at the cost of computational efficiency via adaptive learning rates. Newton's method, which is defined as:

$$w_{(i+1)} = w_{(i)} - \frac{\nabla_{w_{(i)}}}{\nabla^2_{w_{(i)}}}, \tag{27}$$

utilizes the Hessian matrix $\nabla^2_{w_{(i)}}$, derived from the 2nd order Taylor series expression for $w_{(i)}$, to find the root of the gradient. This second order approach is very powerful because it solves directly for the vertex or loss minima, which allows the weights to converge to the nearest local loss minimum in a single update. Thus, this method requires no learning rate η, nor a learning rate annealing function.

However, there are several shortcomings to Newton's method. If the nearest loss minimia is not the global loss minima, the weights can easily converge to the wrong values without the aid of momentum approaches. Additionally, several ''Quasi-Newton'' SGD algorithms (see Table 11) have been developed using the approximation $\nabla^2_{w_{(i)}} \approx (\nabla_{w_{(i)}})^2$, since the Hessian is too computationally expensive proportional to the number of weights m, *i.e.*, $\mathcal{O}(m^3)$ versus $\mathcal{O}(m^2)$.

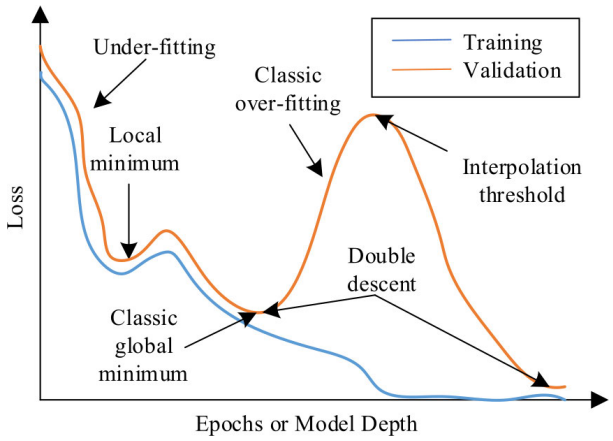


FIGURE 12. When model complexity is small compared to the number of training samples, the test loss follows a U-shaped, "classic" bias-variance tradeoff. A model is classically under-trained when both training and validation loss are high, which can be mitigated by training for more epochs, gathering more data, adding more weights to the model, or validating SGD hyper-parameters (Section V-G). Typically, during SGD a model will under-fit until it begins to over-fit, which is when training loss is low but validation loss is high. Over-fitting can be mitigated by training for fewer epochs, decreasing the number of weights in a model, voting via ensemble learning (Section V-H), or constraining the model with regularization (Section V-C).

The approximation $\nabla^2_{w(i)} \approx (\nabla_{w(i)})^2$ only holds true for full batch SGD, which unfortunately requires the entire training data set to be held in temporary memory which is sometimes impossible. As the batch size decreases, the approximation is decreasingly accurate, and the use of Quasi-Newton SGD methods may become inappropriate. Adam [135] is one such Quasi-Newton algorithm, which updates weights as:

$$m_{(i)} = \beta_1 m_{(i)} + (1-\beta_1)\nabla_{w(i)}, \tag{28a}$$

$$m_{(i)} = \frac{m_{(i)}}{1-\beta_1^i}, \tag{28b}$$

$$v_{(i)} = \beta_2 v_{(i)} + (1-\beta_2)(\nabla_{w(i)})^2, \tag{28c}$$

$$v_{(i)} = \frac{v_{(i)}}{1-\beta_2^i}, \tag{28d}$$

$$w_{(i+1)} = w_{(i)} + \frac{-\eta \times m_{(i)}}{\sqrt{v_{(i)}} + h}. \tag{28e}$$

Adam combines the acceleration/de-acceleration of the learning rate present in RMSprop and the second order statistics of Newtons method. β_1 and β_2 serve as momentum hyper-parameters similar to Nesterov's μ. The term h is used to avoid division by zero errors. Finally, $(\nabla_{w(i)})^2$ is used to solve for the nearest vertex. As long as the Hessian approximation holds, Adam can be thought of as the best of both RMSprop and Newton's method. Consequently, Adam has seen significant use in all ML applications.

For large data sets, deep models, and slow-converging SGD algorithms, the interested reader is directed towards parallel and distributed SGD frameworks such as "Hogwild!" [3], which performs well when data is sparse, or when most elements of the data are near-zero valued. Downpour SGD [138] performs a variety of boosting, where many copies of the model are trained on different subsets of data. However, since the models do not communicate with each other, divergence is a constant concern with this framework. Delay-tolerant Algorithms for SGD [139] have been shown to work well by adapting SGD to past gradients and by updating delays between devices. Elastic Averaging SGD [140] links asynchronous model updates to a central "elastic force" variable to allow for more exploration of the parameter space. Finally, Tensorflow [19] splits a computation graph into sub-graphs for distributed SGD.

While some of the algorithms discussed so far autonomously anneal the learning rate, the learning rate η can additionally be annealed in any SGD algorithm by an "annealing function" to increase weight convergence depth and speed. Step decay [43] is one such annealing function, which reduces the learning rate every s weight update steps by a factor α:

$$\eta_{(i)} = \begin{cases} \eta_{(i-1)} - \alpha, & \text{if } i \mod s = 0 \\ \eta_{(i-1)}, & \text{otherwise,} \end{cases} \tag{29}$$

where $\alpha \in [0, 1]$ and integer s are hyper-parameters. An aggressive annealing function is exponential decay [43]:

$$\eta_{(i)} = \eta_{(i-1)} - \alpha_0 e^{-ki}, \tag{30}$$

where the values α_0 and k are hyper-parameters. This annealing function is more dependent on its hyper-parameters and number of update steps than (29), as the learning rate can quickly become too small to move, even in steep and convex regions of the gradient.

Inverse decay [43] is another annealing function which was developed as a middle-of-the-road option between (30) and (29):

$$\eta_{(i)} = \eta_{(i-1)} - \alpha_0/(1+ki). \tag{31}$$

Finally, an annealed gradient noise [148] can be introduced into SGD updates in order to make weight updates more robust to poor initialization:

$$w_{(i)} = w_{(i-1)} + \mathcal{N}\left(0, \frac{\eta}{(1+i)^\gamma}\right), \tag{32}$$

where near the end of training the added noise will increase training error if not properly annealed by decay term γ, and near the start of training the added noise will not be of benefit if annealed too quickly.

B. WEIGHT INITIALIZATION

If each layer of a L-layer deep NN model scales inputs by k, the final scaling will be k^L, such that for large L, we get $\nabla_{w(i)} = 0$, $k < 1$ (the gradients "vanish"), or $\nabla_{w(i)} = \infty$, $k > 1$ (the gradients "explode") [149]. This means that, weights in deep learning, or any non-convex optimization problem, must be initialized to avoid prohibitively long training times. However, weights cannot be initialized at the same value. For example, a one input, four neuron NN would compute linear logits as $z = xw_0 + xw_1 + xw_2 + xw_3 + b_0 + b_1 + b_2 + b_3$. If all weights and biases are equal, then the

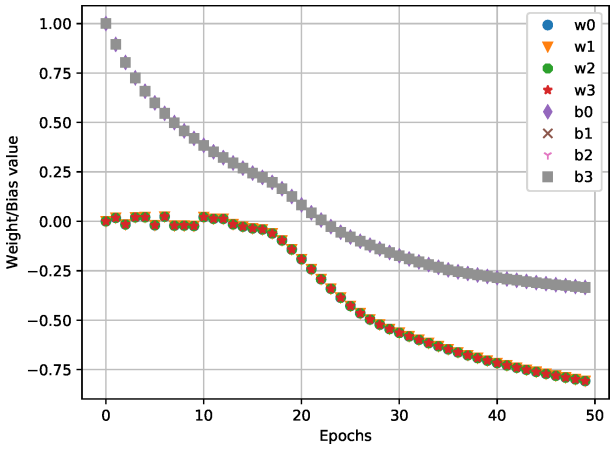


FIGURE 13. Bias and weight values of a four neuron NN over 50 epochs of SGD, initialized as $w = 0$ and $b = 1$. Consequently, all weights and biases compute the same gradient update for each epoch $\frac{\delta z}{\delta x} = w$, making weight convergence to values that correspond to the global loss minimum impossible.

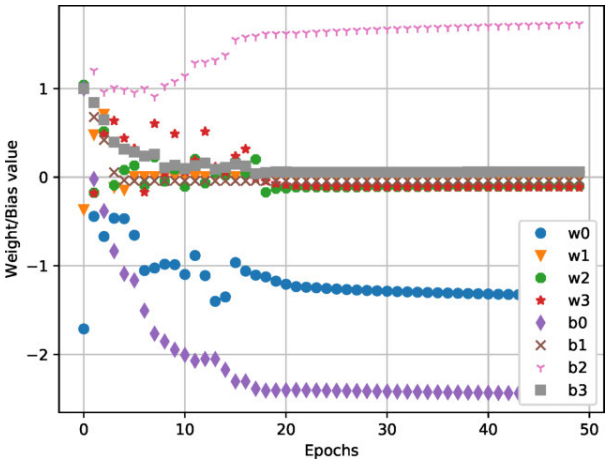


FIGURE 14. Bias and weight values for a single input, four neuron NN over 50 epochs of training, initialized via the Xavier initialization. Gradient computations are now able to take non-zero, unequal values and achieve convergence to values that correspond to a global loss minimum.

gradient for each weight will be equal (*i.e.*, $\frac{\delta z}{\delta x} = w$), such that each neuron learns the same information (Figure 13).

In order to avoid exploding or vanishing gradients during training, weight initialization must follow two rules concerning the activations a or outputs of each layer of the NN:

$$E[a^l] = E[a^{l-1}], \quad \forall l, \tag{33}$$

$$Var(a^l) = Var(a^{l-1}), \quad \forall l. \tag{34}$$

Kumar [150] shows that any initialization scheme that follows these two rules in a NN with linear-near-zero activations such as tanh or sigmoid will express the equality:

$$Var(a^l) = n^{l-1} Var(w^l) Var(a^{l-1}), \quad \forall l. \tag{35}$$

This equality serves as the justification for Xavier (also called Glorot) initialization [150], in which all weights of an n-neuron layer are initialized according to samples from the distribution:

$$w^l \sim \mathcal{N}\left(0, \frac{1}{n^{l-1}}\right), \tag{36}$$

$$b^l = 0, \tag{37}$$

where $Var(w^l) = \frac{1}{n^{l-1}}$ such that equation (35) satisfies constraint (34). The results of this can be seen in Figure 14, where each gradient is unique and each neuron learns different weight updates with no issues concerning vanishing or exploding gradients. Zero-mean normal or uniformly distributed initialization schemes without the proper variance scaling, as seen in Xavier initialization, is sufficient for shallow models. However, they will experience vanishing or exploding gradients with deeper models because they fail to satisfy the constraint (34).

Many deep NN models do not use linear-near-zero activations, such as deep ReLU CNNs. The authors of Kaiming initialization [151] re-derived equation (35) with ReLU instead of tanh activations to determine that the simple modification:

$$w^l \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n^{l-1}}}\right), \tag{38}$$

$$b^l = 0, \tag{39}$$

must be made to the initialization scheme such that the constraint (34) holds.

There also exists a truncated version of Kaiming initialization (and any Gaussian-based initialization), where the weight initializations are bounded by $-1 < w^l < 1$ to avoid sampling large weights, which becomes more common in larger NNs as $n \to \infty$ and $L \to \infty$ and the normal distribution is sampled from more times.

C. MODEL REGULARIZATION

Regularization in SL is to make an assumption and to constrain a model to that assumption. Examples include the assumption of a continuous range of optimal weights and the constraint of the tie-breaking term $\frac{1}{2nC}||w||^2$ in the non-kernelized dual SVM loss function (see Section II-E), the assumption of input equivariance and constraint of filters in CNNs (see Section II-C), and the assumption of varying input dimensions and constraint of hidden nodes in RNNs (see Section II-D).

The given CNN and RNN regularization examples are specifically called regularization by weight sharing. With all else equal, a CNN or RNN model will have fewer weights than a logistic or softmax regression model due to its filters (see Section II-C) and hidden memory units (see Section II-D). To summarize those sections, CNN filters compute logits via small convolutional windows instead of fully connecting every input from the previous layer. RNNs, on the other hand, maintain a small number of hidden memory states whose logits and activations are a superposition of all previous input values to that layer. These models with relatively fewer weights have decreased computational costs

TABLE 10. A summary of momentum-based SGD methods and their improvements upon each other.

Name	Reference	Update	Hyper-parameters & Terms	Advantages	Disadvantages
Vanilla	[141]	$w_{(i+1)} = w_{(i)} - \eta\nabla_{w_{(i)}}$	$\eta \approx 0.01$: learning rate	Simple, stable	Slow
Momentum	[137]	$v_{(i)} = \mu v_{(i-1)} + \eta\nabla_{w_{(i)}}$ $w_{(i+1)} = w_{(i)} - v_{(i)}$	v: velocity, $\eta \approx 0.01$: learning rate, $\mu \approx 0.9$: momentum	Faster convergence than vanilla	Unstable
Nesterov Momentum	[132]	$v_{(i)} = \mu v_{(i-1)} + \eta\nabla_{w_{(i)}}$ $w_{(i)} + (1+\mu)v_{(i)} - \mu v_{(i-1)}$	v: velocity, $\eta \approx 0.01$: learning rate, $\mu \approx 0.9$: momentum	Stability via "look ahead"	Not adaptive
AggMo	[142]	$v^t_{(i)} = \beta^t v^t_{(i-1)} - \nabla_{w_{(i-1)}}$ $w_{(i+1)} = w_{(i)} + \frac{\eta}{K}\sum_{t=1}^{K} v^t_{(i)}$	$\beta^t = 1 - a^{t-1}, i = 1, ..., K$: damping function, $a \approx 0.1$: damping base, $\eta \approx 0.01$: learning rate, $K \approx 3$: count	Stability via redundant damping	Many hyper-parameters, not adaptive

as they have smaller matrices to compute. They also have reduced risks of over-fitting, because fewer weights compute lower-order estimation and classifications of unknown parameters.

The given SVM regularization example is specifically called a regularization loss term. Loss terms can be included in the loss function (*i.e.*, categorical cross entropy or Mean Squared Error (MSE) loss) of any model by summation. Common loss terms are a function of the p-norm of the weights $||w||_p = (\sum w^p)^{1/p}$, and are implemented with the goal of forcing the model to learn certain weight values. Examples of loss terms include L1-norm regularization:

$$\lambda||w||, \tag{40}$$

which forces weights to take on smaller values, and is weighted against the rest of the loss function by hyper-parameter λ. Interestingly, this behavior can equivalently be produced by adding Laplacian-distributed random samples to all training inputs, as outlined by Li and Liu [152]. Another example is the L2-norm or Tikhonov regularization:

$$\frac{\lambda}{2}||w||_2, \tag{41}$$

which also forces weights to be smaller but also forces there to be fewer outliers as $p \to \infty$. This behavior may alternatively be enforced by adding Gaussian-distributed random samples to all training inputs, as outlined by Li and Liu [152]. By increasing the prediction error via loss terms, models trained with L1 and L2 regularization do not learn larger weight values unless their reduction of the prediction loss is larger than their increase of the regularization loss. This constraint mitigates arbitrary learning and creates models that generalize better to testing data.

Dropout [153] is another regularization technique, implemented when a model is assumed to be over-fitting because it has too many weights. Rather than reducing the number of weights in a model, constraining a model by using Dropout allows a subset of weights to be randomly selected and have their values set to zero for a single forward pass. In addition to mitigating over-fitting, Dropout has been found to also mitigate neuron co-adaptation as a tie-breaking constraint [153]. Neuron co-adaptation is the ambiguity of optimal weights caused when the linear logit of a neuron computed during training is equal to zero. This creates ambiguity because the inputs could either all be zero, or their weighted sum could be equal to zero. By randomly dropping weights, the weighted sum can be shifted to a non-zero value, which clarifies the ambiguity.

In SGD, it has been found that the weights converge the fastest when the distribution of weights at each layer in a NN possess zero mean with an identity covariance matrix [154]. Consider an NN with inputs x, hidden layer h, the input weight matrix g_1 and the output weight matrix g_2, and predictions $\hat{y} = g_2(g_1(x))$. During training, backpropagation dictates that $P(h)$ and $P(\hat{y})$ will be updated from the gradient $\frac{\delta f}{\delta g_1}$. It is at this point that "internal covariate shift" occurs: when $P(g_2)$ is updated according to $\frac{\delta f}{\delta g_2}$, $P(h)$ is no longer the same value as it was when $\frac{\delta f}{\delta g_2}$ was calculated. Batch normalization is a form of model regularization in which the i^{th} activation of layer l is controlled:

$$a^*_{i,l} = \frac{a_{i,l} - \mu_l}{\sqrt{\sigma_l^2 + h}}, \tag{42}$$

where $h \approx 0^+$ to avoid divide by zero errors. By using batch normalization in our example, the distribution $P(h)$ would be held constant, such that all gradients are accurate, and internal covariate shift is reduced.

D. DATA REPRESENTATION

In wireless communications, the same analog signal can be represented by a significant number of axes or spaces [155], including amplitude-phase, IQ or complex space, Gram-Schmidt basis vectors, Fourier transforms, Laplace transforms, spectral correlation functions, cyclic autocorrelation functions, wavelet transforms, and spectrograms or "waterfalls". If sampled, the same signal can be represented as digital bits, code-words, symbols, Z-transforms, or any number of categorical attributes such as signal detected/signal not detected, signal transmitted by user 1/user 2, and duration since last transmission has been short/medium/long. This number of data representation choices becomes even more daunting when common arithmetic signal processing operations, such as auto and cross correlation, dot and cross products, energy and power calculations, spectral coherence, and phase, amplitude, or frequency differences, are applied in order to compare two or more of the signal representations mentioned thus far.

Luckily, the ideal data representation to choose for SL model inputs is usually given by existing unknown parameter estimation and classification methods. For instance, control coding schemes classify bits, and demodulators classify sequences of IQ data. There are two reasons behind

these choices that should serve as a guide when no non-SL estimator or classifier is available: inputs should have low correlation and low multicollinearity. In other words, each input should be independent, and each input should have unique indicators that tie it to the ground truth value of its unknown parameter [155].

Input independence is important because models that train on many subsequent inputs of the same class generalize poorly. When many of the initial weight updates during SGD are computed with respect to highly dependent inputs, the model will consequently test successfully on that class, and unsuccessfully on all others. This input dependence is most commonly mitigated by shuffling the training data set and its corresponding labels. The correlation between two continuous signals $f(t)$ and $g(t)$ can be quantified as:

$$(f * g)(\tau) = \int_{-\infty}^{\infty} f^*(t)g(t+\tau)\delta t. \tag{43}$$

Low multicollinearity is important because highly similar inputs with different unknown parameters cannot be correctly estimated or classified. Multiple data representations may be incorporated into a SL model using early, middle, or late integration (Figure 15) in an effort to decrease multicollinearity. Early integration involves combining the data before the SL model, either via the concatenation of the data, or the addition of another dimension to the data space. Middle integration can only be performed with NN models, where each data representation has an independent branch of neuron layers that are concatenated partway into the NNs architecture. Late integration involves training independent SL models for each data representation, and interprets the set of unknown parameter estimates or classifications as a single result via averaging or voting functions. A popular metric used to quantify multicollinearity is the variance inflation factor, which describes how much the variance of a trained weight changes when predictions are correlated. It is computed as:

$$\text{VIF}_i = \frac{1}{1 - R_i^2}, \tag{44}$$

where R_i^2 is the coefficient of determination for the i^{th} input feature, computed as:

$$R_i^2 = 1 - \frac{\sum_{j=1}^{n}(\hat{x}_j - x_j)^2}{\sum_{j=1}^{n}(x_j - \bar{x}_j)^2}, \tag{45}$$

and the predictions are not the label but the i^{th} input feature:

$$\hat{x}_i = b + \sum_{j=1,\ j\neq i}^{n} w_j x_j. \tag{46}$$

A rule of thumb is that a $\text{VIF}_i > 5$ indicates high multicollinearity.

E. DATA PRE-PROCESSING

While many SL models achieve universal approximator characteristics, that does not mean that data can be recklessly thrown into training without any considerations for null values, standardization, categorical variables, one-hot encoding, sample dependence, or multicollinearity.

Null values can appear in signal data sets for any number of reasons, such as poor alignment of signals or varying length samples. These null values will carry through to model predictions and gradients during training if not changed. Imputation, or the process of substituting null values, is one solution, where the null value's true value can be estimated via the mean of that feature across all signals in the data set, or via the mean of that signal across all features in that signal. If the researcher wishes to keep the null value as an indicator that something that caused a null value has occurred, imputation can instead take the form of substituting a categorical value that is outside the range of values taken by all other features (*i.e.*, a value of -1 in an otherwise all positive valued data set). Finally, if all null values occur at the start or end of signals in the data set due to signals of varying length, the signals' null values can be clipped or replaced by zeros in order to make all signals in the data set the same length. This method is typically a last resort, however, as clipping removes information, and adding zeros adds fabricated information.

Standardization of data is to force a zero mean and unit variance distribution on inputs. Symmetric distributions allow for faster convergence during SGD when using momentum, or schemes such as those discussed in Section V-A. Non-symmetric gradient ''landscapes'' cause momentum updates to ''circle the drain'', overshoot loss minima by building up speed on steep declines, and contain ''rougher'' landscapes with more local loss minima. Standardization is implemented as:

$$x_i = \frac{x_i - \mu}{\sigma^2}, \tag{47}$$

where a data sample $x_i, i = 1, \ldots, N$ is zero centered by subtracting the data set mean μ and given unit variance by dividing by the data set variance σ^2.

During classification tasks, the unknown parameter of each input takes the form of a discrete value from a set of possibilities. Inputs can also be discrete values. Both discrete or categorical inputs and outputs of SL models are either ordinal or nominal. Ordinal values are related. If a pulse amplitude modulated signal encodes signals as having amplitudes $A \in -3A, -A, A, 3A$, then signals of $3A$ are more similar to A signals than $-A$ or $-3A$ signals. Nominal categorical values have no relationship: no two values are more or less similar to each other than any other pair of values. While ordinal values can be expressed using decimal values $1, 2, 3\ldots$, nominal values cannot, or a false relationship between values will be taught to the SL model. Nominal values must instead be one hot encoded, such that if m possible categories exist and the value of that category is n, then the value is expressed as a vector of m zero-valued elements with the n^{th} element set to a value of one.

Multicollinearity can be mitigated by pre-processing. Principal Component Analysis (PCA) is an algorithm [116] that

TABLE 11. A summary of Quasi-Newton SGD methods and their improvements upon each other, some of which may also incorporate momentum-analogous terms.

Name	Reference	Update	Hyper-parameters & Terms	Advantages	Disadvantages
Adagrad	[134]	$w_{(i+1)} = w_{(i)} - \frac{\eta \cdot \nabla_{w_{(i)}}}{\sqrt{(\nabla_{w_{(i)}})^2 + h}}$	$h \approx 0^+$: smoothing term, $\eta \approx 0.001$: learning rate	Adaptive learning rate	Stops early
Adadelta	[143]	$\Delta w_{(i)} = -\frac{\text{RMS}[\Delta w_{(i-1)}]}{\text{RMS}[\nabla w_{(i)}]} \nabla_{w_{(i)}}$ $w_{(i+1)} = w_{(i)} + \Delta w_{(i)}$	RMS$[\cdot]$: RMS of exponentially decaying squared parameter updates, Δw: weight update, $\gamma \approx 0.95$: decay	Fixes Adagrad's early stopping, no learning rate	Biased updates too small at first
RMSprop	[133]	$E[(\nabla_{w_{(i)}})^2] = \gamma E[(\nabla_{w_{(i-1)}})^2] + (1-\gamma)(\nabla_{w_{(i)}})^2$ $w_{(i+1)} = w_{(i)} - \frac{\eta \cdot \nabla_{w_{(i)}}}{\sqrt{E[(\nabla_{w_{(i)}})^2] + h}}$	$E[(\nabla_{w_{(i)}})^2]$: exponentially decaying squared parameter updates, $\gamma \approx 0.9$: decay, $\eta \approx 0.001$: learning rate	Fixes Adagrad's early stopping, strong RNN performance	Biased updates too small at first
Adam	[135]	$\hat{m}_{(i)} = \frac{\beta_1 m_{(i-1)} + (1-\beta_1)\nabla_{w_{(i)}}}{1-\beta_1^i}$ $\hat{v}_{(i)} = \frac{\beta_2 v_{(i-1)} + (1-\beta_2)(\nabla_{w_{(i)}})^2}{1-\beta_2^i}$ $w_{(i+1)} = w_{(i)} - \frac{\eta \hat{m}_{(i)}}{\sqrt{\hat{v}_{(i)}} + h}$	m: first moment, v: second moment, $\beta_1 \approx 0.9$: first order decay, $\beta_2 \approx 0.999$: second order decay, $h \approx 0^+$: smoothing term, $\eta \approx 0.001$: learning rate	Adds momentum and bias correction to RMSprop/Adadelta	Exponential moving average has poor generalization, unstable momentum
Adamax	[135]	$\hat{m}_{(i)} = \frac{\beta_1 m_{(i-1)} + (1-\beta_1)\nabla_{w_{(i)}}}{1-\beta_1^i}$ $\hat{v}_{(i)} = \frac{\beta_2^\infty v_{(i-1)} + (1-\beta_2^\infty)(\nabla_{w_{(i)}})^\infty}{1-\beta_2^\infty}$ $w_{(i+1)} = w_{(i)} - \frac{\eta \hat{m}_{(i)}}{\sqrt{\hat{v}_{(i)}} + h}$	m: first moment, v: second moment, $\beta_1 \approx 0.9$: first order decay, $\beta_2 \approx 0.999$: second order decay, $h \approx 0^+$: smoothing term, $\eta \approx 0.002$: learning rate	Stabilizes Adam via ℓ_∞ normalization of second moment	Exponential moving average has poor generalization
Nadam	[144]	$\hat{m}_{(i)} = \frac{\beta_1 m_{(i-1)} + (1-\beta_1)\nabla_{w_{(i)}}}{1-\beta_1^i}$ $\hat{v}_{(i)} = \frac{\beta_2 v_{(i-1)} + (1-\beta_2)(\nabla_{w_{(i)}})^2}{1-\beta_2^i}$ $w_{(i+1)} = w_{(i)} - \frac{\eta}{\sqrt{\hat{v}_{(i)}} + h}(\beta_1 \hat{m}_i) + \frac{(1-\beta_1)\nabla_{w_{(i)}}}{1-\beta_1^i})$	m: first moment, v: second moment, $\beta_1 \approx 0.9$: first order decay, $\beta_2 \approx 0.999$: second order decay, $h \approx 0^+$: smoothing term, $\eta \approx 0.001$: learning rate	Stabilizes Adam via Nesterov	Exponential moving average has poor generalization
AMSGrad	[145]	$\hat{m}_{(i)} = \frac{\beta_1 m_{(i-1)} + (1-\beta_1)\nabla_{w_{(i)}}}{1-\beta_1^i}$ $\hat{v}_{(i)} = \max(\hat{v}_{(i-1)}, \beta_2 v_{(i-1)} + (1-\beta_2)(\nabla_{w_{(i)}})^2)$ $w_{(i+1)} = w_{(i)} - \frac{\eta \hat{m}_{(i)}}{\sqrt{\hat{v}_{(i)}} + h}$	m: first moment, v: second moment, $\beta_1 \approx 0.9$: first order decay, $\beta_2 \approx 0.999$: second order decay, $h \approx 0^+$: smoothing term, $\eta \approx 0.001$: learning rate	Fixes Adam generalization via max function in second moment	Inconsistent improvement over different data sets and models
AdamW	[146]	$\hat{m}_{(i)} = \frac{\beta_1 m_{(i-1)} + (1-\beta_1)\nabla_{w_{(i)}}}{1-\beta_1^i}$ $\hat{v}_{(i)} = \frac{\beta_2 v_{(i-1)} + (1-\beta_2)(\nabla_{w_{(i)}})^2}{1-\beta_2^i}$ $w_{(i+1)} = w_{(i)} - \eta\left(\frac{\alpha \hat{m}_{(i)}}{\sqrt{\hat{v}_{(i)}} + h} + \nabla_{w_{(i)}}\right)$	m: first moment, v: second moment, $\beta_1 \approx 0.9$: first order decay, $\beta_2 \approx 0.999$: second order decay, $h \approx 0^+$: smoothing term, $\eta \approx 0.01$: learning rate, $\alpha \approx 0.001$: ℓ_2 regularization weight	Improves ℓ_2 regularization of Adam via removing from second moment	Additional hyper-parameters
QHAdam	[147]	$\hat{m}_{(i)} = \frac{\beta_1 m_{(i-1)} + (1-\beta_1)\nabla_{w_{(i)}}}{1-\beta_1^i}$ $\hat{v}_{(i)} = \frac{\beta_2 v_{(i-1)} + (1-\beta_2)(\nabla_{w_{(i)}})^2}{1-\beta_2^i}$ $w(i+1) = w_{(i)} - \eta\left[\frac{(1-v_1)\cdot\nabla_{w_{(i)}} + v_1 \cdot \hat{m}_{(i)}}{\sqrt{(1-v_2)(\nabla_{w_{(i)}})^2 + v_2 \cdot \hat{v}_{(i)}} + h}\right]$	m: first moment, v: second moment, $\beta_1 \approx 0.9$: first order decay, $\beta_2 \approx 0.999$: second order decay, $h \approx 0^+$: smoothing term, $\eta \approx 0.001$: learning rate, $v_1 \approx 0.7$: first order immediate discount factor, $v_2 \approx 1$: second order immediate discount factor	Deeper convergence via weighted average updates from multiple algorithms	Additional hyper-parameters

removes input features from all signals of a feature-rich data set. The goal is to remove features with high multicollinearity, such that the low multicollinearity features can dominantly teach the SL model. In PCA, user-defined p low multicollinearity input features are identified by standardizing input data, computing the covariance matrix of the standardized input data, and computing the eigenvalue decomposition of the standardized covariance matrix. Principle components describe the dimension-reduced form of the data set, where each feature is a linear combination of several other features. The singular value decomposition factorization of the standardized covariance matrix cov(X_{ZC}) gives the principal components X_{PCA} as:

$$x_{i_{\text{ZC}}} = x_i - \frac{\sum x_i}{k \times D}, \quad x_i \in X, \quad x_{i_{\text{ZC}}} \in X_{\text{ZC}}, \tag{48a}$$

$$\text{cov}(X_{\text{ZC}}) = \frac{X_{\text{ZC}}^T \cdot X_{\text{ZC}}}{N}, \tag{48b}$$

$$\text{USV}^* = \text{cov}(X_{\text{ZC}}), \tag{48c}$$

$$X_{\text{PCA}} = X_{\text{ZC}} \cdot U_p, U_p \in [N, p], \tag{48d}$$

where eigenvalues are S, conjugate transpose of the unitary matrix V^*, k samples per signal, number of features $D \geq p$, number of samples N, matrix transpose X^T, and the element-wise dot product between two matrices $\cdot$

$$X \cdot Y = x_1 \times y_1 + x_2 \times y_2 + \ldots + x_{n-1} \times y_{n-1}, \tag{49}$$

for elements $x \in X, y \in Y$ of length n. Data pre-processing should always be done in the digital domain to avoid noisy data transformations, unless the SL model is directly connected to analog circuits, as is the case for Hopfield DAC/ADCs.

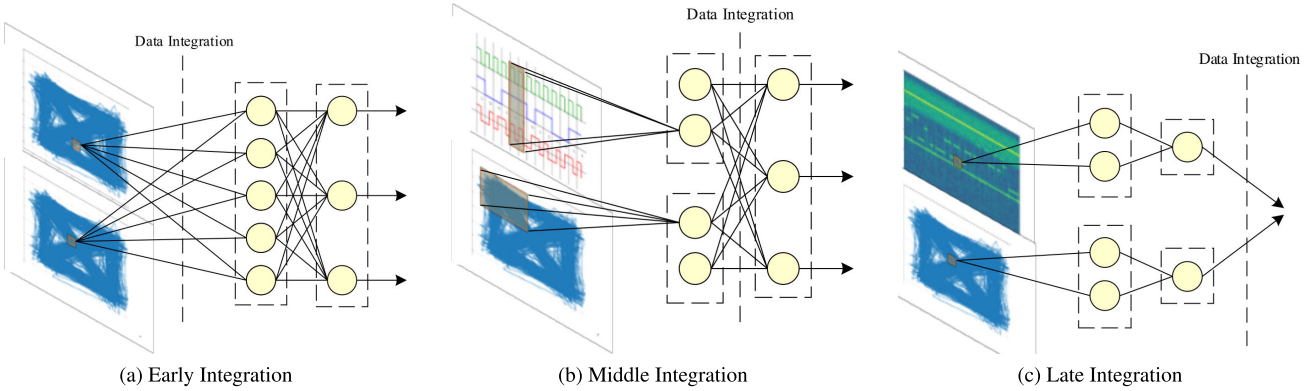


FIGURE 15. Different data representations can be integrated, via multiple approaches, in order to achieve an additional prediction performance gain by reducing multicollinearity. Data from the same domain is typically concatenated and integrated early as shown in (a). Data domains with high correlation between each other are typically integrated in the middle of a model as shown in (b), and highly dissimilar data is integrated at the end of an implementation by some voting scheme (c). Alternatively, late integration of identical inputs can be used as a form of ensemble learning in order to reduce the variance of model predictions introduced by the randomness of SGD (see Section V-H).

F. DATA AUGMENTATION

Small training data sets do not train SL models well because they do not fully describe the testing data's underlying probability distribution. If more data cannot be collected, data augmentation can instead increase the diversity of data in a small training set by estimating the underlying distribution and sampling additional signals to combat data scarcity.

A common way of estimating the underlying distribution is to train an encoding NN that maps inputs to a lower dimension $z(x)$, and a decoding NN that maps inputs back to their original values $h(z(x)) = x$. This is done by forcing the encoder weights via regularization to achieve a Gaussian behavior, $z(x) \sim \mathcal{N}(0, 1)$. If the combined encoder-decoder weights are well-trained to reconstruct images, and the encoder weights are effectively regularized, random Gaussian samples can be input to the decoder $h(\mathcal{N}(0, 1))$ to output simulated training data. This process outlines how Generative Adversarial Networks (GANs) [156] and VAEs [157] are used for data augmentation. Less popular and more constrained distribution estimators are produced through Bayesian Metropolis-Hastings sampling [158], Gibbs sampling [159], importance sampling [160], and rejection sampling [161].

Training data can also be simulated by applying plausible transforms to existing data sets. This form of data augmentation is performed by introducing simulated variations to the training data under the assumption that the test data will also be affected by those variations. Examples of plausible transforms include frequency, amplitude, phase shifts in the data, wireless channel effects such as AWGN, multi-path, scattering, diffraction, reflection, and Doppler, and hardware non-idealities such as clock drift, power amplification characteristics, and other RF front end non-linearities.

G. HYPER-PARAMETER VALIDATION

When designing a SL model, there are often several aspects of the architecture or SGD (*i.e.*, learning rate, number of neurons in a layer) that provide no clear choice towards maximizing the testing accuracy. The process of experimenting with different operating parameters and other implementation values is referred to as hyper-parameter validation [43].

One of the most straightforward methods of hyper-parameter validation is a grid search, through which a SL model is trained with the same training data that uses each combination of hyper-parameters h from the set H. In order to evaluate which hyper-parameter subset is optimal, the training data set must be partitioned into a smaller training data set and a validation set. The hyper-parameters that result in a trained model with the highest validation set accuracy are used at the testing stage of SL model deployment.

The gap between the validation and the test prediction performance is minimized when the number of samples in both partitions are large and equal in count, and when both sets are sampled from the same underlying distribution. Common partition ratios between training, validation, and testing sets include 80:10:10, 60:20:20, and 50:25:25. Ratios with smaller training sets are chosen when the data set size is large, which affects the minimization of the variance in test data performance. Smaller validation/testing ratios are chosen when the data set size is small, in order to ensure the proper training of the model. As the number of hyper-parameters used in grid search increases, computational costs become prohibitive (Algorithm 5).

If the amount of data is limited, validation accuracy can vary significantly, and suboptimal hyper-parameters can be selected. Cross-fold hyper parameter validation, where the training data is split up into k equal-sized "folds", or partitions (instead of just two, as in grid search), can mitigate this issue. For each of the k folds, one fold is chosen as the validation set, and the other $k - 1$ folds are used for training. The data split this way are grid searched. Then, a new fold is chosen as the single validation fold, and the old validation fold becomes part of the new $k - 1$ training partition. This process repeats until each fold has served as the validation fold, then the final hyper-parameters are chosen based on which hyper-parameters had the highest average validation accuracy across each of the k folds. The weights trained with

Algorithm 5 Grid-Search Validation Method [162]

procedure GIVEN TRAINING DATA X_{TR}, VALIDATION DATA X_{VAL}, SET OF HYPER-PARAMETERS H
  Best weights $w^* = 0$
  Best accuracy PPV$^* = 0$
  for Set h in H **do**
    Train model on X_{tr}, Y_{tr} to obtain w using h
    Test model on X_{val} using w to obtain $\hat{Y}_{\text{val}}$
    Compute test accuracy PPV using $\hat{Y}_{\text{val}}$
    if PPV $>$ PPV* **then**
      PPV$^* =$ PPV
      $w^* = w$
    end if
  end for
end procedure

these hyper-parameters are used for final test data evaluation (Algorithm 6). The larger k is, the higher the correlation between each learned model; the highest correlation exists for $k = N$, or leave-one-out validation. The smaller k is, the higher the variance of predictions, as the training set gets smaller. Common compromises include $k = 3, 5, 10$. Cross-fold validation makes up for the high variance that would be experienced in grid search of small training sets by averaging over all folds in order to obtain hyper-parameter choices that are not biased to any one chosen fold.

Algorithm 6 Cross-Fold Validation Method [163]

procedure GIVEN TRAINING DATA X, SET OF HYPER-PARAMETER SETS H, K FOLDS
  Partition X into K folds $X_k, k = 1, \ldots, K$
  Best weights $w^* = 0$
  Best accuracy PPV$^* = 0$
  for Set h in H **do**
    for $k = 1, \ldots, K$ **do**
      Train model on $X \not\subset X_k$ to obtain w using h
      Test model on X_k using w to obtain $\hat{Y}_k$
      Compute test accuracy PPV$_k$ using $\hat{Y}_k$
    end for
    Compute average PPV $= \frac{1}{K} \sum_k$ PPV$_k$
    if PPV $>$ PPV* **then**
      PPV$^* =$ PPV
      $w^* = w$
    end if
  end for
end procedure

This process can be repeated using nested cross-fold validation, where multiple SL models (*i.e.*, SVM, RNN, DT) are tuned to complete the same unknown parameter estimation or classification task.

H. ENSEMBLE LEARNING

A form of late integration (Section V-D) known as ensemble learning can significantly improve the performance of trained SL models. An ensemble of SL models is a collection of n models independently trained on the same data set. Ensembles perform classification or regression by averaging their predictions on test data. The benefit of this can be shown by minimizing the expected squared error of predictions, defined as:

$$E\left[\left(\left(\frac{1}{n}\sum_{i=1}^{n}\hat{y}_i\right) - y\right)^2\right] = \frac{v}{n} + c\frac{n-1}{n}, \tag{50}$$

where the variance $v = E[(\hat{y}_i - y_i)^2]$, and the covariance $c = E[(\hat{y}_i - y_j)^2]$, $i \neq j$. If the covariance is zero, or each model makes different mistakes, then the gain of using an ensemble on the squared error of predictions is $\frac{1}{n}$. If $v = c$, then the ensemble brings no gains, and the prediction MSE remains at v.

While any SL model can be trained in an ensemble, there exist a few ensemble training algorithms specific to a certain family of SL algorithms or that are designed to mitigate training issues. Bootstrapping is a technique used to reduce prediction covariance c by training each model in the ensemble with a different set of examples from the training set, sampled with replacement. The random forest [121] algorithm is an example of bagging, which trains n DT on bootstrapped data. Furthermore, random forest ensemble splits the attributes of each node of each version of the DT randomly, selecting m attributes from p each time. Attribute selection is usually done by computing the entropy of each attribute and selecting the highest entropy feature. Finally, ensembles that perform classification by bagging pick the class with the highest number of votes.

Adaboost [120] is another ensemble algorithm, where SVM or DT models are boosted rather than bagged. The difference between the two is that the classifications of boosted ensembles are weighted, and bagged ensembles have equal weight voting. The weights are determined using error computations, such that weak learners, or models with high prediction variance v, have less influence on classifications.

VI. SUMMARY & CONCLUSION

Since the RF channel performs a black box noisy transform on transmitted signals, low assumption SL-based unknown parameter estimation and classification algorithms have allowed for the development of ground-up, adaptive, fast, cost-effective, and accurate wireless receiver systems. When the appropriate requirements [164] are met, these algorithms can even be applied to real-time systems. However, a comprehensive resource for wireless researchers has been absent until now. In this tutorial, we:

- Over-view logistic and softmax regression in Section II-A, the low-assumption nature of SL models in Section II-B, CNNs in Section II-C, RNNs in Section II-D, and SVMs in Section II-E.
- Survey the foundational publications of the 35-year history of SL-based wireless PHY layer innovations in Section III-A. Additionally, we discuss recent SL-based

DSA, channel correction, AGC, MIMO, ADC, and ACM works in Section III-B.
- Analyze one popular work from each PHY layer domain from Section III-B by describing the paper's problem, solution, and lessons learned, as well as open research problems for the domain as a whole, in Section IV.
- Teach how to navigate trial-and-error SL model choice, design, and training in Section V. Specifically, we cover weight updates and initialization in Sections V-A and V-B, model regularization in Section V-C, choice of data representation and pre-processing in Sections V-D and V-E, hyper-parameter validation in Section V-F, and ensemble learning by Section V-G.

This tutorial is important for the wireless communications community, serving researchers in the following ways:

- Section I-B compares this tutorial to related works, which overwhelmingly survey and teach ML applied to network layers of the wireless communications stack and above. This tutorial provides a rare resource for lower levels of the wireless stack.
- While computational, ML, and wireless technology advances have limited innovation in SL-based PHY layer works to recent years, Section III-A acknowledges a rich history reaching back to the 1980s, providing a wide context not seen in other surveys and tutorials.
- Section III and Section IV provide a detailed view of the progress and open areas of research of SL-based PHY layer communications. In summary (Table 12), Section IV-A highlights that DSA works are in great need of QoS-monitoring spectrum sensing that guarantees minimum performance despite the use of non-linear unknown parameter estimators and classifiers. Section IV-B discusses how channel correction works leave few open challenges that are not incremental improvements to the state-of-the-art. Section IV-C shows that AGC works traditionally have improved computational efficiency, as performance improvements are not currently possible due to label-generation limitations. Section IV-D argues that MIMO works offer many open challenges, including reciprocity calibration, pilot reuse, phase drift correction, favorable propagation environment calibration, beam forming, and cost reduction. Section IV-E provides an analysis of DAC/ADC works, the most recent of which have been motivated by details noticed by those most knowledgeable of memristors' behavior in analog circuits, have been incremental improvements to the Hopfield ADC. Finally, Section IV-F presents ACM works which have been closely tied to signal classification and ground-up radio networks, both of which face the issue of small training data sets. This has motivated the need for accurate data augmentation in order to teach models a large input space given few samples. All domains face a number of common challenges, including the offline training of models that generalize well to data from different wireless channels, the generation of widely-used data sets, the data-throughput efficient training of models online, and the accurate training of models distributed over multiple radios despite noisy gradient updates.
- Section V-A describes why Adam is generally the fastest weight update method as a quasi-Newton SGD scheme, and why vanilla SGD with momentum can sometimes outperform Adam. We also briefly discuss a number of recent SGD schemes not seen in other surveys and tutorials, and why training and testing loss curves follow a double descent shape. Section V-B explains why random weight initialization needs to be variance-scaled to the model, and why the activation function matters when choosing that scaling. Section V-C explains how ℓ_p-norm regularization, dropout, and batch normalization all aid in the training of models that generalize well, which is especially an issue in wireless communications due to the random nature of wireless channels. Section V-D describes how to choose the best data representation for your SL model, why input independence and multicollinearity matter, and how to integrate data from multiple representations into one SL model. Section V-E discusses how training data probability distributions can be estimated, and how those estimations can simulate additional training data in data-scarce wireless scenarios. Section V-F discusses how to tune model hyper-parameters and when to use k-fold validation instead of grid search validation. Finally, Section V-G discusses why ensemble learning reduces unknown parameter estimate variance, and the difference between bootstrapping and boosting.

TABLE 12. A summary of SL-based PHY-layer applications and associated open challenges. There potentially exists liability issues with DSA applications regarding spectral interference issues and thus require bounds and guarantees, while MIMO control and ACM applications have generated significant interest and many open challenges within the research community. Finally, research into channel correction, AGC, and DAC/ADC applications has reached a plateau with some activities contributing to these mature areas.

Application	Opportunities for Innovation and Significant Novel Contributions
DSA	QoS monitoring and guarantees to not disrupt primary users
Channel Corrections	Efficient online training
AGC	Cost reduction and label generation that allows for performance gains
MIMO Control	Reciprocity calibration, pilot reuse, phase drift correction, propagation calibration, beam forming, and cost reduction
ADC/DAC	Incremental memristor-based improvements to cost or performance, non-Hopfield designs
ACM	Generalized training of modulation and protocol classification models robust to test data drift and adversarial perturbations

A number of popular software libraries were presented in Section I for use in creating and training ML models. These libraries handle back propagation using numerical methods, which otherwise would have to be performed by analytically

deriving each gradient using linear algebra. While we encourage the reader to experiment with these libraries and use what works best for them, in our survey of state-of-the-art works, we have found that TensorFlow [19], Keras [21], and Pytorch [23] are the most popular libraries used in open source code. These libraries are easy to learn and powerful due to their large online communities, constant development, and support from both academia and industry.

This work provides a valuable resource for teaching unknown parameter estimation and classification using SL models at the PHY layer. However, there are many other emerging research areas in wireless communications that are benefiting from ML algorithms. We acknowledge both an existing collection of works and a great need for new works that teach ML use in wireless communication systems. The reader is directed towards Section I-B for a collection of ML surveys and tutorials. There exists a significant need for tutorials that teach the use of semi-supervised and UL algorithms for probability distribution estimation and self-organizing data at every layer of the stack. Additionally, a comprehensive survey or tutorial on Reinforcement Learning (RL)-based link and network layer algorithms remains an open area of publication that would be of great value to the wireless community.

ACRONYMS

ACM	Automatic Coding and Modulation.
ADC	Analog-to-Digital Conversion.
AGC	Automatic Gain Control.
AI	Artificial Intelligence.
AWGN	Additive White Gaussian Noise.
BCH	Bose-Chaudhuri-Hocquenghem.
BPSK	Binary Phase Shift Keying.
BS	Base Station.
CDMA	Code Division Multiple Access.
CNN	Convolutional Neural Network.
CR	Cognitive Radio.
CSI	Channel State Information.
CUE	Cellular User Equipment.
DA	Data Augmentation.
DAC	Digital-to-Analog Conversion.
DC	Data Choice.
DPP	Data Pre-Processing.
DSA	Dynamic Spectrum Access.
DT	Decision Tree.
DUE	Device-to-Device User Equipment.
FFT	Fast Fourier Transform.
FIR	Finite Impulse Response.
IE	Input Equivariance.
IF	Intermediate Frequency.
IoT	Internet of Things.
IQ	In-Phase/Quadrature.
ISI	Inter-Symbol Interference.
KKT	Karush-Kuhn-Tucker.
KM	Kernel Method.
KNN	K-Nearest Neighbor.
LDPC	Low-Density Parity-Check.
LNA	Low Noise Amplifier.
LSB	Least Significant Bits.
LSR	Least Squares Regression.
LSTM	Long Short-Term Memory.
MAC	Medium Access Control.
MIMO	Multiple Input Multiple Output.
ML	Machine Learning.
MLE	Most Likely Estimation.
MSE	Mean Squared Error.
NP	Neyman-Pearson.
NLL	Negative Log Likelihood.
NN	Neural Network.
PCA	Principal Component Analysis.
PDF	Probability Density Function.
PHY	physical.
PLL	Phase-Locked-Loop.
PPV	Positive Predictive Value.
QoS	Quality-of-Service.
R	Regularization.
RBF	Radial Basis Function.
ReLU	Rectified Linear Unit.
RF	Radio Frequency.
RL	Reinforcement Learning.
RMS	Root Mean Square.
RNN	Recurrent Neural Network.
RSU	Road Side Unit.
SGD	Stochastic Gradient Descent.
SL	Supervised Learning.
SNR	Signal-to-Noise Ratio.
SON	Self Organizing Networks.
SUMO	Simulation of Urban Mobility.
SVM	Support Vector Machine.
TDD	Time Division Duplexing.
UE	User Equipment.
UL	Unsupervised Learning.
V	Validation.
V2I	Vehicle-to-Infrastructure.
VAE	Variational Auto-Encoder.
WI	Weight Initialization.
WMMSE	Weighted Minimum Mean Square Error.
WSN	Wireless Sensor Networks.
WSR	Weighted Sum Rate.
WU	Weight Updates.

REFERENCES

[1] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," in *Proc. 16th Annu. Conf. Int. Speech Commun. Assoc.*, Sep. 2015, pp. 1–5.

[2] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: A big data—AI integration perspective," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1328–1347, Apr. 2021.

[3] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild!: A lock-free approach to parallelizing stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 24. Red Hook, NY, USA: Curran Associates, 2011, pp. 693–701. [Online]. Available: https://proceedings.neurips.cc/paper/2011/file/218a0aefd1d1a4be65601cc6ddc1520e-Paper.pdf

[4] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," *CoRR*, vol. abs/1905.11946, pp. 1–11, May 2019.

[5] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3039–3071, 4th Quart., 2019.

[6] M. Bkassiny, Y. Li, and S. K. Jayaweera, "A survey on machine-learning techniques in cognitive radios," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1136–1159, 3rd Quart., 2012.

[7] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1996–2018, 4th Quart., 2014.

[8] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

[9] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A survey of machine learning techniques applied to self-organizing cellular networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2392–2431, 4th Quart., 2017.

[10] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2432–2455, 4th Quart., 2017.

[11] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2923–2960, 4th Quart., 2018.

[12] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2595–2621, 4th Quart., 2018.

[13] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo, "Thirty years of machine learning: The road to Pareto-optimal wireless networks," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1472–1514, 3rd Quart., 2020.

[14] R. K. Dwivedi, S. Pandey, and R. Kumar, "A study on machine learning approaches for outlier detection in wireless sensor network," in *Proc. 8th Int. Conf. Cloud Comput., Data Sci. Eng. (Confluence)*, Jan. 2018, pp. 189–192.

[15] M. Kulin, C. Fortuna, E. De Poorter, D. Deschrijver, and I. Moerman, "Data-driven design of intelligent wireless networks: An overview and tutorial," *Sensors*, vol. 16, no. 6, p. 790, Jun. 2016.

[16] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *CoRR*, vol. abs/1803.04311, 2018.

[17] Y. Liu, S. Bi, Z. Shi, and L. Hanzo, "When machine learning meets big data: A wireless communication perspective," *IEEE Veh. Technol. Mag.*, vol. 15, no. 1, pp. 63–72, Mar. 2020.

[18] T. J. O'Shea and J. Hoydis, "An introduction to machine learning communications systems," *CoRR*, vol. abs/1702.00832, pp. 1–13, Feb. 2017.

[19] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, and M. Isard, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2016, pp. 265–283. [Online]. Available: https://www.tensorflow.org/

[20] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems," *CoRR*, vol. abs/1512.01274, pp. 1–6, Dec. 2015.

[21] F. Chollet. (2015). *Keras*. [Online]. Available: https://github.com/fchollet/keras

[22] R. Al-Rfou *et al.*, "Theano: A Python framework for fast computation of mathematical expressions," *CoRR*, vol. abs/1605.02688, pp. 1–19, May 2016.

[23] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[24] S. Dieleman *et al.*, "Lasagne: First release," Tech. Rep., Aug. 2015, doi: 10.5281/zenodo.27878.

[25] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia (MM)*. New York, NY, USA: Association for Computing Machinery, Nov. 2014, pp. 675–678, doi: 10.1145/2647868.2654889.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25. Red Hook, NY, USA: Curran Associates, 2012, pp. 1097–1105.

[27] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst. Appl.*, vol. ISA-13, no. 4, pp. 18–28, Jul. 1998, doi: 10.1109/5254.708428.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[29] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.

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

[31] N. J. Nilsson, *Introduction to Machine Learning: An Early Draft of a Proposed Textbook*. 1996, pp. 175–188. [Online]. Available: http://robotics.stanford.edu/people/nilsson/mlbook.html

[32] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, 1997.

[33] E. Alpaydin, *Introduction to Machine Learning*, 2nd ed. Cambridge, MA, USA: MIT Press, 2010.

[34] J. D. Kelleher, B. M. Namee, and A. D'Arcy, *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. Cambridge, MA, USA: MIT Press, 2015.

[35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[36] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[37] R. D. Luce, *Individual Choice Behavior: A Theoretical Analysis*. Chelmsford, MA, USA: Courier Corporation, 2012.

[38] S. Kay, *Fundamentals of Statistical Signal Processing: Detection Theory* (Prentice Hall Signal Processing Series). Upper Saddle River, NJ, USA: Prentice-Hall, 1998. [Online]. Available: https://books.google.com/books?id=vA9LAQAAIAAJ

[39] E. Angel and V. Zissimopoulos, "On the classification of NP-complete problems in terms of their correlation coefficient," *Discrete Appl. Math.*, vol. 99, nos. 1–3, pp. 261–277, Feb. 2000.

[40] M. M. Saritas and A. Yasar, "Performance analysis of ANN and Naive Bayes classification algorithm for data classification," *Int. J. Intell. Syst. Appl. Eng.*, vol. 7, no. 2, pp. 88–91, Jan. 2019.

[41] J. Fan, C. Zhang, and J. Zhang, "Generalized likelihood ratio statistics and Wilks phenomenon," *Ann. Statist.*, vol. 29, no. 1, pp. 153–193, Feb. 2001.

[42] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[43] F.-F. Li and J. Johnson, "CS231n: Convolutional neural networks for visual recognition," Stanford Univ., Stanford, CA, USA, Tech. Rep., Apr. 2018. [Online]. Available: http://cs231n.github.io/

[44] J. Bruck and M. Blaum, "Neural networks, error-correcting codes, and polynomials over the binary n-cube," *IEEE Trans. Inf. Theory*, vol. 35, no. 5, pp. 976–987, Sep. 1989.

[45] I. Ortuno, M. Ortuno, and J. A. Delgado, "Error correcting neural networks for channels with Gaussian noise," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 4, Jun. 1992, pp. 295–300.

[46] V. Sagar, G. M. Jacyna, and H. Szu, "Block-parallel decoding of convolutional codes using neural network decoders," *Neurocomputing*, vol. 6, no. 4, pp. 455–471, Aug. 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0925231294900221

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

[48] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over the air deep learning based radio signal classification," *CoRR*, vol. abs/1712.04578, pp. 1–13, Dec. 2017.

[49] N. E. West and T. J. O'Shea, "Deep architectures for modulation recognition," *CoRR*, vol. abs/1703.09197, pp. 1–7, Mar. 2017.

[50] T. J. O'Shea and J. Corgan, "Convolutional radio modulation recognition networks," *CoRR*, vol. abs/1602.04105, pp. 1–15, Jun. 2016.
[51] B. Aazhang, B.-P. Paris, and G. C. Orsak, "Neural networks for multiuser detection in code-division multiple-access communications," *IEEE Trans. Commun.*, vol. 40, no. 7, pp. 1212–1222, Jul. 1992.
[52] G. I. Kechriotis and E. S. Manolakos, "Hopfield neural network implementation of the optimal CDMA multiuser detector," *IEEE Trans. Neural Netw.*, vol. 7, no. 1, pp. 131–141, Jan. 1996.
[53] G. Kechriotis and E. S. Manolakos, "A hybrid digital signal processing-neural network CDMA multiuser detection scheme," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 43, no. 2, pp. 96–104, Feb. 1996.
[54] U. Mitra and H. V. Poor, "Adaptive receiver algorithms for near-far resistant CDMA," *IEEE Trans. Commun.*, vol. 43, no. 2, pp. 1713–1724, Mar./Apr. 1995.
[55] C. Park, J. Choi, S. Nah, W. Jang, and D. Y. Kim, "Automatic modulation recognition of digital signals using wavelet features and SVM," in *Proc. 10th Int. Conf. Adv. Commun. Technol.*, vol. 1, 2008, pp. 387–390.
[56] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep learning models for wireless signal classification with distributed low-cost spectrum sensors," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 3, pp. 433–445, Sep. 2018.
[57] L. Danial, N. Wainstein, S. Kraus, and S. Kvatinsky, "DIDACTIC: A data-intelligent digital-to-analog converter with a trainable integrated circuit using memristors," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 1, pp. 146–158, Mar. 2018.
[58] A. Tankimanova and A. James, "Neural network-based analog-to-digital converters," in *Memristor and Memristive Neural Networks*. Apr. 2018.
[59] M. Vidyasagar, "Improved neural networks for analog-to-digital conversion," *Circuits, Syst., Signal Process.*, vol. 11, no. 3, pp. 387–398, Sep. 1992, doi: 10.1007/BF01190983.
[60] D. Tank and J. Hopfield, "Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits Syst.*, vol. CAS-33, no. 5, pp. 533–541, May 1986.
[61] A. Indriyatmoko, T. Kang, Y. J. Lee, G.-I. Jee, Y. B. Cho, and J. Kim, "Artificial neural networks for predicting DGPS carrier phase and pseudorange correction," *GPS Solutions*, vol. 12, no. 4, pp. 237–247, Sep. 2008.
[62] S. Chen, G. J. Gibson, C. F. N. Cowan, and P. M. Grant, "Adaptive equalization of finite non-linear channels using multilayer perceptrons," *Signal Process.*, vol. 20, no. 2, pp. 107–119, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/016516849090122F
[63] D. J. Sebald and J. A. Bucklew, "Support vector machine techniques for nonlinear equalization," *IEEE Trans. Signal Process.*, vol. 48, no. 11, pp. 3217–3226, Nov. 2000.
[64] G. Kechriotis, E. Zervas, and E. S. Manolakos, "Using recurrent neural networks for adaptive communication channel equalization," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 267–278, Mar. 1994.
[65] F. C. Hoppensteadt and E. M. Izhikevich, "Pattern recognition via synchronization in phase-locked loop neural networks," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 734–738, May 2000.
[66] M. Ibnkahla, N. J. Bershad, J. Sombrin, and F. Castanie, "Neural network modeling and identification of nonlinear channels with memory: Algorithms, applications, and analytic models," *IEEE Trans. Signal Process.*, vol. 46, no. 5, pp. 1208–1220, May 1998.
[67] W. Lee, M. Kim, and D.-H. Cho, "Deep learning based transmit power control in underlaid device-to-device communication," *IEEE Syst. J.*, vol. 13, no. 3, pp. 2551–2554, Sep. 2019.
[68] W. Lee, M. Kim, and D.-H. Cho, "Deep power control: Transmit power control scheme based on convolutional neural network," *IEEE Commun. Lett.*, vol. 22, no. 6, pp. 1276–1279, Jun. 2018.
[69] A. Zappone, M. Debbah, and Z. Altman, "Online energy-efficient power control in wireless networks by deep neural networks," in *Proc. IEEE 19th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jun. 2018, pp. 1–5.
[70] F. Liang, C. Shen, W. Yu, and F. Wu, "Towards optimal power control via ensembling deep neural networks," *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1760–1776, Mar. 2020.
[71] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for wireless resource management," 2017, *arXiv:1705.09412*.
[72] D. Kunz, "Channel assignment for cellular radio using neural networks," *IEEE Trans. Veh. Technol.*, vol. 40, no. 1, pp. 188–193, Feb. 1991.
[73] V. K. Tumuluru, P. Wang, and D. Niyato, "A neural network based spectrum prediction scheme for cognitive radio," in *Proc. IEEE Int. Conf. Commun.*, May 2010, pp. 1–5.
[74] R. Mahajan and D. Bagai, "Improved learning scheme for cognitive radio using artificial neural networks," *Int. J. Electr. Comput. Eng.*, vol. 6, no. 1, p. 257, Feb. 2016.
[75] K. Thilina, K. W. Choi, N. Saquib, and E. Hossain, "Machine learning techniques for cooperative spectrum sensing in cognitive radio networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 11, pp. 2209–2221, Nov. 2013.
[76] Y.-J. Tang, Q.-Y. Zhang, and W. Lin, "Artificial neural network based spectrum sensing method for cognitive radio," in *Proc. 6th Int. Conf. Comput. Intell. Softw. Eng. (WiCOM)*, Sep. 2010, pp. 1–4.
[77] A. Klautau, P. Batista, N. González-Prelcic, Y. Wang, and R. W. Heath, Jr., "5G MIMO data for machine learning: Application to beam-selection using deep learning," in *Proc. Inf. Theory Appl. Workshop (ITA)*, Feb. 2018, pp. 1–9.
[78] X. Gao, L. Dai, Y. Sun, S. Han, and I. Chih-Lin, "Machine learning inspired energy-efficient hybrid precoding for mmWave massive MIMO systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.
[79] A. Alkhateeb, "DeepMIMO: A generic deep learning dataset for millimeter wave and massive MIMO applications," *CoRR*, vol. abs/1902.06435, pp. 1–8, Feb. 2019.
[80] D. He, C. Liu, T. Q. S. Quek, and H. Wang, "Transmit antenna selection in MIMO wiretap channels: A machine learning approach," *IEEE Wireless Commun. Lett.*, vol. 7, no. 4, pp. 634–637, Aug. 2018.
[81] N. Samuel, T. Diskin, and A. Wiesel, "Deep MIMO detection," 2017, *arXiv:1706.01151*.
[82] D. Crevier, *AI: The Tumultuous History of the Search for Artificial Intelligence*. New York, NY, USA: Basic Books, 1993.
[83] N. Benvenuto, M. Marchesi, F. Piazza, and A. Uncini, "Non linear satellite radio links equalized using blind neural networks," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 3, 1991, pp. 1521–1524.
[84] S. Benedetto, E. Biglieri, and R. Daffara, "Modeling and performance evaluation of nonlinear satellite links—A Volterra series approach," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-15, no. 4, pp. 494–507, Jul. 1979.
[85] S. A. Billings, S. Chen, and M. J. Korenberg, "Identification of MIMO non-linear systems using a forward-regression orthogonal estimator," London, U.K., Tech. Rep., 1989. [Online]. Available: https://eprints.soton.ac.U.K./251146/
[86] W.-W. Fang, B. J. Sheu, O. T.-C. Chen, and J. Choi, "A VLSI neural processor for image data compression using self-organization networks," *IEEE Trans. Neural Netw.*, vol. 3, no. 3, pp. 506–518, May 1992.
[87] K. Iba, "Reactive power optimization by genetic algorithm," *IEEE Trans. Power Syst.*, vol. 9, no. 2, pp. 685–692, May 1994.
[88] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, no. 8, pp. 2554–2558, 1982.
[89] H. L. Southall, J. A. Simmers, and T. H. O'Donnell, "Direction finding in phased arrays with a neural network beamformer," *IEEE Trans. Antennas Propag.*, vol. 43, no. 12, pp. 1369–1374, Dec. 1995.
[90] A. K. Nandi and E. E. Azzouz, "Algorithms for automatic modulation recognition of communication signals," *IEEE Trans. Commun.*, vol. 46, no. 4, pp. 431–436, Apr. 1998.
[91] R. C. Daniels, C. M. Caramanis, and R. W. Heath, Jr., "Adaptation in convolutionally coded MIMO-OFDM wireless systems through supervised learning and SNR ordering," *IEEE Trans. Veh. Technol.*, vol. 59, no. 1, pp. 114–126, Jan. 2010.
[92] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Comput. Netw.*, vol. 50, no. 13, pp. 2127–2159, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128606001009
[93] I. Angus, "An introduction to Erlang B and Erlang C," *Telemanagement*, vol. 187, pp. 6–8, Jul. 2001.
[94] P. Tran-Gia, D. Staehle, and K. Leibnitz, "Source traffic modeling of wireless applications," *AEU, Int. J. Electron. Commun.*, vol. 55, no. 1, pp. 27–36, Jan. 2001.
[95] I. Demirkol, F. Alagoz, H. Delic, and C. Ersoy, "Wireless sensor networks for intrusion detection: Packet traffic modeling," *IEEE Commun. Lett.*, vol. 10, no. 1, pp. 22–24, Jan. 2006.
[96] Z. Xie, R. T. Short, and C. K. Rushforth, "A family of suboptimum detectors for coherent multiuser communications," *IEEE J. Sel. Areas Commun.*, vol. 8, no. 4, pp. 683–690, May 1990.

[97] C. U. Saraydar, N. B. Mandayam, and D. J. Goodman, "Efficient power control via pricing in wireless data networks," *IEEE Trans. Commun.*, vol. 50, no. 2, pp. 291–303, Feb. 2002.

[98] C. Candan, "A method for fine resolution frequency estimation from three DFT samples," *IEEE Signal Process. Lett.*, vol. 18, no. 6, pp. 351–354, Jun. 2011.

[99] D. R. Stephens, *Phase-Locked Loops for Wireless Communications: Digital, Analog and Optical Implementations*. Springer, 2007.

[100] D. Falconer, S. L. Ariyavisitakul, A. Benyamin-Seeyar, and B. Eidson, "Frequency domain equalization for single-carrier broadband wireless systems," *IEEE Commun. Mag.*, vol. 40, no. 4, pp. 58–66, Apr. 2002.

[101] D. R. Morgan, Z. Ma, J. Kim, M. G. Zierdt, and J. Pastalan, "A generalized memory polynomial model for digital predistortion of RF power amplifiers," *IEEE Trans. Signal Process.*, vol. 54, no. 10, pp. 3852–3860, Oct. 2006.

[102] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, "Massive MIMO for next generation wireless systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 186–195, Feb. 2014.

[103] A. Goldsmith, S. A. Jafar, N. Jindal, and S. Vishwanath, "Capacity limits of MIMO channels," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 5, pp. 684–702, Jun. 2003.

[104] C. Lee, H. B. Yilmaz, C.-B. Chae, N. Farsad, and A. Goldsmith, "Machine learning based channel modeling for molecular MIMO communications," in *Proc. IEEE 18th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jul. 2017, pp. 1–5.

[105] M. J. Demler, *High-Speed Analog-to-Digital Conversion*. Amsterdam, The Netherlands: Elsevier, 2012.

[106] E. Biglieri, "Coding and modulation for a horrible channel," *IEEE Commun. Mag.*, vol. 41, no. 5, pp. 92–98, May 2003.

[107] S. Lin and D. J. Costello, *Error Control Coding*, vol. 2, no. 4. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

[108] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 2, May 1993, pp. 1064–1070.

[109] R. M. Tanner, D. Sridhara, and T. Fuja, "A class of group-structured LDPC codes," in *Proc. ISTA*, 2001, pp. 365–370.

[110] L. Ping, L. Liu, K. Wu, and W. K. Leung, "Interleave division multiple-access," *IEEE Trans. Wireless Commun.*, vol. 5, no. 4, pp. 938–947, Apr. 2006.

[111] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[112] O. Koyejo, N. Natarajan, P. Ravikumar, and I. S. Dhillon, "Consistent binary classification with generalized performance metrics," in *Proc. NIPS*, vol. 27. Princeton, NJ, USA: Citeseer, 2014, pp. 2744–2752.

[113] A. Patel and B. Kosko, "Optimal noise benefits in Neyman–Pearson and inequality-constrained statistical signal detection," *IEEE Trans. Signal Process.*, vol. 57, no. 5, pp. 1655–1669, May 2009.

[114] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. R. Mullers, "Fisher discriminant analysis with kernels," in *Proc. Neural Netw. Signal Process. IX, IEEE Signal Process. Soc. Workshop*, Aug. 1999, pp. 41–48.

[115] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *Proc. Int. Conf. Database Theory*. Springer, 2001, pp. 420–434.

[116] K. Pearson, "On lines and planes of closest fit to systems of points in space," *London, Edinburgh, Dublin Phil. Mag. J. Sci.*, vol. 2, no. 11, pp. 559–572, 1901, doi: 10.1080/14786440109462720.

[117] I. Lyubomirsky, "Machine learning equalization techniques for high speed PAM4 fiber optic communication systems," Stanford Univ., Stanford, CA, USA, Final Project Rep. CS229, 2015.

[118] J. Liu, K. Mei, X. Zhang, D. Ma, and J. Wei, "Online extreme learning machine-based channel estimation and equalization for OFDM systems," *IEEE Commun. Lett.*, vol. 23, no. 7, pp. 1276–1279, Jul. 2019.

[119] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using SUMO," in *Proc. 21st IEEE Int. Conf. Intell. Transp. Syst.*, Nov. 2018, pp. 2575–2582. [Online]. Available: https://elib.dlr.de/124092/

[120] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.

[121] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, Dec. 2002.

[122] X. Guo, F. Merrikh-Bayat, L. Gao, B. D. Hoskins, F. Alibart, B. Linares-Barranco, L. Theogarajan, C. Teuscher, and D. B. Strukov, "Modeling and experimental demonstration of a Hopfield network analog-to-digital converter with hybrid CMOS/memristor circuits," *Frontiers Neurosci.*, vol. 9, p. 488, Dec. 2015.

[123] T. J. O'Shea, T. Erpek, and T. C. Clancy, "Deep learning based MIMO communications," *CoRR*, vol. abs/1707.07980, pp. 1–9, Jul. 2017.

[124] T. J. O'Shea, K. Karra, and T. C. Clancy, "Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention," *CoRR*, vol. abs/1608.06409, pp. 1–10, Aug. 2016.

[125] 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, Dec. 2017.

[126] T. J. Oshea, T. Roy, N. West, and B. C. Hilburn, "Physical layer communications system design over-the-air using adversarial networks," in *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, Sep. 2018, pp. 529–532.

[127] T. J. O'Shea, T. Roy, and N. West, "Approximating the void: Learning stochastic channel models from observation with variational generative adversarial networks," *CoRR*, vol. abs/1805.06350, pp. 1–6, Aug. 2018.

[128] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 168–179, Feb. 2018.

[129] T. J. O'Shea, L. Pemula, D. Batra, and T. C. Clancy, "Radio transformer networks: Attention models for learning to synchronize in wireless systems," *CoRR*, vol. abs/1605.00716, pp. 1–5, May 2016.

[130] T. J. O'Shea, N. West, M. Vondal, and T. C. Clancy, "Semi-supervised radio signal identification," *CoRR*, vol. abs/1611.00303, pp. 1–5, Jan. 2016.

[131] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *Proc. Int. Conf. Eng. Appl. Neural Netw.* Springer, 2016, pp. 213–226.

[132] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," *CoRR*, vol. abs/1212.0901, pp. 1–5, Dec. 2012.

[133] T. Tieleman and G. Hinton. *RMSProp Gradient Optimization*. [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[134] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," Dept. EECS, Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2010-24, Mar. 2010. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-24.html

[135] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, pp. 1–15, Dec. 2014.

[136] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep double descent: Where bigger models and more data hurt," *CoRR*, vol. abs/1912.02292, pp. 1–24, Dec. 2019.

[137] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Netw.*, vol. 12, no. 1, pp. 145–151, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608098001166

[138] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25. Red Hook, NY, USA: Curran Associates, 2012, pp. 1223–1231. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/6aca97005c68f1206823815f66102863-Paper.pdf

[139] B. McMahan and M. Streeter, "Delay-tolerant algorithms for asynchronous distributed online learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27. Red Hook, NY, USA: Curran Associates, 2014, pp. 2915–2923. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/5cce8dede893813f879b873962fb669f-Paper.pdf

[140] S. Zhang, A. Choromanska, and Y. LeCun, "Deep learning with elastic averaging SGD," 2014, *arXiv:1412.6651*.

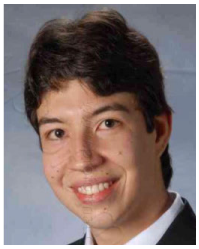
[141] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, 1951.

[142] J. Lucas, R. S. Zemel, and R. B. Grosse, "Aggregated momentum: Stability through passive damping," *CoRR*, vol. abs/1804.00325, pp. 1–22, Apr. 2018.

[143] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012.
[144] T. Dozat, "Incorporating Nesterov momentum into Adam," in *Proc. 4th Int. Conf. Learn. Represent.*, 2016, pp. 1–4.
[145] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," *CoRR*, vol. abs/1904.09237, pp. 1–23, Apr. 2019.
[146] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in Adam," *CoRR*, vol. abs/1711.05101, 2017.
[147] J. Ma and D. Yarats, "Quasi-hyperbolic momentum and Adam for deep learning," *CoRR*, vol. abs/1810.06801, pp. 1–38, Oct. 2018.
[148] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens, "Adding gradient noise improves learning for very deep networks," 2015, *arXiv:1511.06807*.
[149] D. Mishkin and J. Matas, "All you need is a good init," 2015, *arXiv:1511.06422*.
[150] S. K. Kumar, "On weight initialization in deep neural networks," *CoRR*, vol. abs/1704.08863, pp. 1–9, May 2017.
[151] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," *CoRR*, vol. abs/1502.01852, pp. 1–11, Feb. 2015.
[152] Y. Li and F. Liu, "Whiteout: Gaussian adaptive noise regularization in deep neural networks," 2016, *arXiv:1612.01490*.
[153] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html
[154] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, pp. 1–11, Feb. 2015.
[155] M. Kulin, T. Kazaz, I. Moerman, and E. D. Poorter, "End-to-end learning from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications," *CoRR*, vol. abs/1712.03987, 2017.
[156] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27. Red Hook, NY, USA: Curran Associates, 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf
[157] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Found. Trends Mach. Learn.*, vol. 12, no. 4, pp. 307–392, 2019, doi: 10.1561/2200000056.
[158] S. Chib and E. Greenberg, "Understanding the metropolis-hastings algorithm," *Amer. Stat.*, vol. 49, no. 4, pp. 327–335, 1995.
[159] C. K. Carter and R. Kohn, "On Gibbs sampling for state space models," *Biometrika*, vol. 81, no. 3, pp. 541–553, 1994.
[160] R. M. Neal, "Annealed importance sampling," *Statist. Comput.*, vol. 11, no. 2, pp. 125–139, 2001.
[161] W. R. Gilks and P. Wild, "Adaptive rejection sampling for Gibbs sampling," *J. Roy. Stat. Soc. C, Appl. Statist.*, vol. 41, no. 2, pp. 337–348, Jun. 1992.
[162] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.
[163] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. 14th Int. Joint Conf. Artif. Intell. (IJCAI)*, vol. 2. San Francisco, CA, USA: Morgan Kaufmann, 1995, pp. 1137–1143.
[164] M. Stonebraker, U. Çetintemel, and S. Zdonik, "The 8 requirements of real-time stream processing," *ACM SIGMOD Rec.*, vol. 34, no. 4, pp. 42–47, 2005.

KYLE W. MCCLINTICK (Student Member, IEEE) received the B.S. degree in electrical and computer engineering from the Rose-Hulman Institute of Technology, in 2017, and the M.S. degree in electrical and computer engineering from Worcester Polytechnic Institute, in 2019, with a thesis on SL in wireless communications, generously funded by The MITRE Corporation, where he is currently pursuing the Ph.D. degree in data science and computer science (minor) with the Department of Electrical and Computer Engineering, generously funded by the MIT Lincoln Laboratory. He was a recipient of the Best Paper Award for his GlobalSIP 2019 paper on wireless localization using UL techniques.

GALAHAD M. WERNSING (Student Member, IEEE) received the B.S. degree in electrical and computer engineering and the M.S. degree in ECE, with a thesis studying Bluetooth communications through the generous support of octoScope Inc., from Worcester Polytechnic Institute (WPI), in 2018, where he is currently pursuing the Ph.D. degree with the ECE Department, with minors in both computer science and nuclear science.



PAULO VICTOR R. FERREIRA (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from the Universidade Federal de Uberlandia, Brazil, in 2010 and 2012, respectively, and the Ph.D. degree in electrical and computer engineering from Worcester Polytechnic Institute (WPI), Worcester, MA, USA, in 2017. He has worked as a Graduate Research Assistant at the Wireless Innovation Laboratory, within the Department of Electrical and Computer Engineering, WPI, and as a Research and Development Intern at General Electric Global Research. He is a RF Systems Engineer at Cohu, Inc. His research interests include satellite communications, machine learning, wireless communications, cognitive radio, and industrial robotics.

ALEXANDER M. WYGLINSKI (Senior Member, IEEE) received the B.Eng. and Ph.D. degrees in electrical engineering from McGill University, Montreal, Canada, in 1999 and 2005, respectively, and the M.Sc. (Eng.) degree in electrical engineering from Queen's University, Kingston, Canada, in 2000. He is the Associate Dean of graduate studies and a Professor of electrical and computer engineering at Worcester Polytechnic Institute (WPI), Worcester, MA, USA, where he has been the Director of the Wireless Innovation Laboratory, since 2007. His current research interests include wireless communications, cognitive radio, machine learning for wireless systems, software-defined radio prototyping, connected and autonomous vehicles, and dynamic spectrum sensing, characterization, and access. He was the President of the IEEE Vehicular Technology Society, from 2018 to 2019.