NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

ADVERSARIAL MACHINE LEARNING FOR PHYSICAL-LAYER AUTHENTICATION

by

Kenneth W. St. Germain

June 2021

Dissertation Supervisor: Frank E. Kragh

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2021	**3. REPORT TYPE AND DATES COVERED** Dissertation

4. TITLE AND SUBTITLE
ADVERSARIAL MACHINE LEARNING FOR PHYSICAL-LAYER AUTHENTICATION

5. FUNDING NUMBERS

6. AUTHOR(S) Kenneth W. St. Germain

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

In this dissertation, we propose the use of adversarial machine learning to characterize wireless radio transmitters for the purpose of physical-layer authentication. Wireless communication systems are quickly evolving to take advantage of autonomous networking for applications such as 5th generation mobile networks, Internet of Things, and vehicular-to-everything technologies. Robust and efficient network security mechanisms are necessary to protect the authenticity of the data and safeguard the integrity of the greater interconnected network. To this end, we leverage unique channel-dependent differences in received transmissions, known as channel state information (CSI), to make authentication decisions with machine learning algorithms. Many physical-layer authentication techniques are not effective when used in the presence of nefarious users who are able to spoof the underlying physical-layer authentication traits. Our approach uses adversarial learning to counter malicious actions such as spoofing against legitimate transmitter CSI, an already difficult characteristic to emulate. We simulated various radio frequency channel environments and our results indicate that the use of machine learning techniques can produce high authentication accuracy.

14. SUBJECT TERMS
physical-layer authentication, PHY authentication, neural network, generative adversarial network, GAN, cyber, 5G, multiple input multiple output, MIMO, multipath, deep learning

15. NUMBER OF PAGES
177

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

ADVERSARIAL MACHINE LEARNING FOR PHYSICAL-LAYER AUTHENTICATION

Kenneth W. St. Germain
Commander, United States Navy
BSEE, University of Nebraska at Lincoln, 2000
MS, Electrical Engineering, Naval Postgraduate School, 2005
MS, Space Systems Operations, Naval Postgraduate School, 2013

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2021

Approved by:

Frank E. Kragh
Department of
Electrical and Computer Engineering
Dissertation Supervisor and Chair

Chad A. Bollmann
Department of
Electrical and Computer
Engineering

James B. Michael
Department of
Computer Science

Ric Romero
Department of
Electrical and Computer
Engineering

Preetha Thulasiraman
Department of
Electrical and Computer Engineering

Approved by:

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

Orrin D. Moses
Vice Provost of Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In this dissertation, we propose the use of adversarial machine learning to characterize wireless radio transmitters for the purpose of physical-layer authentication. Wireless communication systems are quickly evolving to take advantage of autonomous networking for applications such as 5th generation mobile networks, Internet of Things, and vehicular-to-everything technologies. Robust and efficient network security mechanisms are necessary to protect the authenticity of the data and safeguard the integrity of the greater interconnected network. To this end, we leverage unique channel-dependent differences in received transmissions, known as channel state information (CSI), to make authentication decisions with machine learning algorithms. Many physical-layer authentication techniques are not effective when used in the presence of nefarious users who are able to spoof the underlying physical-layer authentication traits. Our approach uses adversarial learning to counter malicious actions such as spoofing against legitimate transmitter CSI, an already difficult characteristic to emulate. We simulated various radio frequency channel environments and our results indicate that the use of machine learning techniques can produce high authentication accuracy.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

3GPP	3rd Generation Partnership Project
5G	5th generation
Adam	adaptive moment estimation
ANN	artificial neural network
AWGN	additive white Gaussian noise
BCE	binary cross-entropy
C4	command, control, communications, and computer
CFR	channel frequency response
CGAN	conditional generative adversarial network
CNN	convolutional neural network
CSI	channel state information
DNN	deep neural network
DoD	Department of Defense
EVA	extended vehicular A
GAN	generative adversarial network
GRU	gated recurrent unit
I	in-phase
IEEE	Institute of Electrical and Electronic Engineers
iForest	Isolation forest

IoT Internet of Things

kNN k-nearest neighbor

LeakyReLU leaky rectified linear unit

LOF Local outlier factor

LOS line of sight

LSGAN least-squares generative adversarial network

LSTM long short-term memory

LTE Long-Term Evolution

MAC media access control

MIMO multiple-input multiple-output

MLP multiple-layer perceptron

MNIST Modified National Institute of Standards and Technology

MSE mean-squared error

NIST National Institute of Standards and Technology

NLOS non line-of-sight

NPS Naval Postgraduate School

OC-SVM one-class support vector machine

OFDM orthogonal frequency division multiplexing

OSI Open Systems Interconnection

Q quadrature

QAM quadrature amplitude modulation

ReLU rectified linear unit

RF	radio frequency
RNN	recurrent neural network
RSS	received signal strength
SGAN	semi-supervised generative adversarial network
SNR	signal-to-noise ratio
SVM	support vector machine
tanh	hyperbolic tangent
USC	United States Code
USRP	universal software radio peripheral
V2X	vehicular-to-everything
Wi-Fi	wireless fidelity
WSN	wireless sensor network

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Information security [1] — the protection of integrity, confidentiality, and availability — is a challenge in wireless networks. Unlike networks with wired point-to-point connections, the broadcast nature of radio frequency (RF) grants bona fide users and malicious actors the same access to the communication channel. Therefore, other means must be employed to maintain security. Unfortunately, these means and controls still have vulnerabilities that a motivated adversary can exploit, and in some instances provide very little security [2] at the expense of increased complication or worse quality of service [3].

In this work, we investigate physical-layer authentication and leverage deep learning to ensure strong authentication on a variety of devices with disparate applications. Knowing the identity of the person or device one is communicating with is a vital part of secure communications. Without a robust method to ensure authenticity, a fake message can be sent or a valid transmission can be altered.

As the 3rd Generation Partnership Project (3GPP) and 5th generation (5G) mobile network bring the promise of high mobile data rates, these communications must be secure [4]. These networks will connect many different types of devices, dynamically adapt, and support minimum latency [5]. However, without appropriate security, there will be intrusions and attacks, countering the networks' benefits. There is no lack of reporting on computer network attacks, and continued justification in improving security against a myriad of threats at the federal, state, and private level [6]–[9].

As wireless networks evolve to 5G mobile networks to take advantage of technologies such as Internet of Things (IoT), vehicular-to-everything (V2X), and video streaming [10], robust and efficient network security mechanisms are necessary. Physical-layer authentication can be used to mitigate the challenges of cryptographic authentication methods as shown in [11]–[14]. The efficient performance [15] of physical-layer authentication can be used to augment or replace cryptographic-based protocols.

The application of machine learning techniques is well suited for complex problems and

dynamic environments with consistently fluctuating data [16]. Several different algorithms and models have been developed to improve the accuracy and the efficiency of machine learning results based on the type of problem being addressed, the quantity and nature of available training data, and the measure of performance required. One important breakthrough in machine learning was to create a model based on the biological neuron. Just as airplanes were inspired by birds but only vaguely resemble them now, modern artificial neural networks inspired by the architecture of the brain make use of only a few characteristics of their biological counterpart. However, this inspiration has recently been the engine of a wide variety of technical achievements. In particular, a discriminator artificial neural network, also known as a binary classifier, can be used to tell real samples from fakes following a dedicated training period. There are many factors that ultimately determine the unique properties of the received signal such as channel characteristics and variations in the transmitter hardware itself [17]. During training, the discriminator will learn which features from the channel state information are most relevant to the authentic channel.

With respect to notation, and unless otherwise addressed in this dissertation, we use j to represent $\sqrt{-1}$, vectors are indicated with bold lower-case letters, and matrices with bold upper-case letters.

1.1 Objective

The objective of this dissertation is to advance the authentication aspect of cyber security by restricting authentication to transmitters based on the characteristics of received wireless transmissions. Specifically, we investigate authentication by using received channel state information (CSI) in a multiple-input multiple-output (MIMO) channel and the employment of machine learning techniques, including the use of deep learning and the generative adversarial network (GAN). We investigate how GANs and CSI can be used to provide physical-layer authentication in MIMO-enabled wireless networks.

1.2 Authentication

Verifying the identity of a user, process, or device is authentication [18]. When sharing information, particularly of a sensitive nature, integrity as defined in 44 USC §3552 [1] is of paramount importance and necessitates authenticity. As modern society becomes

more reliant on improvements and conveniences offered by digital information systems, attackers can exploit weaknesses in those systems, requiring additional emphasis placed on cybersecurity.

Figure 1.1 depicts two entities, Alice and Bob, authenticating with each other. That is, Alice is communicating to Bob and providing some sort of information to prove to Bob that Alice is indeed Alice and not someone else. Bob, in turn, will also prove himself to Alice. A third entity, Eve, can also communicate with Alice and Bob and eavesdrop on their conversation. Strong authentication ensures that Eve cannot pose as either Alice or Bob.

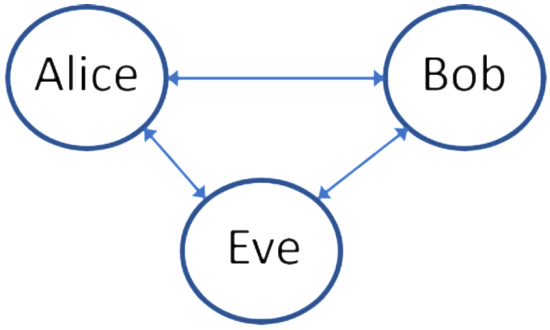


Figure 1.1. Alice and Bob mutually authenticating in the presence of Eve.

Authentication in military and other Department of Defense (DoD) environments is key to ensuring only trusted entities receive and transmit execution orders, intelligence and planning information, and official correspondence. Trust relationships established between information systems may allow further access to other information systems. Among several additional requirements, authentication needs to be scrutinized and maintained to sustain cyberspace defense against unauthorized activity [19]. The requirement for mobile and expeditionary operations is supported by technologies that make use of the RF spectrum. Wireless communication greatly enhances command, control, communications, and computer (C4) systems by allowing for rapid deployment and complex operations over large distances [20].

Authentication is widely and successfully accomplished using key-based cryptography, but as the network grows and becomes more complex, key distribution and management may not scale without causing undue user delays [12], [21]. Each device must have the ability

to recognize unique passwords or keys from every device it's connected to in a network. In a dynamic and interconnected wireless network, maintaining the integrity of passwords or keys presents several challenges, to include: the distribution and management of keys [11], intolerable latencies induced by key generation/detection [12], and the vulnerability of the key or password to compromise [13], [14]. Transmitting an encryption key does not necessarily mean an eavesdropper can determine the decryption key [22]; however, there is an infrastructure and processing power cost for this feature. In addition to maintaining security, seamless handoffs are a necessity in multi-domain networks to support low-latency tolerant applications [10], [23], [24].

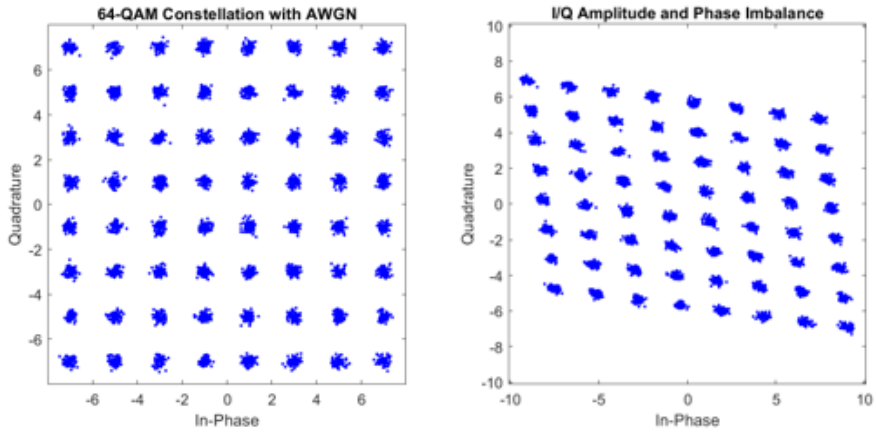
1.3 Physical-Layer Characteristics

The literature proposes different methods to distinguish legitimate from illegitimate devices at the physical layer [12], [25] without the use of a pre-shared secret, cryptography, user-provided credentials, or higher Open Systems Interconnection (OSI)-level processing and we will examine two of the most prevalent techniques. The first relies on unique imperfections of the transmitter hardware that manifest as RF fingerprints or signatures [17], [26], [27]. Based on manufacturing processes and designs, the transmitted signal will be uniquely distorted from device to device, even if only slightly. The second method leverages the wireless channel to take advantage of multi-path fading environments. The temporally and spatially-unique impulse or frequency response can be used to identify the transmitter [28]–[30].

1.3.1 Transmitter Imperfections

Transmitters of the same brand and model often reveal different characteristics in the RF environment, and because these characteristics are dependent on the transmitter and not the channel, these attributes do not change with time [26]. Imperfections may reside in any number of components that comprise a wireless transmitter system and must not cause the system to exceed the bounds of the respective protocol specification. Slight changes in production process and clean-room variations can manifest as signatures during transmitter operation [27]. These signatures, such as transients associated with the start-up period before transmission, can be measured and subsequently used to authenticate trusted devices [31]. Another practically unavoidable signature, in-phase (I) quadrature (Q) imbalance, can be

compensated for with known techniques [32], but can still be detected and used to make an authentication decision. Figure 1.2 shows the result to the constellation of an amplitude and phase imbalance on a 64-QAM system in additive white Gaussian noise (AWGN). Unfortunately, the non-temporal quality of hardware-based transmitter characteristics leads to a vulnerability where malicious adversaries can intentionally distort their transmitter characteristics to mimic other transmitters or to keep their own signatures hidden [33].



(a) (b)

Figure 1.2. Amplitude and IQ imbalance for 64-QAM (a) before and (b) after 3 dB amplitude and 15 degree phase imbalance.

1.3.2 Channel Attributes

Consider a receiver with N antennas in a multipath channel environment. At each antenna, the wideband received signal as a function of time $y_w(t)$ is given by $y_w(t) = r(t)\cos(\omega t + \theta(t))$, where r is the amplitude, ω is the carrier frequency in radians per second, and θ is the received phase of the signal. The received signal is represented as a vector $\boldsymbol{y}_w(t)$ by

$$\boldsymbol{y}_w(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_N(t) \end{bmatrix} = \begin{bmatrix} r_1(t)\cos(\omega t + \theta_1(t)) \\ r_2(t)\cos(\omega t + \theta_2(t)) \\ \vdots \\ r_N(t)\cos(\omega t + \theta_N(t)) \end{bmatrix}. \tag{1.1}$$

We can alter Equation 1.1 to describe the received signal in the complex plane through the use of trigonometric identities, resulting in

$$\boldsymbol{y}_w(t) = \begin{bmatrix} r_1(t)\cos(\omega t + \theta_1(t)) \\ r_2(t)\cos(\omega t + \theta_2(t)) \\ \vdots \\ r_N(t)\cos(\omega t + \theta_N(t)) \end{bmatrix} = \begin{bmatrix} r_1(t)\cos(\omega t)\cos(\theta_1(t)) - r_1(t)\sin(\omega t)\sin(\theta_1(t)) \\ r_2(t)\cos(\omega t)\cos(\theta_2(t)) - r_2(t)\sin(\omega t)\sin(\theta_2(t)) \\ \vdots \\ r_N(t)\cos(\omega t)\cos(\theta_N(t)) - r_N(t)\sin(\omega t)\sin(\theta_N(t)) \end{bmatrix}. \tag{1.2}$$

Breaking $\boldsymbol{y}_w(t)$ into its respective in-phase and quadrature components by letting $y_n^I(t) = r_n(t)\cos(\theta_n(t))$ and $y_n^Q(t) = r_n(t)\sin(\theta_n(t))$, where $1 \leq n \leq N$, we have

$$\boldsymbol{y}_w(t) = \begin{bmatrix} y_1^I(t)\cos(\omega t) - y_1^Q(t)\sin(\omega t) \\ y_2^I(t)\cos(\omega t) - y_2^Q(t)\sin(\omega t) \\ \vdots \\ y_N^I(t)\cos(\omega t) - y_N^Q(t)\sin(\omega t) \end{bmatrix} \tag{1.3}$$

or alternatively in the baseband,

$$\boldsymbol{y}(t) = \begin{bmatrix} y_1^I(t) + jy_1^Q(t) \\ y_2^I(t) + jy_2^Q(t) \\ \vdots \\ y_N^I(t) + jy_N^Q(t) \end{bmatrix} = \begin{bmatrix} r_1(t)\cos(\theta_1(t)) + jr_1(t)\sin(\theta_1(t)) \\ r_2(t)\cos(\theta_2(t)) + jr_2(t)\sin(\theta_2(t)) \\ \vdots \\ r_N(t)\cos(\theta_N(t)) + jr_N(t)\sin(\theta_N(t)) \end{bmatrix}. \tag{1.4}$$

If we assume a channel with bandwidth narrow enough to produce a flat channel [34], we further simplify Equation 1.4 to

$$\boldsymbol{y} = \begin{bmatrix} r_1\cos(\theta_1) + jr_1\sin(\theta_1) \\ r_2\cos(\theta_2) + jr_2\sin(\theta_2) \\ \vdots \\ r_N\cos(\theta_N) + jr_N\sin(\theta_N) \end{bmatrix}. \tag{1.5}$$

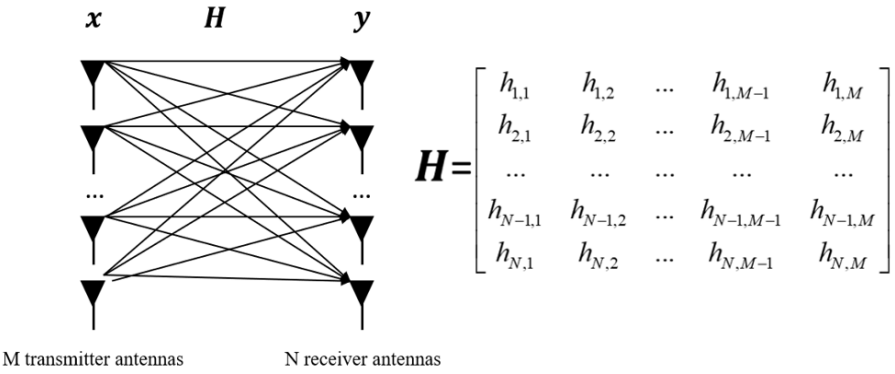
Embedded in the received signal described by Equation 1.5 is the distorted transmitted signal and AWGN. The nature of the wireless medium affects the transmitted signal as it

propagates to the receiver. The channel response, $\boldsymbol{H}$, is also referred to as CSI [29] and in the frequency domain, channel frequency response (CFR) [35]. The result of wireless signal deterioration due to factors including fade, shadowing, scattering, and path loss can be characterized by CSI.

The narrowband model of the wireless channel [36] is given by

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n} \tag{1.6}$$

where $\boldsymbol{y}$ is the received signal, $\boldsymbol{x}$ is the transmitted signal, $\boldsymbol{H}$ is the time-varying channel response, and $\boldsymbol{n}$ is complex AWGN. $\boldsymbol{H}$ is a complex $M \times N$ matrix that represents changes in signal amplitude and phase caused by multiple channel conditions such as multi-path fading and the use of multiple antennas [34]. The number of transmitter antennas is M and the number of receiver antennas is N. Each complex element within $\boldsymbol{H}$, $h_{n,m}$ ($1 \leq n \leq N$, $1 \leq m \leq M$), is composed of real and imaginary zero-mean independent Gaussian random variables with identical variance and with Rayleigh distributed magnitude for non line-of-sight (NLOS) scenarios. Jakes' uniform scattering model [37] states that antennas spatially separated more than two carrier wavelengths from each other will observe sufficiently independent fading channels due to rapid decorrelation of the signal envelope among receivers. The complex coefficient values in $\boldsymbol{H}$, $h_{n,m}$, are random and independent of $\boldsymbol{x}$ and $\boldsymbol{n}$ [36]. Figure 1.3 illustrates the MIMO narrowband channel model.



$$\boldsymbol{H} = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,M-1} & h_{1,M} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,M-1} & h_{2,M} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ h_{N-1,1} & h_{N-1,2} & \cdots & h_{N-1,M-1} & h_{N-1,M} \\ h_{N,1} & h_{N,2} & \cdots & h_{N,M-1} & h_{N,M} \end{bmatrix}$$

Figure 1.3. MIMO narrowband channel model and CSI matrix. Adapted from [38].

In a multipath environment, two identical transmitters in different locations will have unique channel responses $\boldsymbol{H}_0$ and $\boldsymbol{H}_1$. Based on the channel response from one transmitter $\boldsymbol{H}_0$, a receiver can discriminate from another transmitter since that channel response $\boldsymbol{H}_1$, will differ from $\boldsymbol{H}_0$. For example, in [39], Liu et al. developed a test statistic and threshold to differentiate the identities of transmitters for authentication. Liu used an adaptive threshold based on signal-to-noise ratio (SNR) to maintain a consistent false alarm rate.

In dynamic or mobile environments, channel conditions on the received signal are represented as an $N \times M$ matrix of channel responses $\boldsymbol{H}(t;\tau)$ at time t, to an impulse transmitted at time $t-\tau$. Following Biglieri and Taricco in [40] and Pedersen et al. in [41], the wideband channel model describing $\boldsymbol{H}(t;\tau)$ is a tapped delay line as shown in

$$\boldsymbol{H}(t;\tau) = \sum_{l=1}^{L} \boldsymbol{H}_l(t)\delta(\tau - \tau_l), \tag{1.7}$$

where each $\boldsymbol{H}_l$ is an $N \times M$ matrix of circularly symmetric complex-valued Gaussian random variables, τ_l is the path delay and L is the total number of path delays. The received signal vector $\boldsymbol{y}(t)$ is related to the transmitted signal vector $\boldsymbol{x}(t)$ by

$$\boldsymbol{y}(t) = \int_{-\infty}^{\infty} \boldsymbol{H}(t;\tau)\boldsymbol{x}(t-\tau)\mathrm{d}\tau + \boldsymbol{n}(t). \tag{1.8}$$

As before in the narrowband flat-frequency response case, the time-varying channel response between two spatially diverse transmitters in a mobile environment $\boldsymbol{H}_0(t;\tau)$ and $\boldsymbol{H}_1(t;\tau)$ will be different at each sample time t. Additionally, the channel responses from one transmitter will change at each sample time. That is, the elements within $\boldsymbol{H}_0(t_1;\tau)$ will be uncorrelated to the elements $\boldsymbol{H}_0(t_2;\tau)$ due to changes in phase. However, the magnitudes within the CSI elements from neighboring samples $|h_{n,m}(t_1;\tau)|$, $|h_{n,m}(t_2;\tau)|$, ..., $|h_{n,m}(t_s;\tau)|$ will exhibit correlation as long as the difference between sample times is small, as shown in [42]. Because correlation exists for the magnitude of time-sampled elements within $\boldsymbol{H}_0$, $|h_{0,n,m}(t;\tau)|$ with $t \in \{t_1, t_2 \ldots t_s\}$, we can measure the value of $|h_{0,n,m}(t;\tau)|$ and subsequently measure $|h_{0,n,m}(t+1;\tau)|$ and determine if they are correlated. As we increase N and M, and adjacent samples of $|h_{0,n,m}(t;\tau)|$ continue to be correlated, we may be able to correctly distinguish $|h_{0,n,m}(t;\tau)|$ from $|h_{1,n,m}(t;\tau)|$ and differentiate the respective

transmitters.

1.3.3 Authentication Using Channel State Information

If a receiver should authenticate $\boldsymbol{H}_0$ and otherwise deny authentication, the properties of $\boldsymbol{H}_0$ must be identified. When the receiver antennas and transmitters are each spaced greater than a carrier frequency wavelength apart, the complex-valued elements of the CSI are statistically uncorrelated. For NLOS channels, the magnitude of these elements are Rayleigh distributed, and for line of sight (LOS), the magnitudes are Ricean distributed. The phase component is uniformly distributed for both NLOS and LOS.

Unfortunately, based on Equation 1.6, the receiver does not directly detect $\boldsymbol{H}$, but also some amount of noise $\boldsymbol{n}$. Therefore a tolerance about the value of the CSI elements is needed to account for small, random variations of the detected CSI. We could create an authentication scheme where if the values of the received CSI elements are bounded by an established tolerance, the transmitter(s) would be authenticated. If the received CSI elements exceed the tolerance, authentication would be denied.

The attack to this model would be for an adversary to emulate a particular channel response from a legitimate transmitter to the receiver. As no single transceiver would be able to directly measure the CSI between other receiver-transmitter pairs, attempts to conduct an impersonation attack against such an authentication scheme will be akin to guessing random values as shown in [43]. However, a determined actor may expend significant resources to either guessing or using a nefarious scheme to impersonate a legitimate device. For this reason, a strong protection mechanism must be in place.

One such protective mechanism is the judicious selection of the tolerance needed to authenticate. If the tolerance is too large, resourceful and nefarious users may be able to spoof legitimate transmitters. On the other hand, too small a tolerance denies authentication to legitimate transmitters if $\boldsymbol{n}$ exceeds an expected magnitude. This dissertation investigates the use of machine learning algorithms to implicitly find the tolerance that optimizes the authentication accuracy of the receiver.

While the channel response will change when either the transmitter or receiver moves, Xiao et al. [44] provided a scheme to conduct physical-layer authentication for mobile

terminals in a multi-carrier system. In Xiao's work, the authors presented two strategies to overcome the challenge of decorrelation between time-sampled CSI elements from the same transmitter: (1) for inter-burst authentication where more time is expected to have passed between successive frames, initiate the sequence with a higher OSI-level authentication and then transmit the previously measured channel response within the data burst as a key used to discriminate users; (2) for intra-burst authentication between frames in the same time slot after having already initiated authentication with higher OSI-level techniques, they showed that a Neyman-Pearson test against channel response measurements in consecutive frames outperformed a least-squares adaptive filter method in terms of correct detection rates and overhead.

The nature of the RF environment between any two receiver-transmitter pairs has also been researched in an effort to exploit a way to generate cryptographic keys. Instead of using a bit sequence stored by a device, the measured CSI is used to determine how messages will be encrypted. In [45] a secret key generation system was developed following a round of training broadcasts by two cooperative transceivers. In [46], an algorithm was developed based on the channel response and a secret key was generated with relatively low computational power demands. Using the theory of channel reciprocity, Quist and Jensen [47] along with Wilson, et al. [48] explored the upper bound for the number of bits that can be generated based on channel estimation between beamformed antenna systems. Even by simply measuring the received signal strength (RSS), it is possible to differentiate between transmitters as shown in [49], [50].

In addition to distinguishing between legitimate users with CSI, it is also informative to detect spoofing attempts by illegitimate users [28], [44], [51], [52]. By recognizing an attack, that data can be used to adjust security posture or take appropriate measures. Also, dismissing these attempts at the physical layer and not letting illegitimate users try to authenticate to the network prevents a resource attack at higher levels in the OSI.

1.4 Machine Learning in the RF Domain

Machine learning for a variety of uses is well suited to the rich feature-space of raw physical-layer wireless signal data. The application of machine learning within the RF domain can be accomplished by understanding the overarching purpose and applying an algorithm to

satisfy the task. Machine learning tasks that are appropriate to RF-related applications include classification, regression, and anomaly detection. There are a variety of algorithms that may be applied to the same task with various levels of performance. Determining the task and choosing the most suitable algorithm to employ will help ensure the results of the machine learning process are optimized.

Classification is the task of determining which of n classes an input should be labeled. The properties of the input $\mathbf{x}$ are examined by the machine learning algorithm f and an output y provides either the class to which $\mathbf{x}$ belongs, or the probability distribution of the class $\{1, \ldots, n\}$ that $\mathbf{x}$ is a member. Implementing this function, $y = f(\mathbf{x})$ can be used to differentiate among various received signals. Depending on the algorithm and the features contained in the input, a signal can be classified by categories such as modulation type, transmission specification, and/or source transmitter.

The regression task is used to predict a future output or outputs based on a given input(s). Like the classification task, the machine learning function f operates on the input and provides an output. However, for regression, a singular output shares the characteristics of a singular input with respect to dimensionality. An example of an application for regression within the RF domain is the prediction of received signal strength.

The task of determining if inputs conform to regular behavior or if they represent unusual events can be accomplished with anomaly detection. By surveying a range of inputs, those that represent statistical outliers can be flagged as anomalies. Anomaly detection in network traffic can be used to identify malicious behavior. Atypical RF received transmissions based on spectrum use, frequency of broadcast, and signal magnitude can all be used to determine a change in channel environment, transmitter operating characteristics, or receiver performance.

1.5 Machine Learning Algorithms

Machine learning algorithms are broadly separated into two categories: unsupervised learning and supervised learning [53]. Unsupervised learning algorithms extract properties of features contained in a dataset of samples. Supervised learning algorithms use the dataset features and also associate each sample with a label or target variable. The labels may indicate the category to which a sample belongs for a classification task, and for a regression

task, the target variable is the dependent variable correlated to an independent input [54].

A third category for machine learning algorithms is known as semi-supervised learning [55]. This category uses a dataset of labeled and unlabeled samples. Semi-supervised learning requires that only a portion of the training data be labeled. As opposed to supervised learning where all the training data is labeled or unsupervised learning where there are no labels and the algorithm must find its own way to organize the data, semi-supervised algorithms attempt to correctly identify samples when only a small portion of the training data is labeled. This can be very helpful when the dataset is large and it would be laborious and time-intensive for an expert to correctly label every sample manually [55].

1.5.1 Support Vector Machines

The support vector machine (SVM) is a binary classifier, or discriminator, that uses a supervised training dataset to determine which of two classes a sample belongs. The SVM creates a hyperplane that separates the training data based on the space defined by the samples, and then assigns new data to one of the two categories [54].

A limitation to using SVMs for RF-related tasks is that the dataset requires samples from two different classes. Extensions to the traditional SVM include multi-class SVMs [54] as well as the one-class support vector machine (OC-SVM) [56] that is discussed later in this section. Unfortunately, achieving high accuracy with SVMs is a challenge, especially when dimensionality is increased [57].

Physical-layer authentication with a SVM algorithm using channel responses in an orthogonal frequency division multiplexing (OFDM) system was proposed by Liu et al. in [58]. Liu showed that by using CSI, authentication accuracy in indoor environments achieved 92% accuracy, compared with 40% accuracy using received signal strength. Liu also noted that authentication accuracy improved when multiple receiver and transmitter antennas are used with their CSI approach.

1.5.2 K-Nearest Neighbor

One method to classify RF samples is to give the samples the same class as the samples to which it is nearest. The k-nearest neighbor (kNN) algorithm measures the Euclidean

distance from one sample to the other samples, and then filters to the k samples that have the least distance [57]. The new sample is then assigned to the cluster that most of its k neighbors share. kNN can also be used in a regression task, where the target variable for the sample is the average of the values from the k neighbors.

Like the SVM algorithm, implementing kNN is simple and straightforward. However, as the number of samples and dimensions increase, the process is greatly slowed [57].

In addition to using SVMs in [58], Liu et al. also used k-means clustering followed by kNN classification to determine if a spoofing transmitter is active during the physical-layer authenticating process.

1.5.3 One-class Machine Learning Algorithms

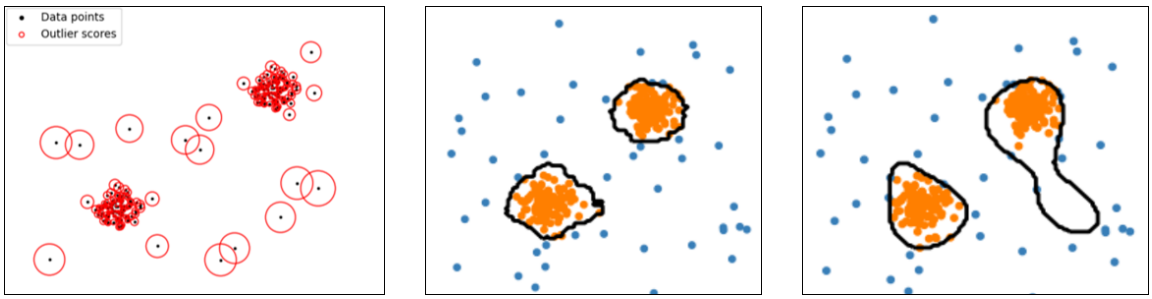
There is broad use for detection systems that identify RF samples with features outside of the expected. A one-class machine learning problem trains on data from a single class, and during testing the tool will provide an output recognizing new samples as either inliers of the statistical distribution or outliers.

Local outlier factor (LOF) from Breunig et al. in [59] quantifies the relative degree of isolation of a sample to its neighbors. For samples with density similar to that of the local cluster, the LOF value approaches 1. Inlier samples with higher density than their neighbors have a LOF less than 1, and outliers have a LOF greater than 1.

Isolation forest (iForest) from the works of Liu et al. in [60], [61] does not use density measurements to detect anomalies, but randomly selects a threshold between maximum and minimum values of a randomly selected feature. The iForest algorithm takes advantage of the defining characteristics of an anomaly: there are few of them, and they have features unlike normal samples.

The OC-SVM as presented by Schölkopf et al. in [56] is an extension to the traditional SVM technique but uses unsupervised learning and unlabeled training data. The goal of the OC-SVM algorithm is to determine the smallest region containing the training data. The algorithm returns a value of 1 if a new sample falls within this region, and a -1 is returned for samples outside the threshold.

A depiction of these one-class algorithms is shown in Figure 1.4. The LOF algorithm in Figure 1.4a illustrates the relative degree of isolation using circles of different sizes, where outlier samples have larger circles around them compared to the inlier samples. In Figures 1.4b and 1.4c, we can see the decision boundary surrounding the samples constructed by the iForest and OC-SVM algorithms, respectively.



(a) LOF (b) iForest (c) OC-SVM

Figure 1.4. One-class machine learning algorithm examples. Source: [62].

A large advantage in implementing these one-class machine learning tools is the speed at which they can be trained and tested [61]. Unfortunately, the performance of these tools may not scale as the dimensionality of the task increases [62]. Additionally, if anomalous samples or CSI from rogue or nefarious transmitters enter the training dataset, these samples would be characterized as belonging to the trusted class since the distribution of the dataset would be changed in such a way to accommodate these illegitimate samples.

1.5.4 Neural Networks

One of the most powerful and versatile implementations of machine learning is the artificial neural network (ANN) (hereafter, neural network). In 1943, McCulloch and Pitts [63] developed the first model based on how neurons in the brain work and demonstrated that a small network of artificial neurons could replicate any logical computation. In 1957, Rosenblatt invented the perceptron algorithm, in which a layer of artificial neurons sum multiple weighted inputs and produce a value based on a threshold or activation function [64]. A neural network that has at least two hidden layers is called a deep neural network (DNN) [16].

By adding multiple perceptron layers, and training through backpropagation, the multiple-layer perceptron (MLP) eliminated earlier limitations and gained the ability to efficiently correct errors by adjusting the weighted connections between neurons [16]. During the training of an ANN, the network-generated output is compared to the ground truth. A loss function acts upon the output and ground truth to create a loss value. The loss value is then applied to layers and neurons in the network so that the network will iteratively improve. The backpropagation process updates the initially randomized weights and biases connecting the neurons. Thus, the network "learns" as it minimizes errors until an acceptable success rate is reached. When the error is minimized, the training of the neural network is stopped, and the network weights can be saved and applied to more samples. If the probability distribution is the same between the training data and the new samples, the network will produce new results with similar performance compared to the last training iterations.

In the remainder of this section, we will discuss three types of ANNs and their uses. First, we introduce the densely-connected neural network, then then convolutional neural network (CNN), and finally the recurrent neural network (RNN).

Densely-Connected Neural Networks

When the artificial neurons in a layer of a MLP network are each connected to every artificial neuron in the previous and the next layer, this is known as a fully-connected, or densely-connected, layer. A drawback to this arrangement is that densely connected layers can easily increase in complexity and are susceptible to producing overfit solutions [54]. One way to prevent overfitting and provide regularization is to implement a technique called dropout [65]. During a forward pass, dropout randomly turns off a fraction of the artificial neurons in a given layer, essentially disconnecting those neurons from the greater network. On subsequent forward passes, neurons in the layer are randomly turned on or off based on the fraction assigned to the dropout, producing a layer that isn't reliant on the response from a single neuron, and resistant to a single neuron dominating the solution.

Densely-connected neural networks can automatically discover features in a dataset, resulting in accuracy improvement over hand-selected features used with the previously mentioned machine learning algorithms [53]. In cases where a high level of accuracy is required, and especially if the samples contain high dimensionality, the neural network is the preferred algorithm [66].

Convolutional Neural Networks

Another method to achieve regularization is the use of a CNN [67]. A CNN applies a convolutional operation at a layer before passing the result to the next layer. Convolutional operations create feature maps that allow the network to produce accurate solutions for the task [68]. For example, a classification task that identifies basic shapes may include feature maps that indicate the edges of objects. If the feature maps present an arcing or round shape, the network may classify the object as a circle. Likewise, with a feature map with strong responses in the vertical and horizontal directions, the network may classify the object as a square, and if the feature maps have the greatest values in a diagonal orientation, the output of the network may indicate the object is a triangle. Each layer in a CNN further abstracts the output from the previous layer, resulting in a tool that is adept at extracting visual features and well-suited for image-related projects.

CNNs take advantage of the spatial relationship of features within a dataset. By filtering the properties of larger features into their constituent parts, the CNN can be used to identify salient details and categorize accordingly. They then are best used when the data contains spatially correlated features [53].

O'Shea and Hoydis in [69] and O'Shea et al. in [66] used a CNN for classifying signals using different modulation schemes. In [69], O'Shea and Hoydis were able to improve the modulation classification performance over other machine learning algorithms using a CNN below 10 dB SNR. In [66], O'Shea modified the CNN architecture using residual units and gained improvement over the baseline CNN with synthetic and universal software radio peripheral (USRP) over-the-air samples.

Recurrent Neural Networks

The application of a RNN to RF tasks is appropriate when the samples can be arranged as time-series data [70]. With time-series data, each sample in the dataset is consecutive with respect to the time domain. RNNs have been shown to be successful with temporally correlated data rather than data that is only spatially correlated.

For a sequence of samples, a RNN uses the output obtained with the previous time step as part of the input to calculate the output for the next time step as shown in Figure 1.5. In this example, **x**, **h**, and **o** are time-series vectors respectively representing the input, hidden layer

state, and the output of the network. The neural network weight matrices are given by $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$. The hidden layer state is also an output from the current cell to the next cell and is calculated as $\mathbf{h}_t = g_1(\mathbf{U} \cdot \mathbf{x}_t + \mathbf{V} \cdot \mathbf{h}_{t-1})$, while the output is $\mathbf{o}_t = g_2(\mathbf{W} \cdot \mathbf{h}_t)$, where g_1 and g_2 are activations that may include functions such as sigmoid or hyperbolic tangent (tanh) which are defined as

$$\text{sigmoid} := \sigma(z) = \frac{1}{1 + e^{-z}} \tag{1.9}$$

$$\text{hyperbolic tangent} := \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}. \tag{1.10}$$

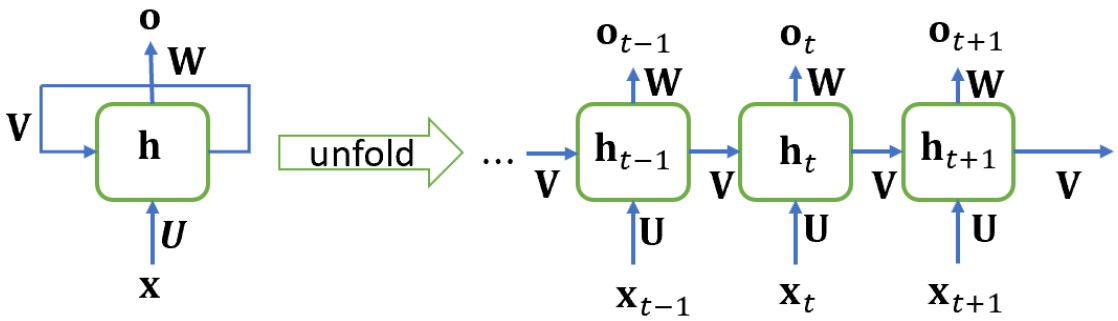


Figure 1.5. Recurrent neural network. Source: [71].

RNNs can be beneficial when the sequential samples are not independent from one another and has applications for time-series prediction [72] and anomaly detection [73]. The RNN architectures discussed in this dissertation use variations of long short-term memory (LSTM) and gated recurrent unit (GRU) cells illustrated in Figure 1.6.

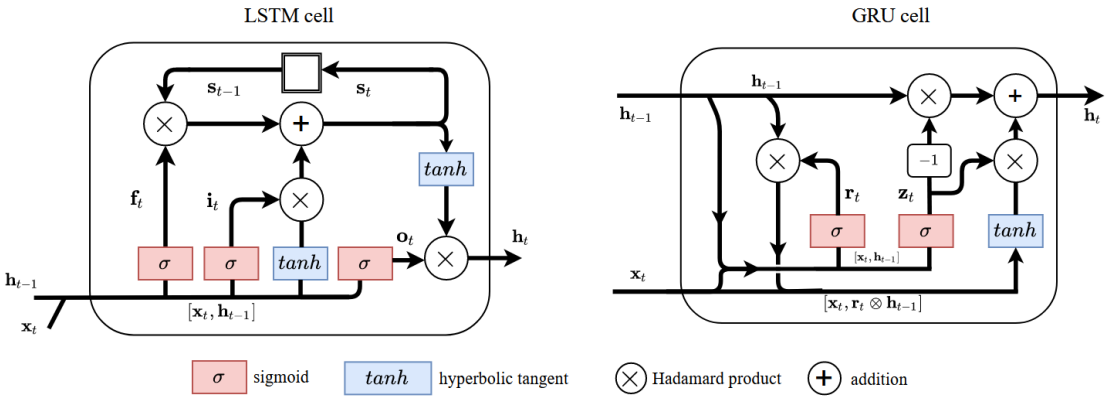


Figure 1.6. LSTM and GRU cell. Source: [71].

LSTM cells addressed the exploding of gradient problem of previous RNNs [74]. The current input vector $\mathbf{x}_t$ is concatenated with the output vector of the previous LSTM cell $\mathbf{h}_{t-1}$. Concatenation is denoted by $[\cdot,\cdot]$, producing $[\mathbf{x}_t, \mathbf{h}_{t-1}]$. The output is $\mathbf{h}_t$ and the current state of the cell is $\mathbf{s}_t$ while the previous cell state is $\mathbf{s}_{t-1}$. The LSTM cell has three gates named *input* ($\mathbf{i}_t$), *output* ($\mathbf{o}_t$), and *forget* ($\mathbf{f}_t$). The equations for these functions are

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \cdot [\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_i) \tag{1.11}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \cdot [\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_o) \tag{1.12}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \cdot [\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_f) \tag{1.13}$$

$$\mathbf{s}_t = \mathbf{f}_t \otimes \mathbf{s}_{t-1} + \mathbf{i}_t \otimes \tanh([\mathbf{x}_t, \mathbf{h}_{t-1}]) \tag{1.14}$$

$$\mathbf{h}_t = \tanh(\mathbf{s}_t) \otimes \mathbf{o}_t, \tag{1.15}$$

where $\mathbf{W}$ and $\mathbf{b}$ are the weight matrices and bias vectors for the LSTM gates. Element-wise multiplication, or the Hadamard product, is symbolized with $\otimes$.

The GRU cell [75] uses two gates, *reset* ($\mathbf{r}_t$) and *update* ($\mathbf{z}_t$), thus requiring less parameters and fewer tensor operations compared to the LSTM cell. These gates and the output $\mathbf{h}_t$ are

calculated by

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \cdot [\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_r) \tag{1.16}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \cdot [\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_z) \tag{1.17}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \otimes \mathbf{h}_{t-1} + \tanh([\mathbf{x}_t, \mathbf{r}_t \otimes \mathbf{h}_{t-1}]) \otimes \mathbf{z}_t. \tag{1.18}$$

Although GRUs are less computationally expensive, performance superiority between GRUs and LSTMs is task dependent [76], [77].

Several works have explored the use of RNNs to predict channel characteristics such as Liu et al. [78] for narrowband prediction, and [79] where Ding et al. used a recurrent complex-valued neural network to improve prediction results. Jiang and Schotten [80] demonstrated the use of LSTM and GRU cells for improving channel prediction over previous RNNs. Wang et al. [81] used convolutional layers and RNNs for physical-layer authentication using CSI in a stationary office environment. Roy et al. [82] used LSTM and GRU cells to classify transmitters based on in-phase and quadrature time-series measurements.

1.5.5 Summary of Machine Learning Algorithms

For most cases simple machine learning techniques, such as SVM, kNN, and the one-class algorithms, are straightforward to implement and provide relatively fast results [54], [62]. For all supervised training algorithms such as SVM and kNN, all the training samples must be fully labeled. Unfortunately, these tools also suffer inaccuracy as dimensionality increases [62]. In modern wireless communications, the physical layer can be modeled with several dimensions to account for properties including magnitude, phase, subcarriers, multiple antennas, and time. Additionally, the kNN algorithm greatly slows down as the number of samples gets larger [57].

When there is sufficient training data available, the processing and time expense required for neural networks can be translated into improved accuracy and speed [54]. While densely connected neural networks can identify parametric features in the data, they do not generally perform well when the data are spatially or temporally correlated [53]. For spatially correlated data, the filtered extraction of low-level features is best accomplished using convolutional operations within the CNN [53]. When the data can be described as a time series, RNNs are the preferred algorithm choice [70].

Even if the training process takes place in a benign environment, it should be understood that none of these algorithms are fully protected against an adversary transmitter. A nefarious user could learn the authentication process and mimic the features that would allow their transmitter to be authenticated. However, this also holds true for password or encryption key protected networks.

1.6 Generative Adversarial Networks

Introduced by Goodfellow et al. [83], the GAN framework trains two artificial neural network models called the discriminator and the generator as they compete against each other in an adversarial competition.

The competition is a minimax game where the discriminator attempts to correctly label training samples from a data distribution and fake training samples created by the generator. The discriminator is trained to maximize the probability of assigning the correct label, while the generator is trained to minimize the same probability. The loss function for the discriminator $J^{(D)}$ is the cross-entropy function when training a binary classifier with sigmoid output [84]

$$J^{(D)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] - \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))], \quad (1.19)$$

where $\mathbb{E}$ is the expectation with respect to the distribution identified in the subscript, $D(x)$ is the probability that x came from the data distribution $p_{data}(x)$ containing real training samples, z is a random variable generator network input, and $D(G(z))$ is the estimate of the probability that the discriminator incorrectly identifies the fake instance as authentic. The weights of the discriminator and generator networks are given by $\boldsymbol{\theta}^{(D)}$ and $\boldsymbol{\theta}^{(G)}$, respectively.

To specify the loss function for the generator, we use the concept of a zero-sum game, where the sum of the loss functions for both players is zero, that is $J^{(G)} = -J^{(D)}$. When optimizing Equation 1.19, the $1/2$ constant is irrelevant, resulting in the value function that describes this game from the original work by Goodfellow [85] given by

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]. \quad (1.20)$$

The generator network attempts to maximize Equation 1.20, while the discriminator network

tries to minimize it. [85]

Figure 1.7 shows a functional depiction of a GAN in training where the discriminator D, receives real samples from the training data or fake samples from the generator network G. The discriminator then assigns a probability from zero to one based on whether the sample is fake (0.0) or real (1.0). The classification error is calculated, and the loss value is provided to the discriminator in order to update the trainable parameters in the network. When the discriminator assesses fake samples, the loss value for the generator is also provided to the generator network [86].

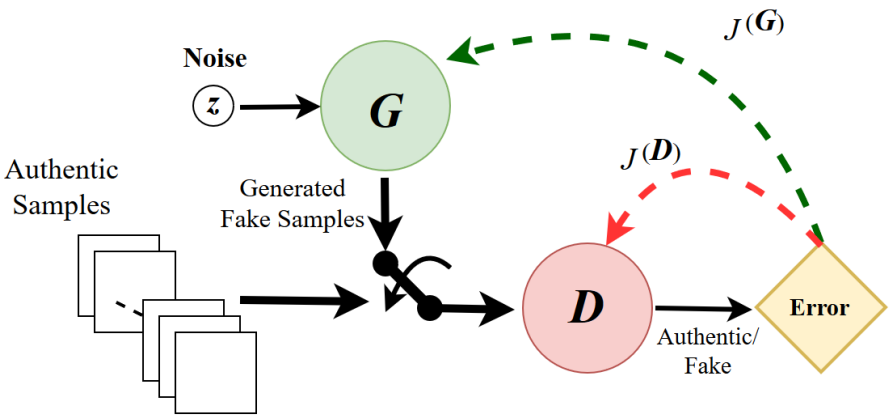


Figure 1.7. Generative adversarial network. Source: [87].

As each entity battles each other, they learn to improve their individual performance. If the discriminator output correctly identifies fake samples created by the generator network, the generator network will update its weights through backpropagation. Likewise, the discriminator network will update its weights when it incorrectly identifies real or fake samples. The results of this training are a generative neural network adept at creating data that closely mimics training data and a discriminator neural network that can identify all but the best fakes. The generator attempts to capture the data distribution $p_{data}(x)$ and the discriminator estimates the probability that a sample came from the training data rather than the generator [85].

The GAN has enjoyed great success in the fields of image processing and computer vision, especially when high resolution is required [88]–[90]. As mentioned by Goodfellow in [84],

after training is completed the discriminator is usually discarded, while the generator is retained and is used to create images or other domain-relevant output. A trained generator may perfectly mimic authentic data resulting in a coin-flip guess by the discriminator. However, at the end of training the discriminator network has also learned, suggesting that a trained discriminator will outperform all but fully-trained generative networks.

1.6.1 Semi-Supervised GAN

With semi-supervised learning, a small percentage of the training data is labeled, and instead of using a binary classifier, the discriminator is a multi-class classifier. For N classes, the model requires $N + 1$ outputs to account for all the authentic classes plus one additional class for the fake generated class [91].

Figure 1.8 shows a functional depiction of a semi-supervised generative adversarial network (SGAN) in training. The training dataset is partially labeled and provided to the $\mathcal{D}/C$ model for classification by C. The remainder of the training dataset, as well as the generated samples from $\mathcal{G}$, are used as input to $\mathcal{D}/C$ for discrimination where $\mathcal{D}$ will predict whether the sample came from the training dataset or if it was created by $\mathcal{G}$.

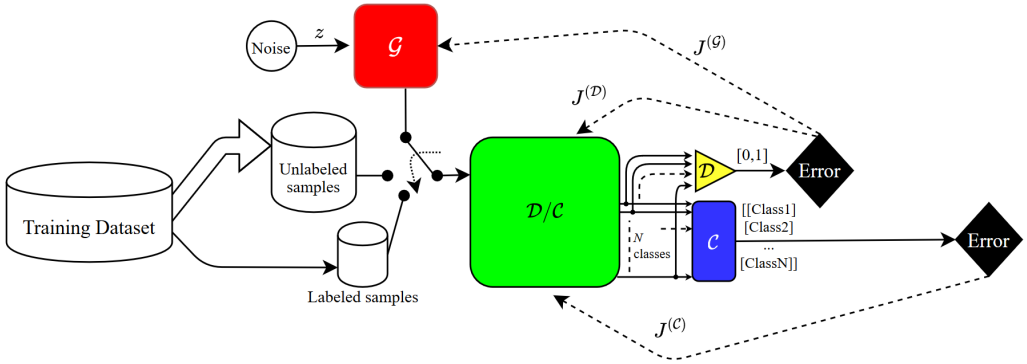


Figure 1.8. Training a semi-supervised generative adversarial network with N classes. Adapted from [38].

Based on earlier work by O'Shea et al. for radio modulation classification [92], Li et al. [93] employed a SGAN to classify 11 different modulation types and improved the classification performance over a CNN model. The CsiGAN was proposed by Xiao et al. in [94] for

CSI-based human activity recognition. Xiao used a SGAN against human activity datasets to improve classification accuracy.

1.6.2 CGAN

Further extending the GAN framework, the conditional generative adversarial network (CGAN) incorporates extra information for the discriminator and generator. In addition to the inputs that the vanilla GAN uses, conditional information is also provided to $\mathcal{D}$ and/or $\mathcal{G}$. The conditional information can be a label or other target variable. Figure 1.9 illustrates a CGAN during training. The CGAN generator and discriminator neural networks can be made using densely connected layers, convolutional layers, recurrent cells, or any combination thereof. With conditional information y, the CGAN value function, based on Equation 1.20, becomes

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|y)))]. \quad (1.21)$$

Works for CGANs using recurrent networks include Esteban et al. [95] where a recurrent CGAN with LSTM cells was used to produce realistic time-series medical information. Additionally, Koochali et al. [96] explored using either LSTM or GRU cells in a CGAN to forecast one-step ahead values in datasets related to weather measurements, the Mackey-Glass time-delay differential equation, and internet traffic. Like many GAN-based applications, the CGAN has also been applied with success in imagery-dependent fields, however within the RF field, a CNN-CGAN was used by Dong et al. in [97] for channel prediction.

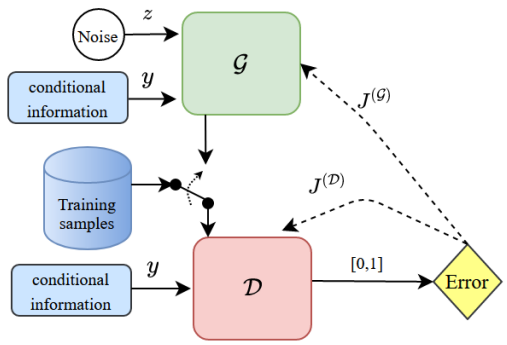


Figure 1.9. CGAN training architecture. Adapted from [71].

1.7 Contributions of This Dissertation

The focus of this dissertation is the improvement of wireless communication security using adversarial machine learning algorithms. We propose the use of three adversarial training frameworks to make an authentication decision at the physical layer in an RF channel. We analyze the amount of training data needed and required iterations of that data through our proposed neural networks to reach accuracy metrics. We also explore the impact of degraded channels on authentication accuracy by using CSI at a variety of SNR levels.

The main contributions of this dissertation are:

- We introduce analysis and simulation illustrating how the received CSI matrix elements and measurement error can be used for physical-layer authentication. We create a hypothesis test and also develop a GAN for the purpose of determining if received CSI samples should be permitted or denied authentication. Our generative model creates fake CSI samples that closely match the characteristics of legitimate CSI matrix samples from a trusted source. We compare the accuracy yielded by the GAN to the hypothesis test approach as well as three one-class machine learning algorithms.
- Using a dataset modeling a multipath NLOS Rayleigh channel, we take a semi-supervised approach to discriminating legitimate from nefarious transmitters. We develop two SGAN models and compare their performance against that of a densely-connected ANN classifier and a CNN classifier at various SNR levels. Our SGAN models can be used to classify transmitters by MIMO CSI as a method to provide physical-layer authentication. Requiring only a small portion of the samples to be labeled, we effectively reduce overhead while still achieving high classification accuracy at low SNR levels.
- Simulating the implementation of a SGAN using 5G millimeter wave carrier frequencies, we assess SGAN classification accuracy based on receivers in close proximity. We create CSI matrices that correspond to receiver positions in an urban setting using a MATLAB-based tool that leverages ray-tracing software. We then compare the performance of the SGAN across a range of SNR levels to other neural network models. Our proposed SGAN-based physical-layer authentication system can be implemented to provide high authentication accuracy at even low SNR values. The system first uses the SGAN-trained discriminator to allow only trusted transmitters to authenticate. The SGAN-trained classifier then identifies the trusted transmitter and can be

used to allow the user a tailored degree of access.

- Finally, we propose two novel methods for accomplishing channel prediction and physical layer-authentication for mobile devices. Specifically, our method uses a CGAN to predict mobile MIMO channel response magnitudes. Our CGAN uses RNNs and conditional information in the form of previous channel responses. We propose two methods to make an authentication decision: (1) an adversarially-trained discriminator can be used to authenticate or deny access, and (2) based on the difference from the predictive channel responses and the true channel response, a threshold based on mean-squared error (MSE) can be used to grant or deny authentication. We then compare the performance of our adversarial models to alternative RNNs.

1.8 Organization

This chapter introduced the objective and contributions of this dissertation and presented an overview on machine learning applied to RF tasks. Chapter 2 reviews the salient literature to identify related works concerning physical-layer authentication, the application of machine learning tools, and deep learning model techniques that are most appropriate for wireless authentication. Next, in Chapter 3, we propose a method to conduct physical-layer authentication by training neural networks in a GAN framework. Using a variant of GAN-trained neural networks, we differentiate transmitters based on CSI in Chapters 4 and 5. Using recurrent neural networks, we propose methods to conduct physical-layer authentication based on time-series CSI in Chapter 6. Finally, we conclude by summarizing the results and contributions of this dissertation and discuss future work in Chapter 7. An appendix is provided to introduce some of the fundamental elements regarding neural networks. The scope of the appendix includes mathematical operations in feedforward neural networks and common terms associated with the training process.

The contents of this dissertation include material adapted from work published and to be published by the author. Sections from Chapter 3 contain some revised material from “Physical-Layer Authentication Using Channel State Information and Machine Learning" by Ken St. Germain and Frank Kragh, published in the 14th International Conference on Signal Processing and Communication Systems [87]. Sections from Chapter 4 contain some revised material from “Multi-Transmitter Physical-Layer Authentication Using Channel State Information and Deep Learning" by Ken St. Germain and Frank Kragh, published in the

14th International Conference on Signal Processing and Communication Systems [98]. Sections from Chapter 5 contain some revised material from “Multi-subcarrier Physical-Layer Authentication Using Channel State Information and Deep Learning” by Ken St. Germain and Frank Kragh, published in the 54th Hawaii International Conference on System Sciences [38]. Sections from Chapter 6 contain revised material from “Channel Prediction and Transmitter Authentication with Adversarially-Trained Recurrent Neural Networks” by Ken St. Germain and Frank Kragh published in the Institute of Electrical and Electronic Engineers (IEEE) Open Journal of the Communications Society [99] and also “Mobile Physical-Layer Authentication Using Channel State Information and Conditional Recurrent Neural Networks" to be published in the 93rd Vehicular Technology Conference: VTC2021-Spring [71].

CHAPTER 2:
Literature Review

This chapter reviews and explores the most recent and seminal works in machine learning in communication systems, physical-layer authentication, and GANs.

2.1 Physical-Layer Authentication

Depending on their use, accessibility, and connectivity, authentication in computer systems must be appropriately strong to prevent attacks resulting in unauthorized access or modification of data [100]. In a wireless communication environment, data is transmitted to a wide audience of users, many who are unintended recipients. Included among these recipients may be nefarious users with malicious intent. For this reason, wireless data transmission is typically protected through the employment of cryptographic methods at the OSI network-layer or higher [13]. Cryptographic systems have been shown to improve transmission confidentiality, however there is a cost in the form of computational power and latency [101]–[103]. Because of issues such as energy needs, processing capability, and storage requirements, many IoT devices do not support strong security mechanisms [104].

Authentication is often employed at various levels of the OSI model: media access control (MAC)-layer authentication [105], network-layer authentication [106], transport-layer authentication [21], and application-layer authentication [107]. While employing authentication at different layers may enhance wireless security, the overhead remains a debt to be paid in computational complexity and latency [108], [109]. Additionally, cryptographic systems have vulnerabilities [14] and have not proven to be computationally unbreakable [13]. Another drawback for cryptographic systems is the challenge of key distribution and management in a decentralized, dynamic, and heterogeneous network [11], [12]. As suggested in [110], authentication can be accomplished at least in part, if not fully, at the physical layer.

2.2 Machine Learning in Communication Systems

Using the spatially-unique properties of the CSI, many researchers have leveraged machine learning and deep neural networks in particular to successfully resolve localization challenges.

There are several examples in the literature where authors mapped collected CSI to the a priori known position of legitimate transmitters, trained machine learning systems to recognize those signatures, then accurately determined the position of the transmitter [111]–[113].

Nerguizian et al. [111] used a neural network to learn and then match signatures generated by a transmitter located in different areas of an underground mine. Four hundred ninety measurements were taken, composed of seven channel parameters and the true position of a transmitter. The channel parameter information was fed to the input of an ANN and was trained using the true position. During testing, the ANN output was the estimated user position and was accurate within 2 meters for 90% of the trained positions.

Wang et al. [112] used a densely connected ANN system with four hidden layers trained on CSI and positional data to determine the location of an indoor transmitter. In two different scenarios, one with many line-of-sight paths and one with few line-of-sight paths as shown in Figure 2.1, CSI measurements were taken off-line at predetermined locations with half-meter spacing. During testing, new positions in each room were selected, and the localization performance exceeded that of methods that solely use RSS and another scheme that used the weighted CSI location average over multiple antennas, known as fine-grain indoor fingerprinting system [113].

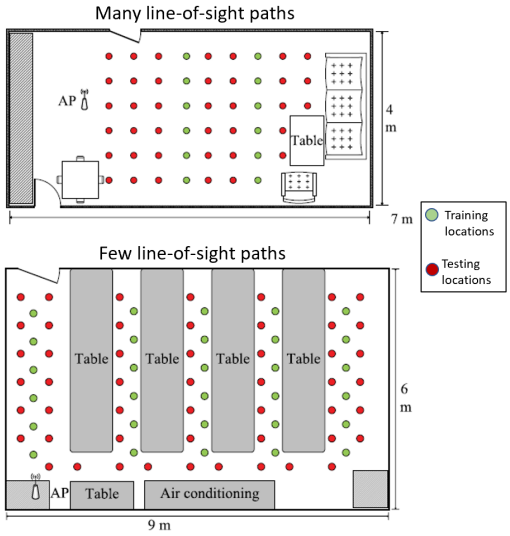


Figure 2.1. Training and testing locations in two different rooms. Adapted from [112].

2.3 Authentication Through Machine Learning and CSI

Taking the concept of position identification a step further, much research has been conducted with machine learning and location information to make an authentication decision based on CSI.

Wang et al. [81] used three USRPs in an office setting to generate an equal number of channel estimates pre-labeled as either legitimate or illegitimate. The networks they explored included a RNN, a hybrid CNN-RNN architecture, and a skip-layer CNN. Wang showed that using machine learning algorithms outmatched a heuristic Neyman-Pearson test with respect to authentication accuracy and minimizing false positives and false negatives. Overall, the neural network using the RNN with the CNN gave the best authentication accuracy, scoring 99.7%.

Xiao et al. [114] used channel information and reinforcement learning to detect spoofing attacks in wireless networks. A zero-sum game was constructed between detection and spoofing to determine a threshold for use in authentication. The performance of false alarm rate and missed detections of this static threshold was worse than when reinforcement learn-

ing techniques were used to ultimately decide whether to authenticate the transmitter. Their results show how machine learning can improve physical-layer authentication performance in a dynamic RF environment.

Pan et al. [115] demonstrated using CSI for physical-layer authentication in various environments including indoors, outdoors, moving, and stationary by using data collected by the National Institute of Standards and Technology (NIST) [116]. A location for the legitimate transmitter is chosen, and the corresponding CSI is then used to calculate the normalized channel difference, based on the Euclidean distance from CSI corresponding to chosen location and subsequently sampled CSI data. Using a predetermined threshold, authentication is granted or denied and detection and false alarm rates are calculated. Among their conclusions, they showed that authentication performance was better in scenarios with stationary systems, abundant multi-path effects, and separation of transmitters by more than one-half wavelength. By dynamically changing the channel-based test statistic threshold based on machine learning results [117], Pan et al. improved their successful physical-layer authentication rate over the static method.

Liao et al. [118] compared deep-learning models for physical-layer authentication in an industrial wireless sensor network (WSN) using numerical simulation and experimentation with USRPs. They used a CSI-based scheme where transmitters with two, four, or eight antennas were placed in different locations in an industrial setting, the CSI was estimated by a base station with eight antennas, and then a variety of neural networks were trained. The results showed that a densely-connected ANN had better performance over the two CNNs in terms of authentication accuracy. Liao noted that the training parameters of the densely-connected ANN will grow exponentially as the CSI dimensions increase, resulting in an undesired increase in computational complexity. In another publication, [119], Liao et al. investigated security threats for mobile-edge computing on their CSI-based physical-layer authentication system that used a densely-connected ANN. Liao compared the use of a densely-connected neural network against hypothesis tests using a basis entropy model and a least-squares estimation algorithm. Testing with SNR values ranging from 0 dB to 8 dB, the neural network had the highest authentication accuracy. At 0 dB, the neural network achieved 94% compared to 86.3% for the basis entropy model and 75% for the least-squares algorithm. At 8 dB, the scores were much closer: 100%, 99%, and 99.8% for the neural network, basis entropy model, and least-squares algorithm. This suggests the neural network

is the more suitable choice for lower SNR environments.

In [120], Abyaneh et al. used a CNN trained on pre-mapped transmitter locations in order to grant authentication. If a broadcasting transmitter emitted from one of the pre-mapped locations, the transmitter would be authenticated. They also included the complex value of the CSI element as input to their neural network, whereas the majority of CSI-related works with machine learning only used the magnitude of the CSI elements.

2.4 General Adversarial Networks

As of 2019, there were over 500 types of GAN models [121] and this number has continued to grow. Based on the task being addressed, different adjustments have been made to advance the field. For example, Radford et al. [122] noted the shortcomings in image quality for generative networks and instability in using CNNs for GANs. They then adopted changes to CNNs including using an all convolutional network [123], eliminated fully connected layers [124], employed batch normalization [125], used the rectified linear unit (ReLU) [126] activation in the generator, and leaky rectified linear unit (LeakyReLU) [127], [128] in the discriminator. As a result, Radford developed the deep convolutional GAN, resulting in a stable generative network that created much improved image samples.

The literature contains several cautionary notes on training GANs. Even in Goodfellow's seminal paper on GANs [85], he mentions the difficulty in training, and instead of minimizing $\log(1 - D(G(z))$ in Equation 1.20, it is better to maximize $\log(D(G(z))$ to ensure a strong gradient when training first starts.

Another problem often seen is called mode collapse [84]. The symptoms of mode collapse are a generator that creates only a narrow portion of the data distribution. An example of how mode collapse can be diagnosed is a GAN built to replicate the hand-written numbers used in the Modified National Institute of Standards and Technology (MNIST) dataset. A generator network in mode collapse would only create a couple of any of these numbers, such as one and seven, but never any of the other numbers from zero to nine. A recognized mitigation against mode collapse is to train the GAN using mini-batch discrimination [129].

Once training is underway and mode collapse is avoided, performance evaluation of the GAN is needed. Unlike other neural network models that use loss as a measure of training

completeness [54], the evaluation of GANs is more subjective. Unfortunately, GANs do not have an objective function, so evaluation is typically difficult without human intervention [129], [130], however, it is possible [131].

Advances are being made based on published research and theory-based principles, including more experimental methods shown to stabilize and improve GAN training performance [132], [133] such as:

- Normalize pixel values for images from -1 to 1
- Use maximize $\log(D(G(z))$ for the generator network loss function
- For the generator noise input z don't use a uniform distribution
- Use mini-batches for real and fake samples to the discriminator
- Avoid sparse gradients such as ReLU, however, LeakyReLU is good
- Smooth labels by avoiding solely using 1.0 for real and 0.0 for fake
- Use the adaptive moment estimation (Adam) [134] optimizer

While GANs have successfully contributed to many areas that rely on image processing such as single image super-resolution (creating high resolution images based on low resolution images) [88], medical radiology [135], facial recognition [136], etc., there have been breakthroughs by applying GANs to problems in the RF field as well.

O'Shea et al. [137] used a GAN to determine the optimal modulation scheme to minimize symbol error in a given channel, showing how GANs can allow for adaptation to the RF environment.

The amplitude-feature deep convolutional GAN was used by Li et al. [138] to reduce the effort and increase the accuracy in creating a MIMO CSI-based fingerprint database for a Wi-Fi localization system. The authors collected CSI from subcarriers in an IEEE 802.11n network and converted that information into amplitude feature maps. A GAN using deep convolutional neural networks created additional samples similar to the amplitude feature maps converted from the collected CSI data. By combining the samples created from collected CSI and samples created with generated CSI data, the error distance was reduced compared to only using processed collected CSI data. The results improved the accuracy of locating the position of transmitters in an indoor, classroom setting.

There have also been some works that have researched GANs with a goal to improve physical-layer security.

In an adversarial situation such as jamming and spoofing, Roy et al. [139] proposed a GAN-based method to determine legitimate from illegitimate transmitters based on the imbalance of in-phase and quadrature components of a symbol constellation. Roy again [140] used GANs to train a receiver to classify trusted receivers and identify rogue RF transmitters based on IQ imbalance.

Shi et al. [43] proposed the idea of spoofing a physical-layer authentication scheme with signals from the generator network. In this research, a generator network is trained by positioning an adversary receiver physically close to the intended target receiver. The adversary receiver acts like a discriminator as it senses the channel with legitimate transmitters communicating to the target receiver. An adversary transmitter acts like a generator and trains by communicating with the adversary receiver. When the adversary transmitter generator network converges, it can spoof signals that the target receiver authenticates more than 75% of the time. Their results using the GAN show improved spoofing ability compared to simple replay attacks with incomplete channel knowledge.

Alshinina and Elleithy [141] used GANs to confuse an adversary attack on a WSN. By generating fake, but real-looking data and mixing it with authentic data, there was an increase in authentic data throughput, a decrease in energy consumption, and improved security through the WSN compared to conventional methods.

2.5 Summary

Several works exist and continue to develop in order to improve the goal of physical layer authentication. The advent of machine learning and the use of neural networks has improved the accuracy of a receiver correctly identifying legitimate transmitters that should be authenticated and illegitimate transmitters that should be denied authentication. While most applications using a GAN are focused on the generator network to create realistic-looking images, data, image-to-image translation, etc., the task of classification or anomaly detection is more suited towards the discriminator network.

Based on a thorough review of the literature, and aside from our own published or submitted

papers, we have not been able to find a published work that uses a GAN for physical-layer authentication using CSI in a MIMO configuration. The closest work to ours is Roy's research in [53], [139], [140], where they used IQ imbalance and GANs for the purpose of identifying rogue transmitters and authenticating trusted transmitters.

To implement many proposals, researchers pre-processed collected raw CSI and used the magnitude of the CSI elements $|h_{n,m}|$ in their implementations. Additionally, those that use neural networks used training data that required advance knowledge of CSI between a monitor and either the CSI of an adversary transmitter or possible legitimate and illegitimate transmitters positions. In Chapters 3, 4, and 5, we retain the complex elements of the CSI to retain a rich set of features. Our methodology does not need a priori CSI samples from an attacker's position while we make use of unsupervised machine learning to reach an authentication decision. We also examine the performance of the SGAN and additional neural networks across a range of SNR. In Chapter 6 we study physical-layer authentication in mobile channels and also use the magnitude of the CSI elements $|h_{n,m}|$. Distinguishing from previous research in Chapter 6, this dissertation explores the use of LSTM and GRU cells incorporated into a CGAN architecture for transmitter authentication based on CSI. We use the generator network from the CGAN for channel prediction to authenticate transmitters based on mobile channel conditions. The CGAN discriminator network is also used to authenticate transmitters by assessing the received impulse response conditioned on previously recorded channel responses.

CHAPTER 3:
Adversarial Learning and Authentication

In this chapter, we investigate physical-layer authentication using received complex-valued CSI. We explore various methods to authenticate. First, we introduce a hypothesis test that uses the covariance of the noise introduced to the signal due to the receiver. Next, we develop a GAN model where the discriminator decides if wireless transmissions should be authenticated. Finally, we test these two methods and also compare those results to one-class machine learning algorithms including LOF, iForest, and OC-SVM.

This chapter includes material adapted from work published by the author. Revised material is included from "Physical-Layer Authentication Using Channel State Information and Machine Learning" by Ken St. Germain and Frank Kragh, published in the 14th International Conference on Signal Processing and Communication Systems [87].

3.1 Model for Authentication with CSI

For the scenario presented here, once a transmitter is initially authenticated, a receiver will continue to authenticate a transmitter if the received CSI varies less than a threshold applied to the received CSI from previous transmissions. This requires some method of initial authentication, such as the use of cryptographic methods or physical-layer authentication using RF fingerprinting from transmitter imperfections. During initial authentication, the receiver makes CSI measurements of the channel and stores that information for future authentication.

During channel measurement, even in a stable static environment, the receiver imparts noise to the received signal, resulting in variation to the measured CSI elements. This error $\boldsymbol{\epsilon}$ is modeled as an additive complex zero-mean Gaussian process $\mathcal{CN}(0, \boldsymbol{\Sigma}_\epsilon)$ where the covariance of the sample mean is $\boldsymbol{\Sigma}_{\bar{\epsilon}} = {}^1/_s \sum_1^s \boldsymbol{\Sigma}_\epsilon$ for s samples during the measurement. Therefore, the kth CSI measured by the receiver $\hat{\boldsymbol{H}}_k$ is given as

$$\hat{\boldsymbol{H}}_k = \boldsymbol{H} + \boldsymbol{\epsilon}_k \qquad k = 1, 2, \ldots, s \tag{3.1}$$

where $\boldsymbol{H}$ is the true CSI from Equation 1.6 and $\boldsymbol{\epsilon}_k$ is a complex $N \times M$ matrix with independent identically distributed elements. Since $\boldsymbol{\epsilon}_k$ is zero-mean, $\boldsymbol{H}$ can be estimated with a variety of techniques including least-squares estimation, minimum mean-square error estimation, and through successive measurements and element-wise averaging of $\hat{\boldsymbol{H}}_k$ for $k = \{1, 2, \ldots, s\}$ as demonstrated in [142].

3.2 Authentication Hypothesis Test Based on a Threshold

A threshold is then applied to each CSI element $h_{n,m}$ where the transmitter is authenticated if the distance from every received element $\hat{h}_{n,m,k}$ to the estimated element $h_{n,m}$ from $\boldsymbol{H}$ is less than or equal to a threshold $z_{n,m}$ based on the average eigenvalue λ_{ave} from the covariance matrix $\boldsymbol{\Sigma}_{\overline{\epsilon}_{n,m}}$. To simplify the notation, we will consider $z_{n,m}$ the same value z for all n and m terms, however, in practice z could vary among CSI elements. The numbered sequential transmission count is represented by k. Following the hypothesis testing in [143], we have the null hypothesis $\mathcal{H}_0$ to authenticate, and the alternative hypothesis $\mathcal{H}_1$ to deny authentication such that

$$\begin{aligned}
\mathcal{H}_0 :&(\mathrm{Re}(\hat{h}_{n,m,k}) - \mathrm{Re}(h_{n,m}))^2 \\
&+ (\mathrm{Im}(\hat{h}_{n,m,k}) - \mathrm{Im}(h_{n,m}))^2 \leq z^2 \quad \forall n, m \\
\mathcal{H}_1 :&(\mathrm{Re}(\hat{h}_{n,m,k}) - \mathrm{Re}(h_{n,m}))^2 \\
&+ (\mathrm{Im}(\hat{h}_{n,m,k}) - \mathrm{Im}(h_{n,m}))^2 > z^2 \quad \exists n, m,
\end{aligned} \tag{3.2}$$

where $\mathrm{Re}(\cdot)$ and $\mathrm{Im}(\cdot)$ return the real and imaginary parts of the CSI matrix elements, respectively and z is a tunable parameter that can be adjusted to suit the requirements of the system. To minimize false positives, z can be set to a relatively small value such as $\lambda_{ave}^{\frac{1}{2}}$, and to minimize false negatives, z can be expanded to a greater value such as $6\lambda_{ave}^{\frac{1}{2}}$. As evident in the equation forms for $\mathcal{H}_0$ and $\mathcal{H}_1$, the regions for authentication are circles of radius z centered on the coordinates provided by the CSI elements $h_{n,m}$. In order to successfully authenticate, all elements in $\boldsymbol{H}$ and z must jointly meet the criteria for $\mathcal{H}_0$.

As an example, consider Figure 3.1 illustrating the measured CSI elements for the 2×2 MIMO case. The markers in shades of blue indicate measured samples gathered during the initial authentication. Comparing the red-shaded CSI elements from $\hat{h}_{n,m,1}$ and gray-shaded

elements in $\hat{h}_{n,m,2}$, the desired outcome is that $\hat{h}_{n,m,1}$ will authenticate, but $\hat{h}_{n,m,2}$ will not authenticate due to $\hat{h}_{1,1,2}$ likely being outside of the error tolerance for $z_{1,1}$, where for this case $z = 5\lambda_{ave}^{\frac{1}{2}}$ for all nm terms.

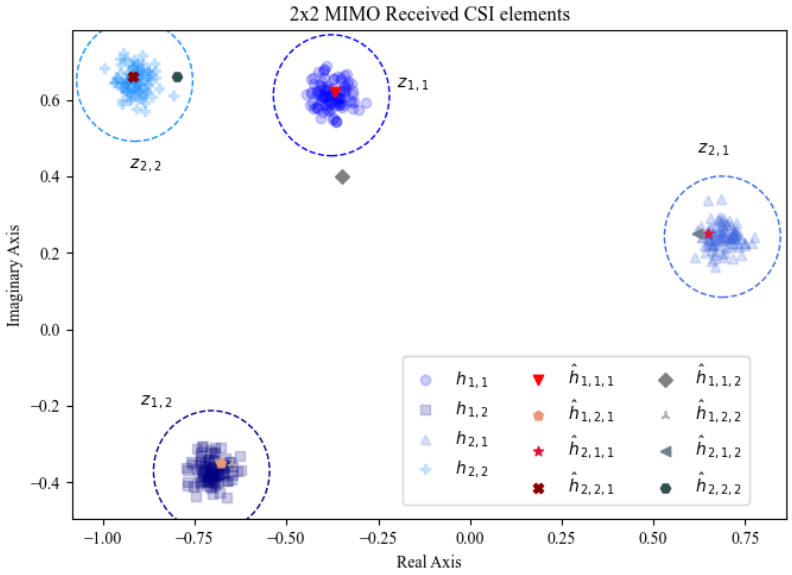


Figure 3.1. Measured 2x2 MIMO CSI elements with receiver noise. Source: [87].

Given an $N \times M$ array of normally distributed random variables, we can determine the probability of one transmitter being accidentally authenticated as another based on the error tolerance for the first transmitter z. Let $a_{n,m}$ be the real part and $b_{n,m}$ be the imaginary part of the complex value for the true CSI element $h_{n,m} = a_{n,m} + jb_{n,m}$. Both $a_{n,m}$ and $b_{n,m}$ are independent Gaussian random variables with variance $\sigma^2/2$. The joint probability distribution function (PDF) for $a_{n,m}$ and $b_{n,m}$ is

$$f(a_{n,m}, b_{n,m}) = \frac{\exp\left(-\frac{a_{n,m}^2+b_{n,m}^2}{\sigma^2}\right)}{2\pi\sqrt{|\boldsymbol{\Sigma}_{a_{n,m},b_{n,m}}|}} \tag{3.3}$$

$$\boldsymbol{\mu}_{a_{n,m},b_{n,m}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \boldsymbol{\Sigma}_{a_{n,m},b_{n,m}} = \begin{pmatrix} \sigma^2/2 & 0 \\ 0 & \sigma^2/2 \end{pmatrix}.$$

To $h_{n,m}$, we add the result of the receiver noise $\boldsymbol{\epsilon}$. The real and imaginary parts of $\epsilon_{n,m}$ are zero-mean independent Gaussian distributed random variables each with sample mean covariance $\Sigma_{\overline{\epsilon}_{n,m}}$.

For a transmitter to be authenticated $\mathcal{H}_0$ must be satisfied for every CSI element $h_{n,m}$. Given $h_{n,m} = a_{n,m} + jb_{n,m}$, we can determine the probability that another transmitter will be authenticated. Let $z = 5\lambda_{ave}^{\frac{1}{2}}$ where λ_{ave} is the average eigenvalue from the receiver noise covariance matrix $\boldsymbol{\Sigma}_{\overline{\epsilon}_{n,m}}$ and u and v be the respective real and imaginary parts of the CSI from another transmitter. The probability of $u + jv$ resulting in $\mathcal{H}_0$ for $h_{n,m}$ is

$$P([u + jv] \in \mathcal{D}_{n,m}) = \iint_{\mathcal{D}_{n,m}} \frac{\exp\left(-\frac{u^2+v^2}{\sigma^2}\right)}{2\pi\sqrt{|\boldsymbol{\Sigma}_{u,v}|}}\, du\, dv$$

where, (3.4)

$$\mathcal{D}_{n,m} = \{(u, v) \mid \left(u - a_{n,m}\right)^2 + \left(v - b_{n,m}\right)^2 \leq z^2\}$$

and,

$$\boldsymbol{\mu}_{u,v} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \boldsymbol{\Sigma}_{u,v} = \begin{pmatrix} \sigma^2/2 & 0 \\ 0 & \sigma^2/2 \end{pmatrix}.$$

With independent u and v, Equation 3.4 can be evaluated using $P(X \cap Y) = P(Y|X) \cdot P(X)$, where $P(X)$ is the probability that $a_{n,m} - z \leq \quad u \quad \leq a_{n,m} + z$, and $P(Y|X)$ is the probability that $b_{n,m} - \sqrt{z^2 - (u - a_{n,m})^2} \leq v \leq b_{n,m} + \sqrt{z^2 - (u - a_{n,m})^2}$. Therefore,

$$P([u + jv] \in \mathcal{D}_{n,m}) = (Q(A) - Q(B)) \cdot (Q(C) - Q(D))$$

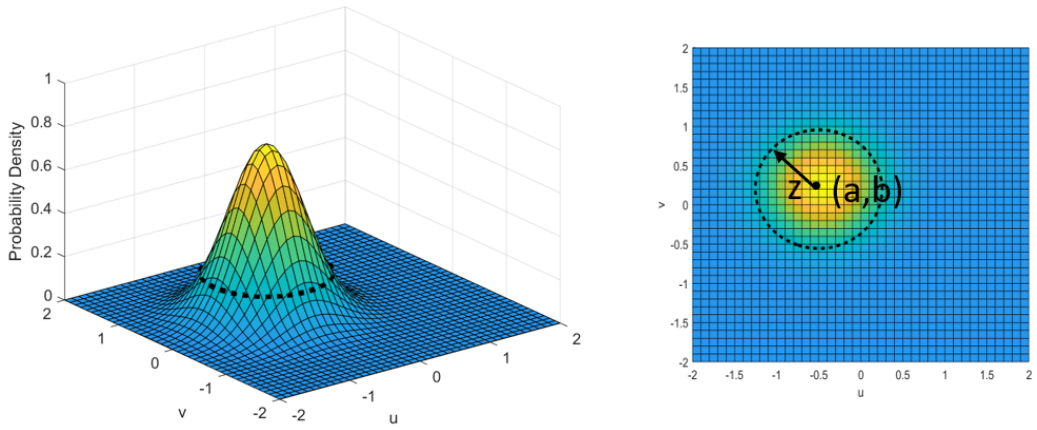
where,

$$\begin{aligned} A &= \frac{a_{n,m} - z}{\sigma} \quad B = \frac{a_{n,m} + z}{\sigma} \\ C &= \frac{b_{n,m} - \sqrt{z^2 - \left(u - a_{n,m}\right)^2}}{\sigma} \\ D &= \frac{b_{n,m} + \sqrt{z^2 - \left(u - a_{n,m}\right)^2}}{\sigma} \end{aligned} \tag{3.5}$$

and the $Q(\cdot)$ function is

$$Q(x) = \int_x^{+\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) dt.$$

As a toy representation, we consider a single CSI element h with $a = -0.5$ and $b = 0.2$. Without added noise, the true CSI element is $h = -0.5 + j0.2$. The noise covariance matrix is given by $\mathbf{\Sigma} = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.2 \end{pmatrix}$. With the addition of AWGN, the joint density function for h is illustrated in Figure 3.2. Note that Figure 3.2b is the view of Figure 3.2a from a vector orthogonal to the real and imaginary axes in the complex plane. The peak of the joint probability distribution in Figure 3.2a and center of the circle in Figure 3.2b is the point (a, b).



(a) (b)

Figure 3.2. Joint density function and threshold for authentication.

To authenticate, u and v must jointly fall within the region illustrated by the dotted line in Figures 3.2a and 3.2b. The dotted line is a circularly-shaped threshold with radius z centered on (a, b) where z is directly proportional to $\sqrt{0.2}$, the average eigenvalue of $\mathbf{\Sigma}$. We see that if $a - z \leq u \leq a + z$ then v is dependent on u, a, and b.

The transmitter must satisfy $\mathcal{H}_0$ for every CSI element as we saw with the red-shaded markers in Figure 3.1. The probability for authentication in a MIMO channel with M transmit antennas and N receive antennas is given by

$$\begin{aligned} &\prod_{m=1}^{M}\prod_{n=1}^{N} P([u_{n,m} + jv_{n,m}] \in \mathcal{D}_{n,m}), \\ &\mathcal{D}_{n,m} = \{(u_{n,m}, v_{n,m}) \mid \left(u_{n,m} - a_{n,m}\right)^2 \\ &\qquad + \left(v_{n,m} - b_{n,m}\right)^2 \leq z^2\}. \end{aligned} \tag{3.6}$$

Simulating Equations 3.5 and 3.6 with $a_{n,m}, b_{n,m}, u$, and v all distributed as $\mathcal{N}(0, 0.5)$, Figure 3.3 illustrates how likely an accidental authentication will be as the number of antenna elements of the receiver and transmitter are increased and the threshold is reduced.

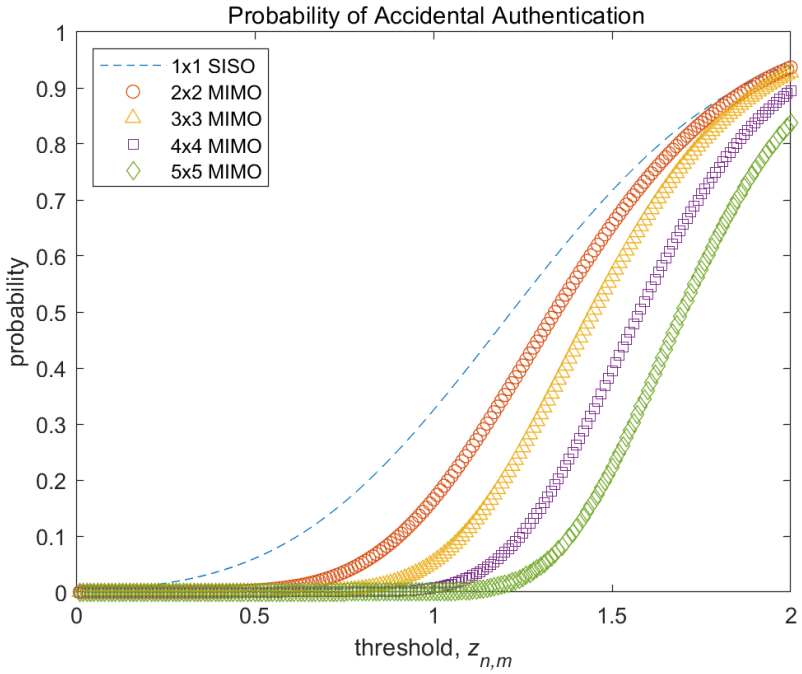


Figure 3.3. Probability of authentication for various MIMO configurations and thresholds. Source: [87].

Fixing the threshold, we see that as transmitter and receiver antennas are increased, the probability of accidental authentication decreases. Fixing the number of transmit and receive antennas, decreasing the threshold also decreases the chance that a transmitter will be accidentally authenticated.

To implement this authentication scheme and determine which hypothesis $\hat{h}_{n,m,k}$ satisfies, we require advance knowledge of the noise power our receiver imparts to $\boldsymbol{H}$ to determine z, and that may change over time and be different among devices. Instead, we will allow a neural network to implicitly determine the threshold and perform the authentication decision. We created a GAN that is trained on authentic samples from a dataset and samples produced by a generative model. The discriminative model then learned the characteristics of $h_{n,m}$ and $\boldsymbol{\Sigma}_{\bar{\epsilon}_{n,m}}$. Following training, two testing datasets validated the performance of the discriminative model to accurately distinguish $\boldsymbol{H}$ between trusted and untrusted transmitters.

3.3 Adversarial System Model

We consider a wireless MIMO communications channel with trusted users and untrusted users, some of the latter group who are malicious adversaries. The adversaries have resources available to change their transmitter antenna characteristics, RF path timing, output power, and/or present reflectors between themselves and the receiver. Thus, they are able to change their CSI as measured by the receiver. The adversaries likely do not have perfect knowledge of what the legitimate CSI should be at the receiver, however we assume they have advanced knowledge of the environment and have the ability to conduct surveys between potential transmitter and receiver locations in order to know the received CSI needed to authenticate. Just as an adversary may be able to authenticate by discovering cryptographic or other credentials in a traditional wireless system, we allow the nefarious actors additional resources to spoof our system. To defeat this scenario, the discriminative model at the receiver is adversarially trained by a generative model that creates authentic looking CSI samples. By training with increasingly high-quality spoofed samples, the discriminative network learns the features of transmitters that should be authenticated and the features of those that should not be authenticated.

3.3.1 GAN Architecture

The adversarial competition in the GAN is a minimax game where the discriminative model attempts to correctly label training samples from a distribution produced by CSI matrix elements $p_{data}(h_{n,m})$ and fake training samples created by the generator. The discriminative model is trained to maximize the probability of assigning the correct label, while the generative model is trained to minimize the same probability. The value function that describes this relationships from the original work by Goodfellow [83] is given by Equation 1.20.

As each entity adversarially trains each other, they learn to improve their individual performance. When the discriminative model correctly identifies fake samples created by the generative model, the generative network will update its parameter weights through backpropagation to make more realistic samples. Likewise, the discriminative model will update its parameter weights when it incorrectly identifies real or fake samples. The results of this training are a generator neural network adept at creating data that closely mimics training data and a discriminator neural network that can identify all but the best fakes.

3.3.2 Discriminative Model

The discriminator estimates the probability that a sample came from the training data, rather than the generator. When training begins, the discriminator won't know $p_{data}(h_{n,m})$, so the accuracy of correctly assigning authentic and fake samples will be near 0.5. The accuracy will increase with more iterations of samples and backpropagation as the authentic data distribution is learned until the generator network creates samples such that the fake sample distribution $p_g(z)$ optimally matches $p_{data}(h_{n,m})$. At this point, the accuracy of correctly assigning authentic and fake samples will return to 0.5 since for the optimal discriminator D^*, and fixed generator G, $D^*_G(x) = \frac{p_{data}(h_{n,m})}{p_{data}(h_{n,m})+p_z(z)}$. When $p_{data}(h_{n,m}) = p_z(z)$, $D^*_G(x) = 0.5$ [85].

3.3.3 Generative Model

Without having direct access to $p_{data}(h_{n,m})$, the generator attempts to capture this distribution through feedback based on the probabilities the discriminator assigns to generated fake samples [85]. The weights of the generator network are updated via the loss function $J^{(G)}$ so that the generator will create better samples.

3.4 Simulation

To simulate the adversarial system model, we create a GAN to process a single subcarrier in a MIMO 4×4 configuration. Therefore, the discriminative model has 16 complex inputs and one real output, while the generative model has one real input and 16 complex outputs. The inputs for the discriminative model and the outputs for the generative model represent the complex elements in the CSI matrix.

3.4.1 GAN development

The GAN is implemented using the Python programming language, Keras [144] front-end, and Tensorflow [145] back-end. Additionally, Numpy, Pandas, and Matplotlib Python libraries were used. The overall GAN design is summarized in Table 3.1, with a total of 9,057 parameters. The file size of the discriminator network was 104 KB.

Table 3.1. GAN architecture. Source: [87].

Discriminator:

Layer	output size	activation
Input 1: $x \sim p_{data}(h_{1,1})$	2	
Input 2: $x \sim p_{data}(h_{1,2})$	2	
⋮	⋮	
Input 16: $x \sim p_{data}(h_{4,4})$	2	
Concatenated	32	
Fully connected	64	LeakyReLU (alpha = 0.3)
Dropout = 0.2		
Fully connected	32	LeakyReLU (alpha = 0.3)
Dropout = 0.2		
Output	1	sigmoid

Generator:

Layer	output size	activation
Input: $z \sim p_z(z)$	5	
Fully connected	16	LeakyReLU (alpha = 0.3)
Fully connected	32	LeakyReLU (alpha = 0.3)
Fully connected	64	tanh
Output 1	2	linear
Output 2	2	linear
⋮	⋮	⋮
Output 16	2	linear

The discriminator network $\mathcal{D}$ has 16 inputs of size 2 merged into one *concatenated* layer. Each of the 16 inputs is a vector of size 2 to accommodate the real and imaginary parts of the complex CSI matrix element. The *concatenated* layer merges the input vectors, resulting in a vector of size 32. Two additional fully connected layers of size 64 and 32 with LeakyReLU activations (*alpha* = 0.3) follow. Both of these hidden layers use dropout of 0.2 to prevent overfitting. The output layer of size one is fully connected and uses a sigmoid activation to provide values (0.0, 1.0). The learning rate for $\mathcal{D}$ was 0.0003 using the Adam [134] optimizer.

The generator network $\mathcal{G}$ has a single input with five neurons fully connected to the first hidden layer of size 16. Two additional hidden layers of sizes 32 and 64 are again fully connected using LeakyReLU (*alpha* = 0.3) and tanh activations respectively. Finally, 16

output layers of size 2 are connected using *linear* activations. The learning rate for $\mathcal{G}$ was 0.0009 using the Adam optimizer.

3.4.2 Datasets

A master dataset was created by adding measurement error in the form of AWGN across a range of SNR levels to a single 4×4 CSI matrix composed of 16 circularly symmetric Gaussian complex values with zero mean, and unit variance $\mathcal{CN}(0, 1)$. The SNR values ranged from 0 dB to 30 dB in steps of 2 dB, and 1,000 samples were created at each SNR level. Each sample is a 4×4 complex matrix.

Splitting evenly across SNR levels, the training dataset uses 70% of the master dataset samples, reserving 30% for the testing dataset. Two testing datasets were created, each consisting of 700 samples for each SNR value. In addition to the 300 valid samples taken from the master dataset, 400 more testing samples were created to simulate two different operating scenarios.

The first testing dataset replicated the accidental authentication case. There are six transmitters, one of which should be authenticated. The 300 samples taken from the master dataset represent the transmitter that should be authenticated. For the remaining transmitters, five new 4×4 CSI matrices with elements taken from $\mathcal{CN}(0, 1)$ were created. To each matrix, AWGN at varying SNR values was added to produce 80 samples at each SNR value. These samples were then added to the accidental authentication dataset, resulting in 300 legitimate samples and 400 illegitimate samples. We will refer to this dataset as the accidental authentication test dataset.

The second testing dataset emulated five nefarious users attempting to authenticate by matching the CSI matrix of a single legitimate transmitter. If by some unlikely method, an adversary were able to know the channel characteristics between two legitimately authenticated transmitters, such as described by Shi et al. in [43], the adversary may also have the resources necessary to spoof their transmitted CSI to appear as the received CSI from another transmitter. As before, 300 test samples from the master dataset for each of the 16 SNR levels ranging from 0 dB to 30 dB in steps of 2 dB are the legitimate samples. To complete this dataset, five different complex number offsets were added to the original legitimate CSI matrix. The offsets values of $0.2 + j0.2$, $0.2 - j0.2$, $-0.2 + j0.2$, $0.5 + j0.5$,

and $-0.5 + j0.2$ were selected to loosely surround the legitimate samples. Samples were created as previously described by the addition of AWGN, again resulting in 300 legitimate samples and 400 illegitimate samples. We will refer to this dataset as the nefarious users test dataset.

Training was restricted to a maximum of 50 epochs in mini-batches of 64 samples. An example of samples from the training, accidental test dataset, and nefarious user test dataset are shown in Figure 3.4, where blue dots correspond to the 16 clusters of legitimate samples, and the orange circles represent illegitimate samples that should not be authenticated. Note that in Figure 3.4, only one example from each of the illegitimate test group is shown. The accidental authentication dataset in Figure 3.4a depicts in orange one of five clusters of CSI elements that should be denied authentication. Likewise, in Figure 3.4b, the orange clusters have an offset of $0.2 + j0.2$ from CSI elements that should be authenticated. Not included in Figure 3.4 are the four additional randomly selected CSI matrices in the accidental authentication test dataset or the four additional CSI matrices created with different offset values from the legitimate CSI in the nefarious users test dataset.

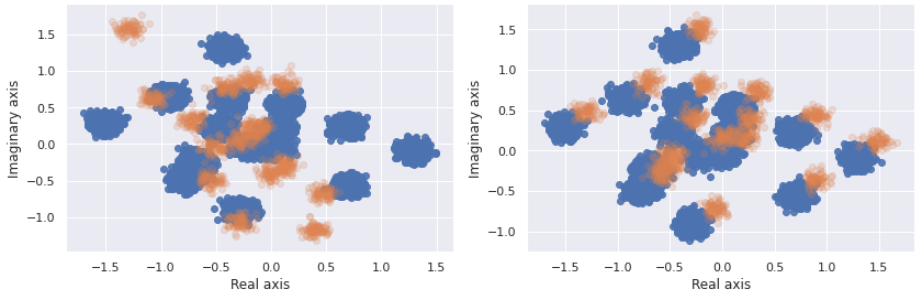


Figure 3.4. Samples from the (a) accidental authentication and (b) nefarious users test datasets at 20 dB SNR.

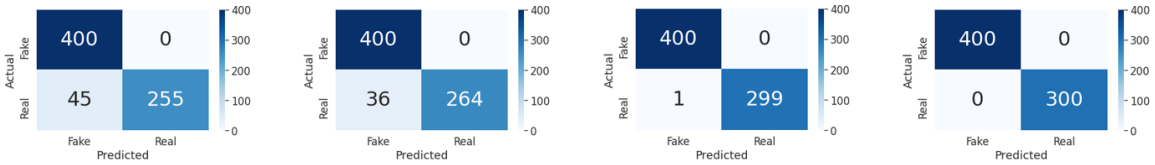
3.5 Accidental Authentication Dataset Results

In this section, we present the results of the simulation using the accidental authentication dataset. We examine the performance of the GAN-trained discriminator, the hypothesis test, and three one-class machine learning algorithms. To illustrate performance, we use

confusion matrices to show how the samples are assigned from the test datasets. The horizontal axis is the predicted result for whether or not authentication should be granted. If the sample is assessed to be legitimate, the sample is tallied in the "Real" column. If the predicted result is that the sample is illegitimate and should not be authenticated, the sample is allocated to the "Fake" column. The vertical axis is the ground truth of the sample from the dataset, where legitimate samples are assigned to the "Real" row, and illegitimate samples are shown in the "Fake" row.

3.5.1 GAN Discriminator Results

Against the accidental authentication testing dataset, the discriminator achieves 100% accuracy for SNR greater than or equal to 10 dB. As shown in Figure 3.5, for SNR less than 10 dB, the discriminator makes errors in correctly identifying legitimate samples but never allows illegitimate samples to be authenticated. The generator was trained to create samples that mimic the legitimate CSI from the training dataset and the illegitimate CSI matrices in the accidental authentication testing dataset were not created with prior knowledge of what CSI values should be used to authenticate.

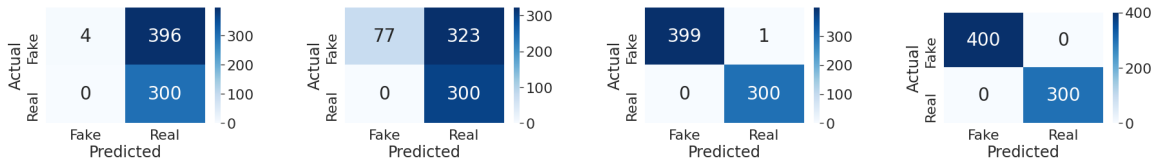


(a) SNR = 0 dB (b) SNR = 4 dB (c) SNR = 8 dB (d) SNR = 10 dB

Figure 3.5. GAN discriminator performance against accidental authentication test dataset with SNR levels at (a) 0 dB, (b) 4 dB, (c) 8 dB, (d) 10 dB.

3.5.2 Hypothesis Test Results

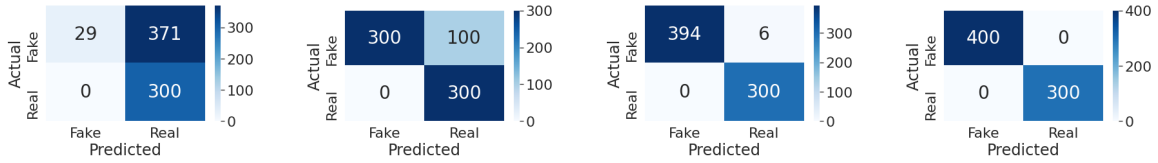
Using the hypothesis test with $z = 5\lambda^{\frac{1}{2}}$, we also achieve 100% accuracy when SNR is greater than or equal to 10 dB. However, unlike the GAN, the hypothesis test incorrectly labeled "Fake" samples for "Real" as we can see with non-zero values in the upper right quadrants in the confusion matrices shown in Figure 3.6. If implemented, this would allow illegitimate transmitters to authenticate.



(a) SNR = 0 dB (b) SNR = 4 dB (c) SNR = 8 dB (d) SNR = 10 dB

Figure 3.6. Hypothesis performance against accidental authentication test dataset with $z = 5\lambda^{\frac{1}{2}}$ and SNR levels at (a) 0 dB, (b) 4 dB, (c) 8 dB, (d) 10 dB.

Because illegitimate transmitters are being authenticated, we need to restrict the threshold. Lowering the threshold to $z = 4\lambda^{\frac{1}{2}}$, and $z = 3\lambda^{\frac{1}{2}}$, we are able to reduce the number of incorrectly classified "Fake" samples. Figure 3.7 shows the confusion matrices based on using the hypothesis test with $z = 4\lambda^{\frac{1}{2}}$. We see in the upper right quadrant of each confusion matrix that the number of illegitimate samples is reduced as compared with $z = 5\lambda^{\frac{1}{2}}$ shown in Figure 3.6. Also, note the change in performance with respect to SNR. For $z = 4\lambda^{\frac{1}{2}}$, the hypothesis test is more accurate for lower SNR levels, reaching 100% at 8 dB and greater.

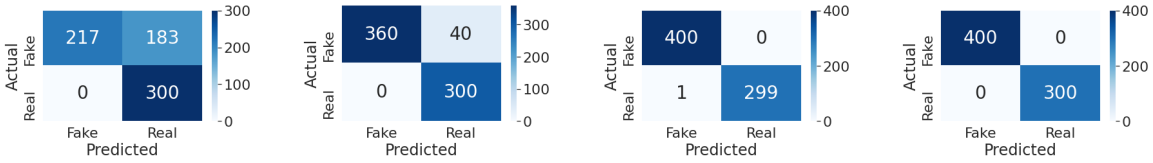


(a) SNR = 0 dB (b) SNR = 4 dB (c) SNR = 6 dB (d) SNR = 8 dB

Figure 3.7. Hypothesis performance against accidental authentication test dataset with $z = 4\lambda^{\frac{1}{2}}$ and SNR levels at (a) 0 dB, (b) 4 dB, (c) 6 dB, (d) 8 dB.

Reducing the threshold again to $z = 3\lambda^{\frac{1}{2}}$, we again see improvement using the hypothesis test against the accidental authentication test data set as shown in Figure 3.8. With this tighter threshold, we reach 100% accuracy at 6 dB SNR or greater, however, at 4 dB SNR, there is one legitimate sample that was incorrectly labeled as "Fake" in Figure 3.8c. As SNR increases, the cluster size of CSI elements decreases while z remains fixed, reducing the probability that higher SNR received CSI measurements will exceed the threshold. Even

with $z = 3\lambda^{\frac{1}{2}}$, there are illegitimate samples categorized as "Real" using the hypothesis test unlike using the GAN discriminator.

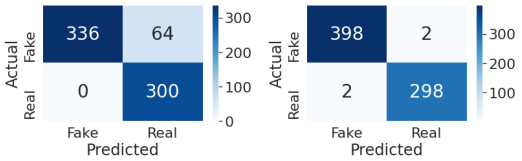


(a) SNR = 0 dB (b) SNR = 2 dB (c) SNR = 4 dB (d) SNR = 6 dB

Figure 3.8. Hypothesis performance against accidental authentication test dataset with $z = 3\lambda^{\frac{1}{2}}$ and SNR levels at (a) 0 dB, (b) 2 dB, (c) 4 dB, (d) 6 dB.

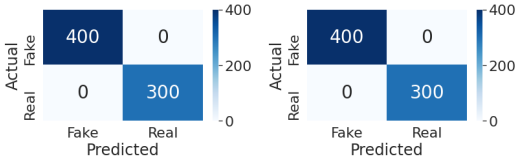
3.5.3 One-Class Machine Learning Algorithm Results

In addition to comparing the GAN discriminator performance against the hypothesis test with a variety of thresholds, we also explore the use of three different machine learning algorithms. Because the master dataset does not contain illegitimate samples, the techniques we use are limited to one-class, novelty, or anomaly detection algorithms. We employed LOF, iForest, and OC-SVM. These techniques are available and were implemented from the Scikit-learn project [62]. These algorithms used the same training and testing datasets previously used with the GAN. Figures 3.9, 3.10, and 3.11 contain the confusion matrices of the one-class machine learning algorithms against the accidental authentication testing dataset.



(a) SNR = 0 dB

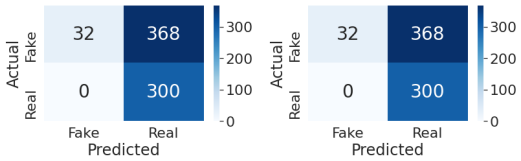
(b) SNR = 2 dB



(c) SNR = 4 dB

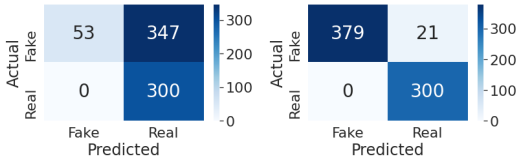
(d) SNR = 30 dB

Figure 3.9. LOF confusion matrices against accidental authentication test dataset with SNR levels at (a) 0 dB, (b) 2 dB, (c) 4 dB, (d) 30 dB.



(a) SNR = 0 dB

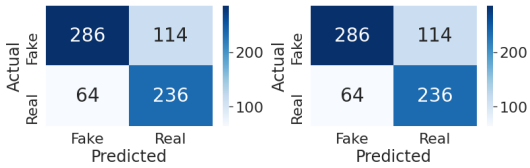
(b) SNR = 2 dB



(c) SNR = 4 dB

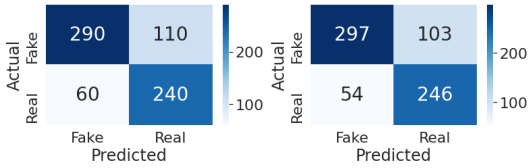
(d) SNR = 30 dB

Figure 3.10. iForest confusion matrices against accidental authentication test dataset with SNR levels at (a) 0 dB, (b) 2 dB, (c) 4 dB, (d) 30 dB.



(a) SNR = 0 dB

(b) SNR = 2 dB



(c) SNR = 4 dB

(d) SNR = 30 dB

Figure 3.11. OC-SVM confusion matrices against accidental authentication test dataset with SNR levels at (a) 0 dB, (b) 2 dB, (c) 4 dB, (d) 30 dB.

In Figure 3.9, we see that the LOF algorithm achieves 100% accuracy at 4 dB and greater SNR. However, below 4 dB, the LOF does categorize some "Fake" samples as "Real", similar to the hypothesis test results with larger threshold values. The iForest and OC-SVM algorithms results against the accidental authentication dataset are shown in Figures 3.10

and 3.11, respectively. Based on the confusion matrices, these algorithms never reach 100% accuracy for any SNR. The iForest results in Figure 3.10 indicate that legitimate users will always be authenticated, based on zero being the value in the lower left quadrant for all SNR values. This is similar to the hypothesis test results with $z = 5\lambda^{\frac{1}{2}}$ and $z = 4\lambda^{\frac{1}{2}}$ shown in Figures 3.6 and 3.7 where the threshold level was too large to achieve high authentication accuracy. The OC-SVM results indicate a relatively flat performance regardless of SNR. Comparing the results with respect to the same quadrants across Figures 3.11a, 3.11b, 3.11c, and 3.11d, we see similar values, suggesting SNR has less impact to OC-SVM as compared to the previous machine learning algorithms. When compared to the other one-class machine learning algorithms, we also note that OC-SVM resulted in a relatively large percentage of legitimate samples being labeled as "Fake", as shown with non-zero values in the lower left quadrants of the confusion matrices in Figure 3.11.

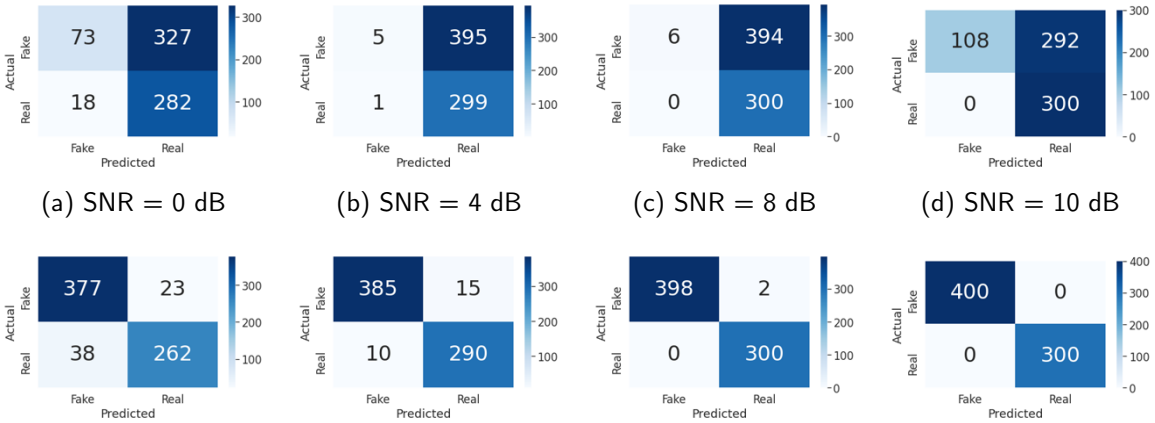
3.6 Nefarious User Dataset Results

In this section, we analyze the results of the GAN, the hypothesis test, and the same one-class machine learning algorithms from the previous section against the nefarious users dataset. The positions of the CSI elements for the nefarious users are distributed much closer to the CSI elements received from the authentic transmitter as illustrated in Figure 3.4, and it should be expected that there will be more difficulty in discerning "Real" from "Fake" at the physical layer with low SNR levels.

3.6.1 GAN Discriminator Results

For the nefarious user testing dataset, the discriminator doesn't achieve 100% until the SNR reaches 20 dB. As shown in Figure 3.12, in addition to mischaracterizing legitimate samples, the discriminator allows some number of illegitimate samples to be authenticated, since there are samples counted in the "Fake" row and "Real" column. This is unlike the case with the accidental authentication dataset where the GAN would not have allowed illegitimate transmitters to authenticate even when the overall accuracy was still below 100%. It appears that during training, the discriminator establishes regions where the legitimate samples are likely to be found. During testing, some of the nefarious user samples are within those regions at low SNR values. This is unlikely to be the case with the accidental authentication testing dataset as those samples were created with elements drawn from $\mathcal{CN}(0, 1)$, whereas the

nefarious users dataset was created with predetermined offsets from the legitimate samples. As the SNR increases, the region the discriminator establishes can become smaller, and is able to correctly distinguish the legitimate samples from the illegitimate samples.



(a) SNR = 0 dB (b) SNR = 4 dB (c) SNR = 8 dB (d) SNR = 10 dB

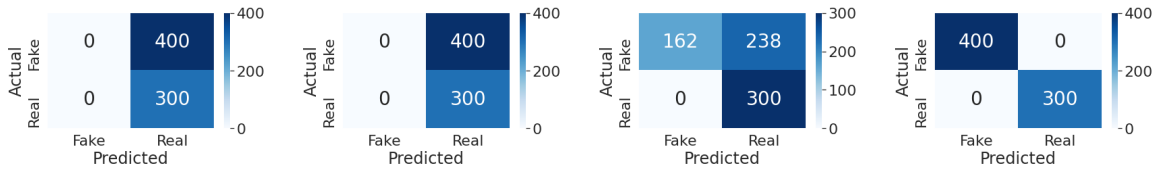
(e) SNR = 14 dB (f) SNR = 16 dB (g) SNR = 18 dB (h) SNR = 20 dB

Figure 3.12. GAN discriminator performance against nefarious users test dataset with SNR levels at (a) 0 dB, (b) 4 dB, (c) 8 dB, (d) 10 dB, (e) 14 dB, (f) 16 dB, (g) 18 dB, (h) 20 dB.

3.6.2 Hypothesis Test Results

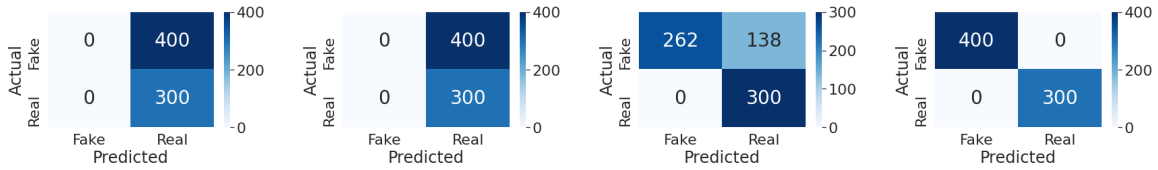
Like the discriminator performance, the hypothesis test with $z = 5\lambda^{\frac{1}{2}}$ reaches 100% accuracy for higher SNR values than was required with the accidental authentication dataset. An accuracy of 100% was reached at 26 dB. Because the mean of each of the received illegitimate CSI elements are close in proximity to the mean of the received legitimate CSI elements, the threshold $z = 5\lambda^{\frac{1}{2}}$ is too large to discern between the two categories. Figure 3.13 shows the confusion matrices for $z = 5\lambda^{\frac{1}{2}}$ with SNR equal to 0 dB, 10 dB, 20 dB, and 26 dB. With SNR at 10 dB or less, all the samples are classified as legitimate, incorrectly allowing 400 illegitimate samples to authenticate.

As before, we reduced the tolerance z to $4\lambda^{\frac{1}{2}}$ and then $3\lambda^{\frac{1}{2}}$, resulting in the respective confusion matrices shown in Figure 3.14 and Figure 3.15. With $z = 4\lambda^{\frac{1}{2}}$, all the sample are categorized as legitimate when the SNR is 8 dB or less.



(a) SNR = 0 dB (b) SNR = 10 dB (c) SNR = 20 dB (d) SNR = 26 dB

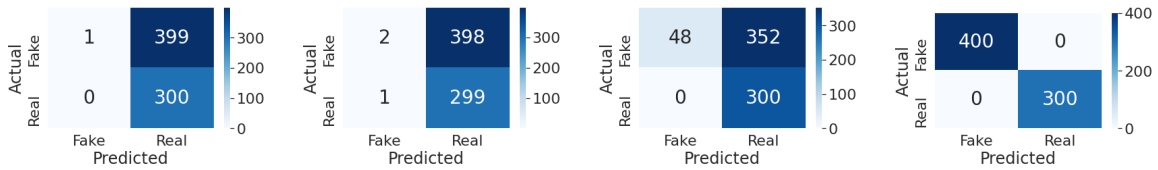
Figure 3.13. Hypothesis test performance against nefarious users test dataset with $z = 5\lambda^{\frac{1}{2}}$ and SNR levels at (a) 0 dB, (b) 10 dB, (c) 20 dB, (d) 26 dB.



(a) SNR = 0 dB (b) SNR = 8 dB (c) SNR = 20 dB (d) SNR = 24 dB

Figure 3.14. Hypothesis test performance against nefarious users test dataset with $z = 4\lambda^{\frac{1}{2}}$ and SNR levels at (a) 0 dB, (b) 8 dB, (c) 20 dB, (d) 24 dB.

By constricting z, we can better filter illegitimate samples if they are far enough away from the predefined threshold. With the nefarious user dataset, we see that the hypothesis test is able to filter the illegitimate samples, but only when z is small enough and the SNR reaches a certain level. It is rare for legitimate samples to be identified as illegitimate, but we see that it can happen when we set z to low enough value such as $z = 3\lambda^{\frac{1}{2}}$ as shown in Figure 3.15b. At the same threshold, we also saw a legitimate sample categorized as illegitimate with the accidental authentication dataset in Figure 3.8c.

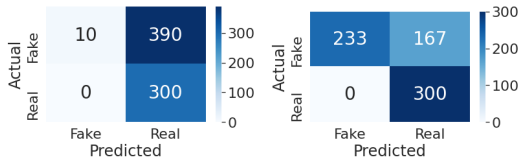


(a) SNR = 0 dB (b) SNR = 4 dB (c) SNR = 8 dB (d) SNR = 20 dB

Figure 3.15. Hypothesis test performance against nefarious users test dataset with $z = 3\lambda^{\frac{1}{2}}$ and SNR levels at (a) 0 dB, (b) 4 dB, (c) 8 dB, (d) 20 dB.

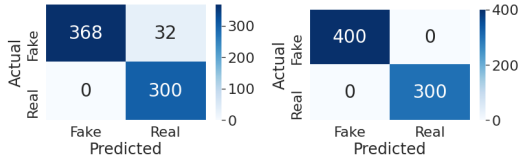
3.6.3 One-Class Machine Learning Algorithms

Finally, we trained and then tested the LOF, iForest, and OC-SVM algorithms against the nefarious users dataset. Figures 3.16, 3.17, and 3.18 show the respective confusion matrices for LOF, iForest, and OC-SVM against the nefarious users dataset. Similar to the discriminator and hypothesis test results, we see that the one-class machine learning algorithms required larger SNR to achieve similar authentication accuracy when compared to the accidental authentication dataset results. Neither the iForest nor the OC-SVM algorithms reached 100% accuracy using the nefarious users test dataset, however, the LOF algorithm achieved perfect authentication accuracy at SNR equal to and greater than 16 dB.



(a) SNR = 0 dB

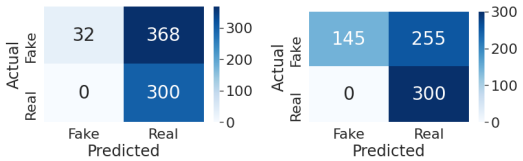
(b) SNR = 12 dB



(c) SNR = 14 dB

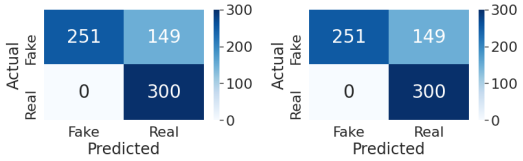
(d) SNR = 16 dB

Figure 3.16. LOF confusion matrices against nefarious users test dataset with SNR levels at (a) 0 dB, (b) 12 dB, (c) 14 dB, (d) 16 dB.



(a) SNR = 0 dB

(b) SNR = 14 dB



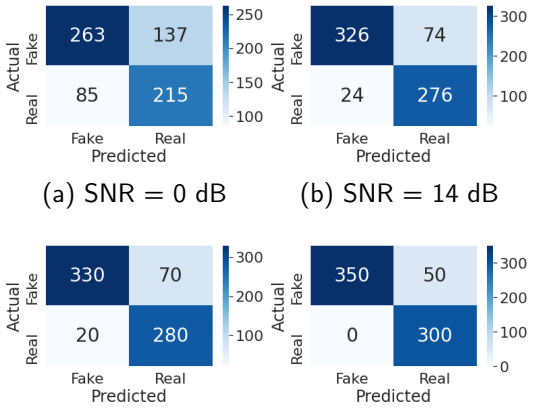
(c) SNR = 16 dB

(d) SNR = 30 dB

Figure 3.17. iForest confusion matrices against nefarious users test dataset with SNR levels at (a) 0 dB, (b) 14 dB, (c) 16 dB, (d) 30 dB.

In Figures 3.16 and 3.17, we see that the LOF and iForest algorithms had similar performance for legitimate samples against the nefarious users datasets as compared to the legitimate samples against the accidental authentication case shown in Figures 3.9 and 3.10. That is, the LOF and iForest algorithms both correctly classified all the legitimate samples as "Real" regardless of SNR. As SNR was increased, these algorithms improved their accuracy against the illegitimate samples, and maintain their accuracy with the legitimate samples.

At higher SNR values, the OC-SVM algorithm improved in accuracy using the nefarious users testing dataset when compared to the accidental authentication dataset. In Figure 3.18



(a) SNR = 0 dB (b) SNR = 14 dB

(c) SNR = 16 dB (d) SNR = 30 dB

Figure 3.18. OC-SVM confusion matrices against nefarious users test dataset with SNR levels at (a) 0 dB, (b) 14 dB, (c) 16 dB, (d) 30 dB.

we see that as the SNR increased from 0 dB to 16 dB, the OC-SVM algorithm accuracy increased at identifying legitimate samples and correctly classified all the legitimate samples for SNR greater than or equal to 16 dB as seen in Figure 3.18c. This is in contrast to the OC-SVM performance against the accidental authentication dataset as we saw in Figure 3.11 where the OC-SVM algorithm accuracy was roughly the same from 0 dB to 30 dB SNR. The reason for this may be explained by the approach used in the SVM algorithm briefly mentioned in Section 1.5.1 where a hyperplane is used to separate different classes. For the single-class case, the OC-SVM algorithm attempts to create a small region in which the training data is contained. The illegitimate samples from the accidental authentication dataset were drawn from the same distribution as the legitimate samples, and the nefarious dataset illegitimate samples were created by applying offsets to the legitimate samples. The regions created by the OC-SVM algorithm based on the training data likely resulted in the testing data from the nefarious user dataset being more linearly separable than the accidental authentication testing dataset.

3.7 Accuracy Comparison of Physical-Layer Authentication Techniques

For the accidental authentication test dataset, Figure 3.19 shows the accuracy performance of the physical-layer authentication techniques we've explored in this chapter. We see that the LOF algorithm reaches 100% accuracy first and that the GAN and hypothesis test with $z = 3\lambda^{\frac{1}{2}}$ are very close in performance.

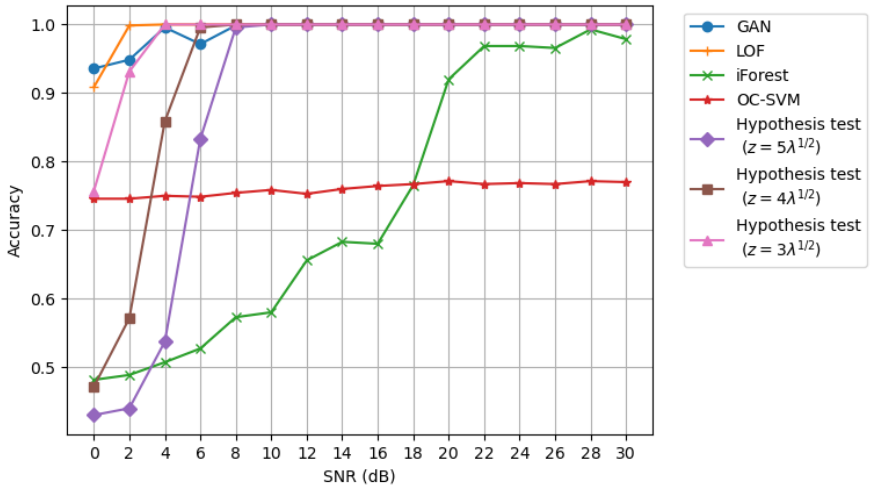


Figure 3.19. Accuracy vs SNR for accidental authentication dataset.

Figure 3.20 illustrates the accuracy of the machine learning techniques against the nefarious users test dataset. While, LOF is the first algorithm to reach 100% accuracy, its performance at low SNR values in this dataset is outmatched by the OC-SVM. The GAN performance again trails that of the LOF algorithm. We noted that all techniques including the GAN, as we've seen in Fig. 3.12, incorrectly categorize illegitimate samples as "Real" for low SNR against the nefarious users who are able to closely match the CSI of legitimate transmitters.

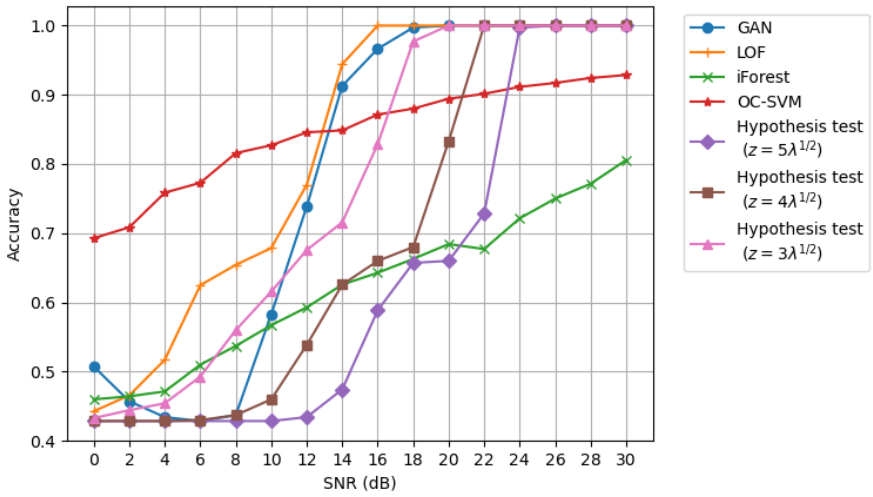


Figure 3.20. Accuracy performance vs SNR for nefarious users dataset.

In the case where there is only one transmitter to authenticate and a single carrier frequency is used in the MIMO system, the LOF is preferred due to the accuracy performance and the increased amount of complexity in training the GAN-based discriminator. For the situation where nefarious users may attempt to emulate the lone legitimate transmitter CSI, we see from Figures 3.19 and 3.20 that additional SNR is required to maintain 100% authentication.

Another benefit to using the LOF algorithm is its relative low complexity as compared to the GAN resulting in less time needed to train. The one-class machine learning algorithms all trained on the order of seconds, while the GAN required minutes to train. However, the drawback to the one-class machine learning algorithms are that their implementations in [62] are limited to two dimensions. Contrast this limitation with neural networks that are easily adapted to multi-dimensional input and are well suited for such tasks [66].

3.8 Summary

In this chapter, we showed how CSI could be used as a method to provide physical-layer authentication. Our analysis illustrated that the probability of accidentally authenticating other transmitters decreases as the number of receive and transmit antennas are increased and a threshold value is judiciously applied.

We created a hypothesis test based on a threshold determined by the amount of receiver noise added to the received CSI. We then developed a GAN that was trained on a dataset of CSI matrices to perform physical-layer authentication in an adversarial environment. The performance of the GAN-trained discriminative model indicates the viability of using a GAN for physical-layer authentication using CSI. We then implemented the LOF, iForest, and OC-SVM one-class machine learning algorithms. Across all levels of AWGN, the LOF algorithm was best at reaching 100% accuracy.

The LOF algorithm was not only the most accurate for all SNR, but it was also much faster to train than the GAN. However, the anomaly detection capability provided by LOF is limited to two dimensional inputs, whereas neural networks can support samples that have multiple dimensions.

Although the GAN was constructed using best practices [84], [129], [133], adjustment of hyperparameters and training methodology for the GAN can be applied to improve results. For example, by increasing the mini-batch size from 64 to 128, we saw an improvement in identifying the illegitimate samples. Unfortunately, this also caused a reduction in the number of legitimate samples identified. Changes to the original GAN architecture in this work is left for future refinement.

In this chapter, we had one authentic transmitter and several illegitimate transmitters. In the next chapter, we will extend our GAN-based system to classify a number of authentic transmitters.

CHAPTER 4:
Multitransmitter Classification

In this chapter, the goal is to correctly classify several different transmitters through received CSI. A system that uses wireless communications may have multiple layers of access available. By classifying trusted transmitters, we can also decide what level of access to provide to a device. Just as a user account shouldn't be granted full admin privileges without additional authentication, we can stratify permissions at the physical layer following classification. In Chapter 3, we illustrated how we can classify trusted and untrusted transmitters to make an authentication decision. Here, we improve the authentication decision by conducting fine-grained classification of trusted transmitters instead of merely assigning a label of "trusted" or "untrusted", or "Real", or "Fake" to all transmitters.

This chapter includes material adapted from work published by the author. This revised material is from "Multi-Transmitter Physical Layer Authentication Using Channel State Information and Deep Learning" by Ken St. Germain and Frank Kragh, published in the 14th International Conference on Signal Processing and Communication Systems [98].

4.1 System Model

We consider a 4×4 wireless MIMO communications channel with 10 trusted transmitters that should be authenticated and 10 illegitimate "Eve" transmitters that should not be authenticated. The channel model used is the same as described in Chapter 3, where each trusted transmitter has CSI elements drawn from $\mathcal{CN}(0, \mathbf{\Sigma}_\epsilon)$. To these elements, AWGN is added to simulate SNR environments from -10 dB to 30 dB.

Once trained, the SGAN discriminator will be used to differentiate the received CSI as either "Real" or "Fake". The received CSI samples that are classified as "Fake" will be denied authentication. If the discriminator classifies a sample as "Real", the transmitter will be authenticated, and subsequently categorized by the classifier network.

4.2 SGAN Architecture

The discriminative model at the receiver is adversarially trained by a generative model $\mathcal{G}$ that creates CSI samples that appear authentic. By training with increasingly high-quality spoofed samples, the discriminative network $\mathcal{D}$ learns the features of transmitters that should be authenticated and the features of those that should not be authenticated. Parallel to the adversarial training between $\mathcal{D}$ and $\mathcal{G}$, the classifier C learns the correct labels assigned to the 10 trusted users.

Once the transmitter is classified as trusted or untrusted, the trusted transmitters can be authenticated at the appropriate level. Here, we will use an SGAN to classify several transmitters for authentication at various SNR levels.

The adversarial competition in the SGAN is a minimax game described by Equation 1.20 where the discriminative model attempts to correctly identify authentic training samples from a distribution produced by CSI matrix elements $p_{data}(h_{n,m})$ and fake training samples created by the generator. A depiction of a SGAN in training is shown in Figure 1.8.

While $\mathcal{D}$ and $\mathcal{G}$ adversarially train each other, they learn to improve their individual performance. Additionally, C is trained on labeled samples from the training dataset. Although C does not directly receive unlabeled authentic or fake samples, the weights of C are affected since it shares weights with $\mathcal{D}$ in the $\mathcal{D}/C$ implementation.

The classifier shares all but the final activation layer with the discriminator and is trained to determine which of the 10 trusted transmitters will be authenticated. The weights of $\mathcal{D}/C$ are iteratively updated as $\mathcal{D}$ and C are trained.

4.3 Simulation

The SGAN processed a single subcarrier in a MIMO 4×4 configuration with 10 trusted transmitters. Therefore, the classifier model has 16 complex inputs and 1 real output for each transmitter label, the discriminative model has 16 complex inputs and 1 real output denoting trusted or untrusted, while the generative model has 1 real input and 16 complex outputs. The inputs for the discriminative and classifier model and the outputs for the generative model represent the complex elements in the CSI matrix. A dataset of 210,000 authentic samples was created, where each sample is a 4×4 complex matrix. For each of the 10 trusted

transmitters, there are 1,000 CSI samples at the 21 SNR levels between -10 dB and 30 dB inclusive, in steps of 2 dB.

4.3.1 SGAN development

The SGAN was implemented using the Python programming language, Keras [144] front-end, and Tensorflow [145] back-end. Additionally, Numpy, and Matplotlib Python libraries were used. The dataset was created using MATLAB [146] and Python. The overall SGAN design is summarized in Table 4.1, with a total of 9,722 parameters for the discriminator and the generator, and 4,890 parameters for the classifier. The file size of the classifier was 112 KB.

Table 4.1. SGAN dense architecture.

Discriminator/Classifier:

layer	output size	activation
Input 1: $x \sim p_{data}(x_{1,1})$	2	
Input 2: $x \sim p_{data}(x_{1,2})$	2	
⋮	2	
Input 16: $x \sim p_{data}(x_{4,4})$	2	
Concatenated	32	
Fully Connected	64	LeakyReLU (alpha = 0.3)
Dropout = 0.5		
Fully Connected	32	LeakyReLU (alpha = 0.3)
Dropout = 0.5		
Fully Connected	16	LeakyReLU (alpha = 0.3)
Fully Connected	10: $l_n = \{l_1, l_2, \ldots, l_{10}\}$	
Discriminator Output	1	sigmoid
Classifier Output	10	softmax

Generator:

layer	output	activation
Input: $z \sim p_z(z)$	5	
Fully Connected	16	LeakyReLU (alpha = 0.3)
Fully Connected	32	LeakyReLU (alpha = 0.3)
Fully Connected	64	tanh
Output 1	2	linear
Output 2	2	linear
⋮	2	
Output 16	2	linear

The discriminator/classifier network $\mathcal{D}/C$ is a dense or fully-connected DNN with 16 inputs

of size 2 merged into one *concatenated* layer. Each input has 2 neurons to accommodate the real and imaginary parts of the complex CSI matrix element. Four additional fully-connected layers of size 64, 32, and 16 with *LeakyReLU* activations (*alpha* = 0.3) follow. The first three of these hidden layers use dropout of 0.5 to prevent overfitting. Prior to the output layers, a fully connected layer of size 10 is used to capture the number of transmitters to be classified. The discriminator output layer of size 1 is fully connected and uses a sigmoid activation to provide values (0.0, 1.0). The classifier output is a *softmax* activation connected to the 10-neuron layer. The learning rate for $\mathcal{D}/C$ was 0.0002 using the *Adam* [134] optimizer and training was done with batches of 128 samples.

The generator network $\mathcal{G}$ has a single input with five neurons fully connected to the first hidden layer of size 16. Two additional hidden layers of sizes 32 and 64 are again fully connected using LeakyReLU (*alpha* = 0.3) and tanh activations respectively. Finally, 16 output layers of size 2 are connected using linear activations. The learning rate for $\mathcal{G}$ was 0.0002 using the Adam optimizer.

4.3.2 Datasets

The dataset consisted of 210,000 samples, of which 70% were randomly allocated for training, and the remaining 30% set aside for testing. Each sample started with a 4×4 matrix consisting of 16 circularly symmetric Gaussian complex values with zero mean, and unit variance $\mathcal{CN}(0, 1)$. Ten of these distinct 4×4 samples were generated, and for each of the ten samples, 1,000 samples of measurement error in the form of 21 different levels of SNR were generated. Simulating thermal noise in the receiver, decreasing amounts of AWGN were combined with the original signal to produce the 21 different levels of SNR ranging from -10 dB to 30 dB in steps of 2 dB. This was inspired by the technique used by O'Shea et al. to create distortion for the modulation classification task in [92], except we used MATLAB instead of Python and GNU Radio libraries.

Classifier Datasets

From the 147,000 training samples, 50 (0.034%) were randomly selected as labeled samples for the supervised training of the classifier. Care was taken to ensure that each class was equally represented in the supervised sample training dataset, however the selection of SNR values for the supervised samples was left to chance. The labels assigned to each of the

10 transmitters were $\{User0, User1, \ldots, User9\}$. These labels were then transformed to a one-hot vector for training and testing the classifier. The testing dataset for the classifier was 63,000 samples (30% of the original dataset).

Discriminator Datasets

In addition to the 30% of the original dataset samples that were set aside for testing, 21,000 more samples were added to complete the testing dataset for the discriminator. Using the same method to create the samples in the original dataset, the new 21,000 "Eve" samples are for testing the ability of the discriminator to discern legitimate from illegitimate samples. Ten 4×4 matrices consisting of 16 circularly symmetric Gaussian complex values with zero mean, and unit variance $\mathcal{CN}(0, 1)$, represent ten different transmitters that should not be authenticated. To these matrix elements, AWGN was combined, resulting in 100 samples for each "Eve" transmitter at every SNR value ranging from -10 dB to 30 dB in steps of 2 dB.

4.3.3 Additional networks

To compare the performance of the SGAN dense DNN classifier, we constructed three additional networks. Instead of training in an SGAN architecture, we first created a standalone dense DNN classifier. This classifier uses the same parameters as our SGAN dense classifier C. Next, we used another SGAN classifier, but we used convolutional layers instead of fully-connected layers. This gives us a SGAN CNN classifier. The layers for this are summarized in Table 4.2. Finally, we implemented a standalone CNN classifier, using the same parameters of the SGAN CNN classifier.

Table 4.2. SGAN CNN architecture.

Discriminator/Classifier:

layer	output shape	activation
Input 1: $x \sim p_{data}(x_{1,1})$	2x1	
Input 2: $x \sim p_{data}(x_{1,2})$	2x1	
⋮	2x1	
Input 16: $x \sim p_{data}(x_{4,4})$	2x1	
Concatenated	32x1	
Convolution	32x32	LeakyReLU (alpha = 0.3)
Dropout = 0.5		
Convolution	32x16	LeakyReLU (alpha = 0.3)
Dropout = 0.5		
Flatten	512	
Fully Connected	10 $l_n = \{l_1, l_2, \ldots, l_N\}$	
Discriminator Output	1	sigmoid
Classifier Output	10	softmax

Generator:

layer	output shape	activation
Input: $z \sim p_z(z)$	5	
Fully Connected	8	LeakyReLU (alpha = 0.3)
Reshape	8x1	LeakyReLU (alpha = 0.3)
Convolution	8x8	LeakyReLU (alpha = 0.3)
Convolution	7x16	tanh
Flatten	112	
Dense, Reshape	112x1	linear
Output 1	2x1	
Output 2	2x1	
⋮	2x1	
Output 16	2x1	

4.4 Results

The performance of the neural networks against the dataset of simulated CSI samples is presented here. The accuracy of the neural networks are assessed and compared.

4.4.1 Discriminator Results – 50 Labeled Samples

During testing, the densely-connected discriminator and CNN discriminator processed 63,000 samples from the original dataset that were not used in testing. All of these samples should be classified as “Real”. Additionally, the 21,000 “Eve” samples should be classified

as “Fake”. Figure 4.1 shows how accurate these discriminators were across the range of SNR values.

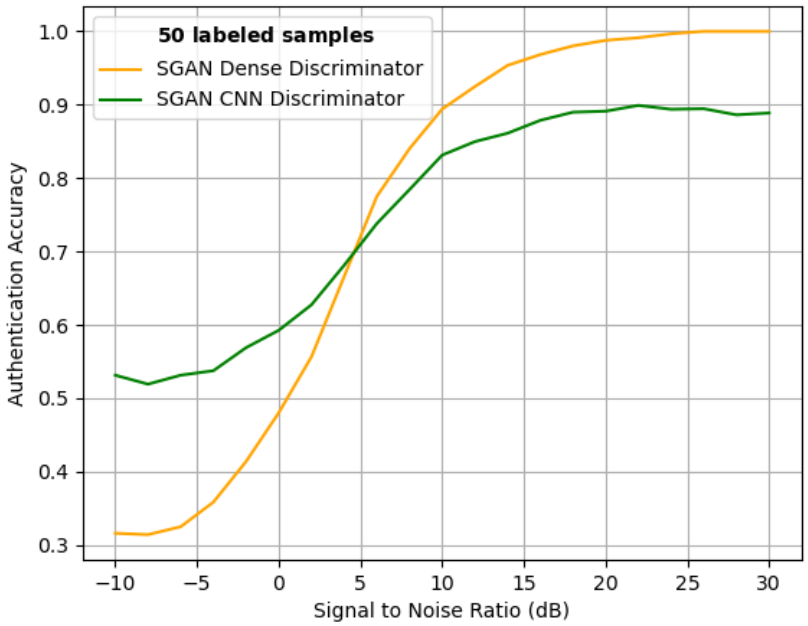
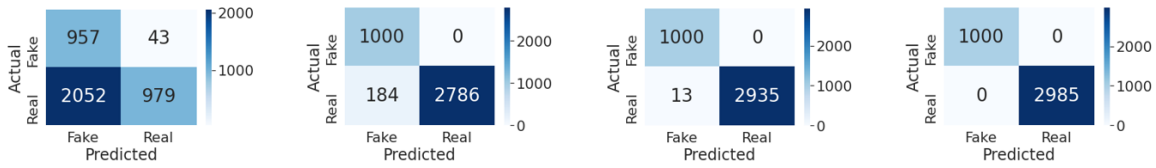


Figure 4.1. Discriminator accuracy performance vs SNR for 50 labeled samples.

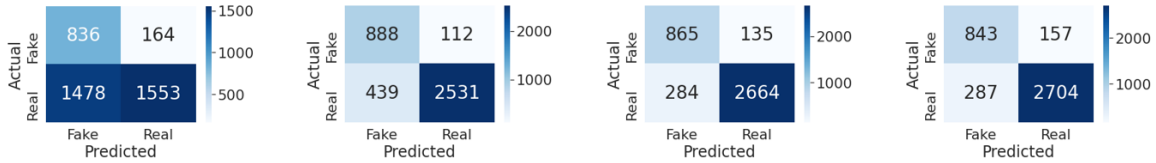
From Figure 4.1, we see the densely-connected discriminator has relatively low accuracy compared to the CNN discriminator at small SNR values, but is more accurate for SNR greater than 5 dB and eventually reaches 100% for SNR $\geq$ 26 dB.

We can use confusion matrices to better understand the discriminator output. Figure 4.2 and Figure 4.3 show the classification predictions of the densely-connected and CNN discriminators, respectively, across a range of SNR values.



(a) SNR = 0 dB (b) SNR = 14 dB (c) SNR = 24 dB (d) SNR = 26 dB

Figure 4.2. Densely-connected discriminator test predictions for SNR levels at (a) 0 dB, (b) 14 dB, (c) 24 dB, and (d) 26 dB after training with the 50 labeled samples dataset.



(a) SNR = 0 dB (b) SNR = 14 dB (c) SNR = 24 dB (d) SNR = 30 dB

Figure 4.3. CNN discriminator test predictions for SNR levels at (a) 0 dB, (b) 14 dB, (c) 24 dB, and (d) 30 dB after training with the 50 labeled samples dataset.

In Figure 4.2, we can see that as the SNR value increases, the densely-connected discriminator simultaneously improves in identifying "Real" samples as the number in the lower left quadrant decreases in value, and also in identifying "Fake" samples as the upper right quadrant diminishes to zero. Contrast with the behavior of the CNN discriminator, where the upper right quadrant does not decrease with larger SNR, and the values in the lower left quadrant decrease quickly early on but then stalls before reaching zero.

When the upper right quadrant of the confusion matrix is zero, this indicates that the discriminator did not misclassify CSI samples from an illegitimate transmitter. Having no illegitimate sample misclassifications is an important trait for authentication as this prevents untrusted transmitters from gaining access to the network. Ideally, the discriminator prevents all untrusted transmitters from authenticating and allows all trusted transmitters to authenticate. Unfortunately, that does not occur until the authentication accuracy is 100% at 26 dB using the densely-connected discriminator as shown in Figure 4.2d.

We also observe that there is a small difference in the numbers of samples being processed at each SNR value. Because the original dataset was separated by choosing random samples to create the training and testing datasets, the same number of legitimate samples is not represented across all SNR values. This is evident by adding the values in the lower right and lower left quadrants in Figures 4.2 and 4.3, and noting the sums are not equal for each SNR value. However, all of the illegitimate "Eve" samples are equally represented at each SNR, as seen by the sums of the upper quadrants in Figures 4.2 and 4.3.

4.4.2 Classifier Results – 50 Labeled Samples

Testing on 63,000 samples not seen during training, the classifiers are tasked to correctly assign each sample to its appropriate class label. The accuracy of the densely-connected and CNN SGAN-trained classifiers is compared to the standalone densely-connected and CNN classifiers in Figure 4.4.

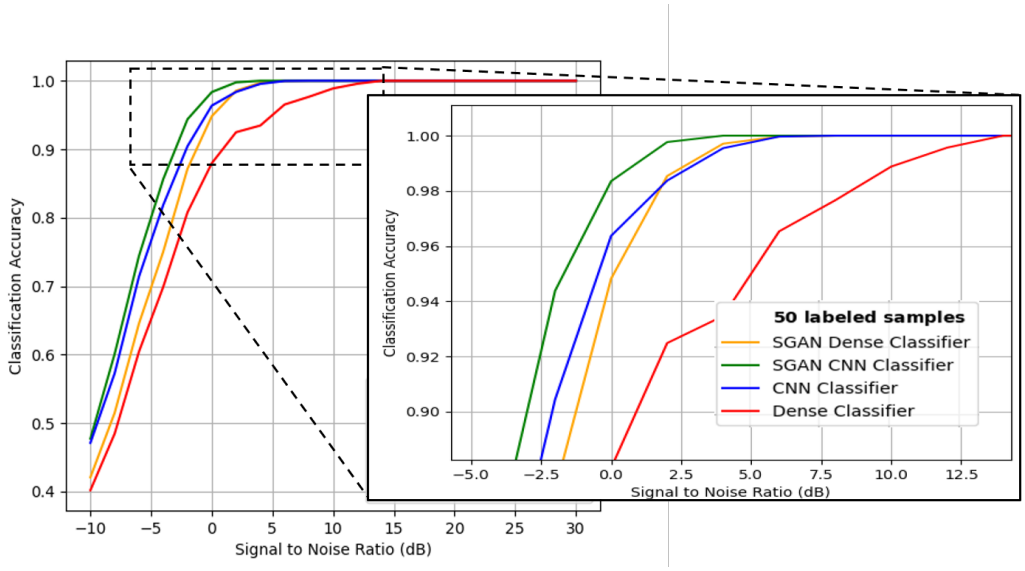
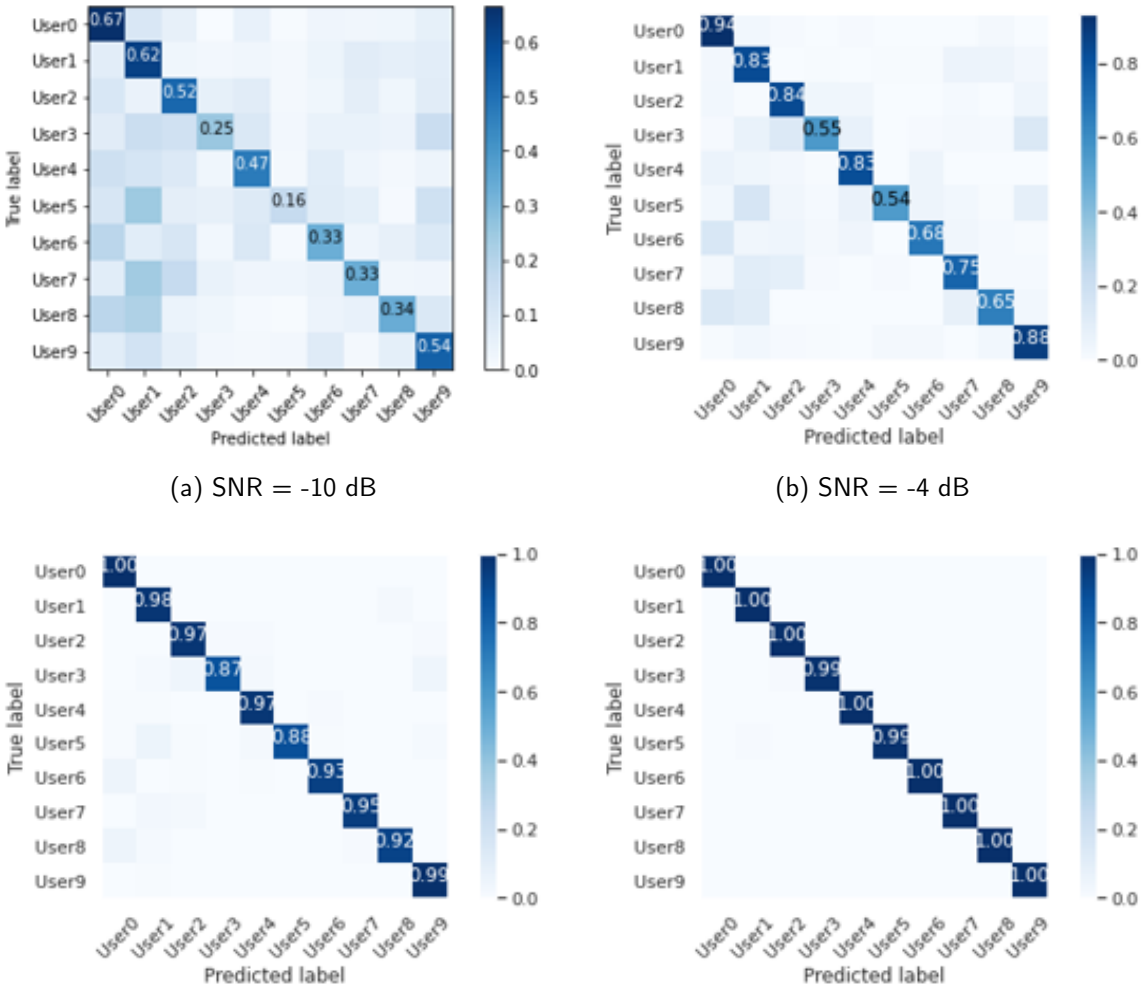


Figure 4.4. Classifier accuracy performance vs SNR after training on dataset with 50 labeled samples.

Figure 4.4 shows that all the classifiers reached 100% accuracy well before the densely-connected discriminator achieved 100% at 26 dB. The CNN SGAN classifier reached 100% accuracy at the lowest SNR value, 4 dB. With an additional 2 dB SNR, we see that

the densely-connected SGAN and the CNN classifiers reached 100% at 6 dB SNR. The densely-connected classifier didn't reach 100% accuracy until the SNR was 14 dB.

Figures 4.5-4.8 show the confusion matrices for the classifiers using 50 labeled samples. The confusion matrices for the classifiers are normalized, so the matrix entries shown are the categories predicted for the samples divided by the actual categories of the sample. The elements along the matrix diagonal correspond to the correct category being predicted. When we compare the classifier confusion matrices, we can see that even at the lowest SNR value of -10 dB, all the classifiers demonstrate a trend of having darker shades of blue along the diagonal, indicating the classifier is correctly identifying many of the legitimate transmitters.



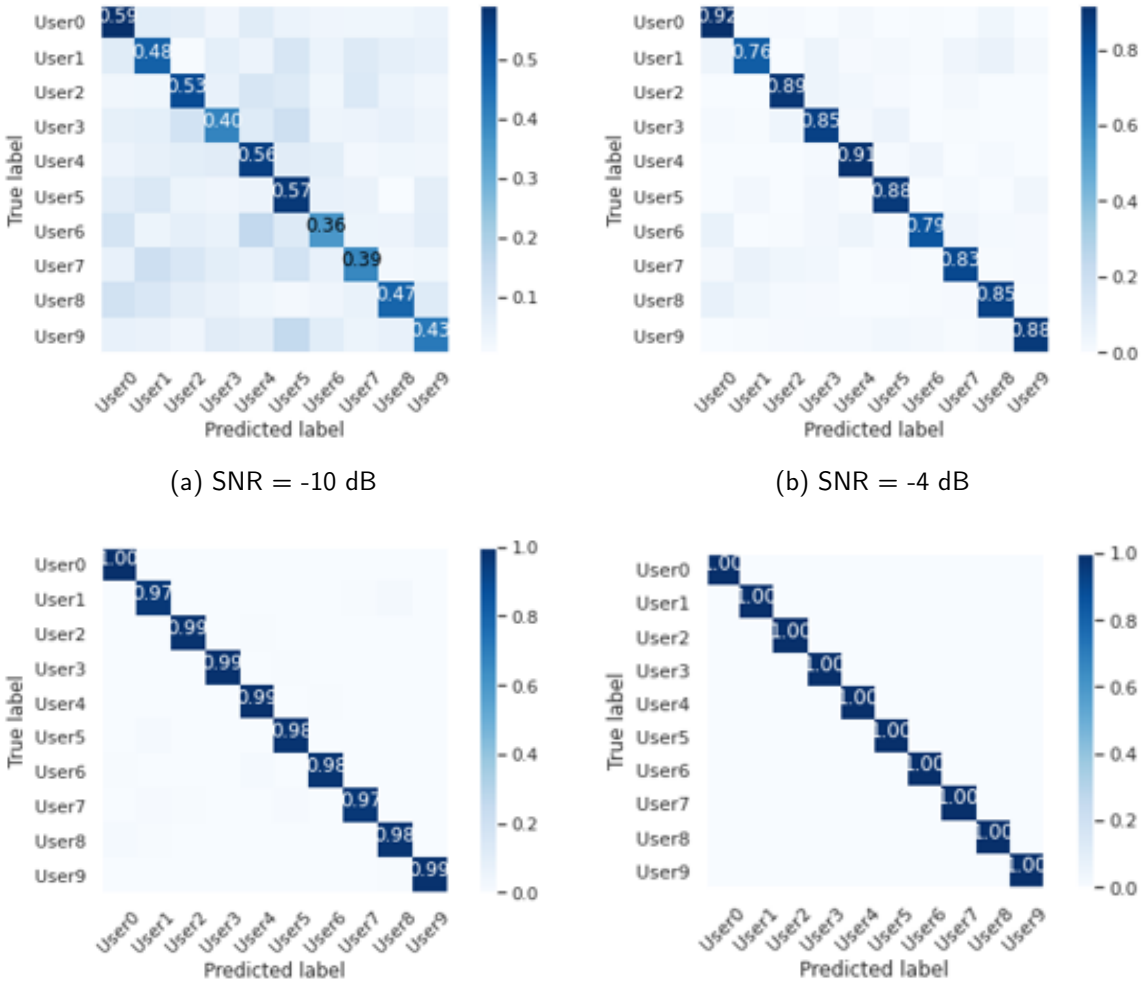
(a) SNR = -10 dB

(b) SNR = -4 dB

(c) SNR = 0 dB

(d) SNR = 4 dB

Figure 4.5. Classification predictions from densely-connected SGAN classifier trained on 50 labeled samples with SNR levels at (a) -10 dB, (b) -4 dB, (c) 0 dB, and (d) 4 dB.



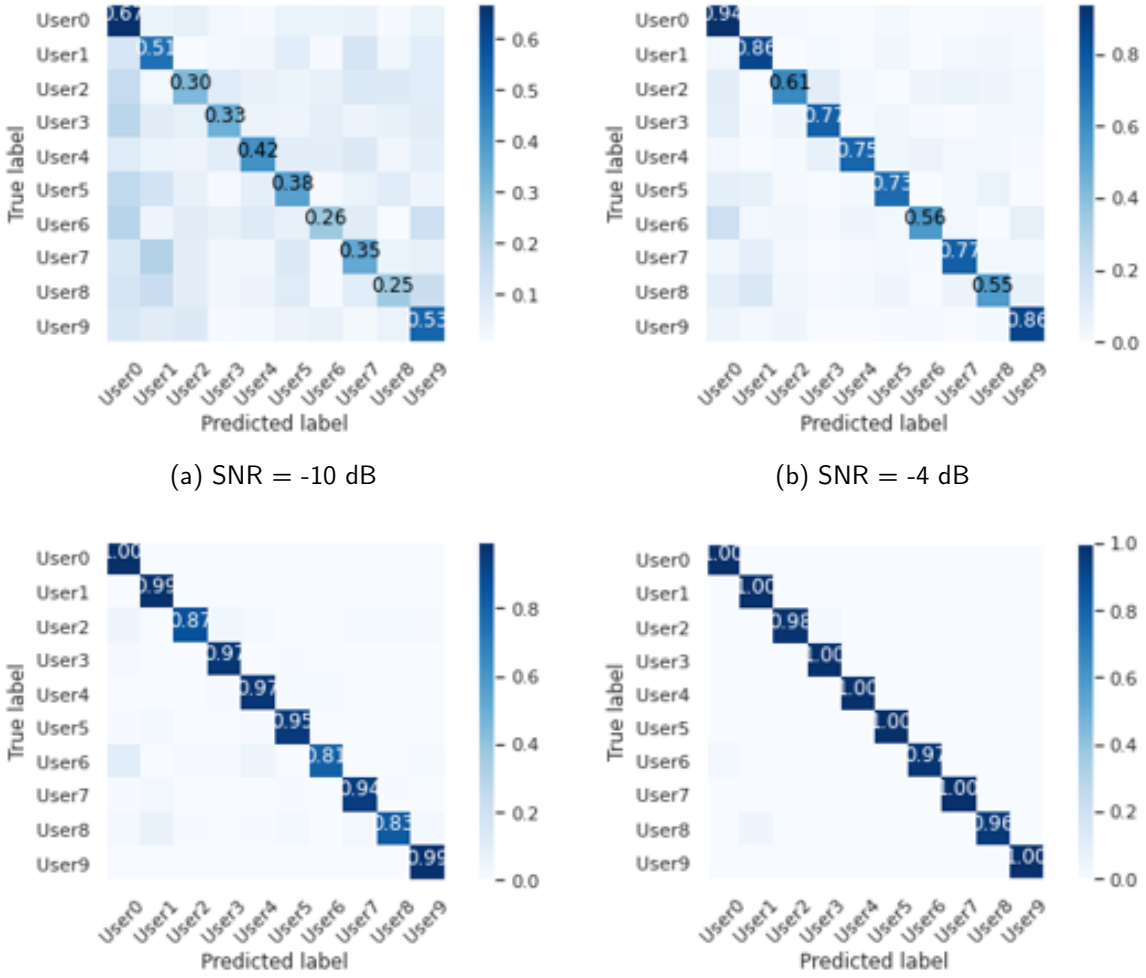
(a) SNR = -10 dB

(b) SNR = -4 dB

(c) SNR = 0 dB

(d) SNR = 4 dB

Figure 4.6. Classification predictions from CNN SGAN classifier trained on 50 labeled samples with SNR levels at (a) -10 dB, (b) -4 dB, (c) 0 dB, and (d) 4 dB.



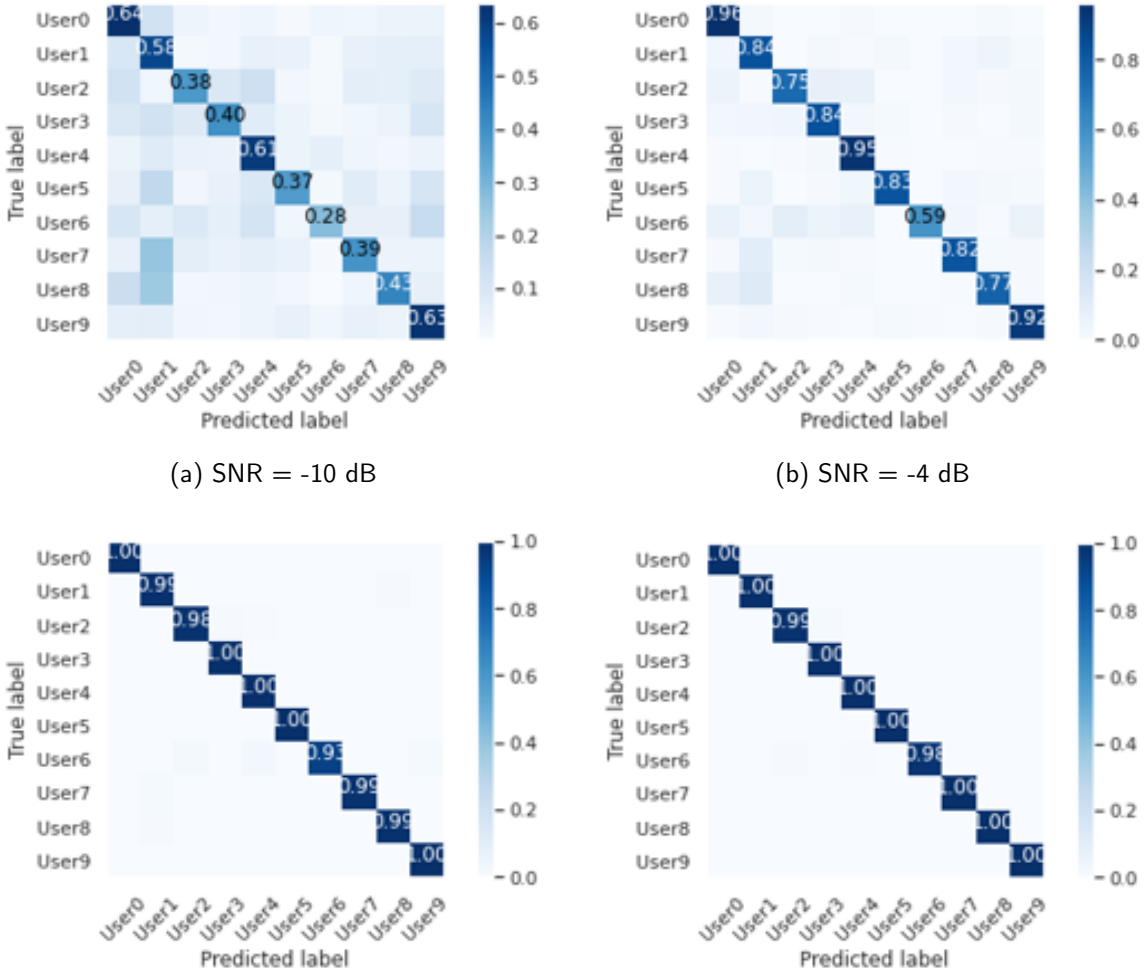
(a) SNR = -10 dB

(b) SNR = -4 dB

(c) SNR = 0 dB

(d) SNR = 4 dB

Figure 4.7. Classification predictions from densely-connected standalone classifier trained on 50 labeled samples with SNR levels at (a) -10 dB, (b) -4 dB, (c) 0 dB, and (d) 4 dB.



(a) SNR = -10 dB

(b) SNR = -4 dB

(c) SNR = 0 dB

(d) SNR = 4 dB

Figure 4.8. Classification predictions from CNN standalone classifier trained on 50 labeled samples with SNR levels at (a) -10 dB, (b) -4 dB, (c) 0 dB, and (d) 4 dB.

The classifier confusion matrix results indicate that all the classifiers have similar classification performance. It does appear that both the SGAN-trained CNN and standalone CNN have slightly better performance at the lowest range of SNR values for this dataset. The spatial correlation of received CSI may be the reason that the CNN classifier networks were more accurate than the densely-connected classifier networks. The convolutional operations performed within the CNNs extract spatial features from the input samples and are used to

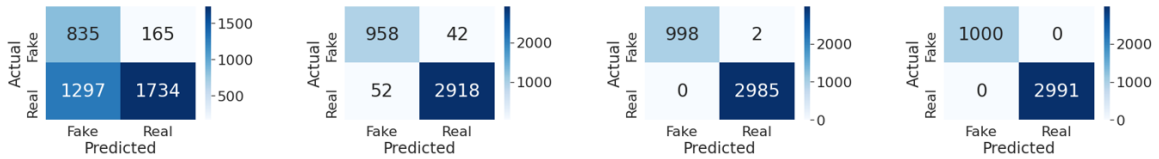
train the networks to make a classification decision.

We note that the densely-connected discriminator performed better than the CNN discriminator, while the SGAN CNN classifier performed better than the SGAN densely-connected classifier. The "Eve" samples were created using the same distribution of CSI elements as the legitimate samples before mixing with AWGN. Because CNNs use spatial features to make classification decisions, it may be more difficult for the CNN network to differentiate between "Real" and "Fake" samples since the same distribution was used to create all the samples, even as the amount of AWGN is reduced at higher values of SNR. That is, the SGAN-trained CNN discriminator may have been trained to identify the spatial relationship among the CSI elements, whereas the densely-connected discriminator network is not as sensitive to any spatial correlation in the input samples.

4.4.3 Discriminator Results – 50,000 Labeled Samples

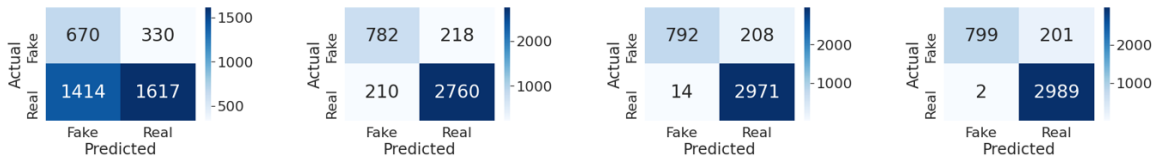
By increasing the amount of labeled samples, we hope to improve the performance of our neural networks, specifically the classifiers. We repeat our simulation, and retrain our networks using 50,000 labeled samples (34% of the 147,000 training samples).

Using this new training dataset, the confusion matrices for the SGAN-trained discriminators against the test dataset are shown in Figure 4.9 and Figure 4.10. For SNR values equal to 0 dB, 14 dB, 26 dB, and 30 dB, Figure 4.9 shows the outputs of the densely-connected SGAN discriminator and Figure 4.10 provides the output results of the CNN SGAN discriminator.



(a) SNR = 0 dB (b) SNR = 14 dB (c) SNR = 26 dB (d) SNR = 30 dB

Figure 4.9. Densely-connected discriminator test predictions for SNR levels at (a) 0 dB, (b) 14 dB, (c) 26 dB, and (d) 30 dB after training with 50,000 labeled samples dataset.



(a) SNR = 0 dB (b) SNR = 14 dB (c) SNR = 26 dB (d) SNR = 30 dB

Figure 4.10. CNN discriminator test predictions for SNR levels at (a) 0 dB, (b) 14 dB, (c) 26 dB, and (d) 30 dB after training with 50,000 labeled samples dataset.

Compared to the discriminators with 50 labeled samples in the dataset, and across all SNR values, we notice an improvement in the predictions made by the discriminators trained with 50,000 labeled samples regarding legitimate transmitters. Comparing the densely-connected discriminator results from Figure 4.2 and Figure 4.9, the discriminators trained with the 50,000 labeled samples made more correct predictions for "Real" test samples. The same was true for the CNN discriminator, with 50 labeled samples in Figure 4.3 and 50,000 in Figure 4.10.

Unfortunately, the discriminators trained on the dataset with 50,000 labeled samples did worse at recognizing illegitimate transmitters across all SNR values compared with the discriminators trained on the dataset with 50 labeled samples. This is evident by noting the larger value in the upper right quadrant of the confusion matrices when comparing the densely-connected discriminator in Figures 4.2 and 4.9 and the CNN discriminator in Figures 4.3 and 4.10. We speculate that this degrading performance is due to the additional number of training samples provided to the classifier and that the classifier received a larger distribution of low-SNR samples in the 50,000 labeled samples case compared to the 50 labeled samples case. The samples processed by the classifier were evenly distributed across classes but did not control for SNR value. When processing 50,000 labeled samples, the discriminator and classifier network weights may have been adjusted to create a larger region for which acceptable CSI element values may occupy than with the region created with the 50 labeled samples. At low SNR values and larger acceptable CSI element ranges, it may have been challenging to differentiate "Real" and "Fake" samples from the test dataset ultimately leading to more illegitimate samples being assessed as legitimate. By virtue of the improvement the discriminators made at recognizing "Real" samples, the overall accuracy

improves at almost every SNR value as shown in Figure 4.11, however, this behavior would allow more instances of illegitimate transmitters to authenticate.

In Figure 4.9, we can see that as the SNR value increases, the densely-connected discriminator simultaneously improves in identifying "Real" samples as the number in the lower left quadrant decreases in value, and also in identifying "Fake" samples as the upper right quadrant diminishes to zero. Contrast with the behavior of the CNN discriminator, where the upper right quadrant does not decrease with larger SNR, and the values in the lower left quadrant decrease quickly early on and then stall before reaching zero.

Figure 4.11 shows the accuracy of the retrained discriminators across the range of SNR values using 50,000 labeled samples in the training dataset as well as the accuracy with 50 labeled samples.

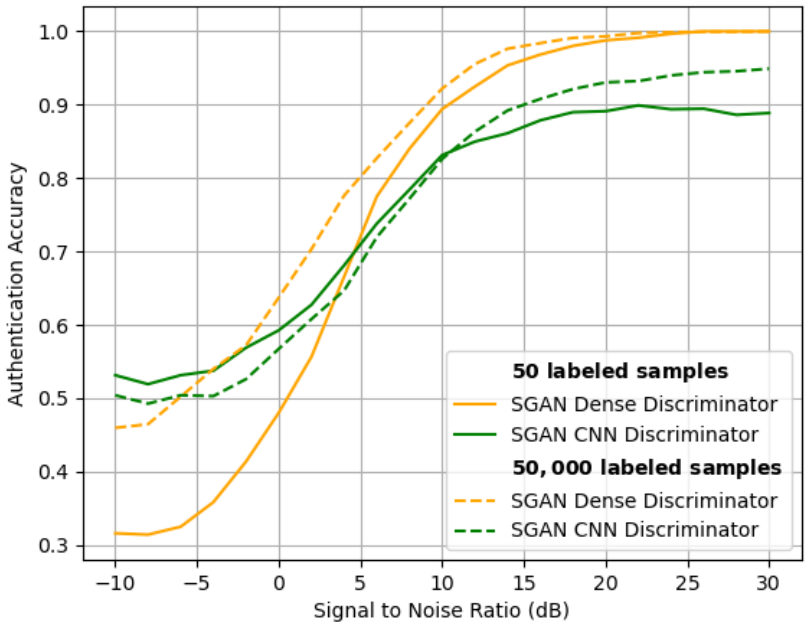


Figure 4.11. Discriminator accuracy performance vs SNR for 50,000 labeled samples.

The performance of the densely-connected SGAN-trained discriminator appears to improve at all values of SNR, however it doesn't reach 100% accuracy until 30 dB. Training on the

dataset with 50 labeled samples, the densely-connected SGAN discriminator reached 100% at 26 dB as shown in the confusion matrix in Figure 4.2d.

4.4.4 Classifier Results – 50,000 Labeled Samples

After training with a dataset containing 50,000 labeled samples, the classifiers are tasked to correctly assign each of 63,000 test samples to the appropriate class label. Comparing the results of these classifiers against the classifiers trained with 50 labeled samples, we see an accuracy improvement for three of the four classifiers as shown in Figure 4.12. Only the SGAN-trained CNN classifier has a slight reduction in classification accuracy.

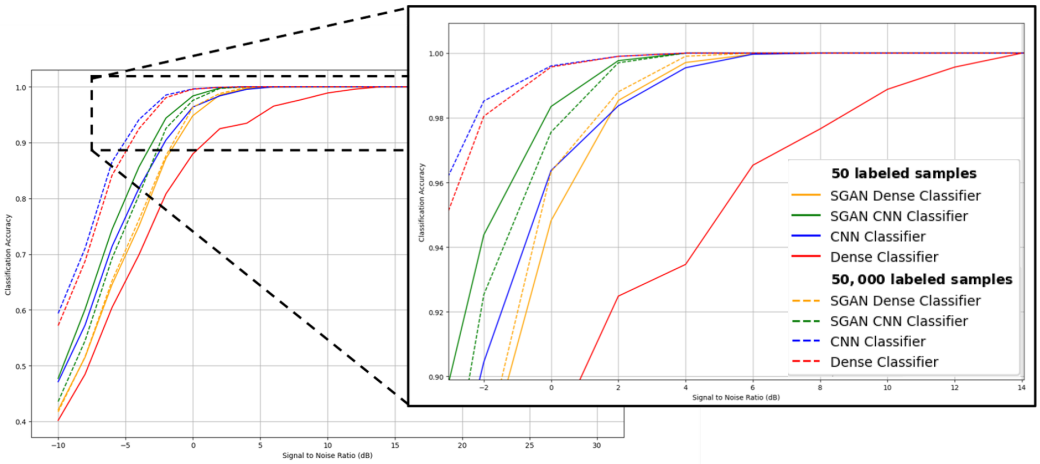
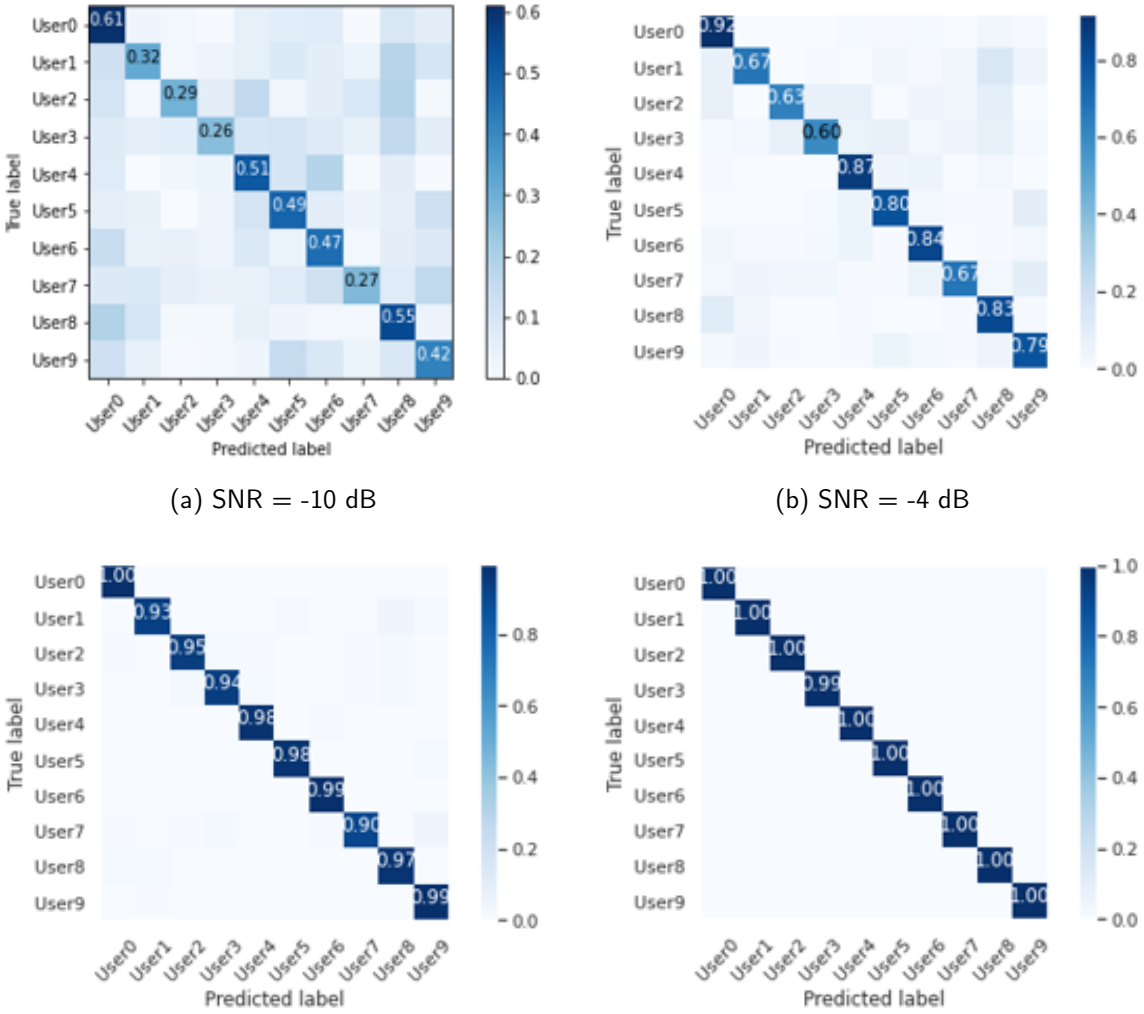


Figure 4.12. Classifier accuracy vs SNR for 50,000 labeled samples.

While the SGAN-trained densely-connected classifier improved after training with 50,000 labeled samples, it was a more modest improvement than that of the standalone classifiers as shown in Figure 4.12. However, all classifiers reached 100% accuracy at 4 dB.

Figures 4.13-4.16 show the confusion matrices for the classifiers after training with 50,000 labeled samples.



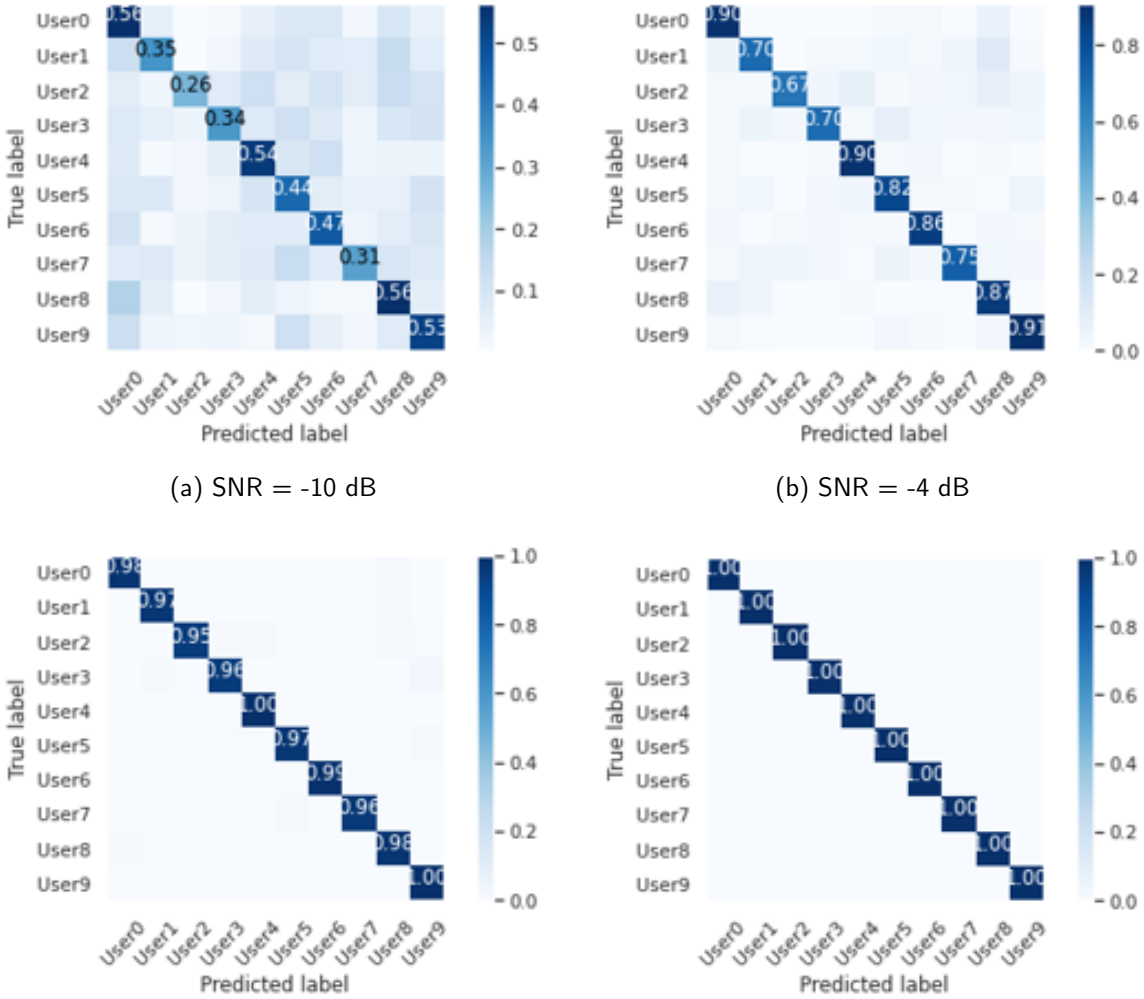
(a) SNR = -10 dB

(b) SNR = -4 dB

(c) SNR = 0 dB

(d) SNR = 4 dB

Figure 4.13. Densely-connected SGAN classifier performance against 50,000 labeled samples test dataset with SNR levels at (a) -10 dB, (b) -4 dB, (c) 0 dB, and (d) 4 dB.



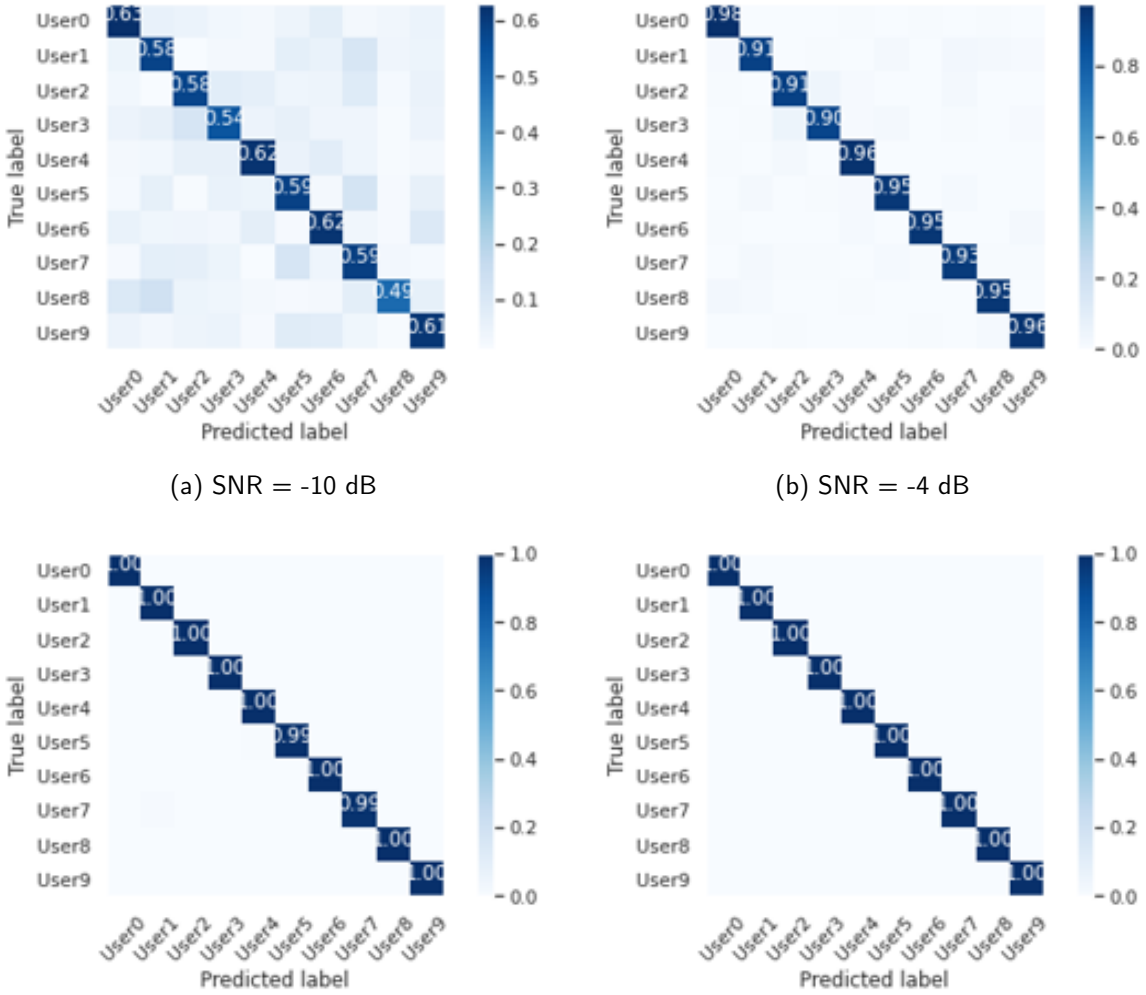
(a) SNR = -10 dB

(b) SNR = -4 dB

(c) SNR = 0 dB

(d) SNR = 4 dB

Figure 4.14. CNN SGAN classifier performance against 50,000 labeled samples test dataset with SNR levels at (a) -10 dB, (b) -4 dB, (c) 0 dB, and (d) 4 dB.



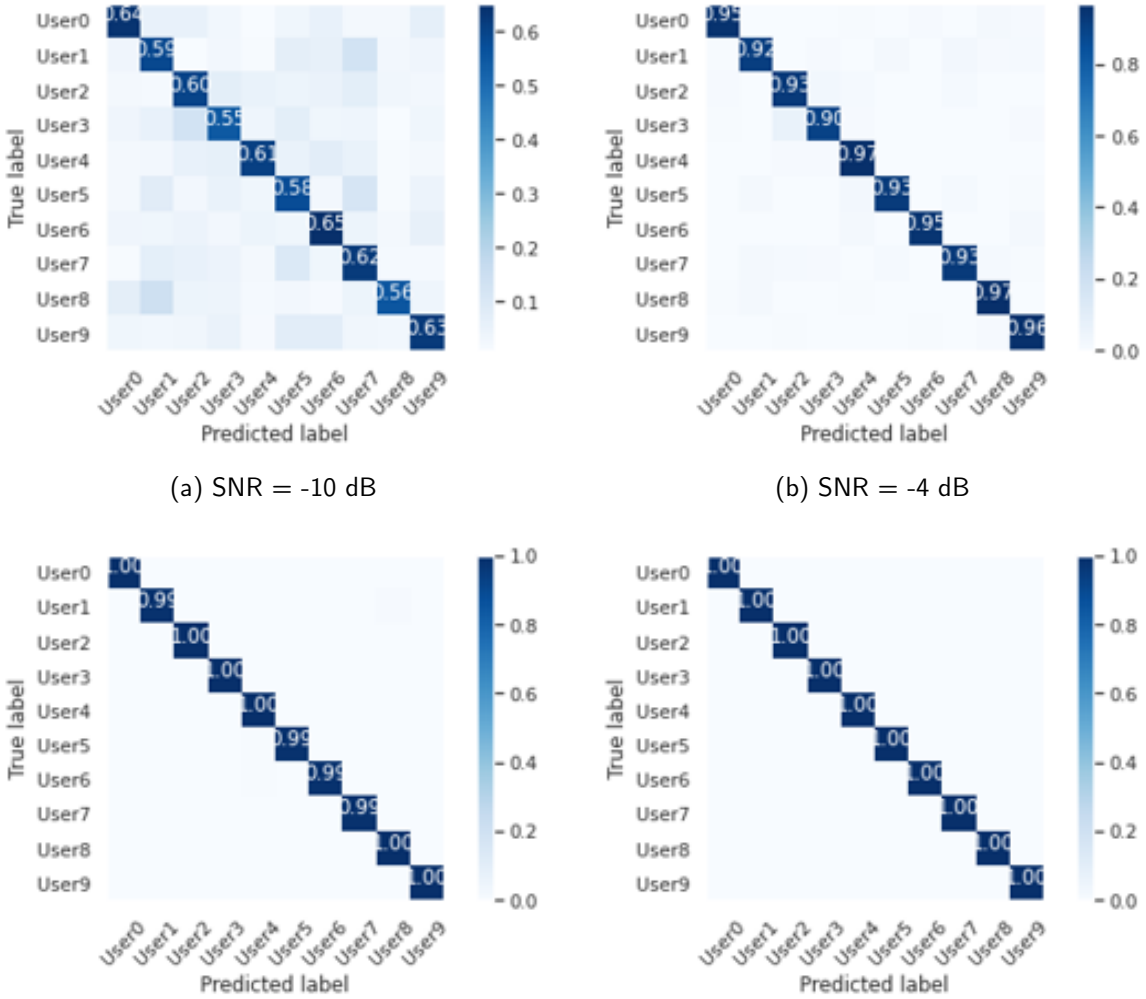
(a) SNR = -10 dB

(b) SNR = -4 dB

(c) SNR = 0 dB

(d) SNR = 4 dB

Figure 4.15. Densely-connected standalone classifier performance against 50,000 labeled samples test dataset with SNR levels at (a) -10 dB, (b) -4 dB, (c) 0 dB, and (d) 4 dB.



(a) SNR = -10 dB

(b) SNR = -4 dB

(c) SNR = 0 dB

(d) SNR = 4 dB

Figure 4.16. CNN standalone classifier performance against 50,000 labeled samples test dataset with SNR levels at (a) -10 dB, (b) -4 dB, (c) 0 dB, and (d) 4 dB.

Based on the changes in accuracy performance from Figure 4.12, we see very little change in SGAN-trained classifier performance comparing Figures 4.5 and 4.6 with Figures 4.13 and 4.14. Across the 10 classes of legitimate users, the confusion matrices appear very similar across the range of SNR. On the other hand and indicating an improvement in prediction accuracy, the standalone classifiers trained with 50,000 labeled samples have

much darker diagonal elements in their respective confusion matrices compared to the standalone classifiers trained with 50 labeled samples. This is especially evident at -10 dB and -4 dB in Figures 4.7 and 4.15 for the standalone densely-connected ANNs and Figures 4.8 and 4.16 for the standalone CNNs.

4.4.5 Training Epochs

During training, we saw that the SGAN discriminators and classifiers reached their highest levels of accuracy in less than 50 epochs. The standalone classifiers took many more epochs to reach their peak performance while using the training dataset with 50 labeled samples, however this was greatly reduced when more labeled samples were made available. Table 4.3 summarizes the amount of training required by the networks to produce the results in this section.

Table 4.3. Training epochs required to produce best result.

	50 labeled samples	50,000 labeled samples
Densely-connected SGAN	32	7
CNN SGAN	49	14
Densely-connected standalone	1005	22
CNN standalone	570	49

Although the GAN-based networks required fewer passes through the dataset to achieve their best results, the training routine required more computations and took more time. While the standalone classifiers shared the same parameters as the SGAN classifiers/discriminators, the training plan was more simple. For each epoch, the standalone classifiers conducted forward passes and backpropagation based on the labeled batch size. The SGAN conducted these same forward and backward passes for the classifier networks, but also added more processing for the discriminator using unlabeled samples and the generator. Additionally, the SGAN training required a serial implementation for the classifier, discriminator, and the generator, meaning that parallel processing could not be used to speed up the task.

4.5 Summary

In this chapter, we showed how machine learning, specifically the use of deep SGAN, can be used to classify transmitters by MIMO CSI as a method to provide physical layer authentication. We considered a system of multiple transmitters each operating on a single carrier. Our simulation results using the training dataset with 50 labeled samples illustrated that with a very small percentage of labeled CSI samples, accurate classification can be made with a SGAN-trained classifier for SNR values greater than 4 dB. During the initial authentication setup between two transceivers, a small number of labeled samples is desired to minimize overhead. While fewer passes through the training datasets were required for the SGAN-trained classifiers, if more labeled samples were available to the receiver, the standalone classifiers produced better results at all SNR levels.

While the classifier networks from the SGANs performed well, especially with a small number of labeled samples, the discriminators did not fare as well. Although the densely-connected SGAN discriminator performed better than the CNN discriminator, our system model as described in Section 4.1 only would use the classifier portion of the SGAN if a sample was categorized as "Real" by the discriminator. Unfortunately, the discriminators didn't perform at 100% accuracy across the range of SNR tested here.

One method to address the shortcoming of the discriminator is to implement the classifiers with a method to categorize $N + 1$ classes consisting of N legitimate transmitters and one additional class for assigning illegitimate transmitters. In this configuration, the classifier is capable of classifying legitimate and illegitimate transmitter CSI. A GAN-based approach can be used here, with the generator creating samples that attempt to match one or several of the CSI matrices produced by a legitimate transmitter.

Another method to improve the discriminator is to provide additional features to the input samples. We showed in this chapter that our SGAN approach was sound and that the results encouraged additional research. We used a simple approach to generate CSI elements, and although our discriminator networks did not achieve 100% accuracy, we were successful in developing neural networks that were able to classify received transmitter CSI at various SNR levels. In the next chapter, we will consider communication systems that use multiple subcarriers. The subcarriers provide another dimension from which additional features can be extracted and learned. Changes to the architecture of the discriminator and classifier

were required to account for the changes to the SGAN input.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Multisubcarrier Authentication and Classification

In Chapter 4, we used a SGAN to prevent illegitimate devices from authenticating, and then correctly identified multiple legitimate transmitters based on features observed at the physical layer. In this chapter, we extend the SGAN for use when there are multiple subcarriers.

To simulate MIMO millimeter wave OFDM subchannels, we take advantage of the DeepMIMO dataset [147], based on ray-tracing data from the Remcom Insite tool [148]. The DeepMIMO dataset is configurable to a variety of wireless applications, and we use it to create samples that train and test the SGAN in a 4×4 MIMO environment operating with a 60 GHz carrier frequency with 16 pilot subcarriers from a 512 OFDM subcarrier system in an urban setting. Although the DeepMIMO dataset scenario is based on an urban environment, more fully appropriate use-cases for static channels might include an uninhabitable industrial setting, or a in a remote, deployed sensor network.

This chapter includes material adapted from work published by the author. This revised material is from "Multi-Subcarrier Physical Layer Authentication Using Channel State Information and Deep Learning" by Ken St. Germain and Frank Kragh, published in the 14th Hawaii International Conference on Signal Processing and Communication Systems [38].

5.1 Channel Model

Although CSI elements may be independent for a single subcarrier, that does not necessarily hold true when there are multiple subcarriers, such as in an OFDM system. For K subcarriers, we extend (1.6) by adding an additional dimension as a superscript where $k = 1, 2, \ldots, K$ is the sampled subcarrier, resulting in

$$\boldsymbol{y}^k = \boldsymbol{H}^k \boldsymbol{x}^k + \boldsymbol{n}^k \tag{5.1}$$

where $\boldsymbol{H}^k$ is a three-dimensional tensor of size $N \times M \times K$. Fading across the subcarrier channels will be correlated if the coherence bandwidth is large [149], resulting in corre-

lated CSI elements across subchannels, for example, $h^k_{1,1}$. For example, depending on the coherence bandwidth, there may be correlation between adjacent OFDM channels, but not across the entire band of subcarriers or adjacent pilots channels.

5.2 Authentication with Measured CSI

We will continue to use our previously-introduced authentication model, as described in Section 3.1. A receiver continues to authenticate a transmitter if the received CSI varies less than a threshold applied to the received CSI from previous transmissions. This requires some method of initial authentication, such as the use of cryptographic methods or physical layer authentication using RF fingerprinting from transmitter imperfections. During initial authentication, the receiver makes CSI measurements of the channel and stores that information for future authentication.

As described in Section 3.1, the measurements made by the receiver of the CSI elements are altered due to noise in the receiver. This error $\boldsymbol{\epsilon}$ is modeled as an additive complex zero-mean Gaussian process on each subcarrier $\mathcal{CN}(0, \boldsymbol{\Sigma}_\epsilon)$ where the covariance of the sample mean is $\boldsymbol{\Sigma}_{\overline{\epsilon}} = \boldsymbol{\Sigma}_\epsilon / s$ for s samples during the measurement. Extending Equation 5.2, the tth CSI measured by the receiver at subcarrier k $\hat{\boldsymbol{H}}^k_t$ is given as

$$\hat{\boldsymbol{H}}^k_t = \boldsymbol{H}^k + \boldsymbol{\epsilon}^k_t \qquad t = 1, 2, \ldots, s \tag{5.2}$$

where $\boldsymbol{H}^k$ is the true CSI from (5.1) and $\boldsymbol{\epsilon}^k_t$ is a complex $N \times M \times K$ tensor with independent identically distributed elements. Since $\boldsymbol{\epsilon}^k_t$ is zero-mean, $\boldsymbol{H}^k$ can be estimated with a variety of techniques including least squares estimation, minimum mean-square error estimation, and through successive measurements and element-wise averaging of $\hat{\boldsymbol{H}}^k_t$ for $t = \{1, 2, \ldots, s\}$ as demonstrated in [142].

5.3 Semi-Supervised GAN

Introduced in Chapter 1 and demonstrated in Chapter 4, semi-supervised learning uses a small percentage of labeled training data. Implementing a multiple-class classifier network from a binary classifier can be done in a variety of ways. Following Salimans et al. [129], we can build an N-class classifier network C with output logits $\{l_1, l_2, \ldots, l_N\}$ prior to the *softmax* activation for C. The logits vector is then used as the input to the activation function

for $\mathcal{D}$, which is given as $D(x) = \frac{Z(x)}{Z(x)+1}$, where $Z(x) = \sum_{n=1}^{N} \exp[l_n(x)]$. In Chapter 4, the last layer of the densely-connect network and the CNN discriminators were a single fully-connected node followed by a sigmoid activation function. Here, we use the implementation suggested by Salimans in [129] to remove the final densely-connected layer and the associated weights to be trained, thereby improving efficiency in the network. Because $\mathcal{D}$ and C share the same weights, both networks act as a single network $\mathcal{D}/C$ that is updated during backpropagation based on their respective loss functions $J^{(\mathcal{D})}$ and $J^{(C)}$. The generator loss function is given by $J^{(\mathcal{G})}$.

The training dataset is partially labeled and provided to the $\mathcal{D}/C$ model for classification by C. The remainder of the training dataset as well as the generated samples from $\mathcal{G}$ are used as input to $\mathcal{D}/C$ for discrimination where $\mathcal{D}$ will predict whether the sample came from the training dataset or if it was created by $\mathcal{G}$.

5.4 The DeepMIMO Dataset

Using the Remcom Insite ray-tracing tool [148], Alkhateeb developed the DeepMIMO dataset generation framework [147]. The framework allows researchers to tailor parameters in a MATLAB [146] program to suit the need of their machine learning based wireless application. This section discusses the setting we use to obtain our training and testing data, and the parameters we selected for our model.

5.4.1 Target data and labels

The target data for our proposed SGAN is the received 4×4 MIMO CSI from a transmitter to 14 distinct legitimate user locations $\{User0, User1, \ldots, User13\}$ across 16 subcarriers. The transmitter and receivers operate at 60 GHz using 512 OFDM subcarriers. Each of the 16 pilot subcarriers are evenly spaced 32 subcarriers apart. A single target sample consists of 256 complex numbers, accounting for 16 CSI elements in each of their respective 16 subcarriers.

For each CSI target sample created in the dataset, there is an accompanying target label, denoting the user. During training, the goal for the SGAN classifier will be to differentiate among the legitimate users. The SGAN discriminator will attempt to categorize these legitimate users as “Real”, and categorize the generator-created samples as “Fake”.

During testing for the discriminator, an additional user, representing a malicious actor will be added to the test dataset. The discriminator will not have seen this data during testing, but will need to identify samples from the malicious user's CSI as "Fake" to prevent authentication. The classifier will attempt to assign the correct label for each users' CSI from the test dataset, however the classifier will not be exposed to the malicious user's CSI, since in application, the discriminator would have already prevented authentication.

5.4.2 DeepMIMO scenario

The setting used for the scenario is denoted "O1" and is described in detail in [147]. The "O1" scenario is an outdoor urban setting with a variety of possible transmitter base station locations and user locations on the streets surrounded by buildings of various heights. Figure 5.1 shows the position of the 14 legitimate users denoted by blue circles and the red square indicating the malicious user in the white patch above the "User Grid 3" label. The user locations are specified by rows in one of the three grids that correspond to the streets in the scenario. We selected three rows to place our 15 users. Base station 7 (BS7) is the transmitter for our case, circled in white and is across the intersection from the users. Both streets are 40 m wide, and the user positions are centered in the street going in the X direction and 7 m from the street going in the Y direction. There are 10 cm between adjacent users, and each user as well as BS7 has four antennas.

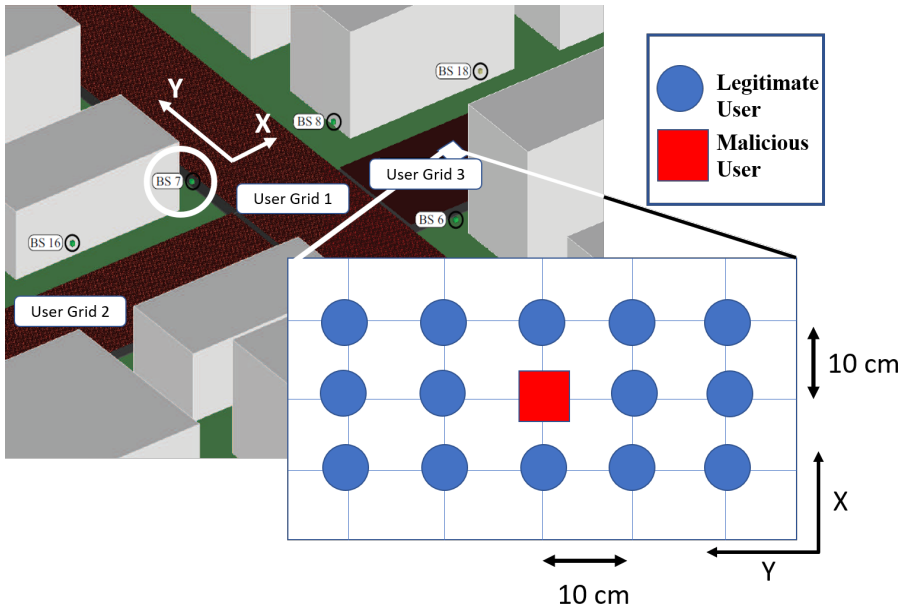


Figure 5.1. DeepMIMO scenario "O1." Source: [38].

5.4.3 DeepMIMO parameters

The parameters were chosen to emulate advanced wireless communication technologies, but they are not intended to model any specific standard. Table 5.1 summarizes the parameters used. Advanced technologies refers to communications systems using millimeter wavelengths and multiple antennas for transmitting and receiving signals. We believe that such devices will become adopted and more commonplace in the future. While the parameters chosen do not match any particular technology, they are analogous to those described by IEEE standards recently released or currently in draft.

Table 5.1. DeepMIMO dataset parameters. Source: [38].

Base Station	7
First row of users	4528
Last row of users	4531
Center frequency	60 GHz
Antenna spacing	1 wavelength
System bandwidth	8.64 GHz
OFDM channels	512
OFDM channel interval	32
Number of paths	3

5.5 System Model

We consider a 4×4 wireless MIMO communications channel using 512 OFDM subchannels with 16 pilots. There are 14 trusted users and some unknown number of untrusted users; some of the latter group are malicious adversaries. The adversaries have resources available to change their antenna characteristics, transmitter RF path timing, output power, and/or present reflectors between themselves and the receiver. Thus, they are able to change their CSI as measured by the receiver and may have an accomplice receiver to provide feedback as described by Shi et al. in [43]. Although the adversaries have the ability to change their CSI, they do not have accurate advanced knowledge of the CSI required to spoof the user. A user becomes a victim if the malicious adversary is able to create CSI that is authenticated as the transmitter by the user.

To defeat this scenario, the discriminative model at the receiver is trained by a generative model that creates authentic looking CSI samples. By training with increasingly high quality "Fake" samples, the discriminative network learns the features of transmitters that should be authenticated and the features of those that should not be authenticated. Parallel to the adversarial training between $\mathcal{D}$ and $\mathcal{G}$, the classifier C learns the correct labels assigned to the 14 trusted users.

During an initial authentication session by other means, the pilot subcarriers from the transmitters are measured and recorded for training the SGAN. Initially authenticating by

other means, higher protocol layers are used, however for subsequent packet transfers, authenticating at the physical layer reduces the workload on these higher-layer protocols as discussed in [150].

The classifier provides identification only after the discriminator successfully authenticates. The discriminator authenticates when a sample is assessed to be "Real". The discriminator may make an incorrect authentication decision (denying authentication when the sample is "Real" or authenticating when the sample was actually faked), therefore we explored how SNR can affect discriminator accuracy.

5.5.1 SGAN architecture

The adversarial competition in the SGAN is a minimax game described by Equation 1.20 where the discriminative model attempts to correctly identify authentic training samples from a distribution produced by CSI matrix elements $p_{data}(h_{n,m})$ and fake training samples created by the generator.

As $\mathcal{D}$ and $\mathcal{G}$ adversarially train each other, they learn to improve their individual performance. When the discriminative model correctly identifies fake samples created by the generative model, the generative network will update its parameter weights through back-propagation to make more realistic samples. Likewise, the discriminative model will update its parameter weights when it incorrectly identifies "Real" or "Fake" samples. The results of this training are a generator neural network adept at creating data that closely mimics training data, a discriminator neural network that can identify all but the best fakes, and a classifier neural network that can determine which trusted transmitter produced the received CSI.

Additionally, C is trained on labeled samples from the training dataset. Although C does not directly receive unlabeled authentic or fake samples, the weights of C are affected since it shares weights with $\mathcal{D}$ in the $\mathcal{D}/C$ implementation.

Best practices from GAN researchers [151] were used to create the SGAN. The architecture for the discriminator and classifier was chosen to enable feature extraction from the input tensor. A reverse architecture was used for the generator to create realistic-looking samples.

5.6 Simulation

This section describes the simulation of the system model from Section 5.5. The dataset for the SGAN is described and results are presented.

5.6.1 Dataset

A dataset of 224,000 authentic samples was created, where each sample was a $4 \times 4 \times 16$ complex tensor. The training dataset was allocated 70% of the greater dataset, while the remaining 30% was set aside for testing. Every sample started as a position-dependent $4 \times 4 \times 16$ tensor created by the DeepMIMO dataset. For each of the samples, 1,000 additional samples of measurement error in the form of 16 different levels of SNR were generated. Simulating thermal noise in the receiver, decreasing amounts of AWGN were combined with the original signal to produce the 16 different levels of SNR ranging from -10 dB to 20 dB in steps of 2 dB.

The SGAN processed 16 subcarriers in a MIMO 4×4 configuration with 14 trusted transmitters. Therefore, the classifier model would need to have 16×16 complex inputs and 1 real output for each transmitter label. However, we separated the real and imaginary parts for processing through the neural networks, resulting in inputs tensors of shape $16 \times 2 \times 16$. The discriminative model also has inputs of shape $16 \times 2 \times 16$ and 1 real output denoting "Real" or "Fake", while the generative model has 1 real input and $16 \times 2 \times 16$ outputs. Additionally, the values of the real and imaginary parts were preprocessed to scale $[-1, 1]$ to allow for the hyperbolic tangent activation function range in the generator network as mentioned in Section 5.6.2.

To mimic the malicious user's attempt to fool a legitimate user, CSI is generated for a user position in the center of the group of legitimate users, as shown in Figure 5.1. This sample is preprocessed as before, to include creating 1,000 samples of 16 different levels of SNR. These samples are then added to the testing dataset, remaining unknown to the SGAN until testing following the completion of training.

5.6.2 SGAN development

The SGAN was implemented using the Python programming language, Keras [144] front-end, and Tensorflow [145] back-end. Additionally, Numpy, and Matplotlib Python libraries

were used. The dataset was created using MATLAB and Python. The overall SGAN design is summarized in Table 5.2, with a total of 9,098,542 parameters for the classifier and the discriminator, and 7,511,664 parameters for the generator.

Table 5.2. SGAN architecture.

	Classifier	Discriminator	Generator
Inputs	16	16	1
Hidden Layers	9	9	9
Outputs	1	1	16
Parameters	9.1M	9.1M	7.5M

The discriminator/classifier network $\mathcal{D}/C$ is a dense or fully-connected DNN with 16 inputs of size 2×16 merged into one *concatenated* layer. Each input has two neurons to accommodate the real and imaginary parts of the complex CSI matrix element. Nine additional fully connected layers with LeakyReLU activations (*alpha* = 0.3) follow. All hidden layers use *Dropout* of 0.5 to prevent overfitting. Prior to the output layers, a fully-connected layer of 14 neurons is used to capture the number of transmitters to be classified. The discriminator output layer of size 1 is fully-connected and uses a custom activation $D(x) = \frac{Z(x)}{Z(x)+1}$, where $Z(x) = \sum_{n=1}^{N} \exp[l_n(x)]$ to provide values $(0.0,\ 1.0)$ as discussed in Section 5.3. The classifier output is a *softmax* activation connected to the 14-neuron layer. The learning rate for $\mathcal{D}/C$ was 0.00009 using the Adam [134] optimizer and training was done with batches of 128 samples.

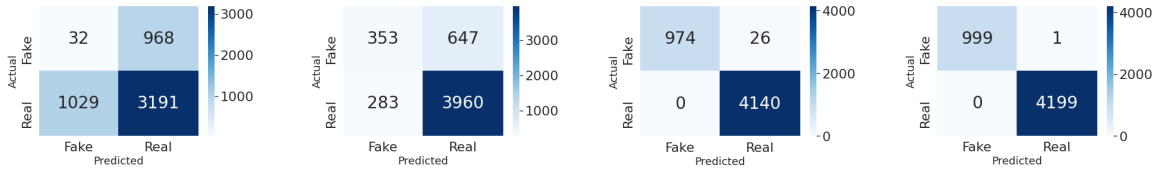
The generator network $\mathcal{G}$ has a single input with 5 neurons fully connected to the first hidden layer of size 16. Seven additional hidden layers are again fully connected using LeakyReLU (*alpha* = 0.3). The last hidden layers are 16 fully-connected layers of size 32 followed by hyperbolic tangent activations. Finally, the output is reshaped to produce 16 output layers of size 2×16. The learning rate for $\mathcal{G}$ was 0.00009 using the Adam optimizer.

5.6.3 Results

Training was conducted over the course of 20 epochs. Of the 156,800 samples in the training dataset, just over 10% (15,988) were labeled. These labeled samples trained the classifier to identify the legitimate users. When selecting the labeled samples, care was taken to ensure

an equal distribution of samples for each of the 14 legitimate users, however the SNR in the samples for each of the users was left to chance. All the training samples as well as those created by the generator were used to train the discriminator. Following training, the classifier and discriminator networks and their respective weights were saved. For testing, the classifier and discriminator networks and weights were reloaded and presented with the test dataset. The test dataset for the discriminator contained additional samples associated with the malicious user.

Figures 5.2 and 5.3 show that the discriminator performs well for SNR greater than 2 dB. The confusion matrices in Figure 5.2 show the discriminator labeled the malicious user's CSI as "Real" for low SNR, but gradually began to correctly categorize them as "Fake" as the SNR increases. At each SNR, there are 1,000 "Fake" samples, however the number of "Real" samples varies slightly due to the random split of the original dataset into training and testing components. For authentication, if there is an error it is likely more favorable to have a false negative rather than a false positive. Although this can be frustrating for the authentic user denied authentication and result in lower throughput rates because of restarting the authentication process, malicious users are kept out of the system.



(a) SNR = -10 dB (b) SNR = -4 dB (c) SNR = 2 dB (d) SNR = 4 dB

Figure 5.2. SGAN dense discriminator performance with SNR at (a) -10 dB, (b) -4 dB, (c) 2 dB, and (d) 4 dB. Source: [38].

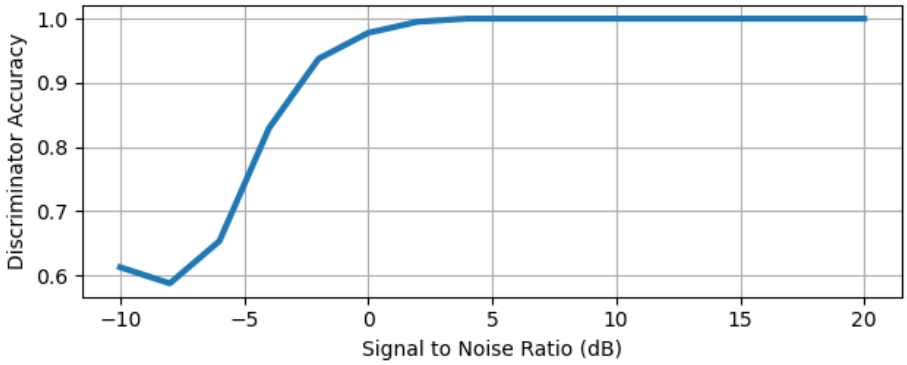
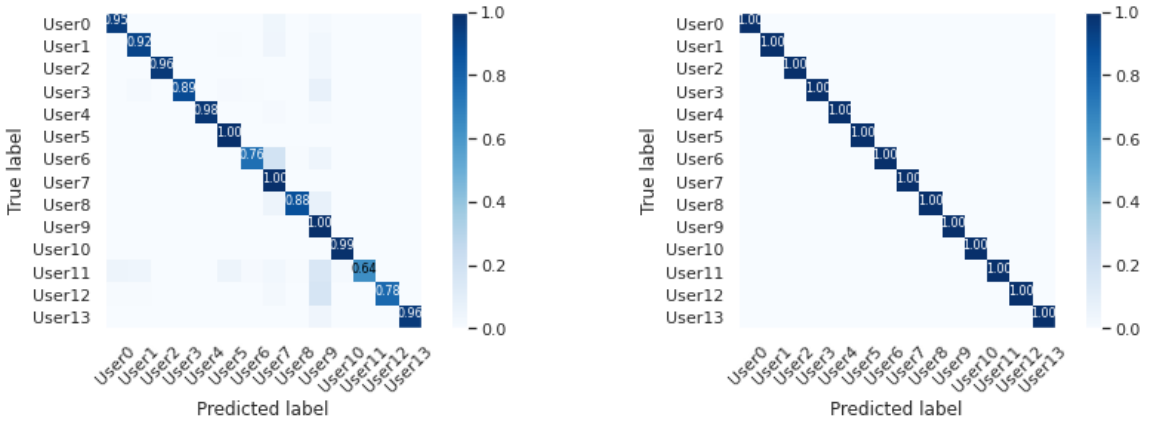


Figure 5.3. SGAN dense discriminator accuracy vs SNR. Source: [38].

The SGAN-trained densely connected discriminator was accurately able to differentiate "Real" from "Fake" at SNR values above 4 dB. This result shows the limitations of the SGAN approach. The quality of the generator is one aspect that determines how well the discriminator will perform. Training a traditional standalone neural network to differentiate "Real" from "Fake" without a robust generator requires "Fake" samples from another source. While this can be obtained, it is likely not feasible to sample every possible fake CSI sample. The SGAN does not need these negative examples because it creates its own and still performs well provided a sufficient SNR.

The test dataset without the CSI samples from the malicious user was then used to obtain the performance for the classifier. As shown in Figure 5.4, the confusion matrices indicate accurate classification performance even at low SNR values. Accuracy is measured by dividing the correctly classified samples by the sum of the correctly and incorrectly classified samples. Figure 5.4a shows that the classifier attained classification accuracy above 90% for most of the users at -10 dB SNR, and Figure 5.4b shows 100% accuracy at -4 dB SNR.



(a) SNR = -10 dB

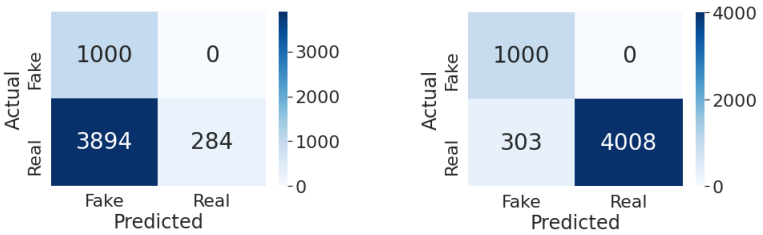
(b) SNR = -4 dB

Figure 5.4. SGAN dense classifier performance with SNR at (a) -10 dB, (b) -4 dB. Source: [38].

5.6.4 Additional networks

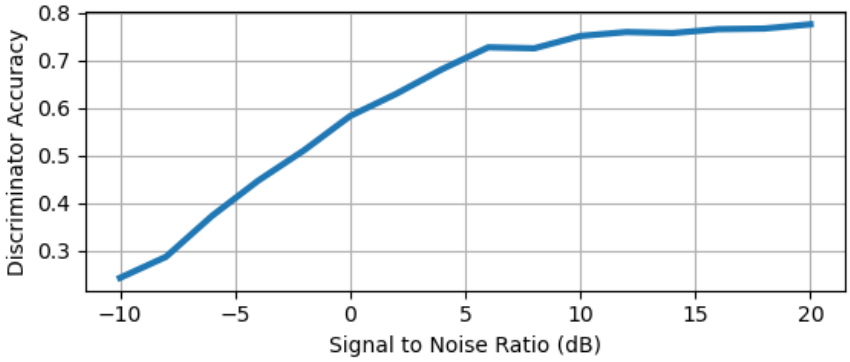
To compare the performance of the SGAN dense classifier, we constructed three additional networks. First, we used another SGAN classifier, but use convolutional layers instead of fully connected layers. This gives us a SGAN CNN classifier. Next, instead of training in a SGAN architecture, we created a standalone dense classifier. This classifier uses the same parameters as our SGAN dense classifier C. Finally, we implemented a standalone CNN classifier, using the same parameters of the SGAN CNN classifier.

By training the SGAN CNN classifier, we also trained a SGAN CNN discriminator. Unfortunately the CNN discriminator did not perform as well as the SGAN dense discriminator. As shown in Figure 5.5, the CNN discriminator did not correctly identify all the legitimate users as "Real". However, at all SNR values, the CNN was able to identify the malicious user as "Fake", so there may be a use case where this is desirable behavior even though it incorrectly prevents some number of users from successfully authenticating.



(a) SNR = -10 dB

(b) SNR = 20 dB



(c)

Figure 5.5. SGAN CNN discriminator performance for (a) -10 dB, (b) 20 dB, (c) -10 dB to 20 dB. Source: [38].

Figure 5.6 shows the results of the various classifier testing after training. All the neural networks reach 100% accuracy with sufficient SNR. The standalone dense classifier trained for 125 epochs and obtained almost 100% accuracy for all SNR except -10 dB. At -10 dB, the standalone dense classifier was 99.929% accurate. The standalone CNN classifier trained for 667 epochs and had very similar performance to the SGAN dense classifier. Finally, the SGAN CNN classifier trained for 30 epochs, and lagging the others, reached 100% accuracy at 6 dB.

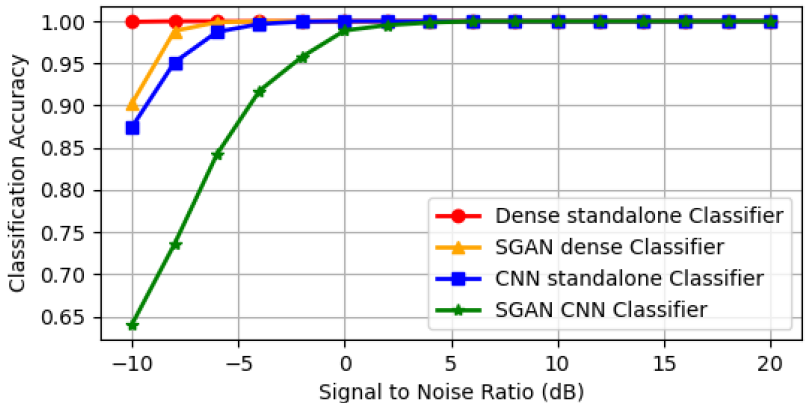


Figure 5.6. Classifier accuracy vs SNR. Source: [38].

Where the discriminators were not able to differentiate between legitimate and generator-produced samples with increased noise levels, the classification results show that the classifiers are able to differentiate among users at these same SNR values. The reason for this is that the generators' samples at low SNR closer approximate the sample distribution from the authentic dataset, making training difficult for the discriminators. Contrast with the classifiers' training where they only receive samples from the dataset and learns the features relevant to the 14 transmitters' CSI.

Recalling the performance of the LOF machine learning algorithm in Chapter 3, we attempted to apply LOF to the task of discriminating the multi-subcarrier CSI samples. Because the implementation from [62] is limited to two dimensions, we flattened the input from a three dimensional tensor to a two dimensional tensor. Unfortunately, the LOF algorithm did not perform well at any SNR, only reaching 21.4% accuracy at 20 dB SNR with the confusion matrix shown in Figure 5.7.

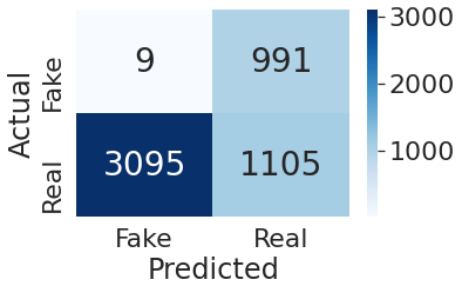


Figure 5.7. LOF confusion matrix with SNR = 20 dB.

5.7 Summary

In this chapter, we demonstrated a SGAN implementation that achieved 100% authentication accuracy for SNR greater than 4 dB and classification accuracy for SNR greater than -4 dB. We improved our architecture from Chapter 4 to take advantage of additional RF features inherit in the multiple-subchannel MIMO wireless communication environment.

We showed how the use of a SGAN can be employed at millimeter wave frequencies with multiple subcarriers in this chapter. Our simulation results illustrated that with a very small percentage of labeled CSI samples, accurate discrimination between legitimate and adversary transmitters as well as classification can be made with a SGAN dense classifier for SNR values greater than 4 dB. An adversary may achieve a high degree of accuracy when spoofing a legitimate transmitter, but by measuring the magnitude and phase of the CSI elements, we have shown that our system can differentiate transmitter CSI from positions 10 cm apart.

Our proposed SGAN-based physical-layer authentication system can be implemented to provide high authentication accuracy at even low SNR values. The system first uses the SGAN-trained discriminator to allow only trusted transmitters to authenticate. The SGAN-trained classifier then identifies the trusted transmitter and can be used to allow the user a tailored degree of access. While this was the goal in Chapter 4, our proposed discriminator in that chapter was not able to achieve high authentication accuracy unless the SNR was relatively high (greater than 26 dB SNR). In this chapter, we were able to realize our goal by creating more robust neural networks and providing additional input features by using

CSI from multiple subcarriers.

As in Chapter 4, we saw that although the SGAN-trained classifiers required less epochs to train with a small amount of labeled samples. While the dense classifier had the best performance overall, the classifiers only can be used to classify legitimate transmitters. By virtue of the training architecture, the SGAN is able to first discriminate and then classify. The networks do not have a mechanism to create a discriminator and must train this network separately.

While our results are promising, up until this point in the dissertation, our system models have used the static environment, where the magnitude and phase of the complex CSI elements don't appreciably change. While we've been able to use this to our advantage and incorporate the features of the real and imaginary parts of the complex elements, the mobile channel is dynamic and time-variant. In the next chapter, we examine a mobile environment where we conduct physical-layer authentication using CSI measured as time-series data.

CHAPTER 6:
Mobile Channel Prediction and Transmitter Authentication

Prediction of channel characteristics allows for a variety of techniques to make efficient use of a wireless link [37], [152]. By applying adaptive measures such as changing transmitter power, modulation, channel coding, antenna diversity, etc., improved performance can be achieved between transmitter and receiver while ensuring symbol error rate remains at acceptable levels [153]. CSI estimation and prediction is crucial in adaptive systems [154] and has been a topic of much research. Notably, as communication systems have evolved and become more complex with the inclusion of MIMO antenna architectures and OFDM with multiple subcarriers, ANNs have been shown to outperform traditional linear algorithms [155] resulting in several neural network-based solutions [78], [156]–[158] to channel prediction.

In this chapter, we leverage machine learning to predict the future characteristics of a channel and use this prediction to make an authentication decision. We explore physical layer authentication and ANNs with the objective of correctly authenticating legitimate devices and denying authentication to illegitimate transmitters. Since elements of the CSI matrix are unique to the coupled positions of the receiver and transmitter in a fading channel environment, we exploit these features to ensure transmission from legitimate transmitters are authenticated, while illegitimate transmitters are denied authentication.

This chapter includes material adapted from work published and work to be published by the author. Already published, this chapter contains revised material from "Channel Prediction and Transmitter Authentication with Adversarially-Trained Recurrent Neural Networks" by Ken St. Germain and Frank Kragh published in the IEEE Open Journal of the Communications Society [99]. Additionally, this revised material is also included from "Mobile Physical-Layer Authentication Using Channel State Information and Conditional Recurrent Neural Networks", by Ken St. Germain and Frank Kragh to be published in the 93rd Vehicular Technology Conference: VTC2021-Spring [71].

6.1 Channel Model

In a dynamic environment, the CSI matrix $\boldsymbol{H}(t_s)$, changes for every sample time, where $t_s \in \{t_1, t_2, \ldots t_S\}$ where S is the number of samples. Likewise, the magnitude of each CSI element $|h_{n,m}|_{N\times M}$ changes. Following Jiang and Schotten in their work in channel prediction [159], we will use the magnitude of the CSI elements to discern the transmitter to be authenticated using the intuitive notion that magnitude will change more slowly than the phases. We construct a time-varying channel gain tensor $\boldsymbol{Q}(t_s) = [|h_{n,m}(t_s)|]_{N\times M\times S}$ where we further decompose the CSI elements into a two-dimensional matrix,

$$\tilde{\boldsymbol{Q}} = \begin{bmatrix} |h_{1,1}(t_1)| & |h_{1,1}(t_2)| & \ldots & |h_{1,1}(t_S)| \\ |h_{1,2}(t_1)| & |h_{1,2}(t_2)| & \ldots & |h_{1,2}(t_S)| \\ \vdots & \vdots & \vdots & \vdots \\ |h_{N,M}(t_1)| & |h_{N,M}(t_2)| & \ldots & |h_{N,M}(t_S)| \end{bmatrix}, \tag{6.1}$$

where $\tilde{\boldsymbol{Q}}$ is a time-series representation of the channel gain matrix where one dimension is time, and the other dimension is spatial. Each column in Equation 6.1 is a vector representation of the magnitude of the CSI elements for a particular channel sampled at one instant in time.

6.1.1 Channel prediction

The time-series channel gain matrix $\tilde{\boldsymbol{Q}}$ is obtained by measuring the magnitude of the CSI elements within $\boldsymbol{H}(t)$. This can be accomplished through pilot symbols where the complex value of the transmitted symbol is known a priori by the receiver. Pilot symbols can be extracted from dedicated pilot channels in an OFDM framework or in the preamble of a transmitted packet. We consider a fading channel where the channel changes from symbol to symbol. We assume channel variations from symbol to symbol are correlated for a short time, consistent with mobile channel measurements [37].

We achieve channel prediction by first measuring a sequence of S channel gain vectors from a series of received transmissions, and then forecasting the next sequence of predictions P. For example, consider Figure 6.1a and Figure 6.1b, where a sequence of S received channel gain vectors are measured, and P channel gain vectors are predicted. This illustrates a single-input single-output channel with a CSI matrix of dimension 1×1. Increasing the

number of receiver and transmitter antennas likewise increases the dimensions of $\boldsymbol{H}(t)$. Although there will be correlation for each of the individual CSI element magnitudes in time, we expect different CSI elements to have independent channel gains.

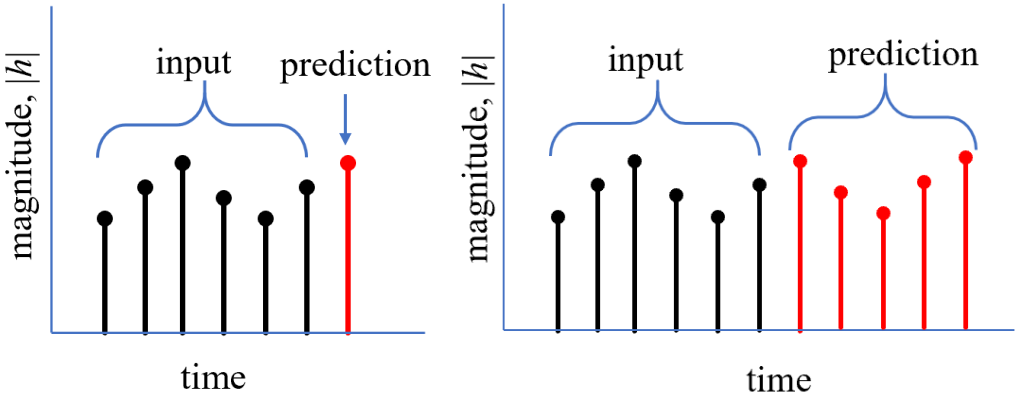


Figure 6.1. Impulse response predictions for (a) $S = 6$, $P = 1$ and (b) $S = 6$, $P = 5$. Source: [99].

6.1.2 CGAN

When training a vanilla GAN in an unsupervised learning architecture, the discriminative model $\mathcal{D}$ is a binary classifier that receives unlabeled authentic samples from the training dataset or fake samples generated by the generative model $\mathcal{G}$. The generative model creates fake samples based on random variable input and the parameters in $\mathcal{G}$.

For the CGAN, we also provide conditional information to the discriminator and generator. The conditional information is the previous magnitudes of the CSI elements associated with $t_s \in \{t_1, t_2 \ldots t_S\}$. The vector $\boldsymbol{q}(t_1)$ is the magnitude of the CSI elements at time t_1, and $\tilde{\boldsymbol{Q}} = \{\boldsymbol{q}(t_1),\ \boldsymbol{q}(t_2), \ldots, \boldsymbol{q}(t_S)\}$. The output of the discriminator is the probability that $\mathbf{X}$ is the channel gain matrix composed of channel gain vectors at time $t_p \in \{t_{S+1}, t_{S+2}, \ldots, t_{S+P}\}$, given the previous channel gain matrix $\tilde{\boldsymbol{Q}}$. The number of future channel measurements is P, while the number of historic channel measurements is S.

The generator output $G(\mathbf{z}|\tilde{\boldsymbol{Q}})$ is a channel gain matrix approximating $\mathbf{X}$, given that $\tilde{\boldsymbol{Q}}$ was the matrix composed of previous channel gain vectors. Latent points from a random noise

distribution are used to create the vector $\mathbf{z}$. Therefore, Equation 1.20 becomes

$$\begin{aligned}\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{X}\sim p_{data}(\mathbf{X})}[\log D(\mathbf{X}|\tilde{\boldsymbol{Q}})] \\ + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\tilde{\boldsymbol{Q}})))].\end{aligned} \tag{6.2}$$

where $\mathcal{D}$ calculates $D(\mathbf{X}|\tilde{\boldsymbol{Q}})$, and $\mathcal{G}$ produces $G(\mathbf{z}|\tilde{\boldsymbol{Q}})$. Figure 6.2 illustrates a CGAN during training. Using Equation 6.2, the loss functions that should be minimized for the discriminator and generator are

$$\begin{aligned}J^{(\mathcal{D})}_{\text{minimax}} &= -\mathbb{E}_{\mathbf{X}\sim p_{data}(\mathbf{X})}[\log D(\mathbf{X}|\tilde{\boldsymbol{Q}})] \\ &\quad - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\tilde{\boldsymbol{Q}})))] \\ J^{(\mathcal{G})}_{\text{minimax}} &= \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\tilde{\boldsymbol{Q}})))]\end{aligned} \tag{6.3}$$

where $J^{(\mathcal{D})}_{\text{minimax}}$ is the sign-opposite of Equation 6.2, since the result of $J^{(\mathcal{D})}_{\text{minimax}}$ should be minimized, and Equation 6.2 calls for the discriminator network to maximize the value function. The term containing $D(\mathbf{X}|\tilde{\boldsymbol{Q}})$ from Equation 6.2 is omitted from $J^{(\mathcal{G})}_{\text{minimax}}$ in Equation 6.3 since the generator is not connected to the discriminator while the samples from the dataset $\mathbf{X}$ are being passed to the discriminator.

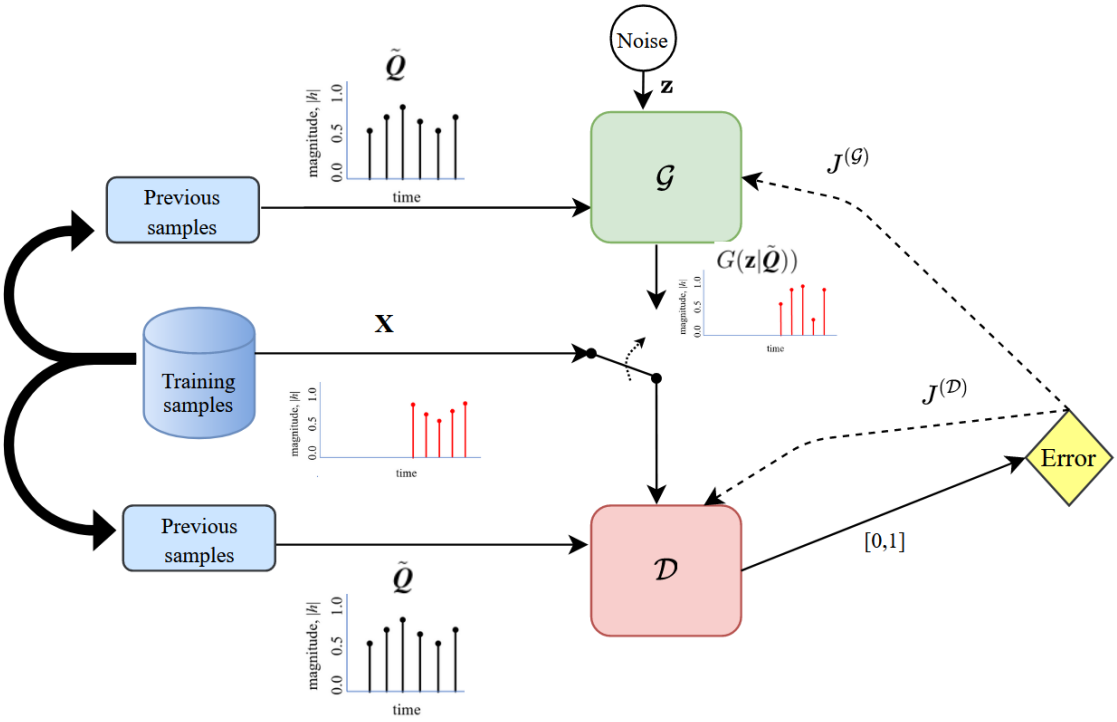


Figure 6.2. CGAN training architecture. Source: [99].

Alternative loss functions can also be used for the discriminator and generator networks. In this work, we use the loss functions in Equation 6.3 and also train networks in a CGAN framework using a MSE loss for the generator network. In [160], Mao et al. introduced the least-squares generative adversarial network (LSGAN) in order to address the issue of vanishing gradients while training GANs by applying a MSE loss to the discriminator. To compare the performance of the discriminator and generator with alternative loss functions to the vanilla cross-entropy loss for both networks, we also use the loss functions from [160], i.e.,

$$
\begin{aligned}
J_{\text{MSE}}^{(\mathcal{D})} &= \mathbb{E}_{\mathbf{X}\sim p_{data}(\mathbf{X})}[(D(\mathbf{X}|\tilde{\boldsymbol{Q}})-1)^2] \\
&\quad + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[(D(G(\mathbf{z}|\tilde{\boldsymbol{Q}}))-0)^2] \\
J_{\text{MSE}}^{(\mathcal{G})} &= \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[(D(G(\mathbf{z}|\tilde{\boldsymbol{Q}})))-1)^2].
\end{aligned}
\tag{6.4}
$$

Additionally, we'll train the CGAN with a hybrid of loss functions: the binary cross-entropy (BCE) loss in Equation 6.3 for the discriminator, and use a MSE loss for the generator, as shown in

$$\begin{aligned} J^{(\mathcal{D})}_{\text{hybrid}} &= -\mathbb{E}_{\mathbf{X}\sim p_{data}(\mathbf{X})}[\log D(\mathbf{X}|\tilde{\boldsymbol{Q}})] \\ &\quad - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\tilde{\boldsymbol{Q}})))] \\ J^{(\mathcal{G})}_{\text{hybrid}} &= \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[(D(G(\mathbf{z}|\tilde{\boldsymbol{Q}}))) - 1)^2]. \end{aligned} \tag{6.5}$$

6.2 Simulation

In this section, we discuss the channel model and the architecture of the neural networks under consideration. We continue with our evaluation methodology and present our results.

6.2.1 System model

We consider an independent and identically distributed 2×2 MIMO Rayleigh multipath fading channel with path delay profile shown in Table 6.1. The path delay profile is specified by the extended vehicular A (EVA) model in [161]. The power spectral density used is the Clarke model [162], with a maximum Doppler shift of 70 Hz. We simulated our channel with a 10 kHz sample rate and scaled the amplitude of the CSI elements [0,1]. To train the networks, 60% of the samples were used for training and the remaining 395 samples were reserved for testing. Figure 6.3 shows the magnitude of the CSI elements and the partitioning of the dataset.

Table 6.1. Channel model power delay profile. Source: [161].

delay number, l	1	2	3	4	5	6	7	8	9
tap delay, τ_l, (ns)	0	30	150	310	370	710	1090	1730	2510
relative power (dB)	0.0	-1.5	-1.4	3.6	-0.6	-9.1	-7.0	-12.0	-16.9

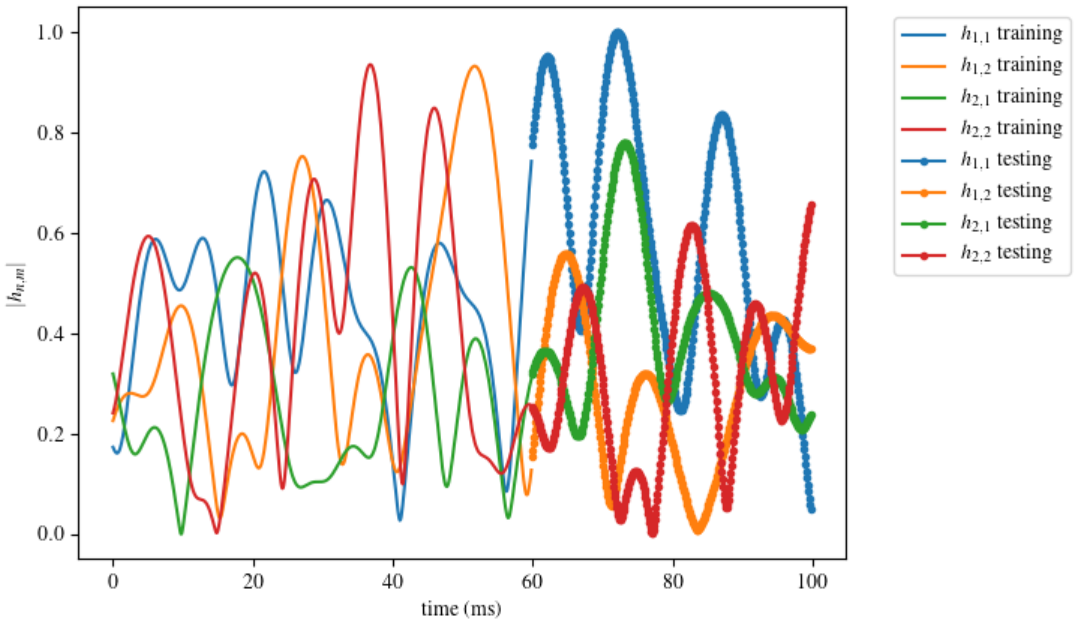


Figure 6.3. CSI element magnitude in channel model. Source: [71].

The training and testing data was sequenced so that every input sample consisted of S time steps and four amplitude features each. The target variable for training and testing was likewise sequenced so that every sample consisted of P future time steps and four amplitude features each.

6.2.2 Neural network development

The RNNs we considered were composed of LSTM cells or GRU cells. To avoid overfitting, all the networks used a recurrent dropout of 0.5 within the LSTM and GRU cells, and the Adam optimizer [163]. The input for the standalone LSTM and GRU networks was the matrix composed of previous channel response vectors $\tilde{\boldsymbol{Q}}$. The LSTM and GRU CGANs had a conditional input consisting of the concatenation of $\tilde{\boldsymbol{Q}}$ and a random seed tensor $\mathbf{z}$. The CGAN generator input is made by combining $\tilde{\boldsymbol{Q}}$ and latent points from $\mathcal{U}(0,1)$. The discriminator input $\boldsymbol{X}$, is either the target variable from the dataset or a channel gain matrix created by the generator $G(\mathbf{z}|\tilde{\boldsymbol{Q}})$. A summary of the architecture of the neural networks is

shown in Table 6.2.

Table 6.2. Neural networks architecture. Source: [71].

	Input Layer		Hidden Layer			Output Layer		
	Input	Output Size	Units	Type	Output Size	Units	Type	Output Size
CGAN-LSTM								
$\mathcal{D}$	$[\tilde{\boldsymbol{Q}}, \mathbf{X}]$	$(S+P, 4)$	128	LSTM	128	1	Dense	1
$\mathcal{G}$	$[\tilde{\boldsymbol{Q}}, z]$	$(S+100, 4)$	128	LSTM	128	$P \times 4$	Dense	$(P, 4)$
CGAN-GRU								
$\mathcal{D}$	$[\tilde{\boldsymbol{Q}}, \mathbf{X}]$	$(S+P, 4)$	128	GRU	128	1	Dense	1
$\mathcal{G}$	$[\tilde{\boldsymbol{Q}}, z]$	$(S+100, 4)$	128	GRU	128	$P \times 4$	Dense	$(P, 4)$
LSTM	$\tilde{\boldsymbol{Q}}$	$(S, 4)$	128	LSTM	128	$P \times 4$	Dense	$(P, 4)$
GRU	$\tilde{\boldsymbol{Q}}$	$(S, 4)$	128	GRU	128	$P \times 4$	Dense	$(P, 4)$

For the standalone LSTM and GRU networks, the loss function used was MSE,

$$\begin{aligned} \text{MSE} =& \frac{1}{T} \sum_{i=1}^{T} e_i \\ e_i =& \frac{1}{NM} \sum_{n,m} \left(\boldsymbol{Q}_{n,m} - \hat{\boldsymbol{Q}}_{n,m} \right)^2 \end{aligned} \tag{6.6}$$

where T is the number of samples, $\boldsymbol{Q}$ is the true value for the channel gain tensor at each time t_p, and $\hat{\boldsymbol{Q}}$ is the networks' predicted value for each t_p. The element-wise mean squared error between $\boldsymbol{Q}$ and $\hat{\boldsymbol{Q}}$ for sample i is e_i. The CGANs used the loss functions described in Equations 6.3, 6.4, and 6.5.

6.2.3 Evaluation criteria

To assess relative performance, we used MSE between the predicted channel gain and the true channel gain as our metric. After training the networks, we evaluated the standalone LSTM and GRU networks and the generator networks from the CGANs. After gathering MSE measurements, we can establish a threshold MSE_T for authentication.

$$\begin{aligned} &\text{MSE} \leq MSE_T \rightarrow \text{authenticate} \\ &\text{MSE} > MSE_T \rightarrow \text{do not authenticate} \end{aligned} \tag{6.7}$$

If the error between the predicted and actual channel measurement is less than or equal to the threshold, the transmitter is authenticated. Otherwise, the transmission is rejected.

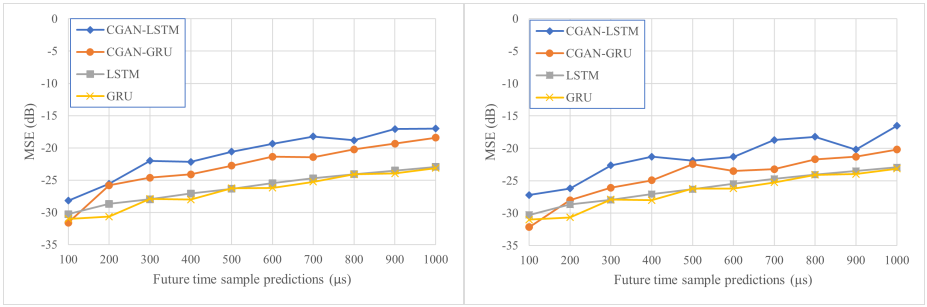
We created additional channel responses representing four other users using the same EVA power delay profile shown in Table 6.1. There were 395 samples in each group of channel responses. These 1,580 samples from the new profiles were not seen by the networks before testing and represent trials to assess the networks' ability to recognize illegitimate transmitters.

Since the CGANs train a discriminator and a generator, we can also use the discriminator to make the authentication decision. The discriminator output will indicate whether the input is "Real" or "Fake". Since the discriminator is trained on the channel gain profile shown in Figure 6.3, these samples should be assessed as "Real", while the samples from the additional profiles should be assessed as "Fake". If the sample is "Real", we authenticate, and if they are "Fake" we will not authenticate.

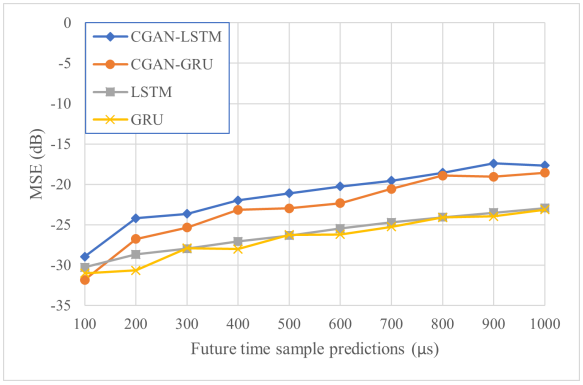
6.3 Results

In this section, we present the results of our simulations applying the BCE loss from Equation 6.3, the MSE loss from Equation 6.4, and the hybrid loss from Equation 6.5 to the CGAN networks. The standalone LSTM and GRU networks both use the MSE loss in Equation 6.6.

With S=5, P is increased from 1 to 10, allowing the networks to predict future MIMO channel response magnitudes. With our 10 kHz sample rate, the amount of time elapsed for every prediction step P is 100 μs. Except for P=1 where the CGAN-GRU predicted the channel better than any other network configuration, the standalone networks performed better than the generator networks in the CGANs, achieving the least amount of error as shown in Figure 6.4.



(a) (b)

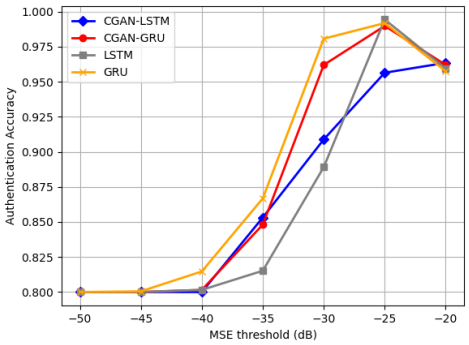


(c)

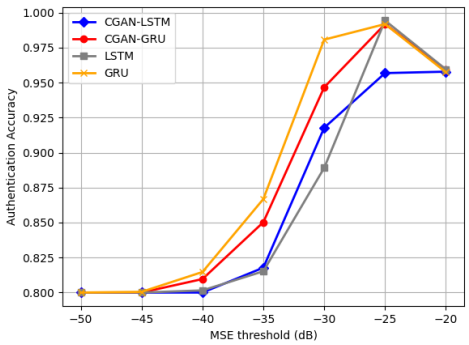
Figure 6.4. Mean square error performance with S=5, P from 1 to 10 for (a) BCE loss, (b) MSE loss, and (c) hybrid loss for CGAN networks and MSE loss used for the standalone networks, where S is the number of previous channel responses, and P is the number of future channel predictions. Source: [99].

We applied Equation 6.7 with S=5 and P=1 for MSE_T values -50 dB $\leq MSE_T \leq -20$ dB to determine which transmitter associated with a channel gain profile to authenticate. Although accuracy increases for all networks up to MSE_T of -25 dB as shown in Figure 6.5, false positives begin to arise at -25 dB and greater, which would allow illegitimate transmitters to authenticate. At -30 dB, there are no false positives, as shown in the upper right quadrants of Figure 6.6a through Figure 6.6d. As shown in Figure 6.5, the standalone GRU network was most accurate in authenticating using MSE from channel prediction, achieving 98.1% accuracy. The generator from the CGAN-GRU using BCE loss was the next-best configuration, reaching 96.2% accuracy in Fig. 6.5a. Note that in Figure 6.5, no model

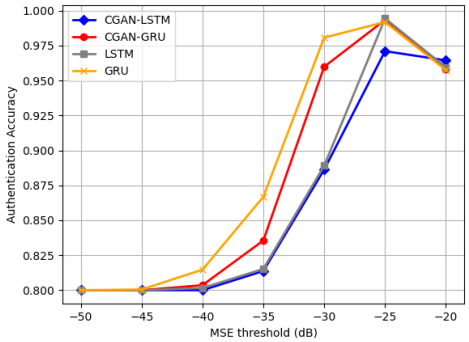
performs worse than 80% when the MSE is -45 dB or less. This is an artifact of the testing dataset and the byproduct of all the samples – the 395 legitimate samples and the 1,580 illegitimate samples – being categorized as “Fake”.



(a) BCE loss



(b) MSE loss



(c) Hybrid loss

Figure 6.5. Authentication performance using mean squared error threshold method for $S = 5$, $P = 1$ for CGAN networks using (a) BCE loss, (b) MSE loss, and (c) hybrid loss and MSE loss used for the standalone networks, where S is the number of previous channel responses, and P is the number of future channel predictions. Source: [99].

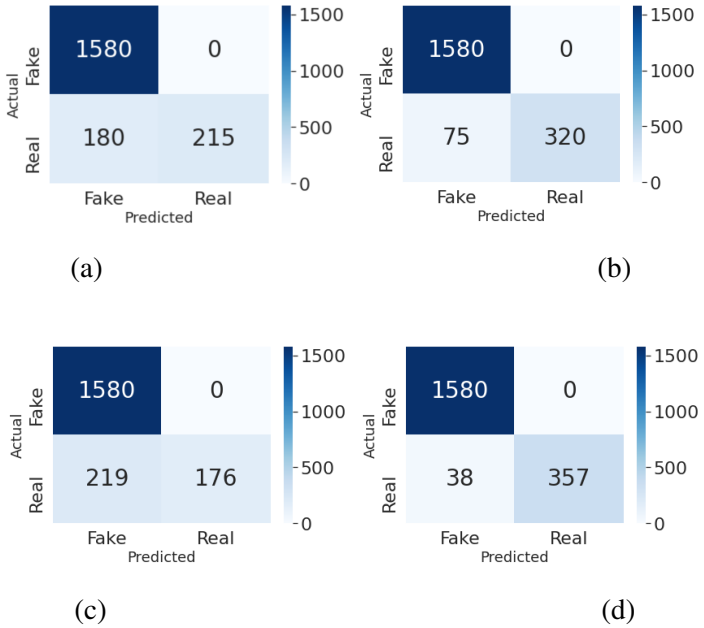


Figure 6.6. Confusion matrices showing authentication performance using mean square error threshold at −30 dB for (a) CGAN-LSTM trained with BCE loss, (b) CGAN-GRU trained with BCE loss, (c) LSTM, and (d) GRU networks. Source: [99].

In addition to authenticating based on channel prediction and applying a threshold to the MSE, the CGAN-trained discriminators can be used for authentication. For $S = 5$ and $P = 1$, Fig. 6.7 shows the confusion matrices for the CGAN-LSTM and CGAN-GRU discriminator networks using BCE loss, MSE loss, and the hybrid loss. Compared to the MSE authentication method, the CGAN-GRU trained with the hybrid loss improves its authentication performance from 96.2% to 98.5% as shown in Figure 6.7f. This matches the authentication accuracy of the standalone GRU while still preventing any illegitimate transmitters from authenticating.

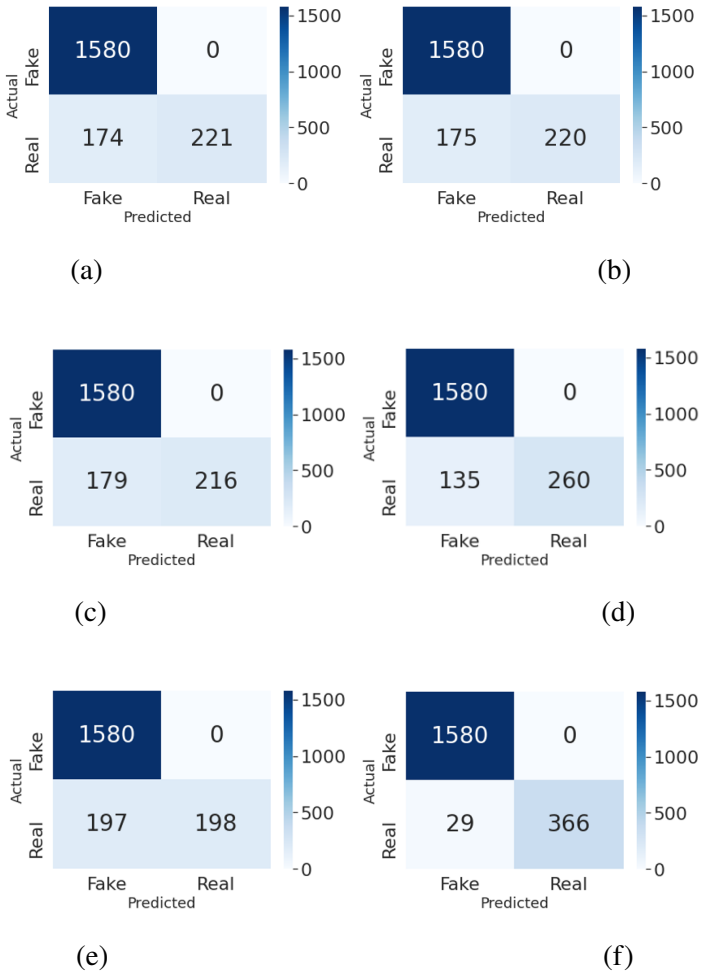


Figure 6.7. Confusion matrices for discriminators performing authentication trained through (a) CGAN-LSTM with BCE loss, (b) CGAN-GRU with BCE loss, (c) CGAN-LSTM with MSE loss, (d) CGAN-GRU with MSE loss, (e) CGAN-LSTM with hybrid loss, and (f) CGAN-GRU with hybrid loss. Source: [99].

Although we have no false positive results using $MSE_T = -30$ dB or the CGAN discriminators as shown in Figure 6.6 and Figure 6.7, respectively, this does not suggest that an illegitimate transmitter will always be denied authentication. However, as seen in our results, all 1,580 testing samples were correctly categorized as "Fake", implying the probability of inadvertently authenticating an illegitimate transmitter is on the order of 0.001 or less.

6.4 Summary

By constructing a time-series channel gain matrix based on the measured magnitude of the received CSI elements, we were able to show how physical-layer authentication can be accomplished in a mobile channel environment. We used RNNs based on LSTM and GRU cells for channel prediction and incorporated RNNs into a GAN framework to accomplish both channel prediction and binary classification. For the tasks of predicting channel responses and classifying whether or not a transmitter should be authenticated based on received CSI element magnitudes, architectures using GRU cells were superior to networks that used LSTM cells.

The standalone RNNs were able to predict future channel responses with less error than the CGAN-trained RNNs. This led to the standalone networks performing more accurately using the MSE threshold authentication technique. By using the discriminator from the GRU-based CGAN trained with the hybrid loss, the authentication accuracy of the standalone GRU RNN was matched.

The value of MSE_T was set to prevent illegitimate transmitters from authenticating. Unfortunately, the channel conditions may change that result in a need to dynamically adjust the threshold to prevent inadvertent authentication. Using the CGAN discriminator as the basis of authentication does not require the constant manipulation of a threshold variable. However, the process of training the CGAN system must be persistent in order to adapt to changing channel conditions.

To implement a CGAN-trained discriminator for mobile physical-layer authentication, a pair of discriminators should be initially trained. Following training, the two discriminators should alternate in making the authentication decision. While one discriminator is processing the real-time received CSI samples, the other discriminator uses those same samples as the conditional information for the next series of real-time received CSI samples. The two discriminators then switch roles, providing the receiver with uninterrupted authentication decisions.

CHAPTER 7:
Conclusion

In this dissertation, we demonstrated the use of adversarial machine learning to characterize RF transmitters for the purpose of physical-layer authentication. We also examined the application of several machine learning techniques for use in the RF domain. Many physical-layer authentication techniques are not effective when used in the presence of nefarious users who are able to spoof the underlying physical-layer authentication traits. Our approach used CSI as a means to prevent unauthorized access since received CSI in a MIMO system is difficult for a third party to estimate, let alone emulate. However, a determined adversary may be able to alter timing, power levels, and other transmitter characteristics in the RF path in order to match the received CSI of a legitimate transmitter. To counter the potential of malicious behavior in wireless systems, we explored the use of adversarial learning.

Our implementation of adversarial machine learning can improve cybersecurity through the use of physical-layer authentication. We established the theoretic construct of using GANs for physical-layer authentication through the use of stochastic and software-based models to represent the wireless domain. However, more challenges and research remain. The next iterative step is the live capture of RF signals and processing the received CSI. Additionally, refinement of the neural networks and hyperparameter optimization to the input features of actual RF signals would be appropriate.

7.1 Summary

In this chapter, we review the results of this dissertation with respect to our overarching goal of using received CSI from the wireless channel to differentiate and ultimately authenticate transmitters. We conclude with future research that can be accomplished to further enhance cybersecurity through physical layer authentication using the GAN framework.

7.1.1 Authentication using CSI

In Chapter 3, we demonstrated the use of adversarial machine learning for the task of robust RF transmitter characterization. We showed how channel information in the form of CSI

could be used as a method to provide physical-layer authentication.

Our analysis illustrated that the probability of accidentally authenticating other transmitters decreases as receive and transmit antennas are increased and a threshold value is judiciously applied. We developed a GAN trained on a dataset of CSI matrices to perform physical-layer authentication in an adversarial environment. After training less than 50 epochs, the discriminator reached 100% accuracy on two separate testing datasets, implicitly determining appropriate thresholds for received CSI matrix elements. When a nefarious user was able to closely match the CSI from a legitimate transmitter, higher SNR was needed to achieve accurate results.

We compared the results of the GAN to a variety of one-class machine learning algorithms, including LOF, iForest, and OC-SVM. Across all levels of AWGN, the LOF algorithm reached and maintained 100% accuracy at the lowest SNR value. The use of LOF is well suited to low-dimensionality challenges such as we explored with a single transmitter and single carrier frequency. However, the performance of LOF degrades as the additional dimensions of features are introduced [62].

7.1.2 SGAN for Classification using CSI

In Chapter 4 demonstrated how physical-layer authentication can be accomplished in a flat-fading single-carrier wireless channel environment. We used an SGAN architecture to categorize multiple transmitters using CSI. Our implementation of SGAN-trained classifiers required fewer labeled training samples to reach high accuracy levels. With only 50 labeled samples, our CNN SGAN classifier reached 100% accuracy at a 4 dB SNR. By reducing the number of labeled samples, data overhead and processing expenditure can be reduced.

We showed how machine learning, specifically the use of deep neural networks, can be used to classify transmitters by MIMO CSI as a method to provide physical layer authentication. Our simulation results illustrated that with a very small percentage of labeled CSI samples, accurate classification can be made with a SGAN-trained classifiers for SNR values greater than 4 dB. During the initial authentication setup between two transceivers, a small number of labeled samples is desired to minimize overhead. However, if more labeled samples are available to the receiver, the standalone classifier produces better results at all SNR levels.

Chapter 4 provided positive results regarding the use of adversarial machine learning for the application of physical-layer authentication using CSI. We provided a method for differentiation and classification of transmitters using received CSI. Unfortunately, the SGAN-trained discriminators did not perform well at low SNR levels. Where the classifiers could reach 100% authentication accuracy at 4 dB and greater, the best discriminator network we implemented was able to reach 100% at 26 dB. By providing additional input features for the discriminator to extract and providing a more robust architecture with which to optimize the weight parameters, we improved accuracy as shown in Chapter 5.

7.1.3 Classification using CSI at Millimeter Wave Carrier Frequencies

By providing the neural network additional input features in the form of CSI measurements across multiple subcarriers and adding additional capability to our SGANs, we improved upon the results presented in Chapter 4. Our proposed SGAN-based physical-layer authentication system can be implemented to provide high authentication accuracy at even low SNR values. The system first uses the SGAN-trained discriminator to allow only trusted transmitters to authenticate. The SGAN-trained classifier then identifies the trusted transmitter and can be used to allow the user a tailored degree of access. While this was the goal in Chapter 4, our proposed discriminator in that chapter was not able to achieve high authentication accuracy unless the SNR was relatively high (greater than 26 dB SNR). In this chapter, we were able to realize our goal by creating more robust neural networks and providing additional input features by using CSI from multiple subcarriers.

We showed how the use of a SGAN can be employed at millimeter wave frequencies with multiple subcarriers in Chapter 5. Our simulation results illustrated that with a very small percentage of labeled CSI samples, accurate classification can be made with a SGAN dense classifier for SNR greater than -4 dB. Likewise the densely-connected discriminator also achieved high accuracy for SNR greater than 4 dB. Based on this result, we can pair the trained discriminator and classifier to: (1) determine whether or not to authenticate a transmitter, and (2) classify the trusted transmitter to provide access at the appropriate level.

The dataset we used was based on ray-tracing software and provided a high degree of accuracy with respect to modeling the environment. An adversary may achieve a high degree of accuracy when spoofing a legitimate transmitter, but by retaining the magnitude

and phase of the CSI elements, we showed that our system can differentiate transmitter CSI from positions 10 cm apart.

7.1.4 Physical Authentication in a Mobile Environment

In Chapter 6, we explored various RNN architectures to make authentication decisions at the physical layer in a mobile MIMO multipath channel. The previous chapters assumed a static environment and treated the channel as invariant.

Alternative loss functions for the CGAN networks were also explored with BCE outperforming MSE in an LSGAN configuration and a hybrid loss where the discriminator used BCE and the generator used MSE. Varying the amount of channel responses and channel prediction time, regression performance was measured using mean square error against the ground truth. Using a MSE threshold of -30 dB, and without any false positive errors, the standalone GRU network achieved the highest authentication performance at 98.1%, followed by the CGAN-GRU at 96.2% accuracy. Using a CGAN-trained discriminator with a hybrid loss function, the network was able to authenticate at a 98.5% rate. For the tasks of predicting channel responses and classifying whether or not a transmitter should be authenticated, architectures using GRU cells were superior to networks that used LSTM cells.

The implementation of our proposed mobile environment can be accomplished using the CGAN-trained discriminator. We recommend two discriminators be used, so that the magnitude of the CSI elements currently being measured can be used as conditional information for future physical-layer authentication. Both discriminators are trained and they are then used in an alternating fashion to produce continuous authentication decisions on received channel gain vector samples.

7.2 Future Work

While we created our neural networks using best practices in the field of machine learning and we made changes to the hyperparameters of our neural networks based on their training performance, we may have not used the optimal settings for our tasks. There may be better hyperparameters chosen to improve the performance as presented in this work. The GANs

should be optimized and reevaluated on additional training and test datasets to demonstrate effectiveness in a variety of wireless environments.

Additionally, transfer learning should be explored. We saw that the standalone classifiers in Chapters 4 and 5 took longer to train than the SGANs, but were more accurate given enough training samples. Transfer learning of the weights from a SGAN-trained classifier to a standalone network could reduce the training time of the standalone classifier and increase overall performance.

Future work for the CGAN in Chapter 6 includes application of the methods in this dissertation to multi-carrier channels as well as classifying a variety of time-series channel gain profiles to enable multiple transmitter authentication. In Chapter 6, we determined whether a single transmitter using a single carrier should be authenticated, however, we did not use multiple subcarriers or use the CGAN to perform transmitter classification.

Finally, throughout this dissertation, none of the frameworks were implemented using live wireless transmissions. Simulated datasets were created using statistical models that represented the RF channel. We also included the use of the DeepMIMO dataset that was developed using ray-tracing in Chapter 5. Software defined radio systems such as USRPs can be used to provide MIMO transmitters and receivers and also provide capability to implement the neural network. Live-captured signals can be used to provide greater insight into real-world application of physical-layer authentication how adversarial architectures can be used to enhance the security of wireless systems.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX: Neural Network Essentials

This appendix provides an introduction to some of the fundamental elements regarding neural networks with emphasis on concepts, terms, and details required for a full understanding of the work described in this dissertation. The topics herein include the mathematical operations in feedforward neural networks and common terms associated with the training process.

A.1 Linear Regression

Consider a regression problem where the task is to predict a scalar value y based on an independent variable $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)^T$, where T is the transpose operation. We seek to find a function f to fit the training data to a simple linear model as suggested in [54],

$$y \approx f_\theta(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_N x_N = \sum_{n=1}^{N} \theta_n x_n + \theta_0. \tag{A.1}$$

The intercept term θ_0 is known as the *bias*, and collectively the parameters of the vector $\boldsymbol{\theta}$ are called *weights* [164]. For convenience, we can add another term $x_0 = 1$ to the vector $\boldsymbol{x}$. Therefore, our new input vector is $\tilde{\boldsymbol{x}} = (x_0, x_1, x_2, \ldots, x_N)^T$ and we rewrite Equation A.1 as

$$f_\theta(\tilde{\boldsymbol{x}}) = \sum_{n=0}^{N} \theta_n \tilde{x}_n = \tilde{\boldsymbol{x}}^T \boldsymbol{\theta}, \tag{A.2}$$

In general, instead of a scalar y, the dependent variable can be $\boldsymbol{y} = (y_1, y_2, \ldots, y_K)$, a vector of length K. In that case, $\boldsymbol{\theta}$ will be an $(N+1) \times K$ coefficient matrix, and $\boldsymbol{f}_\theta(\tilde{\boldsymbol{x}})$ will be a vector function [164].

To learn the parameters in $\boldsymbol{\theta}$, we use a *loss function* that measures how close the predicted values in $f_\theta(\tilde{\boldsymbol{x}})$ are to the actual vector $\boldsymbol{y}$. A common function to use is the squared error function [165]:

$$J(\boldsymbol{\theta}) = \frac{1}{2} \sum_{m=1}^{M} \left(\boldsymbol{f}_\theta(\tilde{\boldsymbol{x}}^{(m)}) - \boldsymbol{y}^{(m)} \right)^2, \tag{A.3}$$

where M is the number of training examples in the training set and the pair $\left(\tilde{\boldsymbol{x}}^{(m)}, \boldsymbol{y}^{(m)}\right)$ is a single training example. Since $J(\boldsymbol{\theta})$ captures the amount of error in the prediction, we want to select $\boldsymbol{\theta}$ that minimizes $J(\boldsymbol{\theta})$. Since Equation A.3 is a quadratic function, there is a single minimum, however that is not the case with all loss functions. For instance, the binary crossentropy loss function used in generative adversarial networks has a saddle point that represents neither the minimum loss for the discriminator nor the generator neural networks, but value where the discriminator and generator reach equilibrium [67]. After an initial random assignment of values to $\boldsymbol{\theta}$, we will use the *gradient descent* algorithm to gradually adjust $\boldsymbol{\theta}$ to values that hopefully converge to a $\boldsymbol{\theta}$ that minimizes $J(\boldsymbol{\theta})$. Until convergence, the gradient descent algorithm repeatedly updates $\boldsymbol{\theta}$ by

$$\boldsymbol{\theta}_{new} = \boldsymbol{\theta} - \alpha \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{A.4}$$

where α is the *learning rate* and $\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ is the gradient of $J(\theta)$. The learning rate, typically $0 < \alpha < 1$, is a tunable *hyperparameter* that can be used to control the size of the update step for $\boldsymbol{\theta}_{new}$. If the learning rate is too small, it may take a long time to reach convergence where $J(\boldsymbol{\theta})$ is minimized. If the learning rate is too large, the algorithm may not converge at a minimum.

In order to perform Equation A.4, we need to compute $\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$. Following [165], using a single training example $M = 1$ and scalar output $K = 1$ $(\tilde{\boldsymbol{x}}, y)$ we have

$$\begin{aligned} \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= \frac{\partial}{\partial \boldsymbol{\theta}} \frac{1}{2} \left(f(\tilde{\boldsymbol{x}}) - y\right)^2 && \text{(A.5)} \\ &= \left(f(\tilde{\boldsymbol{x}}) - y\right) \frac{\partial}{\partial \boldsymbol{\theta}} \left(\tilde{\boldsymbol{x}}^T \boldsymbol{\theta} - y\right) && \text{(A.6)} \\ &= \left(f(\tilde{\boldsymbol{x}}) - y\right) \tilde{\boldsymbol{x}}. && \text{(A.7)} \end{aligned}$$

Substituting Equation A.7 into Equation A.4, we update $\boldsymbol{\theta}$ by

$$\boldsymbol{\theta}_{new} = \boldsymbol{\theta} + \alpha \left(y - f_\theta(\tilde{\boldsymbol{x}})\right) \tilde{\boldsymbol{x}}. \tag{A.8}$$

To make updates to $\boldsymbol{\theta}$ for $M > 1$, we can return the summation term from Equation A.3,

resulting in

$$\boldsymbol{\theta}_{new} = \boldsymbol{\theta} + \alpha \sum_{m=1}^{M} \left(y^{(k)} - f_\theta(\tilde{\boldsymbol{x}}^k) \right) \tilde{\boldsymbol{x}}^{(k)}. \tag{A.9}$$

Equation A.9 is called *batch gradient descent* [165]. This will update $\boldsymbol{\theta}$ only after going through the entire training set, or once per *epoch*. Another method, known as *stochastic gradient descent*, will update $\boldsymbol{\theta}$ for each training example. In Algorithm 1, we see that $\boldsymbol{\theta}$

Algorithm 1 Stochastic Gradient Descent

Given initial $\boldsymbol{\theta}$
for $m \leftarrow 1$ to M **do**
  $\boldsymbol{\theta}_{new} = \boldsymbol{\theta} + \alpha(y^{(m)} - f_\theta(\tilde{\boldsymbol{x}}^m))\tilde{\boldsymbol{x}}^{(m)}$
  Update $\boldsymbol{\theta} = \boldsymbol{\theta}_{new}$
end for

is updated and begins to optimize before reaching the sample M. With stochastic gradient descent, the parameters in $\boldsymbol{\theta}$ progress to a better result before the algorithm completes iterating through the dataset, a large advantage when M is large [165].

A hybrid technique combining batch gradient descent and stochastic gradient descent is called *mini-batch gradient descent* [166]. Instead of updating $\boldsymbol{\theta}$ once per epoch with batch gradient descent, or at every sample with stochastic gradient descent, we update $\boldsymbol{\theta}$ based on a mini-batch block of samples B. The update on $\boldsymbol{\theta}$ is shown in Algorithm 2. If $B = 1$,

Algorithm 2 Mini-Batch Gradient Descent

Given initial $\boldsymbol{\theta}$,
for $m \leftarrow 1$ to M **do**
  $\boldsymbol{\theta}_{new} = \boldsymbol{\theta} + \alpha \frac{1}{B} \sum_{b=1}^{B} (y^{(m)} - f_\theta(\tilde{\boldsymbol{x}}^m))\tilde{\boldsymbol{x}}^{(m)}$
  Update $\boldsymbol{\theta} = \boldsymbol{\theta}_{new}$
end for

mini-batch gradient descent is equivalent to stochastic gradient descent, and when $B = M$, we have batch gradient descent. For $1 < B < M$, there is typically an optimum value where increasing values of B improve training time due to parallelism or efficient matrix multiplication, however a large value for B also results in a decreasing number of updates to $\boldsymbol{\theta}$, slowing convergence [166].

A.2 Linear Classification

In Section A.1, the model f_θ was a linear function of the parameters in $\boldsymbol{\theta}$, however in a classification task, we predict discrete labels or posterior probabilities [54]. To accomplish this, we use a nonlinear function g resulting in a change to f_θ, namely

$$f_\theta(\tilde{\boldsymbol{x}}) = g\left(\tilde{\boldsymbol{x}}^T\boldsymbol{\theta}\right). \tag{A.10}$$

The nonlinear function g is known as an *activation function* [54]. If we consider a two-class classification problem, we can denote one class with a value of 0, and the other class with a value of 1. Thus, with $z = \tilde{\boldsymbol{x}}^T\boldsymbol{\theta}$, we define g as a step function,

$$g(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & z < 0. \end{cases} \tag{A.11}$$

Using g defined in Equation A.11, if we substitute Equation A.10 into Equation A.8, we have the *perceptron learning algorithm* [165],

$$\boldsymbol{\theta}_{new} = \boldsymbol{\theta} + \alpha\left(y - g\left(\tilde{\boldsymbol{x}}^T\boldsymbol{\theta}\right)\right)\tilde{\boldsymbol{x}}. \tag{A.12}$$

Updates to the weight parameters in $\boldsymbol{\theta}$ can be made using the same gradient descent methods as discussed in Section A.1. The drawbacks of using Equation A.10 are that the output of the perceptron $g(z)$ does not provide probabilistic outputs, and that the error is a piecewise constant function of $\boldsymbol{\theta}$. The latter results in discontinuities at the boundary based on slight changes of $\boldsymbol{\theta}$ [54]. The perceptron learning algorithm can be used to categorize an input $\tilde{\boldsymbol{x}}$ as belonging to one of two classes based on a boundary defined by $\tilde{\boldsymbol{x}}^T\boldsymbol{\theta} = 0$. For cases where there are more than two classes, additional decision boundaries need to be defined.

In Equation A.11, we denoted the classes using values 0 and 1. This is known as integer encoding. Another method to encode classes is by using a one-hot vector. The one-hot method consists of vectors the same size as the number of classes with a 1 in the position representing the class, and the other positions are filled with 0 [167]. For example, a task may be to categorize images into one of three classes: 'Dog', 'Cat', and 'Bird'. These strings must be converted to numbers for use with machine learning models. Using integer encoding, we can assign 'Dog', 'Cat', and 'Bird' to 1, 2, and 3, respectively. Or, using

one-hot vectors, we denote 'Dog' as $(0,0,1)^T$, 'Cat' as $(0,1,0)^T$, and 'Bird' as $(1,0,0)^T$. One-hot encoding allows for a simple implementation in digital circuits, however it likely requires more flip-flops to represent binary data (in our example integer encoding would require two flip-flops to encode the classes in binary, and three flip flops for the one-hot classes) [167].

A.3 Activation Functions

As we saw in Equation A.11, activation functions introduce nonlinearity into the output of f_θ, leading to more complex properties compared to the linear regression models [54]. In this section we will introduce commonly used activation functions.

A.3.1 Sigmoid

The sigmoid activation function is a continuous function centered on zero on the independent axis and ranging from $(0,1)$ on the dependent axis [54]. Defined as

$$\text{sigmoid} \coloneqq \sigma(z) = \frac{1}{1+e^{-z}}, \tag{A.13}$$

the sigmoid function is shown in Figure A.1.

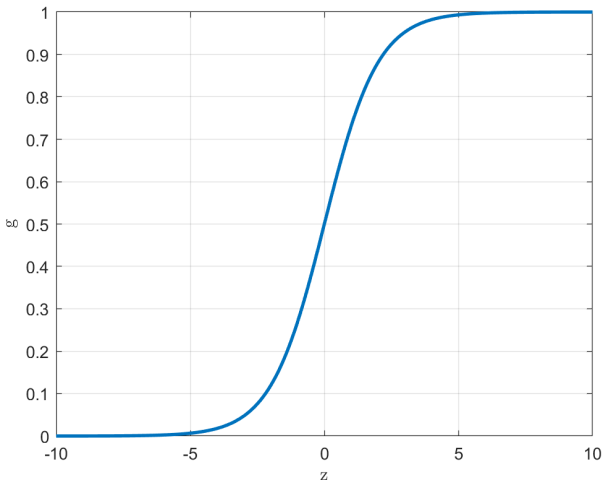


Figure A.1. Sigmoid activation function.

The sigmoid function can readily be used for a two-class ($N = 2$) classification problem, where a value of 0 is used for one class, and a value of 1 is used for the other. The classes can be distinguished using a boundary of 0.5, such as

$$n = \begin{cases} 1, & \text{if } \sigma(z) \geq 0.5 \\ 0, & \sigma(z) < 0.5. \end{cases} \tag{A.14}$$

A.3.2 Softmax

For a classification task where $N > 2$, a normalized exponential, also known as the softmax activation function can be used [54]. Defined as

$$\text{softmax} \coloneqq \text{softmax}(\boldsymbol{z})_j = \frac{e^{z_j}}{\sum_{n=1}^{N} e^{z_n}}, \tag{A.15}$$

we see that the softmax activation function operates on a vector $\boldsymbol{z}$ with elements z_j, as opposed to a scalar z with the sigmoid function.

A.3.3 Hyperbolic Tangent

Like the sigmoid function, the hyperbolic tangent function bounds the dependent variable across all values on the independent axis. However, instead of ranging from $(0, 1)$ as with the sigmoid function, the hyperbolic tangent function bounds are $(-1, 1)$. Shown in Figure A.2, the hyperbolic tangent function is defined as

$$\text{hyperbolic tangent} \coloneqq \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}. \tag{A.16}$$

When dealing with data that is scaled in such a way that the average value is near zero, the hyperbolic tangent activation is preferred over the sigmoid in order to reach convergence faster [168].

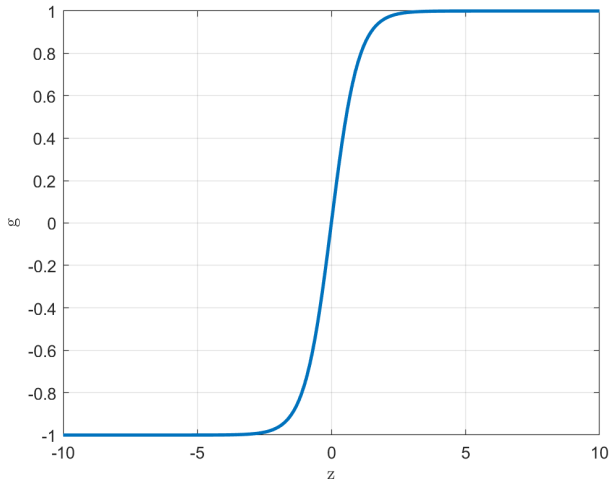


Figure A.2. Hyperbolic tangent activation function.

A.3.4 Rectified Linear Unit

We have seen that both the sigmoid and hyperbolic tangent activation functions limit the output of g to a finite range. When the output of g is then used for follow-on processing, the saturation caused by the finite range makes the task of determining weight parameters challenging due to gradients equal to nearly zero [127].

Rectified linear units are defined by the activation function

$$g(z) = \begin{cases} z, & \text{if } z \geq 0 \\ 0, & z < 0, \end{cases} \tag{A.17}$$

or equivalently $g(z) = \max\{0, z\}$. Figure A.3 depicts the rectified linear unit activation function. Unfortunately the gradient descent optimization methods can't be applied to this activation function when $g(z) = 0$ [67].

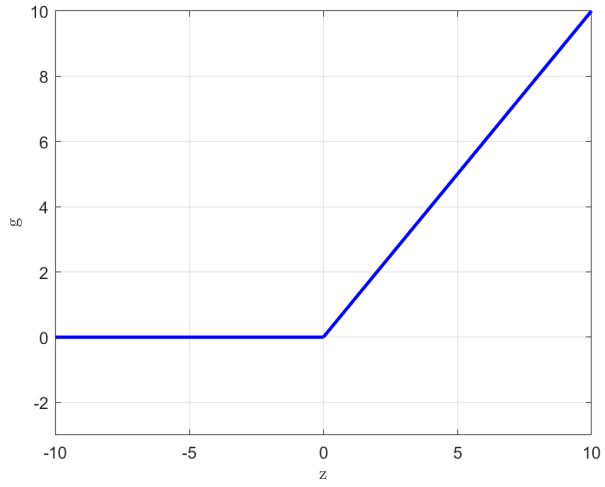


Figure A.3. Rectified linear unit activation function.

An extension to the rectified linear unit is the leaky rectified linear unit [127]. For values of $z < 0$, the leaky rectified linear unit allows for non-zero values of $g(z)$, resulting in a small, but non-zero gradient. The leaky rectified linear unit is shown in Figure A.4 where the slope of $g(z)$ for $z < 0$ is denoted *alpha*.

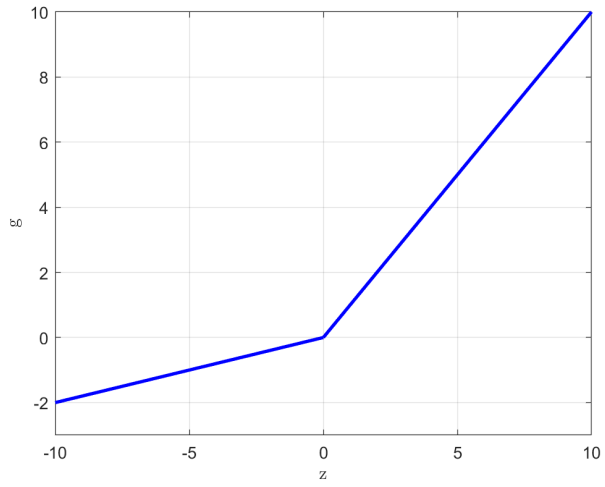


Figure A.4. Leaky rectified linear unit activation function ($alpha = 0.2$).

A.4 Neural Networks

Neural networks are nonlinear statistical models [164]. Capable of performing regression as well as classification, a neural network uses the same processes as previously described in Sections A.1 and A.2. Instead of an output based on a single input vector and an activation function $f_\theta(\tilde{\boldsymbol{x}}^T\boldsymbol{\theta}) = g(\tilde{\boldsymbol{x}})$ an output can be produced by the sum of nodes, called neurons, with their respective input vector, weight parameters, and activation functions. Going further, this output can be used as a portion of an input to another neuron. The neural network uses a linear combination of inputs to model the target as a nonlinear function of the extracted features [164].

A.4.1 Neural Network Training

As an example of how a neural network is trained, we will consider a two-class classification task using a model of a feed-forward densely-connected neural network as shown in Figure A.5. This configuration is also known as a multiple-layer perceptron network, where each neuron is an individual perceptron and the links between the neurons are the weight parameters. From left to right, a vector of size 3 with values x_1, x_2, and x_3, are each connected to all four neurons in the hidden layer h and every hidden layer neuron is then connected to the output neuron $\hat{y}$. The shaded-in neurons denoted 0 are the additional input and hidden variables equal to one and correspond to the biases in each layer. Since this is a classification task, we use the sigmoid activation function $\sigma(z)$ at the output. To simplify this demonstration, the hidden layer neuron activation functions will also use sigmoid and we will only have one sample in the dataset.

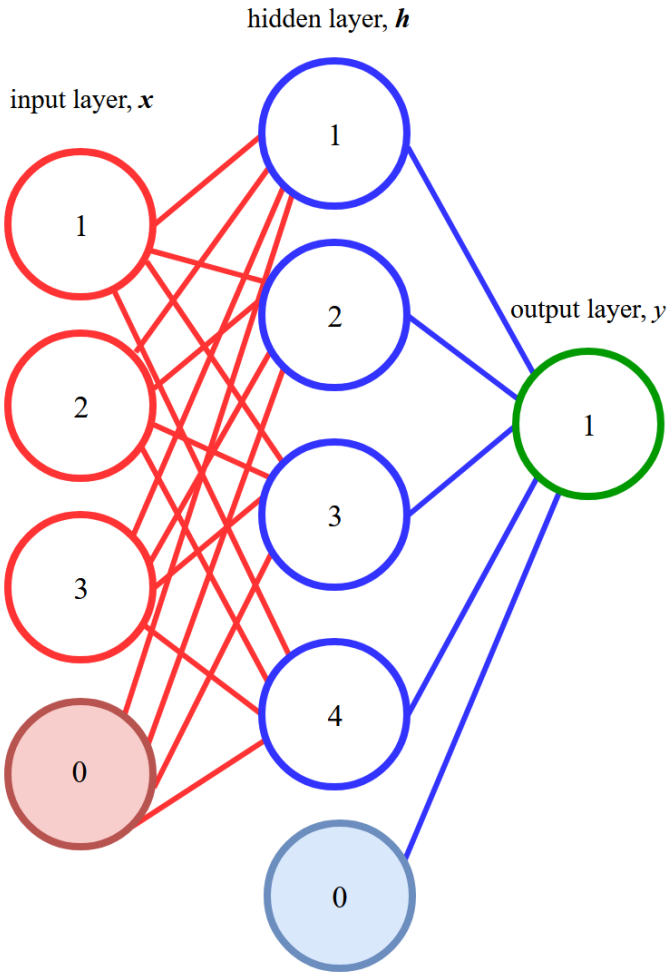


Figure A.5. Densely-connected neural network with one hidden layer.

During the forward pass, a sample $\boldsymbol{x}$ is processed through the neural network to the neurons of the hidden layer and the output layer neurons. We also add another element $x_0 = 1$ to our input layer and $h_0 = 1$, to our hidden layer. The weights between the input layer and hidden layer are identified as $\theta_{i,j}^{x,h}$, where i indicates the neuron corresponding to x and j is the neuron in the hidden layer. Likewise, $\theta_{i,j}^{h,\hat{y}}$ results in the weight parameter from a hidden layer neuron to the output layer neuron. The bias between the input and hidden layer is given

by $\theta_{0,j}^{x,h}$, while $\theta_{0,j}^{h,\hat{y}}$ is the bias between the hidden and output layer. Therefore,

$$h_1 = \sigma\left(\theta_{1,1}^{x,h}x_1 + \theta_{2,1}^{x,h}x_2 + \theta_{3,1}^{x,h}x_3 + \theta_{0,1}^{x,h}\right) \tag{A.18}$$

$$h_2 = \sigma\left(\theta_{1,2}^{x,h}x_1 + \theta_{2,2}^{x,h}x_2 + \theta_{3,2}^{x,h}x_3 + \theta_{0,2}^{x,h}\right) \tag{A.19}$$

$$h_3 = \sigma\left(\theta_{1,3}^{x,h}x_1 + \theta_{2,3}^{x,h}x_2 + \theta_{3,3}^{x,h}x_3 + \theta_{0,3}^{x,h}\right) \tag{A.20}$$

$$h_4 = \sigma\left(\theta_{1,4}^{x,h}x_1 + \theta_{2,4}^{x,h}x_2 + \theta_{3,4}^{x,h}x_3 + \theta_{0,4}^{x,h}\right) \tag{A.21}$$

$$\hat{y} = \sigma\left(\theta_{1,1}^{h,\hat{y}}h_1 + \theta_{2,1}^{h,\hat{y}}h_2 + \theta_{3,1}^{h,\hat{y}}h_3 + \theta_{4,1}^{h,\hat{y}}h_4 + \theta_{0,1}^{h,\hat{y}}\right). \tag{A.22}$$

Using the squared error loss function in Equation A.3, we begin the process of *backpropagation* by calculating the difference between the output predicted value $\hat{y}$ and the target value y.

Illustrated in Figure A.6, backpropagation is the process of updating the parameters in $\boldsymbol{\theta}$, starting from the weights connecting the hidden layer to the output $\boldsymbol{\theta}^{h,\hat{y}}$, and systematically moving back to the weights connecting the input neurons with the hidden layer neurons $\boldsymbol{\theta}^{x,h}$ [169]. To demonstrate backpropagation using gradient descent, we use the chain rule as shown in [170]. Using Equations A.4 and A.8 to update $\theta_{1,1}^{h,\hat{y}}$, we have

$$J(\boldsymbol{\theta}) = \frac{1}{2}\left(\hat{y} - y\right)^2 \tag{A.23}$$

$$z = \theta_{1,1}^{h,\hat{y}}h_1 + \theta_{2,1}^{h,\hat{y}}h_2 + \theta_{3,1}^{h,\hat{y}}h_3 + \theta_{4,1}^{h,\hat{y}}h_4 + \theta_{0,1}^{h,\hat{y}} \tag{A.24}$$

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{A.25}$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_{1,1}^{h,\hat{y}}} = \frac{\partial J(\boldsymbol{\theta})}{\partial \sigma(z)}\frac{\partial \sigma(z)}{\partial z}\frac{\partial z}{\theta_{1,1}^{h,\hat{y}}}. \tag{A.26}$$

Examining each term on the right in Equation A.26,

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \sigma(z)} = (\sigma(z) - y) \tag{A.27}$$

$$\frac{\partial \sigma(z)}{z} = \frac{e^{-z}}{(1 + e^{-z})^2} \tag{A.28}$$

$$= \sigma(z)\,(1 - \sigma(z)) \tag{A.29}$$

$$\frac{\partial \sigma(z)}{\theta_{1,1}^{h,\hat{y}}} = h_1, \tag{A.30}$$

and substituting these terms into Equation A.26, we have

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_{1,1}^{h,\hat{y}}} = (\sigma(z) - y)\,\sigma(z)\,(1 - \sigma(z))\,h_1 \tag{A.31}$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_{1,1}^{h,\hat{y}}} = (\hat{y} - y)\,\hat{y}\,(1 - \hat{y})\,h_1. \tag{A.32}$$

Updating $\theta_{1,1}^{h,\hat{y}}$ by gradient descent, we adjust our weight parameter, finally resulting in

$$\left(\theta_{1,1}^{h,\hat{y}}\right)_{\text{new}} = \theta_{1,1}^{h,\hat{y}} - \alpha \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_{1,1}^{h,\hat{y}}} \tag{A.33}$$

$$= \theta_{1,1}^{h,\hat{y}} + \alpha\,(y - \hat{y})\,\hat{y}\,(1 - \hat{y})\,h_1. \tag{A.34}$$

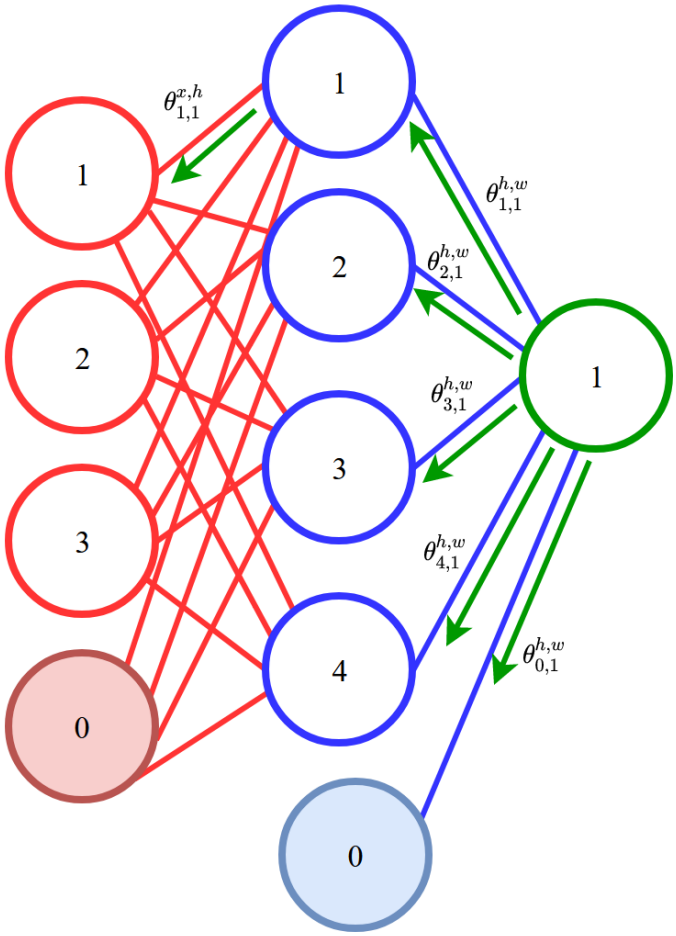


Figure A.6. Backpropagation to update weight parameters.

This process continues for the remaining neurons in the hidden layer, updating $\theta^{h,\hat{y}}_{2,1}$, $\theta^{h,\hat{y}}_{3,1}$, $\theta^{h,\hat{y}}_{4,1}$, and $\theta^{h,\hat{y}}_{0,1}$. For the weight parameters connecting the input and hidden layer neurons $\boldsymbol{\theta}^{x,h}$ the procedure is similar but also includes the forward pass output values of the hidden layer neurons from Equations A.18-A.21. To find the weight parameter $\theta^{x,h}_{1,1}$, we again use the chain rule as shown in [170] to find the change in the loss function with respect to the change in $\theta^{x,h}_{1,1}$, giving us

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta^{x,h}_{1,1}} = \left(\frac{\partial J(\boldsymbol{\theta})}{\partial \sigma(z)} \frac{\partial \sigma(z)}{\partial z} \frac{\partial z}{\partial \sigma(w)}\right) \frac{\partial \sigma(w)}{\partial w} \frac{\partial w}{\partial \theta^{x,h}_{1,1}} \tag{A.35}$$

where

$$z = \theta_{1,1}^{h,\hat{y}} h_1 + \theta_{2,1}^{h,\hat{y}} h_2 + \theta_{3,1}^{h,\hat{y}} h_3 + \theta_{4,1}^{h,\hat{y}} h_4 \tag{A.36}$$

$$w = \theta_{1,1}^{x,h} x_1 + \theta_{2,1}^{x,h} x_2 + \theta_{3,1}^{x,h} x_3. \tag{A.37}$$

Collecting the terms on the right of Equation A.44, we get

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \sigma(z)} = (\sigma(z) - y) \tag{A.38}$$

$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)\,(1 - \sigma(z)) \tag{A.39}$$

$$\frac{\partial z}{\partial \sigma(w)} = \theta_{1,1}^{h,\hat{y}} \tag{A.40}$$

$$\frac{\partial \sigma(w)}{\partial w} = \sigma(w)\,(1 - \sigma(w)) \tag{A.41}$$

$$\frac{\partial w}{\partial \theta_{1,1}^{x,h}} = x_1, \tag{A.42}$$

resulting in

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_{1,1}^{x,h}} = (\hat{y} - y)\,\hat{y}\,(1 - \hat{y})\,\theta_{1,1}^{h,\hat{y}} h_1(1 - h_1)x_1. \tag{A.43}$$

Applying gradient descent to update $\theta_{1,1}^{x,h}$, we have

$$\left(\theta_{1,1}^{x,h}\right)_{\text{new}} = \theta_{1,1}^{x,h} + \alpha\,(y - \hat{y})\,\hat{y}\,(1 - \hat{y})\,\theta_{1,1}^{h,\hat{y}} h_1(1 - h_1)x_1. \tag{A.44}$$

This process for updating the hidden layer weight parameters is then repeated for each connection between the input and hidden layer neurons.

The weight parameter updates are adjusted in part by the learning rate α as seen in Equations A.34 and A.44. Usually a constant, α can also be adjusted at each update to minimize error [164] in optimization schemes such as adaptive momentum estimation [134].

A.4.2 Neural Network Architecture

The type of mathematical operation carried out by the neuron is informed by the nature of the data. Data with parametric features is well suited to the linear combination of functions used in densely-connected neural networks [53]. If the data is spatially correlated such as in an image, the convolutional operations in a convolutional neural network are more appropriate in order to extract input features [53]. Recursive neural networks are preferred when capturing dynamic temporally-based features are needed [70].

Aside from the learning rate, there are other hyperparameter choices to make regarding the architecture of the neural network. Depending on features of the input samples and the task to accomplish, the selection of the number of hidden layers and neurons can determine how well the network performs against new samples after training. Background knowledge of the sample features and the goal of the task as well as experimentation inform the selection of the number of neurons and layers required [164]. With too many weight parameters, neural networks are likely to overfit the training data, however this can be mitigated with stopping training before fully converging and reaching the minimum loss value [164]. Each hidden layer uses input features for regression or classification. The implementation of multiple hidden layers allows for the network to process these features at multiple levels of abstraction [70].

As the number of hidden neurons and layers are increased, the number of weight parameters also increases. Saving these parameters for use at a later time requires storage, so depending on the application, a neural network may be constrained to a certain architecture based on memory available [70]. However, if a neural network doesn't have enough neurons, the network may be difficult to train resulting in inaccurate results and taking a long time to train [171].

A.5 Summary

Neural networks originated in an attempt to represent the biological neuron behavior with mathematical functions [63]. There is no suggestion that artificial neural networks should represent biological information processing [54], however neural networks were loosely inspired by neuroscience [67]. Neural networks can be used for classification as well as regression tasks and can model linear and nonlinear functions.

The architecture of a neural network is partly an art informed by experimentation [164]. Knowledge of the features being processed and the effect of the mathematical functions can be used to design a successful neural network.

In this Appendix, we have provided the essential elements to understand how feedforward neural networks work based on linear algebra. We have also introduced activation functions commonly used employed in neural networks. Finally, we discussed the overall architecture of the neural network and noted considerations for developing a neural network to accomplish a task.

List of References

[1] Information Security Definitions, 44 U.S. Code § 3552. Dec. 2014. [Online]. Available: https://www.law.cornell.edu/uscode/text/44/3552

[2] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," in *Proceedings of the 4th Annual Workshop on Selected Areas of Cryptography*, 2001, pp. 1–24.

[3] D. Menasce, "QoS issues in Web services," *IEEE Internet Computing*, vol. 6, no. 6, pp. 72–75, Nov. 2002.

[4] A. Dua, N. Kumar, A. K. Das, and W. Susilo, "Secure message communication protocol among vehicles in smart city," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4359–4373, May 2018.

[5] D. Fang, Y. Qian, and R. Q. Hu, "Security for 5G mobile wireless networks," *IEEE Access*, vol. 6, pp. 4850–4874, 2018.

[6] L. Ferdinando, "'Terabyte of death' cyberattack against DOD looms, DISA director warns." [Online]. Available: https://www.defense.gov/Explore/News/Article/Article/1414146/terabyte-of-death-cyberattack-against-dod-looms-disa-director-warns/

[7] F. Konkel, "Pentagon thwarts 36 million email breach attempts daily," 2018. [Online]. Available: https://www.nextgov.com/cybersecurity/2018/01/pentagon-thwarts-36-million-email-breach-attempts-daily/145149/

[8] S. Gallagher, "Hacker's paradise: Louisiana's ransomware disaster far from over | Ars Technica." [Online]. Available: https://arstechnica.com/information-technology/2019/11/hackers-paradise-louisianas-ransomware-disaster-far-from-over/

[9] CBS News, "Hackers demand $14 million from nursing homes in ransomware attack," 2019. [Online]. Available: https://www.cbsnews.com/news/hackers-ransomware-nursing-homes-14-million/

[10] T. Ma, F. Hu, and M. Ma, "Fast and efficient physical layer authentication for 5G HetNet handover," in *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, Nov. 2017, pp. 1–3.

[11] Z. Liu, J. Ma, Q. Huang, and S. Moon, "Asymmetric key pre-distribution scheme for sensor networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 3, pp. 1366–1372, Mar. 2009.

[12] X. Wang, P. Hao, and L. Hanzo, "Physical-layer authentication for wireless security enhancement: current challenges and future developments," *IEEE Communications Magazine*, vol. 54, no. 6, pp. 152–158, June 2016.

[13] A. Mukherjee, S. A. A. Fakoorian, J. Huang, and A. L. Swindlehurst, "Principles of physical layer security in multiuser wireless networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1550–1573, 2014.

[14] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, Nov. 2012.

[15] P. L. Yu, J. S. Baras, and B. M. Sadler, "Physical-layer authentication," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 38–51, 2008. [Online]. Available: http://ieeexplore.ieee.org/document/4451099/

[16] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow: Concepts, tools, and techniques to build intelligent systems*, first edition ed. Sebastopol, CA, USA: O'Reilly Media, 2017. [Online]. Available: http://shop.oreilly.com/product/0636920052289.do

[17] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proceedings of the 14th ACM international conference on Mobile computing and networking - MobiCom '08*. San Francisco, CA, USA: ACM Press, 2008, p. 116.

[18] National Institute of Standards and Technology, "Minimum security requirements for federal information and information systems," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST FIPS 200, Mar. 2006. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.200.pdf

[19] *Cybersecurity*, DOD Instruction 8500.01, Department of Defense, Washington, DC, USA, Mar. 2014. [Online]. Available: https://www.esd.whs.mil/portals/54/documents/dd/issuances/dodi/850001_2014.pdf

[20] *Doctrine for the Armed Forces of the United States*, JP-6-0, Joint Chiefs of Staff, Washington, DC, USA, June 2015. [Online]. Available: https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/jp6_0.pdf

[21] R. Fantacci, L. Maccari, T. Pecorella, and F. Frosali, "Analysis of secure handover for IEEE 802.1x-based wireless ad hoc networks," *IEEE Wireless Communications*, vol. 14, no. 5, pp. 21–29, Oct. 2007.

[22] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978. [Online]. Available: http://doi.acm.org/10.1145/359340.359342

[23] Y. Deng, G. Wang, J. Cao, and X. Xiao, "Practical secure and fast handoff framework for pervasive Wi-Fi access," *IET Information Security*, vol. 7, no. 1, pp. 22–29, Mar. 2013.

[24] M. R. Asaar, M. Salmasizadeh, W. Susilo, and A. Majidi, "A secure and efficient authentication technique for vehicular ad-hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 5409–5423, June 2018.

[25] K. Zeng, K. Govindan, and P. Mohapatra, "Non-cryptographic authentication and identification in wireless networks [Security and Privacy in Emerging Wireless Networks]," *IEEE Wireless Communications*, vol. 17, no. 5, pp. 56–62, Oct. 2010.

[26] O. Gungor and C. E. Koksal, "On the basic limits of RF-fingerprint-based authentication," *IEEE Transactions on Information Theory*, vol. 62, no. 8, pp. 4523–4543, Aug. 2016.

[27] A. C. Polak, S. Dolatshahi, and D. L. Goeckel, "Identifying wireless users via transmitter imperfections," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 7, pp. 1469–1479, Aug. 2011.

[28] J. K. Tugnait, "Wireless user authentication via comparison of power spectral densities," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 9, pp. 1791–1802, Sep. 2013.

[29] Yingbin Liang, H. Poor, and S. Shamai, "Secure communication over fading channels," *IEEE Transactions on Information Theory*, vol. 54, no. 6, pp. 2470–2492, June 2008.

[30] N. Al Khanbashi, N. Al Sindi, S. Al-Araji, N. Ali, Z. Chaloupka, V. Yenamandra, and J. Aweya, "Real time evaluation of RF fingerprints in wireless LAN localization systems," in *2013 10th Workshop on Positioning, Navigation and Communication (WPNC)*, Mar. 2013, pp. 1–6.

[31] O. Ureten and N. Serinken, "Wireless security through RF fingerprinting," *Canadian Journal of Electrical and Computer Engineering*, vol. 32, no. 1, pp. 27–33, 2007.

[32] M. Valkama, M. Renfors, and V. Koivunen, "Advanced methods for I/Q imbalance compensation in communication receivers," *IEEE Transactions on Signal Processing*, vol. 49, no. 10, pp. 2335–2344, Oct. 2001.

[33] A. C. Polak and D. L. Goeckel, "RF fingerprinting of users who actively mask their identities with artificial distortion," in *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Nov. 2011, pp. 270–274.

[34] K. Yu and B. Ottersten, "Models for MIMO propagation channels: A review," *Wireless Communications and Mobile Computing*, vol. 2, no. 7, pp. 653–666, 2002. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/wcm.78

[35] B. Yang, K. Letaief, R. Cheng, and Z. Cao, "Channel estimation for OFDM transmission in multipath fading channels based on parametric channel modeling," *IEEE Transactions on Communications*, vol. 49, no. 3, pp. 467–479, Mar. 2001.

[36] S. Verdu, "Spectral efficiency in the wideband regime," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1319–1343, June 2002.

[37] W. C. Jakes, *Microwave Mobile Communications.* New York, NY, USA: Wiley, 1974.

[38] K. St. Germain and F. Kragh, "Multi-subcarrier physical layer authentication using channel state information and deep learning," Kauai, HI, USA, Jan. 2021. [Online]. Available: http://scholarspace.manoa.hawaii.edu/handle/10125/71467

[39] F. J. Liu, Xianbin Wang, and H. Tang, "Robust physical layer authentication using inherent properties of channel impulse response," in *2011 - MILCOM 2011 Military Communications Conference*, Nov. 2011, pp. 538–542.

[40] E. Biglieri and G. Taricco, "Transmission and reception with multiple antennas: Theoretical foundations," *Foundations and Trends® in Communications and Information Theory*, vol. 1, no. 2, pp. 183–332, Aug. 2004, publisher: Now Publishers, Inc. [Online]. Available: https://www.nowpublishers.com/article/Details/CIT-002

[41] K. Pedersen, J. Andersen, J. Kermoal, and P. Mogensen, "A stochastic multiple-input-multiple-output radio channel model for evaluation of space-time coding algorithms," in *Vehicular Technology Conference Fall 2000. IEEE VTS Fall VTC2000. 52nd Vehicular Technology Conference (Cat. No.00CH37152)*, Sep. 2000, vol. 2, pp. 893–897 vol.2.

[42] Z. Liu, L. Zhang, and Z. Ding, "Exploiting bi-directional channel reciprocity in deep learning for low rate massive MIMO CSI feedback," *IEEE Wireless Communications Letters*, vol. 8, no. 3, pp. 889–892, June 2019.

[43] Y. Shi, K. Davaslioglu, and Y. E. Sagduyu, "Generative adversarial network for wireless signal spoofing," in *Proceedings of the ACM Workshop on Wireless Security and Machine Learning - WiseML 2019*. Miami, FL, USA: ACM Press, 2019, pp. 55–60. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3324921.3329695

[44] L. Xiao, L. Greenstein, N. Mandayam, and W. Trappe, "A physical-layer technique to enhance authentication for mobile terminals," in *2008 IEEE International Conference on Communications*, May 2008, pp. 1520–1524.

[45] Y. Liu, S. C. Draper, and A. M. Sayeed, "Exploiting channel diversity in secret key generation from multipath fading randomness," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 5, pp. 1484–1497, Oct. 2012.

[46] H. Taha and E. Alsusa, "Physical layer secret key exchange using phase randomization in MIMO-OFDM," in *2015 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2015, pp. 1–6.

[47] B. T. Quist and M. A. Jensen, "Maximizing the secret key rate for informed radios under different channel conditions," *IEEE Transactions on Wireless Communications*, vol. 12, no. 10, pp. 5146–5153, Oct. 2013.

[48] R. Wilson, D. Tse, and R. A. Scholtz, "Channel identification: Secret sharing using reciprocity in ultrawideband channels," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 3, pp. 364–375, Sep. 2007.

[49] D. B. Faria and D. R. Cheriton, "Detecting identity-based attacks in wireless networks using signalprints," in *Proceedings of the 5th ACM workshop on Wireless security - WiSe '06*. Los Angeles, California: ACM Press, 2006, p. 43. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1161289.1161298

[50] Y. Chen, W. Trappe, and R. P. Martin, "Detecting and localizing wireless spoofing attacks," in *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, June 2007, pp. 193–202.

[51] L. Xiao, L. Greenstein, N. Mandayam, and W. Trappe, "Fingerprints in the ether: Using the physical layer for wireless authentication," in *2007 IEEE International Conference on Communications*, June 2007, pp. 4646–4651.

[52] L. Xiao, L. J. Greenstein, N. B. Mandayam, and W. Trappe, "Channel-based spoofing detection in frequency-selective Rayleigh channels," *IEEE Transactions on Wireless Communications*, vol. 8, no. 12, pp. 5948–5956, Dec. 2009.

[53] D. Roy, T. Mukherjee, and M. Chatterjee, "Machine learning in adversarial RF environments," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 82–87, May 2019.

[54] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[55] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning* (Adaptive Computation and Machine Learning). The MIT Press, Sep. 2006.

[56] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, July 2001. [Online]. Available: https://www.mitpressjournals.org/doi/abs/10.1162/089976601750264965

[57] L. Devroye, L. Gyorfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition* (Applications of Mathematics). New York, NY, USA: Springer-Verlag, 1996, no. 31.

[58] H. Liu, Y. Wang, J. Liu, J. Yang, Y. Chen, and H. V. Poor, "Authenticating users through fine-grained channel information," *IEEE Transactions on Mobile Computing*, vol. 17, no. 2, pp. 251–264, Feb. 2018.

[59] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Record*, vol. 29, no. 2, pp. 93–104, May 2000. [Online]. Available: http://doi.org/10.1145/335191.335388

[60] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, Dec. 2008, pp. 413–422.

[61] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 1, pp. 3:1–3:39, Mar. 2012. [Online]. Available: http://doi.org/10.1145/2133360.2133363

[62] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html

[63] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943. [Online]. Available: https://doi.org/10.1007/BF02478259

[64] F. Rosenblatt, "The Perceptron. A Perceiving and Recognizing Automaton." Tech. Rep. 85-460-1, Jan. 1957. [Online]. Available: https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf

[65] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

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

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

[68] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[69] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, Dec. 2017.

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

[71] K. St. Germain and F. Kragh, "Mobile physical-layer authentication using channel state information and conditional recurrent neural networks," in *IEEE 93rd Vehicular Technology Conference: VTC2021-Spring*. Helsinki, Finland: to be published, Apr. 2021.

[72] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 240–254, Mar. 1994. [Online]. Available: http://doi.org/10.1109/72.279188

[73] T. J. O'Shea, T. C. Clancy, and R. W. McGwier, "Recurrent neural radio anomaly detection," *arXiv:1611.00301 [cs]*, Nov. 2016, arXiv: 1611.00301. [Online]. Available: http://arxiv.org/abs/1611.00301

[74] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: https://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735

[75] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://www.aclweb.org/anthology/D14-1179

[76] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37* (ICML'15). Lille, France: JMLR.org, July 2015, pp. 2342–2350.

[77] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *NIPS 2014 Workshop on Deep Learning*, Dec. 2014.

[78] W. Liu, L.-L. Yang, and L. Hanzo, "Recurrent neural network based narrowband channel prediction," in *2006 IEEE 63rd Vehicular Technology Conference*, May 2006, vol. 5, pp. 2173–2177.

[79] T. Ding and A. Hirose, "Fading channel prediction based on combination of complex-valued neural networks and chirp Z-transform," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 9, pp. 1686–1695, Sep. 2014.

[80] W. Jiang and H. D. Schotten, "Recurrent neural networks with long short-term memory for fading channel prediction," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, May 2020, pp. 1–5.

[81] Q. Wang, H. Li, D. Zhao, Z. Chen, S. Ye, and J. Cai, "Deep neural networks for CSI-based authentication," *IEEE Access*, vol. 7, pp. 123 026–123 034, 2019.

[82] D. Roy, T. Mukherjee, M. Chatterjee, and E. Pasiliao, "RF transmitter fingerprinting exploiting spatio-temporal properties in raw signal data," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. Boca Raton, FL, USA: IEEE, Dec. 2019, pp. 89–96.

[83] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2* (NIPS'14). Cambridge, MA, USA: MIT Press, 2014, pp. 2672–2680, event-place: Montreal, Canada.

[84] I. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *arXiv:1701.00160 [cs]*, Dec. 2016, arXiv: 1701.00160. [Online]. Available: http://arxiv.org/abs/1701.00160

[85] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *arXiv:1406.2661 [cs, stat]*, June 2014, arXiv: 1406.2661. [Online]. Available: http://arxiv.org/abs/1406.2661

[86] J. Langr and V. Bok, *GANs in Action: Deep learning with Generative Adversarial Networks*, Sep. 2019. [Online]. Available: https://livebook.manning.com/book/gans-in-action/about-this-book/

[87] K. St. Germain and F. Kragh, "Physical-layer authentication using channel state information and machine learning," in *2020 14th International Conference on Signal Processing and Communication Systems (ICSPCS)*, Adelaide, AUS, Dec. 2020.

[88] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 105–114.

[89] Y. Li, S. Liu, J. Yang, and M.-H. Yang, "Generative face completion," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 5892–5900.

[90] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan, "Perceptual generative adversarial networks for small object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 1951–1959.

[91] A. Odena, "Semi-supervised learning with generative adversarial networks," *arXiv:1606.01583 [cs, stat]*, Oct. 2016, arXiv: 1606.01583. [Online]. Available: http://arxiv.org/abs/1606.01583

[92] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *Engineering Applications of Neural Networks* (Communications in Computer and Information Science), C. Jayne and L. Iliadis, Eds. Cham: Springer International Publishing, 2016, pp. 213–226.

[93] M. Li, G. Liu, S. Li, and Y. Wu, "Radio classify generative adversarial networks: A semi-supervised method for modulation recognition," in *2018 IEEE 18th International Conference on Communication Technology (ICCT)*, Oct. 2018, pp. 669–672.

[94] C. Xiao, D. Han, Y. Ma, and Z. Qin, "CsiGAN: robust channel state information-based activity recognition with GANs," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 191–10 204, Dec. 2019.

[95] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional GANs," *arXiv:1706.02633 [cs, stat]*, Dec. 2017, arXiv: 1706.02633. [Online]. Available: http://arxiv.org/abs/1706.02633

[96] A. Koochali, P. Schichtel, A. Dengel, and S. Ahmed, "Probabilistic forecasting of sensory data with generative adversarial networks – ForGAN," *IEEE Access*, vol. 7, pp. 63 868–63 880, 2019.

[97] Y. Dong, H. Wang, and Y.-D. Yao, "Channel estimation for one-bit multiuser massive mimo using conditional gan," *IEEE Communications Letters*, pp. 1–1, 2020.

[98] K. St. Germain and F. Kragh, "Multi-transmitter physical layer authentication using channel state information and deep learning," in *2020 14th International Conference on Signal Processing and Communication Systems (ICSPCS)*, Adelaide, AUS, Dec. 2020.

[99] K. St. Germain and F. Kragh, "Channel prediction and transmitter authentication with adversarially-trained recurrent neural networks," *IEEE Open Journal of the Communications Society*, pp. 964–974, 2021.

[100] R. Atkinson and N. Haller, "On Internet Authentication," Oct. 1994. [Online]. Available: https://tools.ietf.org/html/rfc1704

[101] T. Zia, A. Zomaya, and N. Ababneh, "Evaluation of overheads in security mechanisms in wireless sensor networks," in *2007 International Conference on Sensor Technologies and Applications (SENSORCOMM 2007)*, Oct. 2007, pp. 181–185.

[102] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichitiu, "Analyzing and modeling encryption overhead for sensor network nodes," in *Proceedings of the 2Nd ACM International Conference on Wireless Sensor Networks and Applications* (WSNA '03). New York, NY, USA: ACM, 2003, pp. 151–159, event-place: San Diego, CA, USA. [Online]. Available: http://doi.acm.org/10.1145/941350.941372

[103] Y. Kim, A. Perrig, and G. Tsudik, "Group key agreement efficient in communication," *IEEE Transactions on Computers*, vol. 53, no. 7, pp. 905–921, July 2004.

[104] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "IoT: Internet of threats? A survey of practical security vulnerabilities in real IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, Oct. 2019.

[105] M. D. Aime, G. Calandriello, and A. Lioy, "Dependability in wireless networks: Can we rely on WiFi?" *IEEE Security Privacy*, vol. 5, no. 1, pp. 23–29, Jan. 2007.

[106] X. Xing, E. Shakshuki, D. Benoit, and T. Sheltami, "Security analysis and authentication improvement for IEEE 802.11i specification," in *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, Nov. 2008, pp. 1–5.

[107] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, "A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 116:1–116:29, Jan. 2019. [Online]. Available: http://doi.acm.org/10.1145/3292674

[108] G. Apostolopoulos, V. Peris, P. Pradhan, and D. Saha, "Securing electronic commerce: reducing the SSL overhead," *IEEE Network*, vol. 14, no. 4, pp. 8–16, July 2000.

[109] Y. Zou, J. Zhu, X. Wang, and L. Hanzo, "A survey on wireless security: Technical challenges, recent advances, and future trends," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1727–1765, Sep. 2016.

[110] P. Baracca, N. Laurenti, and S. Tomasin, "Physical layer authentication over MIMO fading wiretap channels," *IEEE Transactions on Wireless Communications*, vol. 11, no. 7, pp. 2564–2573, July 2012.

[111] C. Nerguizian, C. Despins, and S. Affes, "Geolocation in mines with an impulse response fingerprinting technique and neural networks," in *IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004*, Sep. 2004, vol. 5, pp. 3589–3594 Vol. 5.

[112] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 763–776, Jan. 2017.

[113] J. Xiao, K. Wu, Y. Yi, and L. M. Ni, "FIFS: Fine-grained indoor fingerprinting system," in *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, July 2012, pp. 1–7.

[114] L. Xiao, Y. Li, G. Han, G. Liu, and W. Zhuang, "PHY-layer spoofing detection with reinforcement learning in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 10 037–10 047, Dec. 2016.

[115] F. Pan, Z. Pang, M. Luvisotto, X. Jiang, R. N. Jansson, M. Xiao, and H. Wen, "Authentication based on channel state information for industrial wireless communications," in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, Oct. 2018, pp. 4125–4130.

[116] R. Candell, K. A. Remley, J. T. Quimby, D. Novotny, A. Curtin, P. B. Papazian, M. Kashef, and J. Diener, "Industrial wireless systems radio propagation measurements," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST TN 1951, Jan. 2017. [Online]. Available: http://doi.org/10.18434/T44S3N

[117] F. Pan, H. Wen, R. Liao, Y. Jiang, A. Xu, K. Ouyang, and X. Zhu, "Physical layer authentication based on channel information and machine learning," in *2017 IEEE Conference on Communications and Network Security (CNS)*, Oct. 2017, pp. 364–365.

[118] R.-F. Liao, H. Wen, J. Wu, F. Pan, A. Xu, Y. Jiang, F. Xie, and M. Cao, "Deep-learning-based physical layer authentication for industrial wireless sensor networks," *Sensors (Basel, Switzerland)*, vol. 19, no. 11, May 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6603790/

[119] R.-F. Liao, H. Wen, J. Wu, F. Pan, A. Xu, H. Song, F. Xie, Y. Jiang, and M. Cao, "Security enhancement for mobile edge computing through physical layer authentication," *IEEE Access*, vol. 7, pp. 116 390–116 401, 2019.

[120] A. Y. Abyaneh, A. H. G. Foumani, and V. Pourahmadi, "Deep neural networks meet CSI-based authentication," *arXiv:1812.04715 [cs, eess, stat]*, Nov. 2018, arXiv: 1812.04715. [Online]. Available: http://arxiv.org/abs/1812.04715

[121] A. Hindupur, "The-gan-zoo: A list of all named GANs!" [Online]. Available: https://github.com/hindupuravinash/the-gan-zoo

[122] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv:1511.06434 [cs]*, Nov. 2015, arXiv: 1511.06434. [Online]. Available: http://arxiv.org/abs/1511.06434

[123] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv:1412.6806 [cs]*, Apr. 2015, arXiv: 1412.6806. [Online]. Available: http://arxiv.org/abs/1412.6806

[124] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," 2015. [Online]. Available: https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

[125] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, June 2015, pp. 448–456. [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.html

[126] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning* (ICML'10). Haifa, Israel: Omnipress, 2010, pp. 807–814. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104425

[127] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," *Proceedings of ICML*, vol. 30, p. 6, 2013.

[128] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv:1505.00853 [cs, stat]*, Nov. 2015, arXiv: 1505.00853. [Online]. Available: http://arxiv.org/abs/1505.00853

[129] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 2234–2242. [Online]. Available: http://papers.nips.cc/paper/6125-improved-techniques-for-training-gans.pdf

[130] L. Theis, A. v. d. Oord, and M. Bethge, "A note on the evaluation of generative models," *arXiv:1511.01844 [cs, stat]*, Apr. 2016, arXiv: 1511.01844. [Online]. Available: http://arxiv.org/abs/1511.01844

[131] Y. Wu, Y. Burda, R. Salakhutdinov, and R. Grosse, "On the quantitative analysis of decoder-based generative models," *arXiv:1611.04273 [cs]*, June 2017, arXiv: 1611.04273. [Online]. Available: http://arxiv.org/abs/1611.04273

[132] S. Chintala, "How to train a GAN." NIPS 2016 Workshop on Adversarial Training: NIPS 2016 Workshop on Adversarial Training, vol. NIPS 2016 Workshop on Adversarial Training. [Online]. Available: https://www.youtube.com/watch?v=myGAju4L7O8

[133] S. Chintala, E. Denton, M. Arjovsky, and M. Mathieu, "soumith/ganhacks," Dec. 2019, original-date: 2016-12-09T16:09:27Z. [Online]. Available: https://github.com/soumith/ganhacks

[134] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980 [cs]*, Jan. 2017, arXiv: 1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980

[135] K. Armanious, C. Jiang, M. Fischer, T. Küstner, T. Hepp, K. Nikolaou, S. Gatidis, and B. Yang, "MedGAN: Medical image translation using GANs," *Computerized Medical Imaging and Graphics*, p. 101684, Nov. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0895611119300990

[136] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua, "CVAE-GAN: Fine-grained image generation through asymmetric training," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2764–2773.

[137] T. J. O'Shea, T. Roy, N. West, and B. C. Hilburn, "Physical layer communications system design over-the-air using adversarial networks," in *2018 26th European Signal Processing Conference (EUSIPCO)*, Sep. 2018, pp. 529–532.

[138] Q. Li, H. Qu, Z. Liu, N. Zhou, W. Sun, S. Sigg, and J. Li, "AF-DCGAN: Amplitude feature deep convolutional GAN for fingerprint construction in indoor localization systems," *IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 1–13, 2019.

[139] D. Roy, T. Mukherjee, M. Chatterjee, E. Blasch, and E. Pasiliao, "RFAL: Adversarial learning for RF transmitter identification and classification," *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2019.

[140] D. Roy, D. T. Mukherjee, M. Chatterjee, and D. E. L. Pasiliao, "Detection of rogue RF transmitters using generative adversarial nets," *IEEE Wireless Communications and Networking Conference*, p. 7, 2019.

[141] R. Alshinina, "A highly accurate deep learning based approach for developing wireless sensor network middleware," Thesis, Sep. 2018. [Online]. Available: https://scholarworks.bridgeport.edu/xmlui/handle/123456789/2492

[142] Y. Chapre, A. Ignjatovic, A. Seneviratne, and S. Jha, "CSI-MIMO: Indoor Wi-Fi fingerprinting system," in *39th Annual IEEE Conference on Local Computer Networks*, Sep. 2014, pp. 202–209.

[143] L. Xiao, L. J. Greenstein, N. B. Mandayam, and W. Trappe, "Using the physical layer for wireless authentication in time-variant channels," *IEEE Transactions on Wireless Communications*, vol. 7, no. 7, pp. 2571–2579, July 2008.

[144] F. Chollet, et al., *Keras*, 2015. [Online]. Available: https://keras.io

[145] M. Abadi, et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: https://www.tensorflow.org/

[146] MATLAB, *version 9.6.0.1072779 (2019a)*. Natick, Massachusetts: The Mathworks Inc., 2019.

[147] A. Alkhateeb, "DeepMIMO: A generic deep learning dataset for millimeter wave and massive MIMO applications," in *Proc. of Information Theory and Applications Workshop (ITA)*, San Diego, CA, Feb. 2019.

[148] Remcom, "Wireless InSite," https://www.remcom.com/wireless-insite.

[149] A. Goldsmith, *Wireless Communications*. Cambridge University Press, Aug. 2005.

[150] L. Xiao, A. Reznik, W. Trappe, C. Ye, Y. Shah, L. Greenstein, and N. Mandayam, "PHY-authentication protocol for spoofing detection in wireless networks," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, Dec. 2010, pp. 1–6.

[151] S. Chintala, "How to train a GAN," *NIPS 2016 Workshop on Adversarial Training*, 2016. [Online]. Available: https://github.com/soumith/ganhacks,https://www.youtube.com/watch?v=myGAju4L7O8

[152] A. Duel-Hallen, "Fading channel prediction for mobile radio adaptive transmission systems," *Proceedings of the IEEE*, vol. 95, no. 12, pp. 2299–2313, Dec. 2007.

[153] A. Duel-Hallen, S. Hu, and H. Hallen, "Long-range prediction of fading signals," *IEEE Signal Processing Magazine*, vol. 17, no. 3, pp. 62–75, May 2000.

[154] L. Hanzo, C. H. Wong, and M. S. Yee, *Adaptive Wireless Transceivers*. Chichester, UK: John Wiley & Sons, Ltd, Feb. 2002. [Online]. Available: http://doi.wiley.com/10.1002/047084776X

[155] X. Gao, J. Tanskanen, and S. Ovaska, "Comparison of linear and neural network-based power prediction schemes for mobile DS/CDMA systems," in *Proceedings of Vehicular Technology Conference - VTC*, Apr. 1996, vol. 1, pp. 61–65.

[156] C.-K. Wen, W.-T. Shih, and S. Jin, "Deep learning for massive MIMO CSI feedback," *IEEE Wireless Communications Letters*, vol. 7, no. 5, pp. 748–751, Oct. 2018.

[157] W. Jiang and H. D. Schotten, "Deep learning for fading channel prediction," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 320–332, 2020.

[158] C. Luo, J. Ji, Q. Wang, X. Chen, and P. Li, "Channel state information prediction for 5G wireless communications: A deep learning approach," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 227–236, Jan. 2020.

[159] W. Jiang and H. D. Schotten, "Neural network-based channel prediction and its performance in multi-antenna systems," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, Aug. 2018, pp. 1–6.

[160] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2813–2821.

[161] "Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) Radio Transmission and Reception." [Online]. Available: https://www.3gpp.org

[162] R. H. Clarke, "A statistical theory of mobile-radio reception," *The Bell System Technical Journal*, vol. 47, no. 6, pp. 957–1000, July 1968.

[163] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, Dec. 2014.

[164] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Statistics). New York, NY, USA: Springer-Verlag, 2001.

[165] A. Ng, "CS229: Machine Learning," 2020. [Online]. Available: https://see.stanford.edu/materials/aimlcs229/cs229-notes1.pdf

[166] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *arXiv:1206.5533 [cs]*, Sep. 2012, arXiv: 1206.5533. [Online]. Available: http://arxiv.org/abs/1206.5533

[167] D. Harris and S. Harris, *Digital Design and Computer Architecture*, 2nd ed. Elsevier, 2013.

[168] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade: Second Edition* (Lecture Notes in Computer Science), G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Heidelberg: Springer, 2012, pp. 9–48. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_3

[169] "Neural network: A complete beginners guide in 2019," Aug. 2019. [Online]. Available: https://gadictos.com/neural-network-pt1/

[170] MIT OpenCourseWare, "12a: Neural Nets," Apr. 2016. [Online]. Available: https://www.youtube.com/watch?v=uXt8qF2Zzfo

[171] R. Livni, S. Shalev-Shwartz, and O. Shamir, "On the computational efficiency of training neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1* (NIPS'14). Cambridge, MA, USA: MIT Press, Dec. 2014, pp. 855–863.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California