

2023-09-12

Covert Communication in Autoencoder Wireless Systems

Mohammadi, Ali Teshnizi

Mohammadi, A. T. (2023). Covert communication in autoencoder wireless systems (Master's thesis, University of Calgary, Calgary, Canada). Retrieved from <https://prism.ucalgary.ca>.

<https://hdl.handle.net/1880/117029>

Downloaded from PRISM Repository, University of Calgary

UNIVERSITY OF CALGARY

Covert Communication in Autoencoder Wireless Systems

by

Ali Mohammadi Teshnizi

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA

SEPTEMBER, 2023

© Ali Mohammadi Teshnizi 2023

Abstract

The broadcast nature of wireless communications presents security and privacy challenges. Covert communication is a wireless security practice that focuses on intentionally hiding transmitted information. Recently, wireless systems have experienced significant growth, including the emergence of autoencoder-based models. These models, like other DNN architectures, are vulnerable to adversarial attacks, highlighting the need to study their susceptibility to covert communication. While there is ample research on covert communication in traditional wireless systems, the investigation of autoencoder wireless systems remains scarce. Furthermore, many existing covert methods are either detectable analytically or difficult to adapt to diverse wireless systems.

The first part of this thesis provides a comprehensive examination of autoencoder-based communication systems in various scenarios and channel conditions. It begins with an introduction to autoencoder communication systems, followed by a detailed discussion of our own implementation and evaluation results. This serves as a solid foundation for the subsequent part of the thesis, where we propose a GAN-based covert communication model. By treating the covert sender, covert receiver, and observer as generator, decoder, and discriminator neural networks, respectively, we conduct joint training in an adversarial setting to develop a covert communication scheme that can be integrated into any normal autoencoder. Our proposal minimizes the impact on ongoing normal communication, addressing previous works shortcomings. We also introduce a training algorithm that allows for the desired tradeoff between covertness and reliability. Numerical results demonstrate the establishment of a reliable and undetectable channel between covert users, regardless of the cover signal or channel condition, with minimal disruption to the normal system operation.

Acknowledgements

I must express my heartfelt gratitude to my dedicated supervisor, **Professor Majid Ghaderi**, whose unwavering guidance, expertise, and encouragement were instrumental in shaping this thesis. I am also immensely thankful to his esteemed colleague, **Professor Dennis Goeckel**, for his valuable insights and contributions that significantly enriched my research journey.

I extend my sincere appreciation to my esteemed examiner committee members, **Professor Abraham Fapojuwo** and **Professor Katie Ovens**, for their meticulous examination, constructive feedback, and thought-provoking suggestions that greatly improved the quality of this work.

In closing, I am indebted to my lovely parents, Maryam and Kheyrollah, for their unending love, support, and belief in my abilities. Had it not been for their enduring encouragements and sacrifices, I would have never achieved my milestones in my life. And to my beloved wife, Nazanin, whose companionship and unwavering faith in me provided the strength to overcome challenges and persevere through this academic endeavor. Your presence consistently gave me the courage to persist without faltering, for which I am forever grateful!

In Loving Memory of the Students of Flight PS752...

Table of Contents

Abstract	ii
Acknowledgements	iii
Dedication	iv
Table of Contents	v
List of Figures and Illustrations	viii
List of Tables	x
List of Symbols, Abbreviations and Nomenclature	xi
1 Introduction	1
1.1 Thesis Motivation	1
1.2 Thesis Objective	4
1.2.1 Autoencoder Wireless Communication Models	6
1.2.2 Covert Communication Model	7
1.3 Thesis Contributions	8
1.3.1 Comprehensive Evaluation of Autoencoder-based Wireless Models	8
1.3.2 Input and Channel Independent GAN-based Covert Model	9
1.3.3 Achieving a Controllable Trade-off between Covertiness and Performance through the Training Algorithm	10
1.3.4 Comprehensive Experimental Results	11
1.4 Thesis Organization	11
2 Background and Related Works	13
2.1 Artificial Neural Networks	13
2.1.1 Autoencoder Networks	16
2.1.1.1 Network Architecture	16
2.1.1.2 Training and Loss Function	17
2.1.1.3 Layers Composition	18
2.1.1.4 Variations	19
2.1.2 Generative Adversarial Networks (GANs)	24
2.1.2.1 Network Architecture	25

2.1.2.2	Training and Loss Function	26
2.1.2.3	Conditional GANs	27
2.2	Covert Communication	28
2.2.1	Introduction to Covert Communication	29
2.2.2	Taxonomy of Covert Communication Techniques	31
2.2.3	Covert Communication At Physical Layer	33
2.2.4	Traditional and Deep Learning Based Techniques	34
2.3	Related Works	36
2.3.1	Image Steganography	36
2.3.2	Traditional Covert Communication Techniques	38
2.3.3	Deep-Learning-Based Covert Communication Techniques	39
2.3.3.1	CNN-Based Techniques	39
2.3.3.2	GAN-Based Techniques	40
2.4	Implementation and Libraries	42
2.4.1	PyTorch	42
2.4.2	Principal Component Analysis (PCA)	42
2.4.3	Matplotlib	43
2.4.4	Training Hardware Setup	43
3	Autoencoder Wireless Systems	44
3.1	An Overview of Autoencoder Wireless Systems	44
3.1.1	Internal Neural Network Architecture	45
3.1.2	Training Procedure and Evaluation Metrics	46
3.2	Single/Multi-User Autoencoder Wireless Systems	47
3.3	System Models	49
3.3.1	Single-User Communication Model	50
3.3.2	Multi-User Communication Model	51
3.4	Neural Network Design	52
3.4.1	Embedding Layer	52
3.4.2	Encoder Network	53
3.4.3	Channel Model Layer	53
3.4.4	Equalization Layers	54
3.4.5	Decoder Network	54
3.5	Evaluation	55
3.5.1	Single-User Autoencoder's Performance	56
3.5.1.1	Methodology	56
3.5.1.2	Results	57
3.5.2	Multi-User Autoencoders' Performance	62
3.5.2.1	Methodology	62
3.5.2.2	Results	63
4	GAN-Based Covert Communication	69
4.1	System Model	69
4.2	GAN-Based Covert Design	73
4.2.1	General Formulation	75

4.2.1.1	Reliability	75
4.2.1.2	Low Interference	76
4.2.1.3	Undetectability	77
4.2.2	Neural Network Architecture	79
4.2.2.1	Generator (Alice)	79
4.2.2.2	Decoder (Bob)	80
4.2.2.3	Discriminator (Willie)	80
4.3	Evaluation	81
4.3.1	Methodology	81
4.3.2	Training Procedure	83
4.3.3	Single-User Experiments	84
4.3.4	Multi-User Experiments	86
4.3.5	Undetectability	88
4.3.6	Constellation Diagrams	89
4.3.7	Results Validation	90
5	Conclusion and Future Work	91
5.1	Thesis Summary	92
5.2	Future Work	94
	Bibliography	96

List of Figures and Illustrations

1.1	A military example of wireless covert communication involving an UAV. . . .	3
2.1	Mathematical model of an artificial neuron [28].	14
2.2	Neural network architecture of an Autoencoder model.	17
2.3	A convolutional autoencoder example trained on MNIST. The input image is encoded to a compressed representation and then decoded [4].	19
2.4	GAN training procedure.	25
2.5	Comparing the original GAN method to cGAN, the generator (G) and discriminator (D) neural networks are trained on real and fake samples in both methods [34].	28
2.6	A typical covert communication scenario that involves covert users Alice and Bob, and detector Willie.	30
2.7	The neural network architecture of an example GAN-based Image Steganography [43].	36
2.8	The dirty constellation consists of a cover QPSK constellation and a covert constellation. [10].	38
3.1	An end-to-end autoencoder-based communication system.	45
3.2	An overview of the single-user communication model with a detailed view of the UserRX decoder network.	50
3.3	An overview of the multi-user communication model with a detailed view of the BaseRX decoder network.	52
3.4	Theoretical BPSK and QPSK constellation diagrams.	57
3.5	Autoencoder BPSK and QPSK equivalent constellation diagrams.	58
3.6	Heatmap displaying the autoencoder model's performance in terms of BLER.	59
3.7	Comparing the BLER performance of autoencoder models in the single-user case with three different equalizations.	61
3.8	Autoencoders' performance in terms of BLER over a range of SNR values is evaluated in our single-user case.	62
3.9	Constellation diagrams of QPSK equivalent autoencoders for two messages .	64
3.10	Constellation diagrams of the autoencoder models for all messages projected onto a 2D plane.	65
3.11	Constellation diagrams of the autoencoder models after applying PCA to reduce dimensionality.	66
3.12	Comparing the BLER performance of autoencoder models in the multi-user case with three different equalizations.	66

3.13	The block error rates (BLERs) of our trained autoencoders compared with simulated results.	67
4.1	Overall architecture of our system model in the single-user scenario.	70
4.2	The overall architecture of our system model in the multi-user scenario.	72
4.3	Evaluation results of our covert and autoencoder models during the training process show the system reaches a stable point after successful training.	84
4.4	Single-user covert models' performance over AWGN channel for different covert data rates on a range of SNRs.	85
4.5	Single-user covert models' performance over Rayleigh fading channel for different covert data rates on a range of SNRs.	85
4.6	Single-user covert models' performance over Rician fading channel for different covert data rates on a range of SNRs.	85
4.7	Multi-user covert models' performances over AWGN channel for systems with different numbers of users over a range of SNRs.	87
4.8	Multi-user covert models' performances over Rayleigh channel for systems with different number of users over a range of SNRs.	87
4.9	Multi-user covert models' performances over Rician channel for systems with different number of users over a range of SNRs.	87
4.10	Comparing AWGN, Rayleigh and Rician fading channels constellation clouds of a sample signal.	89

List of Tables

2.1	Comparison of different variants of Autoencoder Networks.	20
3.1	Autoencoder's detailed network architecture in the single-user and multi-user case.	55
4.1	Alice, Bob, and Willie's detailed network architecture in the single-user and multi-user case.	81

List of Symbols, Abbreviations and Nomenclature

Symbol or abbreviation	Definition
1D Conv	One-Dimensional Convolution
ANN	Artificial Neural Network
ARP	Address Resolution Protocol
ASK	Amplitude Shift Keying
AWGN	Additive White Gaussian Noise
BCE	Binary Cross Entropy
BER	Bit Error Rate
BLER	Block Error Rate
BPSK	Binary Phase Shift Keying
CDMA	Code Division Multiple Access
cGAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSS	Chirp Spread Spectrum
DAE	Denoising Autoencoder
dB	Decibel
DNN	Deep Neural Network
DNS	Domain Name System
ELBO	Evidence Lower Bound
EVM	Error Vector Magnitude
FGM	Fast Gradient Method
FSK	Frequency Shift Keying
GAN	Generative Adversarial Network
GHz	Gigahertz
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HTTP	Hypertext Transfer Protocol
I/Q	In-phase/Quadrature
ICMP	Internet Control Message Protocol
IOT	Internet of Things
IP	Internet Protocol

KL	Kullback-Leibler
L1	Lasso Regression
LSTM	Long Short-Term Memory
MAC	Media Access Control Address
MHz	Megahertz
ML	Machine Learning
MNIST	Modified National Institute of Standards and Technology
MSE	Mean Squared Error
OFDMA	Orthogonal Frequency Division Multiple Access
PCA	Principal Component Analysis
PE	Processing Element
PSK	Phase Shift Keying
QPSK	Quadrature Phase Shift Keying
ReLU	Rectified Linear Unit
RF	Radio Frequency
RX	Receiver
SDR	Software Defined Radio
SNR	Signal-to-Noise Ratio
TanH	Hyperbolic Tangent
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TX	Transmitter
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
VAE	Variational Autoencoder

Chapter 1

Introduction

1.1 Thesis Motivation

In recent years, information and communication technology have witnessed major improvements in many different aspects. The developing need of staying connected and accessing information and resources without being physically tethered to a device or network made wireless communication systems to be at the center of focus among all the communication means. The emergence of novel services and applications such as smart cities, autonomous vehicles, remote medical surgeries, cloud-computing, artificial intelligence, and internet-of-things (IOT) are leading to proliferation of wireless technologies, including cellular networks, Wi-Fi, and Bluetooth.

Due to the broadcast nature of wireless channels, considerable attention has been given to the security and privacy aspects of wireless communications. While the content of messages (i.e., the information transmitted over the channel) can be protected against unauthorized access using cryptography or physical-layer security techniques [65], there are occasions when hiding the very existence of the communication channel is as vital as securing the

communicated messages themselves. Examples of such situations include military operations, cyber espionage, social unrest, or privacy concerns of communication parties. All of the aforementioned use cases have motivated the study of hidden communication channels, which are referred to as “covert communication” [29, 7] in the literature.

The preliminary attempt to obtain covertness started with the study of spread spectrum almost a century ago, with the main purpose of hiding military communications [48]. The idea was to transmit the signal over a wide frequency band, which would make it harder to locate and identify the original signal amidst the background noise. Many works continued to further examine different aspects of this idea [44, 61]. However, the fundamental performance limits of such work were unknown until recently when Bash et al. [5, 6] established a square root limit on the number of covert bits that can be reliably sent over an additive white Gaussian noise (AWGN) channel. Following this work, there has been a surge of interest in examining covert channels [52, 54, 50, 10].

Numerous works have studied the theoretical limits of covert communication over wireless systems in different scenarios [5, 54, 50, 30], but only a few works have focused on the practical implementation of covert communication [13, 10, 31, 36]. Many of these works involve an external factor that covert users rely on to build their covert communication, such as hardware impairments [36], the presence of a cooperative jammer [52], or the cooperation of a relay node [31, 25]. Additionally, the majority of works make some favorable assumptions for covert users, the accessibility of covert users to cover signals and modulation type [17], uncertainty in the knowledge of noise power at the detector’s receiver [20], neglecting the impact of the covert system on normal communication [36], and limiting the channel model to AWGN [36]. Imposing such restricted assumptions and dependencies eliminates

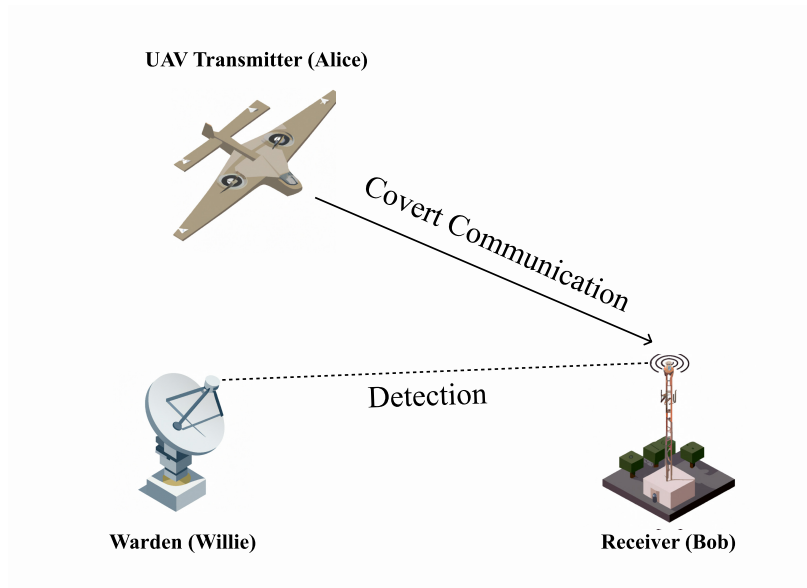


Figure 1.1: A military example of wireless covert communication. An unmanned aerial vehicle (UAV) transmitting highly sensitive data to a covert receiver while trying to evade detection by a vigilant Warden.

the generality of these covert models and makes it difficult for them to adapt to different system deployments with distinct conditions. Moreover, recent studies show that covert communication that causes noticeable divergence in the statistical properties of signals can be easily detected using analytical and steganalysis methods [2, 24]. In our work, we leverage the inherent channel noise present in common wireless communication systems. More specifically, our covert communication relies on an established implicit shared secret key between Alice and Bob, unbeknownst to Willie. This shared secret key is carefully formed through the joint training of Alice and Bob as encoder and decoder networks.

There is a large body of work that focuses on different schemes for covert communication based on traditional design approaches. However, wireless systems research has greatly expanded in recent years to consider the potential impact of machine learning (ML) approaches for a variety of problems [58]. In fact, various network optimization problems, which were

traditionally handled using statistical models, now leverage machine learning techniques [66]. Deep neural networks (DNNs) in particular, the major force in machine learning, have helped address several wireless problems, such as signal classification [38, 39, 59, 32], channel estimation [53], transmitter identification [45, 18], jamming, and anti-jamming [1, 2]. In a recent study, an end-to-end communication model based on deep learning emerged as a replacement for conventional modular-based designs [39]. In this new paradigm, the transmitter and receiver are designed based on DNNs that are jointly trained as the encoder and decoder of an autoencoder network [39]. The autoencoder network can be trained to learn the characteristics of the signal, such as its statistical properties, to develop modulation and coding techniques through a self-learning process. This eliminates the need for hand-crafted methods, giving the system the ability to determine the most effective way to modulate and encode data. Compared to traditional communication systems, this approach offers greater flexibility and robustness, as the autoencoder can adapt to varying channel conditions and noise levels without manual tuning [67]. Despite these benefits, Autoencoder wireless systems, similar to many other deep learning models, are highly susceptible to adversarial attacks [11], such as jamming [2], spoofing attacks [51], signal misclassification [46]. This motivates us to study the vulnerabilities that ML-based wireless communication systems might have against covert communications.

1.2 Thesis Objective

In this thesis, our primary objective is to design a novel covert communication model that overcomes many of the limitations and assumptions of existing approaches. To this end, we

aim to minimize the risk of detection by system monitors and ensure our covert model has a minimal disturbance to existing communication between entities of the system. However, before delving into the design of our covert communication model, it is crucial to gain a thorough understanding of the underlying communication system used by the existing entities, namely normal users. This understanding will serve as a foundation for effectively integrating our covert model into the system. In particular, our focus is on seamlessly integrating our covert model into autoencoder-based communication models. Our objective is twofold: to provide a comprehensive review of these communication models and to present our own implementation, showcasing the key findings that we have made in this area. Our covert scheme leverages the inherent noise effect of the communication channel and operates independently of external factors, eliminating the need for knowledge of cover signals or modulation types. Notably, in scenarios involving a single normal user, our scheme can operate without prior knowledge of the communication channel type. By training our covert models in an adversarial manner against the observer and carefully minimizing the impact on ongoing normal communication, we ensure that our covert signals blend seamlessly with the statistical properties of normal signals, making them difficult to detect using conventional analytical tools. It is important to highlight that while our method is specifically designed for autoencoder-based wireless communication systems, there are no limitations on integrating our model into conventional wireless communication systems.

1.2.1 Autoencoder Wireless Communication Models

The existing body of literature on autoencoder-based communication systems provides some insights into their functioning, but a comprehensive study that thoroughly investigates the implementation and evaluation of these systems across various channel models and system configurations is noticeably absent. Consequently, it becomes imperative for us to bridge this gap by designing, implementing, and rigorously evaluating our own versions of these systems.

The main rationale behind conducting a thorough study of autoencoder communication systems lies in the necessity to ensure the correct behavior and optimal performance of the autoencoder models when our covert model is integrated with them. By gaining a deep understanding of these systems, fine-tune the autoencoder models to ensure their optimal and standard performance.

In the initial phase of our thesis, we thoroughly explore the intricacies of autoencoder communication systems by conducting an in-depth review of existing literature. We draw inspiration from mainstream publications to gain insights into the implementation of these models. This enables us to effectively implement and evaluate our covert mechanism within communication systems that encompass diverse channel models and user configurations. By doing so, we ensure that our findings and conclusions are not limited to specific types of channels or system settings, but are applicable to a wide range of real-world scenarios.

By undertaking this comprehensive approach, we aim to contribute to the advancement of autoencoder-based communication systems. More importantly, we aim to provide a solid foundation for the integration of our covert model, which in turn ensures our covert model

is compatible and effective across various communication environments.

1.2.2 Covert Communication Model

The primary objective of this thesis is to develop a machine-learning based covert communication model by leveraging neural networks in an adversarial framework. Our approach surpasses previously introduced methods in terms of sophistication, rendering it highly resilient against detection.

In the literature, we can find several theoretical and empirical studies that have explored the fundamental limits of covert communication models, however, there is a lack of practical design and implementation guidance for real-world covert communication methods. Moreover, traditional approaches often rely on impractical assumptions that render them inapplicable in realistic settings. This has motivated us to seek covert mechanisms that can be feasibly implemented in real-life scenarios and minimize the reliance on improbable assumptions. Furthermore, many existing works rely on manually-engineered features for covert communication, which, if exposed, could lead to the termination of the covert channel by the observer. Even in recent approaches that utilize machine learning techniques for designing more sophisticated covert signals, the probability of detection by a careful observer remains relatively high. Even though these methods generate covert signals that closely resemble the system’s noise, it was shown that the observer can employ complex techniques to measure the divergence in signal distributions and identify the covert channel [2, 24]. Once the observer obtains an estimate of the transmitted covert signals, they can deactivate the entire channel by averaging out the covert signals.

In this thesis, we propose a novel covert communication technique where the covert users themselves learn to embed their messages into the transmitted signals as a noise vector, without explicit guidance on how to do so. Specifically, covert users learn to establish an implicit shared secret key between themselves, unbeknownst to the observer. The learning process takes place within an adversarial setting to ensure that the generated signals are indistinguishable from the expected noise in the system. Moreover, our covert communication design is independent of any prior knowledge of cover signals or the channel model, making it versatile and adaptable to various communication setups. By adopting this innovative approach, we aim to advance the field of covert communication and pave the way for more robust and secure communication systems.

1.3 Thesis Contributions

In this section, we offer a brief overview of the accomplishments made in this thesis. This includes a comprehensive examination of autoencoder wireless communication systems and our distinctive approach to their implementation. Our contributions are organized into three main subsections, which are outlined as follows:

1.3.1 Comprehensive Evaluation of Autoencoder-based Wireless Models

To ensure a solid foundation for our work, we undertake a comprehensive review of autoencoder-based communication systems, which is a recent advancement in wireless communication. To achieve this, we extended and refined the design and implementation of these models,

ensuring their robustness and proper functionality. Subsequently, we conducted a comprehensive evaluation across diverse communication scenarios, including various channel models as well as single and multi-user communication scenarios. This rigorous evaluation allowed us to fine-tune the models and verify their correct behavior under different communication setups and channel configurations. It also facilitated a comprehensive analysis and performance evaluation of our covert communication scheme, providing valuable insights into its effectiveness and suitability in real-world scenarios.

1.3.2 Input and Channel Independent GAN-based Covert Model

We propose a novel covert communication approach using generative adversarial networks (GANs) that utilizes an input-agnostic generator and discriminator network to represent the covert sender and detector, respectively. These networks are trained in an adversarial manner, similar to the training process of GANs, allowing our scheme to operate independently of specific cover signals, waveforms, and modulation types used in wireless systems. This flexibility enables our approach to be applied across a wide range of communication scenarios without being limited by the specifics of the underlying wireless technology.

To evaluate the effectiveness and versatility of our scheme, we conducted extensive experiments involving three different channel models: Additive White Gaussian Noise (AWGN), Rayleigh Fading, and Rician Fading. By training our model on these diverse channel models, we demonstrated its adaptability to varying channel conditions and its ability to maintain robust performance even in the presence of different noise levels. Importantly, our approach does not require prior knowledge of the specific channel model or the characteristics of the

noise, making it highly practical and flexible in real-world scenarios where such information may not be readily available.

1.3.3 Achieving a Controllable Trade-off between Coverttness and Performance through the Training Algorithm

We developed a training procedure that enables us to attain any desired trade-off between the level of coverttness and the system's performance (i.e., communication rate) regardless of the number of normal users in the system. This is accomplished by utilizing a regularized loss function for covert users, enabling them to prioritize their objectives based on the specific communication scenario. This dynamic training procedure allows us to tailor the covert communication scheme to meet the specific needs of different scenarios. In situations where coverttness is of utmost importance, the system can be configured to prioritize concealing the covert communication channel at the expense of a slightly reduced communication rate and reliability. On the other hand, in scenarios where maximizing the communication rate and accuracy is critical, the system can be optimized to achieve higher performance while still maintaining an acceptable level of coverttness.

By offering this level of flexibility in training, our approach empowers covert users to adapt their communication strategy based on the specific requirements of the system, thereby ensuring optimal performance and coverttness in a wide range of real-world scenarios.

1.3.4 Comprehensive Experimental Results

We conducted multiple experiments to demonstrate that our scheme can be integrated into communication systems with both single and multiple normal users under various channel conditions. Notably, in all cases, including single and multi-user scenarios for the three channel models, our (8,1) covert model is demonstrated to have a negligible impact on the normal users' BLER, while establishing a reliable covert communication link and consistently deceiving the detector at various SNRs. Furthermore, our experiments highlighted that there is a degree-of-freedom effect in our scheme, where increasing the number of users affects the performance of the covert and normal communication systems in the fading channels.

1.4 Thesis Organization

This thesis has been divided into 5 chapters. The following is a summary of each chapter's content:

- **Chapter 1** presents the primary objectives and motivations of our study, and provides a brief summary of contributions made throughout this research.
- **Chapter 2** provides the necessary background information required to understand the concepts discussed and implemented in the thesis. It covers key topics such as artificial neural networks, autoencoder networks, GAN networks, and covert communication. Additionally, this chapter presents a review of pertinent literature that is relevant to this thesis. In the end, it provides a concise overview of the tools, libraries, and infrastructures employed during the implementation of this thesis.

- **Chapter 3** serves as an introduction to autoencoder wireless systems, presenting an overview of the fundamental concepts, neural network architecture, and training procedure associated with these systems. The chapter then delves into the various communication scenarios explored, encompassing both single and multi-user setups. The neural network design of the implemented autoencoders is described in detail, outlining the functionality of each component. The chapter concludes with an extensive evaluation of the system’s performance across different communication setups and channel configurations, emphasizing key findings derived from the analysis.
- **Chapter 4** introduces our proposed GAN-based covert communication model in the context of autoencoder-based communication systems. The chapter begins by discussing the integration of our covert model into these existing systems and outlining the objectives and roles of the involved entities. Detailed insights into the design of our covert model are then presented, including the formulation of objectives as loss functions and the functioning of the neural network components. Finally, the chapter evaluates the performance of our model against the essential criteria for covert communication, providing an assessment of its effectiveness.
- **Chapter 5** concludes our work and discusses potential avenues for future research.

Chapter 2

Background and Related Works

This chapter aims to provide readers with the necessary background knowledge to understand the proposed solutions presented in this thesis. Section 2.1 offer an overview of artificial neural networks and relevant deep learning architectures utilized in this thesis, including generative adversarial networks and autoencoder networks. Section 2.2 gives an introduction to covert communication and their taxonomies. Finally, in section 2.3, we review some of the most relevant studies related to our research.

2.1 Artificial Neural Networks

Artificial neural networks (ANNs) are computer programs that are inspired by how the human brain biologically processes information [28]. Rather than being programmed explicitly, ANNs are designed to identify patterns and correlations in data through a process that is commonly referred to as training. These networks are made up of individual processing elements (PEs) or artificial neurons, which are connected by weights to form structured

layers. The weighted inputs of each neuron are summed and passed through a transfer function to produce a single output, and the behavior of the network is determined by the transfer functions of its neurons, its learning rule, and its architecture. ANNs are highly capable of extracting nonlinear relationships within data, making them useful in various applications, including but not limited to image recognition, prediction, and classification.

A collection of weighted inputs, a transfer function, and a bias term are used to mathematically define ANNs. To get an activation value, the weighted inputs are summed together with the bias term. The activation value is then fed into the transfer function, which generates an output. This procedure is repeated for each neuron in the network, with one neuron’s output acting as the input for the next. During training, the weights and biases are adjusted to minimize the difference between the predicted and actual output. Fig. 2.1 and the equation below outlines how the output of each neuron is computed.

$$y_k = \varphi\left(\sum_{j=1}^p (x_j * w_{kj})\right). \tag{2.1}$$

where y_k is the neuron’s output, φ is the transfer or activation function, x_0, x_1, \dots, x_p are

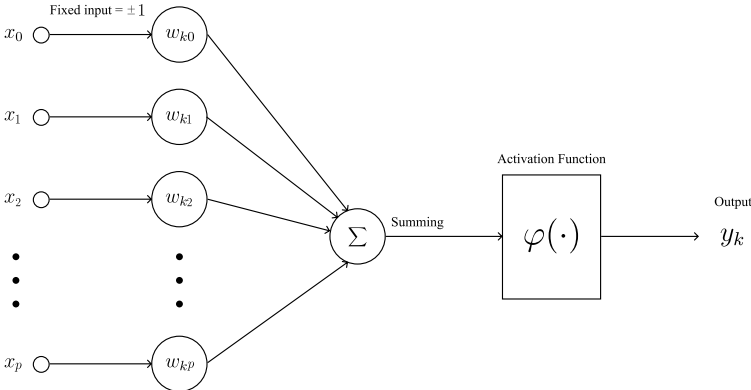


Figure 2.1: Mathematical model of an artificial neuron [28].

the input signals, and w_0, w_1, \dots, w_p are the weights associated with the inputs. The input signal at index 0, also known as the bias parameter, has a constant value and it allows a neuron to offset its output independently of its inputs. To capture more complex and non-linear patterns in the input data, activation or transfer functions are employed. Additionally, activation function keeps the output within a certain scale to ensure that it is normalized and bounded. The most common utilized activation functions are: *Sigmoid*, *Hyperbolic Tangent (Tanh)*, *Rectified Linear Unit (ReLU)*, *Leaky ReLU*, *Softmax*.

The goal of training a neural network is to identify the optimal set of parameters that can accurately map inputs to outputs. During the training phase, data is fed to the network through the input layer and passed to the subsequent layers, known as the hidden layers. The number of hidden layers in a neural network depends on the complexity of the representation it needs to learn and can range from one to many. In the end, the output layer yields the result for a give input and the output error is measured. This error is calculated by a mathematical function called *Loss Function* that measures the difference between the predicted output and the expected output. The most common loss functions can be listed as:

$$\text{Mean Squared Error (MSE)} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2. \quad (2.2)$$

$$\text{Binary Cross-Entropy (BCE)} = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]. \quad (2.3)$$

$$\text{Kullback-Leibler (KL) Divergence} = \sum_{c=1}^m \hat{y}_c \log \frac{\hat{y}_c}{y_c}. \quad (2.4)$$

where y and \hat{y} , respectively, are the expected and predicted outputs, and n and m are the number of samples and number of classes.

2.1.1 Autoencoder Networks

Autoencoders are a type of neural network that has become increasingly popular in recent years. They have a long history in the field of neural networks, dating back to the 1980s. The concept of autoencoders was first introduced as a type of neural network for dimensionality reduction by Rumelhart, Hinton, and Williams in their 1986 paper titled "Learning representations by back-propagating errors". Since then, autoencoders have been widely studied and improved upon. In the early 1990s, they were used for unsupervised learning and feature extraction. In the early 2000s, they were applied to speech recognition and image classification tasks. Today, they are commonly used for data compression, dimensionality reduction, feature learning, and anomaly detection. Autoencoders work by learning to encode input data into a lower-dimensional representation, which can then be used to reconstruct the original data. This process is similar to how humans learn to recognize patterns and simplify complex information. Autoencoders have a wide range of applications in different fields, including computer vision, natural language processing, and finance. They have the potential to transform how we process and analyze large datasets, and their versatility makes them an exciting area of research in the field of machine learning.

2.1.1.1 Network Architecture

The architecture is composed of two main parts: an encoder and a decoder. Fig. 2.2 shows an overview of this architecture. The encoder maps the high-dimensional input data into a lower-dimensional space, which is called the latent space. Inversely, the decoder reconstructs the original data from the compressed representation. The primary objective

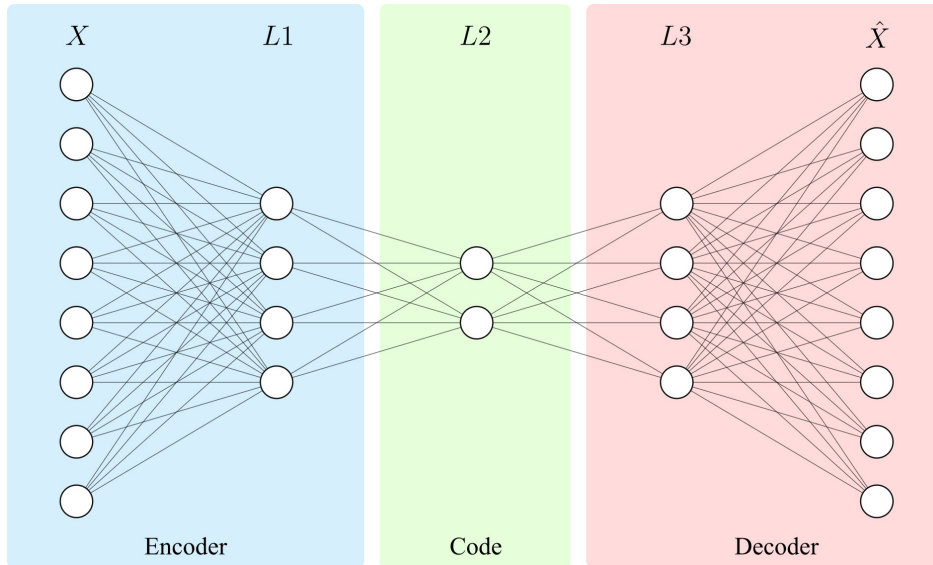


Figure 2.2: Neural network architecture of an Autoencoder model.

of an autoencoder network is to minimize the difference between the original input and the output reconstruction. Autoencoders have demonstrated their effectiveness in various applications, including data compression, feature learning, dimensionality reduction, and anomaly detection.

2.1.1.2 Training and Loss Function

Autoencoders are trained via unsupervised learning, which implies that they don't need labeled data. The primary objective of training an autoencoder is to reduce the difference between the input data and the reconstructed output. To measure this difference, a loss function such as Mean Squared Error (MSE) is commonly used. The formula for MSE can be found in Equation 2.5. The MSE computes the average difference between the input and output over all the examples in the training set.

$$L(\theta, \varphi) = \frac{1}{n} \sum_{i=1}^n (x_i - f_{\theta}(g_{\varphi}(x_i)))^2, \quad (2.5)$$

Where x_i is the input data, $g(\cdot)$ is the encoder function, θ are its parameters, $f(\cdot)$ is the decoder function that reconstruct the original data, φ are its parameters, n is the number of training samples. During training, the input data is passed through the encoder to obtain the compressed representation, which is then fed into the decoder to generate the reconstructed output. This process is repeated for multiple epochs until the network learns to extract the most important features of the input data and encode them in the compressed representation.

2.1.1.3 Layers Composition

Autoencoder networks can be constructed using different types of layers, each of which serves a specific purpose depending on the input data and desired output. The most common types of layers used in autoencoders are fully connected layers, convolutional layers, and recurrent layers such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU).

Fully connected layers are the simplest type of layer and are used in vanilla autoencoders, whereas convolutional layers are commonly used in autoencoders for image data, where they can learn spatial features efficiently. Recurrent layers, on the other hand, are used in autoencoders for sequential data like time series or text, where they can capture long-term dependencies in the data. Additionally, other types of layers such as pooling or activation layers can be added to customize the network's architecture. The versatility of autoencoders allows them to be tailored to various input data types and output tasks, and the choice of layer type can be adjusted to optimize performance. You can find a convolutional-based autoencoder in Fig. 2.3 that has been trained on the Modified National Institute of Standards and Technology (MNIST) dataset [12].

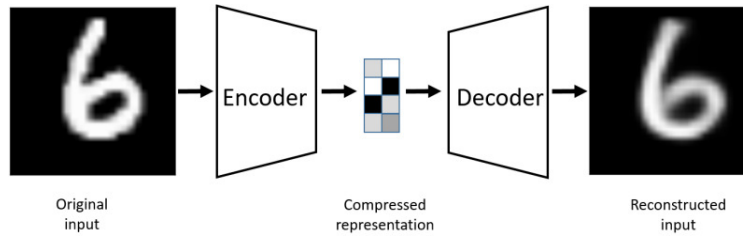


Figure 2.3: A convolutional autoencoder example trained on MNIST. The input image is encoded to a compressed representation and then decoded [4].

2.1.1.4 Variations

Two common variants of autoencoders are the variational autoencoder, which models the latent space as a probability distribution, and the denoising autoencoder, which removes noise from corrupted input data. In the following section, we will explore some important variants of autoencoders and discuss their distinct features and applications. We have also summarized this information in Table 2.1.

- Vanilla Autoencoder:** Vanilla autoencoders are a basic type of autoencoder networks that are composed of fully connected layers. The structure of vanilla autoencoders is often symmetrical, with the number of neurons in each layer gradually decreasing towards the bottleneck or latent space where the compressed representation is learned. The decoder architecture mirrors that of the encoder, gradually increasing the number of neurons until the output layer, which generates the reconstructed data. The middle layers of the autoencoder, also known as bottleneck or code layer, where the data is compressed, is a critical component of the network and is often much smaller than the input and output layers. This forces the network to learn an efficient representation of the input data that captures the most important features. Vanilla autoencoders can be utilized for unsupervised learning, where the network is trained on unlabeled data,

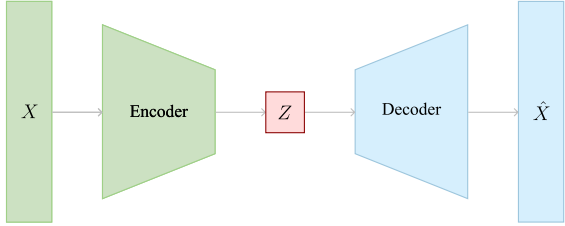
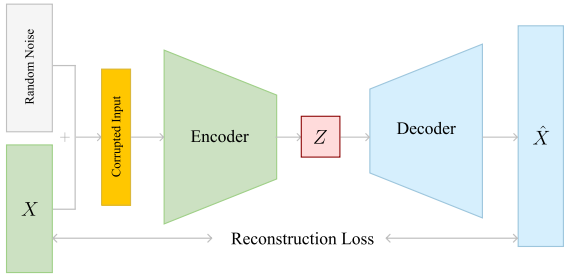
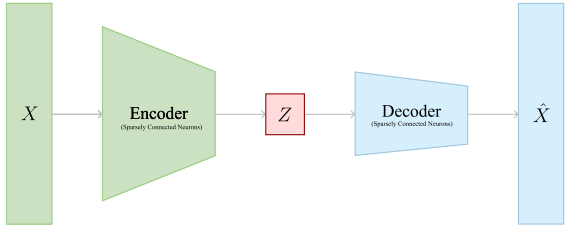
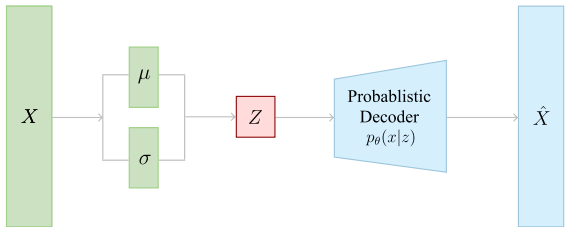
Type	Structure	Objective	Applications
Vanilla		Reconstruction	Data compression Dimensionality reduction Anomaly detection
Denoising		Noise reduction	Image denoising Speech denoising Anomaly detection
Sparse		Sparse representation	Feature extraction Image compression Anomaly detection
Variational		Data generation/manipulation	Image and text generation Data imputation Anomaly detection

Table 2.1: Comparison of different variants of Autoencoder Networks.

or for supervised learning tasks like classification or regression after being pre-trained on unlabeled data.

Vanilla autoencoders have numerous applications such as data compression, dimensionality reduction, feature learning, and anomaly detection. They can compress large datasets into a lower-dimensional representation, extract informative features, and remove redundant information, which aids in the visualization of high-dimensional data. Additionally,

they can identify outliers or unusual patterns in the data.

- **Sparse Autoencoders:** One of the key challenges in autoencoders, and many other machine learning models, is finding the optimal tradeoff between network's ability to fit the training data well (low bias) and its ability to generalize to new data (low variance). A high-bias autoencoder may not fit the training data well, while a high-variance one may overfit the training data and perform poorly on new data. One of the methods to tackle this tradeoff is to use Sparse Autoencoders.

Sparse autoencoders incorporate a sparsity constraint during training to encourage the network to learn a compact representation of the input data, reducing overfitting and improving generalization performance. By constraining the network's capacity, the bias-variance tradeoff is balanced, while still allowing the network to capture important features of the data.

Enforcing sparsity regularization in autoencoders can be achieved through introducing L1 regularization, which is effective in inducing sparsity. This leads to the autoencoder optimization objective being modified as:

$$L(\theta, \varphi) = \frac{1}{n} \sum_{i=1}^n (x_i - f_{\theta}(g_{\varphi}(x_i)))^2 + \lambda \sum_{j=1} |a_j|, \quad (2.6)$$

where a_j represents the activation value of the j -th hidden layer, and the subscript j iterates over all the hidden layers' activations.

Sparse autoencoders have several applications in different fields such as computer vision, speech recognition, recommendation system, and natural language processing.

- Denoising Autoencoders:** Denoising autoencoders are a type of autoencoder designed to remove noise from input data. Unlike regular autoencoders, they work by constructing a compressed representation of the input data, which is then used to recover the original, noise-free data. Denoising autoencoders offer several advantages, including enhanced resistance to noisy or distorted data, and the ability to learn more discriminative input data characteristics. They have been successfully used for tasks such as image denoising, speech denoising, and anomaly detection. Overall, denoising autoencoders are a powerful tool for improving the quality and utility of data in real-world scenarios.
- By corrupting input data with noise, a denoising autoencoder (DAE) is trained to reconstruct uncorrupted data. This is accomplished by utilising a loss function to minimise the reconstruction error between the input and output data. The loss function can be written as follows:

$$L(\theta, \varphi) = \frac{1}{n} \sum_{i=1}^n (x_i - f_{\theta}(g_{\varphi}(\hat{x}_i)))^2. \quad (2.7)$$

Where $\hat{x}_i = x_i + \epsilon$; $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is the corrupted input.

- Variational Autoencoders:** The Variational Autoencoders (VAE) model [27] has significantly enhanced the representation capabilities of autoencoders. VAEs are a type of generative models that integrate deep learning and Bayesian inference to learn a compact and interpretable latent representation of input data. They have attracted considerable attention in recent years due to their capacity to generate novel and high-quality samples, as well as their applicability to various tasks, such as image and text

generation, data imputation, and anomaly detection. Unlike traditional autoencoders, VAEs learn a distribution over the latent space, which enables the generation of diverse and meaningful samples by sampling from the learned distribution. VAEs also offer a principled framework for data generation and manipulation, enabling controllable and meaningful variations in the generated samples.

VAE assumes that the input data is generated by sampling from a latent distribution $p(z)$, which is then passed through a conditional probability distribution $p_\theta(x|z)$, where θ represents the decoder network parameters. The goal is to learn the latent distribution $p(z|x)$ that maximizes the log-likelihood of the input data x . However, computing this true posterior distribution $p(z|x)$ is difficult, so instead VAE uses a variational lower bound to approximate it. This bound is formulated as the Evidence Lower Bound (ELBO) that decomposes the data log-likelihood into two components: the Kullback-Leibler (KL) divergence between the approximated posterior distribution $q_\phi(z|x)$ and the prior distribution $p(z)$, and the expected log-likelihood of the data given the latent variable z . The ELBO is mathematically expressed as:

$$\mathcal{L}_{\text{ELBO}}(x; \theta, \phi) = -D_{\text{KL}}(q_\phi(z|x)||p(z)) + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] \quad (2.8)$$

where ϕ represents the parameters of the encoder network that approximates the posterior distribution, and $D_{\text{KL}}(q_\phi(z|x)||p(z))$ is the KL divergence between the approximate posterior and the prior. During training, the encoder network computes the mean μ and standard deviation σ of the approximate posterior distribution $q_\phi(z|x)$. The latent variable z is then sampled from the reparameterized Gaussian distribution. The loss

function used to optimize the VAE is the negative ELBO. By minimizing this loss, the VAE learns to generate samples that are close to the training set distribution while also forming an interpretable latent space.

2.1.2 Generative Adversarial Networks (GANs)

The introduction of generative adversarial networks (GANs) by Ian Goodfellow and colleagues in 2014 was motivated by the desire to develop a generative model capable of generating realistic samples of complex data, such as images, audio, and video. Prior to the development of GANs, autoregressive model variations and variational autoencoders were popular approaches to generative modeling. However, these methods had limitations in capturing complex data distributions and generating high-quality samples. GANs were a significant breakthrough in deep learning as they introduced a new way of generative modeling through adversarial training.

GANs have a wide range of applications including image and video synthesis, data augmentation, style transfer, super-resolution, and anomaly detection. They have been used to generate realistic images of faces, landscapes, and objects, as well as to create convincing deepfake videos. GANs have also been applied to medical imaging, such as generating synthetic MRI images, and to text and audio data. Additionally, GANs have been used for data augmentation in classification tasks and for enhancing low-resolution images. The ability of GANs to generate new data that follows the same distribution as the training data has made them a powerful tool for various creative and practical applications.

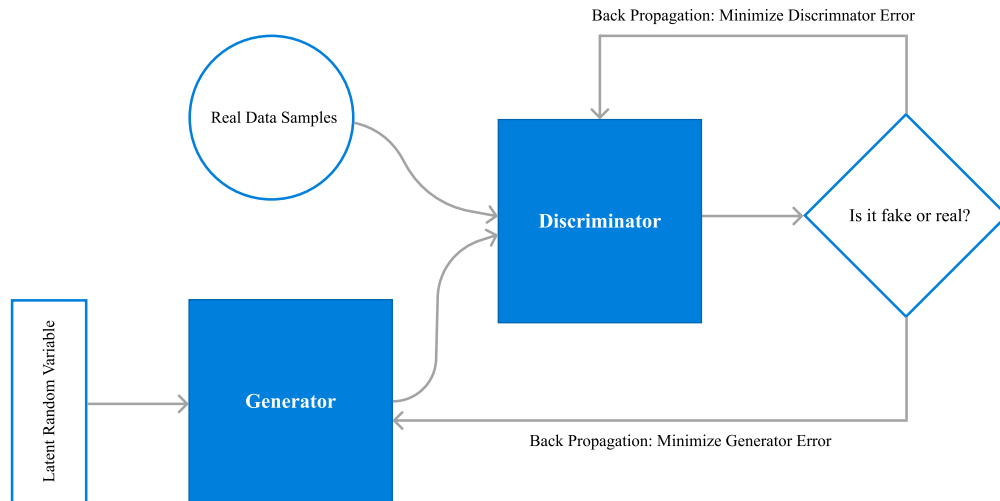


Figure 2.4: GAN training procedure.

2.1.2.1 Network Architecture

Two major components of GANs are *Generator* and *Discriminator* networks. The role of generator network is to deceive the discriminator network with its generated samples to accept fake data as real. In contrast, the discriminator objective is to detect any irregularities or anomalies in the generated samples and accurately classify them as either real or fake. In short, here is the intuition behind each of these networks:

Discriminator: The discriminator network is a supervised approach used as a classifier that distinguishes between real and fake data. It is trained on real data and provides feedback to the generator network.

Generator: It is an unsupervised learning approach that generates fake data based on the discriminator feedback. It is a neural network with hidden layers, activation, and loss function, aimed at fooling the discriminator by making it to classify a fake data as real.

The internal architecture of GANs can be modified in several ways to improve its performance, such as adding additional layers, modifying the loss functions, or incorporating additional

constraints on the generated samples. Some common modifications include the use of convolutional layers for image generation tasks, the addition of regularization terms to prevent mode collapse, and the incorporation of attention mechanisms to focus on important features in the input data. Despite their simple architecture, GANs have shown remarkable success in generating high-quality samples and have become a popular tool for various applications in computer vision, natural language processing, and audio synthesis.

2.1.2.2 Training and Loss Function

The key idea behind GANs is to train two neural networks, a generator and a discriminator, that compete against each other in a zero-sum game. The generator takes random noise as input and produces fake data samples, while the discriminator is trained to distinguish between real and fake samples. At every iteration of training, the generator is optimized to create more realistic samples that can deceive the discriminator, which, in turn, is optimized to distinguish better between real and fake samples. This cycle continues iteratively, with both networks improving until the generator can produce samples that are nearly indistinguishable from the real data.

The loss function consists of two parts: the generator loss and the discriminator loss. The generator loss minimizes the error for generate data samples to be classified as real samples by the discriminator network. Meanwhile, the discriminator loss minimizes the error for correctly classifying the fake samples generated by the generator and the real samples. The loss function can be expressed as:

$$L = \mathbb{E}_{x \sim p_d(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))], \quad (2.9)$$

Here, x is a real data sample, z is a random noise vector, D is the discriminator function, G is the generator function, and $p_d(x)$ and $p_z(z)$ represent the real data distribution and the noise distribution, respectively. The first term $\mathbb{E}_{x \sim p_d(x)}[\log D(x)]$ calculates the probability that the discriminator correctly classifies real samples as real. The second term $\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$ calculates the probability that the discriminator correctly classifies fake samples generated by the generator as fake. The GANs model is trained by minimizing the loss function with respect to the generator and discriminator parameters.

2.1.2.3 Conditional GANs

Conditional Generative Adversarial Networks (cGANs) are an extension of the original GANs framework that condition the generator on extra information such as class labels, text descriptions, or input images. This added information enables the generator to produce more specific and targeted outputs, resulting in improved performance in tasks like image synthesis, image-to-image translation, and text-to-image synthesis.

Both the generator and the discriminator in a cGAN receive additional conditioning information, which is commonly concatenated with the input noise or image tensor. Conditioning the generator on specific inputs allows it to learn to generate samples that meet certain constraints or qualities, such as a given class name or style. Furthermore, conditioning the discriminator on the same information can improve its ability to distinguish between real and fake samples that belong to a specific class or have specific characteristics.

cGANs feature a more complicated design and loss function that incorporates conditioning information than ordinary GANs. Conditioning information is often provided in the form of one-hot vectors or embedding vectors, and the loss function is altered to incorporate

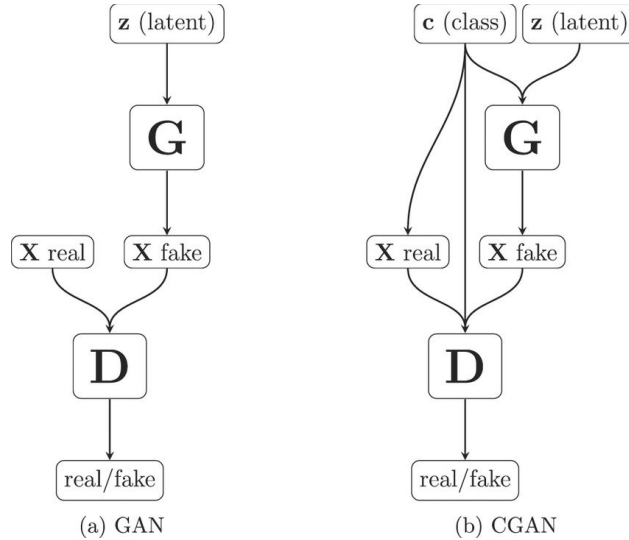


Figure 2.5: Comparing the original GAN method to cGAN, the generator (G) and discriminator (D) neural networks are trained on real and fake samples in both methods [34].

a term that quantifies the difference between the generated samples and the conditioning information. Because of their complexity and versatility, cGANs are a strong tool for a wide range of generating tasks.

The loss function for a cGAN can be expressed as:

$$L = \mathbb{E}_{x \sim p_d(x), y} [\log D(x, y)] + \mathbb{E}_{z \sim p_z(z), y} [\log(1 - D(G(z, y), y))]. \quad (2.10)$$

where y is the conditioning information and is usually provided to both the generator and discriminator as an additional input.

2.2 Covert Communication

In this section, we provide an overview of covert communication and discuss important methods used in this field. We begin with a brief introduction to covert communication in

Section 2.2.1. We then outline common taxonomies for covert communication in Section 2.2.2 and examine covert communication at the physical layer in Section 2.2.3. Finally, in Section 2.2.4, we discuss both traditional and ML-based methods and techniques for covert communication.

2.2.1 Introduction to Covert Communication

It was in 1973, when Lampson [29] coined the term covert communication (covert channels) in the context of monolithic systems as a mechanism by which a high-security process leaks information to a low-security process that would otherwise not have access to it. Although it was first found as a threat to single host systems, the potential of such channels in computer networks was shown to be significantly higher [62]. Today, we refer to as covert communication as a type of communication that is intentionally concealed from individuals or organizations that may wish to intercept or monitor it. This type of communication is often used in situations where the sender and receiver wish to protect the confidentiality and/or integrity of the information being transmitted. Some of these situations are military operations, cyber-espionage, business and law enforcement, social unrest, and private communication. The use of covert communication can be beneficial in these situations by protecting the confidentiality and integrity of sensitive information, allowing for the safe sharing of classified or private information and maintaining secrecy.

In a covert communication scenario, there are typically three parties involved: Alice, Bob, and Willie. Alice, acting as the covert sender, aims to transmit a secret message to Bob, the covert receiver, without being detected by Willie, who serves as the observer. The

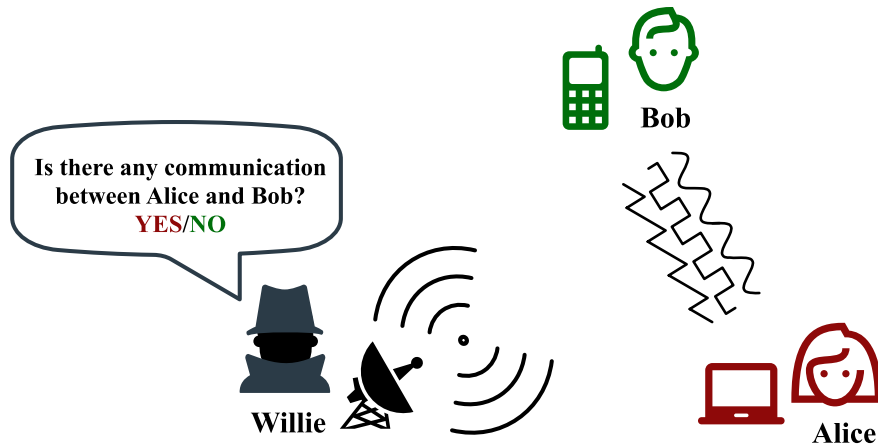


Figure 2.6: A typical covert communication scenario that involves Alice communicating with Bob in secret, while Willie attempts to detect if there is any communication between them.

primary objective of covert communication is to establish a concealed channel that hides both the existence and content of the communication from Willie, who represents potential adversaries or unauthorized individuals. Various techniques, such as steganography, spread spectrum modulation, or frequency hopping, are employed to embed the secret information within the regular communication channel. To ensure that the covert communications cannot be distinguished from background noise or genuine communication while maintaining a trustworthy and effective method of communication, Alice and Bob must use advanced encoding and decoding techniques.

Covert channels and steganography are often used interchangeably, but it is important to distinguish between these two methods of concealing information. Covert communication focuses on disguising the existence of the communication itself, while steganography involves hiding information within other content to avoid detection. In this thesis, we use “covert communication” to refer to sending information secretly, and “steganography” to describe the technique of embedding information within other data.

2.2.2 Taxonomy of Covert Communication Techniques

One of the earliest taxonomies for covert communication channels was proposed by [35]. This taxonomy divides covert channels into two categories based on their underlying mechanism: storage-based and timing-based covert communication. Storage-based covert communication channels utilize changes in computer storage or processing power to convey hidden information, whereas timing-based covert communication channels exploit variations in the timing of events to transfer information. This categorization provides a clearer understanding of the diverse covert communication techniques and their distinctive characteristics. Below we provide a detailed description of each of these categories:

- **Timing-based covert communication:** It is a method of transmitting secret information through the timing of network packets or other events. For instance, an attacker can take advantage of the time differences between Internet Control Message Protocol (ICMP) packets to send a message to a compromised host. Another form of timing-based covert communication is a covert channel that sends information by altering the timing of keystrokes.
- **Storage-based covert communication:** Covert communication can also be achieved through storage-based methods, which involve utilizing fluctuations in storage or computer capacity to send secret data. For example, an attacker can send a hidden message by varying the amount of disk space consumed or the size of a file. Another example is a covert channel that uses fluctuations in memory allocation or Central Processing Unit (CPU) cycles to transmit information.

Covert communication techniques can also be categorized based on the network layer

they operate [62]. Below are some examples of covert communication techniques classified according to their network layer:

- **Physical Layer:** Covert communication techniques that operate at the physical layer of the network exploit the physical medium of transmission's features, such as sound, light, or electromagnetic waves, to secretly communicate information. Some common techniques for this layer are spread spectrum techniques, modulation techniques, and electromagnetic emanations. In spread spectrum techniques, covert users use a wide range of frequencies to spread the signal over a larger bandwidth, making it harder to detect. Modulation techniques modifies a carrier wave to encode the information to be transmitted. In electromagnetic emanationselectro, magnetic radiation emitted from electronic devices are used to transmit data.
- **Data Link Layer:** At the Data Link layer, covert communication techniques involve the use of network protocols and hardware to create concealed communication channels. One such technique is leveraging the Address Resolution Protocol (ARP) to transmit data between two hosts, making it invisible to network monitors. Another technique involves using the Media Access Control (MAC) layer to transmit data within unused network frames.
- **Network Layer:** Techniques at this layer focus on exploiting features of network protocols to establish hidden communication channels. For example, attackers can leverage the fragmentation and reassembly process of IP packets to hide data in the packet payload. Also, using the Internet Protocol (IP) options or fields in the packet header, covert users can transfer their secret information.

- **Transport Layer:** Covert communication techniques can be employed at the transport layer hide information within the regular flow of network traffic. For instance, covert user can use the User Datagram Protocol (UDP) to transmit secret messages within standard packet payloads. Another technique that they can use is to manipulate the timing and sequence of packets to transmit concealed data using the Transmission Control Protocol (TCP). These techniques can be hard to detect because they do not introduce any extra packets into the network, making it challenging for network security measures to identify them.
- **Application Layer:** Covert communication at the application layer is commonly achieved through steganography techniques, which involve hiding data within seemingly innocuous files or media. For instance, an attacker may conceal data inside an image or audio file and then transfer it over the network. Attackers can also embed hidden messages within HTTP protocols, such as hiding messages within Hypertext Transfer Protocol (HTTP) requests and responses. Another method is Domain Name System (DNS) tunneling, whereby attackers encode and send data by modifying the DNS query and response packets to include the hidden data.

2.2.3 Covert Communication At Physical Layer

Compared to other layers in the network stack, covert communication at the physical layer presents distinct advantages and challenges. The primary reason for its need for further research is that it involves manipulating physical signals, which can be difficult to detect using standard network security approaches. Additionally, physical layer covert communication

techniques can bypass security measures implemented in higher layers, making them an attractive choice for attackers.

The complexity of the physical layer itself is one of the major challenges to comprehending covert communication at this layer. Physical layer techniques often require specialized knowledge of hardware and electrical engineering, making them difficult to study and comprehend for network security researchers who specialize in other areas.

Although the complexity of the physical layer presents a challenge in studying covert communication techniques, there are benefits to understanding these strategies. For instance, it can aid in enhancing network security by creating more effective defense mechanisms against physical layer attacks. Additionally, as attackers may increasingly use physical layer tactics to evade detection, researchers must have a thorough understanding of this area of covert communication to improve network security measures.

2.2.4 Traditional and Deep Learning Based Techniques

Traditional covert communication techniques at the physical layer typically involve modifying physical properties of the communication medium, such as signal amplitude, frequency, phase, and timing. These techniques can be broadly categorized into two categories: spread spectrum techniques and modulation-based techniques. Below, we will delve into each category in more detail.

- **Spread spectrum:** Spread spectrum techniques involve spreading a signal across a wide frequency range, making it harder for a Warden to detect and intercept. One method of achieving this is frequency hopping, where the signal changes its frequency

quickly within a predetermined frequency range. This makes it difficult to intercept the signal. Another method is direct sequence spreading, in which the original signal is combined with a pseudorandom sequence to spread it across a larger bandwidth.

- **Modulation-based:** Modulation-based techniques involve modifying signal parameters such as phase or amplitude to transfer information covertly. For instance, phase shift keying (PSK) changes the phase of the signal to encode data, while amplitude shift keying (ASK) modifies the signal's amplitude to represent data.

In recent years, deep learning techniques have been utilized to enhance the effectiveness and resilience of modulation-based wireless covert communication. In particular, autoencoder algorithms have been employed to learn the correlation between the transmitted signal and the concealed message. By training the autoencoder networks on a dataset of known signals and their corresponding hidden messages, the network can understand how to encode and decode concealed messages in signals that have been modified to resemble legitimate signals. Consequently, such covert communication can be more challenging to detect by wardens or other security measures that attempt to detect abnormalities in the signal.

The use of Generative Adversarial Networks (GANs) has also been advantageous in wireless covert communication techniques, where they can be trained to generate hard-to-detect covert signals. GANs comprise a generator network that produces covert signals, and a discriminator network that recognizes the difference between legitimate and covert signals. After the training process, the generator network can create new covert signals that can be transmitted while the discriminator network can identify and classify them. This technique shows promise in enhancing the durability of covert communication techniques

against detection by wardens.

2.3 Related Works

Since the main idea of our work stems from steganography techniques, we first briefly go over the history and current state of this field of research. We then continue this section by reviewing some of the traditional and deep-learning-based approaches to establishing covert communication at the physical layer of wireless networks.

2.3.1 Image Steganography

Deep learning algorithms have proven their efficiency in many aspects. Steganography is one of these areas that has benefited tremendously from deep learning advancements in recent years. Convolutional neural networks (CNNs), for instance, which are generally used in computer vision tasks, have shown outstanding results in image steganalysis [55, 42, 60], replacing traditional statistical methods. One of the earliest works of image steganography using deep neural networks is by Baluja [3]. In this work, Baluja proposes a hiding scheme in

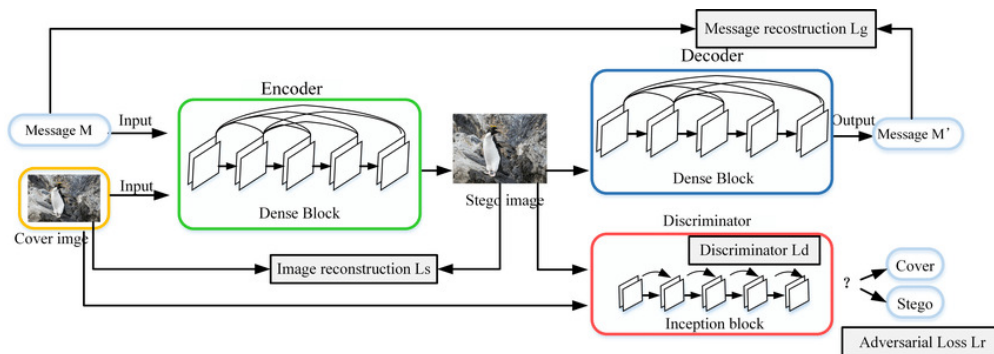


Figure 2.7: The neural network architecture of an example GAN-based Image Steganography [43].

which the three networks of preparation, hiding, and reveal sort out the secret encoding and decoding task. The preparation network transforms the hidden message into features that are commonly used for compressing images. Then, the hiding network embeds it into the cover image and sends it to the reveal network, where the secret message gets extracted from the container image. Following this work, it was discovered that the existence of the preparation network is not necessary, and the framework can be expressed in a simpler form by excluding this network [64]. The disadvantage of these schemes is that the encoding process is reliant on the cover image. To address this, Zhang et al. [63] propose a new architecture in which the secret message can be encoded independently of the cover image. Besides having more flexibility in hiding the information, this approach has also become an effective method for image watermarking. To manifest robustness against steganalysis practices, researchers started to adopt generative adversarial network (GAN) architectures [15]. Volkhonskiy et al. [56] propose one of the first steganography techniques based on GAN networks. The main idea of their work is to use a generative network to produce a new set of cover images that, when carrying the secret message using any of the available steganography techniques, will be less exposed to be detected by a discriminator network (i.e., a steganalysis network). Similarly, Hayes et al. [19] introduce a GAN-based steganography technique with a different objective for the generator network. Instead of generating cover images, the generator learns to embed secret messages into the cover images so that the discriminator cannot distinguish cover images from steganographic images. Although this adversarial scenario was initially introduced for hiding data in images, researchers found it so versatile that it has been applied to other forms of data such as video, audio, and text [33]. This has inspired us to investigate the applicability of such techniques in the wireless communication domain.

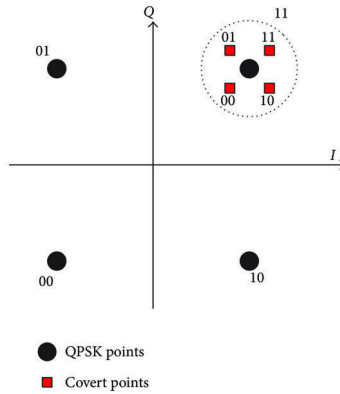


Figure 2.8: The dirty constellation consists of a cover QPSK constellation and a covert constellation. [10].

2.3.2 Traditional Covert Communication Techniques

Covert communication techniques have been applied to various network layers and protocols, including IP [9], MAC [49], and DNS [37]. However, covert communication at the physical layer has only recently gained attention. Dutta et al. [13] are one of the first researchers who developed a covert communication technique for the physical layer of wireless communication systems. They leveraged communication noise, which is caused by either the channel or hardware imperfections, to establish a covert channel. In their method, messages were covertly encoded in the constellation error of normal cover signals. Cao et al. [10] further improved this method to reduce the probability of detection. In order to achieve this, they moved the covert points closer to the ideal cover constellation points and added randomness to the I/Q vectors of the covert constellation by using a Gaussian distribution. Hou et al. [22] proposed an amplitude-based covert channel over LoRa PHY, where covert information is embedded with a modulation scheme orthogonal to chirp spread spectrum (CSS). Bonati et al. [8] introduced SteaLTE, a full-stack wireless steganography method on software-defined cellular networks. To covertly modulate symbols, they employed three different approaches:

dirty QPSK modulation, hierarchical amplitude shift keying (ASK) manipulation, and phase offset of the primary symbols modification. However, the distortions that these methods cause in the statistical properties of the system were later found not to be so difficult to detect using steganalysis methods [24], which in turn compromises the covert channel. This has been one of the main reasons that researchers have turned to machine learning techniques to develop more sophisticated covert methods.

2.3.3 Deep-Learning-Based Covert Communication Techniques

More recent works have explored the viability of deep neural networks in the covert communication problem. We can categorize these techniques into two types based on the neural network architecture they use.

2.3.3.1 CNN-Based Techniques

Sankhe et al. [47] propose a method called Impairment Shift Keying that produces subtle variations in normal signals in a controlled way such that a CNN model can be trained to classify them as zeros or ones. Although the impact of their covert method on the system's communication error rate is as small as 1%, authors in [23] showed that even tiny modifications to the signals' constellation points can be detected using a CNN model trained on the amplitude and phase characteristics of the error vector magnitude (EVM) and constellation points. Besides, their scheme relies on existing hardware impairments in the system, which is not the case in many deployments.

Hegarty et al. [21] introduce a short-range covert communication through Radio Frequency (RF) emanations. In their method, hidden data is encoded in the RF emanations generated

by a microcontroller due to different programs running on it. The microcontroller generates frequency shift keying (FSK) modulated signals by changing the center frequency of a specific part of the spectrum based on the program running. By isolating that part of the spectrum, a signal that acts like an FSK is obtained. These signals are then received and demodulated by a software-defined radio (SDR) at a range of up to 4 feet using a CNN model. The limitation of this approach is that the covert receiver needs to be in close proximity to the covert sender.

2.3.3.2 GAN-Based Techniques

To find an optimal solution for the highest covert rate and minimum probability of detection, Liao et al. [31] employed a GAN model that can adaptively adjust the signal power at a relay station for establishing covert communication. This requires the covert users to have access to a cooperative relay node, which is not applicable in many communication scenarios. Another example of adversarial training for covert communication can be found in [25]. Their setup contains a transmitter communicating with a receiver through a Reconfigurable Intelligent Surface (RIS), and their goal is to keep this communication covert from a prospective eavesdropper. Both the intended receiver and the eavesdropper use CNN classifiers to detect the signals. This scenario raises the same concern as the previous work, which is the necessity of the existence of a relay node in the deployment. Moreover, perturbations added to the signals to deceive the eavesdropper are crafted using the fast gradient method (FGM) [16], which in [2] was shown to be easy to counter using existing countermeasure techniques. Our work differs from these two works in two key aspects. First, our proposed method does not rely on any external entities or external factors, such as relay nodes or hardware impairments,

which helps with the generality of our model. Second, we have specifically designed our covert model to have as little impact on normal communication as possible, ensuring that the error rate of normal communication does not suspiciously rise. This objective is critical and often overlooked in previous works, which could easily expose any covert communication to the system’s observer.

One of the most related works to ours is the covert scheme proposed by Mohammed et al. [36]. They formulate covert communication as a three-player game in which networks compete in an adversarial setting to obtain the optimal solution. In their setup, the encoder and decoder networks learn to covertly communicate through a form of noise while simultaneously trying to confuse a detector network that tries to determine whether the users are covertly communicating or if it is just normal transmissions going through. While our proposed method shares some ideas with this work, there are several critical limitations in this work that ours aims to address. First, our model does not rely on any hardware impairment noise and instead embeds the covert signals into the existing channel’s noise. Second, in the previous work, the impact of added covert signals on the normal communication is unknown, but our model is optimized to preserve the performance of normal communication. Finally, the previous work assumes the channel between users to be AWGN, which is not an accurate model for simulating channel effects in wireless communications due to fading. Our work addresses this by proposing a scheme that is robust against fading channel models, including Rayleigh and Rician fading channels.

2.4 Implementation and Libraries

2.4.1 PyTorch

PyTorch [41] is a widely used open-source deep learning framework that is implemented in *Python* programming language. It provides a flexible and efficient platform for building, training, and deploying neural network models. PyTorch leverages the computational power of Graphics Processing Units (GPUs) to accelerate training and inference processes, making it suitable for large-scale and complex deep learning tasks. In our thesis, we utilized PyTorch as the framework of choice for implementing our autoencoder wireless systems and GAN-based covert models.

2.4.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a popular dimensionality reduction approach that is implemented in Python as a library called *scikit-learn (sklearn)*. The goal of PCA is to convert high-dimensional data into a lower-dimensional representation while keeping the most relevant patterns and variations in the data. This is accomplished by determining the principal components, which are linear combinations of the original features that capture the most variance in the data.

PCA is primarily utilized for preparing data and exploratory data analysis. It is beneficial for reducing the dimensionality of a dataset, which offers several advantages. These include decreasing computational complexity, facilitating visualization of data in lower-dimensional spaces, and eliminating noise or unnecessary features. PCA can simplify data analysis, increase model performance, and provide insights into the underlying structure of the data

by lowering the number of dimensions.

In this thesis, we utilized PCA to reduce the dimensionality of our constellation diagrams. This approach enabled us to visualize high-dimensional signal points on a two-dimensional diagram, providing a more accessible representation of the data. Fig. 3.11, for example, demonstrates the use of PCA for visualization purposes.

2.4.3 Matplotlib

All the diagrams presented in the evaluation chapter of this thesis were created using *Matplotlib*. Matplotlib is a popular Python toolkit for creating high-quality visualisations and plots. It has a wide range of tools and functions for creating different types of charts, graphs, and plots, such as line plots, scatter plots, bar plots, histograms, and more. Matplotlib offers plenty of customization options, allowing users to fine-tune the appearance and style of their plots to meet their specific requirements. It is widely used in data analysis, scientific research, and data visualisation tasks.

2.4.4 Training Hardware Setup

Our GAN-based covert model and autoencoder wireless models were trained on a high-performance desktop computer with an RTX 3080Ti graphics card, which provided superior GPU acceleration for computations, particularly for models that contain convolutional layers. The desktop configuration included an Intel Core i7 10700KF 3.80GHz CPU, 16GB of DDR4 3200MHz RAM, and 12GB of GPU memory.

Chapter 3

Autoencoder Wireless Systems

In this chapter, we provide the necessary information about the autoencoder wireless systems that are utilized in our covert communication model. We begin with a comprehensive background on autoencoder-based wireless systems, including their concept and their significance in wireless communication. Then, we delve into the internal neural network architecture employed by these systems and elaborate on the training process. Finally, we present the evaluation results and provide detailed discussions regarding their implications.

3.1 An Overview of Autoencoder Wireless Systems

Traditional wireless communication systems are designed to reliably transmit data through channels that may impair the signals being sent. These systems consist of several components, including channel coding, modulation, equalization, and synchronization, which are individually optimized using mathematical models that simplify the problem and provide closed-form expressions.

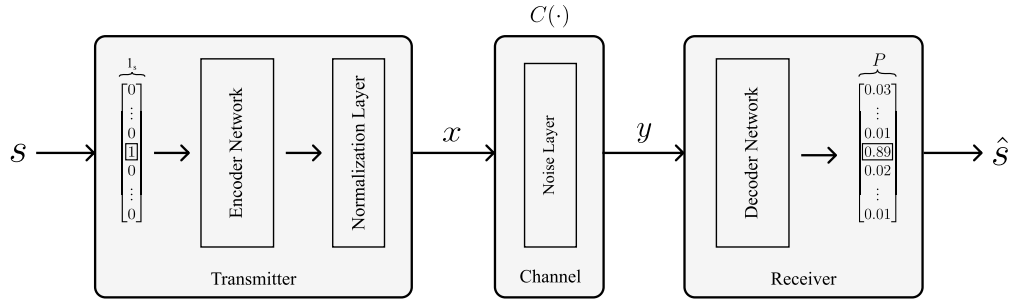


Figure 3.1: An end-to-end autoencoder-based communication system. The system receives a message s as input and generates a probability distribution over all possible messages. The most probable message is then selected as the output \hat{s} [39].

Autoencoders wireless systems, on the other hand, are a relatively new development in wireless communications and have numerous advantages over traditional systems. Their ability to learn from data and adapt to changing conditions in the wireless environment makes them ideal for dynamic wireless environments. They can also be trained to handle noise and errors in the transmitted data, not only for linear, but also for nonlinear channel effects. Autoencoder-based wireless communication systems were introduced as an end-to-end learning paradigm that abstracts the coding and modulation components of traditional modular communication systems by replacing the transmitter and receiver with DNNs.

3.1.1 Internal Neural Network Architecture

The internal neural network architecture of a wireless system that uses an autoencoder can be seen in Fig. 3.1. It consists of an encoder and a decoder network, which work together to compress and reconstruct the input data.

- **Encoder:** The encoder network, also known as the transmitter, learns the underlying statistical properties of the wireless channel and modulates the input data to ensure reliable transmission over the channel. The encoder network compresses the input data

into a lower-dimensional representation, which is then transmitted over the wireless channel.

- **Channel:** Depending on the wireless channel being modelled, the channel effect is simulated by introducing noise or distortion to the signal, such as Gaussian noise, fading, interference, or non-linear distortions. Gaussian noise is one of the most common ones and can be controlled by adjusting its variance.
- **Decoder:** The receiver, or decoder network, receives the degraded signal and attempts to reconstruct the original message. To obtain the encoded representation, the decoder network first demodulates the received signal. This encoded representation is then passed into the decoder network, which learns to transform it back into the original data.

3.1.2 Training Procedure and Evaluation Metrics

Training: To train the autoencoder network, a set of training data is prepared to optimize the weights and biases of the encoder and decoder networks in order to train the autoencoder network. This data is usually randomly created depending on the parameters of the autoencoder model, which are data and coding rates. Since neural networks are designed for continuous data, the encoder uses an embedding or one-hot encoding to convert the categorical/discrete data into continuous vectors. As we also mentioned in Section 2.1.1.2, the objective of the encoder network is to learn to compress the input data into a lower-dimensional representation, while the decoder network learns to recover the input data from the encoded form. This process is regulated by a loss function that quantifies the reconstruction error and tries to

minimize it. Algorithm 1 summarizes this process:

Algorithm 1 Training Autoencoder Wireless System

Time Complexity: $O(n)$, where n is the number of epochs

$X \leftarrow$ training dataset (input signals)

$\Theta_F, \Theta_G \leftarrow$ encoder and decoder neural networks

$T \leftarrow$ embedding or one-hot encoding transformation

$C \leftarrow$ channel noise and distortion model

$L \leftarrow$ network loss

$H \leftarrow$ cross entropy function

for epoch $ep \in \{1 \dots n_{epochs}\}$ **do**

$X_t = T(X) \leftarrow$ embedded or one-hot encoded data

$Y = F(X_t) \leftarrow$ encoder output

$\hat{Y} = C(Y) \leftarrow$ distorted signal

$L = H(G(\hat{Y}), X_t)$

 Update Θ_F, Θ_G parameters to minimize L

end for

Evaluation: During testing, the network's performance is evaluated by checking the accuracy of the reconstructed data and other metrics like bit error rate (BER) or signal-to-noise ratio (SNR). Researchers can analyze the network's performance under different channel conditions to gain insights into how the network performs in real-world wireless situations. This analysis can help identify opportunities to enhance the network architecture or training process.

3.2 Single/Multi-User Autoencoder Wireless Systems

Autoencoder wireless systems can be trained for both single-user and multi-user scenarios. Many concepts that we have discussed thus far have been based on single-user systems. However, it is relatively straightforward to extend these concepts to multi-user systems. Single-user systems are generally simpler to design and analyze due to the absence of

interference. On the other hand, multi-user systems come with an extra layer of complexity, which is the detrimental impact of interfering signals on the communication. We describe in greater detail how each of these systems works.

Single-User Autonencoder Systems: In this type of communication system, the encoder transforms k bits of data into a message s where $s \in \{1, \dots, M\}$ and $M = 2^k$. The encoder then takes this transformed message as an input and generates a signal $X = E(s) \in \mathbb{R}^{2n}$, which is a real-valued vector. This $2 \times n$ -dimensional real-valued vector can be treated as an n -dimensional complex vector, where n is the number of channel uses required for signal transmission. The term “channel use” refers to the utilization of communication resources, such as time, frequency, or codes, within the communication channel for transmitting or receiving data. To account for channel noise, usually additive white Gaussian noise (AWGN), the noise effect is added to the signal vector, denoted as N , which is a n -dimensional independent and identically distributed (i.i.d.) vector with each element coming from a complex normal distribution with 0 mean and σ^2 variance $N_i \sim \mathcal{CN}(0, \sigma^2)$. A fading coefficient H is introduced to account for the varying channel conditions. If there is no fading, H is equal to the identity matrix I_n . However, in the presence of fading, H is a diagonal matrix with each element following a fading distribution, such as Rayleigh or Rician. Ultimately, the received signal at the receiver, carrying the channel’s noise, can be expressed as $Y = H \cdot X + N$. Once the signals pass through the channel, the decoder receives these distorted signals and applies the transformation $D : \mathbb{R}^{2n} \rightarrow M$ to output the reconstructed version of the message s , which is denoted as $\hat{s} = D(Y)$.

Multi-User Autonencoder Systems: A multi-user autoencoder communication system is an extended version of a single-user system with either multiple transmitters and receivers

or multiple transmitters and a central receiver. In this work, we consider the latter case. In this system, each encoder sends a separate message s_i where $s_i \in s$, and i is the index of the transmitter. Each transmitter generates a modulated signal accordingly: $X_i = E_i(s_i) \in \mathbb{R}^{2n}$. We consider a multiple-access system, where all transmitters send their signals at the same time. Consequently, the signals experience channel interference in addition to the channel effects. When passing through the channel, signals are simultaneously faded and interfere with each other. The resulting vector of signals for each transmitter can be expressed as $Y_i = \sum_{i=1}^{n_{tx}} H_i \cdot X_i \cdot e^{j\theta_i} + N_i$, where H_i is the channel coefficient and $e^{j\theta_i}$ is the phase offset for the i^{th} transmitter, X_i is the corresponding encoded signal, and n_{tx} is the number of transmitters. Finally, the signals are received at the decoder, where it uses its decoding function $D(\cdot)$ along with the channel matrix H with the size of $n_{tx} \times n_{rx}$, where n_{rx} is either the number of receivers in the multiple receiver case or the number of antennas at the receiver in the central receiver case, to reconstruct the message $\hat{s} = D(Y, H)$.

3.3 System Models

We will start by describing our system model in the single-user case, which simplifies the system by eliminating the complexity of interference between entities. This scenario applies to systems that already handle user interference at higher levels or by using common multiple access techniques, such as Orthogonal Frequency Division Multiple Access (OFDMA), Code Division Multiple Access (CDMA), and Time Division Multiple Access (TDMA) [57]. Fig. 3.2 gives an overview of this case. We then continue with a more complex scenario in the subsequent subsection, the multi-user case, where we have interfering signal transmissions.

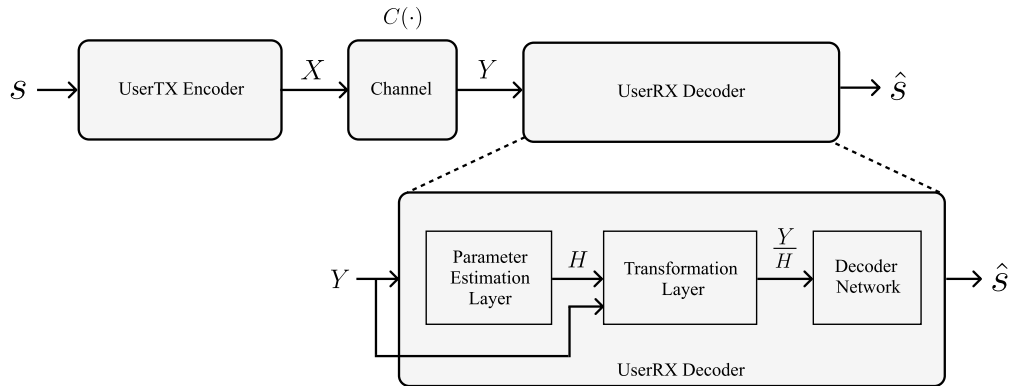


Figure 3.2: An overview of the single-user communication model with a detailed view of the UserRX decoder network.

Fig. 3.3 illustrates the multi-user model.

3.3.1 Single-User Communication Model

Transmitter: In the single-user case, the encoder or the transmitter is referred to as UserTX, and the decoder or the receiver is referred to as UserRX. These two entities together form our autoencoder-based wireless communication system. The communication between these two entities begins with UserTX encoding a binary message to a vector of signals using its encoder network. This vector of signals is then transmitted to UserRX and gets distorted while passing through the channel.

Channel: We consider three channel models of AWGN, Rayleigh fading, and Rician fading. To set the *Signal to Noise Ratio* (SNR) in the AWGN model, we keep the transmitter’s average power at unit power and adjust the noise power accordingly. For the fading channels, we consider a flat-frequency block-fading channel model, where each signal vector (codeword) is assumed to be faded independently.

Receiver: A noisy version of the transmitted signal is received at the receiver side,

where UserRX extracts the message by decoding the signals. In the case of fading channels, receiver equalizes the signals before passing them to the decoder network.

Equalization: The channel matrix is estimated by UserRX using a blind channel estimation technique, by feeding the received signals to a preliminary network to predict the fading coefficients. Using the estimated channel matrix, UserRX equalizes the signals prior to feeding them to the decoder network. In Fig. 3.2 under the expanded view of BaseRX decoder, you can see the two layers of “Parameter Estimation Layer” and “Transformation Layer” working together to equalize the signals before getting passed to the decoder network.

3.3.2 Multi-User Communication Model

Transmitters: In the multi-user case, multiple transmitters (UserTXs) communicate with a single base station (BaseRX), which serves as the central receiver. Each UserTX uses its own encoder network to encode a binary message to a vector of signals. The transmitters have no knowledge of each other’s messages and share no parameters in their networks. Each UserTX and BaseRX pair forms an autoencoder model, with UserTX serving as the encoder and BaseRX as the decoder. After encoding their messages into signals, the transmitters simultaneously send their signals over the channel.

Channel: The multi-user case has the same channel models as described in the single-user case, with the exception that signals experience interference due to simultaneous transmission. The interference effect has been incorporated in the channel function.

Receiver: BaseRX receives the signals from all the transmitters using multiple antennas after they pass through the channel. It decodes the messages in the same way as the UserRX

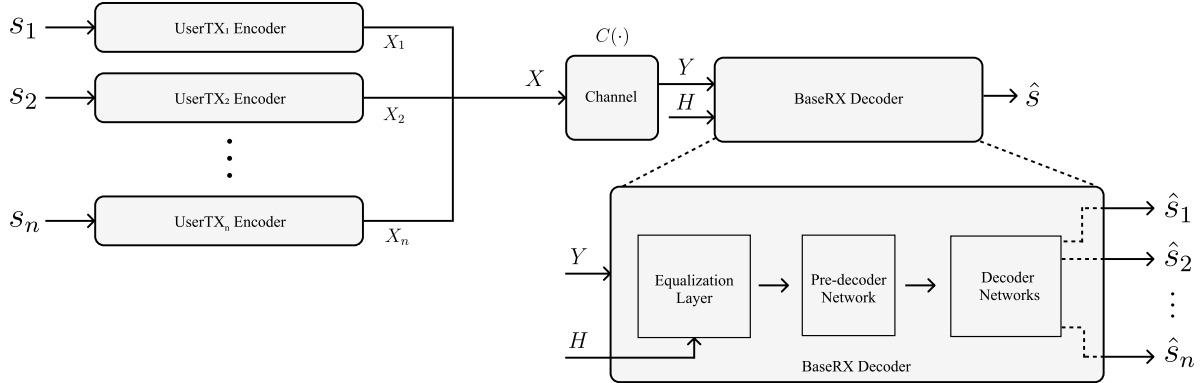


Figure 3.3: An overview of the multi-user communication model with a detailed view of the BaseRX decoder network.

in the single-user case, but it handles the decoding process for the transmitters' signals all at once.

Equalization: Similar to the single-user case, BaseRX equalizes the signals before passing them to the decoder networks. However, unlike the single-user receiver, we assume that BaseRX has access to the channel matrix. In practice, BaseRX can use a pilot-based channel estimation technique, which results in a much more accurate estimation of the channel matrix compared to the blind channel estimation used in the single-user case.

3.4 Neural Network Design

In this section, we break down each component of our autoencoder neural network into individual subsection and discuss their architecture and operation in detail.

3.4.1 Embedding Layer

As inputs to our transmitter are categorical data, we cannot directly pass it to the encoder's neural network. Therefore, transmitter uses a one-hot embedding layer to convert binary

messages s of size k into continuous vectors, which will be understandable by the encoder network.

3.4.2 Encoder Network

The encoder network maps these one-hot encoded messages to vector of signals, which will be of size $2 \times n$, where n is the number of channel uses. In multi-user case, as there are multiple encoders involved, all vector of signals put together will have a size of $n_{tx} \times (2 \times n)$.

The encoder network comprises several Dense layers at the initial stages, followed by multiple 1-Dimensional Convolutional (1D Conv) layers. The purpose of the Dense layers is to expand the dimensionality of the input data, enabling the network to capture more complex patterns and representations. In fact, this expansion of capacity empowers the model to learn complex relationships within the data. Subsequently, the 1D Conv layers assist the encoder to learn more complex coding schemes that typically require convolutional operations.

The output data from the encoder network X is then passed to the channel model layer to apply the channel effects to the signals.

3.4.3 Channel Model Layer

In order to have a differentiable loss function, channel models are mathematically formulated as a layer in the middle of the network. This is important because differentiability allows for the use of gradient-based optimization algorithms, such as backpropagation, to train the neural network effectively. Depending on the channel model, various channel functions $C(\cdot)$

are utilized to apply the desired channel effect to the signals at this layer. For example, in an AWGN channel, the channel model layer adds a noise vector, sampled from a Gaussian distribution with a specific variance based on the set SNR value, to the signals. For Rayleigh and Rician channels, signals are multiplied by a complex Gaussian random variable and then added with Gaussian noise.

3.4.4 Equalization Layers

We use two types of equalization layers in our models based on whether the system is single-user or multi-user. When the system is single-user and there is fading in the channel model, the receiver side utilizes the two layers of “Parameter Estimation” and “Transformation” to equalize the signals. We apply a basic transformation function that simply divides signals by the channel fading coefficients estimated via the parameter estimation layer. Note that more complex transformation functions can be used, as described in [39]; however, optimizing the performance of the autoencoder model is beyond the scope of this thesis and can be considered in future research. In the multi-user case, the decoder is provided with channel coefficients as input. This enables BaseRX to use the zero-forcing technique [14] to equalize the received signals.

3.4.5 Decoder Network

The decoder network receives the distorted signals Y passed from the *Channel Model Layer* or the equalized signals from the *Equalization Layer* and reconstructs the transmitted message \hat{s} . Similar to the encoder network, decoder network also comprises of multiple Dense initially,

followed by multiple 1D Conv layers. The purpose of the Dense layers is to increase the decoder network learning capacity and 1D Conv layers are to learn the convolutional coding that transmitter has developed to encode the messages.

In the case of a single-user system, UserRX utilizes its decoder network to reconstruct the message. However, in the multi-user case, BaseRX simultaneously decodes signals from all transmitters. This is achieved by first passing the signals to a pre-decoder network and then using separate decoders at the final layers.

Table 3.1: Autoencoder’s detailed network architecture in the single-user and multi-user case.

	UserTX Encoder	UserRX Parameter Estimation	UserRX Decoder	BaseRX Pre-Decoder	BaseRX Decoders
input size	16	2×8	2×8	$n_{tx} \times 2 \times 8$	$n_{tx} \times 4 \times 8$
dense layers sizes	$2 \times 8, 2 \times 8$	$2 \times 16, 2 \times 32, 2 \times 4$	$2 \times 8, 2 \times 8$	$n_{tx} \times 2 \times 8, n_{tx} \times 4 \times 8$	$n_{tx} \times 2 \times 8$
dense layers activations	$2 \times \text{ELU}$	$\text{ELU}, 2 \times \text{Tanh}$	$2 \times \text{Tanh}, \text{Softmax}$	$3 \times \text{Tanh}$	$\text{Tanh}, \text{Softmax}$
conv filters	1, 8, 8, 8	-	1, 8, 8, 8	1, 8, 8, 8	-
conv kernel sizes	2, 4, 2, 2	-	2, 4, 2, 2	2, 4, 2, 2	-
conv strides	1, 2, 1, 1	-	1, 2, 1, 1	1, 2, 1, 1	-
conv activations	$4 \times \text{Tanh}$	-	$4 \times \text{Tanh}$	$4 \times \text{Tanh}$	-
output size	2×8	2×1	16	$n_{tx} \times 4 \times 8$	16

3.5 Evaluation

In the subsequent subsections, we outline the configuration details used for evaluation and present the evaluation results for the performance of our trained autoencoder wireless networks. It is important to note that these experiments do not involve covert communication.

The primary objective of these experiments is to fine-tune the parameters of the autoencoder models and verify their accurate performance and behavior in fading environments, in addition to the AWGN channel that is considered in the literature. We measure the models’ performances using BLER. We calculated this by dividing the number of received symbols with errors by the total number of transmitted symbols, expressed as a percentage.

3.5.1 Single-User Autoencoder’s Performance

3.5.1.1 Methodology

We implemented an autoencoder communication network for normal communication between UserRX and UserTX. An *Autoencoder*(n, k) is a neural network communication model that sends k bits of data in n channel uses.

Data Rate and Channel Uses: To make our results comparable with [39], we chose our default parameters to be 8 and 4 for the number of channel uses and binary message size, respectively. However, we also evaluated our models for two other sets of parameters with the same data rate but different numbers of channel uses. This allowed us to examine how increasing the number of channel uses, or signal dimensionality, would affect communication performance.

Dataset Size: To train our autoencoder model, we generated two datasets for training and testing by randomly generating binary messages s of size k . This gives us 2^k unique messages, which are uniformly distributed. More specifically, we used 8192 random binary messages in the training set and 51200 random binary messages in the test set. We created a much larger dataset for testing to ensure that each signal X undergoes various channel distortions, providing a more accurate evaluation of the model’s performance.

Optimization Parameters: We set the learning rate to 0.001 and optimized the model using the Adam optimizer [26]. We used a batch size of 1024 and trained the model for 100 epochs.

Channel Configuration: For the channel configuration, we fixed the SNR value during training but evaluated the model’s performance over a range of SNRs.

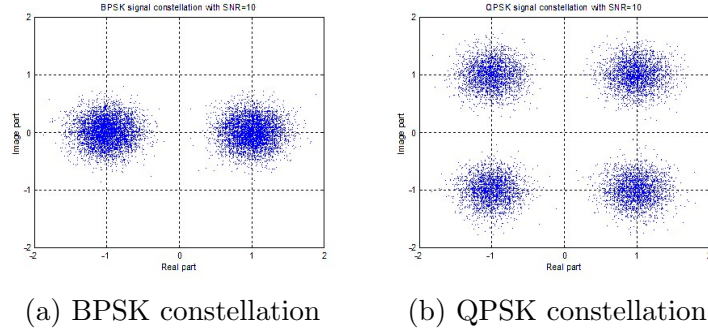


Figure 3.4: Theoretical BPSK and QPSK constellation diagrams.

The SNR value for the AWGN channel was set to 8dB, while the values for the Rayleigh and Rician fading channels were 16dB. We chose these SNR values experimentally by training the models on different SNR values and identifying the value on which the model performed best.

We have used these parameters in all experiments below, unless stated otherwise.

3.5.1.2 Results

Constellation Diagrams: A constellation diagram is a graphical representation of complex symbols that provides a visual depiction of the signal space and the mapping of symbols onto this space. In this diagram, each point corresponds to a specific symbol or combination of bits transmitted through the communication channel. Typically plotted in the complex plane, the x-axis represents the real (in-phase) component of the symbol, while the y-axis represents the imaginary (quadrature) component. Plotting constellation diagrams can help us understand how autoencoder wireless networks work by revealing insights into the learned signal representations and their transmission characteristics.

For this particular experiment, two autoencoder networks were trained: one transmitting one bit and the other transmitting two bits of data per channel use. These parameter

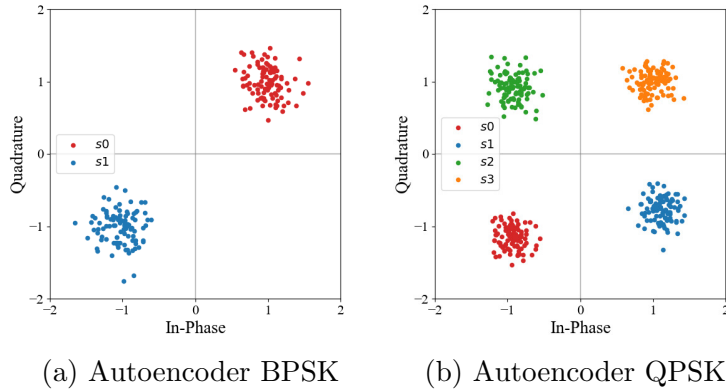


Figure 3.5: Autoencoder BPSK and QPSK equivalent constellation diagrams.

selections allow for a comparison to be made between the first network and a theoretical Binary Phase Shift Keying (BPSK) system, while the second network is comparable to a Quadrature Phase Shift Keying (QPSK) system. In order to have a better visualization of the learned constellations, the models were trained and evaluated on an AWGN channel, which minimally affects the positioning of the points. The SNR was set to 10dB for both the autoencoder and theoretical systems.

Figs. 3.4 and 3.5 present a comparison of constellation diagrams between the theoretical and autoencoder models. The diagrams demonstrate that autoencoder wireless networks exhibit similar characteristics to their theoretical counterparts. In the case of QPSK, both systems generate identical constellation patterns. For the Autoencoder BPSK equivalent, model has learned a phase shifted constellation version of the theoretical system. Despite this shift, however, the performance remains the same as the points are still positioned 180° apart. This learned orthogonalization allows for reliable demodulation and decoding of the transmitted symbols even in the presence of noise or channel impairments.

Training SNR Value Impact: During the training of an autoencoder wireless communication

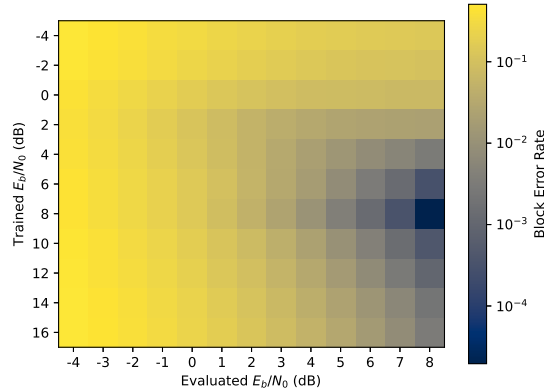


Figure 3.6: Heatmap displaying the autoencoder model’s performance in terms of BLER, with E_b/N_0 values used for training on one axis and E_b/N_0 values used for evaluation on the other axis.

model, the SNR value used significantly impacts its performance. As discussed in Section 3.5.1.1, we conducted experiments to determine the optimal SNR value for training the model. Various SNR values were tested, and the overall performance of the model was measured. To visualize this process, we generated a heatmap (Figure 3.6) that depicts the model’s performance on the AWGN channel.

The x-axis of the plot represents the evaluated E_b/N_0 values, while the y-axis represents the E_b/N_0 values used for training the model. The heatmap reveals that as the SNR increases, the model’s performance steadily improves within the evaluated range. However, beyond a certain threshold, the model’s performance starts to deteriorate again. In the provided example, the optimal SNR value for training the model is determined to be 8dB.

Equalization Impact: Autoencoder wireless models aim to learn and generalize from the training data to effectively handle unseen communication scenarios. In that regard, one of the interesting research directions is to investigate the impact of equalization on the performance of autoencoder wireless models in fading channels. By studying this area, we

can gain insights into the robustness, adaptability of these models, and identify potential approaches for improving their performances.

To this end, we experimented our autoencoder models with three different equalization techniques:

- a) Model without equalization: This approach involves demodulating the signals without applying any equalization. The model architecture, is similar to what we have illustrated in Fig. 3.2, excluding the "Parameter Estimation" and "Transformation" layers, with the data directly passing to the decoder network.
- b) Model with blind equalization: In this method, the model estimates the fading coefficients from the signals and performs equalization using a simple division transformation function. This architecture, shown in Fig. 3.2, was also the default choice in our other experiments.
- c) Model with zero-forcing equalization: Similar to the blind equalization model, this approach uses the same architecture as Option b. However, it omitted the "Parameter Estimation" layer and instead received the fading coefficients as input.

These experiments employed our default parameters (sending 4 bits of data over 8 channel uses) and utilized the Rayleigh fading channel model.

Fig. 3.7 compares the performance of our trained autoencoder using the discussed equalization techniques. One interesting observation from this plot is the close performance similarity between the model without equalization and the other two models that incorporate equalization techniques. This confirms that indeed autoencoder models are well capable to

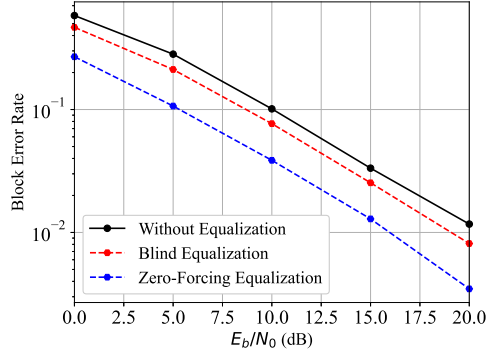


Figure 3.7: Comparing the BLER performance of autoencoder models in the single-user case with three different equalizations: a) without equalization b) blind equalization c) zero-forcing equalization.

effectively adapt to various channel conditions. Additionally, as anticipated, we observe an improvement in the model’s performance when using more complex equalization techniques.

Overall Performance: Fig. 3.8 shows the block error rate (BLER) performance of our trained autoencoder communication models for various sets of parameters across a range of SNR values. The models were trained individually on AWGN, Rayleigh, and Rician fading channels and tested on the same channel they were trained on. The plot reveals that despite having the same data and coding rate, increasing the signal dimension slightly enhances the performance of the autoencoder models. This phenomenon was first identified in [39], which demonstrated that autoencoders trained over an AWGN channel can achieve a coding gain by learning a joint coding and modulation scheme. Our results support this finding and suggest that this behavior holds true for autoencoders trained on other channel models as well. However, it should be noted that a comprehensive study of the performance of autoencoder wireless systems for multiple channel types and parameters (n, k) goes beyond the scope of this work and is not the primary focus of this research.

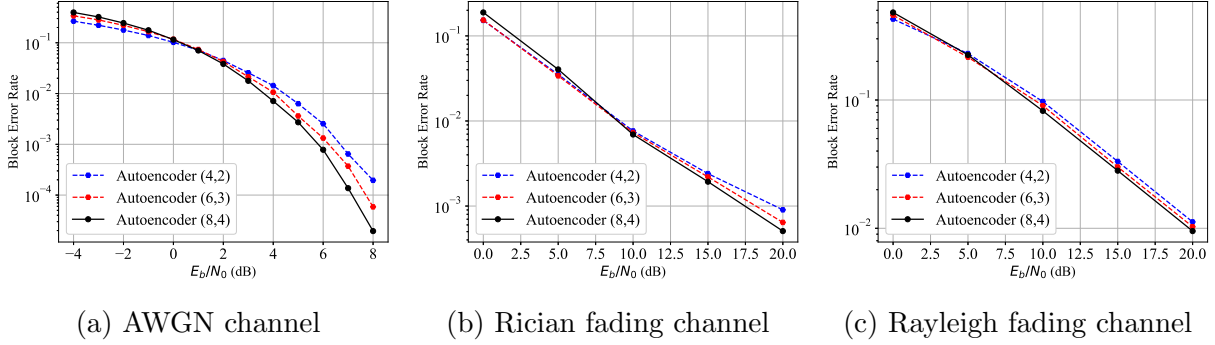


Figure 3.8: Autoencoders' performance in terms of BLER over a range of SNR values is evaluated in our single-user case. The models are trained over AWGN, Rayleigh, and Rician fading channels for a set of parameters that have the same data rate.

3.5.2 Multi-User Autoencoders' Performance

3.5.2.1 Methodology

Data Rate and Channel Uses: In the multi-user case, we have chosen the number of channel uses and the binary message size to be 8 and 4, respectively, as these are our default parameters. There are two reasons for selecting these parameters in this way. First, it allows us to compare our results with those obtained in [39]. Second, by having each user communicate at half the rate of BPSK, the results of our 2-user system are roughly comparable to a single-user system at the BPSK rate, while the 4-user system is comparable to a single-user system at the QPSK rate.

Dataset Size: To generate training and testing sets, we followed the same procedure outlined in the single-user section and generated a dataset for every user in the system by repeating this process.

Optimization Parameters: Parameters such as learning rate, number of epochs, batch size, and optimization algorithm, are kept the same as in the single-user system.

Channel Configuration: For the channel configuration, we have chosen SNR values of

8dB for the AWGN channel, 16dB for the Rayleigh channel, and 14dB for the Rician channel during training. However, we evaluate our models over a range of SNR values.

3.5.2.2 Results

Orthogonalization In Section 3.3, we discussed that common wireless systems employ techniques such as TDMA, CDMA, or OFDMA to mitigate interference in multi-user communications. However, these techniques are designed at levels higher than the autoencoder model operates. Thus, the autoencoder model has to learn how to mitigate interference at the modulation level and the impact of this interference avoidance can be observed in constellation diagrams. An effective interference avoidance techniques would manifest itself on the constellation diagram by forming distinct clusters or sets of symbols that correspond to each user’s signals, with appropriate separation and arrangement. In these experiments, we aimed to gain insight into how autoencoder models handle interference by analyzing their constellation diagrams.

To begin our experiments, we initially set up a simplified configuration to gain an overall understanding of the system’s behaviour. At first, we utilized two autoencoders with a low signal dimensionality. Specifically, we employed two QPSK equivalent autoencoder models, which each transmits 2 bits of data over a single channel use. This low signal dimensionality would enable us to achieve a clear visualization of how signals are distributed across the signal space when we plot their constellation diagrams. Moreover, we paired the messages of the autoencoders so that the same pair was transmitted during transmission. This eliminates the need for orthogonalization across all messages for the autoencoders, reducing the communication’s complexity.

In Fig. 3.9, the constellation diagrams of the two trained autoencoder models are

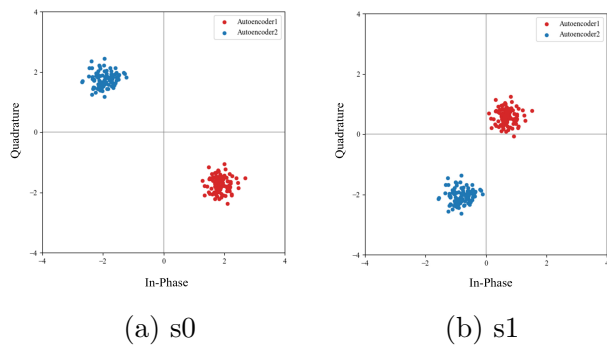


Figure 3.9: Constellation diagrams of QPSK equivalent autoencoders for two messages: a) s1 and b) s2. Autoencoder1 and Autoencoder2 represent separate transmitters that are simultaneously transmitting their signals over the channel. The red and blue circles denote the constellation points of Autoencoder1 and Autoencoder2, respectively.

presented. The diagrams depict the constellation points for two randomly selected example messages, s1 and s2. Notably, both models exhibit a BPSK-like constellation pattern, indicating that they have successfully minimized overlap and correlation between the constellation points. This orthogonality of the constellations aids in mitigating interference between users, as each signal occupies a distinct region in the signal space. The observed spatial separation within the constellation diagram is instrumental in reducing the likelihood of symbol collisions and enhancing the receiver’s ability to accurately separate and recover individual signals from multiple users.

The previous experiment provided a basic understanding of how orthogonalization works in a simple multi-user setup. However, it is crucial to determine if the same behavior persists as we increase the signal dimensionality (i.e., the number of channel uses for transmitting the message). To explore this, we conducted a similar experiment as above but with an increased number of channel uses. In this case, we trained two autoencoder models that each transmits 4 bits of data over 8 channel uses. Also, we did not pair messages of autoencoder this time to observe how the models learn to orthogonalize their transmissions across all possible

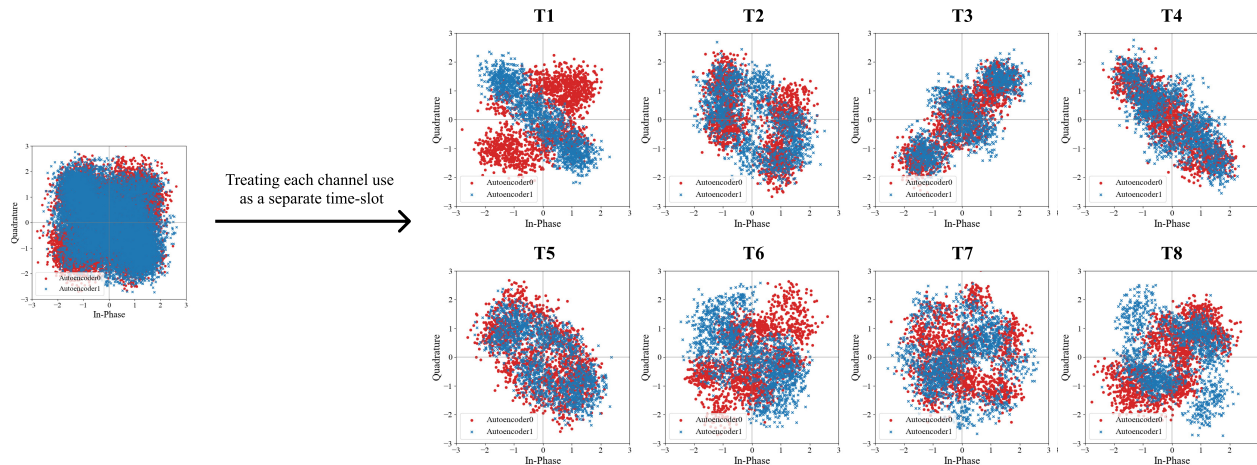


Figure 3.10: Constellation diagrams of the autoencoder models for all messages. On the left side, constellation points of all four users projected onto a 2D plane is shown. On the right side, each channel use is represented as a distinct time slot.

combinations of messages.

In Fig. 3.10, we present the constellation diagrams of these models. On the left side of the figure, you can observe the constellation points of both users projected onto a 2D plane. On the right side, we have decomposed the constellations as if each channel use represented a different time slot. Therefore, with 8 channel uses, we obtained 8 constellation diagrams, each corresponding to a different time slot. This helps us determine whether autoencoder models would learn to avoid interference by achieving orthogonal constellations for every channel use. However, upon examining the figure, it is evident that in the majority of time slots, there is significant signal overlap. This contradicts the aforementioned assumption and suggests that the autoencoder models do not exhibit a per-channel orthogonalization.

Therefore, we sought to investigate whether the orthogonalization occur at higher dimensions. Since our signals have a 16D dimensionality, it is not possible to plot them on a 2D diagram without losing important characteristics, such as spatial distances. To address this, we employed *Principal Component Analysis (PCA)* to reduce the dimensionality while

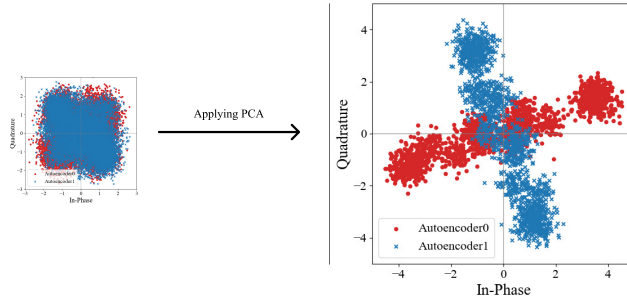


Figure 3.11: Constellation diagrams of the autoencoder models after applying PCA to reduce dimensionality.

preserving the distances between points at higher dimensions (Fig. 3.11). Upon plotting the constellation diagram, we made an interesting discovery: the constellation points were distributed along two orthogonal major axes, confirming that in autoencoders with high signal dimensionality, orthogonalization occurs at higher dimensions.

Equalization Layer Impact: Similar to the single-user scenario, we conducted experiments to assess the impact of equalization on the performance of autoencoder models when there are multiple-users in the system. For this purpose, we tested three different equalization techniques: a) Without equalization, b) blind equalization, and c) zero-forcing equalization.

Fig. 3.12 depicts the performance comparison of our trained autoencoders using these

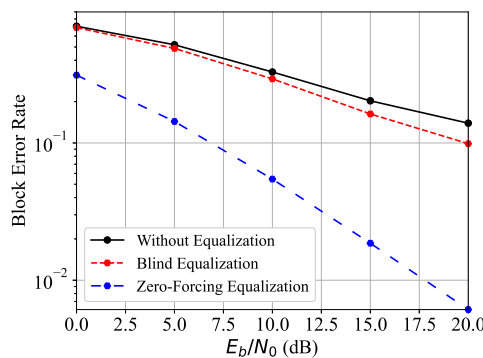


Figure 3.12: Comparing the BLER performance of autoencoder models in the multi-user case with three different equalizations: a) without equalization b) blind equalization c) zero-forcing equalization.

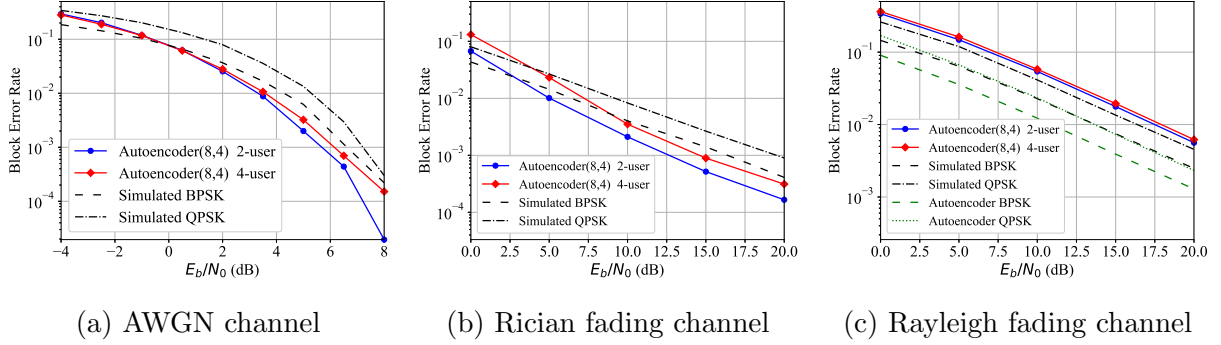


Figure 3.13: The block error rates (BLERs) of our trained autoencoders compared with simulated results for different numbers of users over a range of SNR values in our multi-user case.

equalization techniques. In contrast to the single-user experiment, we observed that the models employing zero-forcing equalization achieved significantly higher performance than the other two techniques. This outcome was expected, considering that equalization becomes more challenging and crucial in the presence of interference. These results also indicate that in multi-user scenarios, autoencoders cannot achieve satisfactory performance without an effective equalization technique.

Overall Performance: The performance of our trained autoencoder-based communication models in terms of block error rate (BLER) for a range of SNR values for different numbers of users is shown in Fig. 3.13. In all charts, the 2-user and 4-user performances are depicted with blue and red colors, respectively, and the results are compared with simulated traditional BPSK and QPSK systems with hard decision decoding. The results indicate that multi-user autoencoder models can achieve almost similar performance to their counterparts in the single-user cases when compared data rate-wise. However, we have also observed that while the AWGN and Rician autoencoders outperform their peers, the Rayleigh fading autoencoders do not. We attribute this to the more complex equalization task that the

receiver in multi-user cases needs to undertake. This becomes more evident when we compare the results of our trained single-user autoencoders with BPSK and QPSK data rates, which were able to outperform all other results.

Having fine-tuned our autoencoder models and confirmed their accurate performance and behavior, we are now ready to proceed to the next phase of our thesis, which involves implementing our covert communication model on top of these autoencoder models.

Chapter 4

GAN-Based Covert Communication

This chapter presents the covert communication model that we have implemented for autoencoder wireless systems. Our aim is to create a hidden communication channel that operates seamlessly alongside the existing wireless infrastructure without causing any disruption to its normal communication activities. To achieve this, we employ a GAN training setup where different roles of the covert communication model are represented as DNNs. We then train them in an adversarial manner to develop a sophisticated and difficult-to-detect covert communication method. In the subsequent sections, we provide an overview of our system model, delve into the workings of our covert model, and conduct a thorough performance analysis of our covert model under various communication settings and channel models.

4.1 System Model

Our system consists of two types of users: normal users who communicate with each other via autoencoder wireless systems and covert users who aim to establish hidden communication

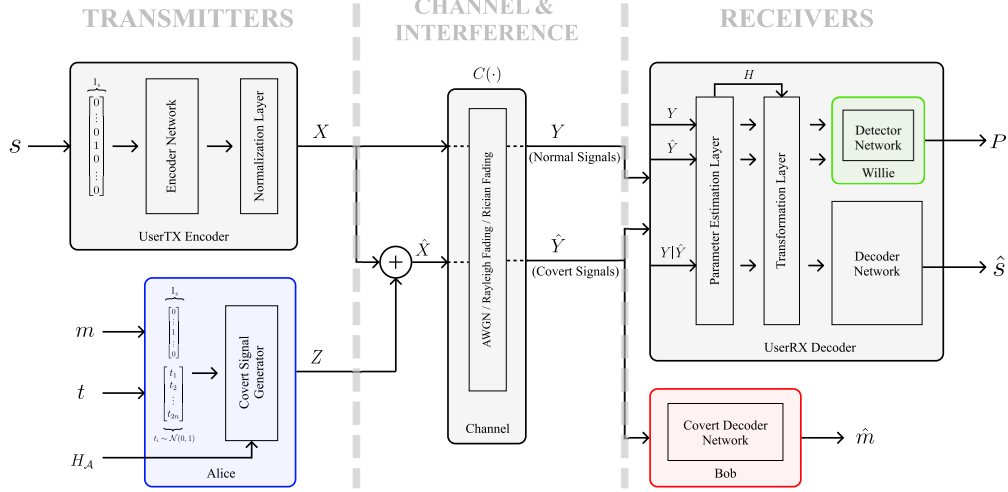


Figure 4.1: Overall architecture of our system model in the single-user scenario. The UserTX uses an encoder network to encode a binary message s into a vector of signals X . Alice generates a covert signal vector Z for a covert message m . Willie receives both normal Y and covert \hat{Y} signals during training, but at the time of operation, performs covert detection on either signal based on covert user activity. Similarly, UserRX extracts the normal messages from either Y or \hat{Y} . Bob decodes the covert message from \hat{Y} . The colored components are accessible to covert users, while the gray components are inaccessible.

in the system without arousing suspicion from the observer. The communication between normal users can be either single-user, i.e., between a transmitter and a receiver, or multi-user, i.e., between multiple transmitters and a base station receiver. In Section 3.3, we have already discussed the communication models for our normal users. Hence, in this section, our focus will be on outlining how our proposed covert communication model can be seamlessly integrated into these existing models. Figs. 4.1 and 4.2 provide an overview of the integration of our covert model into both single-user and multi-user systems. In the following sections, we will delve into each component and provide a more comprehensive explanation.

In our covert communication model, we have a covert sender (Alice) who wants to communicate secretly with her intended receiver (Bob). Bob's existence is no secret to the other entities in the system and anyone, including the observer (Willie), can see his transmissions. However, when Willie suspects that a covert sender like Alice is communicating

with Bob, he becomes alerted. Therefore, the primary goal of our covert users is to maintain Alice's communications covert.

Alice: The covert communication starts with Alice using her generative model to embed a confidential message into a covert signal vector. You can find an overview of Alice's network in Figs. 4.1 and 4.2 under the *TRANSMITTERS* sections. As illustrated in the plots, in non-fading channels, Alice merely needs a covert message and a random trigger to produce a covert signal. However, in fading channels, she requires the channel state information of her channel to Bob in the single-user case, and also the channel state information from other normal users to Bob in the multi-user case. This information will be provided by Bob, which we will explain below. After Alice produces the covert signals, she transmits them into the shared channel irrespective of other users' transmissions. This means that her covert signals should be agnostic to the signals of the normal users.

Bob: Unlike the normal receiver or Willie, Bob only uses a single antenna at his receiver. He receives the covert signals that have undergone channel effects and interference from normal users' transmissions. Without any equalization operation, he uses his decoder network to extract Alice's message from the covert signals. Additionally, he provides Alice with his and other users' channel information. He does first by sending pilot signals to Alice, and second by measuring channel information using the pilot signals sent from normal transmitters to BaseRX and then broadcasting it to Alice. As we mentioned earlier, Bob's existence need not be hidden and him broadcasting this information or pilot signals poses no risk to Alice's secrecy, who is supposed to remain covert.

Willie: Willie listens to all ongoing transmissions on the channel and uses a binary classifier network to determine the likelihood of each signal being covert or normal. Any

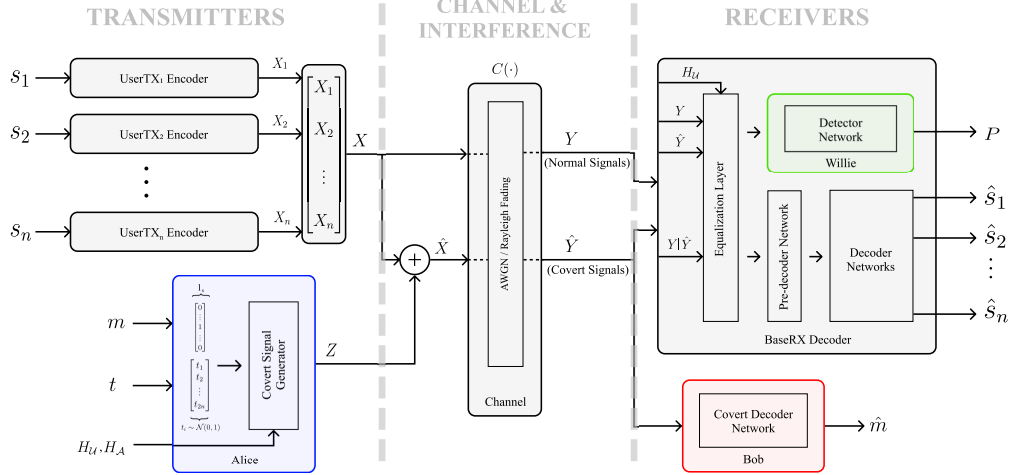


Figure 4.2: The overall architecture of our system model in the multi-user scenario. Each UserTX separately encodes binary messages s_i into individual signal vectors denoted as X_i (X is a vector of these vectors). Alice generates a covert signal vector Z for a covert message m . Willie receives both normal Y and covert \hat{Y} signals during training, but at the time of operation, performs covert detection on either signal based on covert user activity. Similarly, UserRX extracts the normal messages from either Y or \hat{Y} . Bob decodes the covert message from \hat{Y} . The colored components are accessible to covert users, while the gray components are inaccessible.

distortion in the normal signals can alert Willie to the presence of a covert transmission, so Alice must carefully select her covert signals. This means she should not make noticeable changes to the statistical properties of the channel noise or other normal signals. The covert signals should also not increase the system’s error rate, as an unexpected increase in the error rate can raise suspicion. We consider Willie to be an integrated module at the receiver of the normal communication system, i.e., UserRX in the single-user and BaseRX in the multi-user case. This way, he can not only detect incoming covert signals but also measure the communication’s error rate.

We represent the roles of Alice, Bob, and Willie with DNNs in a training setup similar to GANs. In this adversarial training setting, each of the three roles is encouraged to maximize its performance until they all reach a state of equilibrium at the end of training.

4.2 GAN-Based Covert Design

The details of Alice’s transmitter model can be seen in Figs. 4.1 and 4.2. For a given binary secret message m , Alice first applies a one-hot encoding technique, and then utilizes her generator model to produce a covert signal Z . With this stochastic generative model, each time a different covert signal is generated for the same message. Alice then sends this covert signal to the shared channel, which is accessible to all entities within the system. To simplify notations, we use \hat{X} to denote the covert signal prior to propagation through the channel.

$$\hat{X} = X + Z. \quad (4.1)$$

As previously mentioned, we consider three channel models: AWGN, Rayleigh fading, and Rician fading. Therefore, there will be three different channel outputs for these three channel models. We use a mapping function $\mathcal{C}(\cdot)$ to express each of these channels’ outputs. This is illustrated in Figs. 4.1 and 4.2 under the *CHANNEL & INTERFERENCE* section. Since signals in the multi-user case also experience interference, we express single-user and multi-user channel’s outputs separately.

AWGN Channel Output: For the AWGN channel model, the signal received at the receiver carries the channel’s noise effect $N \sim \mathcal{N}(0, \sigma^2)$. Consequently, the channel function $\mathcal{C}(\cdot)$ and the final covert signal \hat{Y} in the single-user case can be represented as:

$$\hat{Y} = \mathcal{C}(\hat{X}) = X + Z + N. \quad (4.2)$$

For the multi-user case the covert signal can be denoted as:

$$\hat{Y} = \mathcal{C}(\hat{X}) = \sum_{i \in U} X_i + Z + N. \quad (4.3)$$

where U is the set that contains all transmitters.

Rayleigh and Rician Fading Channel Outputs: In Rayleigh and Rician fading channels, transmitted signals are multiplied by a complex Gaussian random variable with zero mean and a certain variance, which represents the fading effect caused by multipath propagation. Let H_U be the complex fading coefficient(s) for the normal signal vector(s), and H_A be the complex fading coefficient for Alice's signal. The channel function $\mathcal{C}(\cdot)$ and the resulting covert signal \hat{Y} in the single-user case are given by:

$$\hat{Y} = \mathcal{C}(\hat{X}) = (H_U \cdot X) + (H_A \cdot Z) + N. \quad (4.4)$$

In the multi-user case, the received covert signal including the channel interference can be written as:

$$\hat{Y} = \mathcal{C}(\hat{X}) = \sum_{i \in U} (H_{U_i} \cdot X_i) + (H_A \cdot Z) + N. \quad (4.5)$$

You can find the inputs and outputs of Bob and Willie's networks under *RECEIVERS* section in Figs. 4.1 and 4.2.

At the receiving end, Bob is responsible for processing the covert signal \hat{Y} that carries the covert message sent by Alice. Using his decoder network, Bob performs classification on the received signal to reconstruct the covert message \hat{m} . Notably, Bob's network does not require any equalization process for decoding the covert signals.

Willie’s network is designed to classify sequences of normal signals Y and covert signals \hat{Y} . In order to ensure that the covert signals remain indistinguishable, Alice must generate them in such a way that Willie’s network classifies them as normal signals. To achieve this, Alice requires access to the classifier used by Willie’s network. However, this is not feasible in a real-world scenario. Therefore, Alice and Bob perform a black-box attack against Willie’s network using a substitute binary classifier. Previous studies have shown that adversarial attacks against one model can be effective on another model, even if the two models have different training sets and architectures [40]. Henceforth, we employ Willie’s substitute network to provide feedback to Alice during training. This feedback assists her in adjusting the covert signals to ensure they are indistinguishable from normally transmitted signals. Additionally, it ensures that when the model is deployed in a real communication setup, it is highly unlikely that any observer will detect the covert transmissions.

4.2.1 General Formulation

4.2.1.1 Reliability

The first objective of our covert model is to enable reliable covert communication. In order to achieve this, Bob needs to be able to accurately decode the covert messages sent by Alice. As mentioned earlier, Alice employs a generative model to produce covert signals that correspond to the covert message. Let $\Theta_{\mathcal{A}}(\cdot)$ be the underlying function of Alice’s generative model that takes a random trigger $t \sim \mathcal{N}(0, 1)$, a covert message m , the channel coefficients from Alice to Bob $H_{\mathcal{A}}$, and in the multi-user case, the channel matrix $H_{\mathcal{U}}$ and

¹Parameter is needed only for fading channel.

²Parameter is needed only for multi-user fading channel.

produces a covert signal Z . The corresponding covert signal can be denoted as $Z_{m,t} = \Theta_{\mathcal{A}}(m, t, H_{\mathcal{A}}^1, H_{\mathcal{U}}^2)$. Let $\Theta_{\mathcal{B}}(\cdot)$ be the underlying function of the decoder network that Bob uses to reconstruct the covert message \hat{m} . Then the reliability of communication between Alice and Bob is achieved using the following loss function:

$$\begin{aligned}
\mathcal{L}_{\mathcal{B}} &= \mathbb{E}_m[\mathcal{H}(\hat{m}, m)] \\
&= \mathbb{E}_m[\mathcal{H}(\Theta_{\mathcal{B}}(\hat{Y}), m)] \\
&= \mathbb{E}_m[\mathcal{H}(\Theta_{\mathcal{B}}(\mathcal{C}(\hat{X})), m)] \\
&= \mathbb{E}_m[\mathcal{H}(\Theta_{\mathcal{B}}(\mathcal{C}(\Theta_{\mathcal{A}}(m, t, H_{\mathcal{A}}) + X)), m)].
\end{aligned} \tag{4.6}$$

For the multi-user case, this equation is written as:

$$\begin{aligned}
\mathcal{L}_{\mathcal{B}} &= \mathbb{E}_m[\mathcal{H}(\hat{m}, m)] \\
&= \mathbb{E}_m[\mathcal{H}(\Theta_{\mathcal{B}}(\mathcal{C}(\Theta_{\mathcal{A}}(m, t, H_{\mathcal{A}}, H_{\mathcal{U}}) + X)), m)].
\end{aligned} \tag{4.7}$$

The equation above uses the cross-entropy function $\mathcal{H}(\cdot)$ to measure the difference between the probability distribution of the reconstructed covert message \hat{m} and that of the actual covert message m . This equation can be used to optimize the networks of both Alice and Bob by freezing one or the other's network parameters iteratively.

4.2.1.2 Low Interference

While (4.6) ensures communication accuracy, we also need to ensure that the generated perturbations do not negatively impact normal communication between UserTX and UserRX. Otherwise, this could alert Willie to abnormal activity. To address this, we add a constraint

that minimizes UserRX’s loss function during Alice’s training. In a single-user system, we can express this constraint as follows:

$$\begin{aligned}
\mathcal{L}_U &= \mathbb{E}_m[\mathcal{H}(\hat{s}, s)] \\
&= \mathbb{E}_m[\mathcal{H}(\Theta_U(\hat{Y}), s)] \\
&= \mathbb{E}_m[\mathcal{H}(\Theta_U(\mathcal{C}(\hat{X})), s)] \\
&= \mathbb{E}_m[\mathcal{H}(\Theta_U(\mathcal{C}(\Theta_{\mathcal{A}}(m, t, H_{\mathcal{A}}) + X)), s)].
\end{aligned} \tag{4.8}$$

where $\Theta_U(\cdot)$ refers to UserRX’s decoder network function. It is important to note that during this training, UserRX’s decoder network is frozen and only Alice’s parameters will be updated.

For the multi-user case, since we have multiple transmitters sending signals, we need to minimize BaseRX’s loss function over all individual transmitters’ signals. Thus, (4.8) is rewritten as follows:

$$\begin{aligned}
\mathcal{L}_U &= \sum_{i \in U} \mathbb{E}_m[\mathcal{H}(\hat{s}_i, s_i)] \\
&= \sum_{i \in U} \mathbb{E}_m[\mathcal{H}(\Theta_U(\mathcal{C}(\Theta_{\mathcal{A}}(m, t, H_{\mathcal{A}}, H_U) + X), H_U), s_i)].
\end{aligned} \tag{4.9}$$

4.2.1.3 Undetectability

In our model, Willie acts as a critic network, similar to what is known as a discriminator in GAN models [15]. The so-called real and fake samples in a GANs’ discriminator training process here are mapped to normal and covert signals, respectively. For the single-user case, we use the following loss function to optimize Willie’s network that classifies covert and

Algorithm 2 Covert Model Training

Time Complexity: $O(n)$, where n is the number of epochs
 $X \leftarrow$ normal signals data
 $S, M \leftarrow$ normal and covert messages sets
 $\Theta_{\mathcal{A}}, \Theta_{\mathcal{B}}, \Theta_{\mathcal{W}} \leftarrow$ Alice, Bob, and Willie networks
 $\Theta_{\mathcal{U}} \leftarrow$ UserRX decoder network
 $\mathcal{H} \leftarrow$ cross entropy
 $\mathcal{C} \leftarrow$ channel mapping function
for epoch $ep \in \{1 \dots n_{epochs}\}$ **do**
 $t \sim \mathcal{N}(0, 1)$
 $\mathcal{L}_{\mathcal{W}} = \mathcal{H}(\Theta_{\mathcal{W}}(\mathcal{C}(\Theta_{\mathcal{A}}(M, t, H_{\mathcal{A}}, H_{\mathcal{U}}) + X)), \Theta_{\mathcal{W}}(\mathcal{C}(X)))$
 Update $\Theta_{\mathcal{W}}$ to minimize $\mathcal{L}_{\mathcal{W}}$
 $\mathcal{L}_{\mathcal{B}} = \mathcal{H}(\Theta_{\mathcal{B}}(\mathcal{C}(\Theta_{\mathcal{A}}(M, t, H_{\mathcal{A}}, H_{\mathcal{U}}) + X)), M)$
 Update $\Theta_{\mathcal{B}}$ to minimize $\mathcal{L}_{\mathcal{B}}$
 $\mathcal{L}_{\mathcal{U}} \leftarrow \mathcal{H}(\Theta_{\mathcal{U}}(\mathcal{C}(\Theta_{\mathcal{A}}(M, t, H_{\mathcal{A}}, H_{\mathcal{U}}) + X)), S)$
 $\mathcal{L}_{\mathcal{A}} = \lambda_{\mathcal{B}}\mathcal{L}_{\mathcal{B}} + \lambda_{\mathcal{U}}\mathcal{L}_{\mathcal{U}} - \lambda_{\mathcal{W}}\mathcal{L}_{\mathcal{W}}$
 Update $\Theta_{\mathcal{A}}$ to minimize $\mathcal{L}_{\mathcal{A}}$
end for

normal signals:

$$\begin{aligned}
 \mathcal{L}_{\mathcal{W}} &= \mathbb{E}_m[\mathcal{H}(\hat{Y}, Y)] \\
 &= \mathbb{E}_m[\mathcal{H}(\Theta_{\mathcal{W}}(\mathcal{C}(\hat{X})), \Theta_{\mathcal{W}}(\mathcal{C}(X)))] \\
 &= \mathbb{E}_m[\mathcal{H}(\Theta_{\mathcal{W}}(\mathcal{C}(\Theta_{\mathcal{A}}(m, t, H_{\mathcal{A}}) + X)), \Theta_{\mathcal{W}}(\mathcal{C}(X)))].
 \end{aligned} \tag{4.10}$$

where $\mathcal{H}(\cdot)$ is the binary cross entropy between the predicted outputs for the covert signals \hat{Y} or the normal signals Y and their actual outputs. For the multi-user case, we need to optimize Willie's network over all the transmitters' outputs.

$$\begin{aligned}
 \mathcal{L}_{\mathcal{W}} &= \sum_{i \in \mathcal{U}} \mathbb{E}_m[\mathcal{H}(\hat{Y}, Y)] \\
 &= \sum_{i \in \mathcal{U}} \mathbb{E}_m[\mathcal{H}(\Theta_{\mathcal{W}}(\mathcal{C}(\Theta_{\mathcal{A}}(m, t, H_{\mathcal{A}}, H_{\mathcal{U}}) + X)), \Theta_{\mathcal{W}}(\mathcal{C}(X)))].
 \end{aligned} \tag{4.11}$$

This white-box adversarial training against Alice's network ensures that Willie's network

will be adequately trained to distinguish between covert and normal signals. On the other hand, we do not want the covert signals produced by Alice to deviate from the statistical properties of the normal signals on the channel, as this would increase the likelihood of the observer detecting and mitigating the covert communication. To achieve this undetectability property, we use Willie’s network to act as a discriminator network in Alice’s optimization function. In other words, Alice’s training against this network aims to maximize Willie’s uncertainty about his predictions. This regularizer helps Alice form their covert communication in a way that is indistinguishable from the actual channel’s noise, yet understandable by Bob. Overall, Alice’s loss function can be expressed as a weighted sum of these three objectives:

$$\mathcal{L}_{\mathcal{A}} = \lambda_{\mathcal{B}}\mathcal{L}_{\mathcal{B}} + \lambda_{\mathcal{U}}\mathcal{L}_{\mathcal{U}} - \lambda_{\mathcal{W}}\mathcal{L}_{\mathcal{W}}. \quad (4.12)$$

where $\lambda_{\mathcal{B}}$, $\lambda_{\mathcal{U}}$, and $\lambda_{\mathcal{W}}$ are hyperparameters that determine the relative importance of the different objectives in Alice’s loss function. The algorithmic steps involved in training our covert models are summarized in Algorithm 2.

4.2.2 Neural Network Architecture

4.2.2.1 Generator (Alice)

Alice first transforms a covert message m to its corresponding one-hot encoding representation, where each message belongs to a unique class. She then uses a random trigger t to randomize the process by which the covert noise signal Z is produced, along with the channel coefficients $H_{\mathcal{A}}$ and $H_{\mathcal{U}}$. For Alice’s generator model, we use multiple dense layers with ReLU and Tanh activation functions. The first layer acts as an embedding layer, enlarging the input’s

domain space. The subsequent fully connected layers extract useful features and perform the encoding process. Finally, the last layer performs a dimension transformation, ensuring that the generated covert signal Z complies with the dimension of the normal signal X on the channel.

4.2.2.2 Decoder (Bob)

Bob receives the covert signal \hat{Y} that has been affected by the channel, and he feeds it through his decoder network to extract the secret message. Bob's network is more sophisticated than Alice's, as decoding such a distorted signal is a much more complex task. The received message first goes through a wide dense layer with a Tanh activation function, which increases the input's feature space. The data then passes through multiple 1-Dimensional Convolutional (1D Conv) layers, which learn the coding that Alice has developed to encode the covert messages. We have found that using 1D Conv layers helps Bob and Alice achieve better consistency in the accuracy of their communication, especially when the channel model is more complicated (i.e., when there is also fading in the channel). The rest of Bob's decoder network consists of two dense layers that remap the learned feature space to the covert message domain space. As with UserRX's and BaseRX's decoder networks, Bob eventually predicts the covert message by performing a classification on the received signal.

4.2.2.3 Discriminator (Willie)

Willie's task is to distinguish between the normal signal Y and the covert signal \hat{Y} . To achieve this, he uses a neural network with the same architecture as Bob's, except for the last layer, which has a Sigmoid activation function instead of Softmax. The network takes

Table 4.1: Alice, Bob, and Willie’s detailed network architecture in the single-user and multi-user case.

	Alice (Single-User)	Alice (Multi-User) AWGN	Alice (Multi-User) Rayleigh/Rician	Bob	Willie
input size	$16 + 2^m$	$16 + 2^m$	$16 + 2^m + (n_{tx} \times 2 + 2)$	16	16
dense layers sizes	$32 + 2^{m+1},$ $32 + 2^{m+1},$ 8×2^m	$32 + 2^{m+1},$ $32 + 2^{m+1},$ 8×2^m	$32 + 2^{m+1} + (n_{tx} \times 2 + 2),$ $32 + 2^{m+1},$ $32 + 2^{m+1},$ 8×2^m	$2 \times 8, 16$	$2 \times 8, 2 \times 8$
dense layers activations	$3 \times \text{ReLU, Tanh}$	$3 \times \text{ReLU, Tanh}$	$4 \times \text{ReLU, Tanh}$	$2 \times \text{Tanh, Softmax}$	$2 \times \text{Tanh, Sigmoid}$
conv filters	-	-	-	1, 8, 8, 8, 8	1, 8, 8, 8, 8
conv kernel sizes	-	-	-	1, 2, 4, 2, 2	1, 2, 4, 2, 2
conv strides	-	-	-	1, 1, 2, 1, 1	1, 1, 2, 1, 1
conv activations	-	-	-	$5 \times \text{LeakyReLU}$	$5 \times \text{LeakyReLU}$
output size	2×8	2×8	2×8	2^k	1

an input signal, either normal or covert, and produces a confidence probability P indicating the likelihood of the signal being normal. Using the same network architecture for both Bob and Willie ensures a fair competition between them, as they have the same capacity for learning.

4.3 Evaluation

We evaluated the performance of our covert communication models on three different channel models: AWGN, Rayleigh fading, and Rician fading. We used the same training procedure for all settings, but the network architecture of our covert and autoencoder models in the multi-user case differed slightly from that in the single-user setting. Table 4.1 shows these differences.

4.3.1 Methodology

Dataset: Since each covert message m has to be paired with a normal message s , we created the covert model’s training and testing sets to have the same number of samples as the autoencoder’s. Similar to what we describe in 3.5.1.1, this would give us $2^{|m|}$ number of unique message, where $|m|$ is the length of the covert messages.

Optimization Parameters: All models were trained for 5000 epochs using the Adam optimizer in an adversarial training setting. The entire dataset was fed to the network at each epoch rather than feeding batches of data. In both the single-user and multi-user cases, we started the training with a learning rate of 0.001 for the first 2500 epochs and then made the learning rate ten times smaller for the remaining 2500 epochs.

Hyper Parameters: We adjusted the importance of each of Alice’s objectives by setting $\lambda_{\mathcal{W}} = 2\lambda_{\mathcal{B}} = 4\lambda_{\mathcal{U}}$ for the single-user case, and $\lambda_{\mathcal{W}} = 3\lambda_{\mathcal{B}} = 6\lambda_{\mathcal{U}}$ for the multi-user case in (4.12). We arrived at these numbers by running a grid search on these parameters. However, our solution is not limited to these parameters, and one can use a different set of parameters to emphasize one specific objective more than the others.

In each epoch, we first updated the parameters of Willie’s network using (4.10), then trained Alice’s network for one step using (4.12), and finally optimized Bob’s network based on (4.6).

Although we trained our autoencoder network on a fixed SNR value, we found that our covert scheme performed better when trained on a range of SNR values. We achieved this by randomly switching the SNR value within a predetermined range after each epoch of training. Training our models this way not only helped Alice better preserve the normal communication’s accuracy but also enabled Bob to decode covert messages more accurately on lower SNR values. Accordingly, we started by setting the training SNR to the value that the autoencoder was trained on and incrementally expanded the SNR range from both ends until no further improvement was observed. Algorithm 3 shows the steps of this process. As a result, in the single-user case, we settled on the range of -2dB to 8dB for the AWGN channel and 10dB to 30dB for both the Rayleigh and Rician fading channels. In the multi-

Algorithm 3 Optimal SNR Range Search

Time Complexity: $O(m^2)$, where m is the maximum number of iterations for the search
 $acc_{\mathcal{U}, \mathcal{B}, \mathcal{W}} \leftarrow$ User, Bob, and Willie final training accuracies
 $p, c \leftarrow$ Previous and current average training accuracies
 $snr_{L, U} \leftarrow$ Optimal lower and upper bounds of the SNR range
 $snr_T \leftarrow$ SNR value that the autoencoder was trained on
 $t \leftarrow L$ Tracking the SNR bound that is expanding
 $m \leftarrow$ Maximum number of iterations
 $snr_{L, U} \leftarrow snr_T$
for iteration $it \in \{1 \dots m\}$ **do**
 $acc_{\mathcal{U}}, acc_{\mathcal{B}}, acc_{\mathcal{W}} \leftarrow Train(sn r_L, sn r_U)$
 $c \leftarrow Avg(acc_{\mathcal{U}}, acc_{\mathcal{B}}, acc_{\mathcal{W}})$
 if $c > p$ **then**
 $p \leftarrow c$
 $snr_t \leftarrow snr_t \pm 1$
 else
 if t is equal L **then**
 $t \leftarrow U$
 else
 return $snr_{L, U}$
 end if
 end if
end for

user system, the optimal range was found within the 0dB to 10dB range for the AWGN channel, 0dB to 20dB for the Rician channel, and 10dB to 30dB for the Rayleigh channel.

4.3.2 Training Procedure

Fig. 4.3 shows the progress of each covert actor's accuracy on the test set during the training process for both single-user and multi-user cases. As the training progresses, Bob gradually learns to decode covert messages m and establishes reliable communication with Alice. After a few epochs as the covert communication begins to take form and stabilize, the signals start to deviate from their original distribution, which helps Willie to better detect covert signals. When Willie's accuracy increases, the term $\mathcal{L}_{\mathcal{W}}$ dominates the other two objectives of Alice's loss function in (4.12). As a result, Alice gradually sacrifices accuracy for undetectability.

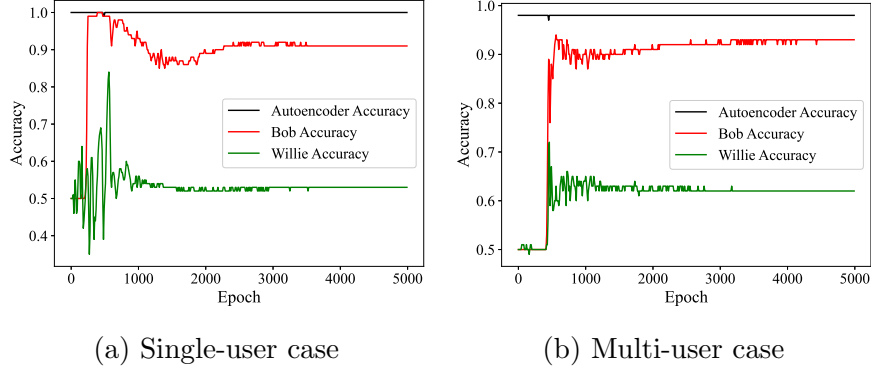


Figure 4.3: Evaluation results of our covert and autoencoder models during the training process show the system reaches a stable point after successful training.

Soon after, the training process reaches a stable point where neither of the covert models sees any significant improvement in accuracy as the training progresses. At the end of the training, the Users' accuracy remains intact, Bob achieves reliable covert communication accuracy, and Willie stabilizes at around 50~60% accuracy, which, for a binary classifier, is very close to random guessing accuracy.

4.3.3 Single-User Experiments

We started our experiments with the single-user case. First, we evaluated our covert models by sending 1 bit of covert data over 8 channel uses and then gradually increased the number of covert bits to see how increasing the covert data rate affected each component of our covert scheme. We used the notations $Alice(n, k)$, $Bob(n, k)$, and $Willie(n, k)$ to differentiate models with different covert data rates, and their interpretation was the same as that of the autoencoder model.

Figs. 4.4, 4.5, and 4.6 illustrate the performance of our scheme for different covert data rates and how reliable our covert models are at different covert data rates. As we expected, with increasing covert data rates, covert communication becomes more unreliable, its impact

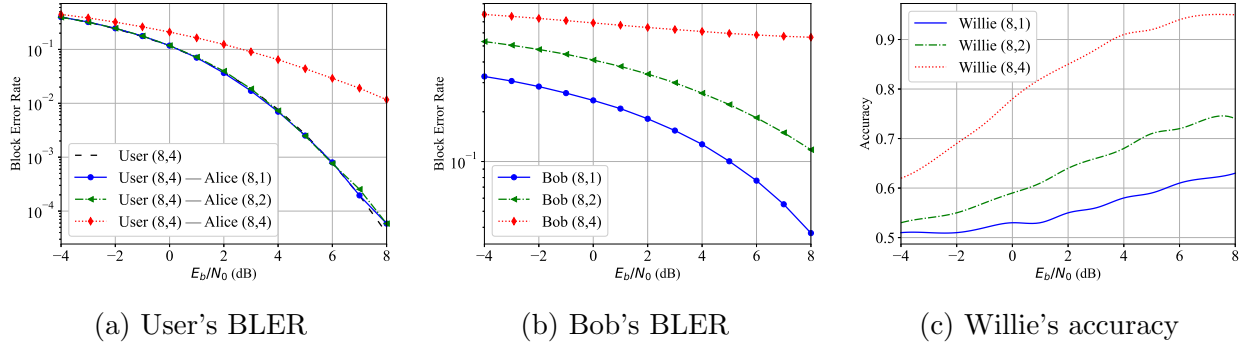


Figure 4.4: Single-user covert models' performance over AWGN channel for different covert data rates on a range of SNRs.

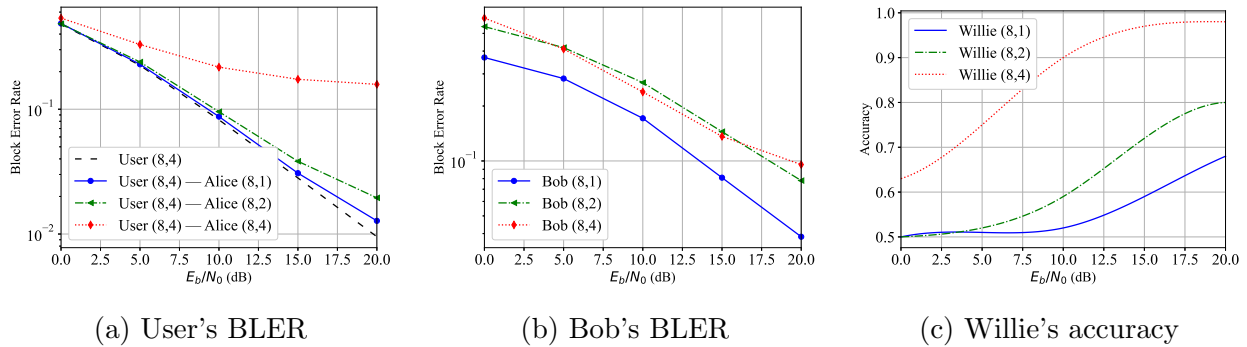


Figure 4.5: Single-user covert models' performance over Rayleigh fading channel for different covert data rates on a range of SNRs.

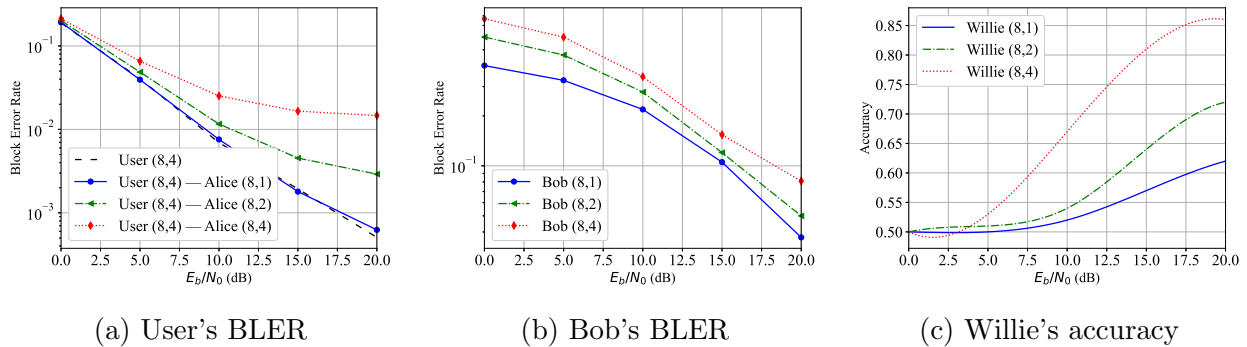


Figure 4.6: Single-user covert models' performance over Rician fading channel for different covert data rates on a range of SNRs.

on the normal communication increases, and detection becomes easier for Willie. Overall, these plots indicate that sending covert data at high rates makes covert communication unreliable.

The plots also reveal how the communication channel affects the performance of each actor. In the AWGN channel, higher covert rates have a relatively smaller impact on the User’s BLER, while in the fading channels, their impact is more significant. On the other hand, increasing the covert rate in the fading channels has less effect on the covert communication accuracy compared to the AWGN channel. For Willie, all channels exhibit a similar trend, where higher covert rates are more susceptible to detection.

Through these experiments, we have concluded that the most reliable covert data rate is achieved by sending 1 bit of data over 8 channel uses. Therefore, we will be using these parameters as the default when evaluating our models in the multi-user scenario.

4.3.4 Multi-User Experiments

After evaluating the reliability of our covert models for different covert data rates, we now aim to measure the robustness of our covert scheme against the number of users in the multi-user scenario. To do this, we evaluate our covert models in systems comprising of 2 and 4 users. This will help us understand how adding users, i.e., increasing interference, affects the performance of our covert models, and whether it has a more significant impact on communication than increasing the covert data rate.

Figs. 4.7, 4.8, and 4.9 present our results for 2-user and 4-user systems, demonstrating how the number of users in the system affects our model’s performance. For the AWGN channel, we observe that adding more users does not change the model’s overall performance. Furthermore, as the number of users increases, there is almost no impact on the normal receivers from the covert transmissions, and Bob and Willie’s performances remain almost

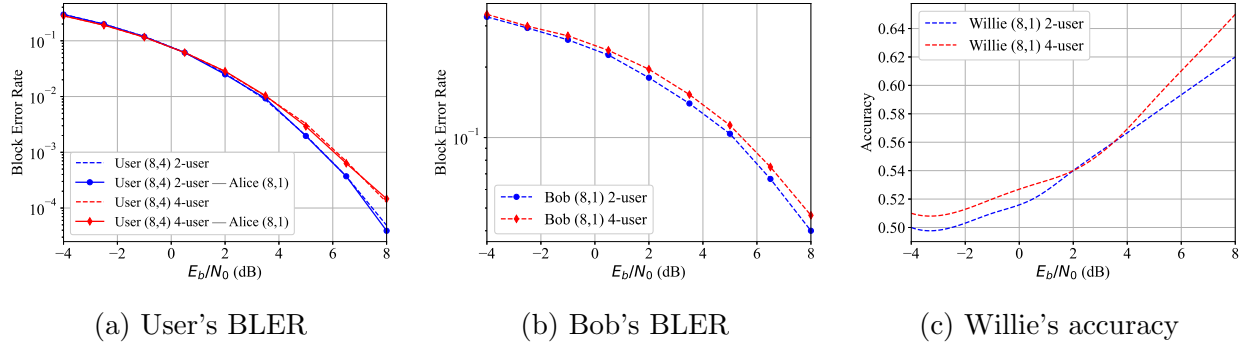


Figure 4.7: Multi-user covert models' performances over AWGN channel for systems with different numbers of users over a range of SNRs.

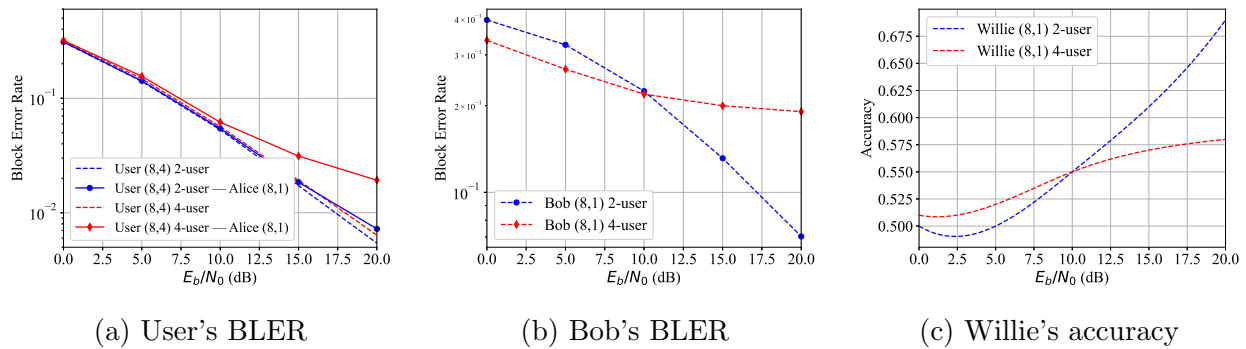


Figure 4.8: Multi-user covert models' performances over Rayleigh channel for systems with different number of users over a range of SNRs.

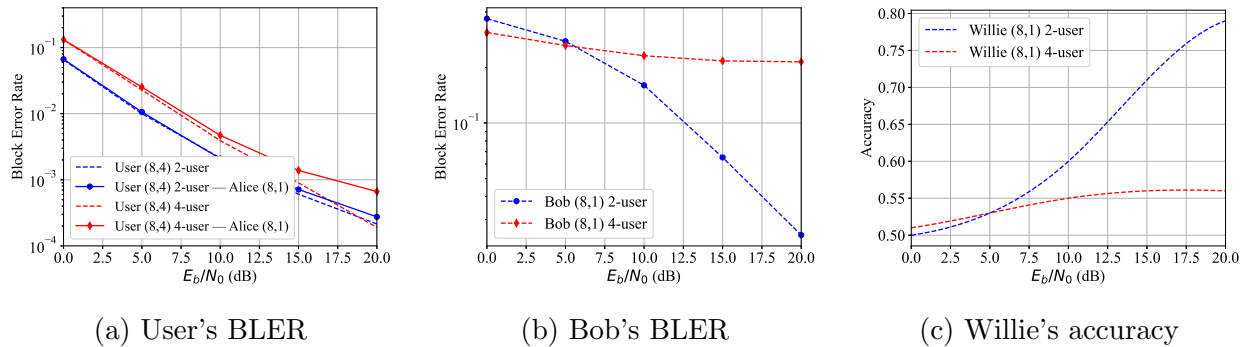


Figure 4.9: Multi-user covert models' performances over Rician channel for systems with different number of users over a range of SNRs.

the same.

However, for the Rayleigh and Rician channels, a degree of freedom effect can be noticed, where increasing number of users makes it more challenging for the covert users to avoid

interfering with the ongoing normal communication. As a result, the impact of covert communication on normal users becomes more detrimental with a higher number of users. Unlike in the AWGN channel, adding more users in these cases significantly affects Bob's performance, rendering covert communication practically ineffective. Looking at Figs. 4.8 and 4.9, we can observe a distinct cross-over pattern for the fading channels. Specifically, Figs. 4.8a and 4.9a reveal that there is a certain SNR at which the covert communication in the 4-user systems begins to have a greater impact on normal communication compared to the 2-user systems. These SNRs are 10dB and 5db in the Rayleigh and Rician channels, respectively. These points indicate that covert users can no longer communicate reliably without causing interference to normal users. This behavior is even more apparent in Figs. 4.8b and 4.9b, which show that Bob's BLER begins to degrade at the same SNR values and eventually plateaus, deviating from the performance of the 2-user system. Likewise, we can see the same pattern in Willie's detection accuracy. Since covert communication has no specific pattern from these points further, Willie is unable to detect it accurately and thereby his detection accuracy remains constant.

4.3.5 Undetectability

Willie's detection accuracy can be found in figs. 4.4c, 4.5c, and 4.6c for the single-user case, and in Figs. 4.7c, 4.8c, 4.9c for the multi-user case. His detection performance is evaluated over a range of SNR values for detecting signals as covert and normal. In the single-user experiments, we observe that as the covert data rate increases, the covert communication becomes more easily detectable. In the multi-user case, we cannot directly compare Willie's

accuracy for different numbers of users because covert users are unable to establish their covert communication in the fading channels in systems with 4 users. However, the results from the AWGN channel indicate that Willie’s accuracy remains roughly the same as we increase the number of users.

4.3.6 Constellation Diagrams

Fig. 4.10 compares the constellation clouds of covert and normal signals for AWGN, Rayleigh, and Rician fading channels in the single-user system. We marked each symbol of the encoder’s output signal as black circle points on the constellation diagrams. The red constellation cloud shows how covert signals scatter after passing through the channel, and the green cloud shows this for normal signals. Since data is sent over 8 channel uses, there are 8 black points on the chart. To maintain consistency with Willie’s accuracy and Bob’s error rate for all channel models, we set the SNR value to 6dB for the AWGN, 15dB for the Rayleigh fading channel, and 16dB for the Rician fading channel. This ensured that, in all channel models, the probability of detection remained relatively the same, and the covert communication BLER stayed below 10^{-1} . This area of operation provided Alice and Bob relative reliability in their covert communication while maintaining their covertness. Looking at these figures,

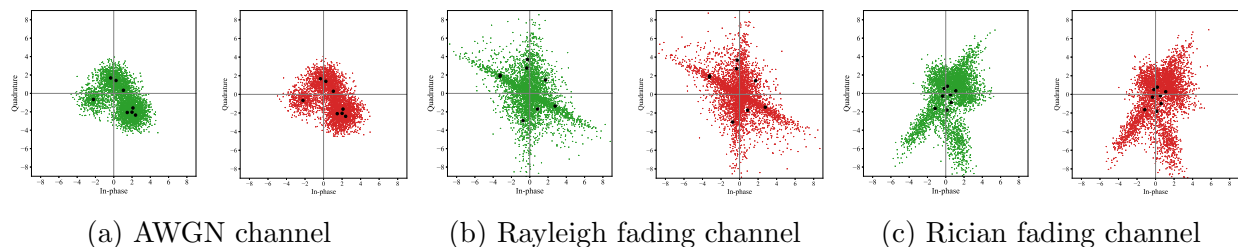


Figure 4.10: Comparing AWGN, Rayleigh and Rician fading channels constellation clouds of a sample signal. The green clouds show the constellation before the covert communication and the red clouds show it after.

the signal constellation diagrams before and after applying our covert model are very similar, showing that to a first-order, Alice has perfectly learned to cloak the covert signals into the distribution of the channel's noise.

4.3.7 Results Validation

To validate the credibility and accuracy of the outcomes derived from our developed autoencoder wireless models, we undertake a comprehensive validation strategy comprising two primary approaches. Firstly, we compare our achieved results with those presented in the original autoencoder wireless model paper. This rigorous comparison serves to establish a strong alignment with established benchmarks and confirms the reliability of our findings within the context of prior research.

Secondly, our validation process extends to the integration of theoretical insights, where theoretical predictions are provided for various experiments. These theoretical benchmarks serve as a crucial reference point, allowing us to evaluate the empirical outcomes against expected trends and behaviors.

Chapter 5

Conclusion and Future Work

In this thesis, we conducted an extensive review of autoencoder wireless systems, exploring their performance in single and multi-user communication scenarios under various channel conditions. Our experiments aimed to understand the factors that influence the performance of autoencoder models in wireless communications. These experiments provided us with a deeper understanding of the concepts and behaviors associated with autoencoder wireless communications. Additionally, we prepared a solid foundation for our covert communication model by fine-tuning the autoencoder models and thoroughly validating their behaviors and performance.

Next, we introduced a novel deep learning-based covert communication approach that seamlessly integrates with autoencoder wireless systems. Our covert model successfully demonstrated its ability to embed secret messages into covert signals without relying on handcrafted features. Through the utilization of the generative adversarial training framework, we significantly reduced the detection probability of the covert signals produced. Furthermore, we proposed a training procedure that allows us to adjust the trade-off between the conflicting

objectives of covert communication, which are reliability of communication and the probability of detection. This adjustment is achieved by introducing regularizers into the model’s loss function, independent of channel conditions or user messages. Our findings demonstrate that our covert model is channel-agnostic and insensitive to cover signals. To evaluate the performance of our model, we conducted assessments across three channel models: AWGN, Rayleigh, and Rician fading. We varied covert rates and the number of system users to analyze the model’s robustness. Additionally, we investigated the impact of our covert signals on the ongoing normal system and confirmed that our covert scheme causes minimal disruption to the system.

5.1 Thesis Summary

In Chapter 1, we discuss the motivations for our research in this thesis, specifically addressing the need to safeguard wireless network security against covert communication threats. We emphasize the significance of understanding covert communication methods and the vulnerability of autoencoder-based wireless systems to such threats. Additionally, we outline the main objectives of our work and highlight our contributions in this field.

In Chapter 2, we provide an overview of the fundamental concepts and solutions that form the basis of this thesis. We begin by reviewing artificial neural networks, including autoencoder networks and generative adversarial networks, which are key components used in our research. We then introduce covert communication, discussing its taxonomies and its application at the physical layer, as well as traditional and machine learning wireless communication techniques. Additionally, we provide a brief summary of the most relevant

research in the field, categorized into image steganography (the field that inspired our idea), traditional covert communication techniques, and deep learning-based covert communication techniques. In the end of the chapter, we discuss the software and hardware tools utilized during the implementation phase of this research. This encompasses PyTorch, Principal Component Analysis (PCA), Matplotlib, as well as our hardware configuration employed for model training.

In Chapter 3, we begin with an overview of autoencoder wireless systems, covering their neural network architecture, training process, and evaluation. We then delve into the specifics of single-user and multi-user scenarios within these systems and present the system models that we have considered. In this chapter, we also provide our neural network design for these system, and thoroughly examine each component of it. Lastly, we assess the performance of our trained models in both single-user and multi-user scenarios.

In Chapter 4, we introduce our novel GAN-based covert communication model: a self-learn covert model that requires no hand-crafted features and can be integrated into autoencoder-based communication systems. We begin this chapter by explaining our system models, outlining the objectives and functionalities of each entity. The two system models that have been considered are illustrated in this chapter: single-user and multi-user communication scenarios. Next, we presented our design for our GAN-based covert model and explained how we formulated our objectives. Furthermore, we detailed the neural network architecture employed for our generator model (Alice), decoder (Bob), and discriminator (Willie). In the concluding section of the chapter, we assessed the effectiveness of our covert model in both system models mentioned earlier, assessing its reliability, covertness, and impact on the ongoing communication.

5.2 Future Work

This thesis offers several potential avenues for further exploration, some of which are outlined below:

- **Black-box Systems:** The covert communication model presented in this thesis assumes covert users have white-box access to the normal communication system. In other words, covert sender can minimize the impact of their covert transmissions on the normal communication by utilizing the normal receiver’s decoder network loss as a regularizer. Future research could investigate the feasibility and effectiveness of our covert scheme in black-box systems, where covert users do not have access to the communication networks of normal users. This would entail training our scheme against a substitute normal receiver and studying its performance in such scenarios.
- **Active Observer:** In this thesis, our assumption was that the system observer is passive, solely detecting the presence or absence of covert communication. However, it would be valuable to investigate the effects of an active observer on the performance of our covert communication model. An active observer can deliberately perturb signals within a threshold that minimally affects normal communication quality but interferes with the covert communication link. For instance, a straightforward example would involve adding Gaussian noise with a specific variance to the signals.
- **Real-world Implementation:** All the results and evaluations presented in this thesis are obtained through simulation. In the simulation, we treat autoencoder wireless communication systems as a cohesive unit, with the channel represented as a

mathematical layer between the encoder and decoder. This give us a computationally tractable model that can be optimized using algorithms like gradient descent. However, in real-world scenarios, the encoder, channel, and decoder, as well as the covert sender and covert receiver of our model, are physically separated components. Therefore, further investigation is needed to explore solutions for systems with unknown or non-mathematically representable channel models.

Bibliography

- [1] Youness Arjoune, Fatima Salahdine, Md Shoriful Islam, Elias Ghribi, and Naima Kaabouch. A novel jamming attacks detection approach based on machine learning for wireless communication. pages 459–464, 2020.
- [2] Alireza Bahramali, Milad Nasr, Amir Houmansadr, Dennis Goeckel, and Don Towsley. Robust adversarial attacks against dnn-based wireless communication systems. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 126–140, 2021.
- [3] Shumeet Baluja. Hiding images in plain sight: Deep steganography. *Advances in neural information processing systems*, 30, 2017.
- [4] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *arXiv preprint arXiv:2003.05991*, 2020.
- [5] Boulat A Bash, Dennis Goeckel, and Don Towsley. Square root law for communication with low probability of detection on awgn channels. In *Proceedings of 2012 IEEE International Symposium on Information Theory*, pages 448–452. IEEE, 2012.
- [6] Boulat A Bash, Dennis Goeckel, and Don Towsley. Limits of reliable communication with low probability of detection on awgn channels. *IEEE journal on selected areas in communications*, 31(9):1921–1930, 2013.
- [7] Matthieu R Bloch. Covert communication over noisy channels: A resolvability perspective. *IEEE Transactions on Information Theory*, 62(5):2334–2354, 2016.
- [8] Leonardo Bonati, Salvatore D’Oro, Francesco Restuccia, Stefano Basagni, and Tommaso Melodia. Stealte: Private 5g cellular connectivity as a service with full-stack wireless steganography. pages 1–10, 2021.
- [9] Serdar Cabuk, Carla E Brodley, and Clay Shields. Ip covert timing channels: design and detection. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 178–187, 2004.
- [10] Pengcheng Cao, Weiwei Liu, Guangjie Liu, Xiaopeng Ji, Jiangtao Zhai, and Yuewei Dai. A wireless covert channel based on constellation shaping modulation. *Security and Communication Networks*, 2018, 2018.

- [11] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.
- [12] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- [13] Aveek Dutta, Dola Saha, Dirk Grunwald, and Douglas Sicker. Secret agent radio: Covert communication through dirty constellations. pages 160–175, 2012.
- [14] Vijay Garg. *Wireless communications & networking*. Elsevier, 2010.
- [15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.
- [16] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [17] Krystian Grzesiak, Zbigniew Piotrowski, and Jan M Kelner. A wireless covert channel based on dirty constellation with phase drift. *Electronics*, 10(6):647, 2021.
- [18] Samer S Hanna and Danijela Cabric. Deep learning based transmitter identification using power amplifier nonlinearity. pages 674–680, 2019.
- [19] Jamie Hayes and George Danezis. Generating steganographic images via adversarial training. *Advances in neural information processing systems*, 30, 2017.
- [20] Biao He, Shihao Yan, Xiangyun Zhou, and Vincent KN Lau. On covert communication with noise uncertainty. *IEEE Communications Letters*, 21(4):941–944, 2017.
- [21] Michael Hegarty, Yalin E Sagduyu, Tugba Erpek, and Yi Shi. Deep learning for spectrum awareness and covert communications via unintended rf emanations. In *Proceedings of the 2022 ACM Workshop on Wireless Security and Machine Learning*, pages 27–32, 2022.
- [22] Ningning Hou and Yuanqing Zheng. Cloaklora: A covert channel over lora phy. pages 1–11, 2020.
- [23] Shuhua Huang, Weiwei Liu, Guangjie Liu, Yuewei Dai, and Huiwen Bai. Detection of constellation-modulated wireless covert channel based on adjusted cnn model. *Security and Communication Networks*, 2021, 2021.
- [24] Shuhua Huang, Weiwei Liu, Guangjie Liu, Yuewei Dai, and Wen Tian. Exploiting correlation characteristics to detect covert digital communication. *KSII Transactions on Internet and Information Systems (TIIS)*, 14(8):3550–3566, 2020.
- [25] Brian Kim, Tugba Erpek, Yalin E Sagduyu, and Sennur Ulukus. Covert communications via adversarial machine learning and reconfigurable intelligent surfaces. pages 411–416, 2022.

- [26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [27] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [28] Andrej Krenker, Janez Bešter, and Andrej Kos. Introduction to the artificial neural networks. *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech*, pages 1–18, 2011.
- [29] Butler W Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.
- [30] Ke Li, Majid Ghaderi, and Dennis Goeckel. Fundamental limits of activity-based covert channels. pages 1–6, 2021.
- [31] Xiaomin Liao, Jiangbo Si, Jia Shi, Zan Li, and Haiyang Ding. Generative adversarial network assisted power allocation for cooperative cognitive covert communication system. *IEEE Communications Letters*, 24(7):1463–1467, 2020.
- [32] Ashok V Makkuva, Xiyang Liu, Mohammad Vahid Jamali, Hessam Mahdaviifar, Sewoong Oh, and Pramod Viswanath. Ko codes: inventing nonlinear encoding and decoding for reliable wireless communication via deep-learning. In *International Conference on Machine Learning*, pages 7368–7378. PMLR, 2021.
- [33] Alejandro Martín, Alfonso Hernández, Moutaz Alazab, Jason Jung, and David Camacho. Evolving generative adversarial networks to improve image steganography. *Expert Systems with Applications*, page 119841, 2023.
- [34] J McGinn, Chris Messenger, Michael Williams, and I Heng. Generalised gravitational wave burst generation with generative adversarial networks. *Classical and Quantum Gravity*, 38, 08 2021.
- [35] Jonathan Millen. 20 years of covert channel modeling and analysis. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No. 99CB36344)*, pages 113–114. IEEE, 1999.
- [36] Hesham Mohammed, Xue Wei, and Dola Saha. Adversarial learning for hiding wireless signals. pages 01–06, 2021.
- [37] Lucas Nussbaum, Pierre Neyron, and Olivier Richard. On robust covert channels inside dns. pages 51–62, 2009.
- [38] Timothy J O’shea and Nathan West. Radio machine learning dataset generation with gnu radio. In *Proceedings of the GNU Radio Conference*, volume 1, 2016.
- [39] Timothy O’shea and Jakob Hoydis. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):563–575, 2017.

- [40] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.
- [41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- [42] Yinlong Qian, Jing Dong, Wei Wang, and Tieniu Tan. Deep learning for steganalysis via convolutional neural networks. 9409:171–180, 2015.
- [43] Jiaohua Qin, Jing Wang, Yun Tan, Huajun Huang, Xuyu Xiang, and Zhibin He. Coverless image steganography based on generative adversarial network. *Mathematics*, 8:1394, 08 2020.
- [44] Brecht Reynders and Sofie Pollin. Chirp spread spectrum as a modulation technique for long range communication. pages 1–5, 2016.
- [45] Debashri Roy, Tathagata Mukherjee, Mainak Chatterjee, Erik Blasch, and Eduardo Pasilliao. Rfal: Adversarial learning for rf transmitter identification and classification. *IEEE Transactions on Cognitive Communications and Networking*, 6(2):783–801, 2019.
- [46] Meysam Sadeghi and Erik G Larsson. Physical adversarial attacks against end-to-end autoencoder communication systems. *IEEE Communications Letters*, 23(5):847–850, 2019.
- [47] Kunal Sankhe, Francesco Restuccia, Salvatore D’Oro, Tong Jian, Zifeng Wang, Amani Al-Shawabka, Jennifer Dy, Tommaso Melodia, Stratis Ioannidis, and Kaushik Chowdhury. Impairment shift keying: Covert signaling by deep learning of controlled radio imperfections. pages 598–603, 2019.
- [48] Robert Scholtz. The origins of spread-spectrum communications. *IEEE transactions on Communications*, 30(5):822–854, 1982.
- [49] Azadeh Sheikholeslami, Majid Ghaderi, and Dennis Goeckel. Covert communications in multi-channel slotted aloha systems. *IEEE Transactions on Mobile Computing*, 2020.
- [50] Azadeh Sheikholeslami, Majid Ghaderi, Don Towsley, Boulat A Bash, Saikat Guha, and Dennis Goeckel. Multi-hop routing in covert wireless networks. *IEEE Transactions on Wireless Communications*, 17(6):3656–3669, 2018.
- [51] Yi Shi, Kemal Davaslioglu, and Yalin E Sagduyu. Generative adversarial network in the air: Deep adversarial learning for wireless signal spoofing. *IEEE Transactions on Cognitive Communications and Networking*, 7(1):294–303, 2020.
- [52] Tamara V Sobers, Boulat A Bash, Saikat Guha, Don Towsley, and Dennis Goeckel. Covert communication in the presence of an uninformed jammer. *IEEE Transactions on Wireless Communications*, 16(9):6193–6206, 2017.

- [53] Mehran Soltani, Vahid Pourahmadi, Ali Mirzaei, and Hamid Sheikhzadeh. Deep learning-based channel estimation. *IEEE Communications Letters*, 23(4):652–655, 2019.
- [54] Ramin Soltani, Dennis Goeckel, Don Towsley, Boulat A Bash, and Saikat Guha. Covert wireless communication with artificial noise generation. *IEEE Transactions on Wireless Communications*, 17(11):7252–7267, 2018.
- [55] Shunquan Tan and Bin Li. Stacked convolutional auto-encoders for steganalysis of digital images. pages 1–4, 2014.
- [56] Denis Volkhonskiy, Ivan Nazarov, and Evgeny Burnaev. Steganographic generative adversarial networks. 11433:114333M, 2020.
- [57] Jean Walrand and Pravin Varaiya. Chapter 7 - wireless networks. In Jean Walrand and Pravin Varaiya, editors, *High-Performance Communication Networks (Second Edition)*, pages 305–361. Morgan Kaufmann, San Francisco, second edition edition, 2000.
- [58] Tianqi Wang, Chao-Kai Wen, Hanqing Wang, Feifei Gao, Tao Jiang, and Shi Jin. Deep learning for wireless physical layer: Opportunities and challenges. *China Communications*, 14(11):92–111, 2017.
- [59] Dehao Wu, Maziar Nekovee, and Yue Wang. Deep learning-based autoencoder for m-user wireless interference channel physical layer design. *IEEE Access*, 8:174679–174691, 2020.
- [60] Guanshuo Xu, Han-Zhou Wu, and Yun-Qing Shi. Structural design of convolutional neural networks for steganalysis. *IEEE Signal Processing Letters*, 23(5):708–712, 2016.
- [61] Shihao Yan, Xiangyun Zhou, Jinsong Hu, and Stephen V Hanly. Low probability of detection communication: Opportunities and challenges. *IEEE Wireless Communications*, 26(5):19–25, 2019.
- [62] Sebastian Zander, Grenville Armitage, and Philip Branch. A survey of covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys & Tutorials*, 9(3):44–57, 2007.
- [63] Chaoning Zhang, Philipp Benz, Adil Karjauv, Geng Sun, and In So Kweon. Udh: Universal deep hiding for steganography, watermarking, and light field messaging. *Advances in Neural Information Processing Systems*, 33:10223–10234, 2020.
- [64] Chaoning Zhang, Chenguo Lin, Philipp Benz, Kejiang Chen, Weiming Zhang, and In So Kweon. A brief survey on deep learning based data hiding, steganography and watermarking. *arXiv preprint arXiv:2103.01607*, 2021.
- [65] Xiangyun Zhou, Lingyang Song, and Yan Zhang. *Physical layer security in wireless communications*. Crc Press, 2013.
- [66] Guangxu Zhu, Dongzhu Liu, Yuqing Du, Changsheng You, Jun Zhang, and Kaibin Huang. Toward an intelligent edge: Wireless communication meets machine learning. *IEEE communications magazine*, 58(1):19–25, 2020.

- [67] Cong Zou, Fang Yang, Jian Song, and Zhu Han. Channel autoencoder for wireless communication: State of the art, challenges, and trends. *IEEE Communications Magazine*, 59(5):136–142, 2021.