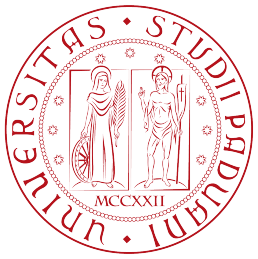


University of Padova
Department of Information Engineering
Master's degree in ICT for Internet and Multimedia

An Adversarial Learning Framework for Privacy Preserving Communications

Supervisor: Prof. Nicola Laurenti
Co-supervisor: Prof. Deniz Gunduz

Student: Thomas Marchioro
ID 1184445

Padova, 2019
ACADEMIC YEAR 2018-2019

Acknowledgements

Since my university studies have been long and harsh, there are a lot of people that I have to acknowledge and I will inevitably forget someone, so I apologize in advance.
First of all, I would like to thank my parents and my sister, who have not kicked me out yet and who have supported me both morally and economically.
I also would like to thank my supervisor, professor Nicola Laurenti, who has endured me since my bachelor's studies and who has been the best supervisor I could ask for. I thank professor Deniz Gunduz, who supervised my work along with Nicola and who has provided excellent ideas for the realization of this thesis.
Moreover, I have to thank professor Leonardo Badia, who helped me in different contexts and allowed me to find the PhD project to which I will participate for the next years.
I would like to thank all the other professors of the department, who taught me a lot more than what I would have been able to learn by myself.
Dulcis in fundo, I thank my friends from the University (in particular Alice, Davide, Gabriella, Gaia, Pier Angelo and Virginia), without whom I probably would not have finished my studies, and my old friends (unfortunately I cannot list them all) who are always there for me, despite of all the times I have bailed on them.
Really, thank you all.

Abstract

We develop a machine learning-based approach that allows to achieve privacy in communications by exploiting an advantage at the physical layer. Our goal is to transmit useful data to the intended receiver while preventing sensitive data from leaking to an eavesdropper who has access to the channel. We adopt an adversarial approach involving two competing neural networks to learn efficient coding schemes that allow to regulate the tradeoff between quality and privacy.

Contents

Chapter 1

Introduction

Over the years, physical layer secrecy and related topics kept gaining popularity among the main conferences and journals concerning communications and information theory. The main reason is that physical layer secrecy has the potential to provide confidentiality in the transmission without using cryptographic methods, and hence without the need of relying on some secret key. In 1975, in fact, an article from A. Wyner [1] has shown from an information theoretic point of view that it is possible to achieve secrecy exploiting physical properties of the communication channel, which may be, as discussed by Leung and Hellman in [2], an advantage in terms of signal-to-noise ratio with respect to an eavesdropper who has access to such channel. The physical advantage is exploited by means of suitably chosen coding scheme that make impossible for the adversary to distinguish the codewords. The efficiency and the secrecy of the transmission depend on the choice of the code, as well as from the channel properties. It has been shown in [16] that artificial neural networks can be used to learn efficient schemes that provide physical layer secrecy and that are robust to noise variation. Moreover, neural network-based frameworks provide the ability of coping on-the-fly with changes in the channel scenario.
Our work addresses a problem which is strongly related to physical layer secrecy: we aim to achieve efficient transmission of useful information along a physical channel while preventing sensitive information leaking to an eventual adversary, who acts as an eavesdropper. The goal of releasing data that enable useful information diffusion while keeping sensitive information secret is known is literature as *privacy preservation*. Such problem has been widely studied in contexts where no transmission is involved [18] and thus a legitimate user and a malicious attacker are able to observe the same data, which is created by some release mechanism. Also in this case, adversarial neural networks have been proven to be a versatile tool allowing to learn release mechanisms that can regulate the tradeoff between distortion of the useful

data and privacy of the sensitive data. The transmission along a noisy channel creates an opportunity to increase the privacy, in case of an advantage of the legitimate user with respect to the eavesdropper. We addressed the problem first from an information theoretic point of view and analyze its connection to physical layer secrecy. Secondly, we try to achieve privacy preserving communications of some data exploiting adversarial neural networks. Since we analyze the transmission of actual data instead of symbols, we adopt a joint source and channel coding learning framework.

1.1 Notation

In this work, we make a wide use of concepts from probability and information theory, therefore in this section we briefly describe the notation that we use.

Vectors and matrices We write both scalars and vectors as lower case characters, the difference will be clear by the context or eventually specified when necessary. The i-th component of a vector x is denoted by $x^{(i)}$. Matrices are written as bold capital letters, in order to distinguish them from random variables.

Probability A random variable (r.v.) X is written with a capital letter. Its alphabet is denoted with $\mathcal{X}$ and it is always assumed to be discrete. A generic realization x of the random variable X is denoted with the corresponding lower case letter. The probability mass distribution (PMD) of a random variable X will be simply called distribution and denoted with $p_X(x)$, which is defined as $p_X(x) = \Pr[X = x], x \in \mathcal{X}$. When it is important to highlight the length of a vector of n random variables, such vector is denoted with X^n and the same is done for its realization x^n. A sequence $X_0, X_1, \ldots, X_n$ of random variables that satisfies the Markov property

$$\Pr[X_k|X_0, \ldots, X_{k-1}] = \Pr[X_k|X_{k-1}],\ \forall k = 1, \ldots, n$$

is called *Markov chain* and is denoted with $X_0 \to X_1 \to \cdots \to X_n$.

Information theory The expectation of a function $g(X)$ applied to X is written $\mathbb{E}[g(X)]$ and is defined as $\sum_{x\in\mathcal{X}} p_X(x)g(x)$.
The *information* provided from a certain realization x of X is

$$I(x) = \log \frac{1}{p_X(x)},\ x \in \mathcal{X},$$

where the logarithm is considered base 2 when the base is omitted. The *entropy* $H(X)$ is the expected value of the information function, i.e., $H(X) = \mathbb{E}[I(X)]$.
The *conditional entropy* of the r.v. Y given X is defined as

$$\begin{aligned} H(Y|X) &= \mathbb{E}\left[\log \frac{1}{p_{Y|X}(Y|X)}\right] \\ &= \sum_{x\in\mathcal{X}, y\in\mathcal{Y}} p_{XY}(x,y) \log \frac{1}{p_{Y|X}(y|x)} \\ &= \sum_{x\in\mathcal{X}} p_X(x) \sum_{y\in\mathcal{Y}} p_{Y|X}(y|x) \log \frac{1}{p_{Y|X}(y|x)}, \end{aligned}$$

where p_{XY} is the joint distribution of X and Y, and $p_{Y|X}$ is the conditional distribution of Y given X. The *mutual information* between two random variables is defined as

$$I(X,Y) = \mathbb{E}\left[\log \frac{p_{XY}(X,Y)}{p_X(X)p_Y(Y)}\right] = \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_{XY}(x,y) \log \frac{p_{XY}(x,y)}{p_X(x)p_Y(y)}.$$

and one remarkable property that we use in our calculations is that $I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$.
Given two distributions p_X and q of the same random variable X, the *Kullback-Leibler divergence* is defined as

$$\mathbb{D}(p_X \| q) = \mathbb{E}_{X\sim p_X}\left[\log \frac{p_X(X)}{q(X)}\right] = \sum_{x\in\mathcal{X}} p_X(x) \log \frac{p_X(x)}{q(x)}.$$

The *cross-entropy* between the distributions is defined as

$$H(p_X, q) = \mathbb{E}_{X\sim p_X}\left[\log \frac{1}{q(X)}\right] = \sum_{x\in\mathcal{X}} p_X(x) \log \frac{1}{q(x)}.$$

but can also be expressed as

$$H(p_X, q) = H(X) + \mathbb{D}(p_X \| q).$$

In this work, we will always assume that a random variable X is uniquely identified by its true distribution p_X. However, in the following chapters we will need to evaluate the cross-entropy between the true distribution p_X of X and other possible distributions of the same random variable.

Chapter 2

Learning framework for the physical layer

In this chapter, we show how physical layer communication scenarios can be modeled using artificial neural networks before going deep into the applications to privacy preservation. We consider a basic end-to-end communication in which the transmitter needs to send a message $s \in \mathcal{M}$ to the receiver over an AWGN channel. At the transmitter side, the message s is encoded into the codeword $x \in \mathcal{X}^n$, with $\mathcal{X} = \mathbb{R}$. The codeword is then transmitted along the noisy channel, that introduces additive Gaussian noise $N \sim \mathcal{N}(0, \sigma_B^2)$ and thus the receiver obtains the vector y, where $y_i = x_i + N$ [1]. At the receiver side, y is decoded into the message $\hat{s}$ and the transmission is successful if $\hat{s} = s$.

2.1 Autoencoders

An autoencoder is a type of artificial neural network that turns out to be particularly handy to model communication scenarios. Autoencoders are aimed to learn a representation for a specific set of data, that for a given input allows to reconstruct in output some data that are as close[2] as possible to the input, even if some noise or compression is introduced in the hidden layers. It is natural to think of using

[1] This notation means that to each component x_i an independent realization of N is added.

[2] Of course, "close"is a generic term and for a specific autoencoder it is necessary to specify a distance function between input and output that needs to be minimized.

this kind of networks to model communications over noisy channel: the input and output layer can be considered as the transmitter and receiver side, respectively, of a communication channel, and the intermediate layers represent the encoder, the noisy channel and the decoder. In the next paragraphs, we introduce some building blocks for the construction of neural networks that are often employed in the design of autoencoders learning end-to-end coding schemes.

One-hot encoding Suppose we have ℓ distinct symbols that we can send along the channel. The first step that is usually performed on such symbols is one-hot encoding. Assuming that among the available symbols, that are numbered from 0 to $\ell - 1$, the j-th symbol is chosen to be transmitted, the one-hot encoding of such symbol is a vector e_j of length ℓ, whose components are defined as

$$e_j^{(i)} = \begin{cases} 1, \text{ if } i = j \\ 0, \text{ if } i \neq j, \end{cases}$$

meaning that the vector is made of all zeros, with the exception of a 1 in the j-th position. The motivation behind this operation will be more clear in the next paragraph, but the main idea is that the one-hot encoding vector e_j can be considered as a pseudo-distribution, where the probability of the symbol being the j-th symbol is 1, and it is 0 for all the other symbols. The autoencoder is trained in order to get an output vector which represents a distribution that must be as close as possible to the one-hot encoding of the input.

Dense layers The most commonly used type of layers, along with the convolutional layers, are the dense layers. A layer is called *dense* or *fully-connected* when the relation between two consecutive hidden layers is given by the concatenation of an affine function and a non-linear activation function. If v_{in} is the input of the dense layer, first the affine function is applied and the output is

$$z = \mathbf{W} v_{in} + b, \tag{2.1}$$

where $\mathbf{W}$ is a matrix whose elements $w^{(i,j)}$ are called *weights* and b is a vector whose elements $b^{(i)}$ are called *biases*. Then, a non-linear activation function σ is applied to the output z of the affine function. The final output of the dense layer is thus

$$v_{out} = \sigma(z) = \sigma(\mathbf{W} v_{in} + b). \tag{2.2}$$

The weights and the biases are tuned during the training phase, while the activation function is usually fixed or contains only few parameters. The activation function is non-linear because concatenating multiple linear transformation leads again to a linear transformation and this would make pointless to use multiple layers.
The name "fully-connected"derives from the fact that each element (*neuron*) $v_{out}^{(i)}$ of the output layer is obtained as a function of z, which is a linear combination of all the neurons $v_{in}^{(j)}$ of the input layer. The weight $w^{(i,j)}$ determine how much the i-th neuron of the input layer affects the j-th neuron of the output layer and since the weights are trained by the neural network it is rare for them to be exactly zero.

Convolutional layers The main components of the so-called *convolutional neural networks* are the convolutional layers. A convolutional layer differs from a dense layer from the fact that in this type of layers not all the neurons are connected. Moreover, while dense layers usually have flat input and output vectors (i.e., one-dimensional), convolutional layers have bidimensional input and output matrices and can be used, for instance, to process images or video frames. The operation performed by such layers is the 2-D convolution, which employs a window filter with a certain dimension $M \times N$ as follows:

$$v_{out}^{(i,j)} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} w^{(m,n)} v_{in}^{(s_1 \cdot i+m, s_2 \cdot j+n)}. \tag{2.3}$$

The parameters s_1 and s_2 are called *strides* and need to be adjusted when the neural network structure is designed, as well as the size of the filter, which is often called *kernel*. The weights are learned during the training phase, as for the dense layers.

2.1.1 Activation functions

There are several possible activation functions that can be applied after the affine transformation of a dense layer. The main idea is that the activation function should introduce some non-linearity, but should also be differentiable and smooth.

ReLU The *rectifier linear unit* (ReLU) is an activation function that is applied independently to each component $z^{(i)}$ of the input vector z, according to the rule

$$\sigma_{\text{ReLU}}(z^{(i)}) = \begin{cases} z^{(i)}, & \text{if } z^{(i)} \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

which can be also written more compactly as $\sigma_{\text{ReLU}}(z^{(i)}) = \max(0, z^{(i)})$. This kind of function is mostly used in networks with a high number of hidden layers. The main

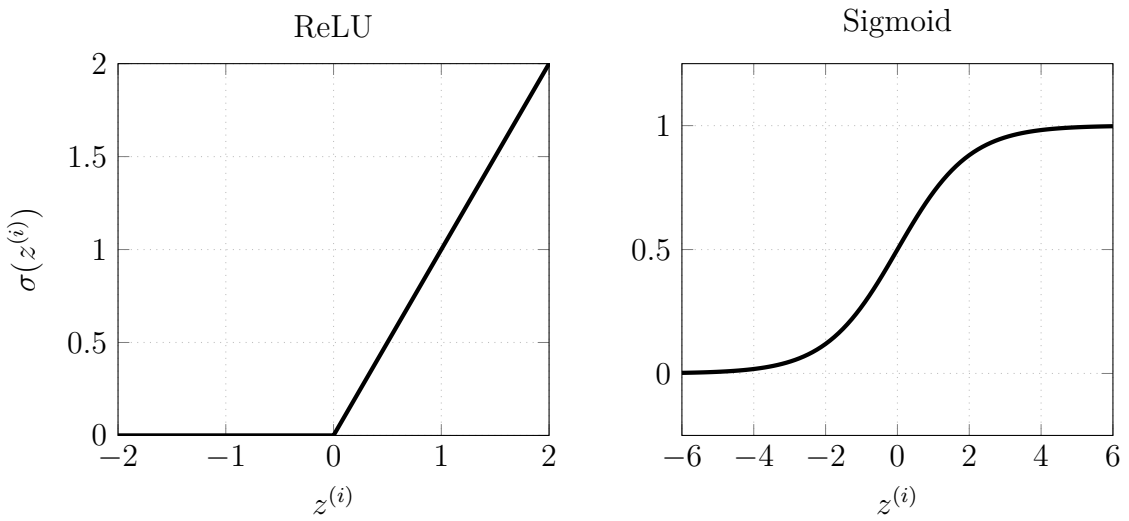


Figure 2.1: The ReLU and sigmoid activation functions.

motivation is that the function is the identity when the value of $z^{(i)}$ is non-negative and thus overcomes the problem of the vanishing gradient which is introduced by applying other activation functions (e.g., hyperbolic tangent or sigmoid) for a high number of hidden layers.

PReLU The *parametric ReLU* (PReLU) is a variation of the ReLU activation function which is defined as

$$\sigma_{\text{PReLU}}(z^{(i)}) = \begin{cases} z^{(i)}, & \text{if } z^{(i)} \geq 0 \\ \alpha z^{(i)}, & \text{otherwise} \end{cases} \tag{2.5}$$

where $\alpha > 0$ is a parameter that is tuned during the training phase along with the other parameters.

Sigmoid The *sigmoid* activation function is defined for each element $z^{(i)}$ of the input as

$$\sigma_{\text{sigmoid}}(z^{(i)}) = \frac{1}{1 + e^{-z^{(i)}}}. \tag{2.6}$$

The output of the sigmoid is a vector whose components are bounded in the range $(0, 1)$ and for this reason using this activation function on multiple layers tends to make the gradient vanish during the backpropagation. Therefore, the sigmoid is usually employed only in the last layer, while in the intermediate layer ReLU and PReLU are more often used. On the other hand, the output of the sigmoid can be interpreted as a probability measure of an element being or not in a certain class and can hence be used for logistic purposes.

Softmax Another activation function that is commonly used for autoencoders is the *softmax*, that given an input z returns a vector whose components are defined by

$$\sigma^{(i)}_{\text{softmax}}(z) = \frac{e^{z^{(i)}}}{\sum_{j=0}^{\ell-1} e^{z^{(j)}}}. \tag{2.7}$$

Assuming v_{out} is the output of the softmax, its components $v^{(0)}_{out}, \ldots, v^{(\ell-1)}_{out}$ are bounded in the range $(0, 1)$ and also satisfy

$$\sum_{i=0}^{\ell-1} v^{(i)}_{out} = 1,$$

meaning that its output can be interpreted as a vector of likelihoods which describe a PMD over the alphabet $\{0, \ldots, \ell - 1\}$ (or any alphabet of cardinality ℓ).

2.1.2 Cross-entropy minimization

One of the most common techniques employed in neural networks to solve multiclass classification tasks is cross-entropy minimization. The main idea is to minimize the average cross-entropy between the one-hot encoded class of the input of the neural network and the output of the last layer of the neural network, that in this case must use the softmax as activation function. The motivation behind this operation is that if we consider the class of the input as a random variable S, with finite alphabet of cardinality ℓ, the one hot encoding of such r.v. is

$$\begin{cases} 1, \text{ if } S = j \\ 0, \text{ if } S \neq j, \end{cases}$$

and hence represents the pseudo-distribution $p_{S|S}$. On the other hand, the output of the softmax can represent a probability distribution, and since it is the results of operations which involve the input, such distribution can be interpreted as a vector of likelihoods of S. Therefore, the output of the softmax represents an estimate distribution q and performing the cross-entropy minimization using a given training set with labels $s_0, \ldots, s_{m-1}, s_i \in \{0, \ldots, \ell - 1\}$ of size m means minimizing

$$\frac{1}{m} \sum_{i=0}^{m-1} H(p_{S|S}, q|S = s_i) = \frac{1}{m} \sum_{i=0}^{m-1} \sum_{j=0}^{\ell-1} p_{S|S}(j|s_i) \log \frac{1}{q(j|s_i)}. \tag{2.8}$$

From the strong law of large numbers, it holds

$$\Pr\left[\lim_{m\to\infty}\frac{1}{m}\sum_{i=0}^{m-1}H(p_{S|S},q|S=s_i)=H(p_{S|S},q)\right]=1, \tag{2.9}$$

meaning that when m becomes large, the minimization of the average cross-entropy over the training set is equivalent to the minimization of the actual cross-entropy between the distributions $p_{S|S}$ and q. When the structure of the neural network allows to reduce significantly the cross-entropy – taking the Kullback-Leibler divergence close to zero – the distribution q gets close to $p_{S|S}$ and thus a good estimation $\hat{S}$ of the true class S for a given input consists in taking the argmax of the output of the softmax, i.e.,

$$\arg\max_{j\in\{0,\dots,\ell-1\}} q^{(j)}. \tag{2.10}$$

For example, if there are four classes $\{0,1,2,3\}$ and the output of the softmax is $[0.1,0.8,0.08,0.02]^\top$, the best estimation of the class is 1.

2.1.3 Stochastic optimization

Artificial neural networks usually have lots of parameters that must be trained and hence require to be trained using a huge amount of data. Nevertheless, computing the gradient of the loss function employing all the data at every iteration of some gradient-based optimization method has a huge computational cost. This is why it is common to compute an approximation function and the relative gradient using only a small portion of the available data, which takes the name of *mini-batch*. The approximate loss function $\mathcal{L}(\theta_t,\xi_t)$ computed at time t[3] over a mini-batch ξ_t using the parameters θ_t obtained at the previous iteration, can be seen as a stochastic realization of a random function, which depends on ξ_t, which can be considered as the realization of a random variable Ξ_t. The mean value of such random function is

$$\mathbb{E}\left[\mathcal{L}(\theta_t,\Xi_t)\right]=\hat{\mathcal{L}}(\theta_t) \tag{2.11}$$

where $\hat{\mathcal{L}}$ is the loss function computed on all the available training set. When the batch size is chosen too small, the variance of the random outcomes might becomes too high and provides unstable updates which do not lead to the optimal solution. On the other hand, if the batch size is too large, the computational cost becomes huge and the optimization process gets slow. That is why the batch size is an important parameter that must be adequately chosen when training a neural network.

[3]Time is intended as iteration number and is discrete.

Stochastic gradient descent One of the simplest and most popular stochastic optimization methods is called *stochastic gradient descent* (SGD) and consists in simply applying a gredient based method – either the standard gradient or Nesterov's method – to the randomized function computed on the mini-batch. The complete algorithm employing the standard gradient method is reported below.

Algorithm 1 Stochastic gradient descent

Input: the objective function $\mathcal{L}$, the training set $\mathcal{T}$

1: Choose the starting parameters θ_0
2: **for** $t = 0, 1, \ldots$ **do**
3: **if** θ_t satisfies some specific condition **then** STOP
4: **end if**
5: Sample the mini-batch ξ_t from $\mathcal{T}$
6: Set $\theta_{t+1} = \theta_t - \alpha_t \nabla \mathcal{L}(\theta_t, \xi_t)$, with α_t suitably chosen stepsize
7: **end for**

The stepsize α_t must be suitably chosen according to the properties of the function and must be always a diminishing stepsize in order to guarantee convergence. Unfortunately, it is rare for neural network loss functions to have properties such as strong convexity or Lipschitz continuous gradient, for which it is easy to define a stepsize that guarantees convergence within a certain number of iterations. The stepsize is usually parametrized by a constant $\eta > 0$, which is called *learining rate* and determines how large are the initial update steps. Again, the learning rate must be chosen carefully: a high learning rate leads to unstable gradient updates, while a low learning rate causes slow convergence.
One last problem that must be highlighted is the choice of the starting parameters. Usually the initial parameter vector θ_0 is chosen at random and in case function $\mathcal{L}$ is convex with respect to the parameters θ this is not a problem. However, for large neural networks, the objective function usually is not convex and thus the optimization process might lead to local optima. In order to avoid this problem, the optimization algorithm should be run multiple times using different starting parameters.

Adam Adaptive Moment Estimation (Adam) is probably the most popular optimization algorithm in the context of neural networks and it is used in particular for convolutional neural networks and other learning frameworks that involve functions having sparse gradient. A sparse gradient means that most of the components of the gradient are zero or close to zero. Since each component of the gradient is associated to one parameter of the neural network, functions having a sparse gradient have some weights that are updated more frequently than others. Adam employs

an adaptive learning rate which is different for each parameter and that provides larger steps for the weights that are updated less frequently.

Algorithm 2 Adam

Input: the objective function $\mathcal{L}$, the training set $\mathcal{T}$, the learning rate $\eta > 0$, the decay rates for the moment estimates $\beta_1, \beta_2 \in (0, 1)$, the constant $\varepsilon > 0$

Choose the starting parameters θ_0
Initialize $m_0 = 0, v_0 = 0, \hat{m}_0 = 0, \hat{v}_0 = 0$
for $t = 0, 1, \ldots$ **do**
    if θ_t satisfies some specific condition **then** STOP
    end if
    Sample the mini-batch ξ_t from $\mathcal{T}$
    Set $g_t = \nabla\mathcal{L}(\theta_t, \xi_t)$
    Set $m_{t+1} = \beta_1 m_t + (1 - \beta_1) g_t$
    Set $v_{t+1} = \beta_2 v_t + (1 - \beta_2) g_t^2$, where g_t^2 denotes the elementwise square
    Set $\hat{m}_{t+1} = m_{t+1}/(1 - \beta_1^t)$
    Set $\hat{v}_{t+1} = v_{t+1}/(1 - \beta_2^t)$
    Set $\theta_{t+1} = \theta_t - \eta \hat{m}_{t+1}/(\sqrt{\hat{v}_{t+1}} + \varepsilon)$
end for

The hyperparameters β_1, β_2 regulate the exponential decay rates for the first and second moment estimates, respectively, while ε is a small constant which is added to the denominator of the update term to guarantee numerical stability. In this work, these hyperparameters are set to their default values $\beta_1 = 0.9, \beta_2 = 0.99$ and $\varepsilon = 10^{-8}$, which are suggested in the original paper. The learning rate η, instead, cannot be fixed to the default vaue, but must be tuned depending on the complexity of the neural network.

2.2 Design of a simple autoencoder

In order to clarify how the tools introduced in the previous section are employed, we describe the design of a simple autoencoder which learns an end-to-end coding scheme that combats the noise introduced by an additive Gaussian channel. The neural network model is structured in three main blocks: the encoder, the channel and the decoder.

Encoder First of all, the input message $s \in \mathcal{M} = \{0, \ldots, \ell - 1\}$ is one-hot encoded into $e_s \in \{0, 1\}^\ell$. Then, e_s is processed by two dense layers:

- the first layer applies a linear transformation $\{0, 1\}^\ell \to \mathbb{R}^\ell$ and employs a ReLU as activation function;

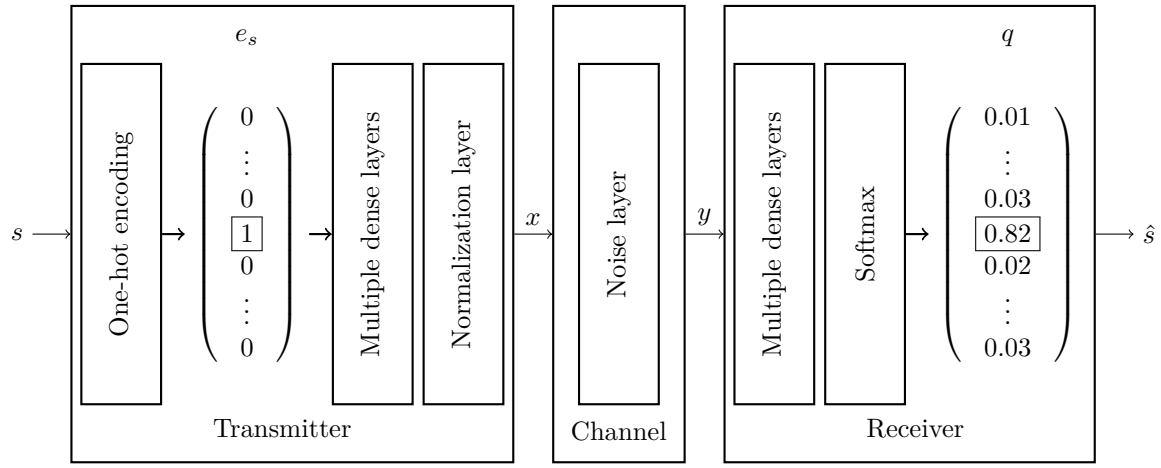


Figure 2.2: Structure of a simple autoencoder that learns a coding scheme for transmission over a noisy channel.

- the second layer applies a linear transformation $\mathbb{R}^\ell \to \mathbb{R}^n$ and performs an ℓ_2-normalization of the output to model the constraints to the energy of the symbols.

The obtained codeword is $x \in \mathbb{R}^n$, that is subject to the power normalization constraint $\|x\|_2 = 1$.

Channel In order to model the transmission along the Gaussian channel, the codeword must be processed by a noise layer, that generates a Gaussian random vector of n i.i.d. components that follow the distribution $\mathcal{N}(0, \frac{1}{\Lambda_B})$ and adds them to x. The variance σ_B^2 of the channel is the reciprocal of the SNR at the receiver side Λ_B an accounts also an eventual gain applied at the transmitter side.

Decoder The decoder receives in input the noisy vector $y \in \mathbb{R}^n$ and returns as output the decoded message $\hat{s} \in \mathcal{M}$. The decoder is also made of two dense layers:

- the first layer applies a linear transformation $\mathbb{R}^n \to \mathbb{R}^\ell$ and the ReLU;
- the second layer applies a linear transformation $\mathbb{R}^\ell \to \mathbb{R}^\ell$ followed by the softmax to compute $q \in [0,1]^\ell$.

The ℓ components of q represent the likelihoods of the symbols $0, \dots, \ell - 1$ being the actual input message. Thus, the natural choice of the output of the decoder is

$$\hat{s} = \arg \max_{j \in \{0,\dots,\ell-1\}} q^{(j)}.$$

Table 2.1: Parameters used for the simulation

Parameter	Symbol	Value
Number of symbols	ℓ	4
Codeword size	n	2
Receiver's SNR	Λ_B	12 dB
Learning rate	η	0.01
Size of training set	m_{train}	3000
Size of test set	m_{test}	1000

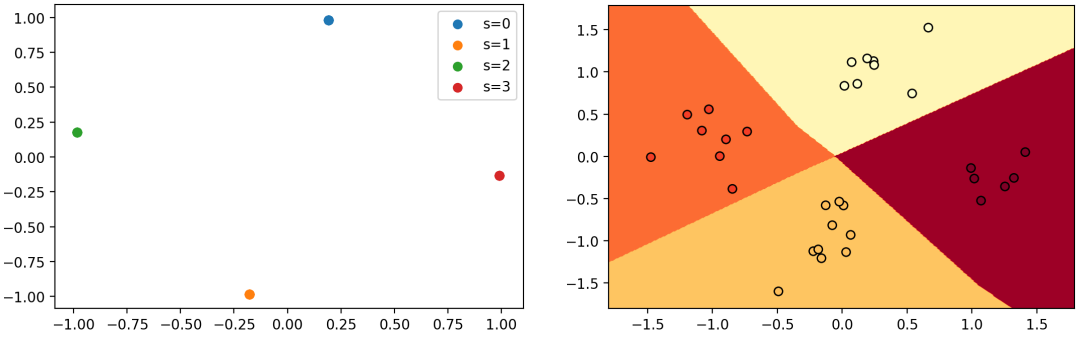


Figure 2.3: The encoding rule and the decision regions obtained by training the neural network on 3000 examples with $\eta = 0.01$.

The parameters θ of the neural network that need to be estimated are the coefficients of the linear functions in the dense layers. The cost function $\mathcal{L}$ to be minimized is the average cross-entropy between the one-hot encoded message e_s and the output of the softmax q.

2.2.1 Training results for a simple autoencoder

As an example, we trained an autoencoder with TensorFlow using $\ell = 4$ symbols and codewords of length $n = 2$, so that the decision regions can be visualized on a bidimensional plane. The parameters used for the training are reported in Table 2.1. We used 3000 examples for the training phase and measured the accuracy, i.e. the fraction of correctly decoded messages, on a test set of size 1000. We obtained, even with such a small training set, that more than 99% of the messages have been correctly classified and both the encoding rule. The decision regions are shown in figure 2.3.

Confusion matrix The confusion matrix is a useful tool that allows to visualize the accuracy of a predictor on a test set. The rows of the matrix represent a predicted label for an example of the test set, while the columns are the actual labels and in the context of our simple autoencoder the rows are the output messages while the

columns are the input messages. The performance of the decoder is hence higher when the confusion matrix is close to a diagonal matrix and the nonzero terms out of the diagonal represent which messages are mostly confused (e.g., if the term $\mathbf{C}^{(i,j)}$ if the matrix is high, this means that the message i is often incorrectly decoded into the message j). In figure 2.4, we inserted a visual representation of the confusion matrix of our simple autoencoder, where darker shades of blue represent higher values of the elements of the matrix.

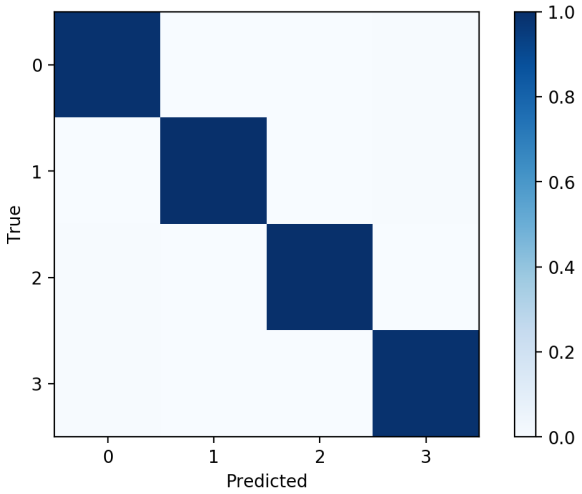


Figure 2.4: The confusion matrix computed on a test set of 1000 samples.

Chapter 3

Physical layer secrecy

Secrecy has always been of vital importance for communication systems, in particular when sensitive information needs to be transmitted. Traditionally, secrecy is achieved by means of cryptographic key-based ciphering mechanisms. In this chapter, we show how the inherent randomness present in physical channels can be exploited to achieve secure communications. The confidentiality obtained from the physical properties of the channel is known in literature with the name of *physical layer secrecy*, because it is handled at the physical layer, differently from encryption algorithm, which are applied in the upper layers.

3.1 The wiretap channel

A general model which represents a communication scenario in presence of an eavesdropper is the wiretap channel, originally introduced by Wyner [1]. The wiretap channel describes the communication from an information theoretical point of view. Three main actors are involved in the model: a transmitter (A), a receiver (B) and an eavesdropper (E). A wants to send some message S to B, while E is eavesdropping to the channel. In order to transmit the message, A encodes it into the codeword X^n, of length n, using a mechanism characterized by the conditional distribution $p_{X^n|S^n}$. The encoded message is transmitted along a channel, which is defined by the distribution $p_{Y^nZ^n|X^n}$, and thus B and E end up receiving Y^n and Z^n respectively. In our discussion, we always refer to a memoryless channel, i.e., such that

$$p_{Y^nZ^n|X^n}(y^nz^n|x^n) = \prod_{i=0}^{n-1} p_{YZ|X}(y^{(i)}, z^{(i)}|x^{(i)}), \tag{3.1}$$

meaning that the noise applied to each symbol is independent from the others. When the channel is memoryless, it is uniquely identified by the conditional distribution

per symbol $p_{YZ|X}$. The legitimate receiver B applies the decoding function f_B to Y^n, obtaining $\hat{S}$. The eavesdropper E, instead, applies the decoding function f_E to Z^n, obtaining $\tilde{S}$. Both B and E aim to acquire a perfect copy of S. Moreover, A and B aim to have E not being able to recover the original message, i.e., $\tilde{S} \neq S$. Since they cannot change the decision rule f_E, the only way for them to reduce the probability of E correctly decoding the original message S is achieving independence between S and the codeword Z^n leaked to E.

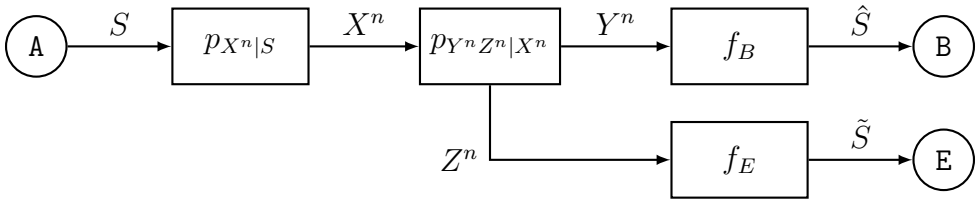


Figure 3.1: Wyner's wiretap channel model.

3.1.1 Secrecy condition

The condition that is used in the context of cryptography to guarantee independence between a plaintext S and the corresponding encoded message Z^n is the *perfect secrecy* condition, which requires

$$H(S|Z^n) = H(S), \text{ or, equivalently, } I(S, Z^n) = 0 \tag{3.2}$$

that intuitively means that no information on S can be obtained from Z^n. Such condition, however, is too stringent to apply for physical layer secrecy between the message S and the codeword Z^n received by the eavesdropper. A more realistic requirement is to achieve asymptotic independence between S and Z^n when the length n of the codeword goes to infinity. This asymptotic independence can be measured by a distance measure between the joint distribution p_{UZ^n} and the product of the marginal distributions $p_S p_{Z^n}$. If Kullback-Leibler divergence is chosen as distance measure, the condition becomes

$$\lim_{n\to\infty} \mathbb{D}(p_{UZ^n} \| p_S p_{Z^n}) = 0. \tag{3.3}$$

Applying the definitions of Kullback-Leibler divergence and mutual information, it holds

$$\mathbb{D}(p_{UZ^n} \| p_S p_{Z^n}) = \sum_{S\in\mathcal{M}} \sum_{z^n\in\mathcal{Z}^n} p_{UZ^n}(S, z^n) \log \frac{p_{UZ^n}(S, z^n)}{p_S(S) p_{Z^n}(z^n)} = I(S, Z^n)$$

and thus the requirement can be rewritten as

$$\lim_{n\to\infty} I(S, Z^n) = 0. \tag{3.4}$$

This condition is the asymptotic equivalent of the perfect secrecy condition and is called *strong secrecy* condition (or simply *secrecy* condition). There exists also a less stringent formulation, that goes under the name of *weak secrecy* condition, which is

$$\lim_{n\to\infty} \frac{1}{n} I(S, Z^n) = 0 \tag{3.5}$$

meaning that the rate of information leaked must vanish when n grows to infinity.

3.2 Secrecy capacity

The metric used in literature to quantify the secrecy that can be achieved on a given channel $p_{YZ|X}$ is the *secrecy capacity*. Such metric has a definition that is similar to the channel capacity and represents the number of secret bits that can be achieved in a single channel use. In order to give a formal definition of the *secrecy capacity*, we first need to formally characterize a code for a wiretap channel.

Definition 3.1. A $(2^{nR}, n)$ code $\mathcal{C}_n$ for a wiretap channel consists of

- a message set $\mathcal{M}$ with finite cardinality $|\mathcal{M}| = 2^{nR}$;
- a random encoding mechanism $p_{X^n|S} : \mathcal{M} \to \mathcal{X}^n$, which maps a message S to a codeword x^n;
- a deterministic decoding mechanism $f_B : \mathcal{Y}^n \to \mathcal{M}$, which maps a codeword y^n to the decoded message $\hat{S}$.

The two requirements that a code for a wiretap channel must satisfy are *reliability*, i.e., the message must be correctly decoded by the intended receiver `B`, and *secrecy*, i.e., having independence between the original message S and the leaked codeword Z^n. The reliability is asymptotically guaranteed if

$$\lim_{n\to\infty} \Pr[\hat{S} \neq S] = 0, \tag{3.6}$$

i.e., if, when n grows, the probability of decoding the wrong symbol using the codebook $\mathcal{C}_n$ becomes negligible. The secrecy is given by the condition on the mutual information $\lim_{n\to\infty} I(S, Z^n) = 0$, as discussed in the previous sections. These two conditions naturally lead to the definitions of secrecy rate and secrecy capacity.

Definition 3.2. The quantity $R_s > 0$ is an *achievable secrecy rate* for the memoryless wiretap channel $p_{YZ|X}$ if there exists a sequence of message sets $\mathcal{M}_n$ and a code $\mathcal{C}_n$ such that

- the code rate R_s is achieved, i.e.,

$$|\mathcal{M}_n| \geq 2^{nR_s}; \tag{3.7}$$

- the reliability condition is satisfied, i.e.,

$$\lim_{n\to\infty} \Pr[\hat{S} \neq S] = 0; \tag{3.8}$$

- the secrecy condition is satisfied, i.e.,

$$\lim_{n\to\infty} I(S, Z^n) = 0. \tag{3.9}$$

The *secrecy capacity* of the memoryless wiretap channel $p_{YZ|X}$ is defined as

$$C_s = \sup\{R_s : R_s \text{is an achievable secrecy rate for the channel}\}. \tag{3.10}$$

If no secrecy rate is achievable, then $C_s = 0$.

3.2.1 Stochastic encoding

A crucial point that needs to be highlighted is that a code for a wiretap channel employs a random encoding mechanism. We assume that `A` has access to a local source of randomness – i.e., such that its realizations are known only to `A` – described by the random variable R of alphabet $\mathcal{R}$, and encodes the messages using the function $f_A : \mathcal{M} \times \mathcal{R} \to \mathcal{X}^n$. Once the random source R and the encoding function f_A are fixed, the encoding mechanisms is uniquely identified by some conditional distribution $p_{X^n|S}$. In order to clarify why stochastic encoding is needed for secret transmission we provide a simple example, from [15].

Example 3.1. Consider a *uniform wiretap channel*, in which the marginal distributions of the channel $p_{Y|X}$ and $p_{Z|X}$ are described by

$$p_{Y|X}(y|x) = \begin{cases} 1/N_{Y|X}, & y \in \mathcal{T}_{Y|X}(x) \\ 0, & y \notin \mathcal{T}_{Y|X}(x) \end{cases}$$

and

$$p_{Z|X}(z|x) = \begin{cases} 1/N_{Z|X}, & z \in \mathcal{T}_{Z|X}(x) \\ 0, & z \notin \mathcal{T}_{Z|X}(x) \end{cases}$$

meaning that a specific symbol x is mapped with uniform probability into a symbol y of the set $\mathcal{T}_{Y|X}(x) \subseteq \mathcal{Y}$ of cardinality $N_{Y|X}$ and into a symbol z of the set $\mathcal{T}_{Z|X}(x) \subseteq \mathcal{Z}$ of cardinality $N_{Z|X}$. For the sake of simplicity, we assume that the cardinalities $N_{y|x}$ and $N_{z|x}$ are the same for all $x \in \mathcal{X}$ and that the codewords are made of a single symbol, i.e., $n = 1$. In order to have a reliable transmission of the symbols, the random encoding mechanisms must map the messages from $\mathcal{M}$ into a sequence of symbols belonging to a subset $\mathcal{X}'$ of $\mathcal{X}$ such that `B` is able to correctly decode the received symbols, i.e.,

$$\mathcal{T}_{Y|X}(x) \cap \mathcal{T}_{Y|X}(x') = \emptyset, \forall x, x' \in \mathcal{X}', x \neq x'.$$

Moreover, in order to have secrecy, the encoding mechanism $p_{X|S}$ must be defined as

$$p_{X|S}(x|S) = \begin{cases} 1/N_{X|S}, & x \in \mathcal{T}_{X|S}(S) \\ 0, & x \notin \mathcal{T}_{X|S}(S) \end{cases}$$

where $\{\mathcal{T}_{X|S}(S)\}_{S \in \mathcal{M}}$ is a suitably chosen partition of $\mathcal{X}'$, for which it holds

$$\bigcup_{x \in \mathcal{T}_{X|S}(S)} \mathcal{T}_{Z|X}(x) = \mathcal{Z}, \forall S \in \mathcal{M}$$

i.e., any message S is mapped to every possible $z \in \mathcal{Z}$ with equal probability. If both the conditions are met, the intended receiver `B` always obtains the correct symbol, i.e.,

$$\Pr[\hat{S} \neq S] = 0,$$

while the eavesdropper `E` is not able to obtain any information from Z, i.e.,

$$I(S, Z) = 0.$$

Therefore, when such conditions hold, the requirement for the secrecy capacity are met. Furthermore, such conditions provide constraints on the cardinality of the message set $\mathcal{M}$. In particular, the reliability condition implies

$$|\mathcal{X}'| \leq \frac{|\mathcal{Y}|}{N_{Y|X}}$$

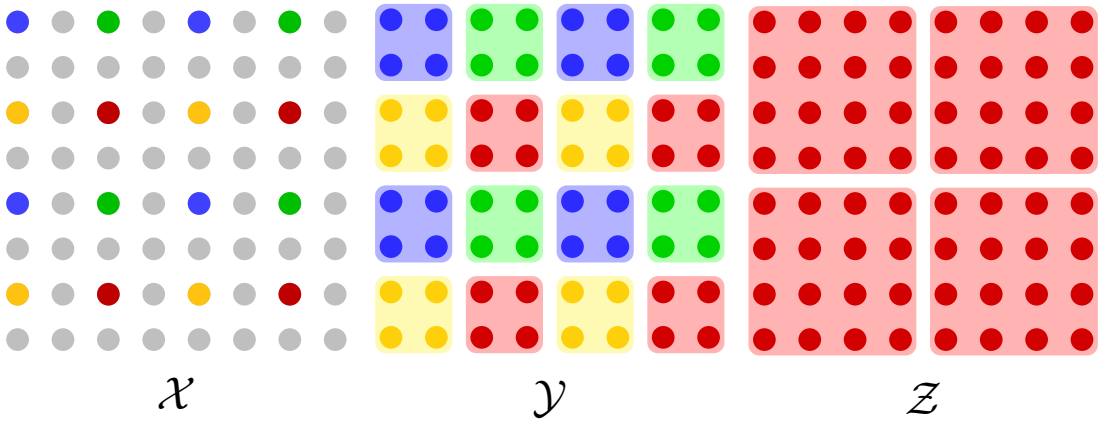


Figure 3.2: A possible configuration of a perfect code on a uniform channe with $|\mathcal{M}| = 4$ possible messages (red, blue, green and yellow). Each message is encoded with uniform distribution in one of the possible bins in $\mathcal{X}'$ and transmitted along the uniform channel. The legitimate receiver is perfectly able to recover the original message, while the eavesdropper receives no information and can only randomly guess the correct message.

and the secrecy condition leads to

$$N_{X|S} \geq \frac{|\mathcal{Z}|}{N_{Z|X}}.$$

A further constraint derives inherently from the stochastic encoding

$$|\mathcal{M}| \leq \frac{|\mathcal{X}'|}{N_{X|S}}.$$

Combining the three constraints, it holds

$$|\mathcal{M}| \leq \frac{|\mathcal{X}'|}{N_{X|S}} \leq \frac{|\mathcal{Y}|}{N_{Y|X}} \frac{N_{Z|X}}{|\mathcal{Z}|}.$$

Assuming a uniform distribution of the input messages over $\mathcal{M}$ and that the symbols received by the eavesdropper are more noisier than the symbols obtained by the legitimate receiver, the number of secret bits achieved per channel use are

$$\begin{aligned}
\log|\mathcal{M}| &\leq \log \frac{|\mathcal{Y}|}{N_{Y|X}} \frac{N_{Z|X}}{|\mathcal{Z}|} \\
&= \log|\mathcal{Y}| - \log N_{Y|X} + \log N_{Z|X} - \log|\mathcal{Z}| \\
&= (\log|\mathcal{Y}| - \log N_{Y|X}) - (\log|\mathcal{Z}| - \log N_{Z|X}) \\
&= (H(Y) - H(Y|X)) - (H(Z) - H(Z|X)) \\
&= I(X,Y) - I(X,Z).
\end{aligned}$$

The upper bound to the number of secret bits that can be achieved per channel use is the secrecy capacity of the uniform wiretap channel, therefore

$$C_s = I(X,Y) - I(X,Z).$$

3.2.2 Characterization of the secrecy capacity

The result obtained in the uniform channel example can actually be generalized to a generic memoryless channel.

Theorem 3.1. *If there exists a probability distribution p_X over $\mathcal{X}$ such that $0 < R_s < I(X,Y) - I(X,Z)$, then R_s is an achievable secrecy rate for the memoryless channel $p_{YZ|X}$.*

The complete proof is given in [8] using the channel equivocation region and in [9] using channel resolvability. The main idea is that when n grows to infinity, a generic symbol vector $x^n \in \mathcal{X}^n$ is ε-typical for $\varepsilon > 0$ arbitrarily small and thus, by the asymptotic equipartition property,

$$\Pr\left[\lim_{n\to\infty} \frac{I(x^n)}{n} = H(X)\right] = 1.$$

Therefore, the information provided by the codeword x^n is $nH(X)$, where $H(X)$ is the entropy of a single symbol, with probability 1. Moreover, the following equalities hold with probability 1, when n approaches infinity:

$$|\mathcal{Y}|^n = 2^{nH(Y)}, \quad |\mathcal{Z}|^n = 2^{nH(Z)},$$

$$|\mathcal{T}_{Y|X}(x^{(0)}) \times \cdots \times \mathcal{T}_{Y|X}(x^{(n-1)})| = 2^{nH(Y|X)}$$

and

$$|\mathcal{T}_{Z|Y}(y^{(0)}) \times \cdots \times \mathcal{T}_{Z|Y}(y^{(n-1)})| = 2^{nH(Z|Y)}.$$

Hence, if $0 < R_s < I(X,Y) - I(X,Z)$, then

$$\begin{aligned}
2^{nR_s} &\leq 2^{n[I(X,Y)-I(X,Z)]} \\
&= 2^{n[H(Y)-H(Y|X)-H(Z)+H(Z|X)]} \\
&= \frac{2^{nH(Y)}}{2^{nH(Y|X)}} \frac{2^{nH(Z|X)}}{2^{nH(Z)}} \\
&= \frac{|\mathcal{Y}|^n}{|\mathcal{T}_{Y|X}(x^{(0)}) \times \cdots \times \mathcal{T}_{Y|X}(x^{(n-1)})|} \frac{|\mathcal{T}_{Z|Y}(y^{(0)}) \times \cdots \times \mathcal{T}_{Z|Y}(y^{(n-1)})|}{|\mathcal{Z}|^n}
\end{aligned}$$

which is the same bound obtained for the uniform channel, meaning that the same approach can be leveraged to infer the existence of a wiretap code $\mathcal{C}_n$ achieving R_s. The secrecy capacity is hence characterized, by applying its definition as the supreme of the set of achievable secrecy rates.

Corollary 3.1. *The secrecy capacity of a memoryless wiretap channel $p_{YZ|X}$ is*

$$C_s = \begin{cases} \max_{p_X}[I(X,Y) - I(X,Z)], & \text{if } \max_{p_X}[I(X,Y) - I(X,Z)] > 0 \\ 0, & \text{otherwise} \end{cases} . \quad (3.11)$$

This result can be expressed more compactly as

$$C_s = \max_{p_X}[I(X,Y) - I(X,Z)]^+ = \max\{0, \max_{p_X}[I(X,Y) - I(X,Z)]\}.$$

3.3 Channels separability

The channel $p_{YZ|X}$ can be split into two channels that are characterized by the marginal distributions $p_{Y|X}$ and $p_{Z|X}$, which represent the legitimate channel and the channel of the adversary, respectively.

Lemma 3.1 (Liang *et al.*)**.** *The secrecy capacity of a wiretap channel $p_{YZ|X}$ depends only on the marginal transition probabilities $p_{Y|X}$ and $p_{Z|X}$.*

Proof. Consider a wiretap code $\mathcal{C}_n$ achieving secrecy rate $R_s > 0$. For such code, reliability is given by $\lim_{n\to\infty} \Pr[\hat{S} \neq S]$, which depends on the marginal distribution $p_{Y|X}$ but does not involve Z. On the other hand, the secrecy condition $\lim_{n\to\infty} I(X^n, Z^n) = 0$ depends on the transition probability $p_{Z|X}$ and, once $p_{Z|X}$ is fixed, the dependence from Y becomes irrelevant. Since this holds for every achievable secrecy rate, it holds for the secrecy capacity. □

This separation makes possible to obtain a lower bound to the secrecy capacity, which relates the capacities of the two marginal channel. In particular, if $C_{\text{AB}} = \max_{p_X} I(X,Y)$ is the capacity of the legitimate channel and $C_{\text{AE}} = \max_{p_X} I(X,Z)$ is the capacity of the eavesdropper's channel, it holds

$$\begin{aligned} C_s &= \max_{p_X}[I(X,Y) - I(X,Z)]^+ \\ &\geq [\max_{p_X} I(X,Y) - \max_{p_X} I(X,Z)]^+ \\ &\geq [C_{\text{AB}} - C_{\text{AE}}]^+. \end{aligned}$$

There are some cases in which the bound is met with equality: a remarkable example is the case of weakly symmetric channels.

Definition 3.3. A channel $p_{Y|X}$ is said to be *weakly symmetric* if the rows of the channel transition-probability matrix are permutations of each other and the sum of the columns

$$\sum_{x \in \mathcal{X}} p_{Y|X}(y|x)$$

is the same for every y.

An example of weakly symmetric channel is the channel with input alphabet $\mathcal{X} = \{0, 1\}$ and output alphabet $\mathcal{Y} = \{0, 1, 2\}$ described by the transition probability matrix

$$\mathbf{P}_{Y|X} = \begin{pmatrix} \frac{1}{12} & \frac{1}{3} & \frac{7}{12} \\ \frac{7}{12} & \frac{1}{3} & \frac{1}{12} \end{pmatrix},$$

where $\mathbf{P}_{Y|X}^{(i,j)}$ denotes the transition probability from the i-th symbol to the j-th symbol.

Proposition 3.1. *If the legitimate channel $p_{Y|X}$ and the eavesdropper's channel $p_{Z|X}$ are both weakly symmetric, then*

$$C_s = [C_{AB} - C_{AE}]^+, \tag{3.12}$$

where C_{AB} is the capacity of $p_{Y|X}$ and C_{AE} is the capacity of $p_{Z|X}$.

3.3.1 Channel orderings

Dividing the channel $p_{YZ|X}$ into two separate channels also allows to compare them and hence better characterize the secrecy capacity, depending on the channel ordering.

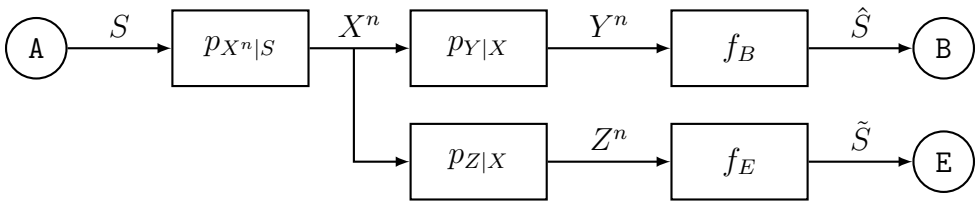


Figure 3.3: Memoryless wiretap channel model with separate channels.

Definition 3.4. Consider two channels, $p_{Y|X}$ and $p_{Z|X}$.

- Channel $p_{Z|X}$ is *physically degraded* with respect to $p_{Y|X}$ if $X \to Y \to Z$ forms a Markov chain, meaning that $\forall(x, y, z) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}, p_{Z|X,Y}(z|x, y) = p_{Z|Y}(z|y)$.
- Channel $p_{Z|X}$ is *stochastically degraded* with respect to $p_{Y|X}$ if there exists a channel $p_{Z|Y}$ such that $\forall(x, z) \in \mathcal{X} \times \mathcal{Z}\ p_{Z|X}(z|x) = \sum_{y\in\mathcal{Y}} p_{Z|Y}(z|y)p_{Y|X}(y|x)$.
- Channel $p_{Z|X}$ is *noisier* than $p_{Y|X}$ if, for any stochastic encoder $p_{X|S}(x|S)$ such that $S \to X \to (Y, Z)$ forms a Markov chain, it holds $I(S, Y) > I(S, Z)$.
- Channel $p_{Z|X}$ is *less capable* than $p_{Y|X}$ if, for any distribution p_X of X, it holds $I(X, Y) > I(X, Z)$.

The relation between the three orderings, in terms of secrecy strength, is

$$\text{physically degraded} \succeq \text{stochastichally degraded} \succeq \text{noisier} \succeq \text{less capable},$$

where "ordering 1" $\succeq$ "ordering 2" means that "ordering 1" provides stronger secrecy than "ordering 2". Moreover, the following chain of implications holds:

$$\text{physically degraded} \Rightarrow \text{stochastichally degraded} \Rightarrow \text{noisier} \Rightarrow \text{less capable}.$$

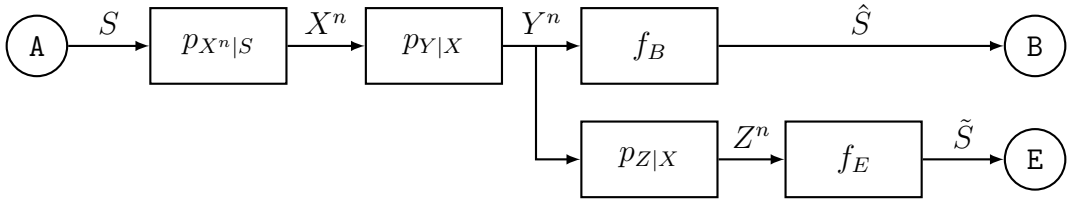


Figure 3.4: Memoryless wiretap channel model with degraded eavesdropper's channel.

Intuitively, when the channel of the eavesdropper is physically degraded with respect to the legitimate channel, it means that the codeword Z^n received by the eavesdropper is the codeword Y^n affected by some additional noise. A physically degraded channel can be modeled as a second channel $p_{Z|Y}$ concatenated to $p_{Y|X}$. The same holds for stochastically degraded channels: since secrecy capacity depends only on marginal distributions, a stochastic degradation is equivalent to a physical degradation.
When the channel of the eavesdropper is noisier than the legitimate one, instead, it means simply that the the legitimate channel enables to share more information than the eavesdropper's. Finally, having a less capable eavesdropper's channel means that $C_{\text{AE}} < C_{\text{AB}}$.

If the channel ordering is known, it is possible to provide an even more precise characterization of the secrecy capacity.

Proposition 3.2. *Let $p_{Y|X}$ be a legitimate communication channel and $p_{Z|X}$ the channel of an eavesdropper.*

- *If $p_{Z|X}$ is less capable than $p_{Y|X}$, the secrecy capacity is given by*

$$C_s = \max_{p_X} [I(X,Y) - I(X,Z)]. \tag{3.13}$$

- *If $p_{Z|X}$ is noisier than $p_{Y|X}$ and both are weakly symmetric, the secrecy capacity is given by*

$$C_s = C_{AB} - C_{AE}. \tag{3.14}$$

The last statement makes possible to provide an explicit formulation of the secrecy capacity for some common channel models.

Example 3.2. A binary symmetric channel (BSC) $p_{Y|X}$ is a memoryless channel in which symbols are binary – meaning that $\mathcal{X} = \mathcal{Y} = \{0,1\}$ – and the transition probability matrix is of the form

$$\mathbf{P}_{Y|X} = \begin{pmatrix} 1-a & a \\ a & 1-a \end{pmatrix}.$$

The parameter $0 < a < 1$, called *incorrect transition probability*, uniquely characterize the channel BSC(a). The capacity of BSC(a) is

$$C = H_2(a) = a \log \frac{1}{a} + (1-a) \log \frac{1}{(1-a)},$$

where $H_2(a)$ is the *binary entropy* of a, i.e., the entropy of a Bernoulli random variable of parameter a.

Consider a wiretap channel in which both the channels $p_{Y|X}$ and $p_{Z|X}$ are binary symmetric, with parameters a and b, respectively, and $0 < b < a < 1$. It is easy to see that the channels are weakly symmetric, by looking at the transition probability matrices, and that the eavesdropper's channel is noisier than the legitimate one. Therefore, it is possible to apply 3.14, obtaining

$$\begin{aligned} C_s &= C_{\mathrm{AB}} - C_{\mathrm{AE}} \\ &= H_2(a) - H_2(b) \\ &= a \log \frac{1}{a} + (1-a) \log \frac{1}{(1-a)} - b \log \frac{1}{b} - (1-b) \log \frac{1}{(1-b)}. \end{aligned}$$

Chapter 4

Deep learning for the wiretap channel

It has been shown in [16] that it is possible to employ neural networks to learn efficient end-to-end schemes that provide physical layer secrecy. In particular, the article focuses on a specific type of channel, which is the *Gaussian wiretap channel*. In this chapter, we describe the properties of the Gaussian wiretap channel and we characterize its secrecy capacity. We also describe how a Gaussian wiretap channel can be modeled using adversarial neural networks and introduce a cross-entropy based approach that allows to learn coding schemes for a Gaussian wiretap channel.

4.1 The Gaussian wiretap channel

Additive white Gaussian noise (AWGN) channels, which we simply refer to as *Gaussian channels*, are a basic channel model that is often used in the context of wireless communications and also for other physical layer communication scenarios. As we mentioned in Chapter 2, a Gaussian channel taken in input a codeword x^n of real valued symbols, generates a vector of n random symbols, which are independent realizations of a Gaussian random variable $N \sim \mathcal{N}(0, \sigma^2)$, and sums each symbol of the codeword with one of the generated symbols. A Gaussian channel is uniquely identified by its the parameter σ of the Gaussian random variable. We assume that the input of the channel is subject to an average power constraint, i.e.,

$$\frac{1}{n} \sum_{i=0}^{n-1} \mathbb{E}[(X^{(i)})^2] \leq P$$

and that the channel has a gain γ.
The channel can be hence equivalently characterized by the SNR, which is defined

as

$$\Lambda = \gamma^2 \frac{P}{\sigma^2}.$$

Gaussian channels are a particularly useful model because they provide an easy and intuitive expression for the capacity.

Proposition 4.1. *The capacity of a Gaussian channel with SNR* Λ *is*

$$C = \frac{1}{2}\log(1+\Lambda). \tag{4.1}$$

When the marginal communication channels of a wiretap channel are both Gaussian, the whole model is called Gaussian wiretap channel.

Definition 4.1. A *Gaussian wiretap channel* is a wiretap channel in which both $p_{Y|X}$ and $p_{Z|X}$ are Gaussian and are characterized by the SNRs Λ_B and Λ_E, respectively, with $\Lambda_B > \Lambda_E$.

A common type of Gaussian wiretap channel, that is widely studied in literature is the *degraded Gaussian wiretap channel*. In this communication scenario, the adversary's channel is degraded with respect to the legitimate channel: if we assume the variance of the legitimate channel $p_{Y|X}$ is σ_B^2, the eavesdropper's can be considered concatenated to $p_{Y|X}$ and hence characterized by a conditional distribution $p_{Z|Y}$, which is again Gaussian with variance

$$(\sigma'_E)^2 = \sigma_E^2 - \sigma_B^2,$$

where σ_E^2 is the overall variance of the eavesdropper's channel (given by the concatenation of the two channels). The degraded Gaussian wiretap channel is depicted in Figure 4.1, where $N_B \sim \mathcal{N}(0, \sigma_B^2)$ and $N_E \sim \mathcal{N}(0, (\sigma'_E)^2)$ represent the additive noise introduced by the two channels. The legitimate receiver obtains

$$Y^n = X^n + N_B^n, \tag{4.2}$$

while the eavesdropper receives

$$Z^n = Y^n + N_E^n = X^n + N_B^n + N_E^n. \tag{4.3}$$

It is easy to see that each Z^n contains more noise than Y^n, thus if $p_{Z|X}$ is degraded with respect to $p_{Y|X}$, then it is also noisier.

The SNRs of the two channels are

$$\Lambda_B = \gamma_B^2 \frac{P}{\sigma_B^2} \quad \text{and} \quad \Lambda_E = \gamma_E^2 \frac{P}{\sigma_E^2},$$

and the respective capacities are

$$C_{\mathrm{AB}} = \frac{1}{2}\log(1+\Lambda_B) \quad \text{and} \quad C_{\mathrm{AE}} = \frac{1}{2}\log(1+\Lambda_E).$$

Since Gaussian channels are not discrete, they cannot possibly weakly symmetric, but it has been shown in [2] that 3.14 holds also for a degraded Gaussian wiretap channel.

Theorem 4.1. *(Leung et al.) The secrecy capacity of a degraded Gaussian wiretap channel is given by*

$$C_s = C_{AB} - C_{AE} = \frac{1}{2}\log\frac{1+\Lambda_B}{1+\Lambda_E}, \tag{4.4}$$

where Λ_B and Λ_E are the signal-to-noise ratios of the intended receiver and of the eavesdropper, respectively.

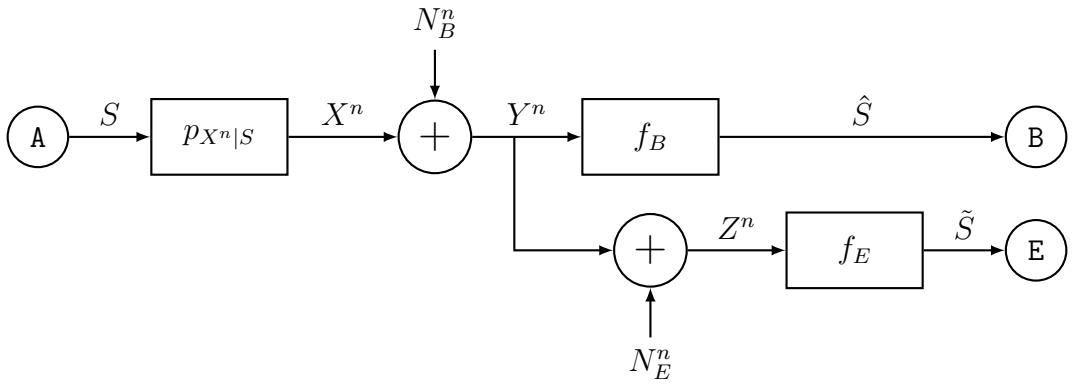


Figure 4.1: Degraded Gaussian wiretap channel model.

4.2 Adversarial learning

In order to build a neural network-based framework that allows to learn efficient schemes for confidential physical layer communications, it is necessary to formulate the task as an optimization problem, i.e., a problem of the kind

$$\min_{\theta\in\Theta} \mathcal{L}(\theta), \tag{4.5}$$

where θ is the vector of the parameters of the neural network.
Actually, in our case we need to model a scenario in which there is a legitimate part, which consists in the transmitter and the intended receiver, and an adversary, which is the eavesdropper. This kind of problem is known as *adversarial learning problem* and is usually modeled by two competing neural networks. The competition consists in the two networks playing a *minimax* game, in which the legitimate network aims to solve

$$\min_{\theta_M} \max_{\theta_E} \mathcal{L}_M(\theta_M, \theta_E), \tag{4.6}$$

while the enemy's network aims to solve

$$\min_{\theta_E} \max_{\theta_M} \mathcal{L}_E(\theta_M, \theta_E), \tag{4.7}$$

where θ_M and θ_E are the parameters that can be tuned by the main network and the adversary, respectively.
The minimax game is played by alternating the training of the two networks, having both minimizing their respective loss function until convergence. If the training is properly done, convergence is reached at the Nash equilibrium of the game.

Algorithm 3 Adversarial training

Input: the loss functions $\mathcal{L}_\mathcal{M}$ and $\mathcal{L}_\mathcal{E}$, the training sets $\mathcal{T}_\mathcal{M}$ and $\mathcal{T}_\mathcal{E}$, the number of training iterations of the two networks N_M and N_E, the optimization algorithms A_M and A_E.

1: Choose the starting parameters $\theta_{M,0}$ and $\theta_{E,0}$
2: **for** $t = 0, 1, \ldots$ **do**
3: **if** $\theta_{M,t}$ and $\theta_{E,t}$ satisfy some specific condition **then** STOP
4: **end if**
5: Sample the mini-batch $\xi_{M,t}$ from $\mathcal{T}_{M,t}$
6: Train the main network for N_M iterations using algorithm A_M on the mini-batch $\xi_{M,t}$, minimizing $\mathcal{L}_\mathcal{M}$ with respect to θ_M and obtaining $\theta_{M,t+1}$
7: Sample the mini-batch $\xi_{E,t}$ from $\mathcal{T}_{E,t}$
8: Train the adversary's network for N_E iterations using algorithm A_E on the mini-batch $\xi_{E,t}$, minimizing $\mathcal{L}_\mathcal{E}$ with respect to θ_E and obtaining $\theta_{E,t+1}$
9: **end for**

Remark: training a network means employing the optimization algorithm to tune the parameter with the aim of minimizing the loss function.

In the case of the wiretap channel, $\theta_M = (\theta_A, \theta_B)$ is the vector of the parameters of the main network, which define the encoder of the transmitter and the decoder of the legitimate receiver. On the other hand, θ_E parametrizes the eavesdropper's decoder. In the following sections, we show a formulation of the optimization problem which is cross-entropy based and that allows to build efficient encoding and

decoding mechanisms for a degraded Gaussian wiretap channel scenario.

4.3 Cross-entropy based approach

We start by assuming that the input of the encoder is a vector s^n whose symbols are independent realizations of a certain random source p_S of alphabet $\mathcal{M} = \{0, \ldots, \ell - 1\}$. We also assume that, for each symbol, both the decoder of the intended user and the adversary's decoder output a vector of likelihoods – $\hat{q}$ and $\tilde{q}$, respectively – who represent an estimation of the probability of each symbol being the transmitted symbol. In order to obtain the best possible estimation, both need to minimize a statistical distance between their likelihood vectors and the pseudo-distribution $p_{S|S}$. We have already shown in 2.1.2 that this can be done, by minimizing the approximate cross-entropy, computed on the training samples. Furthermore, the legitimate network also needs to maximize the statistical distance between $p_{S|S}$ and the adversary's prediction $\tilde{q}$, which can be done by maximizing the cross-entropy between the two. Therefore, a straightforward formulation of the optimization problem could be

$$\min_{\theta_A, \theta_B} \max_{\theta_E} \mathcal{L}_M(\theta_A, \theta_B, \theta_E) = H(p_{S|S}, \hat{q}) - \lambda H(p_{S|S}, \tilde{q}) \tag{4.8}$$

for the main network and

$$\min_{\theta_E} \max_{\theta_M} \mathcal{L}_E(\theta_M, \theta_E) = H(p_{S|S}, \tilde{q}) \tag{4.9}$$

for the adversary. The parameter $\lambda > 0$ regulates how much importance is given to security for the legitimate network. Again, the approximate cross-entropy between $p_{S|S}$ and an estimate likelihood vector q is computed on a given training set with labels $s_0, \ldots, s_{m-1}, s_i \in \{0, \ldots, \ell - 1\}$, of size m as follows

$$\bar{H}(p_{S|S}, q) = \frac{1}{m} \sum_{i=0}^{m-1} H(p_{S|S}, q|S = s_i) = \frac{1}{m} \sum_{i=0}^{m-1} \sum_{j=0}^{\ell-1} p_{S|S}(j|s_i) \log \frac{1}{q(j|s_i)}. \tag{4.10}$$

Even if the loss function

$$\mathcal{L}_M(\theta_A, \theta_B, \theta_E) = H(p_{S|S}, \hat{q}) - \lambda H(p_{S|S}, \tilde{q})$$

can be employed from the legitimate channel, one main issue is that the function does not have a lower bound, or rather, the lower bound is $-\infty$. Such lower bound is reached when the value of $H(p_{S|S}, \hat{q})$ is finite, while there exists a symbol j for

which the likelihood vector $\tilde{q}$ satisfies

$$\tilde{q}^{(j)} = 0,$$

i.e., has the j-th component equal to 0, implying a completely wrong prediction by the adversary's decoder. Even if it is rare to have the components of the likelihood vector being exactly zero, the legitimate network can get close to the lower bound by applying a permutation of the one-hot encoding matrix, so that a symbol j is mapped into a different one-hot encoded vector e_i, with $i \neq j$.

Example 4.1. Consider a simple case in which the symbols take values in $\mathcal{M} = \{0, 1, 2\}$ and the corresponding one-hot encoding matrix is given by

$$\mathbf{I}_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

i.e. symbol 0 is mapped into $[1, 0, 0]^\top$, symbol 1 into $[0, 1, 0]^\top$ and symbol 3 into $[0, 0, 1]^\top$. If the adversary's network is trained on this kind of encoding, if its likelihood vector is, for instance, $\tilde{q} = [0.02, 0.72, 0.26]^\top$, its optimal choice would be to choose $\tilde{s} = 1$ as predicted symbol. In this case, in order to diminish its cost function, the legitimate network just needs to apply the permutation

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix},$$

to the one-hot encoded symbols during the encoding process, so that $[0, 1, 0]^\top$ is mapped into $[1, 0, 0]^\top$ and to apply the inverse permutation $\mathbf{A}^\top$ in the decoding process, for the legitimate decoder. The adversary would be applying the decoding process without applying the inverse permutation, hence mistaking the symbol 0 with the symbol 1, meaning that the cross-entropy value for the symbol 0, for the adversary's decoder, would be

$$1 \cdot \log \frac{1}{0.02} \simeq 7.64386$$

which would give an important negative contribution contribution to the loss function.

4.3.1 PMD equalization

It is easy to see why encoding permutation is not particularly suitable for adversarial learning applications: the adversary can easily mimic the same permutation with further training and therefore the overall adversarial training would lead to an oscillatory behaviour of the loss functions, which makes convergence difficult to be reached. A possible solution, which has been proposed in [16], is to employ a different loss function for the main network. The alternative loss function mimics the initial idea of coset pre-coding, adopted in [11] to construct efficient wiretap codes. The core idea is to divide the message alphabet into clusters and having the adversary seeing the same probability for all the messages in a certain cluster. In order to achieve this goal, the legitimate network must be trained in order to minimize the cross-entropy between an *equalized conditional distribution* and the adversary's likelihood vector $\tilde{q}$. The equalized conditional distribution $\bar{p}_{S|S}$ divides the message alphabet into k clusters of almost equal size and builds a transition probability matrix which maps a symbol into one of the symbols in the same cluster, with equal probability. The transition probability matrix is called *equalization operator*, denoted with $\mathbf{E}$, and is applied by multiplying $\mathbf{E}$ by the one-hot encoding of a message. The clusters are defined according to the well-known k-means clustering algorithm [3].

Algorithm 4 PMD equalization operator

Input: the number of clusters k, the cardinality of the message alphabet $|\mathcal{M}| = \ell$.
Output: the equalization operator $\mathbf{E} \in \mathbb{R}_{+}^{\ell \times \ell}$.

```
Initialize E = 0^{ℓ×ℓ}
Set kmeans.labels = kmeans(I_ℓ)
for t = 0, 1, ..., k − 1 do
    for i = 0, 1, ..., ℓ − 1 do
        if kmeans.labels(i) = t then
            for j = 0, 1, ..., ℓ − 1 do
                if kmeans.labels(j) = t then
                    E_{i,j} = 1
                end if
            end for
        end if
    end for
end for
Normalize E w.r.t. the rows
```

Example 4.2. Consider a message set $\mathcal{M} = \{0, 1, 2, 3\}$ and assume $k = 2$. The

corresponding equalization operator is

$$\mathbf{E} = \begin{pmatrix} 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \end{pmatrix},$$

meaning that messages $\{0, 1\}$ are put into one clusters and $\{2, 3\}$ into the other. For instance, the message $s = 1$ is first one-hot encoded into $e_1 = [0, 1, 0, 0]^\top$ and then equalized into $\mathbf{E}e_1 = [0.5, 0.5, 0, 0]^\top$. If the cross-entropy is minimized between the equalized distribution and the likelihood vector $\tilde{q}$, the adversary won't be able to decide between $\tilde{s} = 0$ and $\tilde{s} = 1$.

Remark. The equalization operator can also be interpreted as the ideal confusion matrix of the adversary's predictions, which is reached in case the cross-entropy between the equalized conditional distribution $\hat{p}_{S|S}$ and the likelihood vector $\tilde{q}$ is taken to zero.

The technique of minimizing the cross-entropy $H(\bar{p}_{S|S}, \tilde{q})$ between the equalized conditional distribution and the adversary's likelihood is called *PMD equalization*. The new loss function of the main network employing the PMD equalization is

$$\mathcal{L}_M(\theta_A, \theta_B, \theta_E) = (1 - \alpha)H(p_{S|S}, \hat{q}) + \alpha H(\bar{p}_{S|S}, \tilde{q}), \tag{4.11}$$

where $0 < \alpha < 1$, is a tradeoff parameter that regulates how much importance is given to security.

4.4 Adversarial network model

We now present the simple adversarial neural network model introduced in [16] to model a degraded Gaussian wiretap channel. The idea is similar to the simple autoencoder shown in Chapter 2, with the difference that in this case one encoder and two competing decoders are employed. The channel is again modeled by a noise layer which generates additive Gaussian noise and the eavesdropper's degraded channel is obtained by inserting an additional noise layer after the noise layer of the main channel.

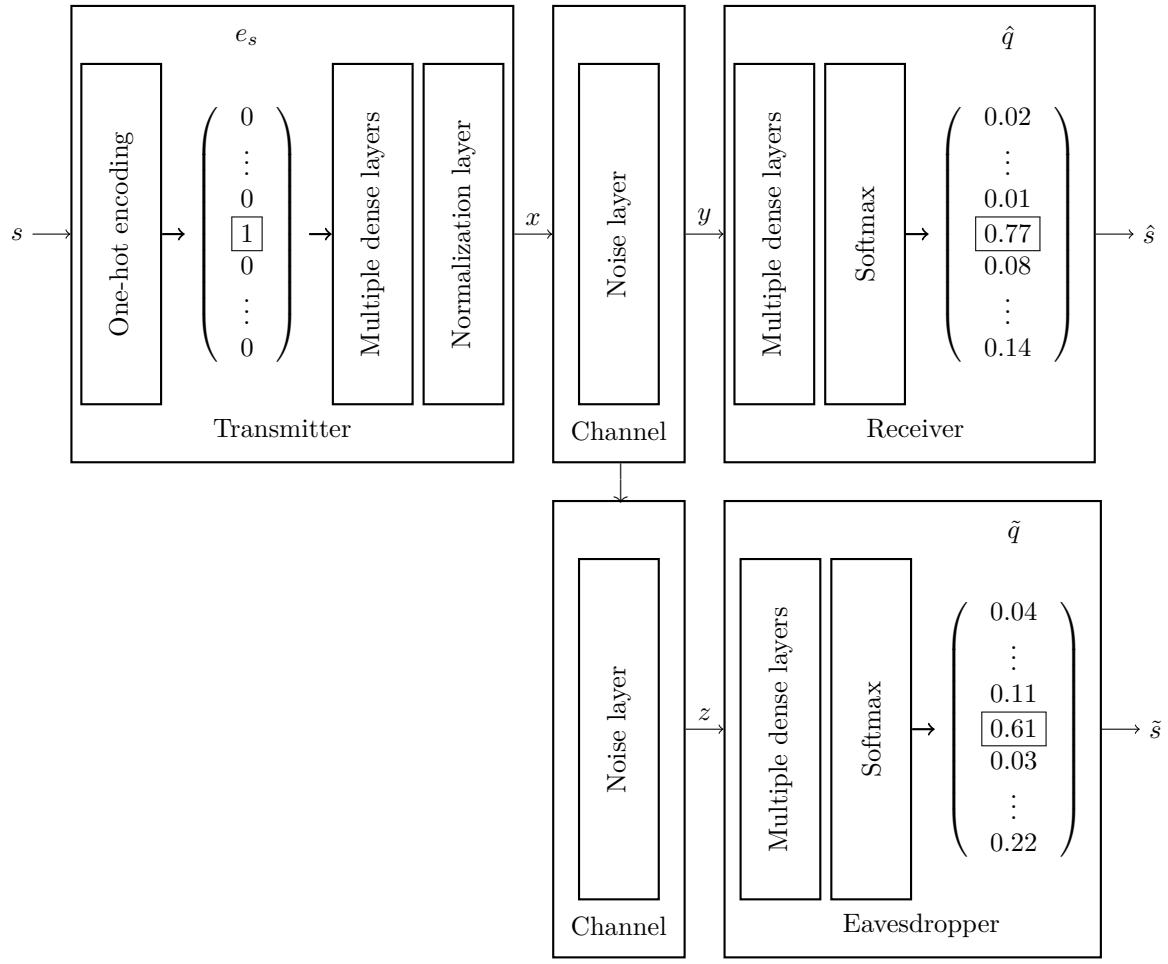


Figure 4.2: Structure of a simple adversarial network that models a Gaussian wiretap channel.

4.4.1 Training results

In [16], the adversarial network model has been implemented in TensorFlow using Adam as optimization algorithm, gradually decreasing the learning rate from $\eta = 10^{-1}$ to $\eta = 10^{-3}$. The size of the training batch is gradually increased from 25 to 300. During the training, the main channel's SNR is $\Lambda_B = 12$ dB, while the eavesdropper's channel has $\Lambda_E = 5$ dB and the average power constraint of the channel's input is set to $\frac{1}{n}\sum_{i=0}^{n-1}\mathbb{E}[X^{(i)}] \leq P = 1$.
Training is divided in three main phases:

1. First, the main network's parameters (θ_A, θ_B) are trained only to efficiently combat noise, i.e., using 4.11 with $\alpha = 0$ as loss function. The parameters θ_E of the adversary's decoder are kept frozen.

2. The adversary is then trained to learn a decoding scheme to decode the symbols

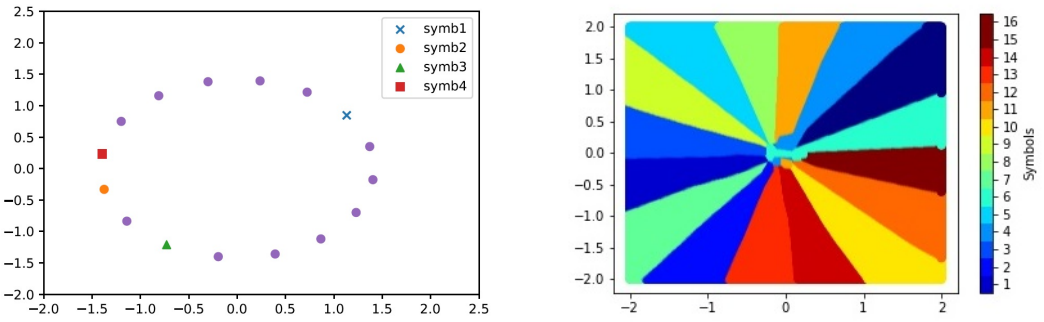


Figure 4.3: The encoding rule and the decision regions obtained training the adversarial network.

sent along the channel, while the legitimate network's parameters are kept frozen. The loss function 4.9 has been minimized.

3. Finally, the main network is trained again using 4.11 with $\alpha = 0.7$, freezing the adversary again.

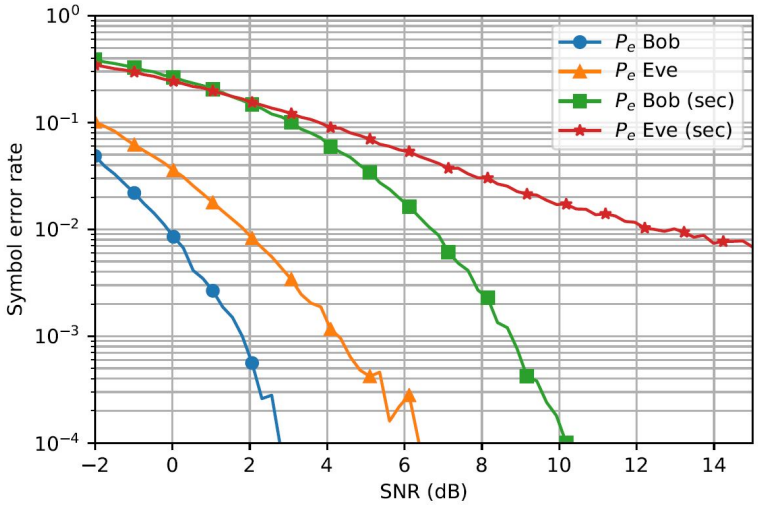


Figure 4.4: Performance for a constellation with $\ell = 16$ symbols and codewords of length $n = 4$, with respect to different values of SNR. The difference between the main network's SNR and the eavesdropper's is fixed to 17 dB.
Note: Bob and Eve are the names which are historically assigned to the legitimate receiver and to the eavesdropper, respectively.

Since steps 2 and 3 are done only once, instead of being repeated until convergence, the results are not the solution of a minimax game and since the last network to be trained is the legitimate one, the performance of the adversary – which is measured in terms of symbol error rate – is worse than the performance that would be achieved with a complete training. Nevertheless, Figures 4.3 and 4.4 show that the network

learns a constellation that hinders the information leakage to the adversary. It is hence possible to conclude that adversarial neural networks can be employed to learn efficient codes that enable a tradeoff between communication rate and secrecy. We used these results as a base for our adversarial learning model, which we will describe in the next chapters.

4.5 Possible extensions

Even if the approach used in [16] turned out to be useful, it does not guarantee to reach the secrecy capacity of the channel. A possible idea to improve the rate might be using the characterization of the secrecy capacity to define a corresponding optimization problem. Assuming that minimizing the cross-entropies $H(p_{S|S}, \hat{q})$ and $H(\bar{p}_{S|S}, \tilde{q})$ is sufficient to achieve reliability and secrecy, the problem of achieving the secrecy capacity can be formulated as

$$\min_{p_{X^n|S}, p_{\hat{S}|Y^n}} I(X,Y) - I(X,Z) \text{ subject to } H(p_{S|S}, \hat{q}) = 0, H(\bar{p}_{S|S}, \tilde{q}) = 0. \tag{4.12}$$

Since constrained optimization is difficult to handle, especially when dealing with neural networks, an unconstrained formulation, like

$$\min_{p_{X^n|S}, p_{\hat{S}|Y^n}} I(X,Y) - I(X,Z) + \alpha H(p_{S|S}, \hat{q}) + \beta H(\bar{p}_{S|S}, \tilde{q}), \quad \alpha, \beta > 0 \tag{4.13}$$

is more favourable. Another unconstrained optimization approach could consist in using the Lagrangian dual of 4.12, which introduces additional parameters but is more likely to provide results that are close to the optimum. The function 4.13 would be employed as loss by the main network, while the objective function of the adversary would remain unchanged, assuming that an eavesdropper has no reason to attempt at reducing the channel's rate.

Chapter 5

Privacy preservation

We have shown in the previous chapters that physical layer secrecy does not come for free and that the price to have secure communications is paid with a lower rate in the transmission. However, when transmitting information, in general we might not want every single bit of information to be secret. The most trivial solution is to apply a secure code to encode the sensitive bits, while using a standard channel code for non-sensitive data. Splitting information in this way, the upper bound to the transmission rate of non-sensitive data would be the channel capacity C, instead of the secrecy capacity C_s. Nevertheless, there are cases in which separating sensitive data from the rest is no trivial task. One may think, for example, to the problem of sending images without revealing the subject that is depicted: there is a lot of information – namely colours, background, details – that alone does not characterize the subject, but that is part of the whole picture. The problem of processing data in order to reduce as much as possible sensitive information leakage is widely studied outside the communication context and is known with the name of *privacy preservation*.
In this chapter, we introduce the problem of privacy preservation and we model it from an information theoretical point of view. We also show that it is possible to employ adversarial neural networks to build privacy-preserving release mechanisms.

5.1 Privacy-preserving data release mechanisms

The objective of a privacy-preserving data release mechanism is, given some *observable data* W, containing some *useful information* U and *sensitive information* S, to provide a *release* X carrying the useful information with minimal distortion while minimizing the amount of sensitive information leaking from X. We assume that U, S, W and X are discrete random variable with finite alphabets $\mathcal{M}$, $\mathcal{S}$, $\mathcal{W}$

and $\mathcal{X}$, and that the *data model* p_{US} and *observation constraint* $p_{W|US}$ are defined by the specific application of the problem. Therefore, the distribution of the triple $(U, S, W) \sim p_{US}p_{W|US}$ is considered fixed and hence cannot be part of the release mechanism.

Definition 5.1. A data release mechanism is a stochastic function uniquely defined by the conditional distribution $p_{X|W}$, which takes some observable data W and returns a release X, such that $(U, S) \to W \to X$ forms a Markov chain.

The efficiency of a release mechanism is determined by two main measures: *distortion* and *privacy*.

Distortion The distortion introduced by a release mechanism $p_{X|W}$ is measured by some distance function $d : \mathcal{M} \times \mathcal{X} \to \mathbb{R}$, which quantifies how much a realization x of X differs from a realization u of U. As a distance function, d must be non-negative and it must be $d(u, x) = 0$ if and only if $u = x$. An efficient release mechanism minimizes the average distortion $\mathbb{E}[d(U, X)]$.

Privacy The privacy guaranteed by a release mechanism $p_{X|W}$ is inversely related to the amount of information about S leaked by X. The quantity used to measure the so-called privacy leakage is thus the mutual information $I(S, X)$, that an efficient release mechanism should minimize.
A perfect release mechanism satisfies both

$$\mathbb{E}[d(U, X)] = 0, \tag{5.1}$$

which is called the *no distortion condition*, and

$$I(S, X) = 0, \tag{5.2}$$

which is called the *perfect privacy condition*.
However, it is almost impossible to have both the conditions holding.

Proposition 5.1. *A data release mechanism $p_{X|W}$ guarantees no distortion and perfect privacy if both the following conditions hold:*

1. *the useful information and the sensitive information are independent;*
2. *the release coincides with the useful information.*

Proof. We first prove by contradiction that in order to have $\mathbb{E}[d(U, X)] = 0$, the output of the release mechanism must be $X = U$. Assume X is different from U

and $\mathbb{E}[d(U,X)] = 0$. Then there exist some $\bar{u} \in \mathcal{M}$ such that $p_U(\bar{u}) > 0$ and some $\bar{x} \neq \bar{u}$ such that $p_{X|U}(\bar{x}|\bar{u}) > 0$. Since $d(u,x) > 0$ for all $x \neq u$, it holds

$$\mathbb{E}[d(U,X)] = \sum_{u \in \mathcal{M}} p_U(u) \sum_{x \in \mathcal{X}} p_{X|U}(x|u) \geq p_U(\bar{u}) p_{X|U}(\bar{x}|\bar{u}) > 0,$$

which is in contrast with the initial assumption.
Then we prove, again by contradiction, that the useful and the sensitive information must be independent. We know that the release of the mechanism must be $X = U$. Assume that $I(S,X) = 0$ and that U and S are not independent, i.e., there exist some $\bar{u} \in \mathcal{M}$ and some $\bar{s} \in \mathcal{S}$ such that $0 < p_{US}(\bar{u},\bar{s}) \neq p_U(\bar{u}) p_S(\bar{s})$. It follows that

$$\begin{aligned} I(S,X) &= I(S,U) \\ &= \sum_{u \in \mathcal{M}} \sum_{s \in \mathcal{S}} p_{US}(u,s) \log \frac{p_{US}(u,s)}{p_U(u) p_S(s)} \\ &\geq p_{US}(\bar{u},\bar{s}) \log \frac{p_{US}(\bar{u},\bar{s})}{p_U(\bar{u}) p_S(\bar{s})} > 0, \end{aligned}$$

which leads to a contradiction. □

We can hence conclude that if there is any dependence between U and S, the only option to guarantee the secrecy of S is to allow some distortion. A release mechanism must hence be suitably chosen according to the *privacy-distortion tradeoff* that must be reached (which depends on the application).

5.2 Adversarial learning for privacy-preservation

The privacy-utility tradeoff can be expressed as an optimization problem, with two possible formulations. One option consists in fixing a *distortion budget* δ, i.e., the maximum value of $\mathbb{E}[d(U,X)]$ that we allow for the sake of preserving privacy, and minimizing the privacy leakage $I(S,X)$ over all the possible data release mechanism satisfying the constraint. This leads to a constrained formulation of the optimization problem, which is

$$\min_{p_{X|W}} I(S,X) \text{ subject to } \mathbb{E}[d(U,X)] \leq \delta. \tag{5.3}$$

The advantage of this formulation is that it is possible to bind the amount of distortion in the release X, but, on the other hand, constrained optimization problem are hard to be handled, in particular when neural networks are involved. Therefore, we

will focus on the second option, which consists in solving the unconstrained problem

$$\min_{p_{X|W}} I(S,X) + \lambda \mathbb{E}[d(U,X)], \tag{5.4}$$

where $\lambda > 0$ regulates how much importance is given to distortion. The tradeoff provided by this formulation is more difficult to regulate, since the mutual information and the distortion are in general completely different metrics and thus tuning λ is not trivial.
Another issue which must be considered is that both the formulations need to minimize the mutual information, which is in general difficult to compute, as it requires an estimate of the input distribution. However, this problem can be overcome exploiting a variational lower bound to the mutual information.

Lemma 5.1. *Let S be a random variable distributed according to p_S over $\mathcal{S}$ and X be a random variable distributed according to p_X over $\mathcal{X}$. Then, for all the distributions of $\tilde{S}$ over $\mathcal{S}$, it holds*

$$I(S,X) \geq H(S) - H(p_{S|X}, p_{\tilde{S}|X}). \tag{5.5}$$

Proof. Using the definitions of mutual information and conditional entropy, it holds

$$I(S,X) = H(S) - H(S|X).$$

The conditional entropy $H(S|X)$ can be rewritten in terms of Kullback-Leibler divergence and cross-entropy, for all distributions $p_{\hat{S}|X}$, as

$$H(S|X) = H(p_{S|X}, p_{\tilde{S}|X}) - \mathbb{D}(p_{S|X} \| p_{\tilde{S}|X}).$$

Combining the two equations, the mutual information can be expressed as

$$I(S,X) = H(S) - H(S|X) = H(S) - H(p_{S|X}, p_{\tilde{S}|X}) + \mathbb{D}(p_{S|X} \| p_{\tilde{S}|X})$$

and the bound follows from the non-negativity of Kullback-Leibler divergence. □

The bound is called "variational" because its tightness depends on the distribution $p_{\tilde{S}|X}$, which in principle can be chosen arbitrarily. It is worth remarking that the bound is met with equality when $p_{\tilde{S}|X} = p_{S|X}$, since the Kullback-Leibler divergence is zero. Moreover, in that case, the cross-entropy is minimized as well. The bound can exploited to formulate an optimization problem which is equivalent to 5.4, but does not require to compute the mutual information.

Proposition 5.2. *The problem*

$$\min_{p_{X|W}} \left(\max_{p_{\tilde{S}|X}} [-H(p_{S|X}, p_{\tilde{S}|X})] \right) \tag{5.6}$$

is equivalent to

$$\min_{p_{X|W}} I(S, X).$$

Proof. The variational lower bound guarantees

$$I(S, X) \geq H(S) - H(p_{S|X}, p_{\tilde{S}|X}).$$

The entropy $H(S)$ is independent from $p_{\tilde{S}|X}$, hence we can work only on cross.entropy. Since the cross-entropy is given by

$$H(p_{S|X}, p_{\tilde{S}|X}) = H(S|X) + \mathbb{D}(p_{S|X} \| p_{\tilde{S}|X}),$$

the value of $H(p_{S|X}, p_{\tilde{S}|X})$ is minimized (and, thus, $-H(p_{S|X}, p_{\tilde{S}|X})$ is maximized) when the Kullback-Leibler divergence is taken to zero, i.e., when $p_{\tilde{S}|X} = p_{S|X}$. In that case, the bound is met with equality, meaning that

$$\begin{aligned} I(U, X) &= H(S|X) - \min_{p_{\tilde{S}|X}} [H(p_{S|X}, p_{\tilde{S}|X})] \\ &= H(S|X) + \max_{p_{\tilde{S}|X}} [-H(p_{S|X}, p_{\tilde{S}|X})]. \end{aligned}$$

Since the equation is always true, it holds also when $I(S, X)$ is minimized with respect to $\min_{p_{X|W}}$. □

It is therefore possible to use

$$\min_{p_{X|W}} \left(\max_{p_{\tilde{S}|X}} [-H(p_{S|X}, p_{\tilde{S}|X})] \right) + \lambda \mathbb{E}[d(U, X)] \tag{5.7}$$

as an alternative to 5.4, in order to optimize the release mechanism. This formulation is a minimax problem and hence it is possible to find its solution by means of an adversarial learning framework, as in the case of secret communications. In particular, an adversarial learning model might consist in two competing neural networks, one modeling the release mechanism, which receives in input U and outputs X, trying to minimize the loss function

$$\mathcal{L}_M(\theta_M, \theta_E) = -H(p_{S|X}, p_{\tilde{S}|X}) + \lambda \mathbb{E}[d(U, X)] \tag{5.8}$$

Figure 5.1: Comparison between the input of a PPAN, the output and the input with the same amount of distortion, applied randomly.

and the other modeling an adversary, who aims to leak sensitive information by minimizing

$$\mathcal{L}_E(\theta_M, \theta_E) = H(p_{S|X}, p_{\tilde{S}|X}). \tag{5.9}$$

This approach has been presented in [18] and has been named *privacy-preserving adversarial networks* (PPAN).

5.3 Results on the MNIST dataset

We show an example in which PPANs are employed to train efficient data release mechanisms that guarantee privacy-preservation. Namely, we present the work of [18] on the MNIST dataset, which consists of 70K labeled images of handwritten digits, 60K for the training set and 10K for the test set. The images are 28×28 in grayscale – normalized in the range $[0, 1]$ – and can hence be considered either as matrices in $[0, 1]^{28\times 28}$ or vectors in $[0, 1]^{784}$. In this example, the images are handled as vectors and are hence flattened and processed using dense layers. The useful data coincide with the observation, i.e., $W = U$, and the sensitive information S is the label of the image. The main network takes as input the images and process it, obtaining a release X, while the adversary acts as a classifier, taking X as input and outputting an estimate $\tilde{S}$ of the true label S. The function used to measure distortion between an original image u and the relative release x is the Kullback-Leibler divergence between corresponding pixels

$$d(u, x) = \frac{1}{784} \sum_{i=0}^{783} s^{(i)} \log \frac{1}{x^{(i)}} + (1 - s^{(i)}) \log \frac{1}{1 - x^{(i)}}, \tag{5.10}$$

where pixels are treated as Bernoulli random variables. The main network also concatenates a vector of 20 random realizations uniformly distributed on $[-1, 1]$, which allow to introduce randomness in the process (otherwise the release would be a deterministic function of the input). Both the networks employ two dense layers of 1000 nodes each, with the hyperbolic tangent as activation function. Moreover, the

release mechanism applies the sigmoid activation function to the output, while the adversary applies first the softmax and then the argmax, like the classifiers described in the previous chapters. Figure 5.1 shows that the noise is exploited by the release mechanism in order to confuse the adversary.

Chapter 6

Privacy-preserving communications

In this chapter, we present the core of our work, which combines physical layer secrecy with privacy preservation. The idea is to extend the principle of the release mechanism to a communication scenario, aiming to obtain an end-to-end coding scheme which transmits as much useful information as possible to the legitimate receiver, while preventing the adversary from be able to leak sensitive information. We considered the problem of transmitting actual data instead of symbols, making it a joint source and channel coding design problem. We built an adversarial learning framework, which is based on neural network, employing the approach used in [17] for learning a joint source and channel coding scheme. Furthermore, we developed an adversarial neural network that implements our approach for labeled images. The network simulates a wiretap channel in which images must be transmitted to a legitimate receiver, while keeping the eavesdropper unable to classify them.

6.1 Information theoretic model

Consider a model with three main actors: a transmitter (`A`), a receiver (`B`) and an eavesdropper (`E`). Suppose `A` has some useful information U and wants to transmit it to `B`, but also wants to prevent some sensitive information $S = h(U)$ – where h is a deterministic function – leaking to `E`.
`A` transmits some message X^n along the channel, having `B` and `E` receiving Y^n and Z^n, respectively. `A` chooses X^n to achieve the following objectives:

- `B` must obtain the useful information U from Y^n;
- `E` must not obtain the sensitive information S from Z^n.

A chooses the message X to be transmitted according to $p_{X|U}$. The channel is assumed to be memoryless and characterized by a conditional joint distribution $p_{YZ|X}$. B obtains $\hat{U}$ by using its decoding mechanism, that is described by a deterministic function f_B, aiming to have $\hat{U} = U$; E, instead, applies f_E, to obtain $\tilde{S}$, aiming to have $\tilde{S} = S = h(U)$. A further assumption is that h is known to all the actors A,B and E, meaning that obtaining U is sufficient to obtain S. If this last assumption holds, the conditional probability $p_{S|U}$ is defined by

$$p_{S|U}(s|u) = \chi[s = h(u)] = \begin{cases} 1, \text{ if } s = h(u) \\ 0, \text{ otherwise} \end{cases} \tag{6.1}$$

and it can be shown that if h is not injective, then for the adversary obtaining S is in general easier than obtaining U.

Proposition 6.1. *Given the true useful-sensitive information pair (U, S), if $S = h(U)$, it holds*

$$\mathbb{P}[\tilde{S} = S] \geq \mathbb{P}[\tilde{U} = U]. \tag{6.2}$$

If h is not injective, then the inequality is strict.

Proof.

$$\begin{aligned}
\mathbb{P}[\tilde{S} = S] &= \sum_s p_S(s) \cdot \mathbb{P}[\tilde{S} = s | S = s] \\
&= \sum_u p_U(u) \sum_s p_{S|U}(s|u) \cdot \mathbb{P}[\tilde{S} = s | S = s] \\
&= \sum_u p_U(u) \sum_{s:s=h(u)} \mathbb{P}[\tilde{S} = s | S = s] \\
&= \sum_u p_U(u) \sum_{s:s=h(u)} \sum_{u:h(u)=s} \sum_{u':h(u')=s} \mathbb{P}[\tilde{U} = u' | U = u] \\
&= \sum_u p_U(u) \sum_{u':h(u')=h(u)} \mathbb{P}[\tilde{U} = u' | U = u] \\
&\geq \sum_u p_U(u) \cdot \mathbb{P}[\tilde{U} = u | U = u] \\
&= \mathbb{P}[\tilde{U} = U].
\end{aligned}$$

□

Intuitively, the sensitive data contains a part of the information carried by the useful data and this makes easier for the adversary to obtain it. However, this allows to transmit the remaining useful information without caring about keeping it secret.

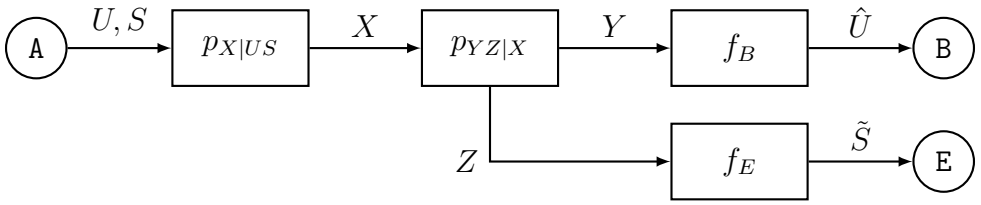


Figure 6.1: Information theoretic model of a wiretap channel used for privacy preservation.

6.2 Privacy capacity

In our model, A is interested in transmitting secretly only a part of the information that is sent along the channel. Therefore, the secrecy capacity would be an excessively stringent bound for the rate that can be achieved. We hence define a new type of capacity, which we call the privacy capacity of the channel.

Definition 6.1. A rate $R_p > 0$ is a *privacy rate* for the memoryless channel $p_{YZ|X}$, if for all $n \in \mathbb{Z}_+$, there exist a message set $\mathcal{M}_n$, an encoder and a decoder such that

- $$|\mathcal{M}_n| \geq 2^{nR_p}; \tag{6.3}$$
- $$\lim_{n\to\infty} \mathbb{P}[\hat{U} \neq U] = 0; \tag{6.4}$$
- $$\lim_{n\to\infty} I(S, Z^n) = 0. \tag{6.5}$$

The *privacy capacity* of the memoryless channel $p_{YZ|X}$ is

$$C_p = \sup\{R_p | R_p \text{ is an achievable privacy rate}\}. \tag{6.6}$$

When no privacy rate is achievable, the privacy capacity is 0.

In order to give an intuition on how transmitting less secret information makes possible to improve the rate, we get back to Example 3.1, that we used to introduce the bounds provided by the secrecy capacity.

Example 6.1. Consider the uniform channel from Example 3.1 and assume that the pair (U, S) is uniformly jointly distributed over $\mathcal{M} \times \mathcal{S}$ so that each realization u of U is associated to a unique realization s of S, i.e., S is deterministic given u.

Assume also that for all $s \in \mathcal{S}$

$$|\mathcal{T}_{U|S}(s)| = \frac{|\mathcal{U}|}{|\mathcal{S}|} = N_{U|S},\ \forall s \in \mathcal{S},$$

where $\mathcal{T}_{U|S}(s)$ is the set of $u \in \mathcal{M}$ that are mapped into s. Since $\mathcal{S}$ is mapped uniformly into $\mathcal{U}$, we can rearrange $(U,S) \to X \to (Y,Z) \to (\hat{U},\tilde{S})$ into the Markov chain $S \to U \to X \to (Y,Z) \to (\hat{U},\tilde{S})$, where the source p_S and the conditional distribution $p_{U|S}$ are both uniform. We can adopt the same reasoning that we followed in the secrecy capacity case. The condition to achieve reliability is

$$|\mathcal{X}'| \leq \frac{|\mathcal{Y}|}{N_{Y|X}}$$

while secrecy requires

$$N_{X|S} = N_{X|U}N_{U|S} \geq \frac{|\mathcal{Z}|}{N_{Z|X}} \Rightarrow N_{X|U} \geq \frac{|\mathcal{Z}|}{N_{Z|X}N_{U|S}}.$$

Combining both with the constraints with the bound on the cardinality of $\mathcal{M}$, it holds

$$|\mathcal{M}| \leq \frac{|\mathcal{X}|'}{N_{X|U}} \leq \frac{|\mathcal{Y}|}{N_{Y|X}}\frac{N_{Z|X}N_{U|S}}{|\mathcal{Z}|}.$$

The bound can be applied to the definition of privacy rate to obtain

$$R_p \leq \frac{1}{n}\log_2|\mathcal{M}_n| \leq \frac{1}{n}\log_2\frac{|\mathcal{Y}|}{N_{Y|X}}\frac{N_{Z|X}N_{U|S}}{|\mathcal{Z}|}, \tag{6.7}$$

which can be rewritten exploiting the properties of the logarithm as

$$R_p \leq \frac{1}{n}\log_2|\mathcal{Y}| - \frac{1}{n}\log_2|N_{Y|X}| - \frac{1}{n}\log_2|\mathcal{Z}| + \frac{1}{n}\log_2|N_{Z|X}| + \frac{1}{n}\log_2|N_{U|S}|. \tag{6.8}$$

Using again the properties of typical sequences, it follows that

$$\begin{aligned} C_p &= H(Y) - H(Y|X) - H(Z) + H(Z|X) + H(U|S) \\ &= I(X,Y) - I(X,Z) + H(U|S) \\ &= C_s + H(U|S), \end{aligned}$$

meaning that privacy capacity can overcome the bound given by the secrecy capacity.

Intuitively, the improvement on the bound of the rate is given by the fact that there is a lot of information contained in U that can be transmitted without caring about such information leaking to the adversary, thus, in principle, when $N_{U|S}$ grows, the

privacy capacity approaches the channel capacity.

Remark. The reliability condition of the privacy capacity can be expressed using the distortion function d adopted in the privacy-utility tradeoff. Since $d(u, \hat{u}) = 0$ when $u = \hat{u}$, if it is possible to find an encoder and a decoder such that

$$\mathbb{E}[d(U, \hat{U})] < \varepsilon_n, \varepsilon_n > 0$$

with ε_n vanishing when n grows, reliability is satisfied.

Remark. The encoder-decoder pair of a wiretap channel can be seen as a relaxation of the concept of release mechanism, since it allows to make the intended receiver obtaining a release Y^n and the eavesdropper obtaining a release Z^n, which is correlated to Y^n but different.

6.3 Minimax game formulation

Since both secret physical layer communications and the privacy-utility tradeoff admit an adversarial formulation, which can be expressed as a minimax game, it is natural to extend the reasoning to privacy-preserving communications. The straightforward approach consists in using the privacy-utility minimax game

$$\min_{p_{X^n|U}, p_{\hat{U}|Y^n}} \left(\max_{p_{\tilde{S}|Z^n}} [-H(p_{S|S}, p_{\tilde{S}|S})] \right) + \lambda \mathbb{E}[d(U, \hat{U})]. \tag{6.9}$$

A clear issue in this formulation is that, even if the problem admits a unique solution from a game theoretical point of view – i.e., the Nash equilibrium of the minimax game – the objective function combines two different metrics. We hence adopted a reparametrization of the problem:

$$\min_{p_{X^n|U}, p_{\hat{U}|Y^n}} \alpha\beta \left(\max_{p_{\tilde{S}|X}} [-H(p_{S|S}, p_{\tilde{S}|S})] \right) + (1 - \alpha)\lambda \mathbb{E}[d(U, \hat{U})]. \tag{6.10}$$

The parameter $\beta > 0$ is suitably chosen to take the two metrics to a similar order of magnitude, while $0 < \alpha < 1$ regulates the tradeoff between the minimization of the two metrics, expressing it as a convex combination.
The corresponding adversarial learning formulation is given by the loss function

$$\mathcal{L}_M(\theta_A, \theta_B, \theta_E) = -\alpha\beta H(p_{S|S}, \tilde{q}) + (1 - \alpha)\mathbb{E}[d(U, \hat{U})] \tag{6.11}$$

to be minimized by the main encoder and decoder – i.e., with respect to $\theta_M =$

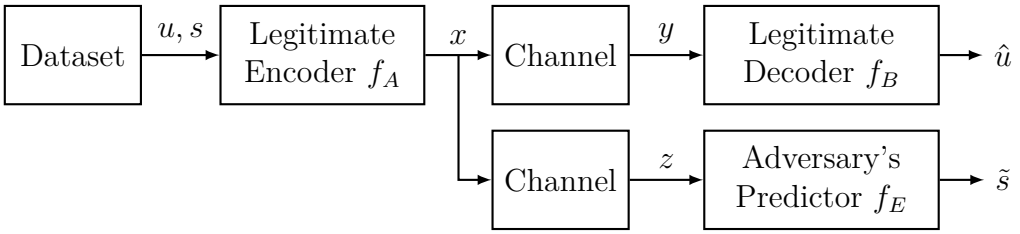


Figure 6.2: Adversarial network model of a wiretap channel used for privacy preservation.

(θ_A, θ_B) – and the loss function

$$\mathcal{L}_E(\theta_A, \theta_B, \theta_E) = H(p_{S|S}, \tilde{q}) \tag{6.12}$$

to be minimized by the adversary – i.e., with respect to θ_E. The distribution $\tilde{q}$ is the output of the softmax of the adversary. It is hence possible to build an adversarial neural network to play the minimax game, by applying Algorithm 3 using the specified loss functions.

6.4 Adversarial network for image transmission

We built a neural network using TensorFlow to evaluate the performance of our framework for the purpose of image/video transmission. We performed both training and tests on the CIFAR-10 datasets, which consists of 60000 (50000 for training and 10000 for test) coloured images of size 32×32 divided in 10 possible classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. The fact that the images are coloured and 32×32 means that they have 3 channels (using the BGR scale) and thus can be handled either as $32 \times 32 \times 3$ matrices or 1024×3 matrices. The main network is an autoencoder which performs joint source and channel coding on the input images, while the adversary acts as a classifier. An input image $u \in \mathcal{M} = [0,1]^{32\times32\times3}$ is first processed by the encoder of the main network and mapped into a codeword $x \in \mathbb{R}^n$, subject to the average power constraint $\frac{1}{n}\sum_{i=0}^{n-1}\mathbb{E}[(X^{(i)})^2] \leq 1$ – without loss of generality with respect to a general power constraint P. The codeword x is fed to the noise layers which represent the two channels. We decided to relax the assumption of the adversary's channel being degraded with respect to the main channel, since it might be too stringent. The two channels map x into y and z, respectively. The noise introduced by the two channels are independent and the amount of noise that is added is determined by the SNRs Λ_B and Λ_E. The codeword y is processed by the legitimate network's decoder, which aims to invert the encoding procedure and outputs the reconstructed image

$\hat{u}$. The adversary's classifier, instead, takes z as input and outputs an estimate of the label $\tilde{s}$. In the next paragraphs, we describe more in details the structure of the main components of the neural network, namely, the encoder, the legitimate decoder and the adversary's predictor. For the main encoder and decoder, we used the network structure employed in [17] for joint source and channel coding. The notation ($\text{kernel}_1 \times \text{kernel}_2 \times$ filter/strides) for a convolutional layer denotes a layer with output dimension equal to the "filter" parameter, convolutional window of size "$\text{kernel}_1 \times \text{kernel}_2$" and strides equal to the "strides" parameter. Dense layers, instead, are denoted only with their output size.

Encoder The encoder processes the input u through five convolutional layers, all applying the PReLU as activation function. The output of the last convolutional layer is fed to a normalization layer, which flattens it and divides it by its power to satisfy the average power constraint, to obtain the actual codeword x. The bandwidth – i.e., the compression rate given by the overall joint source and channel encoding – is

$$\frac{\ell}{n} = \frac{c_{out}}{16 \cdot 3}.$$

We adopted $c_{out} = 16$, hence the bandwidth is $\ell/n = 1/3$. The denominator 16×3 is due to the 2×2 strides that are applied in two convolutional layers, for an overall compression of 16, for all the three image channels.

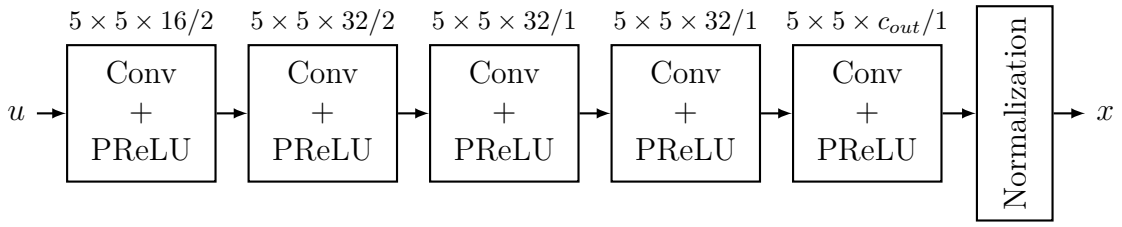


Figure 6.3: Encoder's structure.

Main decoder The intended receiver's decoder expands the noisy codeword y and processes it through five deconvolutional layers, which are basically convolutional layers that also employ upsampling operations, aiming to invert the convolutions performed by the encoder. The first three deconvolutional layers apply the PReLU, while the last one employs the sigmoid that performs a non-linear operation and produces an output whose elements are in $[0, 1]$ and can hence be interpreted as normalized image pixels.

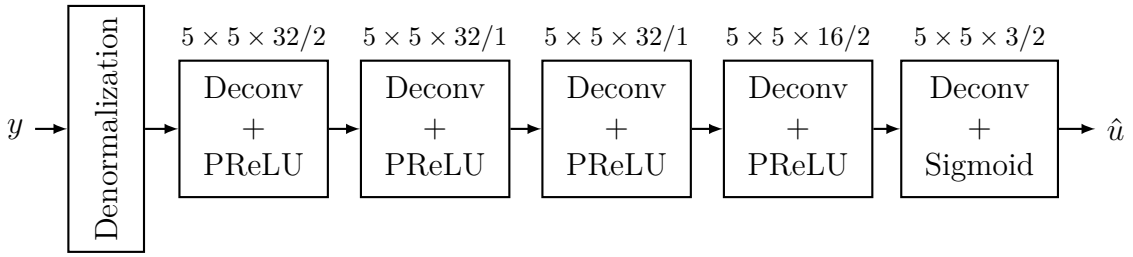
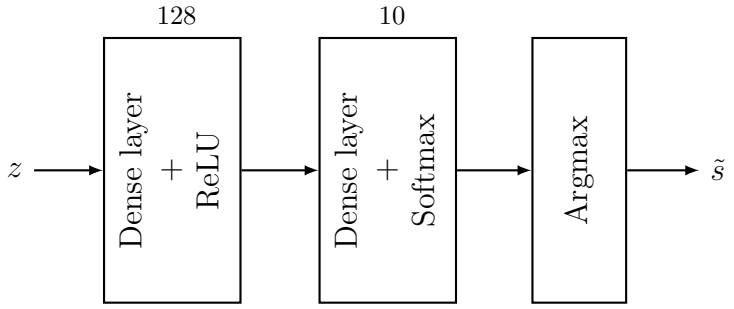


Figure 6.4: Main decoder's structure.

Adversary's classifier The classifier used by the eavesdropper's network, instead of decoding the image, directly tries to leak as much information as possible from the the received vector z, which is more convenient since, by data processing inequality, any additional process can only reduce the amount of information on s carried by z. The classifier is made of two dense layers, the first applies the ReLU activation, while the second employs the softmax to obtain a vector of 10 components, representing the likelihoods of each image label being the correct one. The final prediction $\tilde{s}$ is the argmax of the softmax, i.e., the label with maximum likelihood.



6.5 Distortion function and performance measures

The loss employed by the main network is a convex combination of two functions: one is cross-entropy, which is well defined, while the other is a distortion function d, that must be suitably chosen depending on the application. In this case, the natural distortion function to be used is the *mean square error* (MSE), which is defined between the original image u and the reconstructed image $\hat{u}$, both of size $a \times b$ with 3 channels, as

$$\mathrm{MSE}(u, \hat{u}) = \frac{1}{3ab} \sum_{c=0}^{3} \sum_{i=0}^{a-1} \sum_{j=0}^{b-1} \left(u^{(i,j,c)} - \hat{u}^{(i,j,c)} \right)^2 . \tag{6.13}$$

The measure is averaged on all the the pairs $(u, \hat{u})$ over a batch sample. The MSE is the function to be minimized during the training phases. On the tests, instead, we measure the quality of the transmitted image with the *peak signal-to-noise ratio*

(PSNR), which is a logarithmic scaled ratio between the maximum possible power of a signal and its relative error, measured in decibels (dB). Since CIFAR-10 images take values in $[0,1]^{32\times32\times3}$ the maximum power is 1 and hence PSNR is defined for a batch $u_0,\dots,u_{m-1}$ as

$$\mathrm{PSNR} = \frac{1}{m}\sum_{i=0}^{m-1} 10\log_{10}\left(\frac{1}{\mathrm{MSE}(u_i,\hat{u}_i)}\right) = -\frac{1}{m}\sum_{i=0}^{m-1} 10\log_{10}\mathrm{MSE}(u_i,\hat{u}_i). \tag{6.14}$$

The higher the PSNR, the better the average quality of the reconstructed images. The performance of the adversary's predictor, instead, is measured by the accuracy, i.e., the ratio between the correct predictions and the total number of predictions. For a batch of m predictions $\tilde{s}_0,\dots,\tilde{s}_{m-1}$ compared to the true labels $s_0,\dots,s_{m-1}$, the accuracy is

$$\frac{1}{m}\sum_{i=1}^{m}\chi(\tilde{s}_i = s_0), \tag{6.15}$$

where χ denotes the indicator function.

6.6 Training phases of the adversarial network

We divided the training in three main phases, using an approach similar to [16].

1. On the first phase, the main network trains the parameters of the encoder and the decoder – i.e., (θ_A,θ_B) – in order to learn an efficient code for image transmission only, without combating the adversary (using the loss function with $\alpha = 0$).

2. On the second phase, the adversary trains the parameters θ_E of the predictor. This step is done in order to avoid the adversary's network to be immediately outperformed by the main network, which would make the adversarial training pointless.

3. On the third phases, the two networks play the minimax game by performing the alternate training described in Algorithm 3. Instead of using a stopping criterion based on convergence, we employed a fixed number of adversarial epochs, where one epoch consists in training first the main network (θ_A,θ_B) for a fixed number of iterations and then the adversary's classifier θ_E for a fixed number of iterations. On this phase, the main network actually minimizes the loss function with a value of $\alpha \in [0,1]$ that is specified in advance. The number of epochs is chosen large enough to ensure the performance of both network reaching the steady-state.

6.6.1 Tradeoff regulation with fixed channels

The first problem that we faced was finding a suitable value of the tradeoff parameter α, to guarantee a sufficient level of privacy without excessively compromising the transmitted images. Since the CIFAR-10 dataset contains 10 classes of images, the best achievable privacy would be obtained when the accuracy of the adversary's predictor is 0.1, i.e., having the adversary randomly guessing the class. We hence performed several complete trainings, varying the parameter α and evaluating the PSNR of the images reconstructed by the main network and the accuracy of the adversary's predictor. We report the parameters used for the simulations in Table 6.1. The number of epochs N_{epoch} is referred to the adversarial training epochs. On each epoch, the main network has been trained for N_{AB} iterations, while the adversary has been trained for N_E iterations. Since the main network has an inherent advantage given by the encoder and the decoder being jointly trained, the adversary must train for a higher number of iterations, to avoid having the predictor being quickly outperformed. For the same reason, we did not allow a great advantage in terms of SNR to the legitimate channel.

Table 6.1: Parameters used for training

Parameter	Symbol	Value
Iterations in Phase 1	N_1	30000
Iterations in Phase 2	N_2	30000
Number of epochs	N_{epoch}	40
Main network's iterations	N_{AB}	500
Adversary's iterations	N_E	4000
Main receiver's SNR	Λ_B	10 dB
Adversary's SNR	Λ_E	5 dB
Learning rate	η	10^{-4}
Size of training batch	m_{batch}	32
Size of test set	m_{test}	10000

We show the behaviour of the curves with respect to the number of iterations and the steady-state values for different values of α. For each value, we averaged the results of 5 simulations, randomizing the training batches, and we show the maximum variation from the mean value on the steady-state plot. It is easy to see that changing the parameter α actually enables to regulate the tradeoff between quality and privacy. The tradeoff is non-linear with respect to α, mainly because of the inherent difference between the two metrics that are employed. Nonetheless, it is important to notice that the steady state curves of accuracy and PSNR follow a similar behaviour, meaning that the tradeoff is balanced despite the non-linearity.

Furthermore, for higher values of α the results are subject to a higher variance, which is due to the adversarial training. We focus on the range $0.9 \leq \alpha < 1$, since when α belongs to such range, the adversary's accuracy is taken under 0.5.

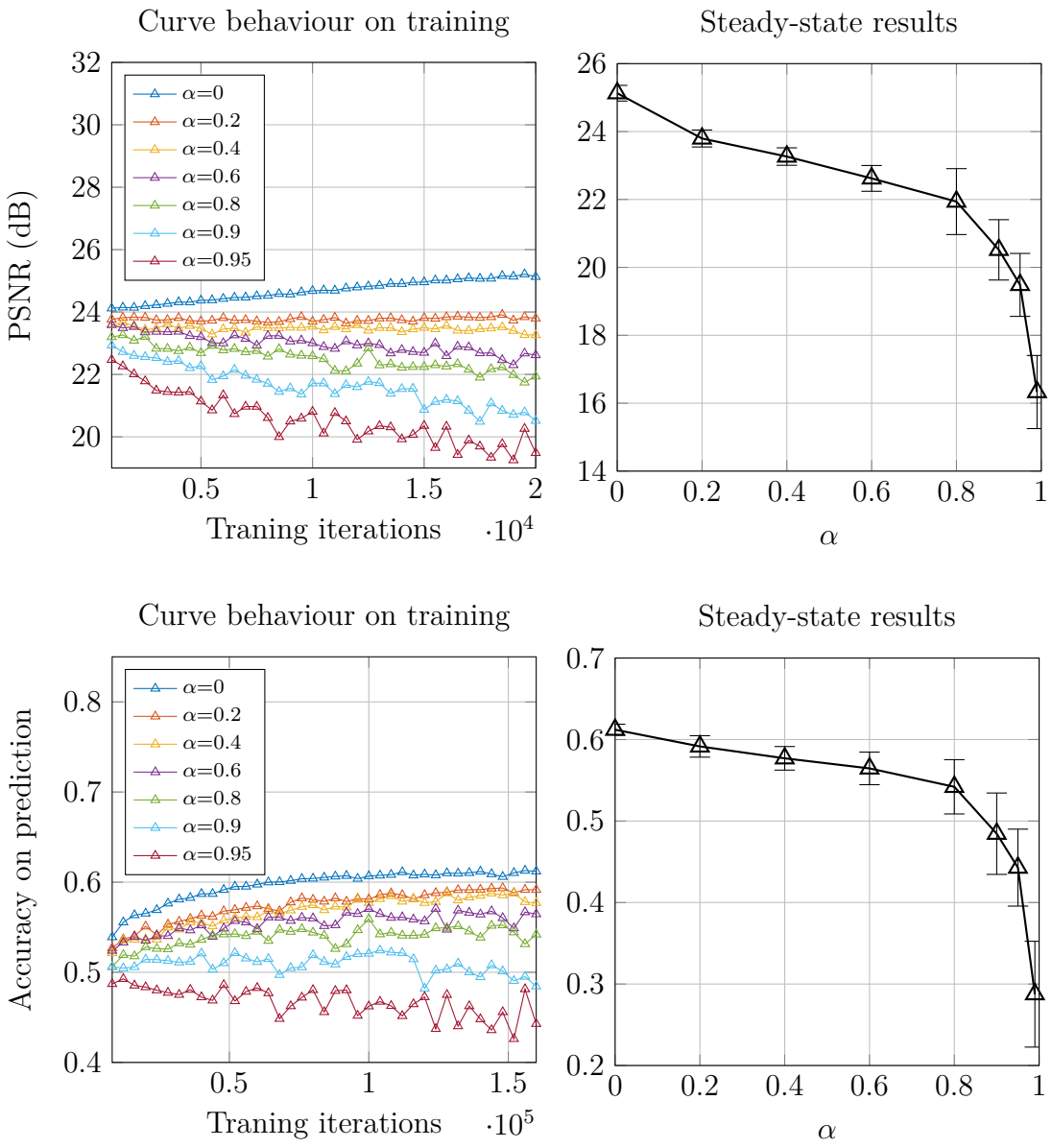


Figure 6.5: PSNR of the reconstructed images and accuracy of the adversary's predictor obtained training the network with the legitimate receiver having an SNR of $\Lambda_B = 10$ dB and the adversary having $\Lambda_E = 5$ dB. The values shown in the curves are sampled at the end of an epoch.

6.6.2 Robustness of the trained model

We saved the network's weights for fixed values of α and tested the robustness of the adversary's network with respect to noise variations. In particular, we show the

test performed on a network trained with $\alpha = 0.95$ with $\Lambda_B = 10$ dB and $\Lambda_E = 5$ dB. On the tests, we varied the value of Λ_E. It can be seen from the plot that the predictor trained by the adversary's network does not improve when the SNR is increased, while instead drops, as expected, when the SNR is lowered.

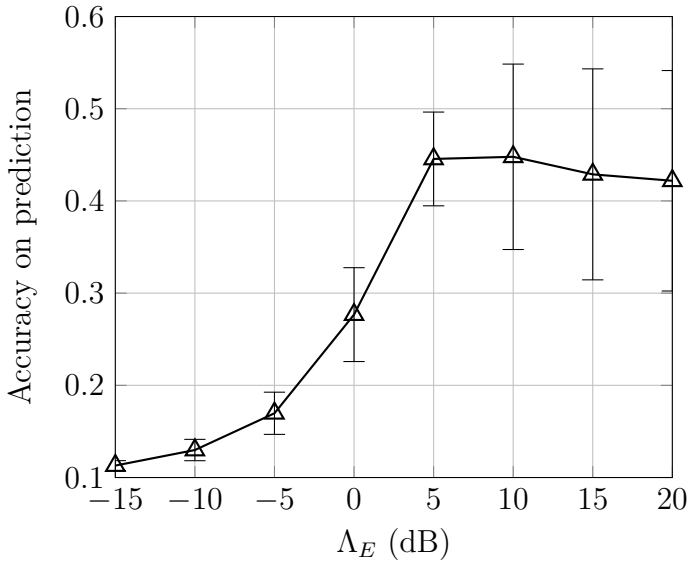


Figure 6.6: Results on the tests performed on the adversary's network trained with $\alpha = 0.95$, $\Lambda_B = 10$ dB and $\Lambda_E = 5$ dB.

The lack of scalability when the SNR increases is mainly due to the fact that when the value of α is increased, the main encoder is trained to behave as a release mechanism for privacy-preservation and hence the codewords transmitted along the channel contain less sensitive information on the labels than the original images. This can be seen also by looking to the image reconstructed by the main decoder, in Figure 6.7.

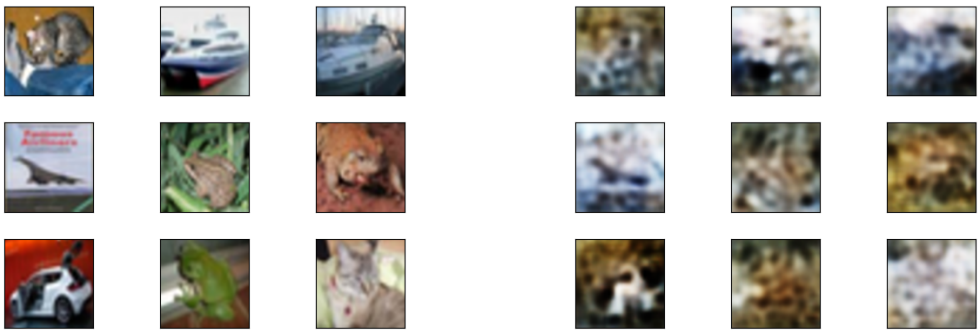


Figure 6.7: The original images compared to the images reconstructed by the legitimate receiver with $\alpha = 0.95$, $\Lambda_B = 10$ dB and $\Lambda_E = 5$ dB.

6.7 Training with PMD equalization

Although the results obtained training the adversarial neural network are positive, the loss function 6.11 presents the same issues of 4.11: the objective of the main network is to maximize the cross-entropy of the adversary's predictor, which can be easily done by performing a permutation of the encoding scheme, as we showed in 4.3. In order to overcome this issue, we adopted the PMD equalization method described in 4.3.1. We employed a unique cluster for the classes, meaning that the goal of the main network is to learn an encoding scheme that makes the output of the adversary's softmax almost uniform, which implies that for a given codeword z all the images must have the same likelihood.

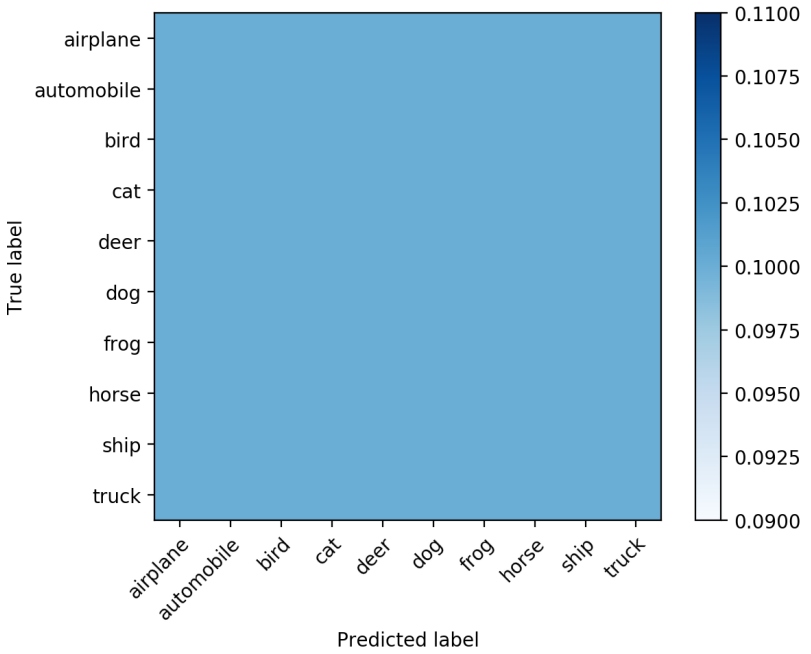


Figure 6.8: The equalization matrix obtained with $k = 1$ clusters.

Therefore, we changed the loss function employed by the main network into

$$\mathcal{L}_M(\theta_A, \theta_B, \theta_E) = \alpha\beta H(\bar{p}_{S|S}, \tilde{q}) + (1 - \alpha)\mathbb{E}[d(U, \hat{U})] \tag{6.16}$$

where $H(\bar{p}_{S|S}, \tilde{q})$ is the equalized distribution, which is represented by the equalization matrix shown in Figure 6.8. We performed again 5 simulations and averaged the results. We employed the same parameters used for the training with the cross-entropy maximization approach, which we refer to as "naive approach". In order to compare the stability of the two approaches, we trained the network both with the naive approach and with PMD equalization, measuring the accuracy and the PSNR

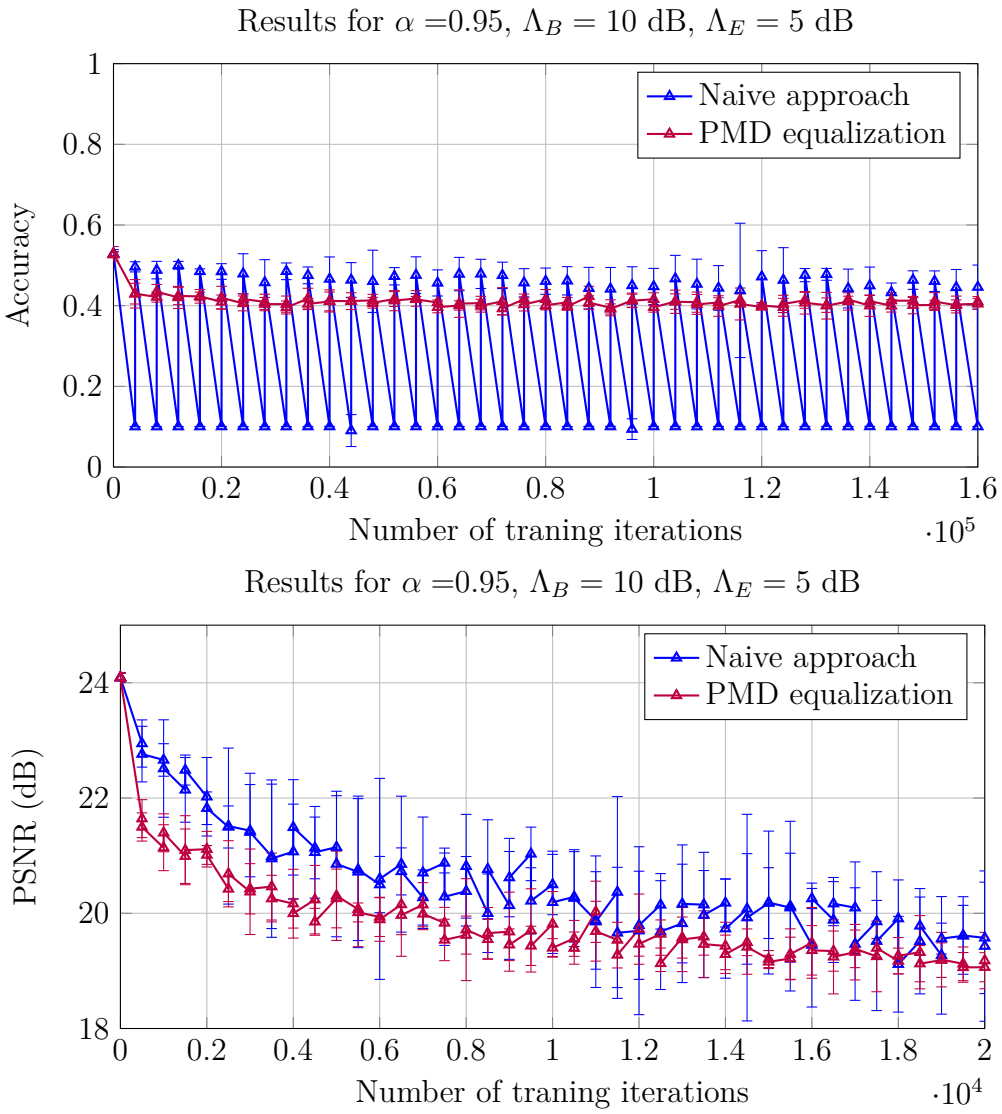


Figure 6.9: Stability comparison between the naive approach and the PMD equalization approach, with $\alpha = 0.95$, $\Lambda_B = 10$ dB and $\Lambda_E = 5$ dB.

both after training the main network and after training the adversary's network. Figure 6.9 shows that at the end of each epoch of the main network the accuracy is brought to 0.1. This means that the encoding scheme learnt by the network is trained to specifically combat the adversary's classifier. At the end of each overall epoch, i.e. after also the adversary has trained, in fact, the accuracy is raised back to a higher value, which is more indicative of the performance of the trained network in terms of privacy. This oscillating behaviour leads also to unstable results on steady-state, which can be seen by comparing the sizes of the error bars. On the other hand, PMD equalization reveals to be a much more stable approach and does not present the oscillating behaviour in the accuracy curve. Then, we trained the network employing the PMD equalization approach with different values of α.

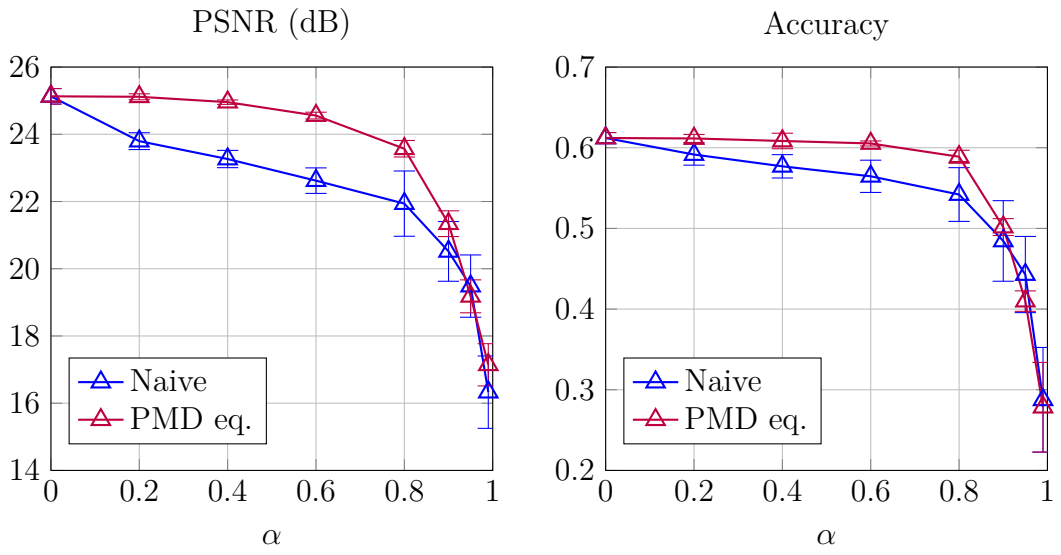


Figure 6.10: Steady-state PSNR with respect to the tradeoff parameter α.

The steady-state results appear different in the range $0 \leq \alpha \leq 0.8$, but in our region of interest, i.e. $0.9 \leq \alpha < 1$, the provide really similar results, on average. Nonetheless, PMD equalization's results are subject to a lower variance, making the approach more reliable.

6.7.1 Robustness with PMD equalization

We tested the PMD equalization approach, measuring PSNR and accuracy while varying the SNR of the adversary's channel, and compared it to the naive approach.

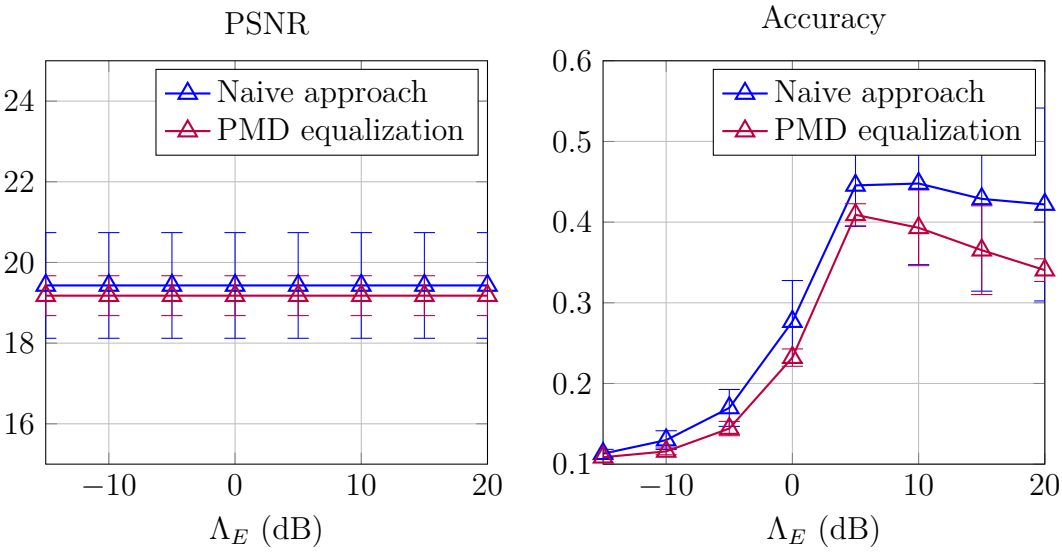


Figure 6.11: Results on the tests performed on the adversary's network trained with $\alpha = 0.95$, $\Lambda_B = 10$ dB and $\Lambda_E = 5$ dB.

Of course the PSNR does not vary at all, since it depends on the main channel: we measured it to show that with $\alpha = 0.95$ the steady-state value of PSNR is almost equivalent, as can be noticed also by the training results. Figure 6.11 shows that PMD equalization leads to a slightly lower accuracy for the adversary, i.e. more privacy, and to a higher stability in the results.

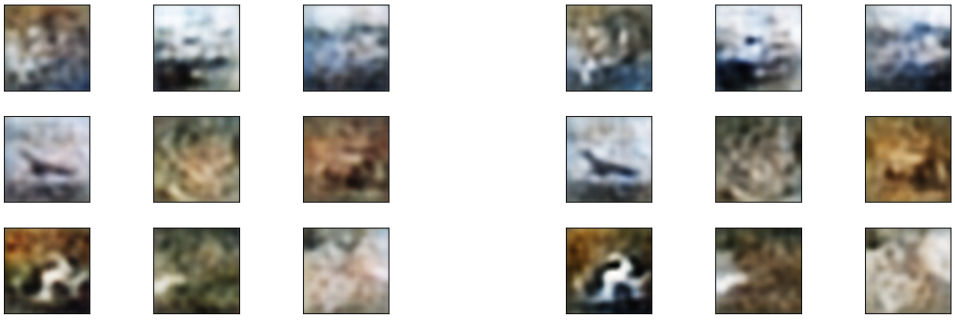


Figure 6.12: Images reconstructed by the legitimate receiver using the naive approach and PMD equalization, with $\alpha = 0.95$, $\Lambda_B = 10$ dB and $\Lambda_E = 5$ dB.

Figure 6.13: An original image of the CIFAR-10 representing a frog, compared to the reconstructed images, with $\alpha = 0.95$, $\Lambda_B = 10$ dB and $\Lambda_E = 5$ dB. The image is correctly classified by the adversary when the adversarial network is trained with the naive approach, while it is misclassified as a deer when the network is trained using PMD equalization.

Figure 6.12 shows that, in any case, the coding scheme learnt by the network with the naive approach and with PMD equalization are quite similar. However, in some cases, even if the reconstructed images at the receiver side are similar, the adversary's classifier makes different predictions. The difference are in some cases due to the decision regions determined by the adversary and in some other cases are due to the encoding rule learnt by the main network.

Chapter 7

Conclusions

In this work, we have developed a neural network-based framework to learn coding schemes allowing to increase the amount of privacy provided for communications over a wiretap channel. We have adopted an adversarial formulation that leads to the solution of a minimax game where a main network and an adversary network compete. The minimax can be interpreted as a game with two players, where at every epoch of training each actor plays their best response to the previous player's move. When the adversarial training reaches the convergence point, that is the Nash equilibrium of the game. We also have implemented our approach to an adversarial neural network that is aimed to transmit image from the CIFAR-10 dataset while preventing the adversary to correctly classifying them. The network is able to guarantee a privacy-distortion tradeoff, which becomes more convenient when the disturb in the adversary's channel is increased. We have adopted first a naive approach, which consists in maximizing the adversary's cross-entropy, and also a more stable approach which aims to take the adversary's softmax output close to a uniform distribution. The results that we have obtained are quite encouraging and we infer that the model should be investigated further in the future.

7.1 Future work

In order to have a better understanding of the limits of the approach, some additional tests should be performed. First of all, it should be verified how much increasing the bandwidth of the channel enables an improvement in terms of privacy. From the results obtained on the secrecy capacity in previous works, we expect that, when the bandwidth is increased, a higher privacy is achievable. Moreover, in order to train the network to be robust to noise variation, fading should be introduced in the channel model. Furthermore, we propose some possible variations that may improve

the model. In our implementation, the encoder is fed with the images, but this does not follow the stochastic encoding principle that is employed in the design of codes achieving the secrecy capacity. In order to mimic the approach with neural network, some randomness should be given in input to the encoding function along with the input image. Another idea to improve the performance of the decoder might be introducing a discriminator that decides whether the reconstructed image belong to the original dataset or not.

Bibliography

[1] A. D. Wyner, *The wire-tap channel*, Bell Syst. Tech. J., vol. 54, pp. 1355-1387, Oct. 1975

[2] S. Leung-Yan-Cheong, M. Hellman, *The Gaussian wire-tap channel*, IEEE Transactions on Information Theory, vol. 24, no. 4, pp. 451-456, 1978

[3] S. P. Lloyd, *Least squares quantization in PCM*, Information Theory, IEEE Transactions on 28.2, pp. 129-137, 1982

[4] D. Barber, F. Agakov, *The IM algorithm: A variational approach to information maximization*, In Proceedings of the 16th International Conference on Neural Information Processing Systems, NIPS'03, pp. 201-208, MIT Press, 2003

[5] T. M. Cover, J. A. Thomas, *Elements of information theory*, Second Edition, Wiley-Interscience, 2006.

[6] Y. Liang, H. V. Poor, S. Shamai (Shitz), *Secure communication over fading channels*, IEEE Transactions on Information Theory, vol. 54, no. 6, pp. 2470-2492, June 2008.

[7] N. Benvenuto, M. Zorzi, *Principles of Communications Networks and Systems*, First Edition, Wiley, 2011

[8] M. R. Bloch, J. Barros, *Physical-Layer Security: From Information Theory to Security Engineering*, First edition, Cambridge University Press, 2011

[9] M. R. Bloch, J. N. Laneman, *Strong Secrecy From Channel Resolvability*, IEEE Transactions on Information Theory, 59(12), pp.8077-8098, 2013

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

[11] M. Yi, X. Ji, K. Huang, H. Wen, B. Wu, *Achieving strong security based on fountain code with coset pre-coding*, in IET Communications, vol. 8, no. 14, pp. 2476-2483, Sept. 2014.

[12] Y. O. Basciftci, Y. Wang, P. Ishwar, *On privacy-utility tradeoffs for constrained data release mechanisms*, in Information Theory and Applications Workshop, Feb. 2016.

[13] S. Ruder, *An overview of gradient descent optimization algorithms*, arXiv:1609.04747, June 2016

[14] T. O'Shea, J. Hoydis, *An Introduction to Deep Learning for the Physical Layer*, arXiv:1702.00832v2, July 2017

[15] N. Laurenti, *Physical layer secrecy*, Course of Information Security, Department of information engineering (DEI), University of Padova. Available: https://elearning.dei.unipd.it/mod/resource/view.php?id=115414

[16] R. Fritschek, R. F. Schaefer, G. Wunder, *Deep Learning for the Gaussian Wiretap Channel*, arXiv:1810.12655v2, Oct. 2018

[17] E. Bourtsoulatze, D. B. Kurka, D. Gunduz, *Deep Joint Source-Channel Coding for Wireless Image Transmission*, arXiv:1809.01733v3, March 2019

[18] A. Tripathy, Y. Wang, and P. Ishwar *Privacy-Preserving Adversarial Networks* arXiv:1712.07008v3, June 2019