universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

Supervised Machine Learning for Pattern Recognition of Frequency Hopping Spread Spectrum Communication

verfasst von / submitted by

Pascal Léon Thiele, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2022 / Vienna, 2022

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 910
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Computational Science UG2002
Betreut von / Supervisor:	Univ.-Prof. Mag. Dipl.-Ing. Dr.techn. Maria Leitner

Abstract

Frequency hopping spread spectrum is a wireless communication technology, used e.g. in Bluetooth. It makes use of a rapidly changing (hopping) carrier frequency of the signal of interest. This master thesis studies the abilities of supervised learners to predict frequency hopping patterns given a small observation window. This is a novel approach in contrast to other pattern prediction algorithms, which rely on a longer monitoring of the signal. Among the possible machine learning architectures, convolutional neural networks are chosen to perform the classification task of the signal presence in a time-frequency representation. By training the neural network with distinguished datasets, four different models are built for comparison. This work shows that the prediction of the signal's time evolution, based on a small observation window, is possible. Furthermore, the possibility to extend the model performances to lower (and higher) signal-to-noise ratios (SNR) than they were trained on is analyzed. It is shown that the extension towards higher SNR is in principle possible. The extension towards lower SNR is possible, but at the cost of a deterioration of the prediction quality.

Kurzfassung

Frequency Hopping Spread Spectrum (Frequenzsprungverfahren) ist eine drahtlose Kommunikationstechnologie, die z. B. bei Bluetooth eingesetzt wird. Sie nutzt eine sich schnell ändernde (springende) Trägerfrequenz. In dieser Masterarbeit werden die Fähigkeiten überwachter Lernsysteme zur Vorhersage von Frequenzsprungmustern, basierend auf einem kleinen Beobachtungsfenster, untersucht. Dies ist ein neuartiger Ansatz im Gegensatz zu anderen Algorithmen zur Vorhersage von Mustern, die auf eine längere Beobachtung des Signals angewiesen sind. Unter den möglichen Architekturen des maschinellen Lernens wird ein convolutional neuronal network gewählt, um die Signalpräsenz in einer Zeit-Frequenz-Darstellung zu klassifizieren. Mit Hilfe von verschiedenen Datensätzen werden vergleichshalber vier unterschiedliche Modelle konstruiert. In dieser Arbeit wird gezeigt, dass die Vorhersage der zeitlichen Entwicklung der Signale auf der Grundlage eines kleinen Beobachtungsfensters möglich ist. Darüber hinaus wird die Generalisierbarkeit des Modelles auf andere signal-to-noise ratios (SNR), als die, die in den Trainingsdaten vorzufinden sind, untersucht. Es wird gezeigt, dass die Erweiterung auf eine höhere SNR prinzipiell möglich ist. Die Erweiterung der Prädiktion auf ein niedrigere SNR gelingt, allerdings nur auf Kosten der Vorhersagequalität.

Acknowledgment

Major thanks to Prof. Maria Leitner for the supervision of my master thesis and her valuable feedback on my academic writing, which go as well to Priv.-Doz. Thomas Zemen for giving me the opportunity of joining the Austrian Institute of Technology (AIT) in his research group and introducing me to the subject of wireless communication technologies. Special thanks to Dr. Laura Bernadó, the project manager of the SCALA project, for her time, patience and open ear. I would also thank the whole of the AIT wireless research group for the warm welcome and (lunch) discussions. Finally, I would like to thank those who supported me during my education and my thesis writing, in particular Chloé and my family.

This work received funding by the SCALA project supported by the Austrian Austrian Research Promotion Agency (FFG) under the programme KIRAS of the Federal Ministry of Finance (BMF).

Contents

List of Figures

List of Tables

1 Introduction

The analysis of data has ever since been an important part of understanding the underlying principles of any subject under study, be it from natural science, social science or other fields, and is thus an important step towards the ability to make predictions. During the XX$^{\text{th}}$ century, first concepts of machine learning (ML) have been developed and experience since the last two decades an ongoing scientific and social interest. Machine learning describes the concept of a machine (computer) learning from data to perform a specific task it is tailored to, and thus being able to make accurate predictions and "smart choices", which improve generally with increasing time and/or data, eventually outperforming humans such as in the strategy game Go or cancer detection [1], [2]. In this master thesis, we will use supervised learning, a specific type of ML, in order to solve a prediction problem of signals employing frequency hopping spread spectrum (FHSS). FHSS is a wireless communication technique in which the signal has a rapidly changing transmission frequency, according to a certain pattern. FHSS is used for different reasons such as efficiency, resilience and security [3]. It is used in various communication protocols, and as such, it is part of the current Bluetooth standard [4].

Narrow-band FHSS signals are also common for remotely controlled unmanned aerial vehicles (UAVs). The detection of radio frequency controlled UAVs is of public interest [5], [6], as they might pose a thread. Therefore, researchers are focusing on the detection and prediction of the FHSS signals used in UAVs. The possibilities of a convolutional neural network to predict the future localisation of specific narrowbanded FHSS signals in a time-frequency representation are investigated in this thesis. The input of the ML model should thereby be solely based on a small observation window. Inspired by this, two research questions will guide the investigation throughout this thesis. First, is it still possible to achieve good results with data with lower (and higher) signal-to-noise ratios than the model was trained on? Second, is it possible for a model to predict the future of measured FHSS signals even if it was solely trained on simulated data? To answer these questions we proceed as the following: First, we acquire measurement and simulation data for different FHSS sources, the latter has the particularity that we have full control of the underlying signal-to-noise ratio. Second, we process the data and create four different datasets used in the training process of the supervised learner. Third, we chose an architecture of our neural network and construct four models based on the respective

datasets. Fourth and lastly, we evaluate the models with a specially designed score in order to respond to the research questions.

This thesis is structured as follows: Chap. 2 "Concepts of Wireless Communication Technology" introduces the subject of wireless communication technologies; some basics in signal modulation, frequency representations, frequency hopping as well as noise and signal power are discussed. Chapter 3 "FHSS Identification and Pattern Prediction" gives an insight into the state of the art of FHSS detection and pattern prediction. Here, the chosen framework is presented and differences to previous works are highlighted. Chapter 4 "FHSS Data Acquisition" describes in detail the measurement process as well as the analysis of the gathered data, which serve as baseline for the simulation setup that is discussed in the following. Concepts of (supervised) ML are presented in Chap. 5 "Supervised FHSS pattern prediction". There, an explanation for the necessary preprocessing, the model construction and the training process, followed by an analysis of the trained models, is given. This thesis gets concluded in Chap. "6 Conclusion".

2 Concepts of Wireless Communication Technology

In this chapter we discuss the basic concepts and terminology of wireless communication technology, especially those necessary for a theoretical and practical treatment of frequency hopping spread spectrum signals.

Communication between a receiver and transmitter can take various forms. Measurable changes of physical quantities, such as sound pressure or field strength, can be interpreted as signals [7]. Some signals require a medium, for example sound waves, in order to be transmitted and others don't, as it is the case for electromagnetic waves (EM waves).

Even though electromagnetic radiation can travel through empty space (vacuum), it can also travel through media like air, but also in glass as it is the case in optical fibres. As the name wireless communication technology suggests, no such fibres or other forms of wires are required in this kind of communication. The behavior of EM waves in a given medium depends strongly on their respective wavelength λ, or equivalently on their frequency $f = \frac{c}{\lambda}$, both being key properties of waves. The c in this equation stands for the wave velocity (which also depends usually on the medium), in the case of EM waves it is exactly 299.792.458 m/s in vacuum and slightly less in air under atmospheric pressure. Frequencies are given in Hertz (Hz), $1\text{Hz} = \frac{1}{\text{s}}$, and quantify how fast a wave oscillates. Due to different behaviours and properties of waves for different frequencies, one categorizes EM waves on the frequency spectrum. Electromagnetic waves in the frequency domain from ca. 30 Hz to 300 GHz are usually called radio waves. Radio frequencies (RF) are especially suitable for wireless communication, examples are WiFi and Bluetooth[1], mainly operating in the 2.4 GHz to 2.5 GHz frequency band[2] [3]. This particular frequency band is also known as the 2.4 GHz Industrial, Scientific and Medical-band (ISM-band), which is a license-free frequency band in contrast to other frequency bands subject to licensing and governmental activities. In the following, we will consider radio signals in the 2.4 GHz band since this is one of the bands where frequency

[1] Micro wave ovens also operate in this frequency band, but they are usually not used for communication

[2] Also, other designated bands exist and continuously subject to novel regulations.

hopping spread spectrum prominently occurs.

We established what radio waves are and that they are the basis of a common type of wireless communication. These waves need to represent in some way the information one aims to transmit. The next section describes how one can encode information in (radio) waves.

2.1 Signal Modulation

Any digital information can be encoded in a bit sequence. In order to transmit this information with an EM wave, one has to encode the bit stream into the waveform, the so-called carrier wave $x(t)$. This process is called signal modulation. Consider for the moment a simple sinusoidal as our radio signal, which is a function of time t of the form

$$x(t) = a\cos\left(2\pi f t + \phi\right), \tag{2.1}$$

with the three parameters, the amplitude a, the frequency f and the initial phase ϕ, defining this specific waveform [7]. The frequency of carrier waves is often referred to as the carrier frequency, as it defines in which spectral region the information gets transmitted. In digital modulation, one defines "symbols", which can be composed of an arbitrary number of bits l, defining thus 2^l different symbols each with a unique bitstream. The bitstream is generated with period T_b. One obtains in this way a symbol sequence which is then encoded via frequency, phase or amplitude modulation in Eq.(2.1). The symbol set is referred to as the symbol alphabet. The symbol period T_s is just equal to lT_b, the symbol rate thus becomes

$$R_s = \frac{1}{T_s} = \frac{R_b}{l}, \tag{2.2}$$

expressed usually in baud (Bd) and with R_b being the gross bit rate. For a fixed symbol rate one could make the bit rate theoretically arbitrarily large, but finite, by increasing l, this has however practical (power) limitations [8]. For a given power limitation, R_b is bounded by the underlying signal-to-noise ratio in the respective communication channel.

We will briefly explain in more detail the frequency shift keying modulation technique, since it will be implemented in the simulation discussed in Sec. 4.3.1.

2.1.1 Frequency Shift Keying

In frequency shift keying (FSK) one assigns symbols to particular frequencies around the nominal carrier frequency f_c, while leaving the amplitude constant. For binary FSK one uses a bit-wise encoding and obtains thus two frequencies, $f_c \pm \delta f$, defined

by the frequency offset δf. Using binary FSK one would switch between these two waveforms depending if a binary 0 or 1 gets currently transmitted, resulting in discontinuities at the switching times. To avoid this, one can use the continuous version of binary FSK, in which a rectangular waveform m, representing the bitstream, gets integrated resulting in a continuous signal at all times t of the form

$$x(t) = a\cos\left(2\pi ft + 2\pi k\int_{-\infty}^{t} m(t')dt'\right), \tag{2.3}$$

in which k takes the role of a modulation parameter controlling the frequency shift [9]. Even though the modulation parameter has an influence on the transmission width of the signal, it is more sensitive to changes of the bitrate R_b. The spectral width of the transmitted signal can be approximated by

$$2R_b(1+k), \tag{2.4}$$

which is Carson's rule [9].

We saw in this section the mathematical description of an EM signal and how it can be modulated to encode information in the considered waveform. We mentioned already that signals experience a spectral widening, depending on the modulation and other factors. The spectral width of the transmitted signal can be best seen in a time-frequency representations, a so called spectrogram. The next section treats the most important tool of signal analysis in radio communication technology, the Fourier transform and its relation to spectrograms.

2.2 Frequency Analysis

There is a fundamental mathematical relationship between the time variable t and the frequency f, they are said to be conjugate variables. This relationship emerges from the one-dimensional Fourier transform (FT) of an integrable time signal $x(t)$. The FT is defined by

$$X(f) = \mathcal{F}\{x(t)\} \coloneqq \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} x(t)e^{-2\pi tf}dt, \tag{2.5}$$

and in the discrete case as

$$X_k = \mathcal{F}\{x_n\} \coloneqq \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} x_n e^{-\frac{2\pi \mathrm{i}}{N}kn}, \tag{2.6}$$

where we can transform the discrete set of (time) $\{x_n\}$ into the discrete (frequency) values $\{X_k\}$. We denote with i the imaginary unit satisfying $\mathrm{i}^2 = -1$ and with N the

number of (time) samples, i.e. the size of the set $\{x_n\}$. One obtains from Eq.(2.6) N discrete frequency points $\{X_k\}$, from which one can recover the original set by performing the inverse or backwards Fourier Transform $\mathcal{F}^{-1}$. Since the x_n are time values, the X_k have the inverse dimension of time, and are thus expressed in Hertz[3]. In communication technologies, one deals with the discrete FT (DFT), because even a continuous RF signal results in discrete data points after being measured/sampled, i.e. we obtain a discrete signal from the receiver's point of view. Usually one always uses the Fast Fourier transform (FFT) to compute the discrete FT. This is because the FFT can speed up the computational process of the discrete FT by exploiting the cyclic properties of complex values [10], [11]. In this thesis we implemented the DFT using an FFT to generate spectrograms from our radio signals.

2.2.1 Time-Frequency Representation

Performing the FT, one can thus get the spectral properties of a time signal, it may however be desirable to have a simultaneous time-frequency representation, an example is shown in the next section. This can be obtained by subdividing the set $\{x_n\}$ into equally sized subsets with consecutive time values. Applying an FFT on each subset results then in a time-frequency representation. This procedure is called short time Fourier Transform (STFT) since the FT is performed only on a short time interval of length τ. To reduce the effect of artefacts of the FFT on a chopped signal, one can apply a window function. The modified FT in the continuous case becomes then

$$X(f;\tau) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-2\pi i t f} dt, \tag{2.7}$$

where w takes the role of a (smooth) windowing function, which is only non-zero in an interval and effectively cuts out all values from $x(t)$ outside this interval, while also attenuating the values on the edges in order to obtain a smooth transition. Various windows are common, an example which we will use later is the Hanning window defined as $\frac{1}{2}\left(1+\cos\left(\frac{2\pi t}{\tau}\right)\right)$ for $-\frac{\tau}{2} \leq t \leq \frac{\tau}{2}$. Without smoothing window, one would just be left with a rectangular window, this introduces however undesired side effects due to the appearing discontinuities and should therefore be avoided [12]. The depiction of the time-evolution of the spectral components is called a spectrogram. In the discrete case of Eq. (2.7), a spectrogram can therefore be seen as two dimensional complex matrices with rows (columns) corresponding to isotimes (isofrequencies)[4].

[3] Fourier transforms are not consistently defined and can differ by a proportionality factor, usually one requires the condition $x(t) = \mathcal{F}^{-1}\{\mathcal{F}\{x(t)\}\}$

[4] Or vice versa, we stick to the proposed convention.

2.2.2 Limitations to Spectral Analysis

In the beginning of this section we explained how spectral information of a time signal $x(t)$ can be recovered. One needs, however, a certain amount of sample points per unit time in order to recover the spectral information completely. It turns out that one needs a sampling frequency $f_s \geq 2f_{max}$ in order to recover the signal with maximal frequency component f_{max}, this inequality is famously known as Nyquist's sampling theorem. The important point here is the number of sample points per unit time. It is thus possible to recover a signal with frequency f_{max} if one measures the complex representation of this signal with $f_s = f_{max}$, since one obtains per sample both an real and imaginary part and it is therefore not violating Nyquist's inequality. A fundamental limitation is also set to the resolution of the STFT in both time and frequency, this limitation is the so called Gabor limit or Gabor-Heisenberg uncertainty principle, which states that one can not retrieve simultaneously arbitrarily precise information on time and frequency of a signal. The exact formulation and the exact bound depends on the context and may not further be discussed. It should be noted that, a priori this has nothing to do with the Heisenberg uncertainty principle from quantum mechanics, but is proper to the field of communication technology, even though the apparent analogy comes from the fact that the mathematical concepts behind the derivation are the same [13], [14].

We described how spectral information can be retrieved from an (arbitrary) signal and how one can construct a time-frequency representation or spectrogram, with Fourier transforms applied only on small time intervals of length τ. Spectrograms are especially suitable if, in contrast to Eq.(2.1) or Eq.(2.3), one has a changing carrier frequency as it is the case for frequency hopping spread spectrum, discussed in the next section.

2.3 Frequency Hopping Spread Spectrum - FHSS

Frequency Hopping Spread Spectrum was initially developed during the Second World War by Hedy Lamarr, to ensure a secure and uninterrupted communication [3]. The FHSS method does not rely only on one frequency to transmit information but instead on a set of frequencies $\{f_i\}$, called channels, from which only one is addressed at any one time. The currently addressed frequency f_c is the carrier frequency of the narrowband signal, which changes ("hops") with a defined hopping rate on the channel set. We call the sequence in which the frequencies get addressed the channel sequence $(f_i)_{i=1}^{K}$, with $K \in \overline{\mathbb{N}}$ [5]. Two examples of frequency hopping

[5] Denoting with $\overline{\mathbb{N}} = \mathbb{N} \cup \infty$, the sequence could potentially be infinite, practically it is not.

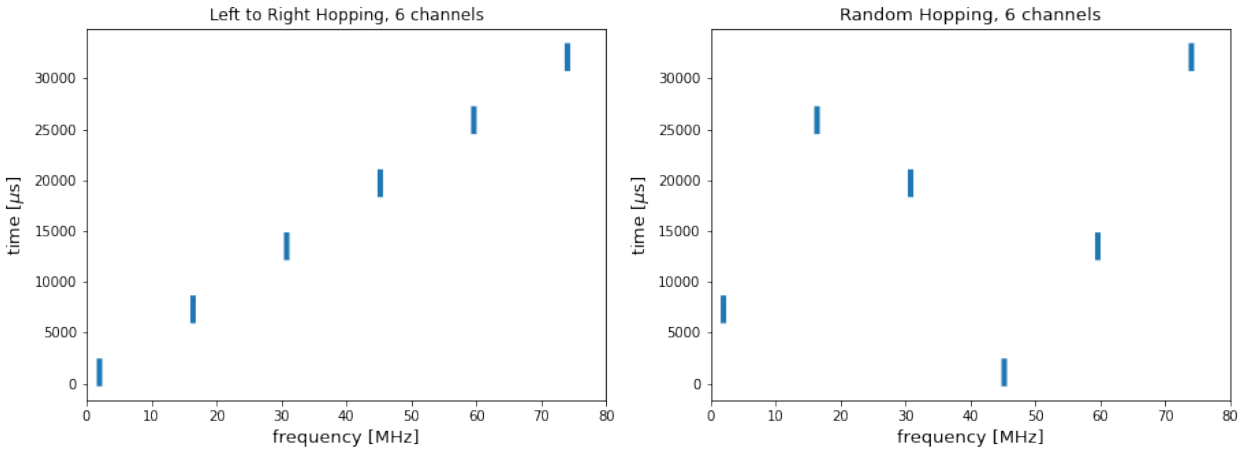


Figure 2.1: Illustration of frequency hopping patterns with 6 channels ($K = 6$), the whole period is depicted. Left, a left to right hopping, in which the next higher frequency channel gets addressed. Right, the same channels, but randomly addressed in time.

patterns are illustrated in Fig. 2.1. Upon completion of the channel sequence, i.e. $i = K$, the FHSS protocol reiterates once again through $(f_i)_{i=1}^{K}$. Thus, depending on the exact FHSS parameters and on the sequence length K, we can observe a periodic behavior of the signal, regardless of the exact hopping scheme. However, if the periodicity exceeds a reasonable amount of time, especially if K becomes very large, the FHSS signal needs to be considered as aperiodic.

The total information carried by the signal is spread in the spectral domain. FHSS has nowadays uses beyond military purposes, however, its big advantages remain the resilience to eavesdropping, to jamming and to the degeneration of the signal due to unwanted interferences [3].

The employed spreading pattern can be either notably deterministic, and if observed long enough periodic because the sequence gets repeated, or generated from a pseudo-random sequence[6]. In both cases the knowledge about the pattern must be shared between transmitter and receiver for the correct synchronisation [15]. One further distinguishes between fast and slow FHSS, the latter being the case for a FHSS implementation in which the symbol rate of the modulated signal is less or equal to the hopping rate, meaning that at least one symbol gets transmitted at any time a carrier frequency is addressed, in contrast to fast FHSS where one symbol may be transmitted over several hops. We only consider slow FHSS in the following

[6] A pseudo-random sequence also has a period, but it is considered to be large enough such that during a reasonable time no periodicity is observed.

[3], [15].

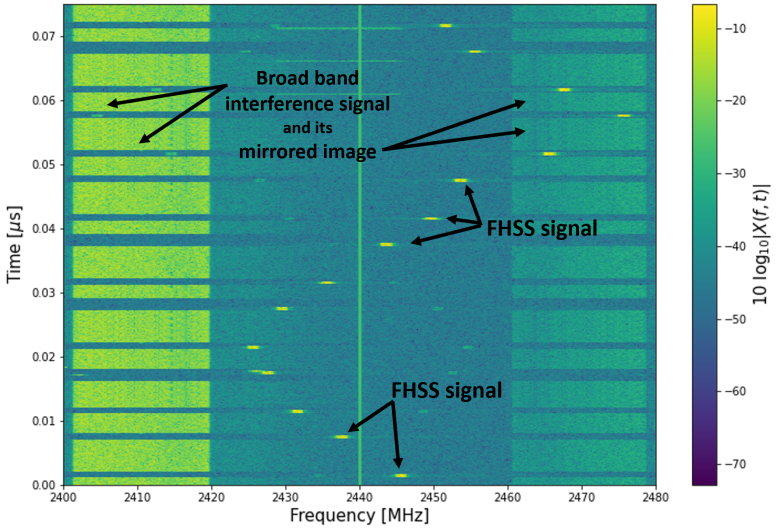


Figure 2.2: Energy time-frequency representation of a FHSS signal recording obtained by FFT and Hanning window function. The colour indicates the signal magnitude in logarithmic scale (brightness for high magnitude). Other signals are also visible, but wide-banded with respect to the FHSS signal. The FHSS signal performs a clearly visible zigzag pattern. One can also distinguish the mirror images of both the FHSS and the interference signal.

2.3.1 FHSS Parameters and Pattern Parameters

We introduce important notions in the context of FHSS, which we gather under the terminology FHSS parameters. Considering the simple case in which each channel $(f_i)_{i=1}^K$ gets addressed an equal amount of time, the time delay between two consecutive hops should be considered uniform as well. We denote by Δt the hop duration and with Δt_{pause} the time (pause) between consecutive hops. The spectral width of the instantaneous signal is denoted by Δf. We introduce the channel repetition rate $C_{\text{r}} \in \mathbb{N}^*$, which describes how often each channel in $(f_i)_{i=1}^K$ gets repeated. Note that this is only a convention we adopted, one could directly encode this in the channel sequence by repeating every channel C_{r} times[7]. We stress this difference because we propose the distinction between FHSS parameters and pattern parameters. For sim-

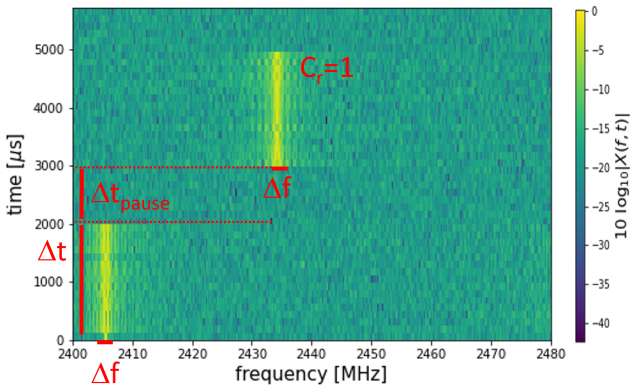
[7]The channel sequence (f_1, f_2, f_3), with $C_{\text{r}} = 2$ results in the same FHSS signal with channel sequence $(f_1, f_1, f_2, f_2, f_3, f_3)$ with either $C_{\text{r}} = 1$ or with ignorance of C_{r}.

plicity, we consider the same Δt_{pause} between repeated channels as between changing channels. Figure 2.3a visualizes the FHSS parameters in a recorded signal.

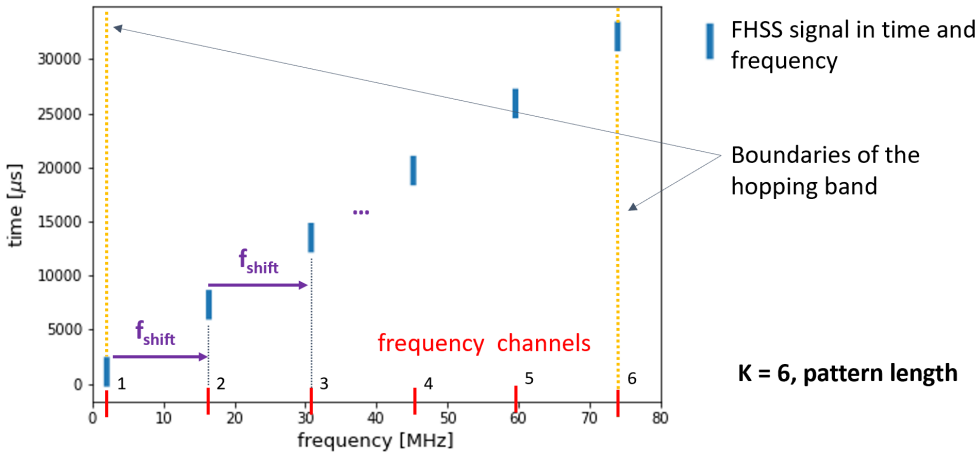
The pattern parameters are separated from FHSS parameters because they describe only properties of $(f_i)_{i=1}^{K}$, notably the sequence length K, which we call pattern length. The sequence length K is equal to the number of channels $|\{f_i\}|$, if no channel gets addressed more than once. Other pattern parameters are $\min(\{f_i\})$ and $\max(\{f_i\})$, i.e the lower and upper bounds of the hopping pattern. We already pointed out that patterns can be clearly deterministic, in particular if the carrier hops a fixed frequency amount, we call that the frequency shift f_{shift}, such pattern can be appreciated in Fig. 2.3b. A FHSS signal is thus exactly determined by its pattern and its FHSS parameters. We define its FHSS period T as

$$T = KC_{\text{r}}(\Delta t + \Delta t_{\text{pause}}), \tag{2.8}$$

which describes the time it takes upon reiterating through $(f_i)_{i=1}^{K}$.



(a) FHSS parameters, represented in an energy spectrogram. No channel repetition ($C_r = 1$).



(b) Pattern parameters, same pattern as in Fig. 2.1 (left). The pattern boundaries are determined by the extremal frequencies.

Figure 2.3: Pattern and FHSS parameters

2.4 Noise and Signal Power

For the moment we only considered the pure form of signals. However, one would have difficulties to record such pure signals, since other radiating sources interfere with the signal. In fact, at non zero temperatures (> 0 K), every object emits EM waves in different frequency ranges, as described by Planck's law. These effects are known as noise and overlap with the signal, thus disturbing the measuring process. Since this noise is the result of random processes in nature, it can be modelled with a suitable probability function, which gives the statistics of the noise contri-

bution. Noise may depend on the frequency range one is considering. In the case of white noise, all frequency ranges are affected equally[8]. In contrast to noise, which is unavoidable, one defines as interferences undesired alien signals which additionally superpose with the signal of interest. In case of additive noise and interferences, one can describe a recorded signal as

$$s(t) = x(t) + n(t) + I(t), \tag{2.9}$$

where $x(t)$ is the signal of interest and $n(t)$ a time-dependent noise and $I(t)$ an interference signal. Assuming that $I(t)$ is negligibly small, one can quantify the effect of noise on the signal with the signal-to-noise ratio (SNR)

$$\text{SNR} = \frac{P_{\text{signal}}}{P_{\text{noise}}}, \tag{2.10}$$

which is defined as the power ratio between signal and noise. The power P_{Signal} of a (periodic) signal is defined in [16] as

$$P = \frac{1}{T}\int_0^T \mid x(t) \mid^2 dt, \tag{2.11}$$

where the integral goes over one period T. The usual unit for (power) ratio in communications technology is decibel (dB), where $\frac{1}{10}$ Bel = 1 dB, which is by definition only a relative power measure[9]. The SNR can be then expressed as

$$\text{SNR}_{\text{db}} = 10 \log_{10}(P_{\text{signal}}/P_{\text{noise}}). \tag{2.12}$$

with $\log_{10}$ indicating the decadic logarithm [7]. Having $P_{\text{signal}} > P_{\text{noise}}$ results in positive SNRs (dB). With increasing SNR, a good signal qualities is expected. Contrarily, $P_{\text{signal}} < P_{\text{noise}}$ makes the signal identification not impossible, but increasingly difficult for more negative SNRs where the signal is completely buried in noise.

This chapter described some basic concepts of communication technology, especially the notation of a radio signal, and its characteristics and how information gets encoded in it. Moreover, the concept of frequency hopping technology was outlined as well as the usefulness and approaches of time-frequency representations. This chapter introduced notions and concepts which we will refer to throughout this thesis. The following chapter gives an overview of implemented techniques in the detection of FHSS signals, the pattern prediction and FHSS parameter estimation.

[8] Something appears white to the human eye if the whole visible frequency spectrum is present with equal strength or power density, i.e. all frequencies are affected equally, hence the name.
[9] Usually relative to 1 W if not stated otherwise. One uses dBm if it is relative to 1 mW.

3 FHSS Identification and Pattern Prediction

The last chapter outlined the advantages of FHSS, notably the resilience to jamming. In fact, if a third person aims to jam such a signal, she[1] can either jam the whole spectral region in which FHSS operates, or she identifies the hopping sequence and acts accordingly. While the first proposition requires only knowledge about the hopping region, it poses several problems. First, it is very costly (in terms of power) to sufficiently interfere with the target signal on the whole spectral range. Second, collateral damage is expected, especially if the operating frequency range lies in a by forth users populated band, such as the 2.4 GHz ISM-band. For these reasons targeted jamming seems to be less costly and less destructive, but requires a more extensive knowledge about the signal to be jammed. Jamming a FHSS signal is a strong motivation in determining and extracting the FHSS pattern, but not necessarily the only one. An example would be a dynamic system which selects the best transmitting frequency where there is no interference. Before any prediction, a signal needs first to be identified. In this chapter we present literature on the detection and classification of different FHSS sources operating in the 2.4 GHz band. We proceed by discussing established methods to extract FHSS parameters and pattern information. The chapter gets concluded by the proposition of our FHSS pattern prediction framework.

3.1 FHSS Detection and Source Classification

The detection of FHSS signals, especially in an interference rich environment, is the first step to address before one aims to extract any information on the underlying pattern. Various (ML) approaches have been proposed in previous works to address this problem.

In [17], two statistical features were extracted from the DFT of the time-signal, which build the input of an adaptive fuzzy inference system classifier. The author's evaluated the FHSS detection accuracies between -3 dB and 25 dB SNRs

[1] In the spirit of quantum key distribution, any attacker on the communication line is denoted by Eve.

and achieved close to 100% detection scores for SNRs ≥10 dB. Support vector machines were used in [18] on recorded time RF signals. A maximum accuracy of 90.1% was achieved by the authors, they found little changes (0.3%) in the classification rate for different SNRs scenarios, ranging from 10 dB to 30 dB. The authors in [19] dealt with time signals as well and used convolutioal neural networks to predict the presence of FHSS (accuracy of 99.7%). They extended also their network to distinguish between different signal sources (three FHSS and one noise only.) and achieved an accuracy of 85.8%. Slightly better accuracies were obtained by the authors in [20], who relied on the XGBoost algorithm and achieved a 99.96% and 90.73% accuracy in the identification and source classification[2] problem respectively. An extended version of the same problem was studied in [21]. The extension here is that the three signal classes where further distinguished into different operating modes, resulting in a 10-class classification problem[3]. The authors opted for an hierarchical ensemble learning approach (XGBoost and k-nearest neighbor)), where the hierarchical learner distinguishes between signal and no-signal classes. They achieved an overall accuracy of 99.2%. A 5-class (4 signal, 1 non-signal) problem was studied in [22] with different classifiers, notably with an artificial neural networks (ANN), for different SNR levels. In contrast to other publications, performances with different interferences (changing amplitude of the interference signal) were addressed. The ANNs performed well regardless of the interference intensity, the overall accuracy of the best performing ANN was ≥ 90% for SNRs ≥ 0 dB. The work presented in [23] investigated a 15 and 17-class classification case (only FHSS sources) for different SNRs and with Bluetooth and Wi-Fi interferences. The presented work used hierarchical learning to discriminate between FHSS signals and non-FHSS signals (with a naïve Bayes decision maker), followed, in case of FHSS signals, by a statistical feature extraction from the RF signal, which were then passed to a ML classifier. Five ML classifier, were studied, the best performing one, at 25 dB, was a random forest with an accuracy of 98.53%. Regardless of the ML classifier, the accuracy was $\gtrsim$ 80% for SNRs ≥10 dB. The detection, or discrimination, of FHSS signals, was investigated with different ML techniques and accuracy performances of %≥ 90 were achieved.

One of the remaining challenges is to extend these performances to low (≤ 10 dB) SNR cases. Once a FHSS signal is identified, one can extract information out of the signal, notably determining the hopping pattern, i.e. $(f_i)_i$ the channel sequence.

[2] With likewise three FHSS sources and one noise only

[3] More precisely, two signal classes were split into four classes respectively, justified by different operating modes of the signal sources.

3.2 FHSS Parameter Extraction and Pattern Prediction

Besides the hopping pattern, an estimation of the FHSS parameters is required in order to address FHSS signals in the most economic and efficient way, or conversely avoid them more efficiently.

The parameter estimation performed in [24]–[27] is based on the time-frequency representation of the signal, obtained by STFT. In [26], a threshold is obtained by a predefined factor multiplied by the (mean) received power. The authors obtained a (binary) signal matrix and analyzed it to extract the number of channels K, the center frequencies, as well as Δt and Δt_{pause}. The authors in [24] proposed a similar dynamical thresholding scheme in the STFT calculation, even for low SNRs. But since the detection procedure missed some signal pixels even for high SNR, the signal matrix underwent a correcting algorithm rectifying the signal areas. With this procedure, the underlying pattern could be analyzed and relevant information such as the channel center frequencies and hop durations could be extracted. The method presented in [27] relied on a time-frequency representation and applied the constant false alarm ratio (CFAR) algorithm. The CFAR algorithm is found to be reliable in applying adaptive thersholds in order to detect narrow band signals, while not triggering broad band interference signals. This makes it suitable for filtering (narrowbanded) FHSS signals. A (dynamical) two-thresholding scheme was performed in [25]. While the first threshold eliminates the noise, a second threshold was applied to filter interference signals. The obtained signal matrix was passed to a DBSCAN clustering algorithm. The DBSCAN clustering parameters were chosen in such a way that signal pixels of one hop were assigned to a distinct cluster. By this method, the relevant FHSS parameters and hopping patterns could be extracted. We refer back to this work in the next chapter to discuss in more detail the findings, since we dealt with the same FHSS signal sources.

The authors in [28] determined the hopping pattern by monitoring channel activities in the considered spectral region, which is assumed to have equispaced channels. If a signal is recorded long enough, the hopping period of the signal can be extracted by searching for repetitions or determining similarity matches. The authors described further how partial sequence matching can be used to overcome the issue if the recording devices can not cover the whole frequency band on its own, but multiple devices need to be used. Similarly, the work presented in [29] assumed a known number of potential channels, i.e. potential center frequencies of the pattern. The authors discussed different approaches in extracting the hopping sequence. One of them consisted in sequentially scanning all possible channels. Since they assumed no channel repetitions and a maximum allowed hopping period, T could be extracted

if an active channel is found and the time instances of two repeated activations are recorded. Knowledge on T enabled then a more efficient monitoring of potential channels. However, the faster approach was to scan in parallel all channel candidates for activeness, but with the cost of multiple measuring devices.

In any prediction approach, the minimum time needed to extract the pattern is T. By means of energy thresholding, it is possible to extract FHSS parameters. It is in principle possible to gain information of the FHSS signal by measuring channel activities and suitable parameter finding methods and therefore determine the pattern, of the same FHSS source.

3.3 Proposed Framework for FHSS Pattern Prediction

To the best of our knowledge, no approach has been proposed yet which predicts the near future of an FHSS signal, based solely on a small observation window. Other approaches, e.g. [28], [29], require some scanning time of the (potential) channels before making future predictions or perform only reactive signal recognition, e.g. [27]. Our first question asks if it is indeed possible to construct a ML model, performing the FHSS signal prediction, whose input is a recording of the signal, immediately prior to prediction window. This framework should allow the user to adjust their own radio signal(s), either to jam the predicted signal or to avoid it preemptively. Another research question we want to address ask if it is possible to predict measured FHSS signals, with a model uniquely trained (and tested) by synthetic data. Furthermore, we explore the last question: can a trained model perform well on other SNR levels than it was trained on?

To answer these questions, we propose a framework which should be able to identify FHSS signals in a time-frequency representation, giving thus also implicit knowledge about hopping parameters such as Δt and Δt_{pause}. The novelty here is that we want to consider the signal within an observation time t_{obs} and predict the signal in the following t_{pred} time interval as visualized in Fig. 3.1.

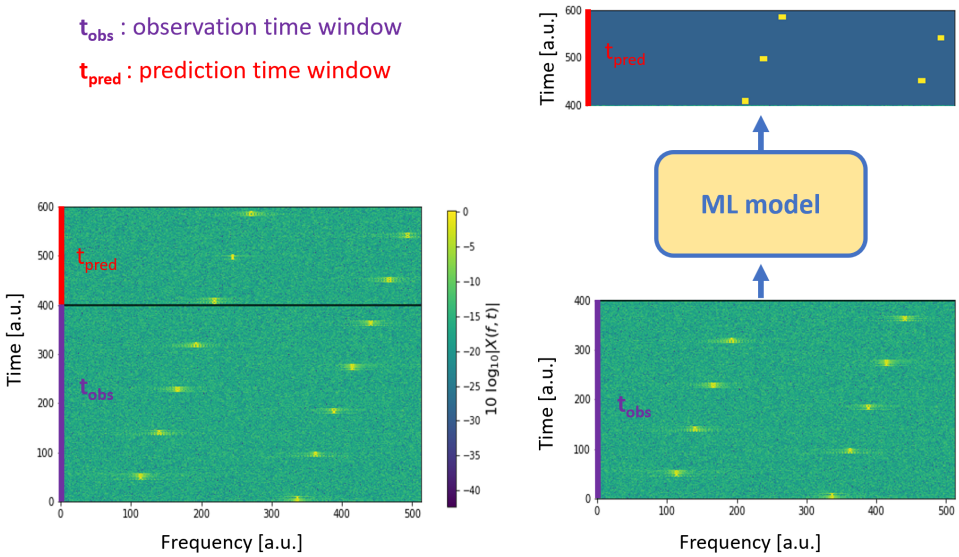


Figure 3.1: Proposed framework for FHSS pattern prediction. On the left, a spectrogram of a FHSS signal, subdivided into t_{obs} and the future time t_{pred}. On the right, the input is feed into the ML model, predicting the signal presence in a time-frequency representation.

The algorithm we propose is obtained by training a supervised ML model. More specifically we want to train a convolutional neural network to handle this task. The ansatz we make here on the data is that the FHSS patterns are clearly deterministic and the pattern length K reasonably small ($K < 100$), in opposite to pseudo-random sequences. To answer our research question, we simulate data resembling those found in measured FHSS sources. Introducing sufficient variation in the simulated signals should allow the model to predict a wider class of measured FHSS signals and enable a controlled and easy generation of different SNR.

4 FHSS Data Acquisition

4.1 FHSS Signal Measurements

In a first step, we gathered measurement data. This data can not only be used for our training process but also builds the baseline for the creation of synthetic data, which will make up the bulk of our training data.

The measurement setup consisted of a USRP-SDR[1], a signal source antenna and one recording antenna. We recorded complex radio signals, meaning a real signal and its by $\frac{\pi}{2}$ phase shifted version, with a sampling frequency f_s of 80 MHz centered at 2.44 GHz. We are thus able to resolve frequencies in the ISM-band from 2.4 GHz to 2.48 GHz as explained in Sec. 2.2.2.

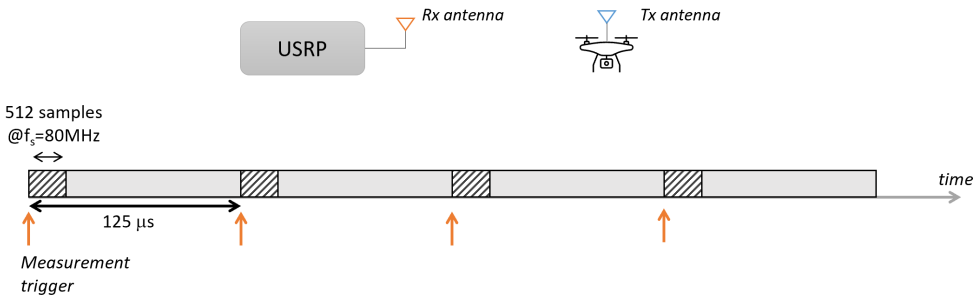


Figure 4.1: Measurement setup. The USRP (Rx) records piecewise 512 continuous samples with f_s of 80 MHz each 125 µs from the FHSS sources (Tx). The center frequency of the measurement device is 2.44 GHz.

A more detailed description on the technical details of the measurement setup can be found in [27]. We measured the radio signals piecewise continuously each 125 µs, and sample 512 consecutive samples at the beginning of each burst, as depicted in Fig. 4.1. This setup enables a low data rate, in order to reduce the required storage capacities, compared to a continuous measurement process. It was performed based on existing knowledge about the signals of interest, which have relevant time scales $\gg$ 125 µs [27]. Each measurement run records 5 s of data and each signal source was measured multiple times as listed in Tab. 4.1. As signal

[1] Universal Software Radio Peripher - Software Defined Radio

sources we use two commercially available drones operating in the 2.4 GHz ISM-band. A DJI Phantom and a Skywalker TITAN drone, equipped with a FrSky control unit. The measurements were performed in the AIT RF-lab with low interference sources, furthermore the signal sources were placed near to the recording antennas, high SNRs are thus expected. All signal sources were measured in the 2.4-2.48 GHz band. Additionally to the signal sources, background noise measurements in the lab were performed. Each drone was measured multiple times in different operating modes as in [30]. The discrepancy in the number of measured runs comes from the fact that Phantom and Skywalker have a different number of operating modes. We were unable to identify any difference in the measurements per drone type, i.e. no (if any existing) effect of the operating modes. We confuse therefore the individual measurement runs.

Signal Sources			
	Skywalker	Phantom 2	Background
Nr. meas. runs	3	5	2

Table 4.1: Number of measurements per signal sources. Each source was measured for 5 s.

The measured signals were stored as binary files and later processed and stored as spectrograms with FFT sizes of 512 hanning windowed samples, each burst gets therefore mapped to a time stamp separated by 125 µs. Choosing as spectrogram size of 40 time stamps × 512 frequency bins, one measurement run resulted in 1000 files. Figure 4.2 shows an extract of the measurement runs. We choose this format such that it fulfills the requirements we set to construct the ML data, which will be described in Chap. 5.

The signal sources we measured are characterized in [25] and the findings are briefly recapitulated. The results discussed there prepare the setup for the synthetic data generation.

4.2 FHSS Signal Analysis

Table 4.2 summarizes the findings of [25], using the terminology established in Sec. 2.3.1. The channel bounds $f_{\min}$ and $f_{\max}$ determine the hopping band. Even though the FHSS operates in the 2.4 GHz ISM-band, the effectively used spectral band is considerably smaller, justifying our previous choice of $f_s = 80$ MHz during the measurements. Additional to the extracted parameters, a proposition for the pattern types was made. The hop sequence follows the shifting-l2r (or shifting-r2l) rule,

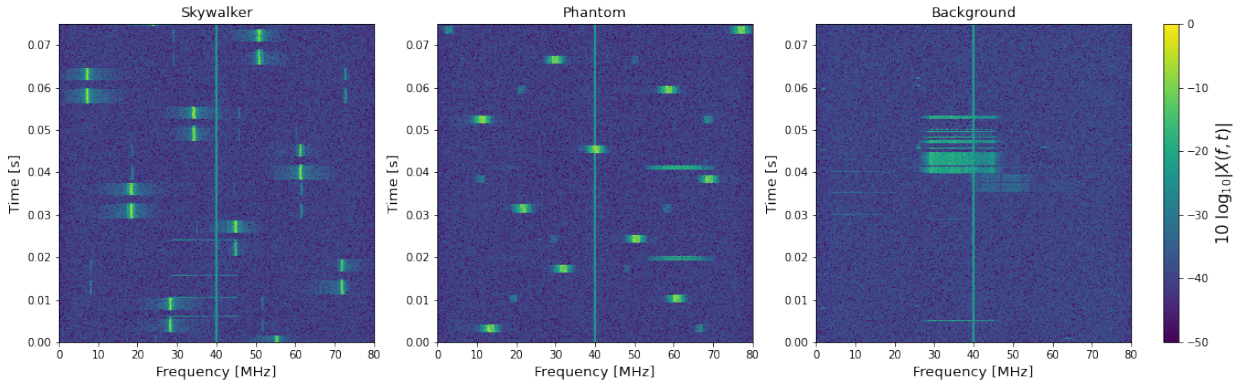


Figure 4.2: Measurement outcomes for the three considered signal sources represented in the base band. As expected, high SNRs are observed. In the Phantom measurement, one observes also interference signals (wide band signals centered at around 62 MHz).

FHSS parameters

	Δt [µs]	Δt_{pause} [µs]	Δf [MHz]	C_{r}
Phantom	1777.0	5200.0	2.565	1
Skywalker	3127.0	3127.0	0.680	2

Pattern parameters

	f_{shift} [MHz]	Ch. bounds [MHz]	K	pattern type
Phantom	26.56	5.5 / 77	36	shifting-l2r
Skywalker	27.40	6 / 75.3	44	shifting-r2l

Table 4.2: Characterization of Skywalker and Phantom drones [25], where the parameter were defined in Sec. 2.3.1. The Ch. bounds describe respectively the lower and upper channel bounds of the pattern. Pattern type describes the rule which the hopping scheme follows.

where l2r (or r2l) stands for left-to-right (or right-to-left). This rule assumes that f_{shift} is constant during the FHSS period. If the signal hops outside the hopping band, determined by the extremal frequencies, it reenters the hopping band from the other side by the amount it still requires to shift a total amount of f_{shift}. In the case of shifting-l2r

$$f_{k+1} = \begin{cases} f_k + f_{\text{shift}} & \text{if } f_k + f_{\text{shift}} \leq f_{\max} \\ f_k + f_{\text{shift}} + f_{\min} - f_{\max} & \text{if } f_k + f_{\text{shift}} > f_{\max} \end{cases}, \tag{4.1}$$

with $f_{\min}$ ($f_{\max}$) the lower (upper) channel bound. This pattern is also illustrated in Fig. 4.3.

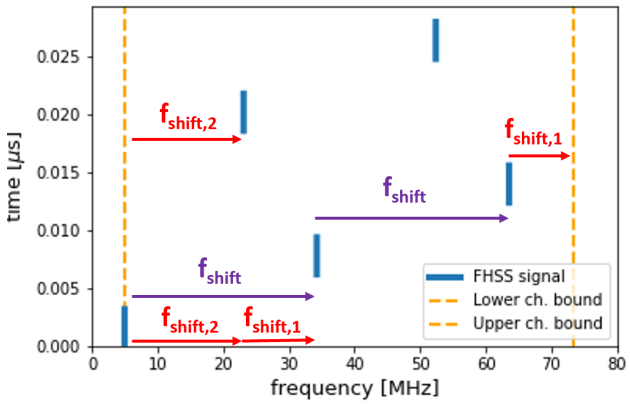


Figure 4.3: Illustration of a shifting-l2r pattern. By periodic boundary conditions, a hop can exit the bounds and reenters on the opposite bound. The total hop distance f_{shift} is thereby maintained. Note that both channel bounds get by definition necessarily addressed during the hopping sequence.

The presented characterization enables us to define the relevant pattern and FHSS parameters needed for the data synthetization.

4.3 FHSS Signal Simulation

Having established and determined the relevant FHSS parameters in previous chapters, we proceed to the generation of artificial data mimicking the signal appearances discussed in Sec. 4.2. This enables us to diversify our training dataset with regard to different SNRs, patterns and FHSS parameters. This diversity shall guarantee that FHSS signals in different contexts can be identified and predicted. The next section gives a detailed description of the simulation setup for our synthetic FHSS data. The further processing of ML data will be explained in section 5.1.1.

As stated previously, we discuss only signals in the frequency range from 2.4 GHz to 2.48 GHz. We perform the simulations in the base band from 0 to 80 MHz with a time resolution of $f_{\text{s}}^{-1} = 12.5$ ns, i.e. with the same sampling frequency as used in the measurement setup, i.e. $f_{\text{s}} = 80$ MHz.

4.3.1 Simulation Setup

The hopping pattern is the core of each FHSS implementation. Since both receiver and transmitter need to share knowledge about the pattern, we constructed a sim-

ulation based on a preknown pattern, i.e. not a pattern generated on the fly during an established communication link. In the simulation we differentiate between the **pattern construction** and the **signal construction**. While the latter requires pattern and FHSS parameter, the pattern construction does not necessarily need to follow a specific scheme. In our case the pattern construction aims to imitate the described patterns in Sec. 4.2 and is therefore dependent on the pattern parameter. Starting with the pattern construction we give a detailed explanation of the simulation setup.

1. **Pattern construction**

   We implement the two types of patterns identified in the rerecorded signals, namely both shifting-r2l and shifting l2r pattern[2]. To construct a shifting pattern channel sequence we define a number of target channels in the pattern, both lower and upper channel bounds, i.e. extremal channel frequencies, and the desired frequency shift f_{shift}. The step-wise process for the FHSS channels construction is described as follows:

   a) **Grid construction:** The first step consists of creating a channel grid with the aforementioned channel boundaries by defining a number of equispaced grid points n. Each grid point represents a channel candidate appearing in the pattern sequence.

   b) **Grid mapping:** In the second step, the starting frequency of the pattern and f_{shift} are mapped to this grid, i.e. both transformed from MHz to a grid position and to a number of grid points, respectively. The pattern is created according to the shifting-l2r or shifting-r2l rule as explained before until the starting channel is reached. This procedure introduces a discrepancy between the desired frequency shift and the observed frequency shift due to the mapping to the grid structure. Furthermore, the number of obtained channels in the pattern is a priori unknown, it is however finite. By defining a rejection tolerance both for desired vs. observed f_{shift}, and for the number of desired channels in the pattern, one can iterate over various n obtaining different patterns and reject those too far from the desired pattern. This method proves to be efficient and allows the creation of various patterns with the specified pattern type, while assuring only reasonable pattern lengths. Figure 4.4 illustrates the idea behind the grid method.

   c) **Channel build:** The final step of the pattern creation consists of mapping the channel sequence in terms of grid points back to the frequency

[2] In fact shifting-r2l is just a shifting-l2r but time reversed.

domain, obtaining $(f_i)_{i=1}^{K}$. Since the starting frequency is arbitrarily chosen, one can further apply a cyclic permutation to the channel sequence. For the sake of simplicity and without loss of generality, we focus on the previous described pattern structures. However, one could also vary channel bounds (and define likewise a tolerance criterion to obtain an even better match to a certain pattern).

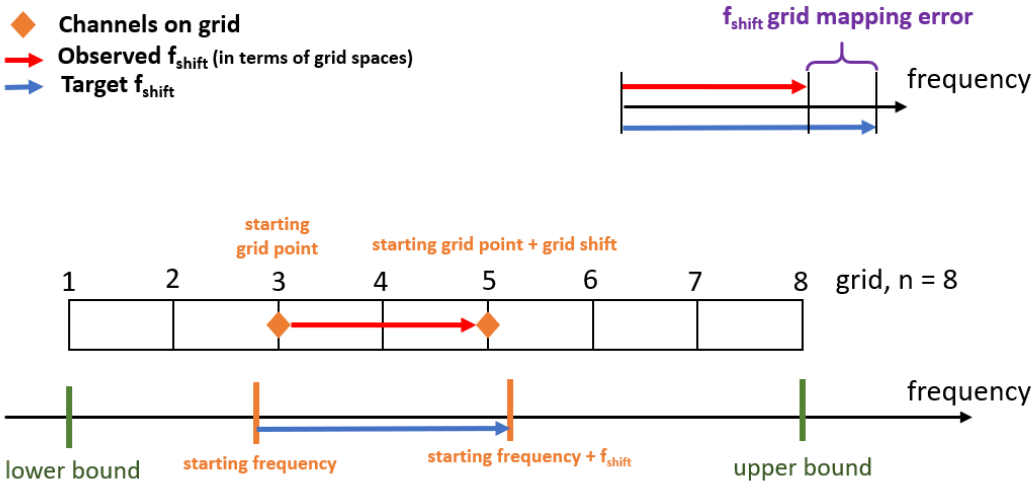


Figure 4.4: Grid method. $f_{\text{shift,target}}$ (blue) gets approximated by $f_{\text{shift,obs}}$ (red). The carrier is thus restricted to hop only on the channel grid.

2. **Signal construction**

   Based on a set of channel frequencies in time, we proceed to the signal construction, which is independent of the pattern creation method. The signal construction can be decomposed in the three following steps.

   a) **Modulation**: Once the pattern is defined we choose a set of FHSS parameters defined by $(\Delta t, \Delta t_{\text{pause}}, C_{\text{r}})$ as well as the total simulation length and the time resolution[3], in order to construct an array with addressed channel frequencies in time $f(t)$. This can be written as,

$$f(t) = \begin{cases} f_k & \text{if } t \in \Omega(k; C_{\text{r}}, \Delta t, \Delta t_{\text{pause}}) \\ * & \text{otherwise} \end{cases}, \tag{4.2}$$

[3] As discussed above, we are interested in signals localized in well defined regions and treat the time resolution as fixed parameter, it can however be changed if one aims to simulate signals in different frequency bands.

with f_k, the k^{th} channel in $(f_i)_{i=1}^{K}$ and

$$\Omega(k) = \bigcup_{c=1}^{C_\text{r}} \Big[\big((k-1)C_\text{r} + c - 1\big)(\Delta t + \Delta t_\text{pause}), \\ \big((k-1)C_\text{r} + c\big)\Delta t + \big((k-1)C_\text{r} + c - 1\big)\Delta t_\text{pause}\Big],$$

which describes the time intervals when the k^{th} channel is activated. Any dummy value $*$ can be used for all times when no channel is addressed, i.e. the signal source is not transmitting. Equation (4.2) gets periodically repeated with periodicity T, cf. Eq. (2.8).

The unmodulated signal is then simply

$$x(t) = \begin{cases} e^{2\pi \mathrm{i} t f(t)} & \text{if channel active} \\ 0 & \text{otherwise} \end{cases}. \tag{4.3}$$

The FHSS parameter Δf gets indirectly defined over the bit rate R_b of the bit sequence transmission. To introduce some transmitted information in our signal we generate a random square wave $m(t)$ with values -1 and 1 with the given bit rate and 0 as dummy value in case of the inactive signal, $t \notin \Omega(k)$. We implemented a binary FSK, the modulation technique described in Sec. 2.1, and the complex version of the modulated signal becomes then

$$x(t) = \begin{cases} e^{2\pi \mathrm{i}\left(t f(t) + k \int_{-\infty}^{t} m(t')dt'\right)} & \text{if channel active} \\ 0 & \text{otherwise} \end{cases}, \tag{4.4}$$

where we fixed the modulation parameter k to 0.5. Recall that R_b and Δf are related via Eq. (2.4).

b) **Noise addition:** In order to get a more realistic synthetic signal we need to add some sort of noise, we choose additive white Gaussian noise (AWGN), which is a common assumption for any system under study. We use complex AWGN since the signal is complex and its power density is defined only by one statistical parameter, its variance. While the signal power is confined in a small spectral region, the noise spreads over the total frequency range under observation, so we cannot use Eq. (2.12) to obtain the necessary noise power to be added to the signal for a given SNR (in dB) and therefore we need to rearrange it in terms of power density instead of total power, as

$$P_\text{noise} = P_\text{signal} \frac{\text{BW}}{\Delta f} 10^{-\frac{\text{SNR}_\text{dB}}{10}}, \tag{4.5}$$

where BW = 80 MHz is the bandwidth under observation and $P_{\text{signal}} = 1$ W by construction. As the name suggests, AWGN is sampled from a Gaussian distribution with zero mean and variance given by P_{noise}. The noise $n(t)$ is then added to the modulated signal from Eq. (4.4) as $s(t) = x(t) + n(t)$.

c) **Processing:** The resulting simulated signal $s(t)$ is a time signal with a time resolution of f_{s}^{-1}. To reduce the number of sample points, we resample the modulated signal in a similar way to the measurement procedure, namely keeping 512 samples every 10000. The resulting time signal blocks are processed and stored as spectrograms in the same way as described on page 20.

The presented simulation procedure consists of the following steps: pattern creation, channel/frequency in time creation, modulation, noise implementation, processing composed of resampling and Fourier Transform for a spectrogrammic representation. Based on the described setup, we perform a variety of simulations which are presented in the following section.

4.3.2 Simulated Data

To obtain a diverse dataset, we simulate FHSS signals similar to those identified in Sec. 4.2 with Python. The extracted FHSS parameters and pattern parameters build the reference for our parameter choices. We choose two simulations stacks corresponding to the Phantom 2 and Skywalker drone found in [25], the respective stacks are called phantom-like and skywalker-like.

For each drone, the pattern type, number of hop repetitions, the upper and lower frequency bounds and the number of channels in the patterns are considered invariant. We introduce variations in Δt, Δt_{pause}, Δf and f_{shift} in form of deviations of +/-10%, +/-20% w.r.t. the nominal value. On the one hand, this diversity should ensure that we imitate a certain signal even though its parameters were not correctly extracted by covering immediately a wider range of the parameter space. On the other hand, we aim to have a broader range of FHSS signals of such types covered, such that our ML model learns not only one particular FHSS signals.

The choices of f_{shift} generates each different pattern as explained in Sec. 4.2. Since we considered the number of channels in a pattern as invariant, we accept only patterns with channel numbers stated in Tab. 4.2. For the observed shifting frequency $f_{\text{shift,obs}}$, we defined a tolerance of 1 MHz, which corresponds approx. to a 1.4% deviation w.r.t the hopping band width for both Phantom- and Skywalker-like hopping schemes. We considered grid points n between 27 and 100, the acceptance tolerance for f_{s} was set to 1 MHz, target number of channels in pattern was 36 /

44 respectively with zero tolerance. The lower and upper channel bounds were set to 5.5 / 77 [MHz] and 6 / 75.3 [MHz] respectively. The outcomes, i.e. observed and thus in the simulation used f_{shift}, are presented in Tab. 4.3.

Phantom-like

$f_{\text{shift,target}}$ [MHz]	$f_{\text{shift,obs}}$ [MHz]	n
22.88	22.47	36
25.74	26.18	72
	26.56	36
28.60	-	-
31.46	-	-
34.32	34.24	36
	34.70	72

Skywalker-like

$f_{\text{shift,target}}$ [MHz]	$f_{\text{shift,obs}}$ [MHz]	n
20.42	20.71	88
	20.95	44
22.98	-	-
25.53	-	-
28.08	27.40	44
30.64	30.27	88
	30.62	44

Table 4.3: Implemented and obtained frequencies with grid method to obtain a shifting-l2r (used in the Phantom-like stack) and a shifting-r2l (used in the Skywalker-like stack) pattern.

With this setup we obtained 5 different patterns for both Phantom and Skywalker-like signals. In both cases, two target frequencies produced two different f_{shift}, which are consequently spaced < 1 MHz apart. This introduces necessarily some imbalance in the dataset since some frequency ranges of f_{shift} are preferred. We note that not every target shifting frequency $f_{\text{shift,target}}$ creates a pattern with the desired properties. The fact that the nominal shifting frequency was not matched in both patterns was considered to be of no particular concern for a diverse dataset.

The obtained $f_{\text{shift,obs.}}$ are used in the simulations. Additionally to the different parameter choices, each simulation run is repeated for 5 SNR values from 5 dB to 25 dB in steps of 5 dB, resulting in $5^5 = 3125$ runs per simulated drone. The respective numerical values used in the simulation setup can be found in Tab. 4.4 and Tab. 4.5.

Each simulation run was setup for 0.42 seconds to guarantee the simulation of one full FHSS period, which was achieved in 99.6 % of the cases. With that, no drone type or parameter sets gets underrepresented in the data. The outcome of each run is saved as small consecutive spectrograms of 40 time stamps $\times$ 512 frequency bins. With a simulation time of 0.42 s and time stamp length of 125 µs, we obtain therefore a total of 84 time-frequency spectrograms.

We compare the effect of different SNR levels and different parameter choices in Fig. 4.5 and Fig. 4.6 respectively. Fig. 4.5 compares Phantom-like simulations under different SNR levels, with the same simulation parameters, note that for 5 dB the signal becomes barely distinguishable from the noise floor. In Fig. 4.6(A) and Fig.

Phantom-like sim. parameter

Δt [µs]	Δt_{pause} [µs]	Δf [MHz]	f_{shift} [MHz]	SNR [dB]
1422	4160	2.052	22.47	5
1599	4680	2.309	26.18	10
1777	5200	2.565	26.56	15
1955	5720	2.821	34.24	20
2132	6240	2.990	34.73	25

lower/upper channel bound [MHz]	C_{r}	Nr. channels
5.5 / 77	1	36

Table 4.4: Phantom-like simulation parameters with shifting-l2r pattern, resulting in $5^5 = 3125$ simulation runs. f_{shift} is obtained via grid-method, cf. Tab. 4.3.

Skywalker-like sim. parameter

Δt [µs]	Δt_{pause} [µs]	Δf [MHz]	f_{shift} [MHz]	SNR [dB]
2501.6	1052.0	0.544	20.71	5
2814.3	1183.5	0.612	20.95	10
3127.0	1315.0	0.680	27.40	15
3439.7	1446.5	0.748	30.27	20
3752.4	1578.0	0.816	30.62	25

lower/upper channel bound [MHz]	C_{r}	Nr. channels
6 / 75.3	2	44

Table 4.5: Skywalker-like simulation parameters with shifting-r2l pattern, resulting in $5^5 = 3125$ simulation runs. f_{shift} is obtained via grid-method, cf. Tab. 4.3.

4.6(B) the same Δf and f_{shift} are used, while Fig. 4.6(C) and Fig. 4.6(B) share the same Δt and Δt_{pause}. Note that in this time frame, due to the differences in hop duration and pause, three complete hops are present in Fig. 4.6(A) while only two in the others. Observe also that f_{shift} is considerably greater in Fig. 4.6(C) compared to the others, resulting in three hops before exiting the hopping band.

For each individual set of parameters we generate additionally 15 independent noise-only spectrograms with the corresponding noise power of the simulation. The small noise-only spectrograms, of same dimensions as the signal ones, are necessary to reduce a bias in the signal predicting algorithm, such that it does not inevitably

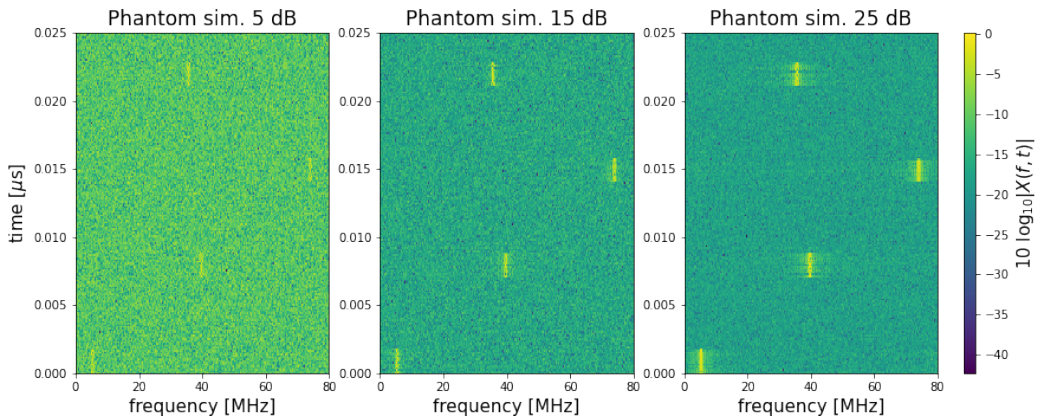


Figure 4.5: Spectrogram of simulated phantom-like signal example for different SNRs. Simulation parameters are $\Delta t = 1777$ µs, $\Delta t_{\text{pause}} = 5200$ µs, $\Delta f = 2.565$ MHz, $f_{\text{shift}} = 22.47$ MHz, see Tab. 4.4.

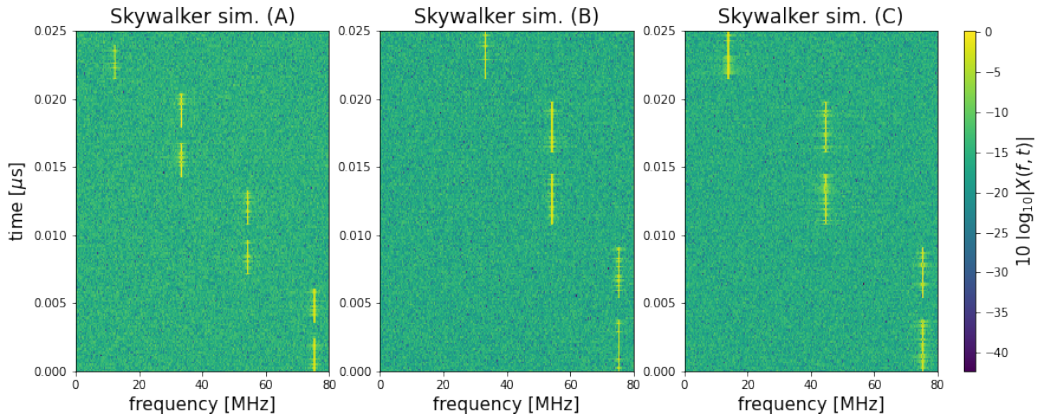


Figure 4.6: Spectrogram of simulated Skywalker-like signals at 25 dB, comparing different simulation parameters. (A) $\Delta t = 2501.6$ µs, $\Delta t_{\text{pause}} = 1052$ µs, $\Delta f = 0.544$ MHz, $f_{\text{shift}} = 20.95$ MHz. (B) $\Delta t = 3752.4$ µs, $\Delta t_{\text{pause}} = 1578$ µs, $\Delta f = 0.544$ MHz, $f_{\text{shift}} = 20.95$ MHz. (C) $\Delta t = 3752.4$ µs, $\Delta t_{\text{pause}} = 1578$ µs, $\Delta f = 0.816$ MHz, $f_{\text{shift}} = 30.62$ MHz, see Tab. 4.5.

predict signals, even if there are none. Also, they will be useful in order to get a reference score for the ML outcome.

This chapter described how we acquired first measurement data, exploited the analyses presented in [25], which subsequently enabled us to build a simulation framework which we used to create a diverse dataset. The simulation setup is based on the theoretical concepts described in Chap. 2 and on a similar implementation

as described in [27]. The data treatment for both synthetic and measured data is the same, which makes them directly comparable.

Before feeding the gathered data into a ML algorithm, it needs to undergo some preprocessing, where we will make only slight discriminations between simulated and measured data. The next chapter explains the supervised learning approach to our detection problem, starting with data preprocessing.

5 Supervised FHSS Pattern Prediction

Having established the fundamentals in FHSS and acquired a diverse dataset we aim now to construct a ML model with the goal to predict the frequency evolution of a signal in a certain time window based on the time-frequency representation in the observation frame. Recall Fig. 3.1 for the proposed framework. We focus here on supervised learning where a set of (possibly multidimensional) input pairs $\{(X_i, y_i)\}$ is needed for the training process. While the X_i represent the inputs, called features, the y_i are the target values or labels. The aim in supervised learning is to optimize the internal parameters of a certain model in order to predict y for a new input X, we denote the output by $\hat{y}$ and the underlying target (ground truth) by y.

This chapter is structured as follows: In Sec. 5.1 we describe in detail the data preparation and preprocessing. Section 5.2 gives a theoretical and practical description of the training setup as well as the model construction. The last Sec. 5.3 is devoted to the achieved results and evaluation of the performance in different test scenarios.

5.1 Data Preparation

This section describes the construction of the ML data, i.e. of the input samples X and the corresponding target y.

The natural representation of (X, y) data is, in this context, a spectrogrammic one, with dimensions determined by the number of time stamps and frequency bins. The spectrogram can be viewed as a one channel picture[1] with respective pixel dimensions. The number of pixels, and consequently the dimensionality of both input and output spectrograms, depends on the desired resolutions in time and frequency and on the observation/prediction time window. We choose an observation

[1] Digital pictures (images), have typically 3 color channels, namely red, green and blue. Each channel gives the intensity of the respective colour on the pixel grid composing the picture. A picture with only one channel represented in greyscale would be associated to a black and white picture. The term channel in this context should not be confused with the channels from an FHSS pattern.

window of 50 ms to predict the following 25 ms of the FHSS. In the spectrogrammic framework presented previously in Chap. 4, this corresponds to 400 and 200 time stamps respectively. These time lengths are based on the measured and simulated time scales of different FHSS signals, such that one observes at least one signal hop in the output sample y. An ad hoc choice was made for the observation window length, such that the appearance of three signals is guaranteed, even for long hop durations/hop pauses.

We discuss the next procedure of the ML data preparation on the simulated data, which is mostly identical to those from the measured data. The key differences are highlighted at the end of next section.

5.1.1 Preprocessing

With these choices for the observation and target window, 69 (X, y) samples are obtained by merging 10 consecutive small spectrograms (to obtain X) as well as the following 5 smaller spectrograms (to obtain y) using a sliding window with length of one spectrogram as illustrated in Fig. 5.1.

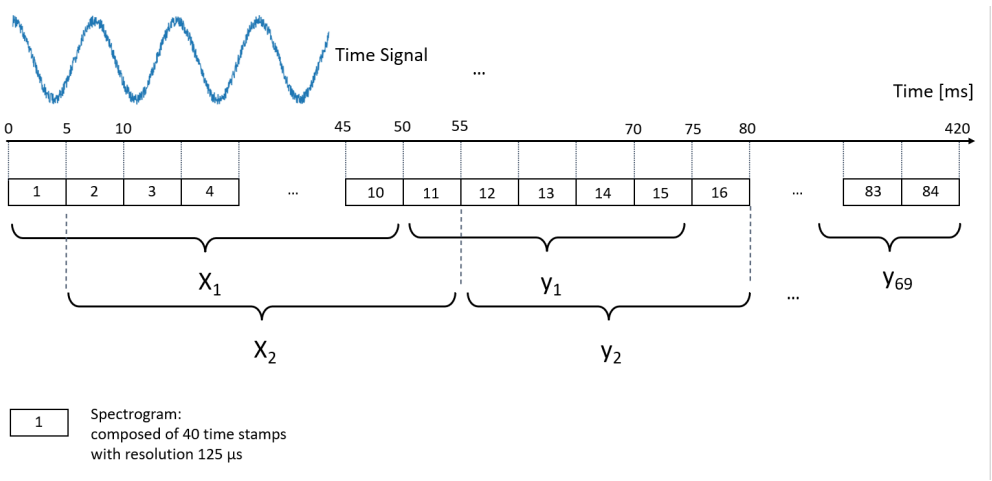


Figure 5.1: Construction scheme of (X, y) samples from a (0.42 s) simulation run.

Additionally, the 15 (small) noise spectrograms from each simulation run are shuffled randomly and represent thus, in the same way as the signal samples, a pure noise (X, y) sample. This process is repeated 10 times, resulting in 10 independent noise samples[2] additional to the 69 signal samples. Per drone type (Phantom-like, Skywalker-like), we generated an overall amount of 215625 signal data pairs and 31250 noise data pairs distributed uniformly among the considered 5 SNR levels.

[2]Since 15! different combinations of creating a particular noise sample exist, it is highly unlikely to sample two times the same.

With 512 frequency bins, the model input dimension becomes 204800 while the output dimension results in 102400. While high dimensional input data is not unusual in ML, this can pose a problem when paired with likewise high dimensional target samples since the model complexity, i.e. number of model parameters, scales[3] among others with the size of X and y. For this reasons, we construct spectrograms with dimension $80/40 \times 256$ by averaging over adjacent signal pixels while still maintaining the time windows of 50/25 ms. Retrospectively, the ML data dimension should have been considered already in the measurement and simulation process alike. The considered time and frequency resolution of the samples become therefore 625 µs and 0.3125 MHz. For each simulated drone type and each SNR level, the samples get normalized with respect to their maximum value (signal and noise samples alike), such that each pixel value $\in [0, 1]$. This data setup in itself should be sufficient to train and test a model, the target samples y however, as they were presented here, have a technical and conceptional drawback, which can be overcome by a suitable annotation.

5.1.2 Annotation of Dataset

The main drawback is that the model would need to learn a representation of the target y by matching all 40×256 pixel, with values $\in [0, 1]$. However, most of the pixels are of no interest since they represent only the noise floor. The model would thus need to learn Gaussian noise, but since it is inherently of stochastic nature, one could possibly never achieve a perfect score, i.e. a perfect mapping of $\hat{y}$ to y. Furthermore, we do not even aim to "learn" the underlying noise, but only to predict signal areas and therefore we should build the model in such a way. Optimization of a proper loss function might thus be difficult, the concept of loss functions and its importance for supervised learning will be discussed in Sec. 5.2.1. To overcome this problem we propose an annotation method for signal pixel areas in y. The annotated y samples therefore represent the actual used labels, to which we refer from now on as y. The annotation procedure involves a signal detection part followed by a hop rectification. The signal detection consists of a two step thresholding scheme as it was performed in [25].

The first threshold is used to cut out noise and the second to find the target signals. The signal detection scheme requires two user defined input thresholds δ_1, δ_2 and computes y time stamp-wise. The first threshold sets every value in the time stamp (τ) to zero which is $< \delta_1 \text{median}(\tau)$, the second threshold, sets every frequency

[3] In a very simply constructed neural network consisting only of one dense layer, one has already $\sim 10^{10}$ parameters, which exceeds our hardware capacity, which is constrained to 10^9 parameters. Changing the architecture to a hourglass type reduces the number of model parameters but brings other challenges, and would need a detailed understanding of autoencoder type models.

bin in τ to 1 or 0 depending on whether $\delta_2 \max(\tau)$ holds true or not. With the right parameter choices, this scheme proves to be robust in detecting signals while not detecting any noise. Table 5.1 summarizes the chosen parameters δ_1, δ_2.

	Phantom Meas.	Phantom Sim.	Skywalker Meas.	Skywalker Sim.
δ_1	90	2.2	125	2.1
δ_2	0.5	0.5	0.25	0.25

Table 5.1: Annotation parameter choice for the measurements and for simulation. δ_1 is the scaling factor of the median per time stamp to filter out noise, δ_2 is the scaling factor for the maximum value per time stamp, to detect the signal of interest.

The thresholds were not optimized for every simulation parameters, notably they are the same regardless of the SNR level. The choice of δ_1 depends on the underlying data. We choose the same δ_2 for both measurement and simulation data for each drone typ, $\delta_2 = 0.5$ defines the common full width at half maximum definition. For the Skywalker drone, since the signals are notably more narrowbanded, we used however $\delta_2 = 0.25$, such that we obtain a wider target area.

The resulting binary matrix encodes signal regions with 1 and noise regions with 0. In order to obtain a well defined signal region in terms of Δf and Δt we rectify the signal areas. Examples of detected signal pixels and the rectified version of the spectrogram are shown in Fig. 5.2.

The rectifier function assumes that inside a hop, the signal pixels are connected in time. In particular, a hop with missed signal pixels in one time stamp would be considered as two separated hops[4]. We make the assumption that these cases occur only rarely and are negligible in the amount of data. Different from the time extension of the hop, the centre frequency and the spectral width are averaged over the hop. The signal area gets thus replaced by the rectified version. Similar annotation (correction) was performed also in [24], [31]. Only signal samples undergo the annotation process, the noise y samples are only zero matrices.

5.1.3 Data Processing for Measurement Data

The processing of the measurement data was performed analogously, 15 small saved spectrograms were stepwise merged into a (X, y) sample, we obtain therefore 985 samples per measurement run. We constructed three datasets, corresponding to

[4] In fact, disconnected pixels are recognized as being part of the same hop if they lie in a certain range around the hop centre, but as stated, missing pixels in one τ would result in the end of one block.

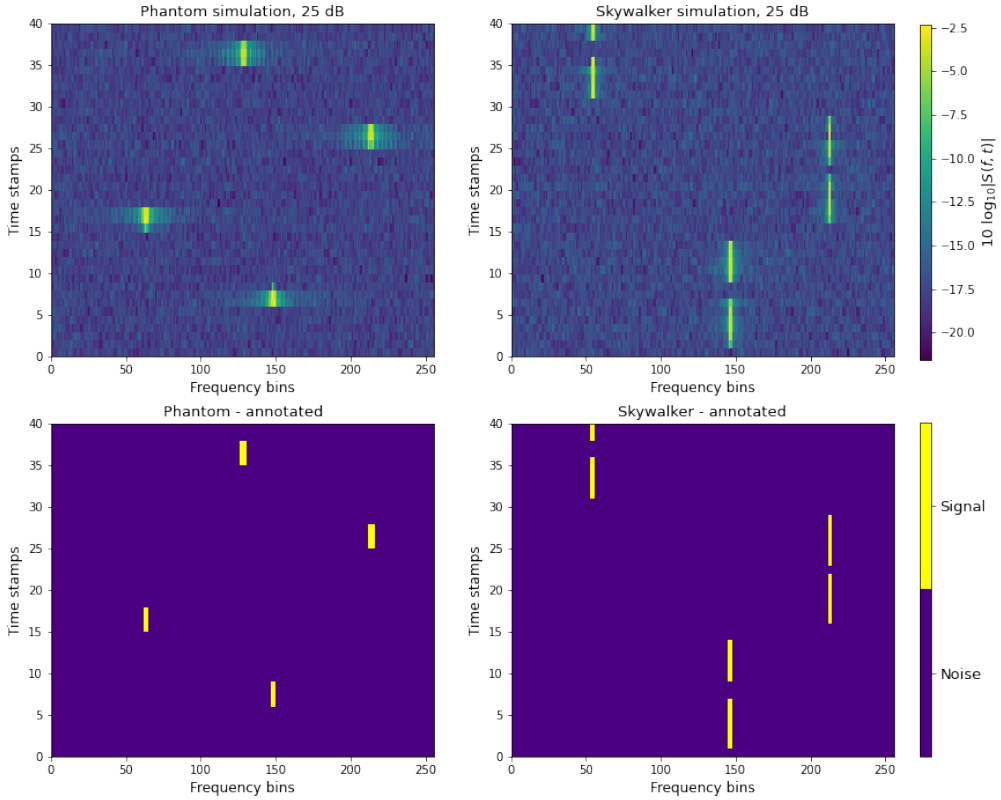


Figure 5.2: Annotation outcomes on 25 dB simulated data for Phantom and Skywalker. The annotated spectrogram is a binary matrix (1 for signal and 0 for noise/no-signal).

the Phantom-, Skywalker-, background noise measurements respectively. All three combined consist therefore of 80% signal (out of which 62.5% are attributed to Phantom) and 20% noise samples. Note that in contrast to the simulated data, the noise-only samples from the (background) measurement contain occasionally interferences (like the signal data). Dimensionality reduction was performed as well as normalization w.r.t. the maximum value, and annotation[5] where we expect signals of interest, i.e. not in the background noise measurements.

Figure 5.3 shows two y samples from the Phantom and Skywalker measurements obtained by the methodology described above. The thresholds show to be robust

[5]During the annotation, we found an loophole in the rectification code. The loophole consisted in failure of the rectification if the signal had a mean width of one pixel. This limit case occurred only during the annotation of the Skywalker measurement data.

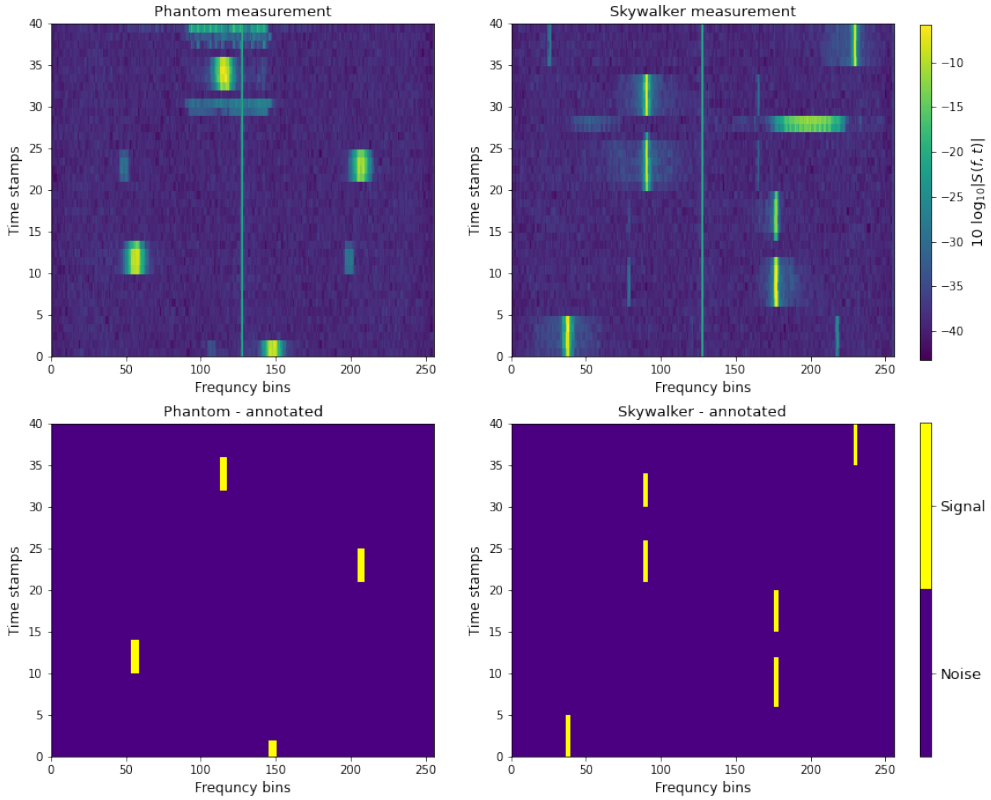


Figure 5.3: Annotation outcome on measurement data for Phantom and Skywalker. The annotated spectrogram is a binary matrix (1 for signal and 0 for noise/no-signal). Interferences (present in both measurements) do not get detected.

enough to not falsely detect interference signals. In the case of Skywalker annotation we annotated occasionally interference signals. Since all annotated signals of interest had a width of less than five pixels, we sorted out any annotated signal blocks with a width ≤ 5.

5.2 Machine Learning Setup

The setup we present here is a multivariate binary classification problem. The multiple class instances represent thereby the upcoming spectrogram which can be seen as a form of (annotated) image extension. It can also be seen as a form of image

recognition, since signal areas need to be identified, information extracted, and then instances classified.

Convolutional neural networks (CNNs) are a common choice for model architectures in image recognition problems. We focus therefore on neural networks and more specifically on CNNs. The next section establishes shortly the concepts of neural networks, loss function, the idea of backpropagation as well as the mechanism of CNNs. We further describe our model architecture construction and the training process of our model.

5.2.1 Concepts of Neural Networks

Deep learning is a terminology associated with any type of neural networks, CNNs being an example of it. Neural networks are typically layer structured constructs. Each layer has a certain number of nodes, called neurons. In the case of feedforward networks, the neurons of one layer are only connected to neurons in adjacent layers (in contrast to recurrent networks where cyclic connections are possible). ML techniques aim to obtain a trained model which processes an input (feed into the input layer) through the network, i.e. activates the neurons in such a way that the output layer gives the desired result. Between the required input and output layer, an arbitrary number of so-called hidden layers can be placed. Increasing the number of layers (depth) and/or the size of the layers (width) increases the models complexity. Figure 5.4 visualizes a five layer neural network, with input dimension n and output dimension k. In the case of k-variate binary classification problem; the k output neurons would be associated with probabilities of belonging either to class 1 or class 2.

In order to obtain probabilities (in the binary classification case), the output values are passed to an activation function which maps $\mathbb{R} \to [0, 1]$, the natural choice is

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \tag{5.1}$$

the sigmoid function [33]. In the last layer, the need for an activation function is given by the interpretation we accord to the neuron's output. Generally, different activation functions can be used at any time in the network (elementwise on the neurons of a layer) to introduce non-linearities in the neurons' activations and therefore in the model itself. These activation functions need to be differentiable, at least mostly everywhere, in order to train a model through backpropagation[6].

The amount of information (activation) passed through the network is determined by the weights stored in the weight matrix W, associated with the

[6]Backpropgation will be discussed in a moment.

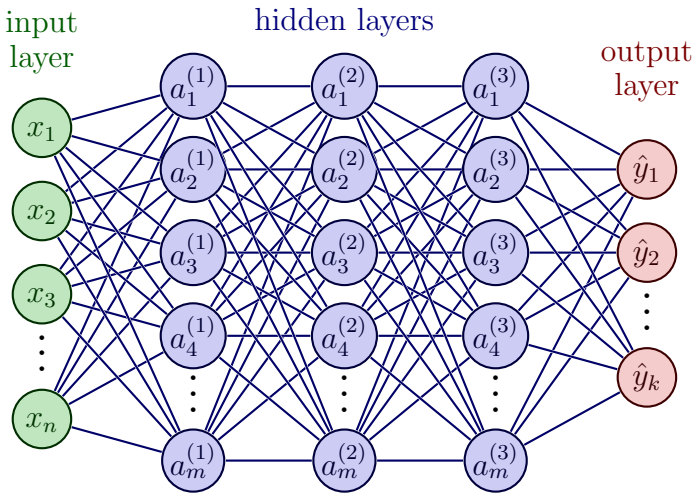


Figure 5.4: Schematic structure of a dense neural network. The green layer of size n takes the feature vector x. The information gets processed through the network along the edges between layers associated with weight matrices $W^{(i)}$ $(i = 1, 2, 3, 4)$ until the output layer with output vector $\hat{y}$ of size k. Figure adapted from [32].

edges in the graph in Fig. 5.4. A neuron's activation is calculated by linear combination of these weights with the previous nodes values[7]. Mathematically, the layer values can be described as vectors, so that the output $a^{(l)}$ of a layer l is calculated by a matrix-vector multiplication, $W^{(l)}a^{(l-1)} = a^{(l)}$, with weight matrix $W^{(l)}$.

The network's weights are the trainable parameters. Weight optimization is therefore the process which the network undergoes during the training stage. In supervised learning, training is performed with the training dataset $\{(X_{i,\text{train}}, y_{i,\text{train}})\}$ a subset of the entire dataset. The test dataset $\{(X_{i,\text{test}}, y_{i,\text{test}})\}$, on the other hand, is used only after training to assess the performance of the (optimized) model. Feeding X_{train} into the network results into an output $\hat{y}$ which is used together with y_{train} to optimize a suitable loss function L, depending on the networks weights. A common choice of loss function in binary classification problems is the binary cross entropy (BCE). Averaging over single loss values, the k-variate BCE then reads

$$L(y, \hat{y}) = -\frac{1}{k}\sum_j y_j \log \hat{y}_j + (1 - y_j) \log (1 - \hat{y}_j)) \tag{5.2}$$

where $y_j \in \{0, 1\}$, the class labels of the k-nominal sample y and $\hat{y}_j = \sigma(z_j)$, with $z_j = \left(W^{(4)}a^{(3)}\right)_j$, the j^{th}-component of the last weight matrix multiplied by

[7] And as mentioned before, sometimes squished in some activation function.

the output vector of the last hidden layer. Since the activation of every layer[8] is dependent on the previous weights, Eq. (5.2) is in fact a function of all model weights (all weight matrices $W^{(i)}$ between the layers). Using the chain rule we can propagate the loss error backwards in the neural network and subsequently optimize each weight, this process is called backpropagation. The minimization of the loss function is performed in the high dimensional weight space by a numerical optimization scheme, with gradient descent (GD) based schemes. Gradient descent schemes are iterative methods on the whole parameter space where the next parameter point w_{t+1} is determined by

$$w_{t+1} = w_t - \eta \nabla_w L, \tag{5.3}$$

with learning rate $\eta in \mathbb{R}$. The learning rate is the step size, terminology mostly utilized in the ML community. The loss $\nabla_w L$ may be calculated by the entirety of the training set. If w_{t+1} is determined for a single sample in X_{train}, one speaks of stochastic gradient descent (SGD). If batches (subsets of the training data) are used, one calls it mini-batch gradient descent[9]. More elaborate schemes exist such as the Adam algorithm [34]. Optimizing a model requires multiple iterations through the dataset. These iterations through the entire training set are denoted by epochs.

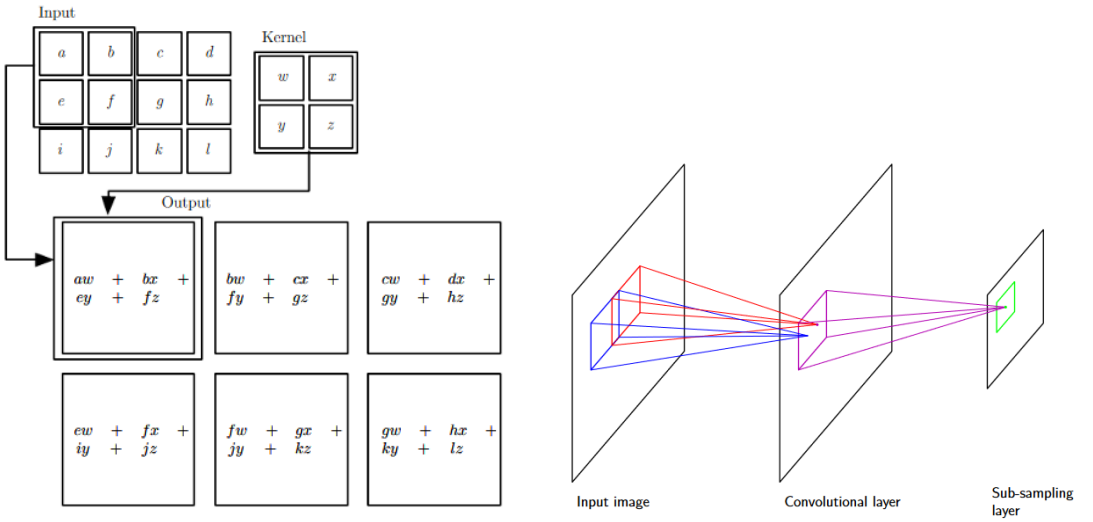
CNNs are neural networks with additional convolutional layers, often paired with pooling layers[10]. Convolutional layers make use of set of kernels (filters). In the 2D case, these kernels can be seen as 2D-matrices, sliding over the input and yielding a convoluted output value. Figure 5.5 illustrates the effect of a 2D convolutional layer. As shown in the examples, this can lead to a dimensionality reduction, which can be a desired side effect. Padding can be used around the edges to ensure the same or even higher output dimension, [36] gives an overview of the underlying convolutional arithmetics. Pooling layers function in the same way but are, unlike convolutional layers, untrainable. Their only effect consists in reducing the features dimension according to a certain defined rule, examples are maximum and average pooling.

[8] Except the input layer.

[9] SGD is "stochastic" since only one random training sample is used, the assumption is that the gradient's direction is a good-enough approximation of the true GD over the whole dataset. Likewise, mini-batch gradient descent is also stochastic, but a compromises between computational efficiency of SGD and approximation accuracy w.r.t. to the true ∇L (calculated with all training samples).

[10] We stick here to the convention of that a convolutional and a pooling layer are two separated layers, although they can be viewed as one layer consisting of multiple stages, which is in its whole called convolutional layer. See [35] for the detailed reasoning in terminology.

The last topic we want to mention here is the problematic of overfitting, common to all ML types. Overfitting describes the situation when the model minimizes a certain loss function during training, but performs badly on X_{test}. It is not able to generalize to unknown data, since it learned a task too specific to the training set, it is overfitted. Detecting overfitting during training, and not only while testing a model, is therefore crucial. Methods to counter overfitting are called regularization. One way to detect overfitting is to split the training set into a training set and a validation set, analogously to the train-test split. The performance of the model under training is evaluated on the validation set (which is not used to update the model weights) continuously. If the validation loss starts to increase, one has reached the point of overfitting. A simple way of avoiding overfitting is to stop the training process at this point. The concept of early stopping is a simple and effective regularization method. Other regularization methods exist, notably penalty terms in the loss function and, in the context of neural networks, dropout layers which randomly nulls some nodes [37].



(a) A 3x4 input gets convoluted to a 2x3 output with a 2x2 kernel since no padding is applied.

(b) Illustration of 2D convolution followed by (2D) pooling (sub-sampling), e.g. average or maximum pooling

Figure 5.5: Illustration of a 2D convolution arithmetic (left, Fig. from [35]) and illustration of 2D convolution with pooling (right, Fig. from [33]) and the resulting dimensionality reduction.

5.2.2 Model Architecture and Training

Any ML algorithm does not need to be implemented from scratch anymore: numerous libraries exist with a variety of different model types and implementation flexibility. We worked with the PyTorch[11] library for Python, developed for deep learning. Training was performed on a NVIDIA Tesla V100 graphics card with 16 GB RAM.

Inspired by the models presented in [19], we constructed a 12 layer CNN, whose architecture is depicted in Fig. 5.6. Out of the 12 layers, 8 are trainable.

The convolutional and pooling layers have both symmetric kernel sizes and padding is applied along both directions equally. Layer (10) in Fig. 5.6 is a non-trainable layer, it fulfils two purposes. First, it flattens any multi-dimensional input, i.e. makes it one-dimensional. Second, it serves as a dropout layer representing thus a regularization method. Dropout layers randomly zeros some nodes during the training process. The zeroed elements are sampled from a Bernoulli distribution with probability p, in our setup we chose $p = 0.2$. The dropout layer is ignored during the validation and testing phase of our model. Blue layers are dense layer, meaning that they are fully connected: each node in the input is connected to every output node. In the same figure, the dense layer (12)'s output dimension needs naturally to be identical to the dimension of the flattened target y, $k = 20 \times 256 = 10240$.

The training and testing sets were obtained by a random 0.7-0.3 split of the respective datasets. Additionally, we defined a validation set from the training data by segregating randomly 10% from it. We used the binary cross entropy, see Eq. (5.2), as a loss function[12] and mini-batch gradient descent as optimizer[13]. The batch sizes were set to 32 and the GD learning rates were chosen heuristically, depending on the dataset.

We implemented early stopping as regularization method. Due to the stochastic nature of mini-batch GD, we assess the average loss over the last 30 epochs. If this average becomes greater than the average loss over the 30 epochs previous to 30 epochs of the just calculated mean, the training stops and the model is ready for testing.

[11] `https://pytorch.org/`

[12] The PyTorch implementation segregates loss functions from the last activation function used (if any) on the output. It provides, however, a BCE loss function with integrated sigmoid activation, equivalent to Eq. (5.2), assuring numerical stability.

[13] The PyTorch implementation is called SGD, it is however only a SGD if the training updates the model weights, by passing one sample after another. The PyTorch SGD is a "simple" gradient descent (with optional parameters) w.r.t. what is fed into it, e.g. in our case batches, thus our optimizer strategy is mini-batch gradient descent.

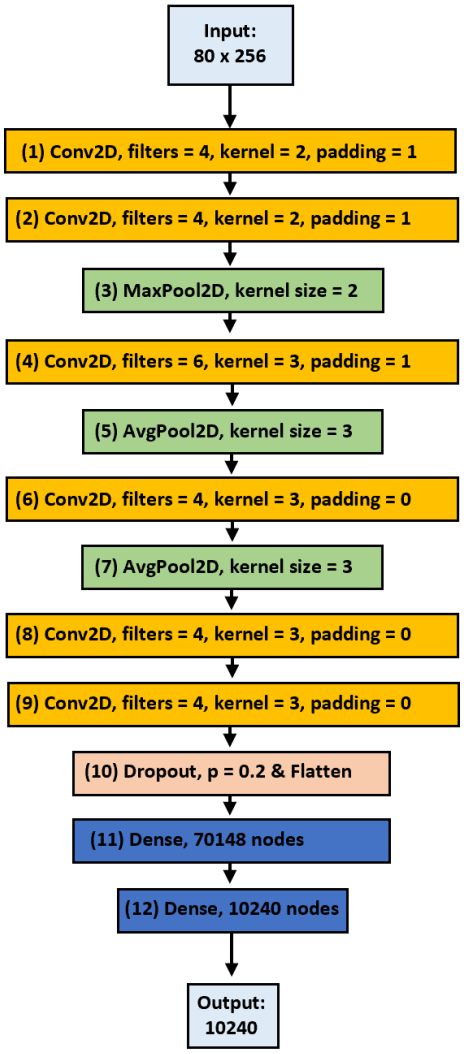


Figure 5.6: Architecture used in the model construction. The model input is 2D and output 1D with indicated pixel numbers, the (2D) convolutional layers are orange, the (2D) pooling layers (maximum and average) are light green. A dropout and flatten layer were combined in one (salmon colored) layer and dense layers are in blue.

Several models M_i with the same architecture were trained, obtained from different training sets X_{train}. Table 5.2 presents the different models, the number of epochs needed for training and the used learning rate.

The data choice for the models are the following. Model M_1 was created to investigate the ability to perform on the measurement data, namely on the Phantom,

Model	Data	η	Nr. epochs
M_1	Phantom&Skywalker 25dB simulation	0.50	300
M_2	Phantom&Skywalker 15dB simulation	0.10	182
M_3	Phantom, Skywalker, and Background measurements	0.07	1541
M_4	20% of Phantom&Skywalker 25dB simulation Phantom, Skywalker, background measurements	0.10	178

Table 5.2: Constructed models M_i. Since the models architecture is unchanged, the models differ by the datasets on which they are trained. The learning rate η of the mini-batch GD is given as well as the number of epochs needed to complete the training, according to averaged early stopping criterion of 30 epochs.

Skywalker and background dataset. It can also be used to investigate the possibilities to predict signals with different SNRs. The later reason was also the motivation to create M_2. Here we can compare the performance for higher and lower SNRs alike. As we will see in Sec. 5.3, M_1 was not able to achieve good results for the measurement data. As a proof of concept that it is indeed possible to train a model on the measurement data, we constructed M_3. The model M_4 was build in order to obtain a model with supposedly the highest variety in the training data. Since we suspected a high bias in M_4, if we would have used the entirety of the 25 dB simulation data, due to the relative abundance w.r.t the measurement data. Therefore, we selected randomly 9850 samples from the Phantom and Skywalker simulations, which results in a ratio of 2:1 of simulation to measurement data. The amount of selected samples corresponds to approximately 20% of the total dataset[14].

The performances of the respective models are discussed in the following section. Figure 5.7 shows the learning curve of M_1. The validation loss starts to saturate while the training loss is decreasing (on large scales) monotonically. Locally, one can observe kinks in the losses, which are attributed to the randomness inherent to the mini-batch GD optimization scheme. Once the validation saturates, early stopping prevents the model from overfitting and the training is interrupted.

5.3 Results and Discussion

Feeding an input into the model yields prediction $\hat{y}$, which corresponds to a spectrogram with pixel values $\in [0, 1]$. Each pixel represents the probability to be a signal pixel. Associating classes to each pixel, depending on a probability threshold, is necessary in order to obtain a qualitative measure of the model performance. We

[14]To be precise: 19.95%.

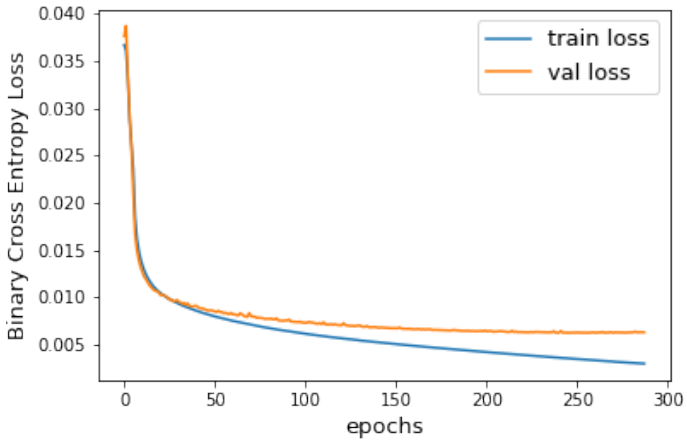


Figure 5.7: Learning curve of M_1 with BCE loss. Training stopped after 300 epochs. Training set size to validation set size ratio is 9:1.

define the vector $\hat{\hat{y}}$ whose elements are given by

$$\hat{\hat{y}}_i = \begin{cases} 1 & \text{if } \hat{y}_i \geq p \\ 0 & \text{if } \hat{y}_i < p \end{cases}, \tag{5.4}$$

and use the standard probability threshold choice $p = 0.5$ in the following.

5.3.1 Performance Measure

To evaluate the performance of the prediction outcomes, we propose two different scores. The standard score for any classification problem is the accuracy. The accuracy counts the number (pixels) of correctly identified classes over the size of the sample space. In our case it can also be written as

$$\text{Accuracy}(y, \hat{\hat{y}}) = 1 - \frac{1}{k} \sum_{i=1}^{k} \left(y_i - \hat{\hat{y}}_i \right)^2, \tag{5.5}$$

with $y, \hat{\hat{y}}$ the (vectorized) target and thresholded output spectrograms respectively[15], since the elements of both vectors are only in $\{0, 1\}$. We presented here the accuracy in such a way to stress the difference to the following score. The accuracy ranges from 0, worst possible prediction, to 1, perfect reconstruction of the target sample y. Because this score takes into account non-signal and signal pixels alike, we expect

[15] We will confuse the term output for both $\hat{y}$ and $\hat{\hat{y}}$, the specific meaning can be derived from the context and the formulas.

"very good" performances even for poor models, which predict only noise, i.e. no-signal pictures. This can be explained by the fact that only $\sim 1\%$ of the pixels are signal pixels[16]. It can be used, however, to compare the models' performances, e.g. with different architectures or models obtained from different datasets.

Therefore, we propose a second score function which accounts only for signal regions and gives a more intuitive quantification of the predicted outcome. We introduce the LP-score,

$$\text{LP-score}(y, \hat{y}) = \begin{cases} 1 & \text{if } \forall \ i \ \ y_i = \hat{y}_i = 0 \\ \frac{\langle y, \hat{y} \rangle}{\|y\| \|\hat{y}\|} & \text{otherwise} \end{cases}, \tag{5.6}$$

where $\langle *, * \rangle$ denotes the standard scalar product and $\|*\|$ is the Euclidean vector norm. This score is an adapted version of the correlation matrix distance between two matrices, or their vectorized representation. The correlation matrix distance was used in [38], [39] as a measure to quantify the overlap between two peak regions of two matrices in an otherwise mostly flat (zero) landscape. Similar to the score in Eq. (5.5), the LP-score $\in [0, 1]$, with the major difference that only signal regions are compared (last case in Eq. (5.6)). A score of 1 gets returned if a noise sample gets correctly identified as such (first case). The returned value in the first case could be seen as a too generous score, since one would expect that predicting accurately small pixel regions should be more valuable than correctly predicting noise, i.e. a zero-matrix. This could be accounted for by weighting the respective samples accordingly. In order to evaluate if a model has a good LP-score we propose the comparison to a naive classifier. A naive classifier could randomly annotate some pixels, or randomly some pixel regions, corresponding to the mean signal pixel number in the underlying dataset. This would, however, almost certainly result in a zero score. For this reason, we suggest a naive classifier, predicting only noise, i.e. 0 for every pixel value. This classifier would yield a low BCE loss and is therefore comparable with the BCE loss of a fully trained model. In the scope of the LP-score, it would perform as good as the ratio of noise-only samples to signal samples in the selected dataset. This will be the reference score with which we can quantify performances of M_i on the respective test sets.

[16] In the data used to train M_4, the average signal pixels amount was 75, which corresponds to 0.73% of the whole data. Generally, the mean signal pixel number never exceeded 100, 1% is thus a conservative estimate.

5.3.2 Evaluation of Models

Table 5.3 summarizes the performances of the models, see Tab. 5.2, on their respective test sets. Additionally to the mean accuracy and mean LP-score, we give also the associated standard deviation σ_{LP} of the mean LP-score. The standard deviation of the finite dataset $\{z_i\}$ of size k is calculated as $\sigma = \sqrt{\frac{1}{k}\sum_{i=0}^{k}(z_i - \mu)}$, with μ being the arithmetic mean of the dataset values.

Model	Accuracy	LP-score	σ_{LP}	Naive LP-score
M_1	0.998	0.858	0.123	0.127
M_2	0.997	0.775	0.167	0.126
M_3	0.999	0.907	0.081	0.193
M_4	0.997	0.715	0.251	0.080

Table 5.3: Test results of the trained models. With comparison of mean accuracy and mean LP-score as well as the naive LP-score (score obtained from a naive classifier corresponding to the amount of noise-only samples).

As expected, in terms of accuracy all models perform very well (close to 1). The models achieve good LP-scores. The considerably higher LP-score of 0.907 from M_3 can be explained by the fact that nearly 20% of the testing data consists of noise-only samples which were never missclassified, thus contributing to a high score, cf. Eq. (5.6). A standard deviation of up to 0.167 assures an overall good model performance. In the case of σ_{LP} being around 0.25 the result needs to be interpreted with some care.

For the reasons discussed previously, we evaluate in the following model performances solely based on the LP-score. A more qualitative understanding of different LP-scores is presented in Fig. 5.8[17]. Here we compare the output of four different approximated score values (0.6, 0.7, 0.8 and 0.9) for M_1 obtained during its testing phase. For that, four different samples were selected from the data.

In Fig. 5.8 (and Fig. 5.11b), correctly matched signal pixels (true positive) are colored in yellow, correctly matched noise pixels (true negative) are colored in pink, predicted pixels are colored in cyan (false positive) and missed signal pixels are colored in red (false negative). A score of 0.9 indicates that the model matches most of the signal areas, in the specific case of Fig. 5.8 it is still reached even if some pixels are wrongly predicted (in cyan). Note that the score does not distinguish the localisation of wrongly predicted signals, in a human interpretation one would prefer (if any) to have them close to the signal islands. A score of 0.7 has already

[17] In the following the output is always represented in 2D for obvious reasons, even though it is formally 1D.

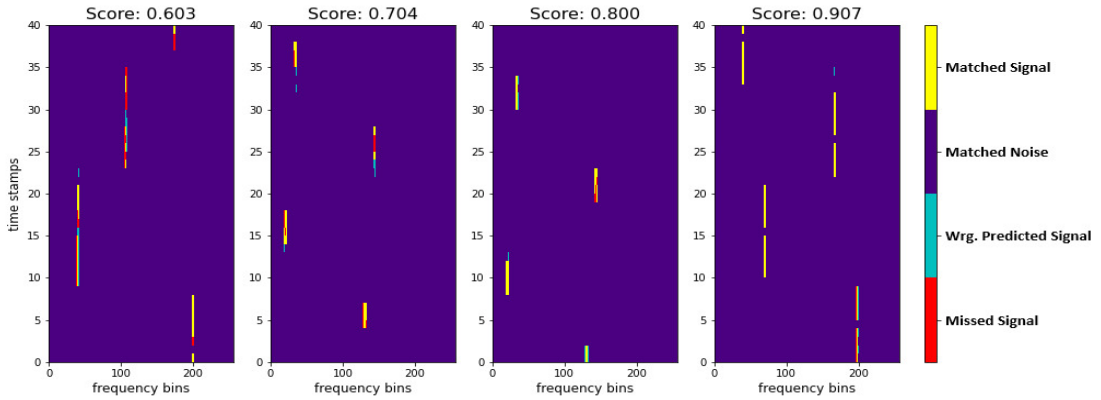
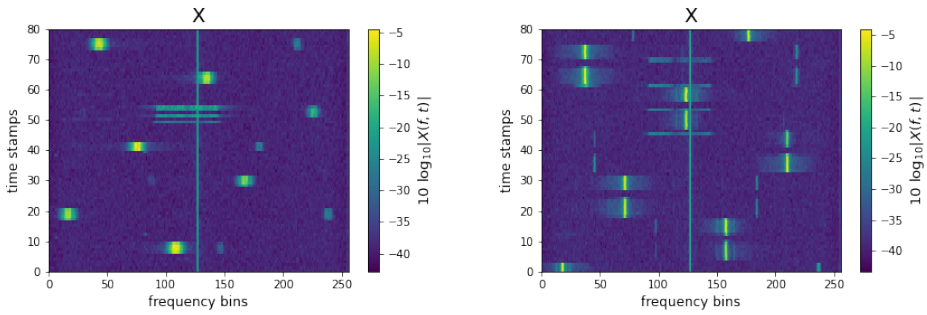


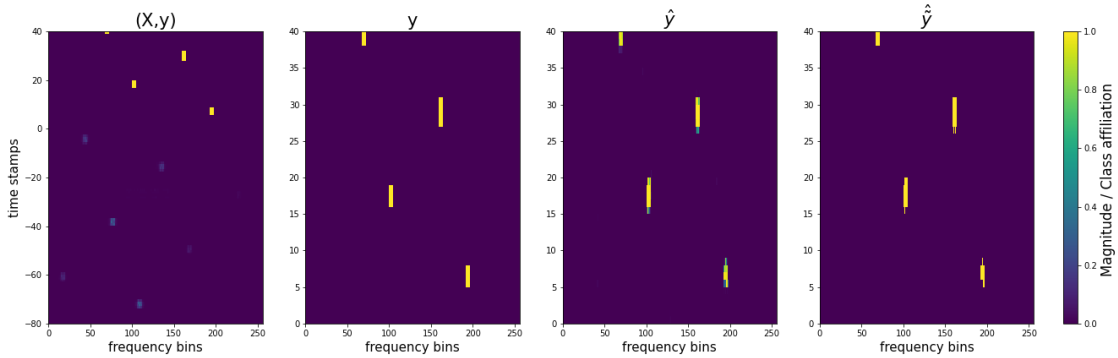
Figure 5.8: Comparison of different LP-scores, approximately 0.6, 0.7, 0.8 and 0.9 (left to right) of 4 different samples during the M_1 test phase. The color code distinguishes missed signal pixel (false negative) in red and wrongly predicted signal pixels (false positive) in cyan.

some amount of false positive and negative pixels. However, the overall shape of the signal is recognized and correctly predicted. Here, the third hop (in time) has matched signal pixels at the beginning and end of it. With prior knowledge of the underlying FHSS source, this could be sufficient to identify approximately the extend of the hop. A score of 0.6 still gives some information about the future FHSS signal localisation, the prediction is however considerably deteriorated by the presence of too many missed and wrongly predicted signal pixels.

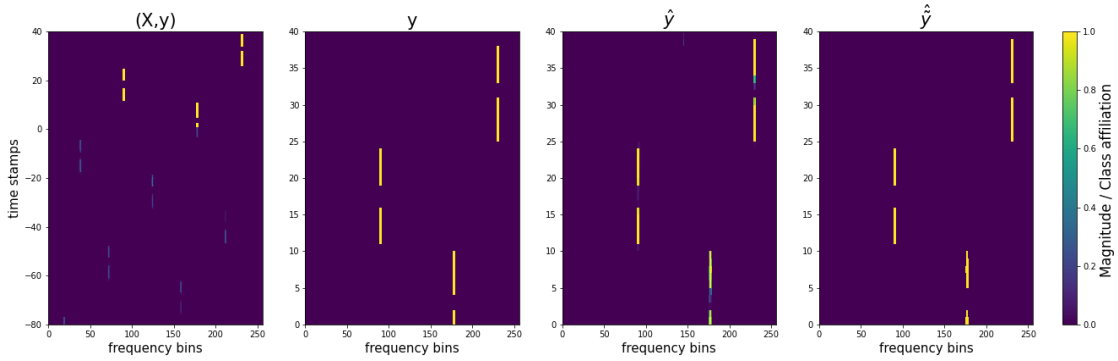


(a) Test sample X of a Phantom measurement.

(b) Test sample X of a Skywalker measurement.



(c) Prediction outcome of a Phantom sample Fig. 5.9a, yielding a score of 0.913.



(d) Prediction outcome of a Skywalker sample Fig. 5.9b, yielding a score of 0.942.

Figure 5.9: Test results of M_3, in the Phantom case (c) with input sample (a) and the Skywalker case (d) with input sample (b). The left picture in (c) and (d) shows the (X,y) sample. The second picture (from the left) shows again for better comparison y, followed by the $\hat{y}$ ML output. On the most right is the thresholded output $\hat{\hat{y}}$ with $p = 0.5$. The signal pixels are encoded with 1, noise pixels with 0.

Figure 5.9 shows the ML (probability) outcome $\hat{y}$ of both a Skywalker and Phan-

tom sample during testing of M_3. We compare it to the ground truth y and the thresholded version $\hat{\hat{y}}$. We also depict the respective input samples in logarithmic scale, as well as in linear scale and in time continuity its annotated y pair. We can observe that, despite the interferences being distinguishable in log-scale, the time continuity of X gets accurately predicted. The high test LP-score of M_3, cf. Tab. 5.3, compared to the other models can partially be explained by the fact that every noise sample gets predicted as such and by the relatively high amount of them in the respective test set, as indicated by the naive LP-score.

The presented results suggest successful training of all models. In some cases, training, especially for M_4 with a score of 0.7, could be considered as insufficient. Pursuing the training, however, bares the risk of overfitting. In the scope of LP-scores, all models trained well enough to get reasonable performances with scores > 0.7. We proceed now to the analysis of the individual model performance of non-native datasets, i.e. sets different from what was used in testing and training.

First, we investigate the abilities of the models M_1 and M_2 to generalize to other SNR levels. We use the entirety of the simulation data with the respective SNR to perform the evaluation. The performance scores are summarized in Tab. 5.4.

Model	5 dB	10 dB	15 dB	20 dB	25 dB
M_1	0.012	0.119	0.332	0.725	-
M_2	0.139	0.510	-	0.683	0.642

Table 5.4: Evaluation of M_1 and M_2 on different SNR levels. The mean LP-score is displayed, the naive LP-score is throughout 0.127.

The performance of M_1 is still remarkable on the 20 dB simulation data, with a mean score of 0.725 and a standard deviation of 0.17, but it deteriorates significantly for all SNRs below 20 dB. As expected, M_2 performs better than M_1, for low SNR, since it was trained on 15 dB data. The LP-score distribution for 10 dB, 20 dB and 25 dB is shown in Fig. 5.10. While the distribution in the 10 dB cases seems to be very even, with the exception of the noise-only samples (all scoring 1), the 20 and 25 dB distributions have a notably skewness towards higher score values. The plot illustrates that the model is, on the one hand, oversensitive and detects more pixel regions than it should (in the 10 dB case). On the other hand, the model misses considerably more signal pixels for the higher SNR cases. These false negative cases are thus the contributing factor to the decrease of the score.

The generalization of M_2 to higher SNRs is possible, as one observes in Tab. 5.4, but with a mean score of 0.642 and with $\sigma_{\text{LP}} = 0.237$ in the 25 dB case. Thus, a more satisfying generalization requires improvements. Since a generalization is in

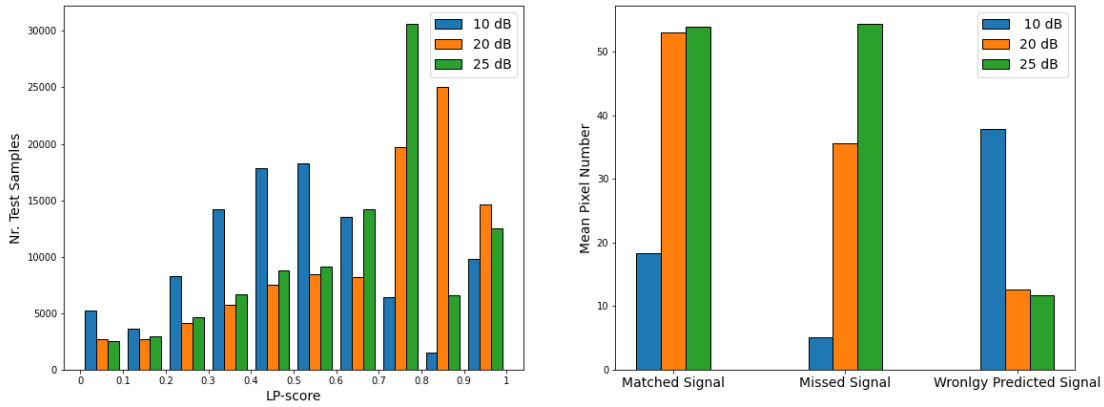


Figure 5.10: On the left, LP-score distribution of M_2's performances on different SNR data. On the right, mean pixel value of matched signal, missed signal and wrongly predicted signals for the same three distributions shown left.

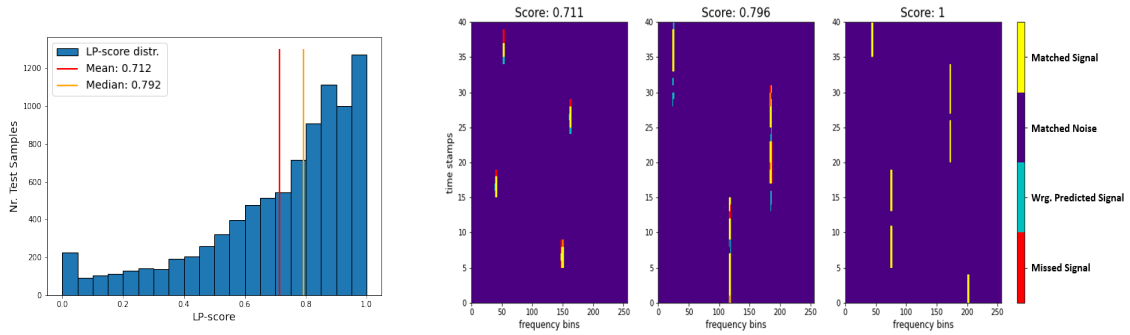
principle possible, one may use a combination of different SNR datasets, e.g. 25 dB and 15 dB, to cover in between SNR levels, e.g. 20 dB.

We further investigated the possibility of M_1 (and M_2) to predict samples from measurement data. This however failed, both models were found to be unable to generalize to the measurement data. Conversely, we checked also the possibility of M_3 to handle simulation data (of any SNR level). Likewise, the model was unable to generalize. In this regard, expectations to predict signals in measurement data, with a synthetic data trained model, couldn't be met. We suggest therefore one of the following approaches in order to predict measured FHSS signals: Either one relies solely on measurement data, which needs to be performed in controlled environments. Or one improves the simulation procedure, such that the signals become (nearly) indistinguishable in a spectrogramic representation compared to the measurement data. Or one uses a mixture of simulation and measurement data as we did in the construction of M_4.

The score distribution of M_4 on the test set is depicted in Fig. 5.11a. Even though the contribution of noise-only samples is considerable (728 samples) in the last bar in Fig. 5.11a, due to the fact that those were never missclassified[18], the mean and median test score are good enough to assure the prediction of either a simulated or measured FHSS signal. Similar to the score distribution in Fig. 5.10, it is a skew distribution towards higher score values, explaining the by 0.08 score points higher

[18] And thus resulting in a score of 1.

median (0.792) w.r.t. the mean (0.712). This puts into perspective the high σ_{LP}, cf. Tab. 5.3, and justifies the successful training. This is also under the consideration that reference LP-score is comparably low. Consequently, the mean LP-score is better than it seems. Figure 5.11b shows exemplary samples scoring of the order of mean and median. Also, it shows that occasionally (in 0.6% of all cases) a signal sample achieves a perfect score of 1.



(a) (b)

Figure 5.11: Test results of M_4. Plot (a) shows the LP-score distribution with bin size 0.05. One should keep in mind that 93% of the LP-score 1 cases is attributed to the noise-only samples. However, occasionally a signal sample achieves a perfect sore as seen in (b). The other two plots in (b) illustrate scores close to the mean (left) and median (middle) of the distribution in (a).

The results show that a mixture of simulated and measured data can be used to successfully train (and test) a model. Some care needs to be taken if one aims to train a model with a mixture of synthetic and measurement data. In our case, we selected randomly 20% of the simulation, which might have been not ideal, since a (small) possibility remains that a particular simulated pattern from either the Phantom or Skywalker runs is not completely represented in the dataset[19].

In summary, this chapter explained in detail the data preparation and processing. We introduced some general concepts of ML, more precisely all relevant theoretical concepts of the tools used from the PyTorch library. We built four models in total. Two models were obtained by training on the 25 dB and 15 dB simulation data respectively. A third one was exclusively constructed with measurement data and a forth one by a mixture of 25 dB simulation and measurement data. We introduced the LP-score with which we evaluated the performance of the models in different

[19] More likely in the case of the Skywalker-like simulations, due to its higher FHSS periods.

test scenarios. We found that the training process resulted in meaningful models in all four cases. However, a generalization of models constructed with synthetic data was not possible. We also found that extending the model performance to other SNR data was possible in some degree. The model trained on 25 dB data can also be applied without further adjustment to 20 dB, yielding a score of 0.725 corresponding to a 0.133 degradation w.r.t. the test score. The model trained on 15 dB achieves a 0.032 and 0.073 score deterioration w.r.t. the test score on the 20 dB and 25 dB data, respectively. The score obtained on the 10 dB data can be considered as insufficient, especially considering the underlying score distribution.

6 Conclusion

The aim of this master thesis was the prediction of FHSS pattern based on a small observation window with supervised learning methods, in a time-frequency representation.

First, I introduced the reader to fundamental concepts in wireless communication technology and especially to FHSS and to the possibilities of time-frequency representations. Second, some related work to the problem of FHSS detection and FHSS pattern prediction was presented as well as the framework with which I worked. The novelty of my approach is that the FHSS pattern prediction is based on a small observation window of only 50 ms, capable of predicting the signal for the following 25 ms. Third, I discussed the FHSS data acquisition from measurements and simulations alike. The simulation setup is versatile and allows the choice of different FHSS parameters and FHSS patterns. I simulated FHSS signals resembling the measured ones, and additionally accounted for different FHSS parameters to introduce further variations. Every simulation was performed for five different SNR values, allowing comparisons to environments deteriorated by noise. Fourth, I described the challenges encountered in the ML data preparation, notably the dimensionality issue and the problematic of a regression. I circumvented these problems by reducing considerably the pixel dimensions in time and frequency, at the cost of reduced resolution, for the former. The latter problem was tackled by a suitable annotation, thus setting up a classification task. The constructed data consisted of a 80×256 matrix for the input samples X, and a 40×256 binary matrix for the target samples y. The latter implies a 25 ms prediction window of the signals evolution. Fifth, concepts of (convolutional) neural networks and important terminology related to the field of ML as well as the idea and problems of overfitting were discussed.

Finally, I built four models by training a CNN architecture on different datasets. I proposed a score measure, the LP-score, and showed based on this score that successful prediction of the signal's evolution is possible, with models obtaining a score up to 0.907 on their respective test set. However, it was not possible to extend the performances of models trained with simulated data towards measurement data. A generalization of models trained on a high SNR towards lower SNR is possible, but remains poor for an SNR further away from the training SNR. In contrast, a model trained on a low SNR can, with a score decrease up to 0.133, still perform relatively well on a higher SNR. Furthermore, I found that a model may

be trained and tested successfully on a mixture of simulation and measurement data.

Future work may consist in adjusting the architecture of the presented CNN. Having less trainable model parameters would reduce the training time and allow for lower hardware requirements. Adjusting the observation and the prediction time windows, as well as the time and frequency resolution to particular needs may be beneficial in order to obtain a more efficient memory data usage. The models themselves may be enhanced by means of a better optimizer during training, adjustable learning rates etc., such that one obtains a more accurate prediction of the signal evolution.

Bibliography

[1] D. S. et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, 2016.

[2] L. Shen, L. R. Margolies, J. H. Rothstein, E. Fluder, R. McBride, and W. Sieh, "Deep learning to improve breast cancer detection on screening mammography," *Scientific Reports*, vol. 9, 2019.

[3] A. F. Mölisch, Ed., *Wireless Communications*. John Wiley and Sons, 2005.

[4] B. SIG, *Bluetooth Wireless Technology*, https://www.bluetooth.com/learn-about-bluetooth/tech-overview, accessed 12.11.2021.

[5] *SCALA - Sicherheit im urbanen Raum*, https://www.kiras.at/gefoerderte-projekte/detail/d/scala-sicherheit-im-urbanen-raum/, accessed 19.08.2021.

[6] DGA, *Note aux rédactions, mercredi 7 juillet - lutte anti-drone - déplacement de Florence Parly sur le site de la direction générale de l'armement à Biscarosse*, https://www.defense.gouv.fr/salle-de-presse/note-aux-redactions/note-aux-redactions_mercredi-7-juillet-lutte-anti-drone-deplacement-de-florence-parly-sur-le-site-de-la-direction-generale-de-l-armement-a-biscarosse, accessed 18.08.2021, 2021.

[7] K.-D. Kammeyer, Ed., *Nachrichtenübertagung*, 3rd ed. Teubner, 2004.

[8] F. Hlawatsch, *Lecture notes on Modulations- und Detektionsverfahren*, 2010.

[9] T. S. Rappaport, *Wireless communications - principles and practice*. Prentice Hall, 1996, ISBN: 978-0-13-375536-7.

[10] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation*, vol. 19, pp. 297–301, Apr. 1965.

[11] W. M. World, *Fast fourier transform*, https://mathworld.wolfram.com/FastFourierTransform.html, accessed 16.11.2021, 2021.

[12] D. Rowell, *2.161 signal processing - continuous and discrete*, 2008.

[13] D. Gabor, "Theory of communication," *Journal of Institution of Electrical Engineers*, vol. 93, pp. 429–457, 1946.

[14] M. Marcolli, *Chapter 6 gabor representations*, www.its.caltech.edu/~matilde//GaborLocalization.pdf, accessed 25.02.2022.

[15] D. Torrieri, *Principles of Spread-Spectrum Communication Systems*. Springer, 2015.

[16] J.-R. Ohm and H. D. Lüke, *Signalübertragung*, 11th ed. Springer, 2010.

[17] S. A.-w. Hadi Athab Hamed Ahmed Kareem Abdullah, "Frequency hopping spread spectrum recognition based on discrete fourier transform and skewness and kurtosis," *International Journal of Applied Engineering Research*, 2018.

[18] J. Huang, Y. Zhou, N. Sun, and Z. Hu, "Identification of frequency-hopping spread spectrum signals using svms with second generation wavelet kernels," in *2010 International Conference on Communications and Intelligence Information Security*, 2010, pp. 160–163.

[19] S. Al-Emadi and F. Al-Senaid, "Drone detection approach based on radio-frequency using convolutional neural network," in *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, 2020, pp. 29–34.

[20] O. O. Medaiyese, A. Syed, and A. P. Lauf, "Machine learning framework for rf-based drone detection and identification system," *arXiv*, 2020.

[21] I. Nemer, T. Sheltami, I. Ahmad, A. U.-H. Yasar, and M. A. R. Abdeen, "Rf-based uav detection and identification using hierarchical learning approach," *Sensors*, vol. 21, no. 6, 2021, ISSN: 1424-8220. DOI: 10.3390/s21061947. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/1947.

[22] M. M. A. Z. Muhammad Turyalai Khan Ahmad Zuri Sha'ameri, "Classification of fhss signals in a multi-signal environment by artificial neural network - unpublished," 08.2021.

[23] M. Ezuma, F. Erden, C. Kumar Anjinappa, O. Ozdemir, and I. Guvenc, "Detection and classification of uavs using rf fingerprints in the presence of wi-fi and bluetooth interference," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 60–76, 2020. DOI: 10.1109/OJCOMS.2019.2955889.

[24] B. Kaplan, Kahraman, A. Görçin, H. A. Çırpan, and A. R. Ekti, "Measurement based fhss–type drone controller detection at 2.4ghz: An stft approach," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020, pp. 1–6. DOI: 10.1109/VTC2020-Spring48590.2020.9129525.

[25] L. Bernadó, D. Löschenbrand, P. Thiele, and T. Zemen, "Scala project - d5.3 ml methoden zur drohnenintervention," Project deliverable, 2022.

[26] H.-B. Kil1, J.-S. Lee, and E.-R. Jeong, "Analysis of frequency hopping signals in commercial drones," *International Journal of Pure and Applied Mathematics*, 2018.

[27] J. Skorepa, *Frequency-hopping parameter estimation for unmanned aerial vehicle radio links*, 2019.

[28] H. Shin, K. Choi, Y.-S. Park, J. Choi, and Y. Kim, "Security analysis of fhss-type drone controller," in *WISA*, 2015.

[29] M. Song and T. Allison, "Frequency hopping pattern recognition algorithms for wireless sensor networks," in *PDCS*, 2005.

[30] M. S. Allahham, M. F. Al-Sa'd, A. Al-Ali, A. Mohamed, T. Khattab, and A. Erbad, "Dronerf dataset: A dataset of drones for rf-based detection, classification and identification," *Data in Brief*, vol. 26, 2019.

[31] J. Wan, D. Zhang, W. Xu, and Q. Guo, "Parameter estimation of multi frequency hopping signals based on space-time-frequency distribution," *Symmetry*, vol. 11, no. 5, 2019, ISSN: 2073-8994. DOI: 10.3390/sym11050648. [Online]. Available: https://www.mdpi.com/2073-8994/11/5/648.

[32] I. Neutelings, *Neural networks*, https://tikz.net/neural_networks/, accessed 17.05.2022.

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

[34] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014.

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

[36] V. Dumoulin and F. Visin, *A guide to convolution arithmetic for deep learning*, 2016.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[38] M. Herdin, N. Czink, H. Ozcelik, and E. Bonek, "Correlation matrix distance, a meaningful measure for evaluation of non-stationary mimo channels," in *2005 IEEE 61st Vehicular Technology Conference*, vol. 1, 2005, 136–140 Vol. 1.

[39] L. Bernadó, T. Zemen, F. Tufvesson, A. F. Molisch, and C. F. Mecklenbräuker, "The (in-) validity of the wssus assumption in vehicular radio channels," in *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, 2012, pp. 1757–1762.