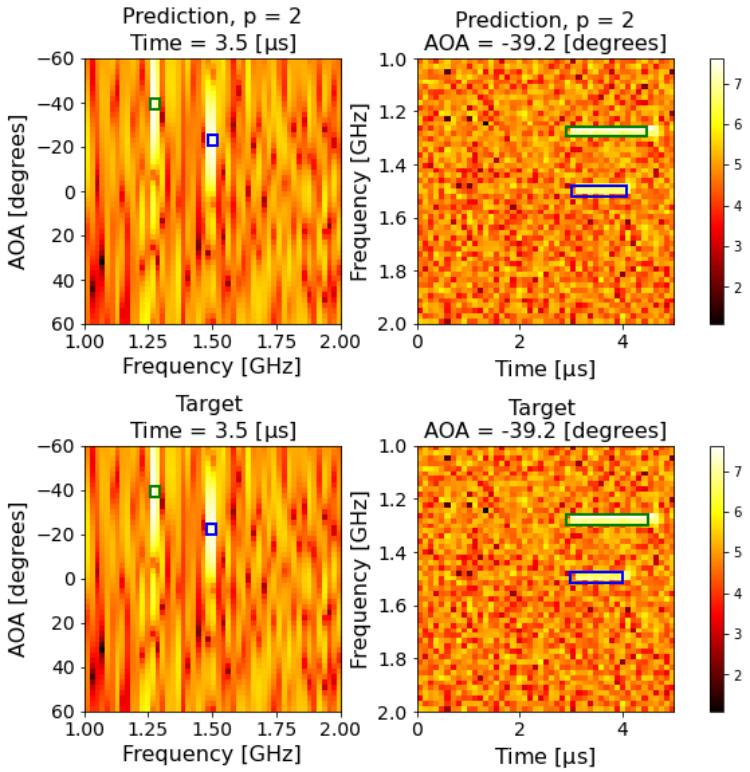
CHALMERS
UNIVERSITY OF TECHNOLOGY



Signal Detection with a Digital Antenna Array using Machine Learning

Image Recognition Methods Applied for Joint Detection and Signal Parameter Estimation

Master's thesis in Physics

Elin Ohlman

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Signal Detection with a Digital Antenna Array using Machine Learning

Image Recognition Methods Applied for Joint Detection and Signal Parameter Estimation

ELIN OHLMAN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Communications, Antennas, and Optical Networks
Chalmers University of Technology
Gothenburg, Sweden 2023

Signal Detection with a Digital Antenna Array using Machine Learning
Image Recognition Methods Applied for Joint Detection and Signal Parameter Estimation
ELIN OHLMAN

Supervisor: Eric Norgren, SAAB
Supervisor: Alexander Karlsson, SAAB
Examiner: Rob Maaskant, Department of Electrical Engineering, Chalmers

Master's Thesis 2023
Department of Electrical Engineering
Division of Communications, Antennas, and Optical Networks
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Predictions and targets for a 3D image with the angle of arrival, frequency, and time as the three axes. The predictions are made by the network developed in the thesis.

Typeset in LaTeX
Gothenburg, Sweden 2023

Signal Detection with a Digital Antenna Array using Machine Learning
Image Recognition Methods Applied for Joint Detection and Signal Parameter Estimation
ELIN OHLMAN
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Radars and radar warning receiver systems engage in a perpetual battle for superiority. This constant struggle demands the development of new methods, and recent advancements in machine learning and image recognition may serve as sources of inspiration. In this thesis, a "You Only Look Once"-style neural network is designed to detect pulsed radar signals and estimate their angle of arrival, frequency, time of arrival, and pulse width. Employing a digital multi-channel uniform linear antenna array, a short-time Fourier transform is applied to the output from each antenna element. Afterward, beamforming is performed for various angles combining the spectrograms, producing a 3D image upon which image recognition is applied. We find that using simulated data this method works for detecting two sinusoidal signals, even in noisy environments.

Keywords: Angle of Arrival (AOA), Beamforming, Convolutional Neural Network (CNN), Digital Antenna, Signal Detection, Spectrogram, Uniform Linear Array (ULA), You Only Look Once (YOLO)

Acknowledgements

First I would like to thank my supervisors at Saab, Eric Norgren, and Alexander Karlsson for the opportunity to do this thesis work. This project would never have been possible without your support, patience, and guidance. To my academic supervisor at Chalmers, Rob Maaskant, thank you for your support and understanding. To my partner Lukas Falk, thank you for your love and unwavering faith in me, even when I myself lacked it. Lastly, I thank my family, Börje, Hanna, Elsa, and Erik Ohlman for your everlasting support and encouragement.

Elin Ohlman, Stockholm, May 2023

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order.

AOA	Angle of Arrival
CNN	Convolutional Neural Network
FFT	Fast Fourier Transform
IoU	Intersection over Union
IQ	In-phase and Quadrature
ML	Machine Learning
NN	Neural Network
PW	Pulse Width
ResNet	Residual Neural Network
RF	Radio Frequency
RMSE	Root Mean Square Error
SNR	Signal to Noise Ratio
STFT	Short Time Fourier Transform
TOA	Time Of Arrival
ULA	Uniform Linear Array
YOLO	You Only Look Once

Contents

1

Introduction

Radars attempt to obtain information about their surroundings by transmitting electromagnetic signals and then analyzing the echo. A radar warner receiver system does not transmit its own signals but instead passively listens to the signals transmitted by other's radars. The radar warner receiver has an advantage by receiving a stronger signal than the radar since the signal only has traveled half the path. However, the radar warner receiver is at a disadvantage by not knowing what it is looking for.

Signal detection as a problem can vary a lot in complexity depending on your prior information. If you know how many signals you are looking for, and in what directions, frequencies, and signal types, there are many available methods to detect the desired signals, even if they are weak. For example, if the source direction is known, an antenna array can either mechanically or electronically be steered to amplify signals from that direction and suppress signals from other directions. But if some or even all of the signal parameters are unknown, detection becomes much harder.

There are different ways to steer an antenna. The simplest way is to mechanically turn the array toward the source direction, such that the plane wave hits the different array elements simultaneously. This could also be modeled as a time delay, so instead of turning the array, a time delay can be added to the received signals, to steer the array. If a digital antenna array with a separate data channel for each antenna element is used, it is possible to steer the antenna in multiple directions simultaneously, since the same data can be processed multiple times. A multi-channel digital antenna array requires one analog-to-digital converter for each antenna element and will produce large data streams.

The large data streams generated by a digital antenna array necessitate processing, and machine learning excels at managing vast quantities of data. Another compelling reason to employ machine learning in signal detection is the ability to transform signal data streams into images through the utilization of a short-time Fourier transform, thereby generating spectrograms. Image recognition methods, which have been extensively developed and proven successful, can then be applied to these spectrograms. Furthermore, machine learning proves to be an apt solution for handling noisy data. When it comes to detecting weak signals, the capability to effectively manage noisy data is essential.

1.1 Related work

There are many interesting possible applications of machine learning in the signal processing domain. Some of the pioneers in the area are the authors of [1], who used convolutional neural networks on time domain IQ data for radio modulation classification. For using image recognition in signal processing some examples are; in [2] a pre-trained convolutional neural network is applied to spectrograms of speech signals for recognition of emotional speech. Also using CNNs in [3], they perform Radar-Spectrogram-Based UAV Classification.

Focusing on image recognition approaches for signal detection, classification, and parameter estimation there has been some interesting work done. Creating an image, a spectrogram, and then using image recognition on that to detect and classify signals has been performed by for example, [4]–[6], who all use "You Only Look Once"-style networks. Neither of them does any investigation into the angle of arrival of the signals and only a single data channel is used.

The angle of arrival estimation is usually separated into a different problem than detection and classification. There are many available classical methods for the angle of arrival (also called direction of arrival) available. For example sub-space methods like MUltiple SIgnal Classification (MUSIC) and MVDR, sparsity-inducing methods, and maximum likelihood methods. Attempts to make improved machine learning-based methods have been made by for example [7]–[10].

Image recognition in 3D is similar but not identical to 2D image recognition. With a normal camera you obtain a 2D image, but with for example LiDAR or CT scans you obtain a 3D image. Image recognition on LiDAR data has been performed in for example [11]. In [12] a 3D CNN is used on CT scan images to identify and localize the area of a lumbar vertebra of interest.

1.2 Scope and limitations

The idea of this thesis project is to treat signal detection and parameter estimation as an image recognition problem. We will preprocess the signal data from the different antenna elements into spectrograms. Then we will steer the antenna in multiple directions, adding the spectrograms together multiple times. With this, a 3D image is created, with time, frequency, and angle as the three axes. By performing this preprocessing we encode our previous knowledge of the frequency content of the time domain signal, and how the data from the different antenna elements are connected to each other. This should make it easier for the network to learn since it does not have to learn this information as well.

There have been similar approaches, using image recognition on spectrograms, for example, [4]–[6], though only using a single data channel. The novel idea in this project is to use an image recognition approach on multi-channel data. With this kind of data, it is possible to also infer knowledge about the angle of arrival of the signal and facilitate the detection of signals in a wide range of angles simultaneously.

To make the scope of the project reasonable some limitations are required. We will study signals between 1 and 2 GHz, with an angle of arrival in the range -60° to 60°, and use a measurement period of length 5 µs. We assume a 2D world and such only azimuth angle will be considered, and elevation angle ignored. Only the detection of two single-frequency sinusoidal signals will be studied. Simulated data and a digital six-element uniform linear antenna array will be used. Since there is only one class of signals no classification will be performed. The signal parameters angle of arrival, frequency, time of arrival, and pulse width will be estimated. The focus of this thesis is to study first-order dominant effects, more realistic antenna models are left for future work. Thus a simple and ideal antenna model will be used. Only the method designed in this thesis will be investigated and it will not be compared to some benchmark method.

1.3 Outline

This thesis is structured as follows: In Ch. 2 the required theory in signal processing and machine learning to understand the method is presented. It is assumed that the reader has some basic knowledge in these areas. In Ch. 3 the method is described. First, how the data is simulated and preprocessed. Second, the design process of the neural network is explained, and finally, the loss and metrics used for training and evaluation of the network are introduced. Ch. 4 first describes the training of the network. The trained network is then evaluated on test data, investigating the performance at different Signal to Noise Ratios (SNR), and also some investigation into incorrect predictions is made. Additional predictions from the network are included in App. A, for the reader to obtain a better understanding of the network performance. In Ch. 5 a discussion is held concerning the performance of the network and what could be improved upon. Suggestions for future work are made. Finally, in Ch. 6 the thesis is concluded.

2

Theory

In this Chapter, the required theory for understanding the work in this thesis will be explained.

2.1 Signal parameters

In this thesis, pulsed sinusoidal signals will be investigated. To describe these pulses the following parameters are required:

- AOA - the Angle of Arrival, measured from the normal to the array
- RF - the Radio Frequency
- TOA - the Time of Arrival
- PW - the Pulse Width, the length of the signal
- A - the Amplitude
- ϕ - the phase

2.2 Short time Fourier transform

The Fourier transform is essential when studying signals. With the Fourier transform the signal can be studied in the frequency domain instead of the time domain. The frequency content of a signal is essential for distinguishing different signals from each other. With an input stream of data from a receiving antenna, a Fourier Transform (FT) can be performed and the frequency content can be studied. The frequency content of a signal can change over time, for example, the signal could end, be pulsed, change frequency (chirp), etc. Then it is useful not to perform the Fourier transform on the entire data stream but instead, take it over smaller sections of the data and thus get the frequency content for different time steps, a short-time Fourier transform. With this method a spectrogram is created, an image with frequency on one axis and time on the other, the pixel values representing the value of the Fourier transform for that specific frequency and time bin.

In this thesis, we make use of a "weighted overlap-add" structure for the STFT [13] . The advantage of using this method is lower sidelobes and a flatter passband response. In short, the method works by using P windows of size 2^K, applying some

window function to them, before adding the windows together. Then applying the FFT to the result. With some stride S, the oldest time samples are discarded and new samples are added, and the process is repeated producing a spectrogram.

2.3 Beamforming

An antenna array receives incoming signals with some delay between the different antenna elements depending on the AOA of the incoming signal. If the signal is perpendicular to the antenna array there is no delay. However, if the signal is arriving from some angle there is a delay in the signal arriving to the different antenna elements. If the signals from the different antennas are summed, a perpendicular signal would be amplified since the signals interfere constructively. On the other hand, a signal incoming from an angle will interfere destructively. Thus the array can be used to amplify signals incoming perpendicularly to the array. To target another direction the antenna array can be turned toward that direction. It is also possible to instead introduce a delay before adding together the signals from the antenna elements to replicate steering toward the target direction.

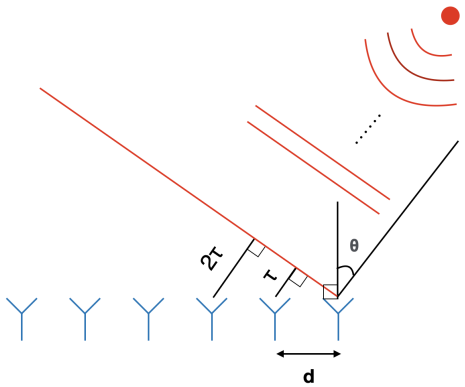


Figure 2.1: Signal time delay for a uniform linear array.

Assuming we have a far-field source generating the signal it is illustrated in Fig. 2.1 how the time delay of a signal between two antenna elements can be calculated by,

$$\tau = \frac{d}{c} \sin \theta. \tag{2.1}$$

Here τ is the time delay between two antenna elements, d is the distance between the antenna elements, c propagation speed of the wave, in this case, assumed to be the speed of light, and θ is the angle of arrival for the incoming signal compared to the normal of the array.

For a narrowband signal such as a sinusoidal signal, this is simply a phase shift of the signal,

$$\exp\left[j(2\pi f(t+\tau)+\phi)\right] = \exp\left[j(2\pi ft + 2\pi f\tau + \phi)\right] = \exp\left[j(2\pi ft + \Delta\phi + \phi)\right]. \tag{2.2}$$

This phase shift is the same between any two adjacent antennas such that $\Delta\phi_n = 2\pi f\tau n$, where $n = 0, 1 \ldots (N-1)$ is the antenna number. To simulate more advanced broadband signals it would instead be required to implement the delay as a time delay directly, which is possible if the formula for the signal as a function of time is available.

If the AOA of the incoming signal is known, the array can be steered in that direction by applying a reverse phase shift corresponding to the $\Delta\phi_n$ for that AOA. This is done by multiplying the output from the different antenna elements with

$$\exp\left[-j\Delta\phi_n\right] = \exp\left[-j2\pi f\frac{dn}{c}\sin\theta\right]. \tag{2.3}$$

With a digital antenna array, it is possible to apply multiple such transformations for different AOA for the same input data, and can thus be steered in multiple directions simultaneously.

2.4 Neural networks

A Neural Network is inspired by how the human brain works with many simple neurons that are connected to some other neurons. It is assumed that the reader has some familiarity with how neural networks work. In this section, we focus on some specific neural network ideas that are used in this thesis.

2.4.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are neural networks with convolutional layers. A convolutional layer uses a small kernel to perform a cross-correlation, a sweeping convolution over all dimensions with some stride. The convolution is only performed on the center value and some surrounding values. Some common kernel sizes are 1,3,5,7. The weights and bias of the kernel are learned when training the network, but important to note is that the same weights and bias are used for the entire input volume. Assuming the data should be treated similarly, invariant to where in the input the data is located, and that nearby inputs likely are connected, a convolutional network may be suitable. Using a convolutional layer reduces the number of weights in the network compared to a fully connected layer. The kernels learn to recognize features in the input, such as edges. This is why CNN:s are extremely popular for image recognition problems.

The output of a convolutional layer is produced through multiple convolutions over the input volume. In this thesis we will work with 3D image data, thus a 3D convolution is required. The layer takes an input of shape (Channel, Depth, Height,

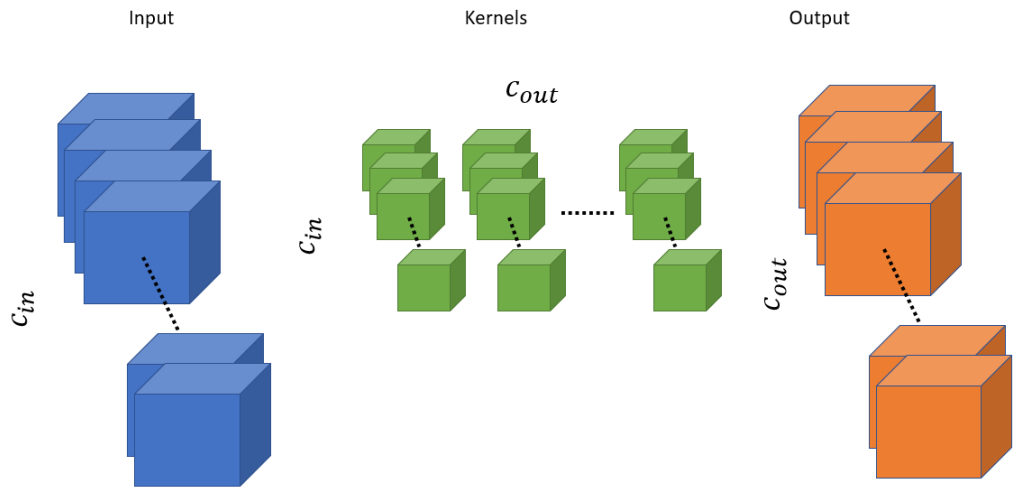


Figure 2.2: The input, kernel, and output shapes of a convolutional layer. A kernel of shape C_{in}*F*F*F produces one output. Thus by choosing the number of such kernels the number of output channels C_{out} is determined.

Width) and the kernel has shape (Channel, F, F, F), where F is the kernel size. For every channel, the (F, F, F) sub-kernel is swept over the entire 3D volume and convoluted, the cross-correlation operator. This is then summed over all the input channels and added with the kernel bias, which produces one output channel. This is described by

$$\text{Output}(C_{\text{out}_j}) = \text{Bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{Weight}(C_{\text{out}_j}, k) \star \text{Input}(k), \tag{2.4}$$

with $\star$ being the cross-correlation operator, with some stride s. How many kernels you use determines the number of channels in the output. Fig. 2.2 shows an overview of how the shape of the kernels connects with the input and output shapes.

A stride of 1 means all values will be used in the center of the kernel, and a stride of 2 means every other value will be used. A stride larger than one reduces the dimension of the output. For a kernel size greater than 1, the edges of the input need to be handled in some way. Some padding can be used, for example, adding zeroes around the edges, or copying the values at the edges. It is also possible to not use any padding which would result in an output with a reduced dimension. The shape of the output is determined by the stride and the padding, and the number of channels by the number of kernels.

A kernel size of size 1 makes a convolutional layer that is a different, but still useful layer. This operation involves multiplying each element of the input tensor by a learned scalar weight and then summing the results from all channels and also adding the learned bias to produce a single output value for each cell. This can be used to increase or decrease the depth of the input tensor without changing its spatial dimensions. By combining multiple channels with size 1 convolutions, the network can learn to capture more complex relationships between different channels

of the input tensor. When using a kernel size of 1 no padding should be used since the kernel only acts on a single cell within a channel.

2.4.2 Residual Network

With a deep neural network one problem that can arise is so-called *degradation*, when increasing the depth of the network, the training loss increases compared to more shallow networks. In [14] a residual network is created that addresses this problem, by introducing shortcut connections. The shortcut connections allow the output from a previous layer to skip two layers before being added together with the output from those layers as in Fig. 2.3. This allows the gradient to propagate relatively unchanged throughout the network, allowing a larger magnitude gradient to reach deeper layers.

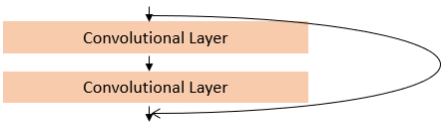


Figure 2.3: Shortcut connection used in the residual network.

2.4.3 You Only Look Once

You Only Look Once (YOLO) is a popular and well-performing image recognition network that only requires a single pass through the network. It uses convolutional layers and its main idea is to divide the image into a grid. Then each cell in the grid makes a prediction for the bounding box, class scores, and confidence score. The bounding box is a tight box around the object. The cell at the center of the object is the cell "responsible" for detecting that object.[15] After the original YOLO paper several improvements to the network have been made as in [16] and [17].

3

Method

This chapter provides a comprehensive overview of the methodology used to simulate and preprocess the data. Additionally, it details the approach taken to design the neural network architecture, as well as the process of training and evaluating the network.

3.1 Data generation

We choose to study a 6-element multi-channel digital uniform linear array (ULA) with omnidirectional antenna elements. The distance between the antenna elements is set to $d = \frac{\lambda}{2} \approx 7.5$ cm of the highest frequency we wish to detect, 2 GHz. The signal is sampled with a sampling frequency of 5 GHz.

To simulate the received signals a simulator is used that generates the transmitted signals. The signals are complex sinusoidal signals, with some frequency f, phase ϕ, and amplitude A,

$$A \exp(j(2\pi f t + \phi)). \tag{3.1}$$

To simulate the array, 6 signals with the same parameters but with different time delays depending on the angle to the source. Using equations (2.1) and (2.2) and with $\phi = 0$ for the n=0 antenna element we obtain the formula for the signal received by the n:th antenna:

$$\exp(j(2\pi f t + 2\pi f \tau_n)) = \exp(j2\pi f(t + \frac{dn}{c}\sin(\theta))), \tag{3.2}$$

with θ being the AOA.

To create the data set we simulate sinusoidal signals with uniformly sampled frequency in the range 1 to 2 GHz. The amplitude is set to 1 for all the signals. To generate data samples with different Signal to Noise Ratios (SNR) the amplitude of the signals is kept constant and instead, the noise is varied.

Signals are generated with the signal parameters being uniformly sampled. The AOA is between -60° and 60° and frequencies are between 1 and 2 GHz. A measurement period of 5 µs is used and signals generated have a minimum length of 1 µs. Two

signals are generated for every data set sample and both have the same amplitude, $A = 1$. Complex valued additive white Gaussian noise is added to achieve the desired SNR. In this report, we define SNR to be the signal amplitude over the noise floor for a *single* antenna element.

For training data, a total of 9000 samples is generated at 9 different SNRs, -40, -35 -30, -25, -20, -15, -10, -5, and 0 dB with 1000 samples at each SNR level. For validation and testing, two separate data sets containing a total of 2700 samples each are created, with 300 samples per SNR level.

3.2 Data Preprocessing

Prior to inputting data into the neural network, preprocessing is required. The idea of this project is to treat the data as a 3D image with angle, frequency, and time as the three axes of the image. The signal data is given as an array of IQ values, complexly sampled amplitude values at a known sampling frequency for each antenna. Then we want to perform an STFT such that we get 6 spectrograms, one for each antenna. Following that the 6 spectrograms are beamformed together into one spectrogram for each angle.

3.2.1 Spectrogram

The "weighted overlap-add" method introduced in section 2.2 is used to produce the spectrograms. We choose an FFT window size of nfft $= 2^K = 2^8 = 256$ and use $P = 4$ windows. Before adding the windows together a window function is applied. We used a Parks McClellan window function [13], with a passband range of $[0, \frac{f_s}{2*2^K}]$, stop-band range of $\left[\frac{f_s}{2^K}, \frac{f_s}{2}\right]$, passband weight of 1, and a stopband of 111. The four windows are then added and an FFT is applied. A stride of $S = 512$ was used meaning two windows were discarded every iteration and two new ones added, giving an overlap of 50% between responses. With a measurement period length of 5 μs and a sampling frequency of 5 GHz, 25000 samples are obtained per measurement. This produces $\frac{25000-1024}{512} \approx 47$ frequency responses. With a sampling rate of $f_s = 5$GHz and a desired range of [1GHz, 2GHz] the number of frequency bins is $\frac{2GHz-1GHz}{5GHz-256} \approx 51$ per response. Thus, using this method a spectrogram with 51 frequency bins and 47 time bins can be acquired for each output channel of the antenna array.

3.2.2 Beamforming

The 6 spectrograms obtained can be combined and steered in some direction using equation (2.3). This is performed 50 times using 50 evenly spaced angles between $-60°$ and $60°$. The phase shift is applied to each frequency bin for every antenna with a corresponding value of f used in the equation. The results from each frequency bin of the different antennas are then summed together creating a complex-valued 3D volume of data, with time, frequency, and angle as the three axes.

After preprocessing the input data we obtain images with 50x51x47 pixels, spanning (−60°, 60°)x(1 GHz, 2 GHz)x(0 µs,5 µs) resulting in a pixel having the resolution approximately (2.4°, 19.6 MHz, 0.11 µs).

3.2.3 Input data

After creating the spectrograms and beamforming them together, we have 3D IQ images to try and extract the signal parameters from. To simplify the problem we only use the magnitude of the spectrograms. By doing this information is lost. This information could have been used to separate two signals with the same RF, AOA, TOA, and PW but with different phases. To handle complex-valued input a NN that can handle complex data could have been used instead. There are some implementations of complex-valued neural networks, but then some problems arise with activation functions and such that are not trivially translated for complex data. The other common method for handling complex data is to give multi-channel input to the network, for example, the I and Q data have one channel each, or using the magnitude in one channel and the phase in one channel. But in this project, we limit ourselves to only using the magnitude since it makes the implementation much simpler, and likely not much would be lost in terms of performance given the used simulated data.

The inputs and targets to the network should be normalized. The input is difficult to normalize because the limits of the values of the input data are not necessarily known. Instead, a rough version of normalization is implemented by applying log(1+magnitude). This assures the inputs are larger than 0 and not extremely large. With these transformations, the input data is observed to be in the range between 0 and 10, which is deemed to be sufficient for this thesis.

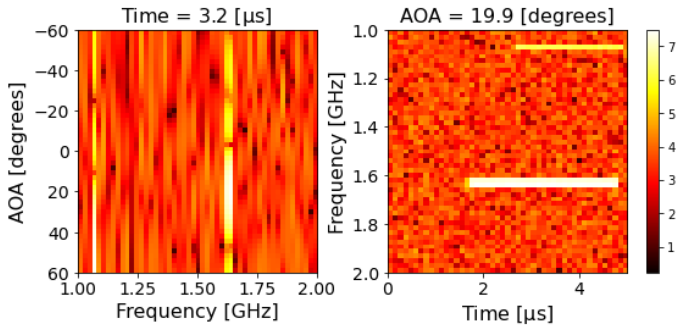


Figure 3.1: A preprocessed data sample with SNR=0dB. Time and AOA at the top indicate which slice of the 3D volume is shown.

The input data to the network has been acquired. In Fig. 3.1 one data sample is shown. Since the data is 3D, two cross-sections of the 3D image are shown. At the top of the image, it is indicated which slice is shown. With this illustration, it is however not certain that both signals will be shown. The AOA and time for the middle of the pulse for one of the signals are used to decide which slices are shown.

But if the other pulse is at a different time or has destructive interference for that AOA the other pulse will not be visible in one or both of the slices.

3.2.4 Target data

To train a network some targets must be defined for the network to train against. The target data is generated from the same parameters used to simulate the signal. To describe a signal the four signal parameters AOA, RF, TOA, and PW are used.

As in the original YOLO paper [15] the image is divided into a grid, but a 3D variant. Since the image is roughly cubic to start with a cubic grid is used. A (7, 7, 7) grid is used, resulting in 343 cells. For each cell, the four signal parameters and a confidence score are predicted for the cell that is responsible for detection. The target vector is thus a (7, 7, 7, 5) matrix, with a length of five vectors for each cell describing the target. The five values are [AOA, RF, TOA, PW, Confidence]. The confidence score is set to 1 if the cell is responsible for detection and to 0 if it is not. Since there is only one bounding box target for each cell it is impossible to detect more than one signal per cell. In Fig. 3.2 the bounding boxes are shown for a data sample. Since there only exists a width in time we have added a width of 2 pixels in the images for AOA and frequency for clarity.

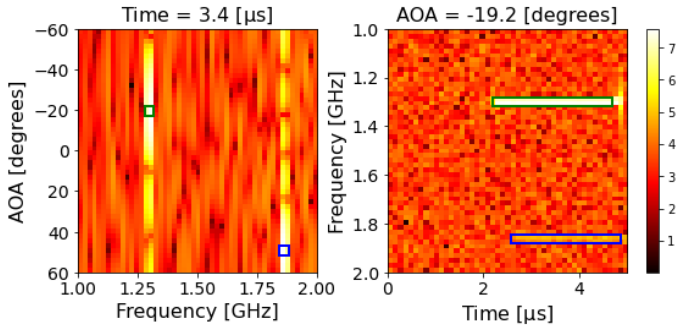


Figure 3.2: A preprocessed data sample with SNR=0dB. Bounding boxes for the target signal parameters are shown with an increased width of two pixels in the AOA and frequency dimensions. Note that the blue signal appears very weak since the frequency-time slice of the 3D volume is shown for AOA $= -19.2°$ which is a destructive interference angle for the blue signal.

The target vector should be normalized between 0 and 1, and should also be normalized within each cell. AOA should be 0 for the minimum value within the cell and 1 for the maximum value. Similarly for RF. In the time dimension, we also have the width of the bounding box. In image recognition, the center and the width of the bounding box are commonly used, thus for TOA the middle of the pulse is predicted instead and that value is normalized to be between 0 and 1. The PW can be larger than one cell so it is instead normalized against the entire width of the image. The confidence is already defined to be between 0 and 1.

3.3 Neural Network Design

Designing a network architecture, loss function, and tuning the training hyperparameter is an iterative process. This section will explain the processes of how the network took shape.

3.3.1 Initial Attempts

Initially, attempts were made using a regular CNN without the YOLO structure with grids and without residual connections. This worked for initial attempts with only one signal. For the two signal case, it is not as straightforward, since it is ambiguous what part of the vector should predict which signal. Simply training the network directly on the data and targets would most likely not be successful. For example, if the network predicts the targets in the wrong order it would give a large loss even though it predicted correctly. An attempt to solve this was made by making the loss invariant to the order of the targets. However this was not successful, and the network was not able to specialize in predicting the two separate targets. After training the network ended up predicting both targets being almost at the same location somewhere in between the two actual signals. Many training iterations with changed hyperparameters, increased data sets, and data augmentation ended up with either this result or the network becoming overfitted to the training data. Thus the YOLO-inspired network was created and also the residual connections were introduced.

3.3.2 YOLO and ResNet34 Inspiration

The architecture of the network in this thesis is inspired by the 34-layer residual network (ResNet34) in [14] and the YOLO network in [15]. The ResNet34 is designed for much higher resolution images and then used to classify images from 1000 classes. The YOLO network is used for classifying 30 different classes of objects and also predicts bounding boxes around the objects. We have a 3D image that is quite low resolution and a target vector of a (7, 7, 7) grid with a length of 5 target for each cell. It is attempted to construct a neural network architecture that will suit the input and targets that incorporates the ideas from the ResNet34 and the YOLO networks. We choose to look primarily at the first YOLO paper and not as much at the later papers since we have a quite different problem. The later YOLO versions make improvements for their problems but we are more interested in the basic ideas of YOLO to build our own network upon.

3.3.3 Building blocks

In our network, two different kinds of residual blocks are used. Both use kernels of size 3 and a padding of 1 and two convolutional layers between the shortcut connection.

The first type of residual block is the normal residual block which keeps the same dimension of the data, by using a stride of one and keeping the number of channels

the same. The block is simply passing the input through the two convolutional layers before adding the input to the output.

The second kind of block downsamples the data. The downsampling block uses a stride of 2 in the first layer which decreases the spatial dimension of the data. The number of channels is on the other hand doubled. The second layer keeps the same dimensions as the output of the first. To be able to add the input of the first block to the output of the second they need to have the same dimensions. To do that we do as suggested in [14] and use a convolutional layer with a kernel of size one and a stride of two, and the number of channels as desired.

3.3.4 Network Architecture

After the preprocessing of the data, the input shape to the network is (1, 50, 51, 47), with dimensions (Channel, θ, f, t). The first layer uses a kernel of size 5 and a stride of 2 and increases the number of channels from 1 to 16. Then two normal residual blocks, followed by one downsampling block, one normal, one downsampling block, and finally another normal residual block. The goal of the downsampling is to decrease the spatial dimensions of the data to the shape of our desired grid, in our case (7, 7, 7), while increasing the number of channels that hopefully contain information about the features of the image. At the last layer, a kernel with size one is used to aggregate the information from the different channels down to 5 outputs for each cell. With some reordering of the data, the output is of the same shape as the targets and thus can be trained to match it. This architecture is illustrated in Fig. 3.3.

After all layers except the last, batch normalization [18] is used. The Rectified Linear Unit (ReLU) activation function is used after every layer except the last, but when using a shortcut connection it is used after adding the input and the output of the block together. After the last layer, a sigmoid function is used to obtain outputs in the range between 0 and 1. Recall that the target vector is normalized such that all values are between 0 and 1.

3.3.5 Tried but discarded additions to the network

We tried several additions to the network that were not used in the end. Attempts were made dropout layer after the first layer but no significant improvements were seen. The residual connections may have filled the same function instead [14]. Data augmentation was deemed unnecessary since more training data was generated more easily. Pooling layers were tried but since the dimensions are not high to start with it seemed better to instead use the downsampling layers. If starting with a higher-resolution image in future experiments, adding some kind of pooling layer might be necessary to decrease the computational requirements. Attempts were also made using binary cross entropy loss for the confidence score, since the output ideally should be either 0 or 1. It produced decent results but the training of the network was unstable.

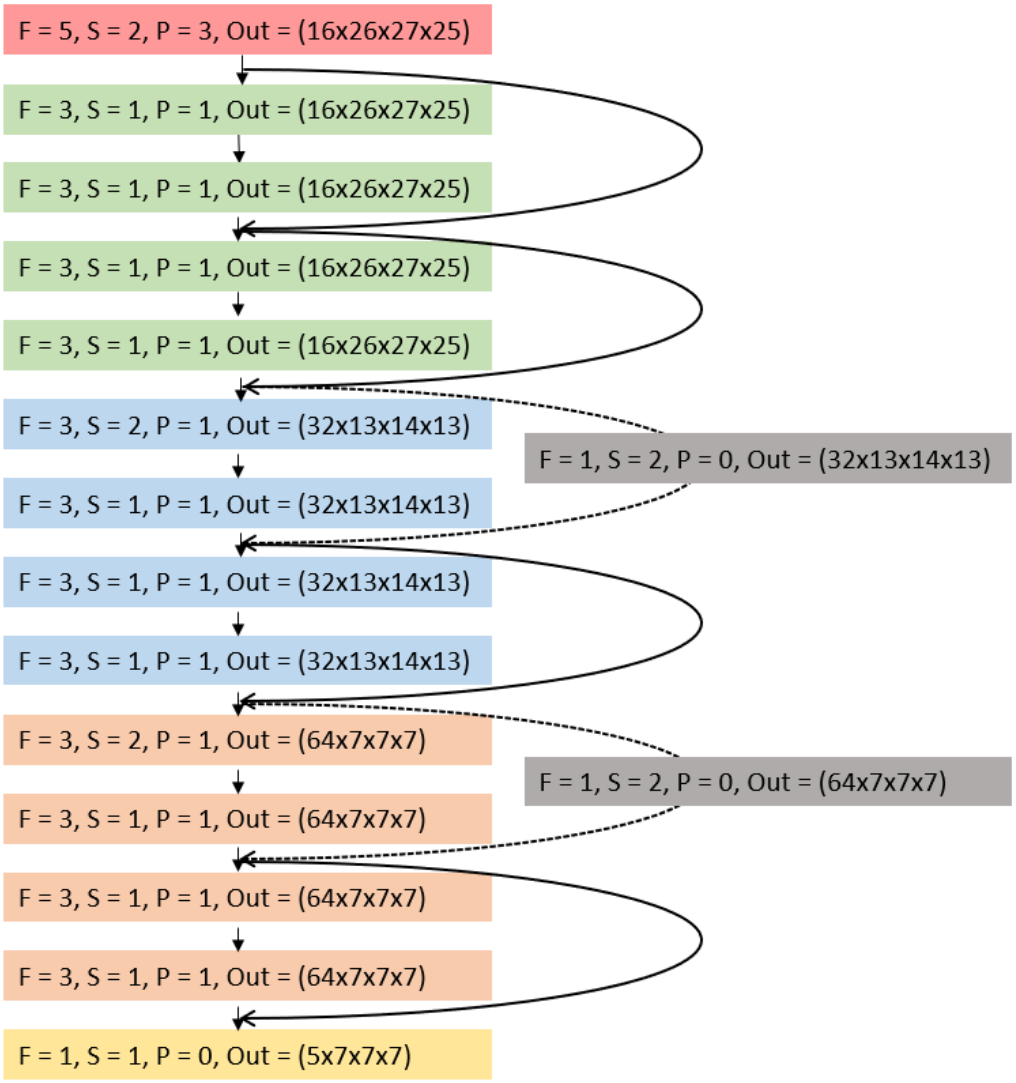


Figure 3.3: The network architecture. Input is of shape (1x50x51x47). F is the kernel size, S is the stride, P is the zero-padding, and Out is the output shape after the layer. Shortcut connections are shown with arrows. Shortcuts between layers of different output shapes are shown with a dashed line.

3.4 Training and evaluation

The network needs to be trained and afterward, the performance of the network evaluated. In this section, the loss and hyperparameters used for training the network are described. It also is explained what metrics are used to evaluate the performance.

3.4.1 Loss function

To optimize the network a loss function with three parts is used. The first is that the network should predict a confidence score of 1 for the cell responsible for detection. The second part is that it similarly should predict zero for cells not responsible for detection. Thirdly, for a correct prediction, the error for the AOA, RF, TOA, and PW should be minimized. To accomplish this essentially the same mean squared error loss function as in the YOLO paper [15] is used, but exempt of the class probability part since we don't have classes. The loss is thus described by:

$$\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^3} \mathbb{1}_i^{\text{obj}} \left[(\text{AOA}_i - \text{AOA}_i^*)^2 + (\text{RF}_i - \text{RF}_i^*)^2 + (\text{TOA}_i - \text{TOA}_i^*)^2 \right] \\
&\quad + \lambda_{\text{coord}} \sum_{i=0}^{S^3} \mathbb{1}_i^{\text{obj}} (\sqrt{\text{PW}_i} - \sqrt{\text{PW}_i^*})^2 \\
&\quad + \sum_{i=0}^{S^3} \mathbb{1}_i^{\text{obj}} \left[(\text{C}_i - \text{C}_i^*)^2 \right] \\
&\quad + \lambda_{\text{noobj}} \sum_{i=0}^{S^3} \mathbb{1}_i^{\text{noobj}} \left[(\text{C}_i - \text{C}_i^*)^2 \right],
\end{aligned} \tag{3.3}$$

where * indicates the target value, $\mathbb{1}_i^{obj}$ denotes if the center of the signal is in the cell i in the (S, S, S) grid, and thus is responsible for detecting the signal. Similarly, $\mathbb{1}_i^{obj}$ denotes if the cell is not responsible for detection. To weigh the terms of the loss function $\lambda_{coord} = 5$ and $\lambda_{noobj} = 1$ are used. An equal error for a long and a short signal should produce a similar loss. Thus the square root of the PW is used to roughly achieve this.

3.4.2 Metrics

To study the performance of the network some metrics are defined. The number of correct and incorrect predictions by the network is studied, and also the Root Mean Square Error:

$$\text{RMSE} = \sqrt{\frac{\sum_{n=0}^{N-1} (y_n - y_n^*)^2}{N}}, \tag{3.4}$$

of the signal parameters for the correctly predicted signals in the data set. How is a signal classified as correct? For simplicity, a prediction is defined as correct if the network predicts the same cell as the target as responsible for detection. This is

done by looking at the predicted confidence of the network output. A confidence of over 0.5 is defined as the network predicting the cell is responsible, and if lower it is not responsible.

3.4.3 Training hyperparameters

Training hyperparameters were determined through trial and error. The optimization algorithm AdamW [19] was used. Some attempts were made using regular Adam [20] and Stochastic Gradient Decent, but they had no obvious performance increase so it was decided to continue with AdamW. For AdamW the only hyperparameters that changed from their default values in the PyTorch implementation were the learning rate and the weight decay. The weight decay used in the final model was 0.001, compared to the default 0.01. For the learning rate, the training was initiated with a learning rate of 0.01 and then decreased by a factor of 0.1, when the training loss had not decreased for 10 epochs until the learning rate became less than 10^{-6}. The learning rates for the different epochs are shown in Fig. 3.4. The network was trained for 377 epochs, with a batch size of 64.

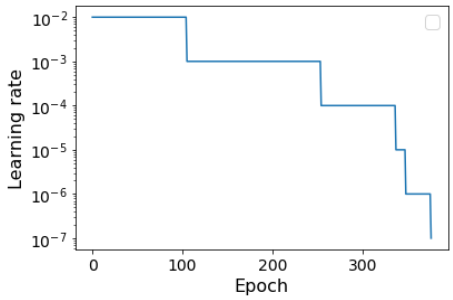


Figure 3.4: Learning rate used during training of the network.

3.5 Tools used

The data generation, preprocessing, and network were implemented using Python and PyTorch. A computer with an NVIDIA GeForce GTX 1080 Ti GPU and an Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz CPU was used to run the code.

4

Results

In this chapter, the results from the training and evaluation of the network will be presented.

4.1 Training of the network

In Fig. 4.1 the loss for the training and validation data is shown. The loss for the training data is steadily decreasing, but the loss for the validation data only decreases a little before then starting to increase for later epochs. It will be explained why this happens and why it despite that results in a network that performs quite well.

Instead, the other metrics defined to evaluate the network can be studied. The percentage of correct predictions is shown in Fig. 4.2(a) and the average number of incorrect predictions per image in Fig. 4.2(b). The RMSE for the four signal parameters is shown in Fig. 4.3. For the percentage of correct predictions, the network predicts almost all of the correct cells for the validation data at the beginning of the training. But it also predicts more than one incorrect signal per image. When further training the network the number of correct predictions as well as the number of incorrect predictions decreases. Optimally the percent correct would stay high while the number incorrect would decrease. For the training the percentage of correct predictions does not decrease, it is instead increasing to almost 100%. This

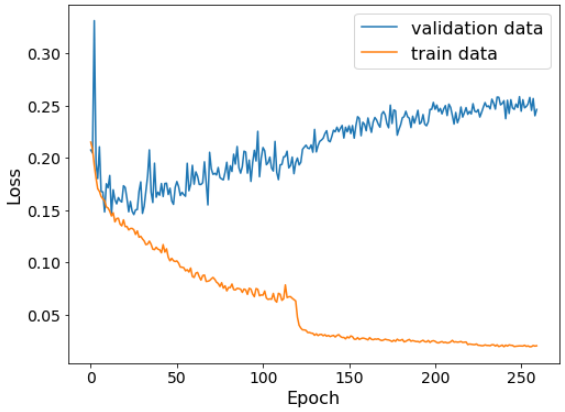
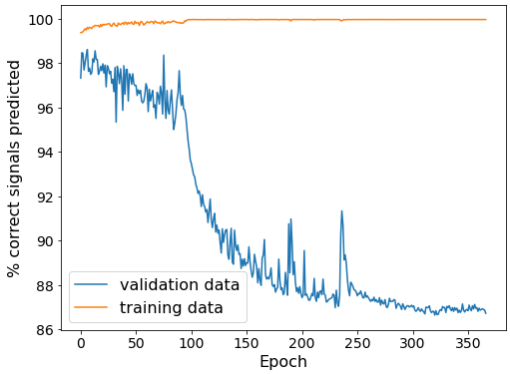
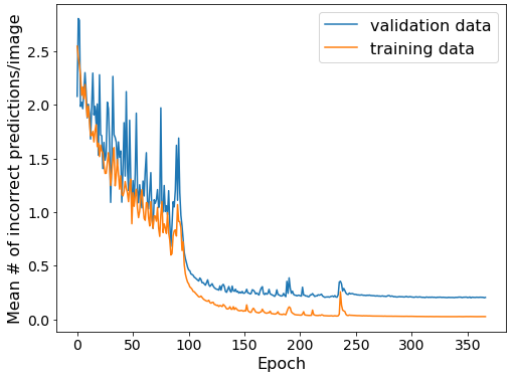


Figure 4.1: Training and validation loss during training of the network. Loss according to equation (3.3).

metric is the only one where the metric for the validation does not roughly follow the pattern of the training. This is the explanation for why the training loss decreases while the validation loss increases.

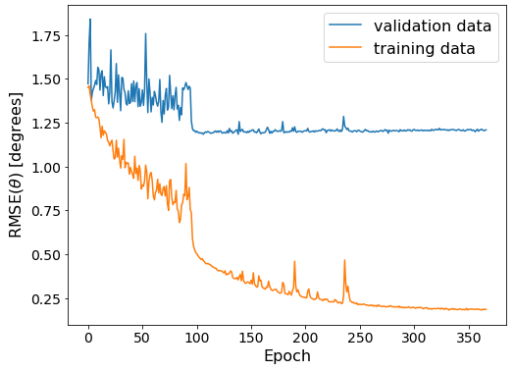


(a) Percentage correct predictions by the network during training.

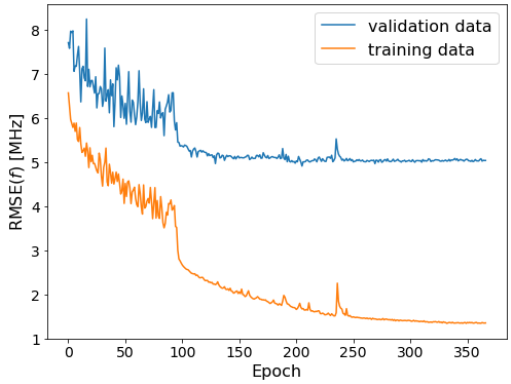


(b) Average number of incorrect predictions by the network per image during training.

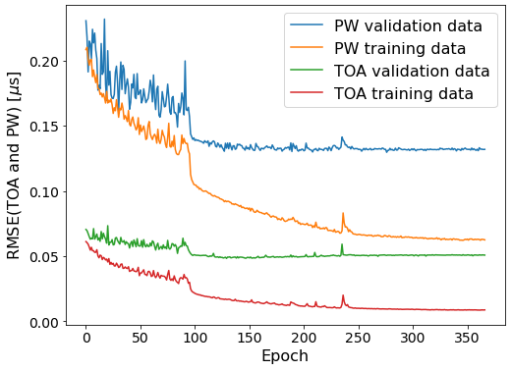
Figure 4.2: Metrics measuring how many of the network's predictions are correct or incorrect.



(a) RMSE for the AOA.



(b) RMSE for the frequency.



(c) RMSE for the TOA and PW.

Figure 4.3: RMSE metrics for the different signal parameters for the correct network predictions.

Does the discrepancy between the training and validation metric for the percentage correct predictions indicate that the network is overfitted? In some ways it does. The network quite quickly learns what cells it should use to predict the correct cells for the training data, but this does not perfectly translate to it predicting the correct cells for the validation data. This does however not necessarily mean that the network makes bad predictions.

Looking at our metrics for the validation data it seems like they have stabilized at around epoch 200, except for the number of correct predictions which continues to trend downward. However, when training for that shorter period and inspecting the predictions visually it seemed like the results were worse. How can that be if the metrics indicate that the predictions should be better? It is believed to be because of errors in the metrics. Consider this, at epoch 200 if a signal was at the edge of a cell it got two predictions, one by the correct cell and one by the neighboring. Then instead by epoch 300 the network instead only gave one prediction, but it gave it using the incorrect cell. There would still be the same amount of "incorrect" predictions but a decreased amount of "correct" predictions. This would explain why the metrics being "worse" at epoch 300 than at epoch 200 would still produce a better-performing network.

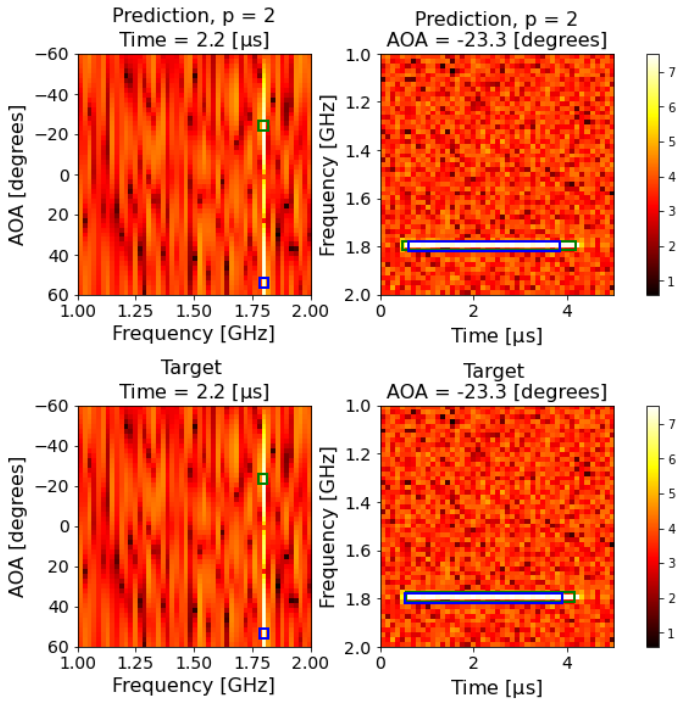


Figure 4.4: Predictions and targets for test data with SNR=0dB, p indicates the number of predictions made. Note that it manages to separate two signals with very similar RF, TOA, and PW.

4.2 Performance on test data

After training the network the predictions made on the test data set are studied. Some chosen samples will be used to take a closer look. A larger selection of predictions on the test data is found in App. A, where six predictions each for 6 different SNRs are shown. In the figures, the predictions are illustrated with an increased bounding box by two pixels in the frequency and angle dimensions. At the top of the images, the shown slice of the 3D image is indicated by the time or AOA value, and p indicates the number of predictions made by the network.

In general, the network performs very well. Fig. 4.4 shows that it manages to separate two signals at very similar frequencies, 1795 and 1798 MHz, and also with a large overlap in time for the signals.

There are some issues with the network either predicting too many or too few signals, especially for lower SNR. In Fig. 4.5 only one of the signals is correctly predicted and in Fig. 4.6 one superfluous signal is predicted.

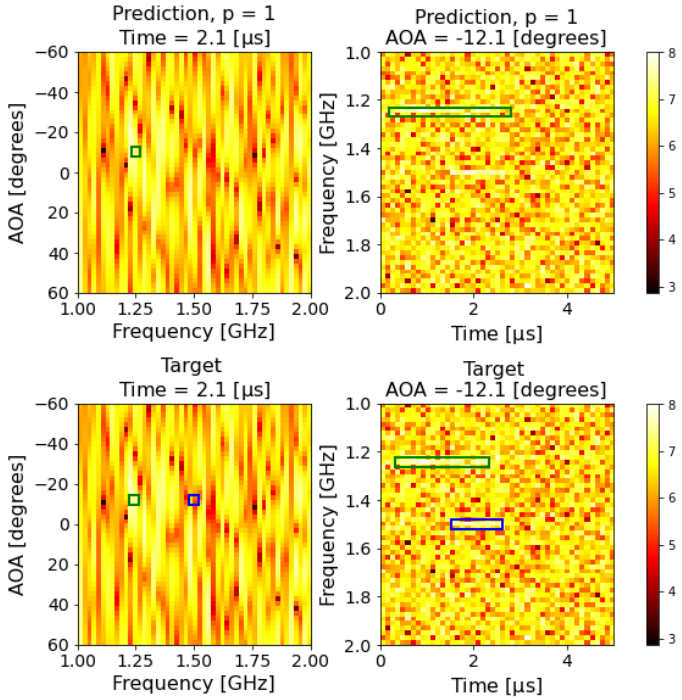


Figure 4.5: Predictions and targets for test data with SNR=-40dB. In this case, the network only finds one of the signals

Fig. 4.7 shows a case where the network makes two good predictions but our metrics say this is one correct and one incorrect prediction. This is caused by the blue signal for which the AOA is close to the edge between two cells. The true AOA is -8.09°and the predicted is -8.58°. Half a degree off is a very decent prediction but the grid cell border is at -8.57°and thus the prediction is classified as incorrect.

After visually inspecting the results, the metrics are applied to the entire test data

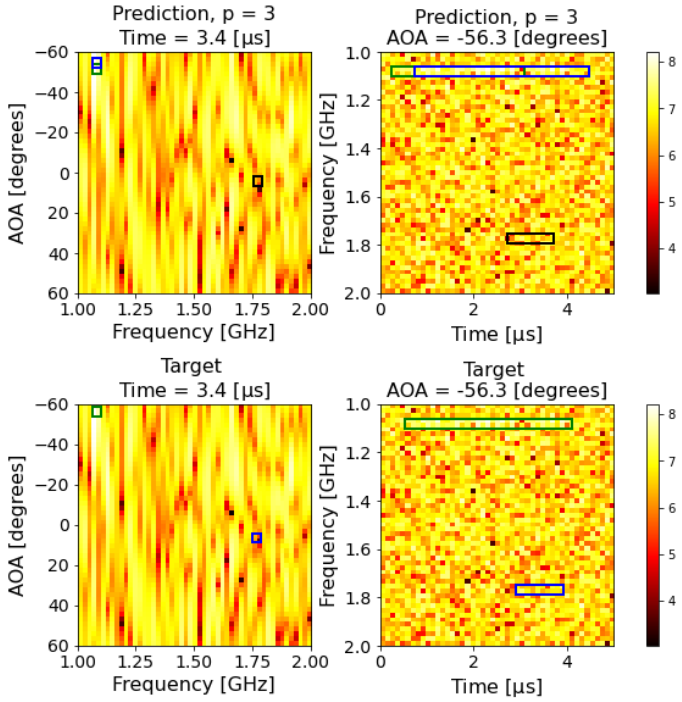


Figure 4.6: Predictions and targets for test data with SNR=-40dB. In this case, the network incorrectly predicts an additional signal.

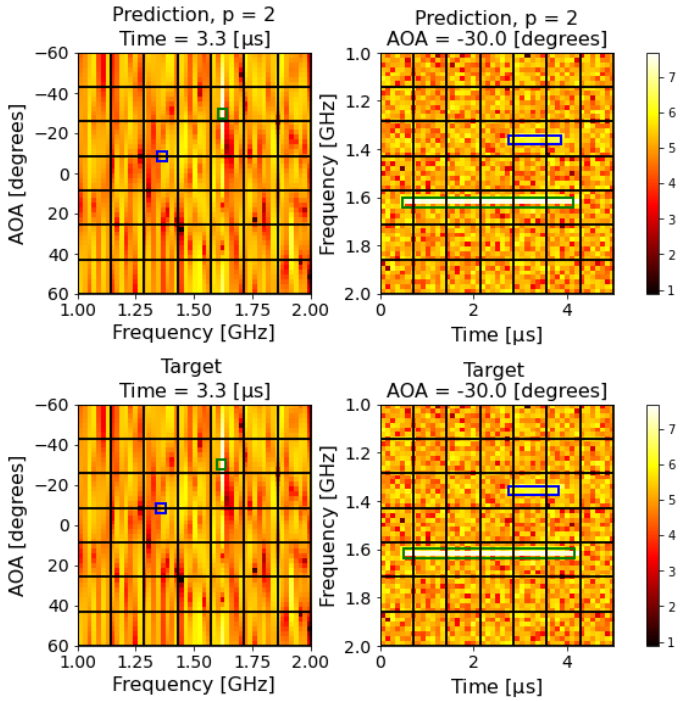
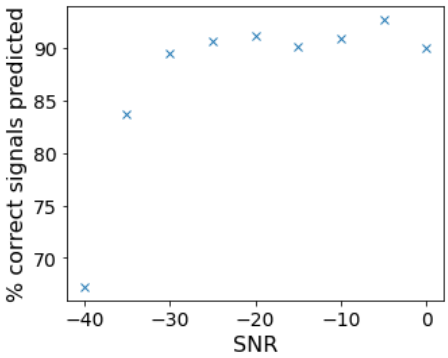
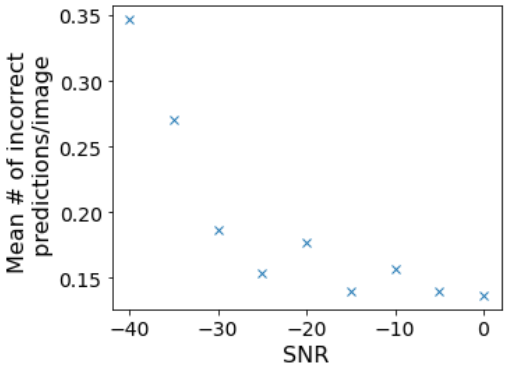


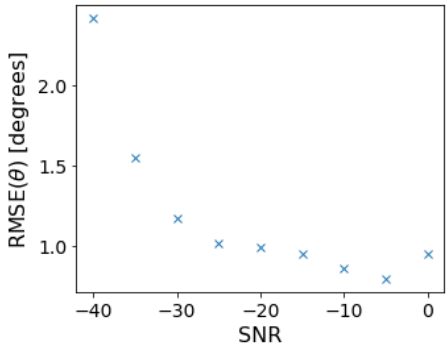
Figure 4.7: A prediction where our metrics say our green prediction is correct but our blue prediction is incorrect due to the ”wrong” grid cell being used for the prediction.



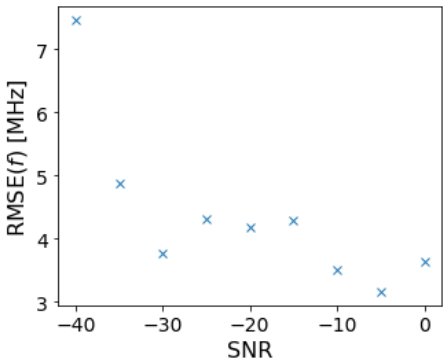
(a) Percentage correct predictions by the network for the test data.



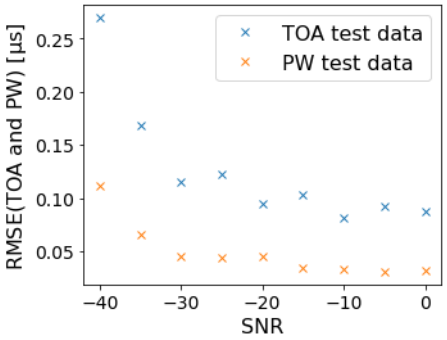
(b) Average number of incorrect predictions by the network per image for the test data.



(c) RMSE for the AOA for the correct predictions made on the test data.



(d) RMSE for the frequency for the correct predictions made on the test data.



(e) RMSE for the TOA and PW for the correct predictions made on the test data.

Figure 4.8: Metrics for the different pulse parameters for the correct network predictions on the test data.

set. The metrics in Fig.s 4.2 and 4.3 are for the entire set of the training and validation data, containing samples from -40dB to 0 dB. When the network is evaluated on the test data the performance of the different SNRs is studied separately. Recall that the SNR is in this report defined as the signal amplitude over the noise floor for a *single* antenna element. Studying Fig. 4.8 some trends appear for the different SNRs. Most notably that the network works increasingly poorly for lower SNRs.

It was also attempted to use the network for -50dB signals. There it either made no or incorrect predictions, for some examples see App. A. With our eyes, it is impossible to distinguish the signals from the noise and thus it is not unreasonable that the network which was designed using regular computer vision methods is not able to detect the signals. The network was trained without this SNR in the data set for the final model. When it was included, the performance for all SNRs was worse than in the final model. This is most likely because with this network design, it is impossible to learn how to find those low SNR signals and thus when it tries to learn that anyway it instead worsens the performance.

5

Discussion

With our research objective and limitations in mind, it is clear from our results that we have managed to create a method that can successfully detect two sinusoidal signals and estimate their parameters. Now it remains to investigate how well it performs and what possible improvements could be made. In this Chapter, a discussion is held regarding the performance, evaluation, and design of the neural network. Further, possible areas of improvement are discussed and possible investigations in future work are proposed.

5.1 Performance of the network

The obtained RMSE for the metrics is smaller than the resolution of the input image, which may seem improbable. But the network does have information from the nearby pixels as well. Imagine that the network only finds the pixel with maximum amplitude and decides the angle from that. In that case, it would not achieve this high RMSE. But the network can also look at the other pixels, and perhaps if the pixel on one side has a higher amplitude than the pixel on the other. The network could use this kind of information to make a "fit" and from that infer an AOA that is more precise than the resolution of the input image.

In the evaluation metrics of the network, there exist cases when a signal center may be close to the edge of a grid cell and the network predicts the signal center to be in the neighboring cell, resulting in an accurate prediction with a low RMSE. But in the correct cell, it predicts there to be no signal. This case would be counted as if the network did not make a correct prediction for the signal, instead, it would actually contribute to the number of wrong predictions. In Fig. 4.7 we showed one occurrence of this. It has not been investigated how common this issue is. Similarly, the network may make multiple close predictions for the same signal in neighboring cells. One has to consider how large of an issue it is that one signal is predicted twice but with slightly different parameter value estimations. Maybe some heuristics can be used to combine the guesses into one guess if the signals are close enough to each other. This would mean an increased uncertainty in the parameters, but depending on the application this may or may not be an issue.

The SNR level given is for one antenna element in the time domain. To better estimate the performance of the network, the processing gain from the STFT and the beamforming should be taken into account.

5.2 Loss and metrics

The metrics of how often the network guesses correctly could be improved upon. It makes the assumption that if it is predicted that there is a signal in the cell responsible for the detection of a signal, a correct prediction has been made. If the network makes a prediction to a nearby cell, that is classified as an incorrect prediction. If a signal has a center close to the edge of a cell and the network uses the nearby cell it would not produce a correct prediction by this heuristic even though the RMSE for the bounding box for that prediction may even be smaller than some other prediction that was classified as correct. The heuristic also puts an upper bound on the RMSE for the bounding box. It is required in some way to define when a predicted signal is correct or not, ideally, you would set some value for the RMSE for the bounding box. Here a version of this has been implemented, but the limit depends on where in the cell the signal is.

It has been concluded that the metric used is not optimal. But it is not clear how it should be improved. The simplest way would be to implement some heuristic where the distance between the prediction and the target is measured and the prediction would belong to the target it is closest to. If there is more than one prediction belonging to the same target the closest target would be chosen and the others are incorrect. It would also be required to determine some maximum distance a prediction can be from a target before it is instead classified as incorrect. It is however unclear how this distance would be defined since we have different units in the different dimensions. Also, how would the PW be represented in this distance?

Another interesting approach to a different metric is the Intersection over Union (IoU) which is a popular metric to use for object detection. Its most basic form is simply the intersection between the predicted bounding box and the true divided by their union. There are several smart improvements to the metric that can also be applied [21]. However, we do not think it would be suitable to use for the current kinds of targets. One minor problem is that 3D bounding boxes are used which would require an extension of the metric. The larger issue is that our bounding box is not a volume, but a line. Therefore most of the time the intersection of the lines would be zero. Thus it is not trivial to convert to an IoU metric. In [22] a different approach instead of bounding boxes is used. Instead, centerline and some offsets are used. There are many different ways to create a different way of defining some metrics and a new loss but it is not clear what would be suitable to our problem.

5.3 Network design

Note there was no assumption made of there only existing two signals in the design of the network or the targets. Thus it should not be a problem to apply the exact same network to more than two signals. If the network would be retrained with input data with a varying number of signals the network should be able to handle a larger number of signals than two.

It is required to choose some threshold value for detection. In our target data, the

confidence score should be 0 if the cell is not responsible for detecting the signal and 1 if the cell is responsible. But the network will produce a number between 0 and 1, close to 0 if it is sure the cell has no signal in it, and close to 1 if it thinks the cell is responsible for detecting a signal. But what if the network predicts 0.405 or 0.673? At what threshold should the network predict there to be a signal at that location? The simplest solution is to put the threshold at 0.5, which is what was done in this thesis. By studying the performance of the network for different threshold values, perhaps the method could be improved upon. If the network makes many false positive predictions possibly increasing the threshold would improve the performance of the network.

One worry at the start of the project was that the side lobes would be interpreted as separate signals by the network. But that was not the case, the network learned to only use the main lobe. However, this prompts the question if the network could distinguish the main lobe from grating lobes, that occur in for example sparse arrays. Investigation, if this method could be used in the presence of grating lobes, would be an interesting avenue for future work.

In this thesis, only the azimuth angle was considered. In reality, the elevation angle should be estimated as well. When scanning over all possible elevation and azimuth angles a fourth dimension to the image would be added which would increase the amount of data. This should be possible to be implemented with an extension of the current method. This would require a for example square of a circular array instead of a linear array.

5.4 Possible Improvements

More fine-tuning of the network training hyperparameters and loss is possible, but a sufficiently good model has been obtained to show that this method is viable. The next step to evolve this method would not be to make this model even better but instead to expand the model to more signals, other signal types, higher input data resolution, etc. Such an expansion of the model would require the underlying neural network to be adapted and likely more complicated, and then a new tuning of the hyperparameters would have to be performed. In this thesis, the original version of the YOLO network was used as a basis. There have been many improvements to it since and some of those improvements should be able to be incorporated into the network.

It would be interesting to compare our results to some conventional methods. But it would be difficult to compare the methods using our chosen metrics. Since those methods do not use grids how would a prediction be defined as correct or not? We also did not find some method that jointly detected and estimated the AOA, RF, TOA, and PW. It would also be preferable to use a method that makes use of the digital antenna array in a comparison.

In this thesis, an idealistic antenna model has been used. While sufficient for initial use a more realistic antenna model should be used going forward.

5.5 Future work

The most evident future work would be to extend to more than two signals. Furthermore, to include more complex signal shapes, with some bandwidth in the frequency or a chirp. Different amplitudes for the signals should also be implemented. The network should then also be extended to also predict the amplitude of the signal.

The network can likely be extended to also classify different types of signals. The YOLO framework already is able to predict classes, but in this thesis, this ability was not utilized since only one class of signals was used. To perform classification the IQ data would need to be used as input for the network instead of only the magnitude as used in this thesis.

The loss and metrics used have been concluded to be an area of improvement, thus in future work, it would be recommended to investigate and define some new loss and evaluation metrics. The new loss would also have to be adapted to the proposed additions to the target vector, with an ability to predict for example the class, the amplitude, and width in frequency.

With further investigation into this method, a study of the speed and accuracy of the method would be required. Comparison to some alternative methods would be of interest. The simulations should also be made more realistic. Both in terms of antenna modeling and also by considering the elevation angle, and not only the azimuth angle.

This project has only utilized simulated data but it needs to be investigated if the method is possible to use on real-world data. A future project could collect real-world data and try to detect incoming signals. We would suggest that simulated data still be used for training, or possibly in combination with real-world data as training data. However, this poses a challenge in how to define the bounding box of real-world data.

6

Conclusion

In this thesis a method utilizing image recognition to detect, and estimate the angle of arrival, frequency, time of arrival, and pulse width of two unknown signals has been developed and tested. With the delimitations set for the project, such a method has successfully been developed. It is of note that except for very low SNRs, RMSE values that are smaller than the image resolution are obtained.

The developed method has potential, but further study is required to realize it. The loss function and the metrics used to evaluate the network have been identified as areas of improvement. Some investigation has been made into the method performance for different signal-to-noise ratios, but further investigation is required to determine how well the method performs in terms of speed and accuracy, preferably in comparison to some alternative methods. Interesting avenues of future studies could entail the extension of the method to more than two signals, wider parameter ranges, and to also include classification of the signals. These extensions should be possible with minor adjustments to the current method.

Bibliography

[1] T. J. O'Shea, J. Corgan, and T. C. Clancy, *Convolutional radio modulation recognition networks*, 2016. arXiv: 1602.04105 [cs.LG].

[2] N. Cummins, S. Amiriparian, G. Hagerer, A. Batliner, S. Steidl, and B. Schuller, "An image-based deep spectrum feature representation for the recognition of emotional speech", Oct. 2017, pp. 478–484. DOI: 10.1145/3123266.3123371.

[3] D. Park, S. Lee, S. Park, and N. Kwak, "Radar-spectrogram-based uav classification using convolutional neural networks", *Sensors*, vol. 21, no. 1, 2021, ISSN: 1424-8220. DOI: 10.3390/s21010210.

[4] T. O'Shea, T. Roy, and T. C. Clancy, "Learning robust general radio signal detection using computer vision methods", in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, 2017, pp. 829–832. DOI: 10.1109/ACSSC.2017.8335463.

[5] A. Vagollari, V. Schram, W. Wicke, M. Hirschbeck, and W. Gerstacker, "Joint detection and classification of rf signals using deep learning", in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, 2021, pp. 1–7. DOI: 10.1109/VTC2021-Spring51267.2021.9449073.

[6] K. Zhang, K. Sun, D. Yi, Y. Duan, and X. Meng, "A joint detection and recognition algorithm of non-cooperative communication signal based on improved yolov3", in *Proceedings of 2022 10th China Conference on Command and Control*, Singapore: Springer Nature Singapore, 2022, pp. 211–222, ISBN: 978-981-19-6052-9.

[7] G. K. Papageorgiou, M. Sellathurai, and Y. C. Eldar, "Deep networks for direction-of-arrival estimation in low snr", *IEEE Transactions on Signal Processing*, vol. 69, pp. 3714–3729, 2021. DOI: 10.1109/TSP.2021.3089927.

[8] S. Chakrabarty and E. A. P. Habets, "Broadband doa estimation using convolutional neural networks trained with noise signals", in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2017, pp. 136–140. DOI: 10.1109/WASPAA.2017.8170010.

[9] R. Fan, C. Si, W. Yi, and Q. Wan, "Yolo-doa: A new data-driven method of doa estimation based on yolo neural network framework", *IEEE Sensors Letters*, vol. 7, no. 2, pp. 1–4, 2023. DOI: 10.1109/LSENS.2023.3241080.

[10] Y. Qin, "Deep networks for direction of arrival estimation with sparse prior in low snr", *IEEE Access*, vol. 11, pp. 44 637–44 648, 2023. DOI: 10.1109/ACCESS.2023.3273126.

[11] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition", in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 922–928. DOI: 10.1109/IROS.2015.7353481.

[12] R. Janssens, G. Zeng, and G. Zheng, "Fully automatic segmentation of lumbar vertebrae from ct images using cascaded 3d fully convolutional networks", in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, 2018, pp. 893–897. DOI: 10.1109/ISBI.2018.8363715.

[13] R. G. Lyons, *Understanding digital signal processing*. Prentice Hall PIR, 2011, ISBN: 978-0137027415.

[14] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. DOI: 10.48550/ARXIV.1512.03385.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, 2016. arXiv: 1506.02640 [cs.CV].

[16] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement*, 2018. arXiv: 1804.02767 [cs.CV].

[17] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, *Yolov4: Optimal speed and accuracy of object detection*, 2020. arXiv: 2004.10934 [cs.CV].

[18] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, 2015. arXiv: 1502.03167 [cs.LG].

[19] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, 2019. arXiv: 1711.05101 [cs.LG].

[20] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[21] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, *Distance-iou loss: Faster and better learning for bounding box regression*, 2019. arXiv: 1911.08287 [cs.CV].

[22] W. Li, K. Wang, and L. You, "A deep convolutional network for multitype signal detection and classification in spectrogram.", *Mathematical Problems in Engineering*, pp. 1–16, 2020, ISSN: 1024123X.

A

Predictions from the Network

In this appendix, more data samples with targets and predictions are shown. Five random samples for a selection of different SNRs from the test data set are illustrated.

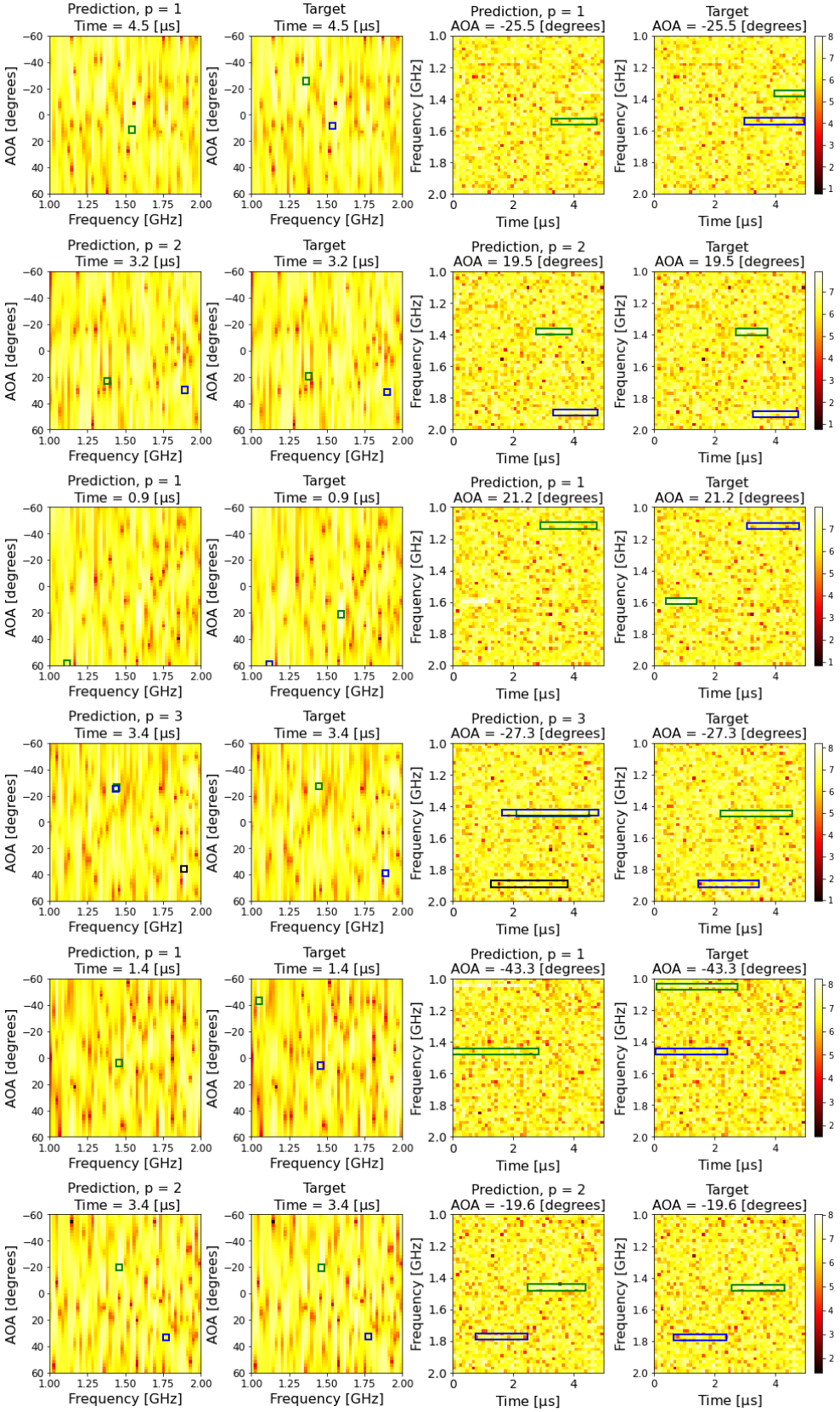


Figure A.1: Predictions and targets for test data with SNR=-40dB, p is the number of predictions made. One row shows predictions and true targets for one sample.

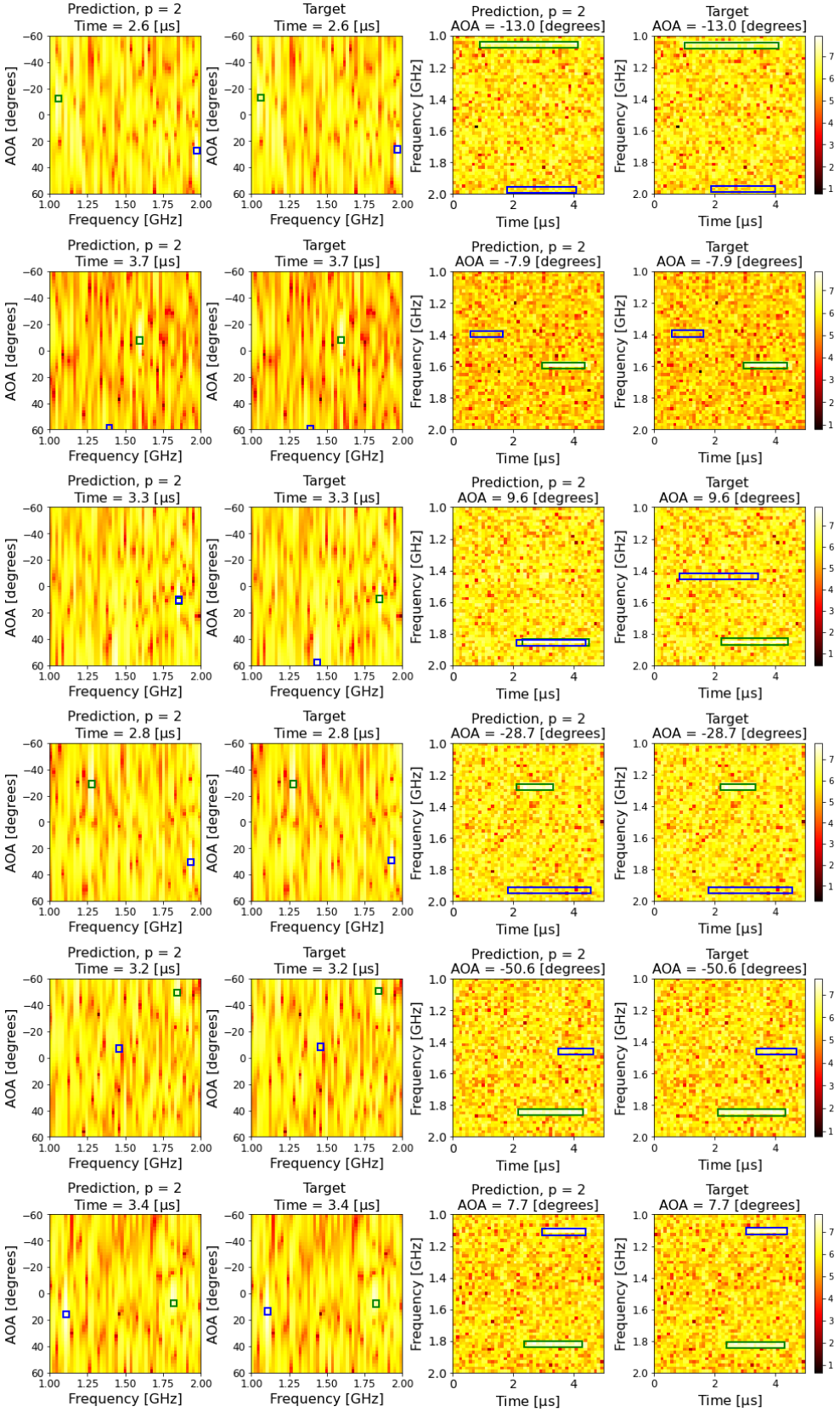


Figure A.2: Predictions and targets for test data with SNR=-30dB, p is the number of predictions made. One row shows predictions and true targets for one sample.

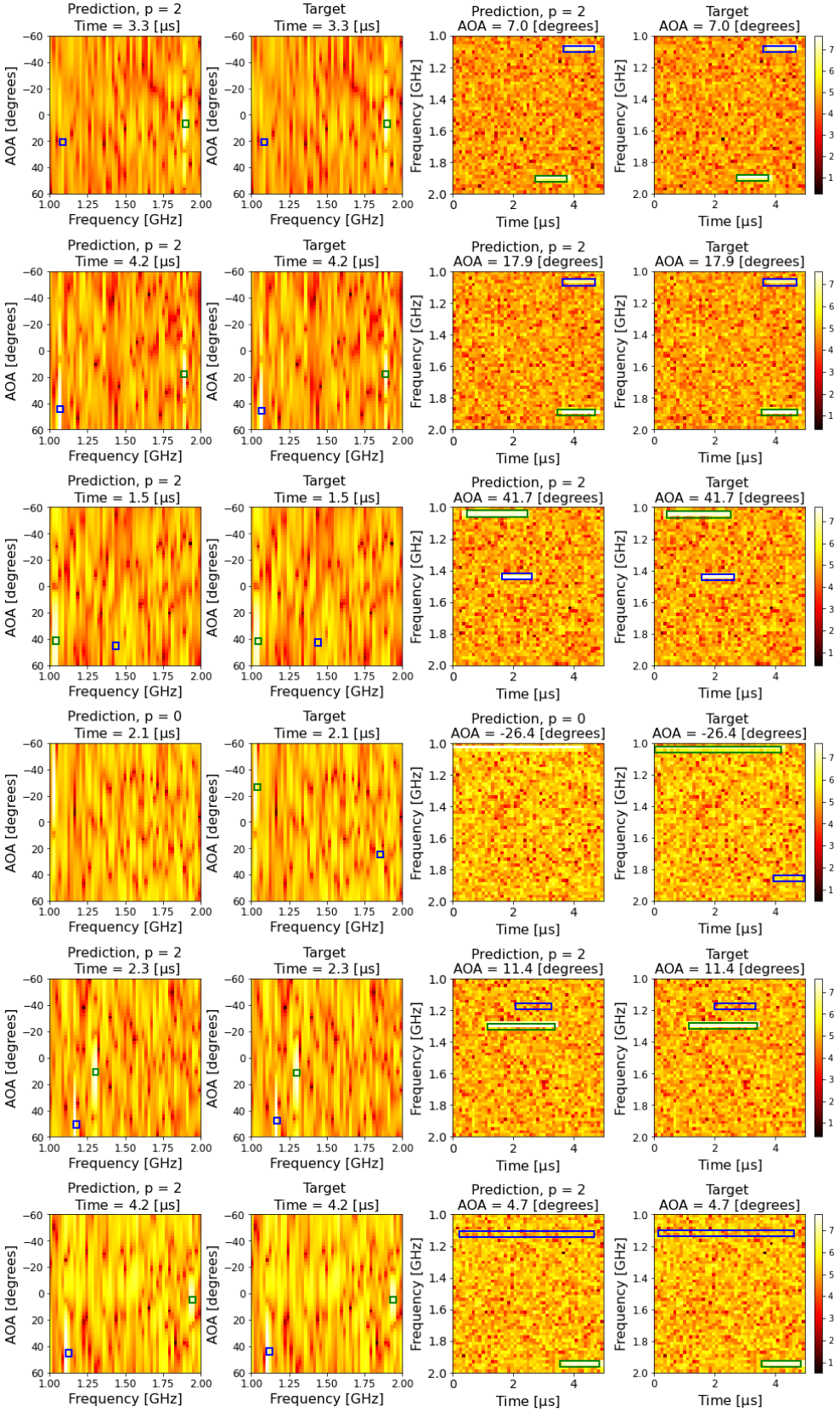


Figure A.3: Predictions and targets for test data with SNR=-20dB, p is the number of predictions made. One row shows predictions and true targets for one sample.

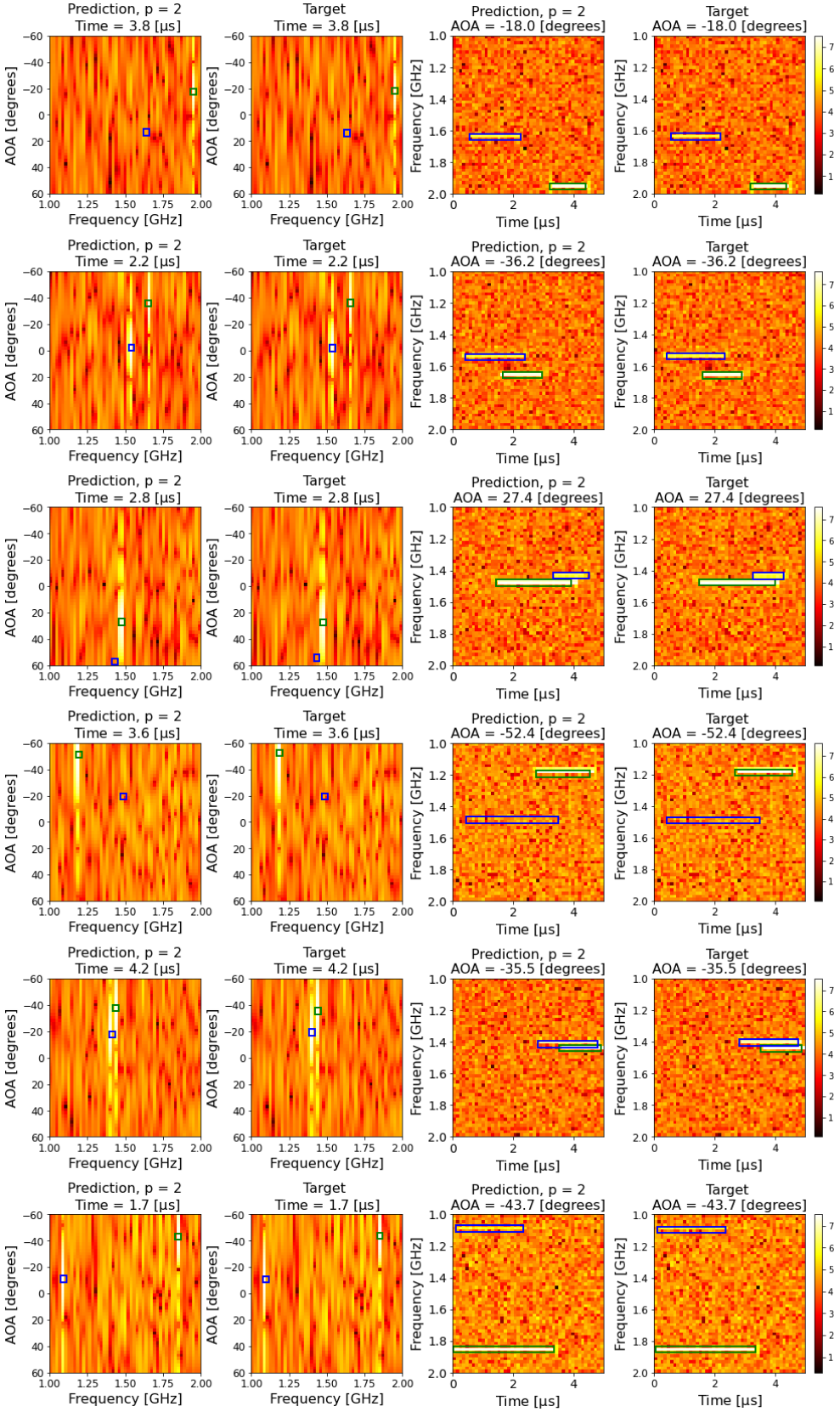


Figure A.4: Predictions and targets for test data with SNR=-10dB, p is the number of predictions made. One row shows predictions and true targets for one sample.

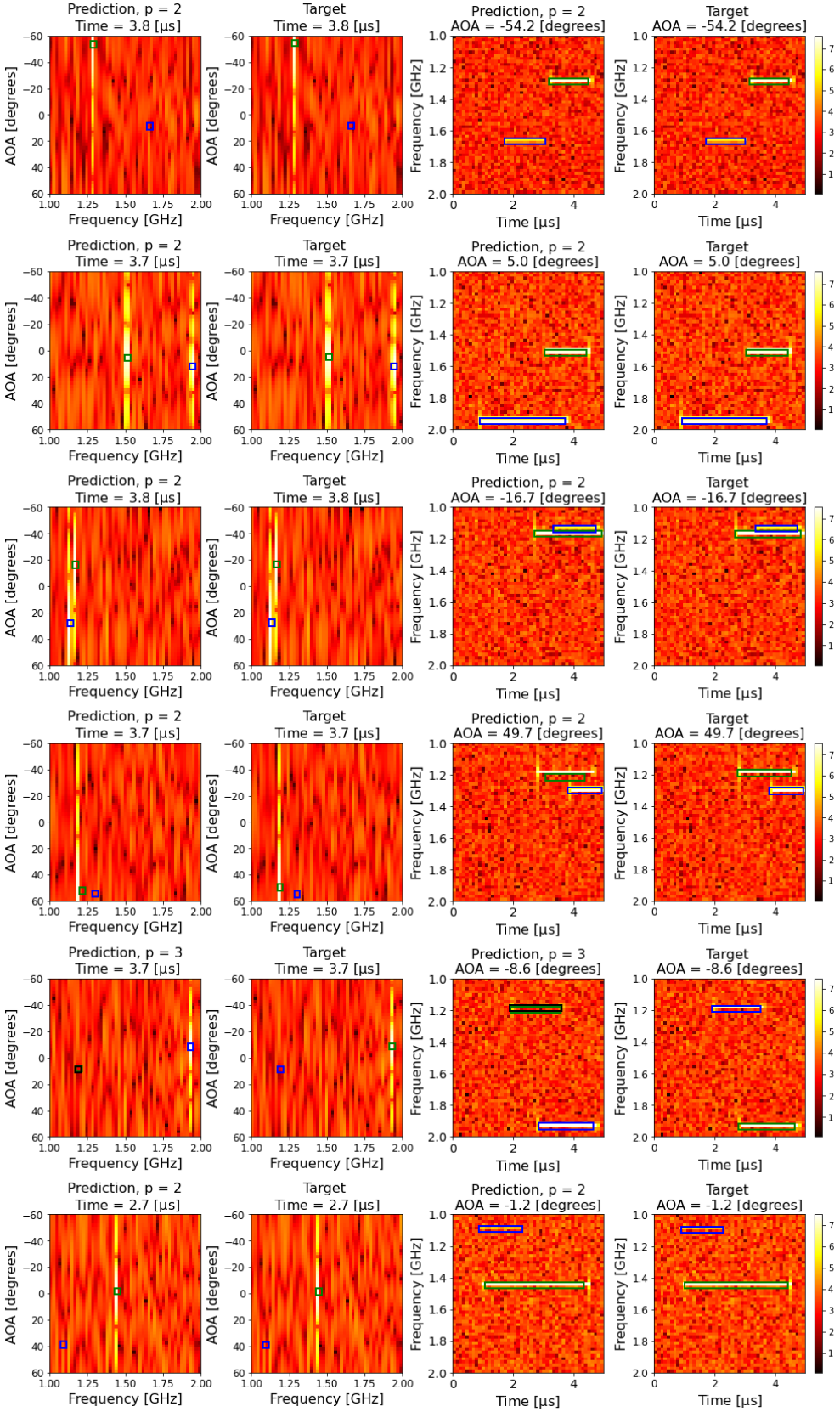


Figure A.5: Predictions and targets for test data with SNR=0dB, p is the number of predictions made. One row shows predictions and true targets for one sample.

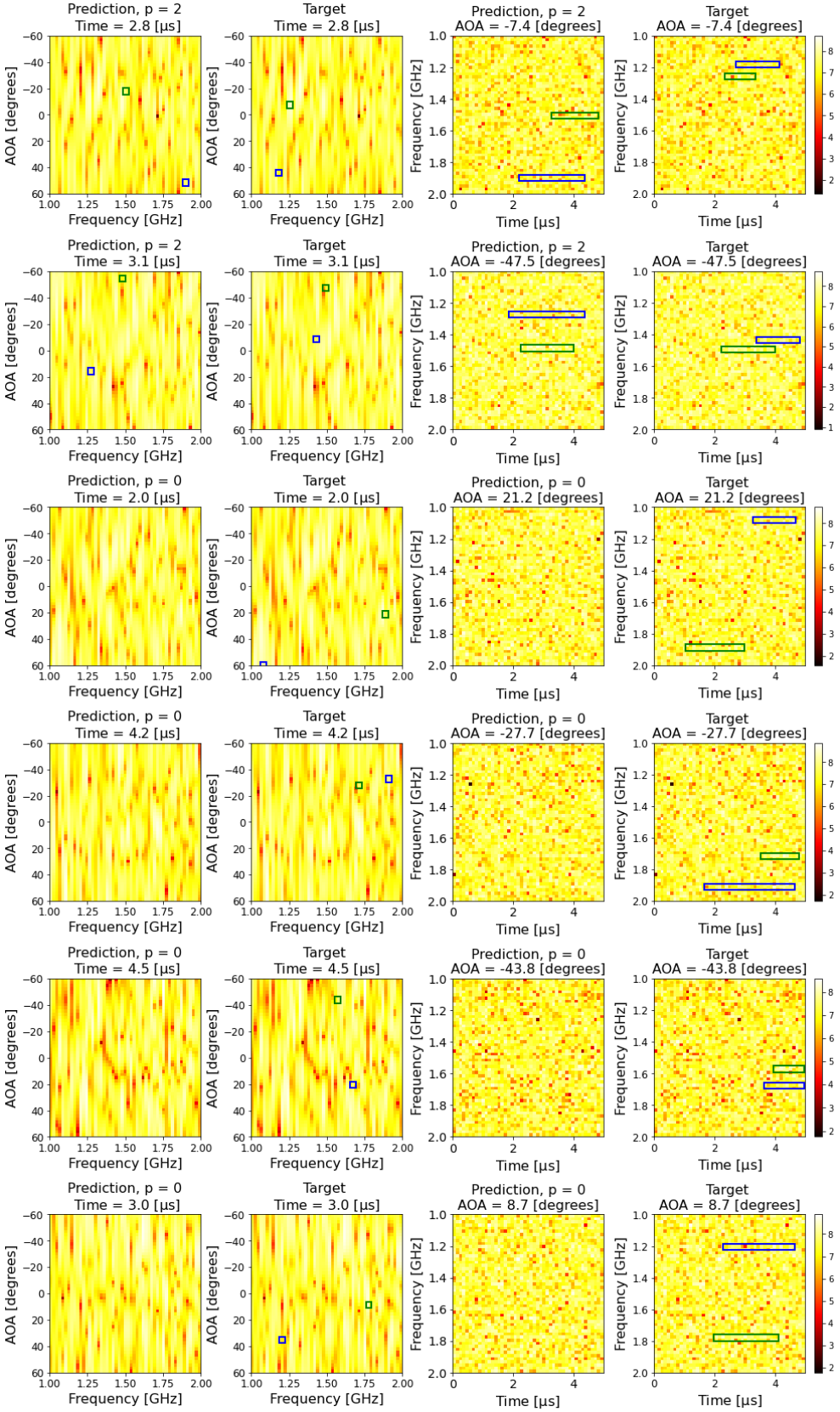


Figure A.6: Predictions and targets for test data with SNR=-50dB, p is the number of predictions made. One row shows predictions and true targets for one sample.

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY