UNIKASSEL
VERSITÄT

MATHEMATIK
UND
NATUR
WISSENSCHAFTEN

PHYSIK KASSEL

---

Experimentalphysik I:
Licht-Materie-Wechselwirkung

# Master Thesis

**to obtain the academic degree of Master of Science in Physics**

# Machine learning improved search for nitrogen-vacancy color centers with long coherence times

| | |
|---|---|
| Author: | Ricky-Joe Plate (uk037269@student.uni-kassel.de)<br>Gräfestraße 11 in 34121 Kassel |
| Matriculation number: | 333 005 86 |
| First examiner: | Prof. Dr. Kilian Singer |
| Second examiner: | Prof. Dr. Martin E. Garcia |
| Location: | Kassel |
| Date of submission: | 24.01.2024 |

## Declaration of Authenticity

I hereby confirm that I wrote this work independently and did not use any tools other than those specified. The parts of the work whose wording or meaning were taken from other works (this also includes internet sources) have been identified by stating the source.

Ort, Datum: ______________________________

Unterschrift: ______________________________

## Zusammenfassung

Unreinheiten in einem Diamanten bestehen aus Defekten, wie beispielsweise das Fehlen von einem Kohlenstoffatom oder das Ersetzen eines Kohlestoffatoms durch ein anderes Atom. Emittieren diese Defekte Wellenlängen im sichtbaren Spektrum, werden diese als Farbzentren bezeichnet. Dies verleiht dem eigentlich farblosen Diamanten eine Farbe. Die in dieser Masterarbeit betrachteten Defekte bestehen aus einer Kombination aus einem Stickstoffatom und einer Fehlstelle. Diese sogenannten NV-Zentren verleihen dem Diamanten eine leicht gelbe Färbung. Mittels *Konfokalmikroskopie* und *Optically detected magnetic resonance* (kurz ODMR)-Spektroskopie können die NV-Zentren im Diamanten gefunden und klassifiziert werden. Diese repetitiven Messmethoden sind jedoch sehr zeitintensiv für den Experimentator. Dazu muss, um möglichst genaue Messdaten zu erhalten, der Messprozess häufig wiederholt werden. Zudem erwärmt sich die Messumgebung durch Einstrahlen von Mikrowellen, sodass das NV-Zentrum nicht mehr im Fokus des Konfokalmikroskops sein könnte. In diesem Fall muss der Experimentator nach einer bestimmten Zeit das Mikroskop nachjustieren.
Diese Masterarbeit befasst sich damit, dass die Aufgabe der Klassifikation von NV-Zentren automatisiert wird, sowie der Neufokussiereung. Dabei wird eine künstliche Intelligenz, basierend auf einem Neuralen Netzwerk entwickelt, ihr beigebracht, was NV-Zentren sind und wie sie zu klassifizieren sind. Dies erlaubt dann konstante Messungen von beispielsweise Pulssequenzen, welche für gute Resultate über Tage hinweg laufen müssen.

## Abstract

Inconsistencies in a diamond consist of defects, such as for example the missing of a carbon atom or the replacement of a carbon atom by another atom. When these inconsistencies emit wavelengths in the visible spectrum, they are called color centers. This leads to a colored diamond which is otherwise colorless. The defects considered in this masters thesis consist of a combination of a nitrogen atom with a vacancy. The so called NV-centers give the diamond a slight yellowish tint. Using *confocal microscopy* and *optically detected magnetic resonance* (ODMR)-spectroscopy, NV-centers in the diamond can be identified and classified, but these repetitive measurement methods are very time consuming for the experimenter. In addition to get highly accurate measurement data, the measurement process often needs to be repeated. Furthermore, the measurement environment heats up due to the microwaves of the ODMR-spectroscopy which leads to the NV-center no longer be in the focus of the confocal microscope. In such a

case the experimenter would need to readjust the microscope.
This masters thesis deals with automatisation of the task of refocusing and classifying NV-centers. An artificial intelligence based on a neural network is developed and trained to understand what NV-centers are and how to classify them. This allows measurements like pulse sequences which need to run over days to achieve good results.

# Contents

**1 Introduction** **6**

**2 Neural Networks** **7**
- 2.1 Artificial Neurons 7
- 2.2 Artificial Neural Network 9
  - 2.2.1 Backpropagation 9
  - 2.2.2 Gradient Descent Variants 16
  - 2.2.3 Overfitting vs. Underfitting 17
- 2.3 Convolutional Neural Network 19
  - 2.3.1 Feature Extraction 19
  - 2.3.2 Filter Backpropagation 24
- 2.4 Position Detection 26
- 2.5 Datasets 29
- 2.6 Confusion Matrix 30

**3 Color Centers** **31**
- 3.1 Color Centers in Diamonds 31
  - 3.1.1 NV-center 31
- 3.2 ODMR-Spectroscopy 32
  - 3.2.1 Electron Spin 32
  - 3.2.2 Nuclear Spin 33
- 3.3 External Magnetic Field 33
- 3.4 Rabi-Oscillation 35
- 3.5 Bloch Sphere 36
  - 3.5.1 Pulsed ODMR 37
  - 3.5.2 Composite Pulses 38
- 3.6 Longitudinal Relaxation Time $T_1$ 39

**4 Experiment** **41**
- 4.1 Measurement Routine 42

**5 Object detection** **44**
- 5.1 Center Determination 44
- 5.2 Neural Networks 45

**6 Scan of ROI Anomaly** **46**
- 6.1 Network Architecture 46
- 6.2 Preprocessing 47
  - 6.2.1 YOLO Algorithm 48
- 6.3 Backbone 49
  - 6.3.1 Artificial Neural Network 49
  - 6.3.2 Convolutional Neural Network 53
  - 6.3.3 Comparison of the Neural Networks 61
- 6.4 Postprocessing 62
  - 6.4.1 F1-Score 64
- 6.5 Final Architecture 65

**7 Data Analysis** **70**
- 7.1 Backpropagation Fitting 70
- 7.2 Weighted Backpropagation Fitting 71

**8 SoRA** **74**
8.1 Prediction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 74
8.2 Training . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
8.2.1 Training dataset . . . . . . . . . . . . . . . . . . . . . . . . . 76
8.3 Routines . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
8.3.1 Autofocus . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
8.3.2 ODMR Measurment . . . . . . . . . . . . . . . . . . . . . . . 79
8.4 Specific NV-Center Search . . . . . . . . . . . . . . . . . . . . . . . 80
8.5 $T_1$ Measurement . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80

**9 Summary** **82**

**10 Outlook** **83**
10.1 Object Recognition . . . . . . . . . . . . . . . . . . . . . . . . . . . 83
10.2 Data Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 83
10.3 Measurement Routines . . . . . . . . . . . . . . . . . . . . . . . . . 83

# List of Figures

1 Example data for an artificial neuron . . . . . . . . . . . . . . . . . . . 7
2 Artificial neuron . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
3 Artificial neural network . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
4 Possible paths during backpropagation . . . . . . . . . . . . . . . . . . . 11
5 Influence of the learning rate on backpropagation . . . . . . . . . . . 11
6 Example neural network . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
7 Backpropagation hidden layer 2 - output layer . . . . . . . . . . . . . 13
8 Backpropagation hidden layer 1 - hidden layer 2 . . . . . . . . . . . 14
9 Backpropagation input layer - hidden layer 1 . . . . . . . . . . . . . 15
10 Gradient decent variants for the learning process . . . . . . . . . . . 17
11 Overfit and underfit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
12 Sweet spot in the learning process . . . . . . . . . . . . . . . . . . . . 18
13 Applied edge filter to the image of Eileen Collins . . . . . . . . . . . 20
14 Feature extraction using a digit as example . . . . . . . . . . . . . . 20
15 Applying a filter to an example image . . . . . . . . . . . . . . . . . . 21
16 Applied filter to the image of the digit nine . . . . . . . . . . . . . . 22
17 Applied ReLU and pooling on an example . . . . . . . . . . . . . . . 23
18 Object detection using a convolutional neural network for vehicles . 24
19 Backpropagation for filters . . . . . . . . . . . . . . . . . . . . . . . . . 25
20 Process of object detection using the method of R CNN . . . . . . . 27
21 Process of object detection using the method of YOLO . . . . . . . . 29
22 Crystal structure in a diamond with a defect . . . . . . . . . . . . . 31
23 Energy level diagram of the NV-centers . . . . . . . . . . . . . . . . 32
24 ODMR spectrum of the electron spin . . . . . . . . . . . . . . . . . . 33
25 Effect of an external magnetic field on the electron and nuclear spin in an ODMR spectrum . . . . . . . . . . . . . . . . . . . . . . . . . 34
26 Rabi-oscillation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
27 Bloch sphere . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
28 ODMR spectrum of pulsed ODMR . . . . . . . . . . . . . . . . . . . . 37
29 Effects of composite pulses . . . . . . . . . . . . . . . . . . . . . . . . 38
30 Measurement of the longitudial relaxation time $T_1$ . . . . . . . . . . 40
31 Experiment setup . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41
32 Measurement routine . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
33 Center determination . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
34 Network architecture . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
35 Resolution upscaling . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
36 Linked YOLO algorithms . . . . . . . . . . . . . . . . . . . . . . . . . 48
37 ANN training process . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
38 Effects of the learning method and the bias on the predictions of the ANN . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
39 Effects of repetitions on the predictions of the ANN (SGD) . . . . . 52
40 Predictions of the ANN for 6 scans . . . . . . . . . . . . . . . . . . . 53
41 Initialized parameters for the filters of the CNN. . . . . . . . . . . . 54
42 CNN training process . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
43 Effects of the learning method and the bias on the predictions of the CNN . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56
44 Effects of repetitions on the predictions of the CNN (SGD) . . . . . 57
45 Predictions of the CNN for 6 scans . . . . . . . . . . . . . . . . . . . 58
46 Calculated filters for MBGD and bias . . . . . . . . . . . . . . . . . . 59
47 Calculated filters for SGD and bias . . . . . . . . . . . . . . . . . . . 59
48 Calculated filters for MBGD and without bias . . . . . . . . . . . . . 60
49 Calculated filters for SGD and without bias . . . . . . . . . . . . . . 60

50 Architecture of the postprocessor . . . . . . . . . . . . . . . . . . . . . 63
51 F1 Score development of the postprocessor . . . . . . . . . . . . . . . 65
52 Final architecture for detecting NV-centers . . . . . . . . . . . . . . 67
53 Comparison between data analysis algorithms . . . . . . . . . . . . . 71
54 SoRA prediction window . . . . . . . . . . . . . . . . . . . . . . . . . . 74
55 SoRA training window . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
56 SoRA traingdataset window . . . . . . . . . . . . . . . . . . . . . . . . 77
57 Results of the XY autofocus method . . . . . . . . . . . . . . . . . . . 78
58 Results of the 3D autofocus method . . . . . . . . . . . . . . . . . . . 78
59 SoRA data analysis window . . . . . . . . . . . . . . . . . . . . . . . . 79
60 ODMR measurement for an NV-center with $^{13}$C . . . . . . . . . . . 80
61 $T_1$ measurement with and without refocus . . . . . . . . . . . . . . . 81

# 1 Introduction

Over centuries new technologies or discoveries have led to rapid advancements in human civilization. These include simple discoveries such as fire in the Stone Age to the use of steam engines during the Industrial Revolution. Each of these technologies influenced and simplified human life. While the interval between new discoveries was at the beginning quite long, the gap exponentially shortened at a rate of approximately 5% over time[1]. The 20th century was marked by a lot of new technologies such as airplanes, computers and the internet, which all had a significant impact on the progress and science. The fast sharing of knowledge over the internet coupled with the process of globalization provides the foundation for collaborative research across national borders.

The most significant advancements at the beginning of the 21st century are taking place in the field of quantum technologies and artificial intelligence. In quantum mechanics, areas such as *quantum metrology, quantum networks, quantum teleportation, quantum sensing* and *quantum computing* are important application examples. On the other hand, artificial intelligence focuses on efficient training methods. While these application examples were largely independent of each other at the beginning, they now benefit from each other. On the one hand, quantum mechanical effects can make neural networks more energy efficient and faster by using photons for computation instead of electronics, where photons have the additional advantage of representing complex numbers through their magnitude and phase[2], which could make neural networks for example smaller without loose of complexity. On the other hand, physically extremely time consuming simulations can be accelerated by neural networks[3][4].

Additionally, a trained artificial intelligence can also handle repetitive tasks in an experiment. In the case of NV-centers in a diamond, they need to be classified before doing experiments with them. Classification involves various parameters such as position, cw ODMR, axis orientation, resonant Rabi-frequency and more where some of them require multiple measurements. For example, some of these measurements are necessary to determine the optimal alignment between an external magnetic field and the NV axis or to find the best resonant Rabi-frequency. Another problem in this context is the long measurement times. When microwaves are applied, the system heats up, which might cause it to no longer be focused on the NV-center. Manual refocusing every half an hour over several days is an intensive task for the experimenter. This masters thesis primarily focuses on such artificial intelligence and its construction as well as the most efficient network architecture. This architecture then allows measurements over several days without the need for the experimenters intervention.

# Fundamentals

To understand the goal of this master thesis, some fundamentals need to be explained first. Since this thesis deals with both, neural networks and color centers in a diamond, the fundamentals primarily focus on these two topics. First, it will focus on neural networks before explaining color centers. In case of the neural networks, two types will be introduced along with specific mechanisms on which the principles of neural networks are based. In the case of color centers, the fundamentals will focus mainly on a nitrogen vacancy (NV)-center combination. In this chapter, some of the physical properties as well as measurement methods will be presented.

# 2 Neural Networks

This chapter focuses on neural networks. It explains two types of neural networks and their training processes. Here should be noted that there are other types of neural networks, but they are not relevant to this thesis. First, the chaper will start with a simple artificial neural network before expanding it into a convolutional neural network in the second part. Futhermore, algorithms will be introduced to use the network for solving object detection problems and to evaluate the results in the end.

## 2.1 Artificial Neurons

In its simplest form, an artificial neural network consists of individual artificial neurons connected to each other through weights[5]. Therefore, to understand how a neural network works, it is necessary to explain what a neuron is. Figure 1 shows an example using fictitious data that describes the status of whether a student has achieved their degree (1) or not (0) after a certain number of months.

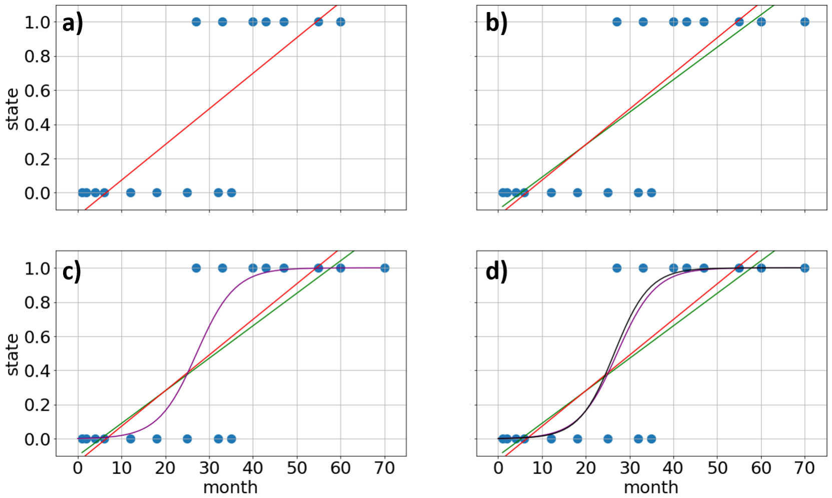


**Figure 1:** The diagrams shows fictitious measurement data fitted with linear and nonlinear functions. In a) the diagram shows a red linear curve that has been adjusted to all the measurement data. In b) an additional data point at the edge has been included but only considered in the regression of the green function. The lower diagrams shows a nonlinear function in the form of a sigmoid function which uses the linear functions as input value for the red curve c) and both curves d).

The diagrams in Figure 1 shows fictitious data. In a) a linear function of the form $y(x) = mx + b$ has been fitted to this data taking all the points into account. In b) another data point at $x = 70$ is added. When this value is considered in the regression, it changes the fit parameters of the linear function (green curve). The predictions of the red and green curves differ significantly in some regions due to the added data point. To solve this problem, a nonlinear function is used. In this example a sigmoid function of the form $sig(y(x)) = \frac{1}{1+\exp(-y)}$ was chosen. Figure 1 c) shows the sigmoid function (purple), which uses the fit parameters of the red function. In d) an additional sigmoid function is shown in black, which uses the fit parameters of the green curve. The diagram shows that the difference in the case of an additional data point is minimal so that the predictions are not much changed. Another problem solved by the sigmoid function is that it has no negative values or values greater than 1. Therefore, the problem is solved through a combination of a linear and a nonlinear function.

A neuron is a combination of a linear function and a non-linear function[6]. It takes a value as input and produces the corresponding output. The non-linear function is called the *activation function* and does not always have to be a sigmoid function[5]. There are many different activation functions, such as the hyperbolic tangent (tanh) or the Rectified Linear Unit (ReLU)[7]. The ReLU function sets all values less than zero to zero and leaves all values greater than zero unchanged.

The biggest advantage of combining a linear with a non-linear function is that the dimension of the input can be increased without major problems. Although this is also possible with a simple non-linear function, where the dimension would increase with each additional input value, but this would make the fit process more and more computationally- and time-intensive. In the examples mentioned above additional parameters, such as a students ambition or financial pressure can be added as new dimension. Each additional dimension is then considered in the linear function which subsequently passes a single parameter to the non-linear function. Figure 2 shows such an artificial neuron with multiple input variables.

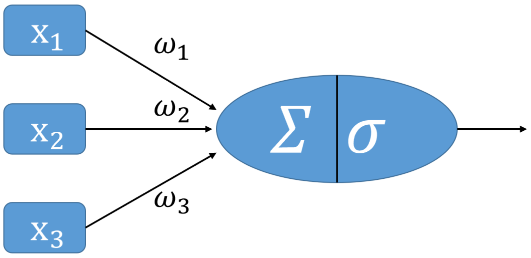


**Figure 2:** Structure of an artificial neuron with three different inputs and a sigmoid activation function. $\sum$ stands for the linear part and $\sigma$ for the non-linear part of the neuron.

In this case, the neuron has three independent input variables $x_1$ to $x_3$, which are multiplied by their respective factors $\omega_1$ to $\omega_3$. These factors $\omega_1$, $\omega_2$ and $\omega_3$ are called weights. The weights determine how important a particular input parameter is in the prediction calculation. During a training process these weights are adjusted so that the predictions align as closely as possible with the training data. The linear function for the artificial neuron shown in Figure 2 can be written as follows:

$$f(x_1, x_2, x_3) = \omega_1\, x_1 + \omega_2\, x_2 + \omega_3\, x_3 + b. \tag{1}$$

The last parameter $b$ is the bias, which is optional[5].

## 2.2 Artificial Neural Network

The so called Artificial Neural Network (ANN) represents the simplest way to create a neural network. It operates without prior processing of input data and consists of interconnected neurons. The ANN consists of an input layer, one or more hidden layers and an output layer[6]. The input layer is formed by individual measurements, which could be for example an image or data that needs to be analyzed for patterns. The hidden layers can contain any number of neurons and the number of hidden layers in general can also be set freely by the creator. It should be noted that the larger the number of neurons in a hidden layer and the more hidden layers are added, the longer the computation time for a training process. That is why for each problem an individual neural network is often created and fine-tuned doing experimentation. The final layer is the output layer, which is responsible for providing results to the user. Its configuration depends on the specific problem and can deliver various amounts of information. The following figure shows such a simple artificial neural network.

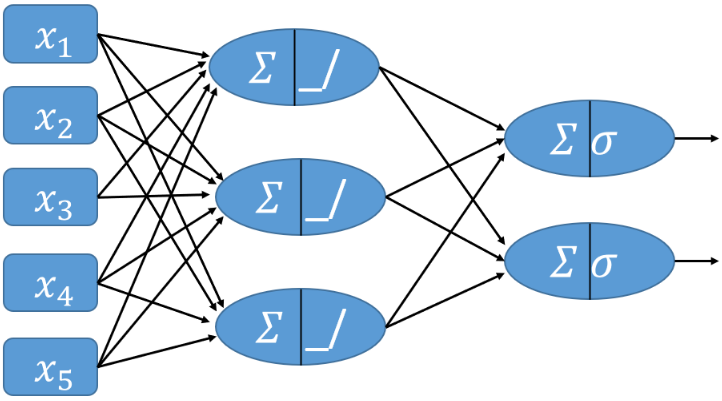


**Figure 3:** Structure of a simple ANN with five inputs, a hidden layer consisting of three neurons and an output layer with two different options. ReLU was used as the activation function in the hidden layer and a sigmoid function in the output layer.

### 2.2.1 Backpropagation

In Chapter 2.1 and Chapter 2.2 it was demonstrated how a neuron or ANN calculates and produces outputs. In order to make the most accurate predictions possible, the ANN must first be adjusted using training data. The weights as introduced in Chapter 2.1, are used for this. These weights along with the bias are adjusted during the training process to get a better prediction. This process is called backpropagation.

To train the neurons training data must be created first. To show this process, the problem described in Chapter 2.1 is used as an example. Additionally, more input data such as financial stress and student diligence have been added, resulting in the artificial neuron taking the form shown in Figure 2. The data used for this purpose is presented in the following table.

To adjust the weights during the training process, it is necessary to determine the accuracy of the neurons predictions. To do this, the error needs to be calculated. Generally, the mean squared error or the logarithmic error is used for this but similar to the activation function there are many other functions that can be chosen

**Table 1:** Example data for a training process.

| time [month] | effort $[\frac{\text{houres}}{\text{day}}]$ | money [+/-] | result |
|---|---|---|---|
| 1 | 7 | 1.3 | 0 |
| 12 | 4 | 2.1 | 0 |
| 27 | 9 | -0.4 | 1 |
| 32 | 8 | 0 | 0 |
| 47 | 5 | 0.9 | 1 |

individually[7]. In this example, the mean squared error (MSE) is used[5]:

$$\text{MSE} = (z - \hat{z})^2. \tag{2}$$

$z$ stands for the value of the sigma activation function predicted by the neuron and $\hat{z}$ for the actual value which is known from the training data (result column in table 1).

In order to change the weights and with it the accuracy of the predictions, the dependecy between the error and the weights must be determined[5]. Since the error does not depend directly on the weights but only indirectly, the equation is expressed as follows:

$$\frac{\partial \text{Error}}{\partial \omega_1} = \frac{\partial \Sigma}{\partial \omega_1} \frac{\partial \sigma}{\partial \Sigma} \frac{\partial \text{Error}}{\partial \sigma}. \tag{3}$$

This is known as the chain rule[7]. Insert the equations for $\Sigma$, $\sigma$ and the error leads to following equation for the neuron shown in Figure 2:

$$\frac{\partial \text{Error}}{\partial \omega_1} = \frac{\partial}{\partial \omega_1} \left(\omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + b\right) \frac{\partial}{\partial \Sigma} \frac{1}{1 + \exp^{-\Sigma}} \frac{\partial}{\partial \sigma} \left(\sigma - \hat{z}\right)^2 \tag{4}$$

$$= x_1 \frac{1}{1 + \exp^{-\Sigma}} \left(1 - \frac{1}{1 + \exp^{-\Sigma}}\right) 2 \left(\sigma - \hat{z}\right). \tag{5}$$

Equation 5 then describes how a change in the weight $\omega_1$ can affect the error. If the value of the first weight is 0.17, the second weight is 0.32, the third weight is -0.71 and the neurons prediction is 0.34, the deviation can be calculated using the linear function

$$\Sigma = 0.17 \cdot 12 + 0.32 \cdot 4 - 0.71 \cdot 2.1 = 1.829$$

$$\frac{\partial \text{Error}}{\partial \omega_1} = 12 \cdot \frac{1}{1 + \exp^{-1.829}} \left(1 - \frac{1}{1 + \exp^{-1.829}}\right) 2 \cdot (0.34 - 0) \approx 0.9728.$$

To finish the training process, the dependencies of the other weights and if used the bias $b$ must also be determined in a similar way. From there, the new weights are calculated. For $\omega_1$ this process looks as follows:

$$\omega_1 = \omega_1 - lr \, \frac{\partial \text{Error}}{\partial \omega_1}. \tag{6}$$

Or for the example with $lr = 0.01$:

$$\omega_1 = 0.17 - 0.01 \cdot 0.9728 \approx 0.1603.$$

'$lr$' stands for the learning rate.

Once all training data has been processed, in this example there are five, the process is repeated. One iteration is referred to as an *epoch*[5]. After each epoch, the

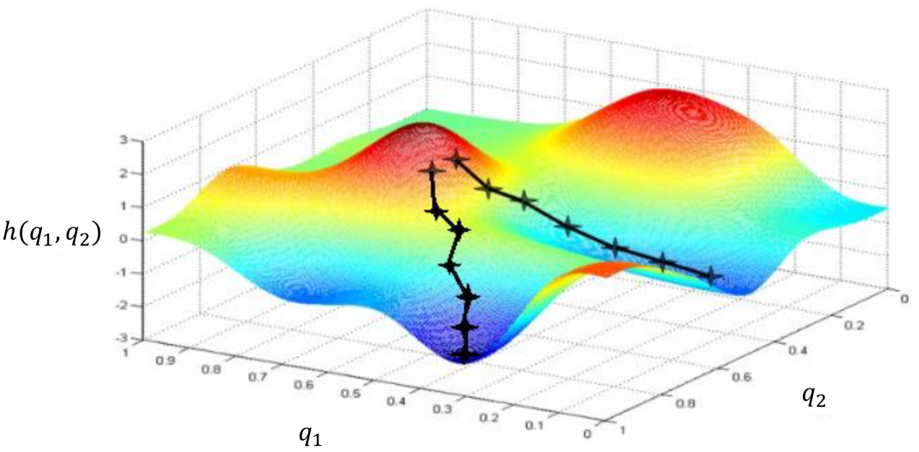


**Figure 4:** Two paths that lead to different minima with slightly different initial conditions.[10]

weights are adjusted slightly until the minimum is reached, which can be seen in Figure 4.
The figure also shows that slightly different adjustments to the initial parameters could lead to different outputs. In Figure 4 two paths are shown leading to different minima. Therefore, choosing different starting parameters can result in different accuracy of predictions made by the network[10][12].

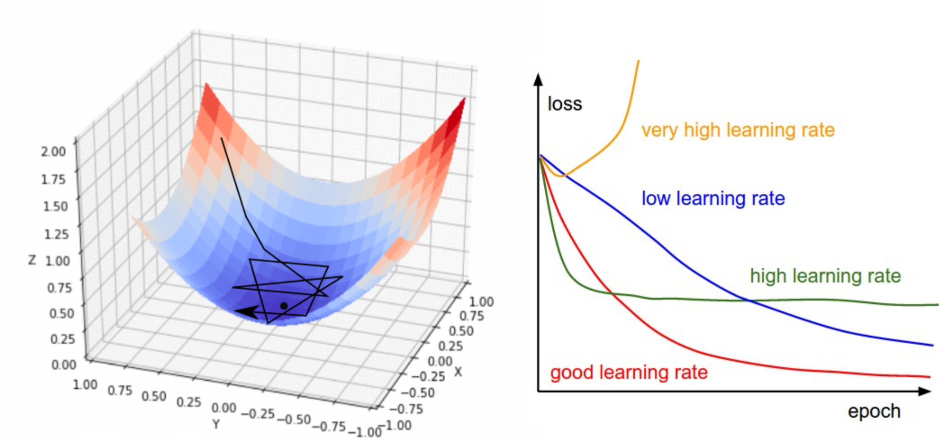


**Figure 5:** Example of not reaching the minimum because the learning rate was chosen too high is shown on the left side. The minimum error will never be achieved.[9] Different learning rates and their influence on the loss are shown on the right side.[8]

To solve this problem one possible solution could be to increase the learning rate. However, it is important to note that if the learning rate is chosen to be too high, the minimum may never be reached (Figure 5 on the left). A solution to the problem can be to use a learning rate that changes over time.
Figure 5 on the right side shows the change in loss plotted against the epochs at different learning rates. The green curve demonstrates the problem seen in Figure 5 (left) where the minimum is never reached. The blue curve additionally shows that with a very small learning rate, it takes a long time to optimize the weights which

could require a significant amount of computation. The learning rate of the red curve is between the other two learning rates but it must be manually determined and it is also specific for each neural network.

While determining the new weights of individual neurons is relatively straightforward, it becomes more complex in neural networks with multiple hidden layers. Matrices are often used to provide a better overview. The principle of the training process remains identical. The chain rule for the derivatives just becomes more extended. In this part, the backpropagation of a neural network is demonstrated based on the following Figure 6. The matrices are inspired by the source [13].

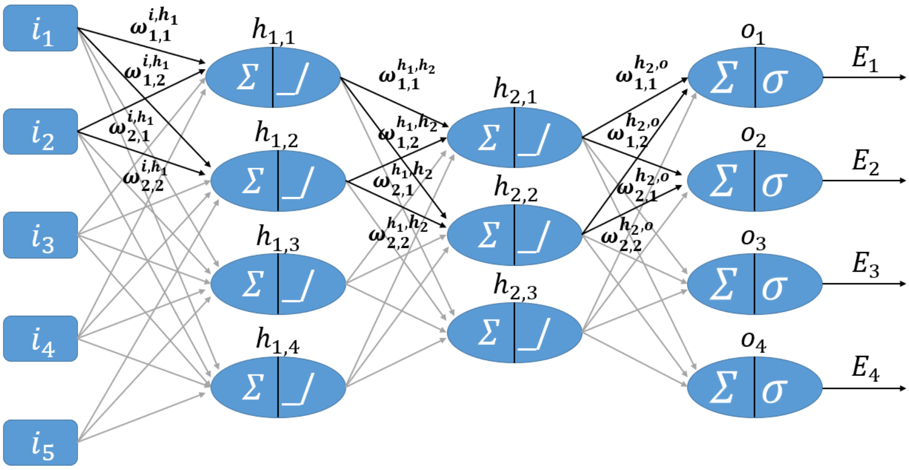


**Figure 6:** Example of a simple artificial neural network. The network consists of two hidden layers with different numbers of neurons and ReLU as an activation function. For the output, a Sigmoid function is used as the activation function. (Figure based on [13])

**Hidden Layer 2 - Output Layer:** The calculation of the weights between the second hidden layer $h_2$ and the output layer $o$ is done similar to the previous chapter, using the chain rule. The only difference here is that there is more than one output neuron. Each of these output neurons has its own individual error. The total error and accuracy of the neural network is determined by the sum of these individual errors[7][12].

$$\text{Error} = \sum_{i=1}^{4} E_i = E_1 + E_2 + E_3 + E_4. \tag{7}$$

Each $E_i$ then represents an error function according to equation 2.
Figure 7 shows the first and second neurons in the output layer from Figure 6. The impact of the weights can be determined using the chain rule[11]. For $\omega_{1,1}^{h_2,o}$ it appears as follows according to Equation 3:

$$\frac{\partial E_1}{\partial \omega_{1,1}^{h_2,o}} = \frac{\partial o_{\text{in},1}}{\partial \omega_{1,1}^{h_2,o}} \frac{\partial o_{\text{out},1}}{\partial o_{\text{in},1}} \frac{\partial \mathrm{E}_1}{\partial o_{\text{out},1}} \tag{8}$$

The same applies to the other derivatives with respect to the weights. In general for $i$ neurons in the previous layer and $j$ neurons in the output layer, the matrix

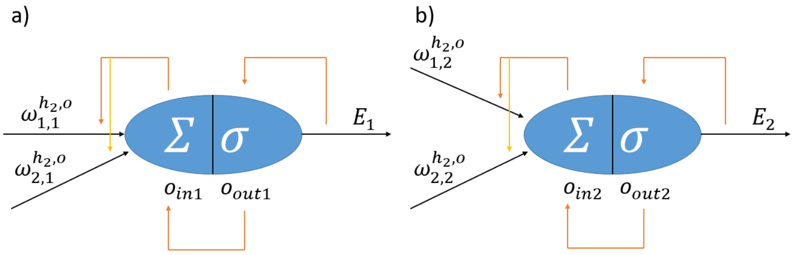


**Figure 7:** First and second neuron in the output layer. The orange and yellow arrows stand for backpropagation and the black arrows for the forward propagation. (Figure based on [13])

has the following form:

$$\partial\omega_{i,j} = \begin{pmatrix} \frac{\partial E_1}{\partial\omega_{1,1}^{h_2,o}} & \frac{\partial E_2}{\partial\omega_{1,2}^{h_2,o}} & \cdots & \frac{\partial E_j}{\partial\omega_{1,j}^{h_2,o}} \\ \frac{\partial E_1}{\partial\omega_{2,1}^{h_2,o}} & \frac{\partial E_2}{\partial\omega_{2,2}^{h_2,o}} & \cdots & \frac{\partial E_j}{\partial\omega_{2,j}^{h_2,o}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial E_1}{\partial\omega_{i,1}^{h_2,o}} & \frac{\partial E_2}{\partial\omega_{i,2}^{h_2,o}} & \cdots & \frac{\partial E_j}{\partial\omega_{i,j}^{h_2,o}} \end{pmatrix} \tag{9}$$

The new weights from the example network can then be determined as follows[12]:

$$\hat{\omega}^{h_2,o} =$$

$$\begin{pmatrix} \omega_{1,1}^{h_2,o} - (lr\partial\omega_{1,1}^{h_2,o}) & \omega_{1,2}^{h_2,o} - (lr\partial\omega_{1,2}^{h_2,o}) & \omega_{1,3}^{h_2,o} - (lr\partial\omega_{1,3}^{h_2,o}) & \omega_{1,4}^{h_2,o} - (lr\partial\omega_{1,4}^{h_2,o}) \\ \omega_{2,1}^{h_2,o} - (lr\partial\omega_{2,1}^{h_2,o}) & \omega_{2,2}^{h_2,o} - (lr\partial\omega_{2,2}^{h_2,o}) & \omega_{2,3}^{h_2,o} - (lr\partial\omega_{2,3}^{h_2,o}) & \omega_{2,4}^{h_2,o} - (lr\partial\omega_{2,4}^{h_2,o}) \\ \omega_{3,1}^{h_2,o} - (lr\partial\omega_{3,1}^{h_2,o}) & \omega_{3,2}^{h_2,o} - (lr\partial\omega_{3,2}^{h_2,o}) & \omega_{3,3}^{h_2,o} - (lr\partial\omega_{3,3}^{h_2,o}) & \omega_{3,4}^{h_2,o} - (lr\partial\omega_{3,4}^{h_2,o}) \end{pmatrix} \tag{10}$$

**Hidden Layer 1 - Hidden Layer 2:** After determining the new weights between the second hidden layer and the output layer, the next step is to determine the new weights between the first hidden layer $h_1$ and the second hidden layer $h_2$. The calculations for the new weights also rely on the chain rule. The biggest difference in the determination of the weights compared to the previous step is that there are no explicit errors present.

The figure shows the dependencies that need to be considered when determining the new weights. It can be seen that the error depends on each output error $E_1$ to $E_4$[12]. The dependencies can be determined using the chain rule as shown in the following equations[11].

For the dependence on $E_1$ the following applies:

$$\frac{\partial E_1}{\partial\omega_{1,1}^{h_1,h_2}} = \frac{\partial h_{2\text{in},1}}{\partial\omega_{1,1}^{h_1,h_2}} \frac{\partial h_{2\text{out},1}}{\partial h_{2\text{in},1}} \frac{\partial o_{\text{in},1}}{\partial h_{2\text{out},1}} \frac{\partial o_{\text{out},1}}{\partial o_{\text{in},1}} \frac{\partial \text{E}_1}{\partial o_{\text{out},1}}, \tag{11}$$

This also applies to the other error derivatives with respect to their weights. The total error $E^{h_2}$ is then the sum of the individual errors so that this can be written

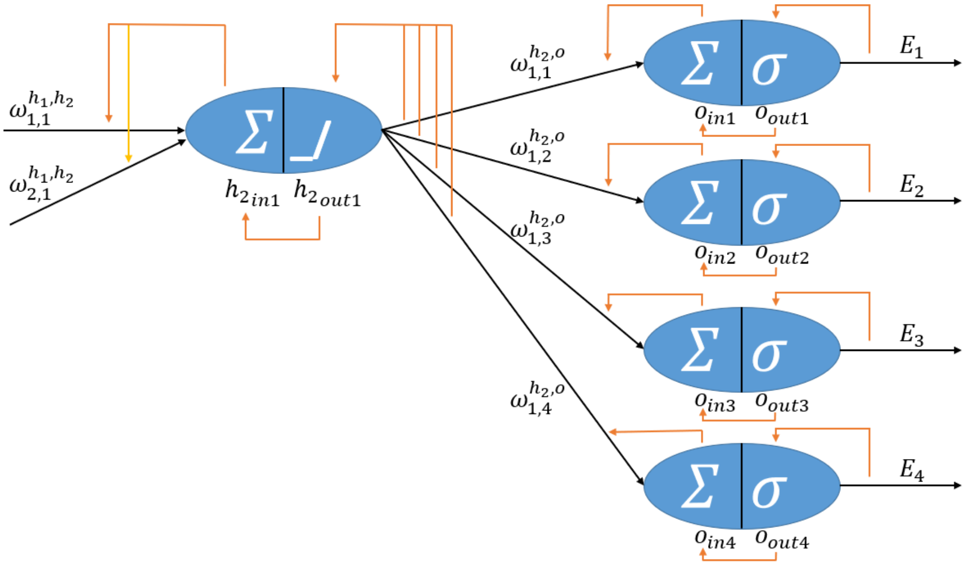


**Figure 8:** The first neuron in hidden layer 2. The orange and yellow arrows stand for the backpropagation and the black arrows for the forward propagation. (Figure based on [13]).

as:

$$\frac{\partial E^{h_2}}{\partial \omega_{1,1}^{h_1,h_2}} = \frac{\partial E_1}{\partial \omega_{1,1}^{h_1,h_2}} + \frac{\partial E_2}{\partial \omega_{1,1}^{h_1,h_2}} + \frac{\partial E_3}{\partial \omega_{1,1}^{h_1,h_2}} + \frac{\partial E_4}{\partial \omega_{1,1}^{h_1,h_2}} \tag{12}$$

$$= \frac{\partial h_{2\text{in},1}}{\partial \omega_{1,1}^{h_1,h_2}} \frac{\partial h_{2\text{out},1}}{\partial h_{2\text{in},1}} \frac{\partial E^{h_2}}{\partial h_{2\text{out},1}}. \tag{13}$$

Again, in general for $i$ neurons in the previous layer and $j$ neurons in the hidden layer, the matrix has the following form:

$$\partial \omega_{i,j} = \begin{pmatrix} \frac{\partial E^{h_2}}{\partial \omega_{1,1}^{h_1,h_2}} & \frac{\partial E^{h_2}}{\partial \omega_{1,2}^{h_1,h_2}} & \cdots & \frac{\partial E^{h_2}}{\partial \omega_{1,j}^{h_1,h_2}} \\ \frac{\partial E^{h_2}}{\partial \omega_{2,1}^{h_1,h_2}} & \frac{\partial E^{h_2}}{\partial \omega_{2,2}^{h_1,h_2}} & \cdots & \frac{\partial E^{h_2}}{\partial \omega_{2,j}^{h_1,h_2}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial E^{h_2}}{\partial \omega_{i,1}^{h_1,h_2}} & \frac{\partial E^{h_2}}{\partial \omega_{i,2}^{h_1,h_2}} & \cdots & \frac{\partial E^{h_2}}{\partial \omega_{i,j}^{h_1,h_2}} \end{pmatrix} \tag{14}$$

The new weights from the example network can then be determined as follows:

$$\hat{\omega}^{h_1,h_2} = \begin{pmatrix} \omega_{1,1}^{h_1,h_2} - (lr\partial\omega_{1,1}^{h_1,h_2}) & \omega_{1,2}^{h_1,h_2} - (lr\partial\omega_{1,2}^{h_1,h_2}) & \omega_{1,3}^{h_1,h_2} - (lr\partial\omega_{1,3}^{h_1,h_2}) \\ \omega_{2,1}^{h_1,h_2} - (lr\partial\omega_{2,1}^{h_1,h_2}) & \omega_{2,2}^{h_1,h_2} - (lr\partial\omega_{2,2}^{h_1,h_2}) & \omega_{2,3}^{h_1,h_2} - (lr\partial\omega_{2,3}^{h_1,h_2}) \\ \omega_{3,1}^{h_1,h_2} - (lr\partial\omega_{3,1}^{h_1,h_2}) & \omega_{3,2}^{h_1,h_2} - (lr\partial\omega_{3,2}^{h_1,h_2}) & \omega_{3,3}^{h_1,h_2} - (lr\partial\omega_{3,3}^{h_1,h_2}) \\ \omega_{4,1}^{h_1,h_2} - (lr\partial\omega_{4,1}^{h_1,h_2}) & \omega_{4,2}^{h_1,h_2} - (lr\partial\omega_{4,2}^{h_1,h_2}) & \omega_{4,3}^{h_1,h_2} - (lr\partial\omega_{4,3}^{h_1,h_2}) \end{pmatrix} \tag{15}$$

**Input Layer - Hidden Layer 1:** The final weights in the example network shown in Figure 6 are the weights between the input layer and the first hidden layer. Similar to determining the weights between the first and second hidden layers, the weights of this layer depend on all neurons of the previous layer. Figure 9 shows the dependencies that must be considered when determining the impact of the weights on the networks error. For example, the impact of the first error $E_1$ is no longer dependent on a single neuron but on all neurons in the second hidden layer since all neurons in the second hidden layer have an impact on the error $E_1$. In Figure 9 these influences are shown by the green arrows.

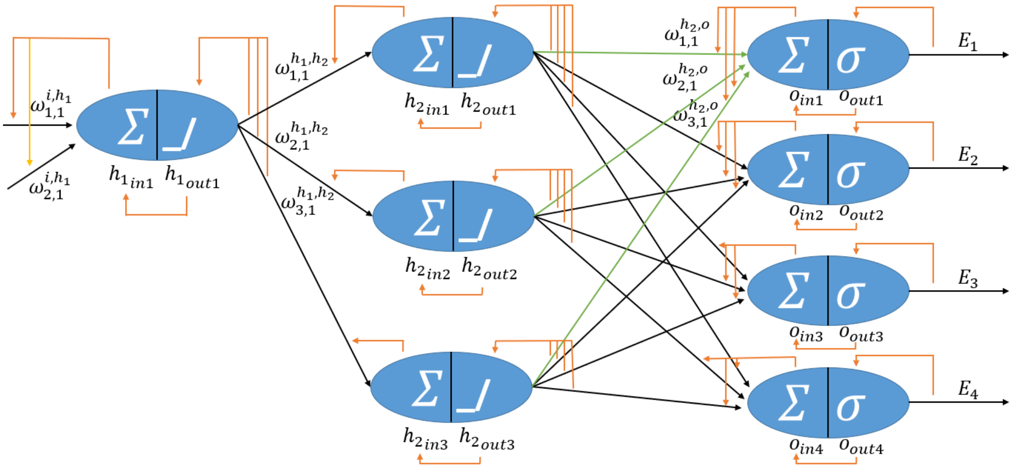


**Figure 9:** The first neuron in hidden layer 1. The orange and yellow arrows stand for the backpropagation, the green arrows stands for the dependencies on $E_1$ of the neurons and the black arrows for the forward propagation. (Figure based on [13]).

The influence of $\omega_{1,1}^{i,h_1}$ on $E_1$ is determined as follows[11]:

$$\frac{\partial E_1}{\partial \omega_{1,1}^{i,h_1}} = \frac{\partial h_{1\text{in},1}}{\partial \omega_{1,1}^{i,h_1}} \frac{\partial h_{1\text{out},1}}{\partial h_{1\text{in},1}} \left( x_1 \frac{\partial E_1}{\partial h_{2\text{out},1}} + x_2 \frac{\partial E_1}{\partial h_{2\text{out},2}} + x_3 \frac{\partial E_1}{\partial h_{2\text{out},3}} \right) \tag{16}$$

with

$$x_1 = \frac{\partial h_{2\text{in},1}}{\partial h_{1\text{out},1}} \frac{\partial h_{2\text{out},1}}{\partial h_{2\text{in},1}}, \qquad x_2 = \frac{\partial h_{2\text{in},2}}{\partial h_{1\text{out},1}} \frac{\partial h_{2\text{out},2}}{\partial h_{2\text{in},2}}, \qquad x_3 = \frac{\partial h_{2\text{in},3}}{\partial h_{1\text{out},1}} \frac{\partial h_{2\text{out},3}}{\partial h_{2\text{in},3}}.$$

This also applies to the other error derivatives with respect to the weights. The total error $E^{h_1}$ is then the sum of the individual errors so that this can be written as:

$$\frac{\partial E^{i,h_1}}{\partial \omega_{1,1}^{i,h_1}} = \frac{\partial E_1}{\partial \omega_{1,1}^{i,h_1}} + \frac{\partial E_2}{\partial \omega_{1,1}^{i,h_1}} + \frac{\partial E_3}{\partial \omega_{1,1}^{i,h_1}} + \frac{\partial E_4}{\partial \omega_{1,1}^{i,h_1}} \tag{17}$$

$$= \frac{\partial h_{1\text{in},1}}{\partial \omega_{1,1}^{i,h_1}} \frac{\partial h_{1\text{out},1}}{\partial h_{1\text{in},1}} \frac{\partial E^{h_2}}{\partial h_{1\text{out},1}}. \tag{18}$$

Again, in general for $i$ neurons in the previous layer and $j$ neurons in the hidden layer, the matrix has the following form:

$$\partial\omega_{i,j} = \begin{pmatrix} \frac{\partial E^{h_1}}{\partial\omega_{1,1}^{i,h_1}} & \frac{\partial E^{h_1}}{\partial\omega_{1,2}^{i,h_1}} & \cdots & \frac{\partial E^{h_1}}{\partial\omega_{1,j}^{i,h_1}} \\ \frac{\partial E^{h_1}}{\partial\omega_{2,1}^{i,h_1}} & \frac{\partial E^{h_1}}{\partial\omega_{2,2}^{i,h_1}} & \cdots & \frac{\partial E^{h_1}}{\partial\omega_{2,j}^{i,h_1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial E^{h_1}}{\partial\omega_{i,1}^{i,h_1}} & \frac{\partial E^{h_1}}{\partial\omega_{i,2}^{i,h_1}} & \cdots & \frac{\partial E^{h_1}}{\partial\omega_{i,j}^{i,h_1}} \end{pmatrix} \tag{19}$$

The new weights from the example network can then be determined as follows:

$$\hat{\omega}^{i,h_1} = \begin{pmatrix} \omega_{1,1}^{i,h_1} - (lr\partial\omega_{1,1}^{i,h_1}) & \omega_{1,2}^{i,h_1} - (lr\partial\omega_{1,2}^{i,h_1}) & \cdots & \omega_{1,4}^{i,h_1} - (lr\partial\omega_{1,4}^{i,h_1}) \\ \omega_{2,1}^{i,h_1} - (lr\partial\omega_{2,1}^{i,h_1}) & \omega_{1,2}^{i,h_1} - (lr\partial\omega_{2,2}^{i,h_1}) & \cdots & \omega_{2,4}^{i,h_1} - (lr\partial\omega_{2,4}^{i,h_1}) \\ \omega_{3,1}^{i,h_1} - (lr\partial\omega_{3,1}^{i,h_1}) & \omega_{1,2}^{i,h_1} - (lr\partial\omega_{3,2}^{i,h_1}) & \cdots & \omega_{3,4}^{i,h_1} - (lr\partial\omega_{3,4}^{i,h_1}) \end{pmatrix} \tag{20}$$

### 2.2.2 Gradient Descent Variants

To determine the new weights between the layers in the neural network, training data is used. This is data that has already been manually evaluated such as the data in Table 1. The neural network is then trained, using this data, so that it can evaluate unknown data. In addition to the training methods presented below, the most significant factor is the number of repetitions, which was referred to as *epochs*[5].

During the training process, it is essential to note that the neural network does not produce ideal values for the training data with too many epochs because in that case, while it may predict those well, the accuracy on test data may suffer[5]. This problem is called *overfitting*. The opposite scenario occurs when the training process is not successful, which can happen if for example the number of epochs is chosen to be too low. This is called *underfitting*. Chapter 2.2.3 goes further into these scenarios.

To train a neural network, there are mainly three different methods to deal with the training data[7]. The first method is known as *Batch Gradient Descent* (BGD), the second as *Stochastic Gradient Descent* (SGD) and the last as *Mini-Batch Gradient Descent* (MBGD). Figure 10 shows the training process as a function of the number of epochs and loss. A exponential function is used to create artificial data as training data with the values slightly changed by adding and subtracting random values to create noise.

The following will explain the trainings methods used in the diagrams.

**Batch Gradient Descent:** Batch Gradient Descent is the simplest way to train a neural network. In this method, training data is processed one by one[7]. This process is repeated for a certain number of epochs and after each iteration the weights are updated. The result can be seen in Figure 10 in the top left. The advantage of this method is the precise adjustment of the neural network to all training data. But it is not commonly used because it can be quite time consuming and computationally intensive, especially for a large number of training data.

**Stochastic Gradient Descent:** The second method is a stochastic selection of training samples. Instead of using all the data from the training dataset, only randomly selected training data samples are used[5]. The randomly selected training sample is then used to determine the new weights. After this, another random training sample is randomly selected and the process is repeated. The number of repetitions corresponds to the number of epochs. As shown in Figure 10 in the upper right, it can be seen that the training process is not ideal due to the presence of many outliers. But nevertheless this allows to train a network with a large number of training samples.

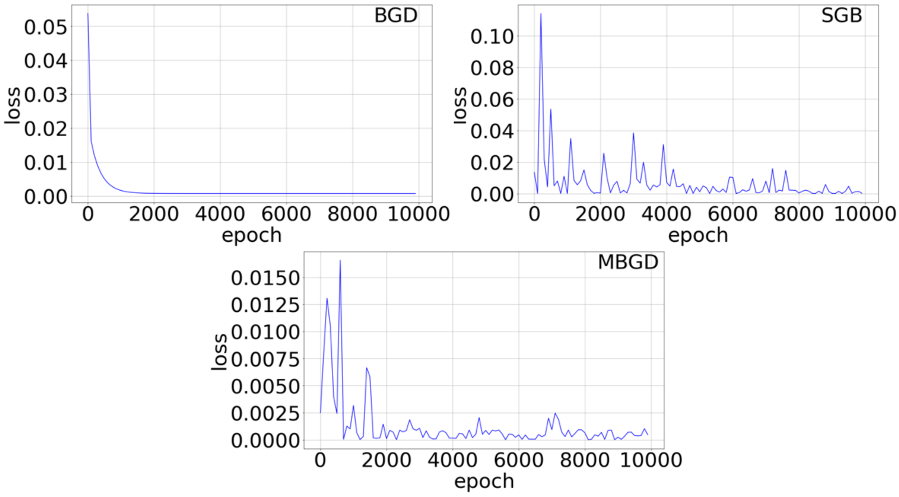


**Figure 10:** The training process of the different training methods is shown in the graph. For the demonstration an exponential function of the form $y(x) = \exp(0.01x)$ with random noise was used as artificial data. A Python program was used to implement the different training methods.

**Mini Batch Gradient Descent:** The last method combines the strengths of the previous methods. On one hand, it allows the use of a large number of training samples similar to SGD and on the other hand, it also offers more precise weight adjustments like in BGD[7]. In this case it does not use just a single training sample but a batch of training samples. These randomly selected training samples are then evaluated following the BGD method. This has the advantage of minimizing or even eliminating outliers as seen in Figure 10 below. This method is therefore the most commonly used method when dealing with a large number of training samples.

### 2.2.3 Overfitting vs. Underfitting

When training a neural network using training data, it tries to improve its predictions by determining new weights[14]. The number of epochs plays a crucial role in how well the network evaluates unknown test data after training. Figure 11 shows how well the network can evaluate training data using measurement data[5]. The epochs increase from left to right. The left and right diagrams show the extremes where the training process was either too short (left, *underfit*) or too long (right *overfit*).
As an example of this problem, consider a neural network that has been trained to recognize apples in images. If the network is trained with many apple samples and a high number of epochs, it becomes capable of accurately identifying the training data but struggles to classify unfamiliar images of apples that slightly deviate from the training data. These deviations could be an apple with a leaf or a worm peeking out of a hole. In this case the neural network is trained too much on the training data. This is than called overfitting. On the other side, it is also possible that a neural network has not been trained enough, which leads for example to not correctly distinguishing between a ball and an apple. This is then called underfitting. Figure 12 shows the relationship between the accuracy of predictions on training

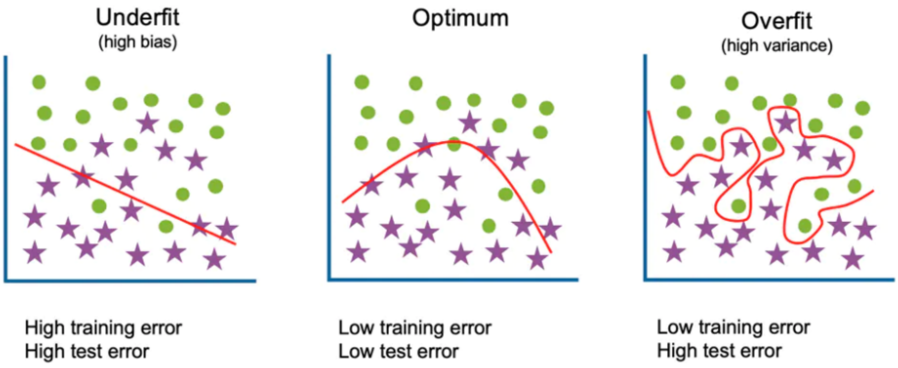


**Figure 11:** Measurement data for an underfit, overfit and an optimum. The dots and stars symbolize measurement data and the red curve symbolizes the prediction of the neural network.[15])

data and the accuracy of predictions on unknown test data.

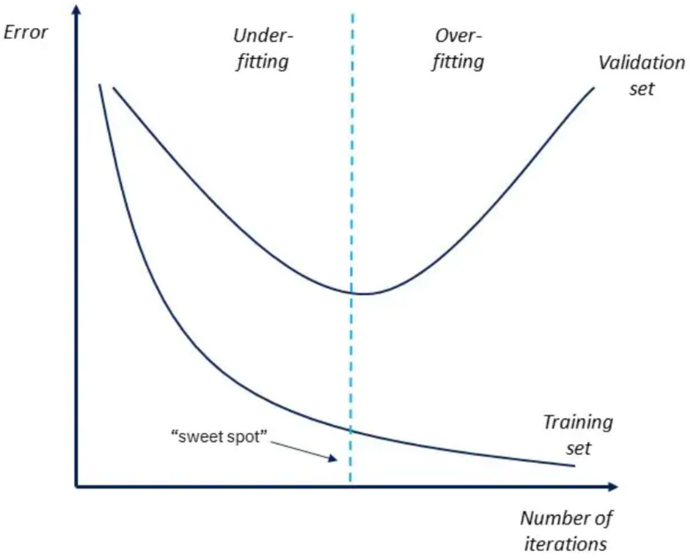


**Figure 12:** Relationship between the error and the number of iterations. The upper curve describes the accuracy of predictions on unknown test data (validation data) while the lower curve represents the accuracy on training data. The left side shows the accuracy of predictions in the case of underfitting and the right side in case of overfitting. The optimal accuracy in the middle of the graph is called *sweet spot.*[15]

Figure 12 shows the accuracy of the neural network by plotting the error against the number of training iterations. On the left side, underfitting can be observed, where the neural network struggles to classify both the training data and the unknown test or validation data resulting in low accuracy. On the right side overfitting is shown. This means that the training data can be classified very accurately but the accuracy decreases when applied to unknown data. The best accuracy is achieved at the so called '*sweet spot*' between overfitting and underfitting where the network

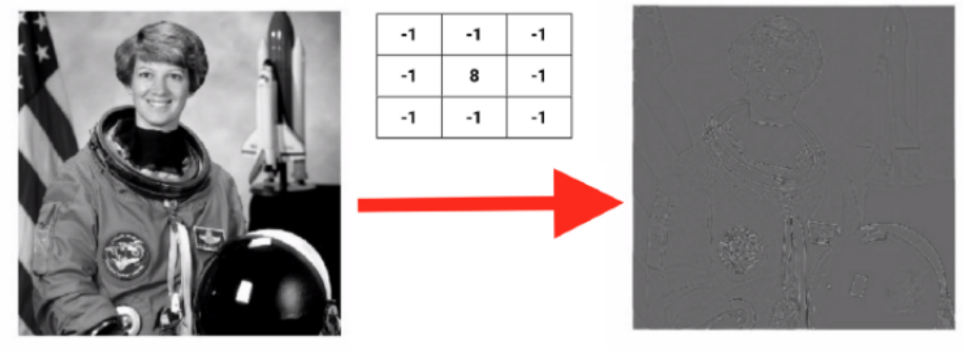
predicted well to both training and unknown data.[7]
To find this sweet spot, the number of iterations can be varied until the neural network makes the best predictions over the unknown dataset. Additionly, factors such as the model size, the size of the data set or the learning rate also have an influence on the sweet spot[5]. Therefore, experimentation is often necessary to determine the ideal settings for a particular neural network.

## 2.3 Convolutional Neural Network

The ANN has its advantages in evaluating relationships in measurement data butbut it is not suitable for object recognition on images.[16]. Measurement data as shown in Figure 1 have a specific relationship or pattern that can be discovered by an ANN. However, image recognition is more complex because the object is not always centered. For example when recognizing a digit in an image it should be centered or at least stay at the same position to find patterns. But this is not always the case. The digit can be shifted to the left, right, up, down or even slightly rotated. Here a new type of ANN was developed known as the Convolutional Neural Network or CNN for short. This was first introduced by the researchers Kunihiko Fukushima and Yann LeCun in their works from 1980 and 1989[22]. Research on this type of neural network was further advanced by Yann LeCun and his team in the 1990s and improved through increasingly larger datasets. The most well known datasets that are still used today are MNIST and CIFAR-10.
The difference between these two neural networks is that in the CNN the data to be evaluated is preprocessed. This preprocessing step is referred as *Feature Extraction* which is explained in the following chapter 2.3.1. After this an ANN is used to evaluate the data that has been preprocessed. This part is referred to as *Classification* and follows the structure of a typical neural network as described in chapter 2.2. So a CNN consists in the simplest form of two main components: Feature Extraction and Classification[7].

### 2.3.1 Feature Extraction

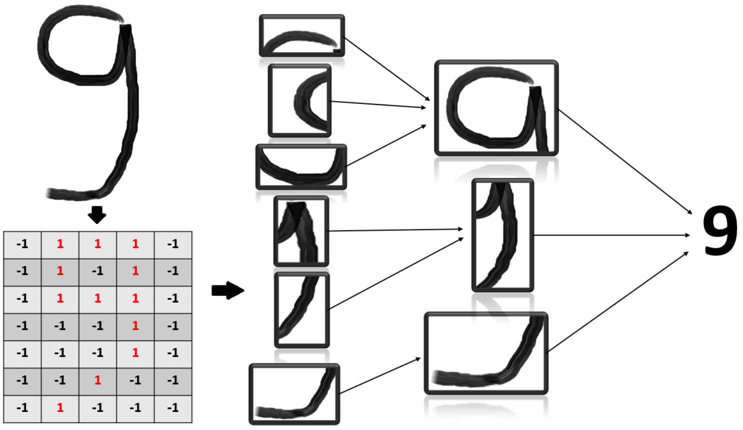
The processing of measurement data before feeding it into the ANN is called feature extraction. This process uses filters to make shapes and patterns visible[16]. Multiple filters are often used, where each filter targets specific features. A feature does not have to be limited to simple shapes like circles or edges, for example when recognizing a car, a feature could also represent a mirror or a tire. The filters can be either defined by the user or adjustable by the program allowing the program to modify not only the ANN used for classification but also the filters during the training process. In this case the number of filters has to be specified. The training process for the filters follows a similar procedure to that described in Chapter 2.2.1[23]. More details on the training of filters are described in Chapter 2.3.2.
The used filter is then shifted across the image and multiplied to the individual pixels, creating a new image. Figure 13 uses the image of Eileen Collins as an example to demonstrate the principle of feature extraction by applying a filter to detect dark edges.
After applying the filter to the image, the output represents a new image that in this case highlights the positions of dark edges on the astronauts photo. This new image can then be further modified by more filters or directly fed into an ANN for further processing.
When evaluating images, one of the limitations of ANNs becomes visible, which is the number of data points. An image taken with a modern smartphone can have a resolution of 4416 x 2944 pixels (3:2) or 4163 x 3122 pixels (4:3) which leads to approximately 13 megapixels[18]. In the case of colored images, the number of pixels



| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |

**Figure 13:** Image of NASA photo by Eileen Collins in black and white on the left side. The same photo after a filter was applied to the original photo that search for dark edges on the right side.[17]

has to be multiplied by three due to the three color channels (RGB). This means that the input layer of the ANN would consist of 39 million data points requiring a large number of neurons in the hidden layers and potentially more than just two hidden layers to make accurate predictions. To solve this issue, CNNs significantly reduce the number of input data that eventually enter the ANN for classification. This reduction is achieved through a process called *pooling* which will be discussed later in this chapter[6].



| | | | | |
|---|---|---|---|---|
| -1 | 1 | 1 | 1 | -1 |
| -1 | 1 | -1 | 1 | -1 |
| -1 | 1 | 1 | 1 | -1 |
| -1 | -1 | -1 | 1 | -1 |
| -1 | -1 | -1 | 1 | -1 |
| -1 | -1 | 1 | -1 | -1 |
| -1 | 1 | -1 | -1 | -1 |

**Figure 14:** Image of the digit nine, which is to be detected as nine by the neural network. This is first translated into machine language (array below the 9). The individual images are shapes that are searched for in the image during feature extraction in order to characterize it as a nine.

In the following, the operation of feature extraction is visualized using the example of image classification. Figure 14 shows the process of recognizing the digit as a nine using a CNN with predefined filters. The figure shows an input image which first needs to be translated into machine language. From the image an array of the digit nine is constructed. In this case grayscale values are considered and values above a certain threshold are set to '1' while values below the threshold are set to '-1'. However, it is also possible to analyze the grayscale values individually as it was done with the astronaut image in Figure 13.

Figure 14 shows on the left side an image of the digit. The goal of the program is to identify this image as a nine. To achieve this, specific filters are used to search for certain characteristics. The first layer of filters is in this case designed to detect horizontal, vertical and diagonal lines. The data generated from this is then passed to a second layer of filters, which then identify more complex shapes like circles or long lines[6]. The resulting data is then combined, transformed into a vector and fed into an ANN for classification. Ideally, this network will recognize the image then as a nine.

The individual layers which include the filters are called *convolutional layers*[5]. In addition to the filter, they also have a ReLU function which filters out all values less than zero and often include a pooling operation to reduce the output size. The following sections will provide more detailed information on these aspects.

**Filters:** The first step is the already mentioned filter. This filter can be divided into three different types depending on their dimension[23]. The filter, which will be used for the example of the digit nine, is the 2D filter. Other possible filters are the 3D filter which could consider for example color information[23] and the 1D filter which only considers depth. As shown in Figure 15 with the example of the 3D filter, the filter scans the image and provides the result as an output[21]. It can be seen that the first step in Figure 15 a) and the last step 15 d) in the sequence extend beyond the image. This is called *padding* and is not necessary[7]. So this is an additional user defined option to achieve better predictions by the CNN or not. The purpose of padding is to ensure that pixels on the edges are also respected. Another aspect of padding is to maintain the pixel number of the image after applying the filter and not reducing it.

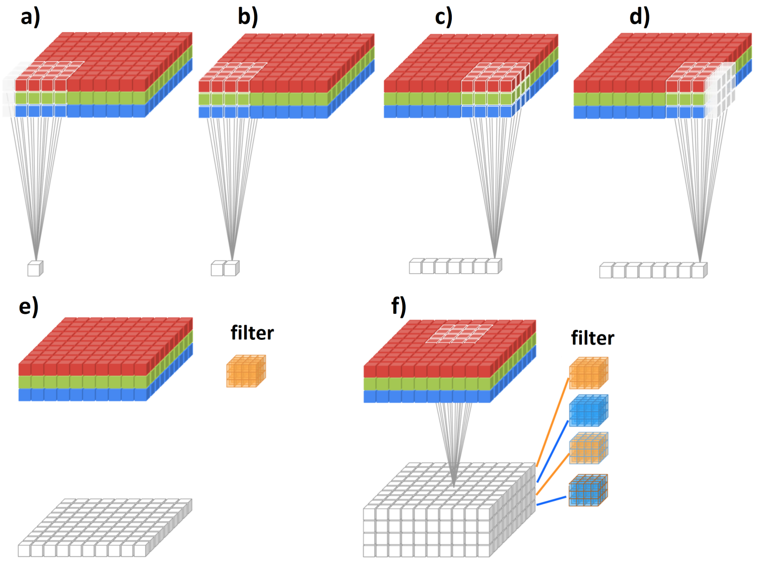
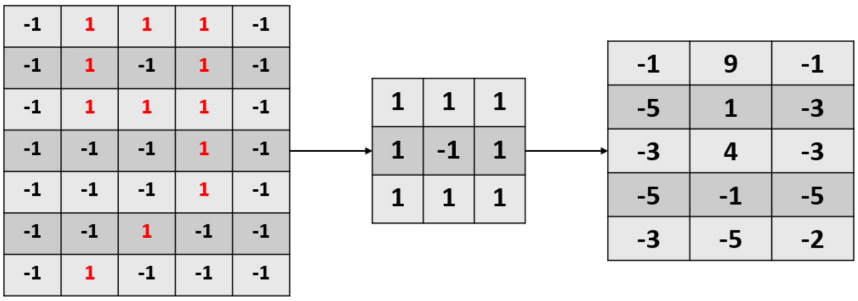


**Figure 15:** The scanning process on a colored image with a 3D filter is shown. The steps a) to d) show the steps of the filter which slowly scans the entire image. Step a) and d) also show transparent cubes to the left and right of the image which is called *padding*. Step e) shows the output after all pixels have been remapped by the filter. If several filters are used, a multi-layered output is created as shown in f).[17]

After the entire image has been scanned, an output is generated which can either have the same size as the input image or be slightly smaller depending on whether padding was applied or not. As mentioned earlier, multiple filters are typically used instead of just a single filter. Figure 15 f) shows a 3D construct as the output with each layer created by a different filter. In the case of the digit nine from Figure 14, the filters in the first convolution are 2D as the image is in black and white.
Mathematically the output of a filter is formed as the sum of the multiplications of individual elements of the input with the filter. The first element without considering padding is computed as follows[6]:

$$O(1,1) = \sum_{i=1}^{\text{row}} \sum_{j=1}^{\text{colum}} I(i,j) \cdot F(i,j). \tag{21}$$

Here, $O$ is the output, $I$ is the input and $F$ is the filter element at positions $i$ and $j$. The sum runs over the number of rows and columns. Once the sum of the products for the first element of the output image has been calculated, the filter is shifted to the right, down, etc..
In the example of the digit nine a 3x3 filter which is searching for circles leads to the following output matrix:



| | | | | |
|---|---|---|---|---|
| -1 | 1 | 1 | 1 | -1 |
| -1 | 1 | -1 | 1 | -1 |
| -1 | 1 | 1 | 1 | -1 |
| -1 | -1 | -1 | 1 | -1 |
| -1 | -1 | -1 | 1 | -1 |
| -1 | -1 | 1 | -1 | -1 |
| -1 | 1 | -1 | -1 | -1 |

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | -1 | 1 |
| 1 | 1 | 1 |

| | | |
|---|---|---|
| -1 | 9 | -1 |
| -5 | 1 | -3 |
| -3 | 4 | -3 |
| -5 | -1 | -5 |
| -3 | -5 | -2 |

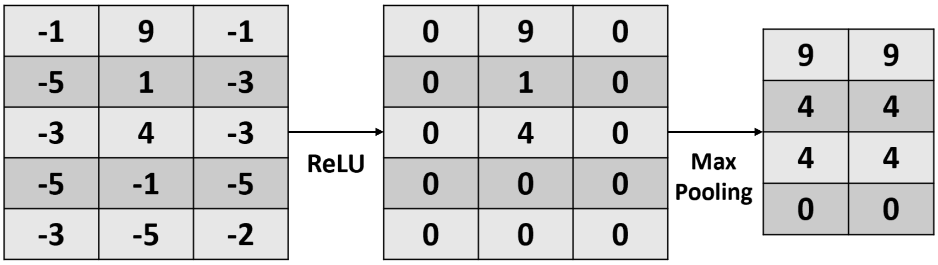
**Figure 16:** Mathematical calculation for applying a filter to the example digit nine.

To make sure that the values are in the range between -1 and 1 the output is often divided by the number of elements in the filter although this is not strictly necessary. In the upper example, no padding was used which leads to a smaller number of rows and columns in the output matrix.

**Non linear function:** In the second step, the output from the filter is further modified by, for example, setting all unimportant elements to zero[7]. Chapter 2.1 shows already a function that sets all values above zero to their original values and all values below zero to zero. This function is the ReLU function. However, other non-linear functions can also be used[5]. It processes each element in the output layer and modifies them. In the example image of the astronaut from Figure 13, the output would consist of an array that is mostly consisting of zeros resulting in a mostly gray image. In the case of the digit nine, this would mean that only three entries remain non-zero.

**Pooling:** The final step after going through a convolution is the process of pooling[5]. Unlike the previous steps, pooling is not necessary for the convolution but it is very useful as it reduces the number of input data that goes into the next convolution or ANN. Similar to the filters, pooling considers and processes only a small region at a time[23]. The size of the region depends on the user but often they are taking four to nine data points as input to avoid losing too much information. Pooling can be done in different ways. One method is to pass on the maximum value which is called *Max Pooling*[6]. Another option is to pass on the average of the input data known as *Average Pooling*[23]. But there are also many other pooling methods.. Max Pooling is the most used method and often leads to the best results for CNNs but the choice of pooling method also depends on the user.

If the steps of ReLU and pooling are applied after using the filter, the following output from the first convolution for the first filter is obtained:



| | | |
|---|---|---|
| -1 | 9 | -1 |
| -5 | 1 | -3 |
| -3 | 4 | -3 |
| -5 | -1 | -5 |
| -3 | -5 | -2 |

| | | |
|---|---|---|
| 0 | 9 | 0 |
| 0 | 1 | 0 |
| 0 | 4 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |

| | |
|---|---|
| 9 | 9 |
| 4 | 4 |
| 4 | 4 |
| 0 | 0 |

**Figure 17:** Mathematical calculation for applying ReLU and max pooling to the output of the filter for the example digit nine.

The calculation shows that without a loss of important information the number of data points is reduced. In this example the number of data points has been reduced from 35 to 8. However, it is important to note that multiple filters are applied and for each filter an output of 8 data points is generated and needs to be added. In the case of the digit nine example, three different filters are used so that the final output of the first convolution is 24 which is still almost one third smaller. When multiple convolutions are used on an image which consists of 13 megapixels this process can reduce the size.
As the example of nine seems quite simple, the following Figure 18 shows multiple convolutions and clarifies the principle of CNN.
Figure 18 shows a more complex CNN with multiple convolutions which is designed to determine whether the input image shows a car, truck, van, etc. The upper part of the figure represents feature extraction and the lower part represents the classification done by a regular ANN.

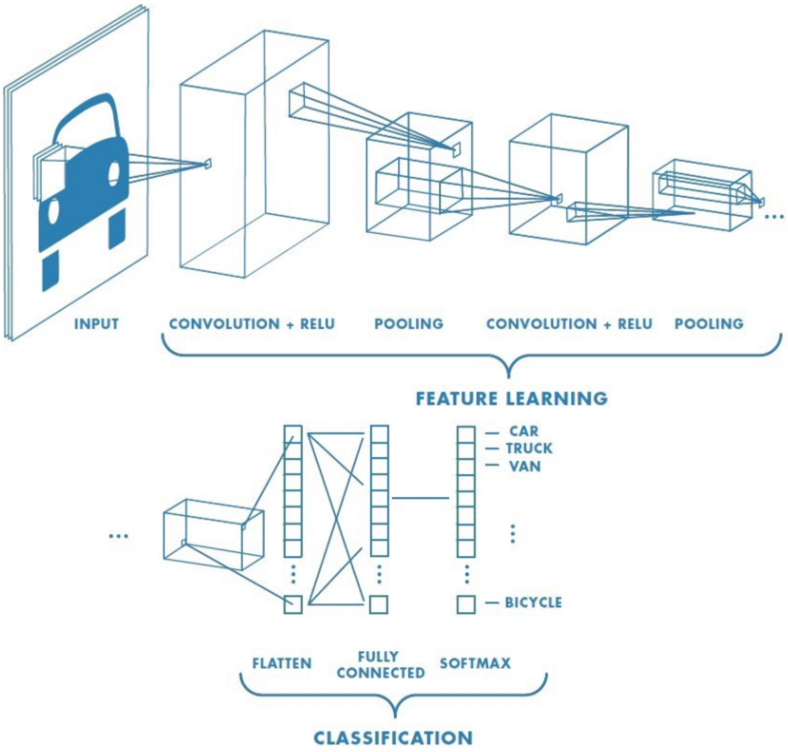


**Figure 18:** Example of a CNN with multiple convolutions that is used to detect a vehicle.[19]

### 2.3.2 Filter Backpropagation

Before a CNN is ready to make predictions, it also needs to be trained what for example a car or a truck is, how it looks and what kind of features it has. The training process for classification was already discussed in Chapter 2.2.1 so it will not be further mentioned in this chapter.

In the feature extraction the filter is changed during the training process[7]. To understand how this change works, feature extraction can be rewritten in a way similar to the already known ANN with weights and bias. The principle is shown in the following Figure 19.

Figure 19 shows what the training process of a filter looks like in the context of the already known ANN. The main difference to the ANN method is that the number of weights that are changed during the training process corresponds to the number of elements of the filters[23]. In Figure 19 this is indicated by the different colors. Each of these colors represents an element of the filter. So it follows that when a weight is changed, almost all outputs are changed too.

Compared to the ANN which determines the output using a sum ($O_i = \sum_{n=1}^{N} \omega_n I_n$), it also should be noted that with the filter not all input data is used to determine the output. The sum for the filters can be expressed as follows[24]:

$$O_{i,j} = \sum_{n=1}^{N} \sum_{m=1}^{M} \sum_{c=1}^{C} I_{i+m,j+n,c} F_{m,n,c}. \tag{22}$$

In this case the first two sums run over the dimensions of the filter (N x M) and C runs over the dimension of the image in the z-direction. For example, if C=1, the image is grayscale and if C=3, it is in RGB scale (red, green, blue).

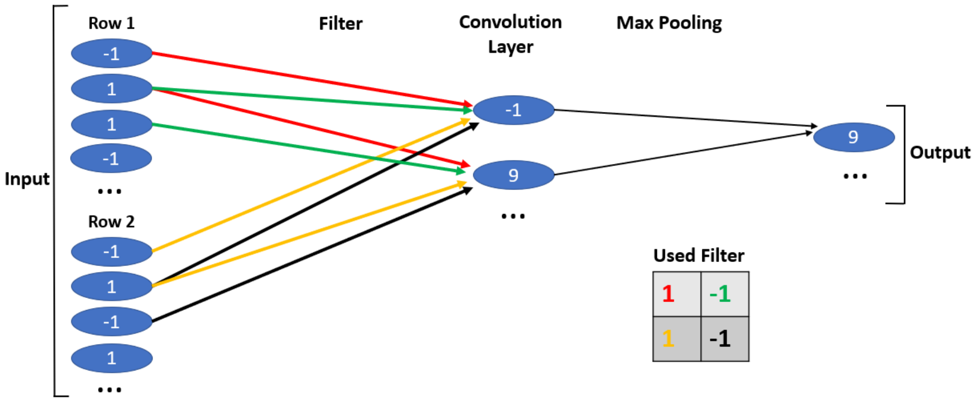


**Figure 19:** Representation of the training process of filters using the context of an ANN. The colored arrows between the input and convolutional layer represent the elements of the used filter (Figure based on [20]).

The systems error can be determined using the mean squared error like in Chapter 2.2.1. The other error calculations are also possible. Like in the backpropagation of a regular neural network, the filters dependence is determined using the chain rule[5]:

$$\frac{\delta E}{\delta F} = \frac{\delta E}{\delta I}\frac{\delta I}{\delta F} \tag{23}$$

Here, $F$ represents the filter. To get the dependence of any filter $F_{i,j}$, the individual errors must be added together:

$$\frac{\delta E}{\delta F^l_{i,j}} = \sum_{n=0}^{H-N}\sum_{m=0}^{W-M}\frac{\delta E}{\delta I^l_{n,m}}\frac{I^l_{n,m}}{\delta F^l_{i,j}}.$$

Here, $H$ and $W$ represent the dimensions of the input array. The last part of the equation can be expressed in terms of the output of the previous layer as follows:

$$\frac{\delta E}{\delta F^l_{i,j}} = \sum_{n=0}^{H-N}\sum_{m=0}^{W-M}\frac{\delta E}{\delta I^l_{n,m}}O^{l-1}_{i+n,j+m}. \tag{24}$$

The new filter then takes the following form:

$$\hat{F}_{i,j} = F_{i,j} - \left(lr\,\frac{\delta E}{\delta F^l_{i,j}}\right). \tag{25}$$

The procedure for determining the input data of the previous layer is very similar[5]. Again, the dependency is determined using the chain rule. To get the dependency of any input $X_{i,j}$, the individual errors must be added together:

$$\frac{\delta E}{\delta X^l_{i,j}} = \sum_{n=0}^{N}\sum_{m=0}^{M}\frac{\delta E}{\delta I^l_{n,m}}\frac{I^l_{n,m}}{\delta X^l_{i,j}}.$$

The difference from the new filter lies in the additional activation function:

$$\frac{\delta E}{\delta F^l_{i,j}} = \sum_{n=0}^{H-N}\sum_{m=0}^{W-M}\frac{\delta E}{\delta I^l_{n,m}}\omega^{l+1}_{i+n,j+m}f'(x^l_{i,j}). \tag{26}$$

The new input then takes the following form:

$$\hat{X}_{i,j} = X_{i,j} - \left( lr \frac{\delta E}{\delta X^{l}_{i,j}} \right). \tag{27}$$

For simplicity the matrix notation is not shown.
Another point that plays a role in the training process is pooling, which also has an effect on the training process. For max pooling only the error of the winning value from the previous layer is used leaving the other values unchanged[5].

## 2.4 Position Detection

The shown neural networks make it possible for an AI to recognize and classify objects. But a simple classification of objects is often not enough for many scenarios. For example, such a method cannot predict the position or size of an object. Another problem arises when multiple objects are present in the same image whether they are of the same type or have different types. This can lead to less accurate predictions by the network. This problem is also noticeable in reading words or numbers. If an image contains a multi digit number like '42', it can only be recognized using a neural network that has an output for the number '42.' This would mean that every number needs to be represented in an output which leads to infinitely large neural networks. Additionally, a lot of training samples would be required for each number.
A solution to this problem would be to determine the position in the image. In the example of '42', if the numbers '4' and '2' and their positions in the image are recognized, the neural network can easily predict the number '42' without ever seeing an example image of the number '42.' In this case the output layer would not be huge, it would simply consist of 10 neurons with each one representing one digit.
This is made possible by various algorithms such as the *Sliding Window Object Detection* (short SWOD)[25], the *Region-based CNN* (short R-CNN), the *Fast Region-based CNN* (short Fast R-CNN), the *Faster Region-based CNN* (short Faster R-CNN) or the *You Only Look Once* (short YOLO) algorithm[26]. The following section provides a brief explanation of these algorithms before going deeper into the YOLO algorithm.

**SWOD:** The simplest way to detect the position of an object in an image is to raster it[25]. In this method an area which represents just a small part of the image is evaluated by the neural network. After this part has been processed through the CNN, it is shifted by a small step to the right or down, etc. and the new area forms the input to the CNN. If an object is present in an area, the position and size is noted additionally to the prediction value. These values then form the output of the CNN. While this method is straightforward, it has several disadvantages. For example, the size of the object is determined by the size of the area which depends on the users choice rather than the neural network itself. This means that if the selected area is too small or too large, the object may not be detected. Additionally, this approach requires a significant amount of computational time.

**R CNN:** The second method was introduced 2014 by a group of researchers from the University of California, Berkeley[27]. In this method the image is first divided into 2000 regions[28]. The Selective Search algorithm is used for this process, which selects regions based on color, texture, size or shapes[25]. The operation of this algorithm is shown in Figure 20. The algorithm is applied multiple times to reduce the number of regions that belong together until it is reaching 2000 regions. The size of each region then determines the size of the rectangle. These regions are then scaled into a normed vector for the CNN. Then a trained Support Vector Machine algorithm is used to classify each region as either a region of interest or as background. Since the image is not individually scanned, the computational time is lower compared to the SWOD algorithm but it still does not operate in real-time as the 2000 regions are individually evaluated by a CNN, which takes up to 47 secounds per image[7]. Additionally, this algorithm consists of different operation layers which do not allow a training process from start to finish.

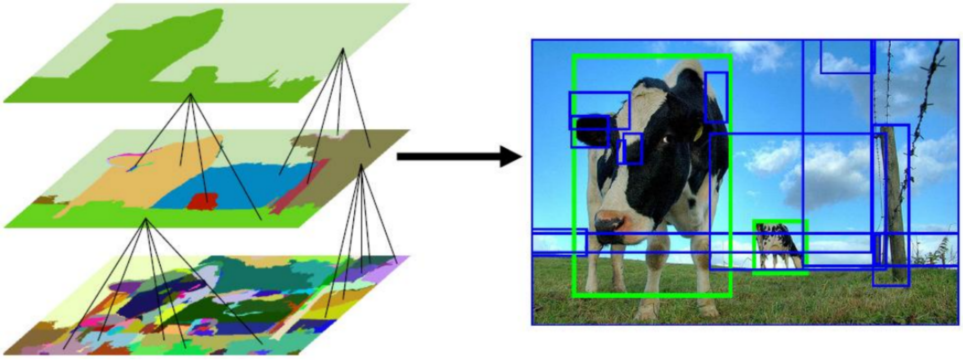
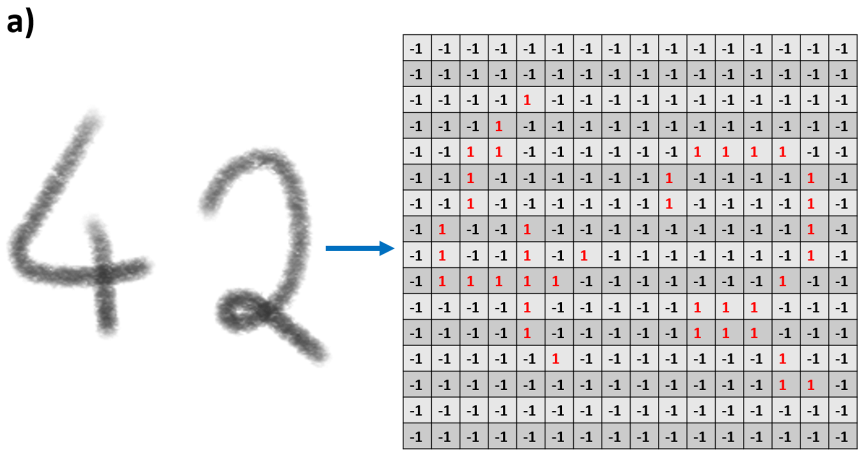


**Figure 20:** Selective search algorithm using the example of a cow. The algorithm is applied three times on the example image to make the number of regions smaller and smaller. The remaining regions then form the size of the rectangle and the area to be examined by the CNN.[29]

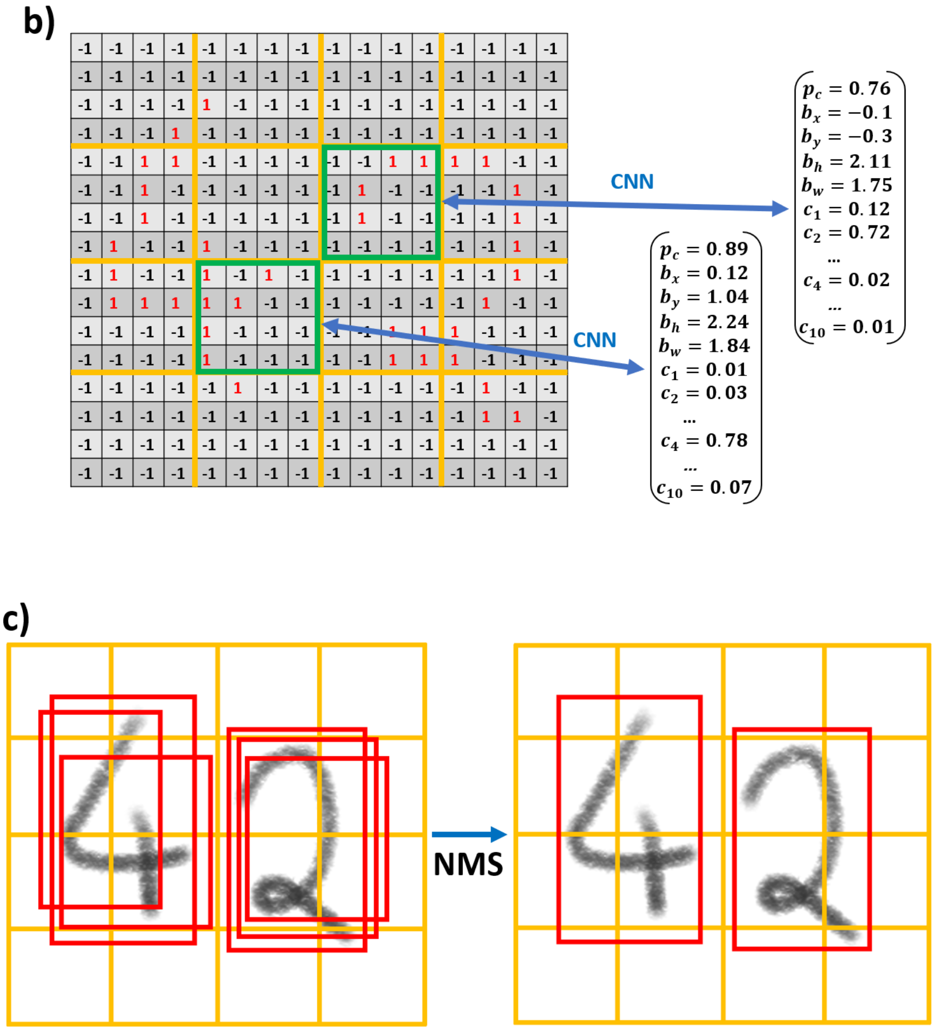
**Fast R CNN:** A year later Ross Girshick, who had previously worked on the R-CNN, introduced a faster method called the Fast R-CNN[27]. This method is based on the R-CNN algorithm with some small modifications. The biggest difference that makes it faster is that the CNN does not need to be applied 2000 times to different regions of the image[28]. Instead of 2000 separate applications, the CNN is applied just once to the entire image. For this, the Fast R-CNN introduces another layer called the ROI Pooling Layer. In the first step, the regions of interest determined by the Selective Search algorithm are passed through the CNN. The resulting output is then downsized using an ROI pooling layer. Unlike the squares mentioned in Chapter 2.3.1, these squares are not identical in size but depend on the size of the corresponding region determined by the Selective Search algorithm. This data is then fed into a neural network that outputs the probability of the object and its position[7]. Compared to the R-CNN algorithm, this method achieves higher speed and accuracy and can be learned through a training process since only one CNN is used. However, this algorithm still relies on the Selective Search algorithm which slows it down. As a comparison, a run through the Selectet Search algorithm takes 2 seconds while for the CNN it takes only 0.1 seconds[16].

**Faster R CNN:** Another year later, the algorithm was further simplified[30]. It was introduced by Ross Girshick and others. This algorithm is almost identical to the Fast R-CNN algorithm with the only difference that the Selective Search algorithm is replaced with the Region Proposal Network (RPN) algorithm[28]. The RPN consists of a neural network which guides the Fast R-CNN where to search[16]. Unlike the Selective Search algorithm, the RPN is trainable and can be combined with the Fast R-CNNs CNN to create a large neural network[27]. This allows further speed improvements, but still does not enable real-time detection.

**YOLO:** After none of the algorithms in the R-CNN family could perform fast real-time analysis, a completely new algorithm was created. This algorithm was first introduced in a 2016 paper by Joseph Redmon, Ross Girshick and others. This algorithm allows real-time analysis with a speed of 45 to 150 images per second, depending on the computational power[31]. Additionally, it provides more accurate object detection compared to the Fast R-CNN because, like seen in Figure 20, the half of the mistakes are errors in the background determination. The only disadvantage of this algorithm compared to the R-CNN algorithms is that the object position is determined sometimes less precisely.
The difference lies in the neural network used in the YOLO algorithm, which does not just have a single output which indicats the probability of an object being in the image. It also provides the objects position, height and width of the bounding box as well as what type of object it is[16]. This results in a minimal vector with six dimensions. The first element represents the probability of an object being in the image, the next two elements represent the x and y position, the next two elements represent the height and width of the bounding box and the last one or more elements indicate what type of object is in the image, such as a person, a dog, a car, etc. What is new about the YOLO algorithm is that it does not take the entire image as a whole, but it divides it into sections. The user can choose the size of these sections freely to achieve the best possible results. Each of these sections is then analyzed for probability, position and size using a normal CNN. The process is shown in the example of '42' in Figure 21.
Figure 21 shows the image of the number '42'. In the first step, similar to Figure 14, the image is converted into an array based on the intensity of the grayscale. The array is then divided into 16 sections using the YOLO algorithm (as indicated by the yellow lines in 21 b)). Each of these sections is then classified by a CNN which produces an output including the probability of an objects presence in the section, the class of the object (if it exists either 0, 1, 2, 3, ..., or 9) and the position and size of the object. The resulting data is then visualized using bounding boxes as seen in 21 c). Since an object can be present in multiple sections at the same time, there may be multiple bounding boxes for the same object. These groups of bounding boxes which belong togehther are then isolated using the Non-Maximum Suppression (NMS), which chooses the bounding box with the highest probability score and removes the others[32]. The final image in Figure 21 shows the output predicted by the network.



| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 | 1 | -1 | -1 |
| -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | 1 | -1 |
| -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | 1 | -1 |
| -1 | 1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | -1 |
| -1 | 1 | -1 | -1 | 1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | -1 |
| -1 | 1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |



| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 | 1 | -1 | -1 |
| -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | 1 | -1 |
| -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | 1 | -1 |
| -1 | 1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | -1 |
| -1 | 1 | -1 | -1 | 1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | -1 |
| -1 | 1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

**Figure 21:** Representation of the process that the YOLO algorithm used to determine the number '42'. The translation into machine language in a), the division into different sections which are then analyzed using the CNN in b) and the resulting predictions and bounding boxes which are processed using Non-Max Suppression (NMS) in c) are demonstrated.

By using this algorithm, the '42' was recognized as '42' without having '42' as an output in a neural network. With this method any number can be predicted if the digits 0 to 9 are known.

## 2.5 Datasets

In the previous chapters, a dataset was used to train and test a network. Generally, a dataset is divided into three parts: the training dataset, the validation dataset, and the test dataset.[5]

**Trainigs dataset:** The training dataset forms the largest part of the entire data set. This is used to train the network to find patterns and relationships between the individual samples. The weights are then determined using this data.

**Validation dataset**: The learning process is monitored during training using the validation dataset. This consists of data that is not used to change the weights. The goal of this dataset is to adjust the hyperparameters such as the learning rate, batch size, etc. during training in order to achieve better results.

**Test dataset:** The test dataset is used to analyze the final architecture with its hyperparameters. The final accurcy is then determined from the networks incorrect and correct predictions for this data.

## 2.6 Confusion Matrix

In order to evalute the results of a test dataset or to determine optimal parameters through a validation dataset the output of the network must be evaluated. The confusion matrix or error matrix could be used for this purpose in machine learning. After the network has made certain predictions, they are compared with the actual data. There are now 4 options how the evaluation looks like as shown in the following table. The first and desired case is *True Positive* (TP for short)[32]. In this case the network has correctly predicted an object. The contrast to this is *True Negative* (TN for short) in which the network did not predict an object and no object is present. In this two cases the output of the network is correct. The other two cases then describe mistakes that the network made. First of all there is the case that the network made a prediction without an object is present which is referred to as *False Positive* (FP for short) or that the network did not predict an object even though it is present. This is than called *False Negative* (FN for short).

**Table 2:** Confusion matrix

| | object | no object |
|---|---|---|
| prediction | TP | FP |
| no prediction | FN | TN |

To determine the quality of the network from the values mentioned above, the *F1-score* is used. It is determined through the *Precision* and the *Recall.* Precision describes the probability that a result is relevant and Recall describes the probability of finding a relevant object. Both probabilities can be determined from the confusion matrix as shown:

$$\mathrm{PE} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \tag{28}$$

$$\mathrm{RE} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{29}$$

The F1 Score is then calculated as follows:

$$\mathrm{F1} = 2\frac{\mathrm{PE} \cdot \mathrm{RE}}{\mathrm{PE} + \mathrm{RE}}. \tag{30}$$

# 3 Color Centers

The methods of object recognition, presented in the previous chapter, will then be applied to color centers in a diamond. Nitrogen-vacancy (NV)-centers serve as color centers in the thesis. In this chapter their properties and measurement methods will be explained.

## 3.1 Color Centers in Diamonds

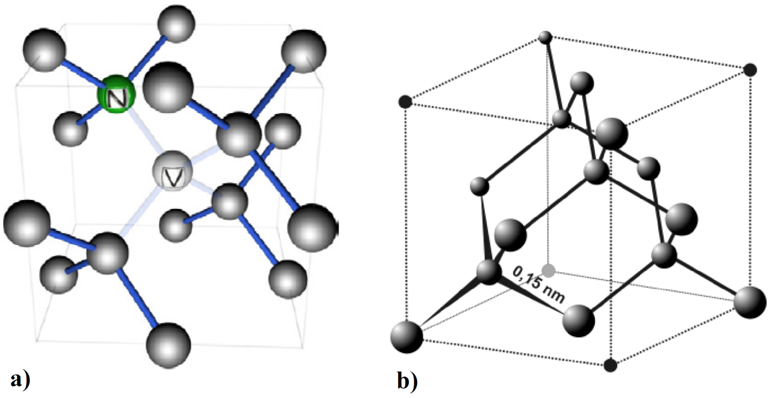


**Figure 22:** Crystal structure in a diamond with a defect in a) and without a defect in b). The defect consists of a combination of a nitrogen atom and a vacancy next to it.[33]

Diamonds with a Mohs scale of 10 are the hardest naturally existing material in the world, requiring no artificial creation or cultivation[34]. Diamonds consists of a periodically repeating structure, also known as a lattice. Figure 22 shows this periodic structure. Additionally, to that Figure 22 a) already shows a color center within this lattice. This color center consists of a vacancy and another atom, in this case a nitrogen atom. Both defects are located at theoretical positions of carbon atoms so that the lattice structure remains unchanged.
The use of a diamond as a medium for color centers offers several advantages. Firstly, diamonds are highly stable under external conditions allowing the color centers to remain in a constant position even for experiments at room temperature. Secondly, diamonds have a large band gap of $\approx 5.48\,\text{eV}$, which allows precise excitation of the color centers without influencing the diamond itself. Additionally, diamonds consist of 98.9% of the isotope $^{12}$C[35] which has a total nuclear spin of $I = 0$ which does not interact with the spin of the color center.

### 3.1.1 NV-center

The in the following experiment used color centers are the NV$^-$-centers as seen in Figure 22 a). There are different types of NV-centers that differ in charge. The most commonly discussed types are the neutral NV$^0$-center and the one time negatively charged NV$^-$-center. The main difference between these two types is their energy spectrum. The zero-phonon line for the NV$^0$-center is located at approximately $575\,\text{nm}$ ($\approx 2.156$ eV) while for the NV$^-$-center it is at $637\,\text{nm}$ ($\approx 1.945$ eV) and at $1042\,\text{nm}$ ($\approx 1.190$ eV)[36]. Figure 23 shows the energy spectrum of both, the neutral and negatively charged NV-center. Since this work focuses only on NV$^-$-centers they will simply be called NV-centers in the rest of this thesis.

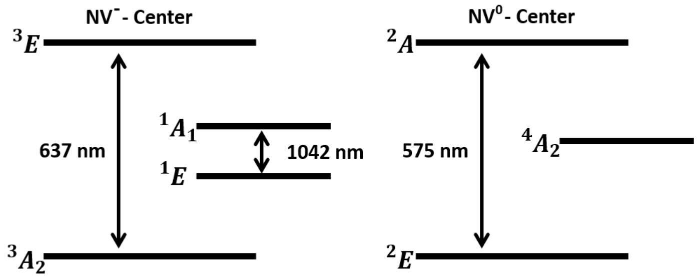


**Figure 23:** Energy level diagram of the NV$^-$-center on the left and for the NV$^0$-center on the right.

To measure the NV$^-$-centers, the energy level diagram shown in the figure is used. A two-dimensional intensity profile of a defined location is measured using a confocal microscope. The exact setup of such a confocal microscope, with its components used in this experiment, is explained in Chapter 4. In this process a laser with a shorter wavelength than the zero-phonon line is used. This is to enableing absorption after the stokes shift. The electron then goes non-radiatively to the vibronic ground state of the first excited state. From there, the NV$^-$-center emits a photon with a wavelength of 637 nm by transitioning the electron back to the ground state. These photons are then measured in the confocal microscope. Intensity maxima in the captured scan are highly likely to correspond to a NV$^-$-center.

## 3.2 ODMR-Spectroscopy

The energy level diagram, shown in Figure 23, only shows the electronic states but due to the electron spins these states are further split. Spin states of electrons or the nucleus as well as measurements for Rabi-oscillations or relaxation times can be measured through Optical Detected Magnetic Resonance-Spectroscopy (short ODMR spectroscopy). In this measure technique microwave radiation in the GHz range is directed at the NV-center. The intensity is then measured using the confocal microscope. Dips in the spectrum can then be attributed for example to spin states. The measurements enabled by this method will be briefly introduced in the following chapters.

### 3.2.1 Electron Spin

As mentioned earlier, Figure 23 only shows the electronic states. These states are further split due to the electron spin. The splitting of the individual levels is shown in Figure 23 by the upper index of the energy level. In this case $^3E$ and $^3A_2$ represent triplet states, $^2A$ and $^2E$ are doublet states, $^1A_1$ and $^1E$ are singlet states and $^4A_2$ corresponds to quartet state[37][36]. This leads to an integer spin for an NV$^-$-center and a half-integer spin for an NV$^0$-center.

The NV$^-$-center has two unpaired electrons leading to magnetic quantum numbers of $m_s = 0, +1, -1$. Due to the magnetic dipole moment of the NV-center this results in a split of energy levels even without an externally applied magnetic field. This so called zero-field splitting creates an energy gap of 2.88 GHz for the first excited state and 1.42 GHz for the ground state. In the ODMR spectrum, which is shown on the left side in Figure 24, this can be seen as dips.

Figure 24 shows the ODMR measurement with the two dips corresponding to the first excited state and the ground state on the left. The dips at these frequencies can

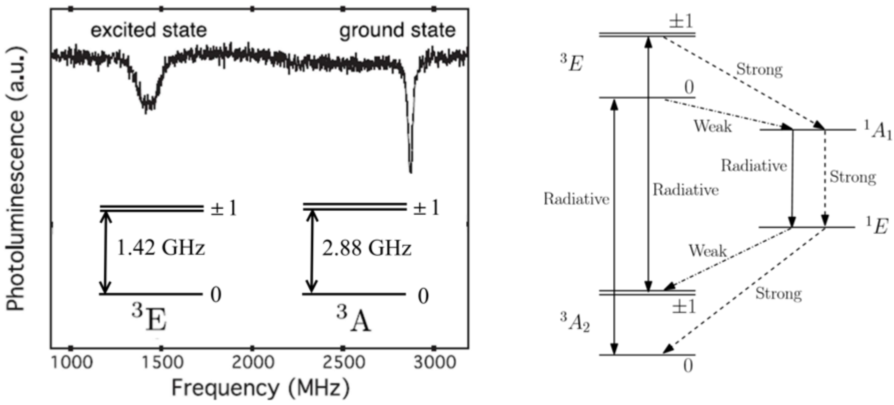


**Figure 24:** ODMR spectrum from a NV$^-$-center with dips at the frequencies of 2.88 GHz for the first excited state and 1.42 GHz for the ground state (left). Energy scheme with probabilities of transitions (right).[36]

be explained using the energy level diagram and their probabilities and lifetimes. In Figure 24 on the right, the transition probabilities are also shown in the energy level diagram. The possible transitions are indicated by the arrows, where dashed arrows representing phonon emission and solid arrows representing photon emission. Above the arrows, the transition probabilities are noted which can be either strong or weak. For this process it is important that the lifetime of singlet states is much longer then that of triplet states. If the system is initially in the state $m_{s,g} = 0$ it is excited whitout changing its spin to the state $m_{s,e} = 0$. Then it falls back to the ground state by emitting a photon where it is excited again and so on, which keeps the measured intensity constant. But if the spin is changed to $m_{s,g} = \pm 1$ or $m_{s,e} = \pm 1$ state due to a fitting microwave radiation there is a much higher probability that it will go through the singlet states and since the lifetime is longer this results in a decrease in intensity. These effects are then reflected as dips in the ODMR spectrum. Furthermore, the process of pumping to the ground state, as mentioned later, can also be shown in the pathways through the energy level scheme which most likely lead to the $m_{s,g} = 0$ state after a while.

### 3.2.2 Nuclear Spin

In addition to the electron, the nucleus of the nitrogen atom also has a spin. There are primarily two isotopes of nitrogen that occur. The most existing isotopes are the $^{15}$N isotope and the $^{14}$N isotope. The $^{15}$N isotope has a nuclear spin of $I = \pm\frac{1}{2}$ while the $^{14}$N isotope has a nuclear spin of $I = 0, \pm 1$. Due to the interaction between the nuclear spin and the electron spin an additional splitting occurs as shown in Figure 25 a) in the energy level diagram and in b) the ODMR scans. This is known as hyperfine structure and is much smaller in comparison to the electron spin. For example the interaction results in only a 3.05 MHz splitting in the ground state of a $^{15}$N isotope compared to the 2.88 GHz splitting caused by electron spin[38].

## 3.3 External Magnetic Field

In addition to the zero-field splitting and the interaction between nuclear and electron spins, an externally applied magnetic field can also influence the splitting. This leads to a splitting of the $m_s = \pm 1$ state. So the dips, shown in Figure 24, are further split in addition to the hyperfine structure. Increasing the external magnetic field strength also increases the energy gap between the $m_s = 1$ and $m_s = -1$ states. In the ODMR measurements this can be seen as a shift in the dip to higher or lower frequency ranges. Figure 25 c) and d) shows the position of the dip in the frequency spectrum with respect to the strength of the external magnetic field. It is important to note that only the external magnetic field parallel to the NV axis contributes to this splitting.

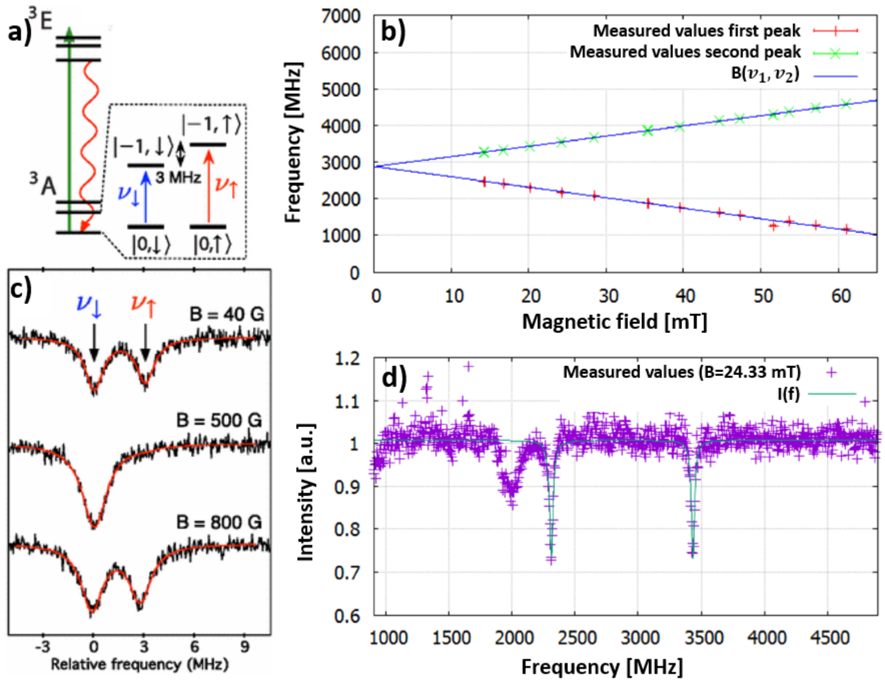


**Figure 25:** Energy level diagram showing the splitting due to the interaction between electron and nuclear spin in a $^{15}$N isotope in a) as well as the ODMR spectrum at different applied external magnetic fields in c)[38]. The measured shift of dips in the first excited state with increasing magnetic field strength is shown in b) as well as a ODMR spectrum at a parallel to the NV-axis magnetic field strength of 24.33 mT, which shows the dips for the first excited state and the ground state in d)[39].

The impact of an external magnetic field on the energy level diagram is shown in Figure 25. In a) the energy level diagram from Figure 23 is updated with a nuclear splitting for a $^{15}$N isotope. Below in c) the ODMR spectrum for the nuclear spin at different external magnetic field strengths is shown. It can be seen that the transitions are shifted to higher or lower frequency ranges and that they can overlap at a magnetic field strength of 500 G. On the right side of the figure the behavior of the shift of dips in the first excited state with increasing magnetic field strength is shown in b). In d) an ODMR spectrum at a magnetic field strength of 24.33 mT parallel to the NV axis is shown. It can be seen that the split energy levels of the ground state and the first excited state can also overlap at a specific magnetic field strength.

The system of the energy level scheme can be described using the following Hamilton operator[38][40]:

$$\hat{H} = \mu_B g_e \vec{B}\hat{\vec{S}} + D\left(\hat{S_z}^2 + \frac{\hat{S}(\hat{S}+1)}{3}\right) + E\left(\hat{S_x}^2 - \hat{S_y}^2\right) + A\hat{S}\hat{I}. \tag{31}$$

Here $\mu_B$ stands for the Bohr magneton, $g_e \approx 2$ for the g-factor of the electron, $\vec{B}$ for the external magnetic field, $\hat{S}$ and $\hat{I}$ for the electron spin or nuclear spin operator, $D$ for zero-field splitting and $A$ for hyperfine coupling. In this equation, the first term describes the external magnetic field, the second and third terms describe the axial and non-axial parts of the zero-field splitting and the fourth term describes the interaction between the electron and nuclear spin.

From Equation 31, the angle between the external magnetic field axis and the NV axis can be determined. To do this, the external magnetic field is described in terms of the polar angle $\theta$, the azimuthal angle $\varphi$ and the magnetic field strength $B_0$. The last term is ignored for simplicity. Inserting the magnetic field the spin operators and eigenenergies $\lambda_0 = E_0$, $\lambda_1 = E_0 + \nu_1$, and $\lambda_2 = E_0 + \nu_2$ leads to the parallel component of the external magnetic field[40][41]:

$$B = \frac{1}{\mu_B g_e} \sqrt{\frac{\nu_1^2 + \nu_2^2 - \nu_1 \nu_2 - D^2 - 3E^2}{3}}. \tag{32}$$

And for the angle between the external magnetic field axis and the NV axis under the approximation $D \gg E$:

$$\alpha = \frac{1}{2} \arccos \left( \frac{7D^3 + 2P - 3D(\nu_1^2 + \nu_2^2 - \nu_1 \nu_2 + 9E^2)}{9D(\nu_1^2 + \nu_2^2 - \nu_1 - D^2 - 3E^3)} \right) \tag{33}$$

with

$$P = (\nu_1 + \nu_2)[2(\nu_1^2 + \nu_2^2) - 5\nu_1 \nu_2 - 9E^2]. \tag{34}$$

By using these equations, the orientation of the NV-center in diamond can be determined based on a known magnetic field. This not only simplifies the Hamiltonian operator by eliminating the non-axial term but also allows to set the maximum possible external magnetic field splitting in experiments.

## 3.4 Rabi-Oscillation

When a two-level system is disturbed by an electromagnetic field or an oscillating magnetic field, it oscillates between the two states. This oscillation is called Rabi-oscillation. In the case of NV-centers it involves the states $m_s = 0$ and $m_s = 1$ or $m_s = -1$ as shown in Figure 26 a). As described in the previous chapter, an external magnetic field must be applied to split the states. The Rabi-oscillation can be derived using the time dependent Schrödinger equation by inserting the Hamiltonian. The Hamiltonian consists of two parts, the non disturbed term $\hat{H}_0$ and the disturbed term $\hat{H}_I$[42][43].

$$\hat{H} = \hat{H}_0 + \hat{H}_I = \hbar \omega_{eg} \left| e \right\rangle \left\langle e \right| + \frac{\hbar \Omega_0}{2} \left( \mathrm{e}^{i \omega_I t} \left| g \right\rangle \left\langle e \right| + \mathrm{e}^{-i \omega_I t} \left| e \right\rangle \left\langle g \right| \right). \tag{35}$$

$\left| e \right\rangle$ and $\left| g \right\rangle$ stands for the excited and ground state, $\omega_{eg}$ for the frequency between the states, $\omega_I$ for the frequency of the disturbing electromagnetic field, $\hbar$ for the Planck constant and $\Omega_0$ for the Rabi-frequency at resonant excitation.

As solution the following wave function is used:

$$\left| \Psi(t) \right\rangle = c_g(t) \mathrm{e}^{i \frac{E_g t}{\hbar}} \left| g \right\rangle + c_e(t) \mathrm{e}^{-i \frac{E_e t}{\hbar}} \left| e \right\rangle. \tag{36}$$

In this case $E_e$ and $E_g$ represent the eigenenergies of the excited and ground states. Substituting Equation 35 and Equation 36 into the time dependent Schrödinger equation and solving the coupled differential equations leads to the following expressions for $c_g(t)$ and $c_e(t)$ [44]:

$$c_g(t) = \mathrm{e}^{i\frac{\Delta t}{2}} \left( \cos\left(\frac{\Omega_R t}{2}\right) - i\frac{\Delta}{\Omega_R} \sin\left(\frac{\Omega_R t}{2}\right) \right), \tag{37}$$

$$c_e(t) = i\frac{\Omega_0}{\Omega_R} \mathrm{e}^{i\frac{\Delta t}{2}} \sin\left(\frac{\Omega_R t}{2}\right). \tag{38}$$

And for the probabilities of being in the ground state or in the excited state:

$$p_g(t) = 1 - p_e(t), \tag{39}$$

$$p_e(t) = \left(\frac{\Omega_0}{\Omega_R}\right) \sin^2\left(\frac{\Omega_R t}{2}\right). \tag{40}$$

With the Rabi-frequency $\Omega_R = \sqrt{\Omega_0^2 + \Delta^2}$ and the detuning $\Delta$, which is the deviation between the Rabi-frequency and the Rabi-frequency on resonant excitation.

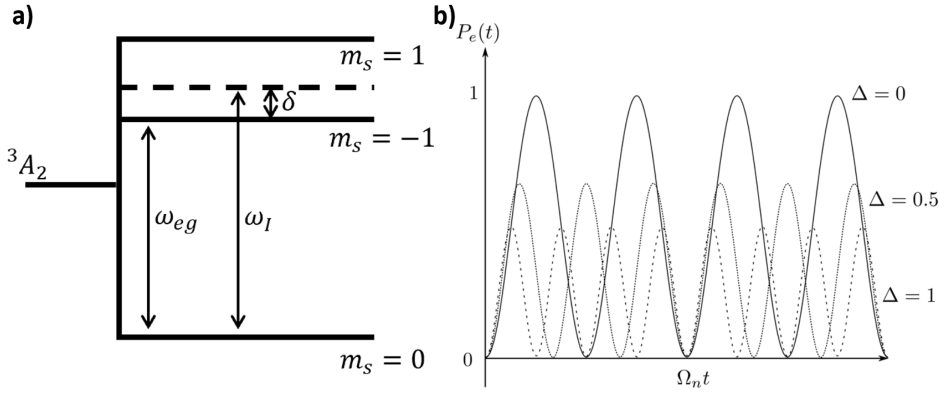


**Figure 26:** Energy level diagram of the split ground state with additional dashed line for the detuned excitation on the left. Probability of being in the excited state at different detunings on the right[44].

Figure 26 a) shows the energy level diagram of the ground state $^3A_2$ as well as an additional dashed line representing the detuning $\Delta$. In b) the probability of being in the excited state as a function of time for specific detunings $\Delta$ is shown. It can be seen that when there is no detuning, the amplitude of the Rabi-frequency is at its maximum.

In the context of the experiment, Rabi-oscillations can be indirectly measured by varying the time intervals during which the microwaves are applied. It is important to note that this measurement needs to be repeated many times. From the measured Rabi-oscillation the Rabi-frequency can be determined which then forms the frequency of the $\pi$-pulse. But it is important to test if there is a detuning by making measurements at slightly different transition frequencies. Once the time for the $\pi$-pulse is determined out of the frequency, further measurements can be made such as pulsed ODMR, determination of the relaxation times or the use of composite pulses.

## 3.5 Bloch Sphere

To show superposition and quantum states in a two-level system a Bloch sphere is often used. This sphere represents two pure states, which are typically denoted as

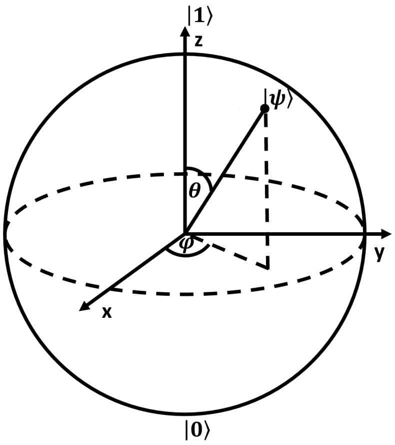


**Figure 27:** Bloch sphere for a two-level system with the states $|0\rangle$ and $|1\rangle$. In addition, a superposition state $|\Psi\rangle$ is shown.

$|0\rangle$ and $|1\rangle$ and located at the poles of the sphere as shown in Figure 27. These states form the orthonormal basis of the vector space. The remaining states are then constructed as superpositions of the $|0\rangle$ and $|1\rangle$ states.

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{41}$$

Where $\alpha$ and $\beta$ represent the probability amplitudes with the condition $|\alpha|^2+|\beta|^2 = 1$. To describe the state $|\Psi\rangle$ the following expression is used with $\theta$ and $\varphi$ being the angles shown in Figure 27 for the quantum state[45].

$$|\Psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \mathrm{e}^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{42}$$

With the condition that $0 \leq \theta \leq \pi$ and $0 \leq \varphi \leq 2\pi$. When a pulse is applied to the two-level system, it causes a displacement of the quantum state. If the initial state is $|0\rangle$, a $\pi$-pulse would result in a shift to $|1\rangle$ while a $\frac{\pi}{2}$-pulse would lead to the state $\frac{1}{\sqrt{2}}|0\rangle + \mathrm{e}^{i\varphi}\frac{1}{\sqrt{2}}|1\rangle$.

### 3.5.1 Pulsed ODMR

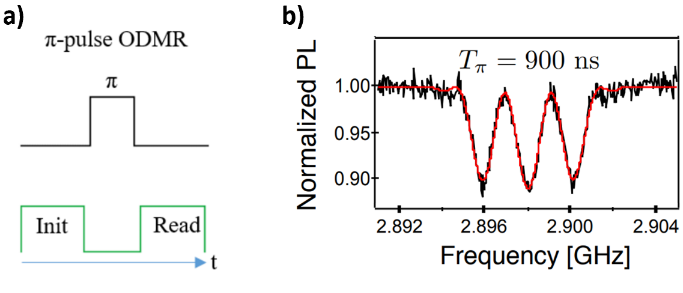


**Figure 28:** Pulse sequence for performing pulsed ODMR-spectroscopy in a) as well as a pulsed ODMR image of a $^{14}$NV-center at a $\pi$-pulse of 900 ns in b).[47][46].

In Chapter 3.2.1, the ODMR measurement was introduced but the focus was mainly on cw ODMR. Another method is the use of pulsed ODMR, where as an example

a $\pi$-pulse is used. Experimentally, such a pulse can be determined as written in Chapter 3.4 using the Rabi-frequency. For pulsed ODMR, the system is pumped into the $m_s = 0$ state[47]. Due to the probabilities of the transitions shown in Figure 24 this is achieved by the laser. For the NV-centers a length of approx. 300 ns is enough. After this, the $\pi$-pulse is applied. Similar to cw ODMR the frequency is varied for the measurement.
Figure 28 shows the sequence of measurement steps in a) and a measured ODMR spectrum in b). The ODMR spectrum has three dips which correspond to the three nuclear spins of the $^{14}$N, so that the NV-center is identified as a $^{14}$NV-center.

### 3.5.2 Composite Pulses

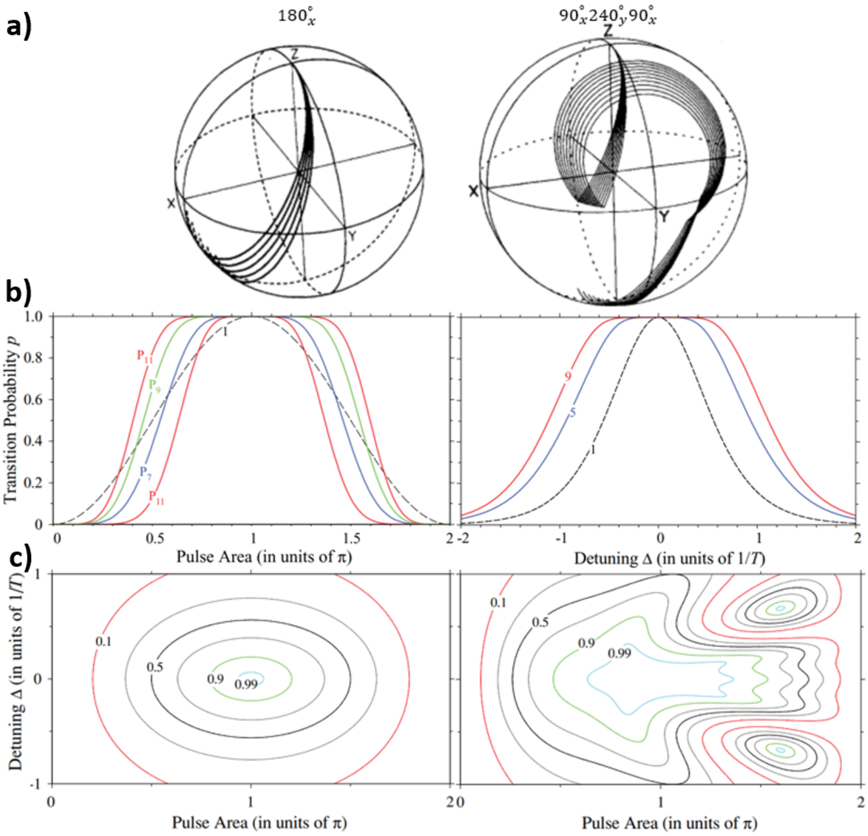


**Figure 29:** A $\pi$-pulse and a pulse sequence (consisting of two $\frac{\pi}{2}$ pulses and one $\frac{4\pi}{3}$ pulse) is shown in a). For the paths, different detunings $\delta$ were used. The path with the end point is visualized on a Bloch sphere in a)[48]. Probability of a transition to a different pulse area on the left and detuning on the right for different numbers of pulses in a pulse sequence is shown in b)[49]. Probability of a transition to different pulse areas and detunings simultaneously for one pulse (left) and for 5 pulses (right) is shown in c)[49].

In the previous chapter, it was described how to achieve a better resolution using a pulse to measure, for example, the nuclear spin and distinguish between the isotopes of nitrogen. With the same method, a $^{13}$C isotope can be detected as well as the distance it has to the NV-center[41]. But also instead of just one pulse multiple pulses can be used too. For example the $T_2$-time can be determined by using two

$\frac{\pi}{2}$-pulses.

Another possibility is to increase the stability of a transition. This is necessary because, for example, the $\pi$-time was determined from a previous measurement, which can deviate from the exact value due to unavoidable inaccuracies in the measurement process. Depending on how precisely the $\pi$-pulse has been determined, this can lead to significant errors when a large number of pulses are used. Therefore, one idea is to use a specific pulse sequence to improve the probability of being in a specific state even when measurement inaccuracies occur[49]. This can be shown using example pulse sequences on the Bloch sphere, which was presented earlier as shown in Figure 29 a). In this example two Bloch spheres are shown, one with a pulse sequence (on the right) and one without a pulse sequence (on the left). A $\pi$-pulse was then used for a transition which was applied with different detuning frequencies. It can be seen that when the exact resonance frequency is not used, it does not necessarily lead to a stable transition. When the same resonance frequency detuning is applied using a pulse sequence (in this case a combination of two $\frac{\pi}{2}$-pulses and one $\frac{4\pi}{3}$-pulse) the probability of being in the desired state is higher. Such pulse sequences are so called composite pulses.

Figure 29 b) shows the results achieved by using pulse sequences. There are three possibilities: stabilizing the excited state, stabilizing the ground state or stabilizing both states simultaneously. In Figure 29 b) the behavior for stabilizing the excited state can be seen. Here two parameters were varied: the detuning and the pulse area. Pulse sequences with up to 11 pulses were used. It can be seen that using more pulses leads to a higher stability of the states. In addition Figure 29 c) shows the combination of both measurement inaccuracies, for a single pulse on the left side and for a pulse sequence of 5 pulses on the right side.

## 3.6 Longitudinal Relaxation Time $T_1$

The longitudinal relaxation time $T_1$ can also be measured using a pulse. This corresponds to the decrease in lifetime for NV-centers population initialized in the ground state $m_s$=0[50]. This decay is caused by the interaction of the spin with the diamonds lattice. To determine this decay for NV-centers, two measurements are made. One of the measurements starts in the $m_s$=0 state and the second in the $m_s = \pm 1$ state which is achieved by the $\pi$-pulse determined from the Rabi-oscillation.

The pulse sequence is shown in Figure 30 a) and b). The time $\tau$ between the initialization of the respective state and the readout is changed in certain steps via the measuring process. The probability of being in the state $m_s$=0 is shown below the sequence. In these diagrams, it can be seen that after a certain period of time, it is no longer possible to distinguish which state the initialization was in. The diagrams c) and d) show the fluorescence plotted against time. The left diagram shows the total fluorescence of the measurement, once with $\pi$-pulse and once without and the right diagram shows the difference between the two measurements. By fitting an exponential function, the $T_1$ time can then be determined. For NV-centers it is typically in the range of milliseconds[50].

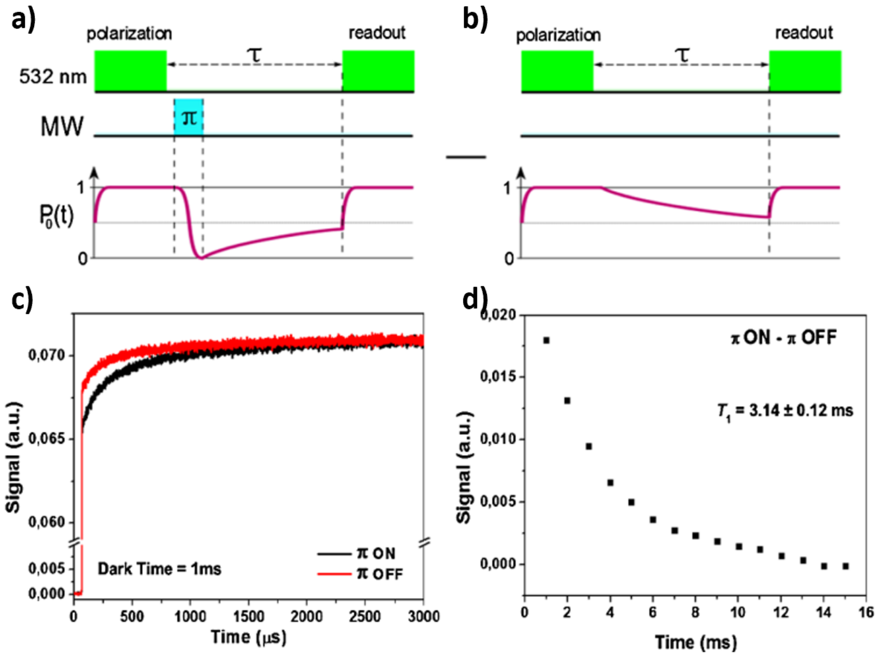


**Figure 30:** Pulse sequence for measuring the longitudinal relaxation time $T_1$ as well as the probability of being in the state $m_s$=1 both for the measurement with $\pi$-pulse in a) and without in b). Recorded fluorescence for the pulse sequence with $\pi$-pulse in black and without in red is shown in c) as well as the difference between the two measurements in d).[50].

# 4 Experiment

Before starting the development of a program that automates many measurement processes, it is necessary to first explain the experiment itself. In this chapter this will be done by demonstrating the setup of the experiment and then proceed to describe the measurement routine that is to be automated. The setup is separated in two parts: the laser and the confocal microscope as shown in Figure 31.

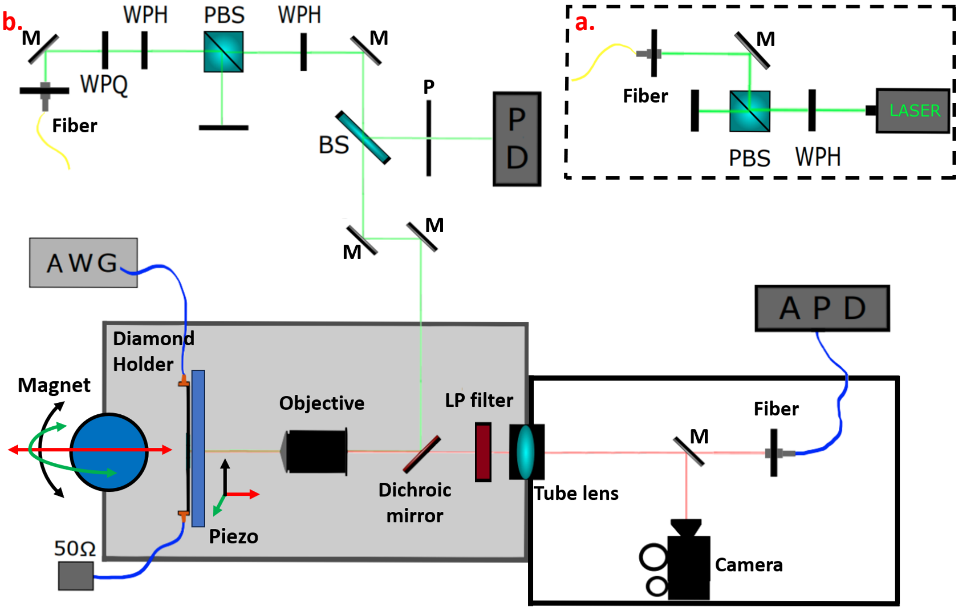


**Figure 31:** The structure of the experiment which has been divided into two sections: the laser in section a) and the confocal microscope in section b). Here the path of the laser beam (shown in green) as well as the fluorescence beam (shown in red) through the individual optical elements is shown.

The first part of the setup is shown in Figure 31 a), which is the upper right section. A pulsed laser with a wavelength of 515 nm is used to generate the laser beam. It then passes a half wave plate ($\frac{\lambda}{2}$-plate) (WPH) and a polarizing beam splitter (PBS) to adjust its intensity. The intensity could normally be adjusted in the program settings but this is not the case because at too low energy this can lead to fluctuations, which is why the power in the programm is usually kept at 100% and the intensity is decreased using a WPH. Afterwards, the beam enters the optical fiber and exits the first part of the experiment.

The second part of the setup is shown in Figure 31 b), involving the measurement and manipulation of the NV-centers in the diamond. The beam leaving the fiber passes through two $\frac{\lambda}{2}$-plates, a $\frac{\lambda}{4}$-plate (WPQ) and another polarizing beam splitter to adjust its polarization as the fiber is not polarization maintaining. Once the polarization is adjusted, the beam passes a beam splitter. This beam splitter reflects a small part of the beam which then hits a photodiode, allowing real-time measurement of the laser beams power during the experiment. The transmitted part of the beam is then going directly towards the diamond.

Before being focused on the NV-centers in the diamond via an objective lens, the beam passes a dichroic mirror. What is special about this mirror is, that it reflects a large part (between 90-95%) of smaller wavelengths like that of the laser while allowing larger wavelengths, such as the fluorescence of the NV-centers to

pass through.
The NV-centers in the diamond absorb photons from the laser beam and subsequently emit a photon with a higher wavelength. The emitted photons are then emitted in random directions. Some of these photons are captured by the objective lens, collimated and then goes directly towards the dichroic mirror. As previously described, these photons are transmitted by the dichroic mirror. The beam behind the dichroic mirror consists not only of fluorescence photons with a wavelength of 637 nm but also photons from the laser with a wavelength of 515 nm. These laser derived photons are then blocked by a long pass filter to 99.9999%, so that the measured beam consists almost entirely of fluorescence photons from the NV-centers. The tube lens then focuses the beam depending on whether the flip mirror is flipped or not either onto the fiber for the avalanche photodiode (APD) or onto the CCD camera. The CCD camera is primarily used to orientate on the surface of the diamond but it can also be used to visualize individual NV-centers. The fiber leads to an APD which counts the number of photons. It is worth mentioning, that a box was built around this part of the setup to further eliminate external factors.
Figure 31 b) shows two additional components. Firstly, the arbitrary waveform generator (AWG) which is connected to the diamond holder (PCB). This generator excites the energy levels of the spins in the electronic states. The second component is the magnet positioning system, which was set up and discussed in the context of my Bachelors thesis[39]. This system allows a precise alignment of the external magnetic field axis.

## 4.1 Measurement Routine

The main task of this masters thesis was to develop a program that automates the repetitive measurement routines. The software needed to be capable of detecting and classifying NV-centers in images. The typical measurement routine, which is normally performed by a human, can be seen in Figure 32.

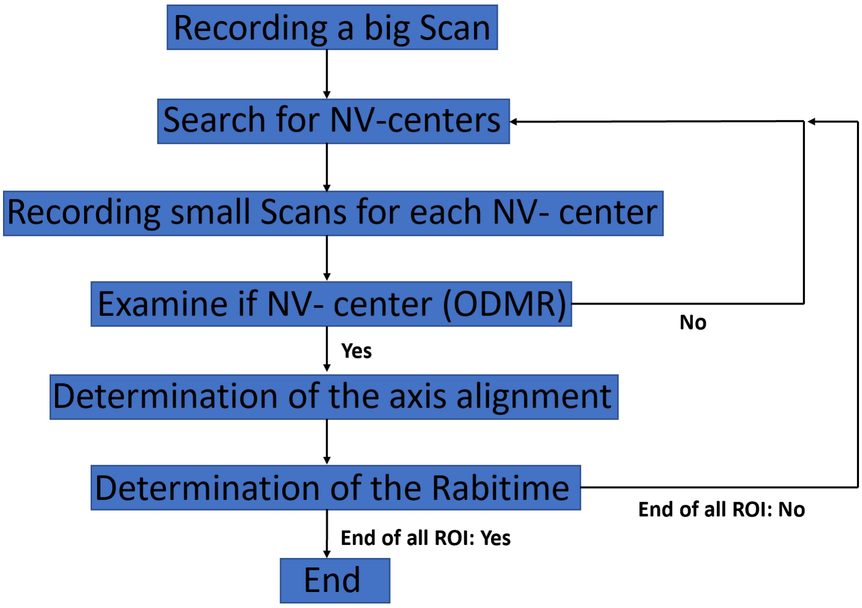

Recording a big Scan
Search for NV-centers
Recording small Scans for each NV- center
Examine if NV- center (ODMR)
No
Yes
Determination of the axis alignment
Determination of the Rabitime
End of all ROI: No
End of all ROI: Yes
End

**Figure 32:** The measurement routine used to characterize the ROI must be executed by the software.

Figure 32 shows a rough outline that is used to characterize the NV-centers. This outline serves as the roadmap that the software must follow. The first step will be the greatest challenge, because a program needs to to know what an NV-center is, what it looks like and how to recognize it. Additionally, the position and size of the NV-center must be determined. Machine learning is used to enable this, which learns from previously analyzed data to make highly accurate predictions. To reach this goal, the principle of neural networks is used. This allows a program to dynamically learn what NV-centers are. Chapter 6 goes into the structure of such a neural network. To predict the positions the neural network was expanded with another algorithm. The so called YOLO algorithm then provides the position and size of the Region of Interest (ROI).

After the objects have been recognized, they must be confirmed. This is done through the ODMR-spectroscopy mentioned in Chapter 3.2. A spectrum is recorded which should have a dip at around 2870 MHz for an NV-center if no external magnetic field is present. If this is the case, the ROI is further examined. If not, the next ROI is investigated as shown in Figure 32.

Once a NV-center has been confirmed, the next step is to determine its orientation. As shown in Chapter 3.1, there are four possible orientations for an NV-center in a diamond. To determine the orientation, an external magnetic field is applied. This magnetic field, as described in Chapter 3.3, affects the energy levels and thus the ODMR spectrum. The spin state $m_s = \pm 1$ is split. Since only the proportional component of the external magnetic field with respect to the NV-axis has an impact on the splitting of the spin states, the orientation can be determined through a certain angle selection.

For further measurements, such as for example composite pulses, the time of a $\pi$-pulse is required. This can then be determined by the software from the previously obtained data.

The subsequent measurements like composite pulses, $T_1$ time, $T_2$ time, etc. are not shown in Figure 32 because they do not belong to the characterization routine of individual NV-centers but rather to the subsequent measurements for specific pulse sequences.

# 5 Object detection

In the previous chapter, object detection and size classification were already mentioned as challenges for the software. To integrate this object detection into a program, multiple approaches were tested. Three different approaches were investigated: a classical non-dynamic variant and two dynamic variants involving two types of machine learning. These will be briefly discussed in the following.

## 5.1 Center Determination

To characterize NV-centers, they first need to be located. To achieve this, a large scan is made which has certain characteristics that can then be identified as NV-centers. These characteristics are typically circles with a diameter between 300 and 400 nm and a count rate ranging from 30 to 80. The simplest method is to raster the large scan, dividing it into smaller sections and evaluate them individually. During this process, all pixels below a certain threshold value can be set to zero beforehand so that only areas with increased counts are left. Figure 33 shows such an example section with a centered NV-center.

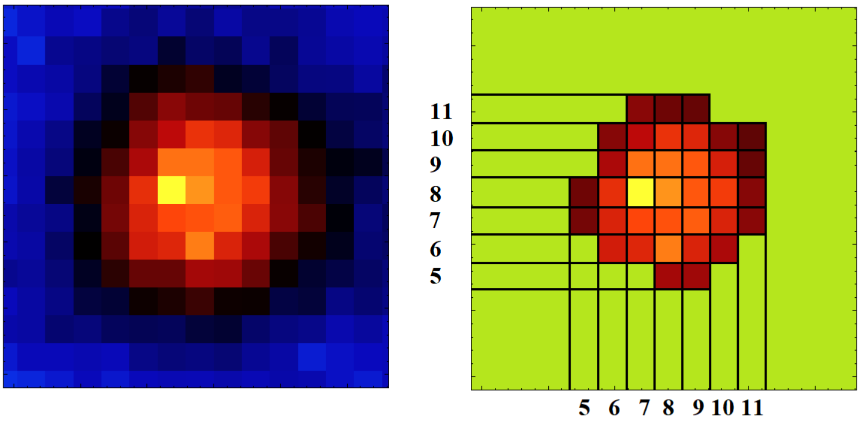


**Figure 33:** NV-center captured through confocal microscopy on the left side and with the background subtraction on the right side. Additionally, the pixel positions are indicated by lines.

To evaluate the section, the center must first be determined. It can be calculated using the following formula for $\bar{x}$:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N_x} i \sum_{j=1}^{N_y} \begin{cases} 1 & \text{if } z(i,j) \neq 0, \\ 0 & \text{else} \end{cases} \tag{43}$$

Where $N$ is the number of pixels that are greater than zero. For the section shown above, the center would be located at the following coordinates:

$$\bar{x} = \frac{1}{36}(2 \cdot 5 + 5 \cdot 6 + 6 \cdot 7 + 7 \cdot 8 + 7 \cdot 9 + 5 \cdot 10 + 4 \cdot 11) \approx 8.19$$
$$\bar{y} = \frac{1}{36}(2 \cdot 5 + 5 \cdot 6 + 7 \cdot 7 + 7 \cdot 8 + 6 \cdot 9 + 6 \cdot 10 + 3 \cdot 11) \approx 8.11.$$

The disadvantage of this method is that it can not verify whether the NV-center is round and determine its diameter. This can be checked through the number of squares (pixels), allowing larger objects to be excluded. But for that it is essential to consider whether the NV-center is positioned centrally or at the edge of the scan, which could result in only a part of the NV-center being captured in the scan which then significantly affects the number of squares. Another problem arises when two NV-centers are located in the same section. In this case the center would not be correctly aligned with either of them, which than potentially leads to the objects exclusion due to the high pixel count. Additionally, determining the background in the presence of double NV-centers or NV-centers near an edge or impurity spot would be challenging. Each of these exceptions would require specific code in the program. Moreover, new errors that occur would need to be solved with additional lines of code, consuming a lot of time. So the program would be non-dynamic.

## 5.2 Neural Networks

To address the disadvantages discussed in Chapter 5.1, a completely new approach was used. The static program was transformed into a dynamic one which can be improved without updating its code and it can even learn and adapt without human intervention. This program autonomously learn from its mistakes and continually improve itself to make better predictions. This can be achieved by using the field of machine learning or in general neural networks. In the next chapter such a neural network is presented and a method for solving the object recognition problem, using machine learning, is shown and evaluated. Two approaches with different settings will be discussed and explained in Chapter 6.3.1 and 6.3.2.

# 6 Scan of ROI Anomaly

For the object detection of NV-centers, machine learning based on neural networks is used. To achieve this, an AI system consisting of an input generator (preprocessor), a backbone and a header (postprocessor) was built. Different versions with different training parameters were experimented with and evaluated in order to obtain the best architecture with the highest accuracy in object detection. The final configuration can be found in the last subsection of this chapter.

## 6.1 Network Architecture

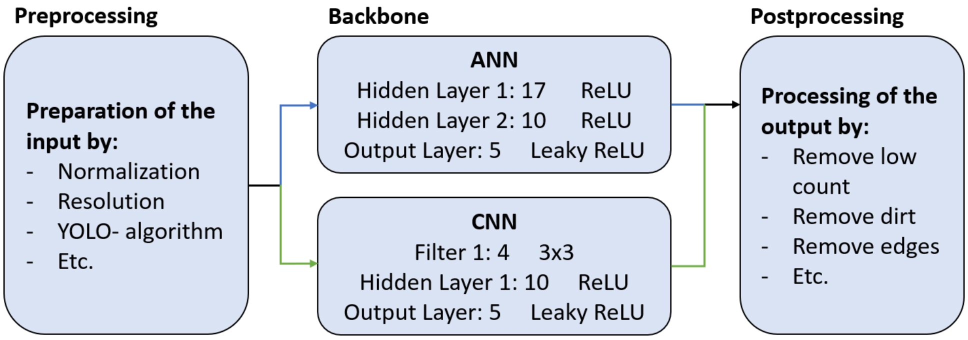


**Figure 34:** Architecture of the neural network used for object detection. The architecture consists of three components: the preprocessing, the backbone and the postprocessing.

Before starting with the individual subchapters in which the components of the final AI are explained and examined, the architecture of the neural network and the training dataset will first be introduced. The architecture is shown in Figure 34. The first part of the architecture describes the preparation of the input. The captured scans may have different resolutions, sizes or intensity ranges depending on the measurement settings. Since it is challenging to train the network with all these settings, the input is standardized to a fixed size, resolution and intensity range. The output after the preprocessor then serves as the input to the backbone. The backbone consists of the trained neural network. In Figure 34 it is divided into two parts, symbolizing that the architecture has been tested with both, a simple ANN and a CNN. The output of the backbone is then further processed in the postprocessing stage to filter out conspicuous ROIs. The output after postprocessing forms the final predictions of the AI.
As mentioned in Chapter 2.5, the dataset is divided into a training, a validation and a test dataset. Table 3 shows the number of NV-centers in each dataset. The exact generation of the individual data is discussed later in the chapter 8.2.1.

**Table 3:** Division of the data set into a training, validation and test dataset. The number indicates the number of NV-centers in a dataset.

| Training dataset | Validation dataset | Test dataset |
|---|---|---|
| 1000 | 104 | 70 |

## 6.2 Preprocessing

To evaluate scans for NV-centers, they need to be standardized in the first step to make it easier for the network to identify NV-centers. These various standards are briefly mentioned below and in some cases illustrated with images.

**Size:** Since the number of input nodes in a neural network is fixed, the input size needs to be standardized. Various sizes were experimented with and it was finally decided to use an input size of 5x5 pixels, which leads to 25 input values. This relatively small size offers several advantages. Firstly, it allows the network to be trained quickly and secondly, since an NV center can be on several sections, it leads to multiple trainingdata to be generated from a single NV-center as explained in Chapter 8.2.1. This reduces the number of NV-centers that need to be captured without reducing the amount of training data. A disadvantage is, that for a large scan the time it takes for the network to evaluate the scan completely is higher because the network needs to be processed more often.
To achieve a standardized size for scans, that are larger than 5x5 pixels, a slightly modified YOLO algorithm described in Chapter 2.4 is used.

**Resolution:** In the confocal microscopy for the experiment, the resolution and pixel size can be freely varied between 25 nm and 200 nm. As a result, the 5x5 pixel part can appear very different. An NV-center can have a size of either 15 pixels at higher resolution or only 3 pixels at lower resolution. This variability needs to be taken into account in the network. Therefore, the network would need to be trained for different resolutions which is possible but would require a hight amount of effort since it would require a much larger number of training data. One way to solve this problem is through artificial upscaling of scans. For example, if a scan has a resolution of 100 nm per pixel, an additional pixel is introduced between two pixels, which is created using the average of the neighboring pixels. This doubles the resolution and with it almost the number of pixels. Figure 35 shows such an upscaling process.

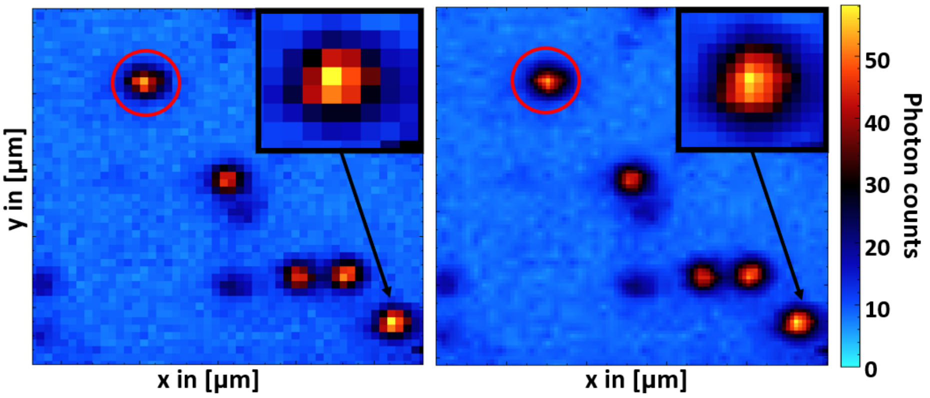


**Figure 35:** Increasing the resolution by inserting a new pixel between two pixels. The resolution goes from 100 nm per pixel (left) to 50 nm per pixel (right).

On the left side, the resolution is set to 100 nm per pixel while on the right side, it is reduced to 50 nm per pixel. To better show the difference, a NV-center has been enlarged in the upper corner of both scans. It is shown from this that the increased scaling achieves the desired result for most NV-centers. But there are also NV-centers like the one in the upper left circle that appear wider due to the upscaling. This problem can be addressed by training the neural network with such upscaling data, making it more robust and resistant to such inaccuracies.

**Normalization:** Analogous to the resolution, the intensity range also depends on the experiment. Both, the power of the laser beam and how well it is coupled into the fiber can vary. As a result NV-centers may appear with intensities ranging from 40 kcounts to 80 kcounts or even more or less in the scan. To solve this problem, the scan is normalized between 0 and 1. This has also the benefit of reducing the scale of values that the neural network needs to calculate with and thereby reducing the time required for each pass.

### 6.2.1 YOLO Algorithm

To normalize the size of the input scan, the YOLO algorithm is used. In the ANN and CNN, which will be explained in the following chapters, the image is divided into sections twice. In the first step, the image is divided into 20x20 pixel sections and in the second step, these 20x20 pixel sections are further subdivided into 16 5x5 pixel sections. These 5x5 pixel sections form the input for the ANN or CNN. The reason for this two segmentations are impurities or edges in the scan. These areas have significantly higher kcount rates than NV-centers, causing the average value of the entire scan to be much higher than the background should be. After normalization, some NV-centers in Figure 36 (left) might get lost in the noise. To solve this problem only the 20x20 pixel sections are normalized and not the entire scan. This normalization constant is then used for the 16 5x5 pixel sections within the 20x20 pixel section.

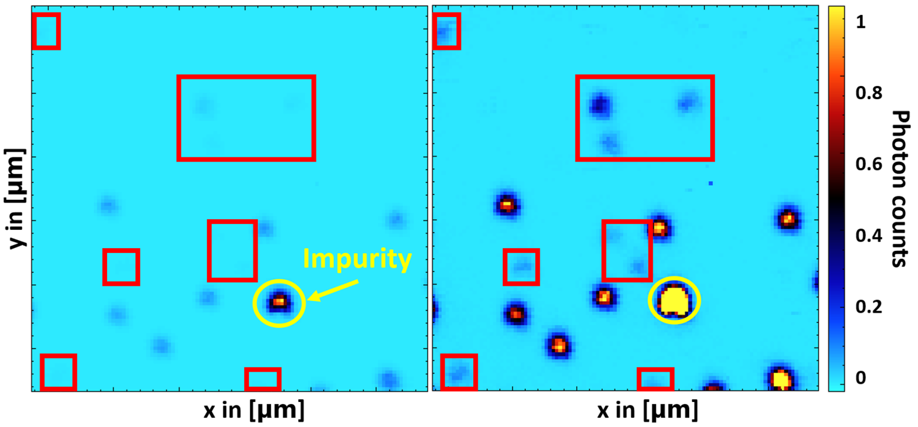


**Figure 36:** NV-center close to an impurity. The normalized measurement data for the respective x and y positions form the colormap. On the left, normalization including the impurity is shown and on the right, normalization with the principle of the two segmentations is shown.

Another solution to solve the problem would be to cap values above a certain kcount rate or in other words to set a maximum value and any values higher than this maximum would be set to that limit. But as described above, the number of kcounts can vary, making it challenging to choose a general threshold value. This is why the first method was chosen.
It is important to note, that the problem exists not only for the AI but also for the experimenter as seen in Figure 36 (left). In this scan it is also challenging for the experimenter to locate NV-centers. To solve this in the experiment control software,

the values for the colormap could be capped allowing the experimenter to identify more NV-centers in the noise.
Applying the method, as seen in the right part of the Figure 36, simplifies the process for the AI in a similar way to the experimenter. Scanning the image by forming multiple segmentations resulted in the discovery of 9 additional NV-centers in this image.

## 6.3 Backbone

After the scan has been normalized it enters the neural network. Two types of neural networks were tested here. On one hand, the simplest architecture an ANN and on the other hand, a slightly more complex variation which is normaly used in image recognition a CNN was used. Both networks were trained with bias and without bias. The results, as well as the exact structure, will be explained in this chapter. In addition the results of both neural networks will be compared and a final architecture for the backbone will be chosen.

### 6.3.1 Artificial Neural Network

The first neural network that could be used for the backbone is a simple ANN. In this chapter two different variations of this network are introduced and evaluated. The following sections will show and explain the structure, the training process and the output of such a network.

**Structure of the ANN:** As described in the fundamentals, a neural network can take on various forms. This includes the choice of activation functions, the number of neurons in a hidden layer, the number of hidden layers, the error function, whether to use bias or not, etc.. Since most of these variations resulted in only minor differences this masters thesis focused on the aspect that made the biggest difference: Network architectures with and without bias. In addition, two different training processes, the SGD and MBGD, were used and compared for each of these variations. The initial weights were distributed around zero, using a normal distribution with the condition that the value zero could not be used as a weight. The reason for using a normal distribution is to ensure that the sum of all neurons in a layer does not become too large especially when the output values, for example, represent probabilities ranging from 0 to 1. This approach also helps to save computational time.
The final values of the neural network are listed below:

**Table 4:** Architecture of the tested ANN.

| | |
|---|---|
| Number of hidden layers: | 2 |
| Number of neurons in the first hidden layer: | 17 |
| Activation function in the first hidden layer: | ReLU |
| Number of neurons in the second hidden layer: | 10 |
| Activation function in the second hidden layer: | ReLU |
| Number of neurons in the output Layer: | 5 |
| Activation function in the output Layer: | Leaky ReLU |

**Training process of the ANN:** In the next step, the neural network mentioned above is trained. For this purpose, training data with various NV-centers are created and characterized by their positions and sizes. The total training dataset consisted of about 1000 images containing parts of NV-centers. The network is trained using the SGD and MBGD methods, which are described in Chapter 2.2.2 while the BGD method is not used due to the large size of the training dataset, as this method is more suitable for smaller datasets. As part of the training process each training iteration consisted of 50 epochs for SGD and 50 epochs with 40 images per batch for MBGD. This process was repeated 1000 times. After each iteration the loss was recorded to better document the training process. The following Figure 37 shows this training process for an ANN with bias (top) and without bias (bottom).

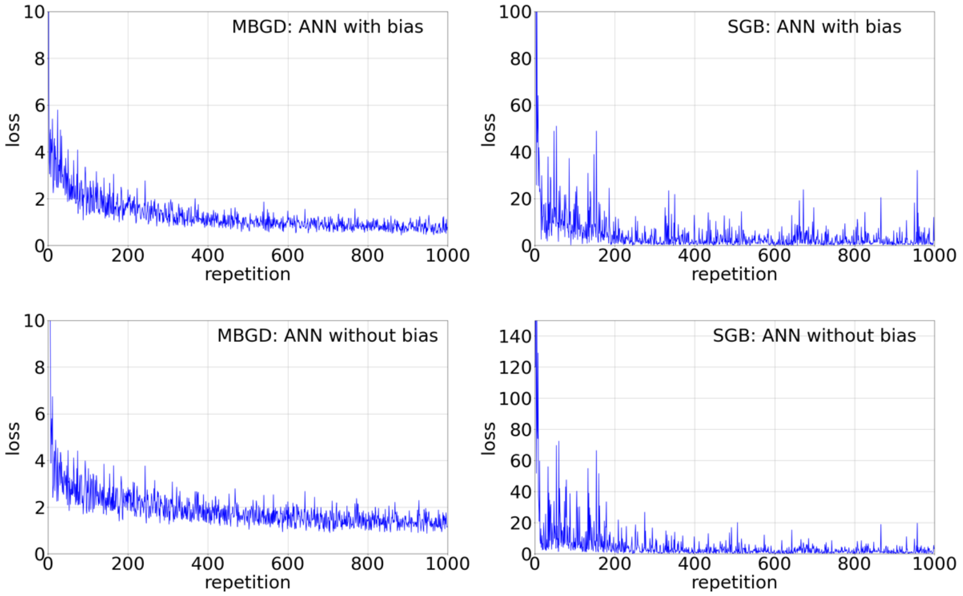


**Figure 37:** The training process of an ANN with bias (top) and without bias (bottom) using the MBGD method (left) and SGD method (right) is shown in the subfigures. The loss is determined after each training iteration. Both training methods used 50 epochs and the batch size for MBGD was 40 images.

Figure 37 shows the progress of the training processes for the SGD and MBGD methods. To get a better comparison of the loss between SGD and MBGD, the results of SGD were multiplied by 40. It should be noted, that the SGD training process considered 1 000 random images while the MBGD process, due to batches, consisting of 40 images, considered 40 000 random images. It can be seen that even though fewer images were used in the SGD method the loss is often smaller than in the MBGD method. This is expected as the loss is calculated for a single image and not for the average of 40 images. The network computes the values to match only the corresponding single image and not 40 images simultaneously. The downside of this method is, that when focusing on a single image, other images that may show the opposite result do not have good results in the form of a low loss. This can be seen in the graph as individual outliers which occur more frequently in the SGD method compared to the MBGD method.

To compare the results of the training processes, the NV-centers can be predicted using an example image. In the case of an image with no NV-centers all four methods correctly recognized this non event (TN). Figure 38 shows the data output of the ANN if the image has a NV-center.

As an example, a randomly selected NV-center of the test dataset, was chosen and the AIs task was to recognize this NV-center and predict its position and size. Different training methods were used independently. The top images in Figure 38 show the results of the ANN with bias, while the bottom images show the results of the

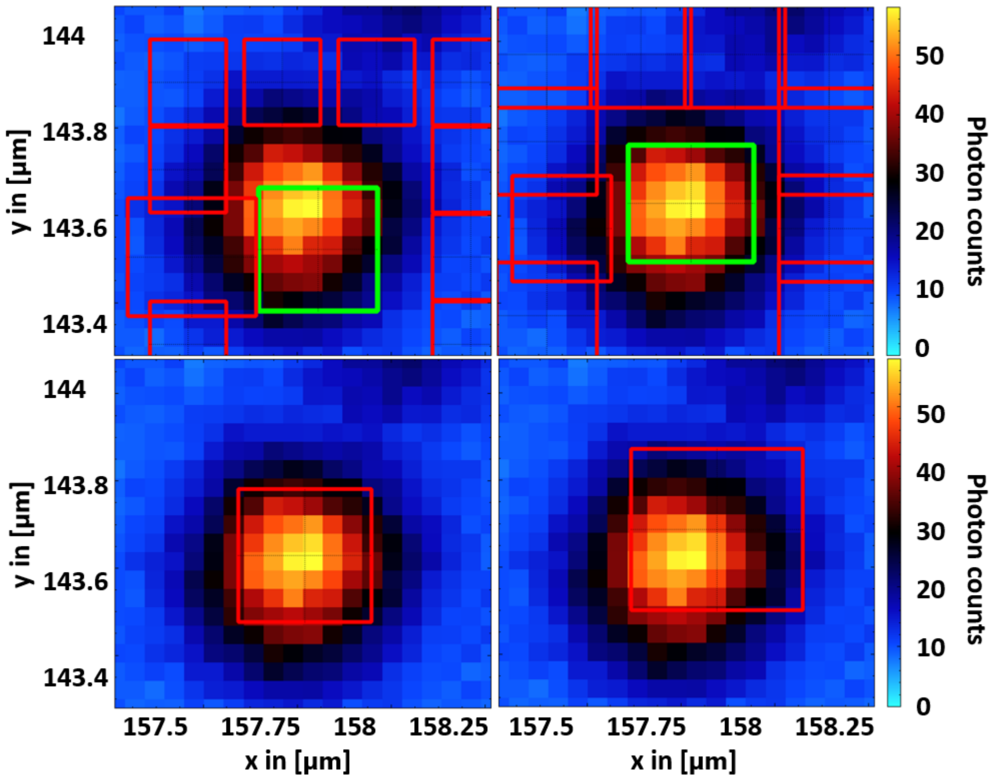


**Figure 38:** The positions and sizes of the NV-centers, predicted by the AI through the ANN, are shown in the images. The top images show the ANN with bias while the bottom images show the ANN without bias. The left images represent the MBGD training method and the right images represent the SGD training method. The red or green rectangles indicate the position and size of the predicted NV-center.

ANN without bias. It can be seen from this example that small variations in the neural network can lead to significantly different results.

The reason can be found in the training data. The output layer consists of 5 neurons as mentioned earlier: one neuron for the probability of whether there is an NV-center or not, two neurons for the x and y positions and two neurons for the size of the rectangle in the x and y directions. In the training data the first value is set to 1 for an NV-center so the ANN tries to bring this value closer to 1 to achieve a smaller loss. Since the weights also influence the other neurons, which always have different values, the ANN sets the bias value for the first neuron approximately to 1. This results in the ANN predicting multiple NV-centers in the image (visible in the top images in Figure 38) as the bias has more influence on the probability than the sum of the weights. Looking at the values for the bias (in the file) can confirm this.

In the case of the ANN without bias, this behavior does not occur so the output of the first neuron in the output layer must be determined through the weights. As a result the bottom images in Figure 38 show only one rectangle with good prediction values. The values for the position and size are listed in the following Table 5.

As mentioned earlier, the comparison between the SGD training method and the MBGD training method was achieved by simply multiplying the loss of the SGD by 40. The key difference between the methods is that one used 1 000 training data repetitions while the other used 40 000 training data repetitions. To better compare

**Table 5:** Predictions of the ANN for the position and size of the NV-center. For the upper images from Figure 38 the green rectangle was chosen as the rectangle.

| | position (x/ y) [$\mu$m] | size (height/ width) [nm] |
|---|---|---|
| ANN with Bias (MBGD): | (157.844/143.522) | (252.537/281.922) |
| ANN with Bias (SGB): | (157.824/143.628) | (267.043/267.984) |
| ANN without Bias (MBGD): | (157.816/143.629) | (283.204/304.894) |
| ANN without Bias (SGB): | (157.877/143.686) | (365.695/371.054) |

the methods, additional training repetitions are made so that the SGD method has also used 40 000 training data repetitions. Figure 39 shows the result of a prediction between the SGD method with 1 000 data repetitions (left) and the SGD method with 40 000 data repetitions (right).

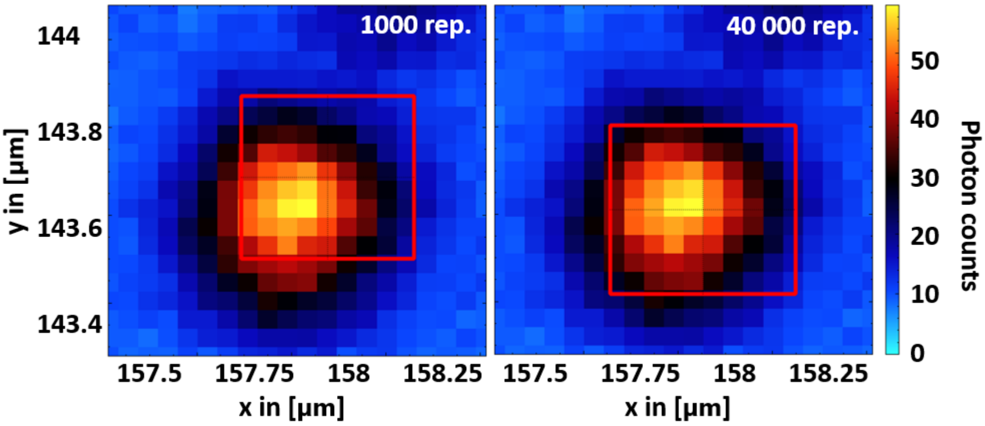


**Figure 39:** The prediction of the ANN using the SGD training method after 1 000 repetitions (left) and 40 000 repetitions (right). The red rectangle indicates the predicted position and size of the NV-center.

In Figure 39, the prediction of the ANN for the position and size of the NV-center is shown. It can be seen that the difference for the prediction values between the MBGD and SGD with the training method in Figure 38 is smaller. The difference in the x position is less than 40 nm and the y position is less than 20 nm. Only the size of the ROI deviates from the MBGD values. The values are displayed below.

**Table 6:** Predictions of the ANN for the position and size of the NV-center using 40 000 repetitions.

| | position (x/ y) [$\mu$m] | size (height/ width) [nm] |
|---|---|---|
| ANN without Bias (SGB): | (157.853/143.610) | (392.434/386.325) |

The comparison of the results from Figure 38 and 39 shows that in this case the ANN method without bias, trained through MBGD, provides the best results for these sample data. To further support the model, more images of NV-centers, of the test dataset, with varying resolutions or different numbers of NV-centers in a single image, were taken. Some of these are shown in the following Figure 40.
It is shown that the ANN method without bias is capable of detecting NV-centers in a recorded measurement. In the sample images from Figure 40, it can be seen that on the one hand, all NV-centers have been detected but on the other hand,

many false positives are also present which predict the detection of objects that are not NV-centers. To solve this problem, predefined characteristics in the code such as the size of rectangles or a maximum of counts could be used to reduce the number of false positives. But this would introduce a static component to the dynamic process.

Therefore, the ANN method is extended with an additional component: filters. By using filters, the neural network can search for specific features in the measurement data and highlight them. This additional component is part of the CNN discussed in the next chapter.

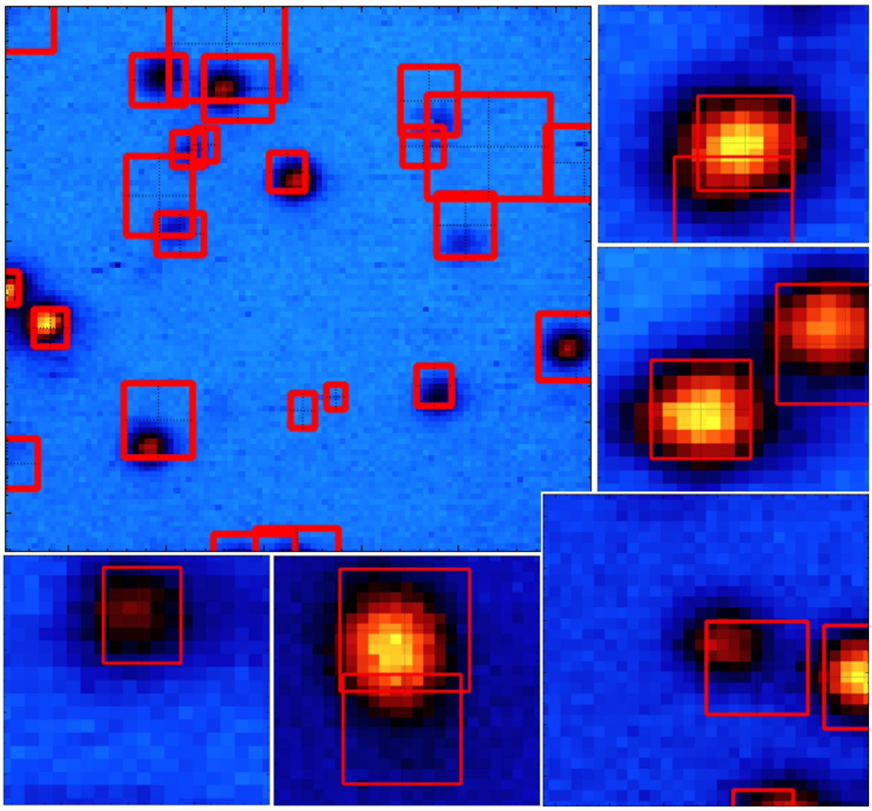


**Figure 40:** Predictions of the network for the position and size of the NV-centers. Here, the training method of MBGD with the ANN without bias is used.

### 6.3.2 Convolutional Neural Network

In the previous chapters 5.1 and 6.3.1, two different approaches for detecting NV-centers are shown. Chapter 5.1 focused on a static method, while Chapter 6.3.1 introduced a dynamic method. Both methods are capable of detecting NV-centers and predicting their positions. But the first method struggles with the detection of closely spaced NV-centers while the second method has problems in predicting NV-centers where none are actually present.

One way to solve the problem of the ANN is by extending it with filters which acts on the input data. This approach is known as CNN.

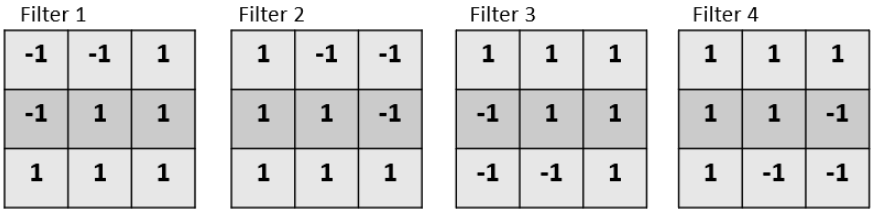
**Structure of the CNN:** The CNN consits of a characterization, which is a standard ANN, as described in the previous chapter and a feature extraction. This

feature extraction consists of multiple filters that also can be dynamically changed during the training process. The training process of these filters is integrated directly into the functions of the SGD and MBGD. In this chapter, the focus is primarily on the most significant difference which is also the bias.
The final parameters of the structure are listed in the following:

**Table 7:** Architecture of the tested CNN.

| | |
|---|---|
| Number of hidden layers: | 1 |
| Number of neurons in the hidden layer: | 10 |
| Activation function in the hidden layer: | ReLU |
| Number of neurons in the output layer: | 5 |
| Activation function in the output layer: | Leaky ReLU |
| Number of filters: | 4 |
| Size of the filters: | 3x3 |

The ANN is created as described in the previous chapter, using a normal distribution centered around zero without using the value zero. The filters can be created following the same principle or can be initialized by the programmer. In this example, the filters were initialized to observe how the network changes the values of the filters. The initialized filters have the following form:



Filter 1

| | | |
|---|---|---|
| -1 | -1 | 1 |
| -1 | 1 | 1 |
| 1 | 1 | 1 |

Filter 2

| | | |
|---|---|---|
| 1 | -1 | -1 |
| 1 | 1 | -1 |
| 1 | 1 | 1 |

Filter 3

| | | |
|---|---|---|
| 1 | 1 | 1 |
| -1 | 1 | 1 |
| -1 | -1 | 1 |

Filter 4

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | -1 |
| 1 | -1 | -1 |

**Figure 41:** Initialized parameters for the filters of the CNN.

**Training process of the CNN:** In the next step, the CNN was trained using the same training dataset as the previously presented ANN. Again, the training methods of SGD and MBGD were used. The number of epochs was set to 50 and in the case of MBGD 40 images per package were chosen again. Similar to the ANN shown in the previous chapter each training process was repeated 1000 times. After each step, the loss was calculated to better document the training process. The loss was then plotted against the repetition as shown in the following Figure 42.
Figure 42 shows four subplots: two for a CNN with bias (top) and two for a CNN without bias (bottom). The left subplots show the training process of MBGD while the right subplots show the training process of SGD. To get a better comparison of the loss between the training methods, the values of SGD have been multiplied by 40. It can be seen, that analogous to the ANN, the loss of SGD is often smaller than that of MBGD. But MBGD provides a constant loss value whereas the SGD method has many outliers. Generally, these outliers cause the average loss of SGD to be higher compared to the average loss of MBGD. The reason is, that the CNN examines only one image at a time similar to the ANN training process. The CNN adjusts the weights and bias, as well as the filter, either for a batch size of 40 images repeated 50 times each epoch or for just one image repeated 50 times each epoch. While the subplots on the bottom and top right look identical to those of the ANN in Figure 37, the subplot on the top left looks significantly different. The subplot at the top left first stays reasonably constant before the loss decreases as expected.

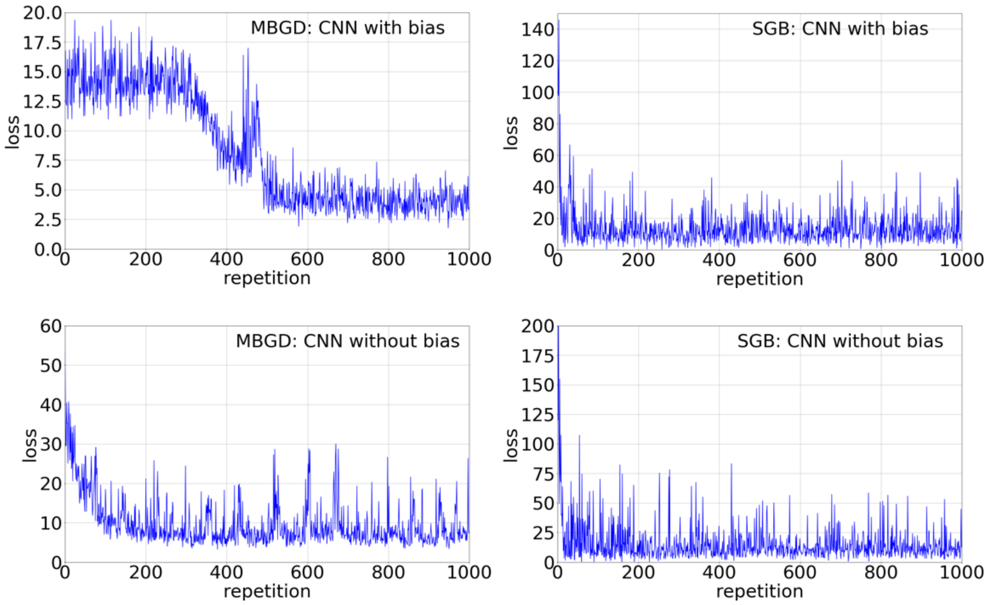


**Figure 42:** Training process of a CNN with bias (top) and without bias (bottom) using the methode of MBGD (left) and SGD (right). The loss was determined after each training iteration. The number of epochs was set to 50 for both training methods and the number of images per package in MBGD was 40.

In addition, there is also a region around 480 repetitions where it increases sharply. This can be explained by the fact that some training data had significant variations causing the weights, bias and filter to be trained in such a way that the error is increasing again for subsequent repetitions. Another explanation can be found in Figure 4 in chapter 2.2.1. In this example, it is possible that the CNN was first in a local minimum during the training process but then left it through further repetitions and navigated to a deeper local minimum. The first phase, at which the loss is constant, can also be explained by the training process not finding a minimum. This training process could be improved by using a higher learning rate, which was set at 0.01 in all examples. But since the weights were randomly distributed and random training data was used, the process can not be repeated to see, if a higher learning rate would improve it as it would likely converge to a different minimum. To show the outcome of the training process, predictions were tested using sample data analogous to the ANN. The task for the network was to recognize a NV-center and predict its position and size. The predictions are shown in the following Figure 43. The top images in Figure 43 represent the CNN with bias, while the bottom images represent the CNN without bias. The left images were trained using the MBGD and the right images were trained using the SGD. It can be seen that the difference between the training methods is much bigger in the case of the CNN, compared to the ANN. In this case, the network correctly identified the NV-center in only three out of four cases. The problem with the bias, which was already figured out in the ANN, has increased, which can be seen in the top left image. In each square, the network makes identical predictions whether an NV-center exists or not. In the case where the CNN has a bias and was trained using the MBGD method, the AI at least recognizes the position of the NV-center. But this network architecture has similar problems by predicting NV-centers in almost every other square as well. In the case where the CNN has no bias, both training methods lead to accurate position predictions but the network predictions of size differ from each other. The predicted position and size of the NV-center are listed in the following table 8. For the CNN with bias, which was trained with the MBGD method, the values from the green rectangle were taken.

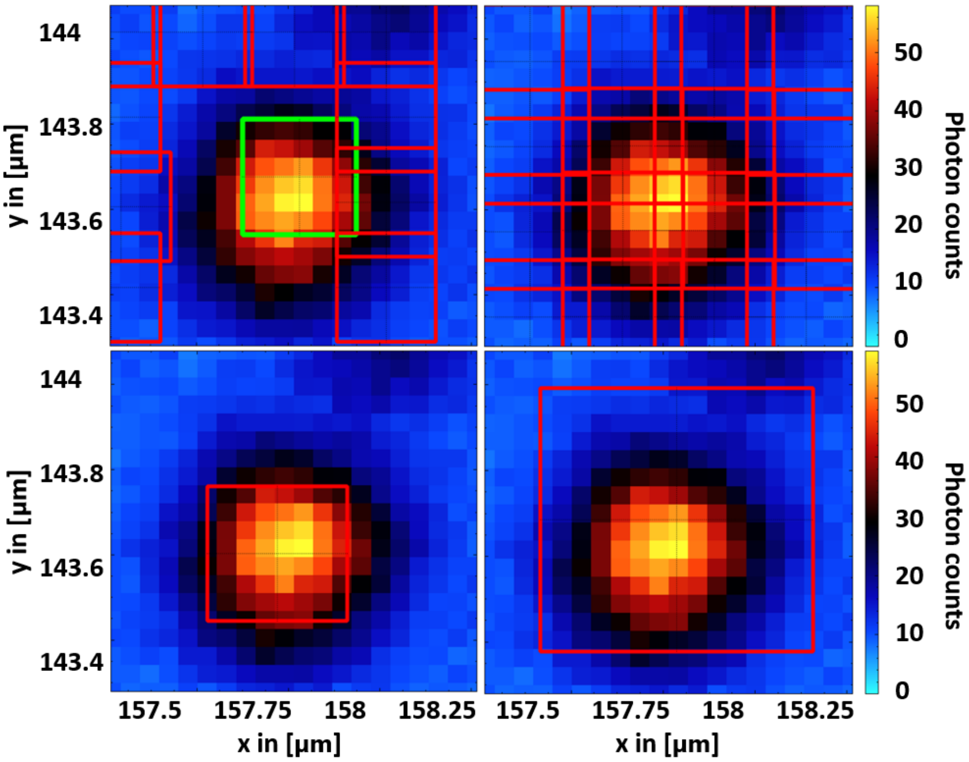


**Figure 43:** The positions and sizes of the NV-centers predicted by the AI through the CNN are shown in the images. The top images show the CNN with bias while the bottom images show the CNN without bias. The left images represent the MBGD training method and the right images represent the SGD training method. The red or green rectangles indicate the position and size of the predicted NV-center.

**Table 8:** Predictions of the CNN for the position and size of the NV-center. For the upper images from Figure 43 the green rectangle was chosen as the rectangle.

| | position (x/ y) [$\mu$m] | size (height/ width) [nm] |
|---|---|---|
| CNN with Bias (MBGD): | (157.825/143.676) | (247.883/271.719) |
| CNN without Bias (MBGD): | (157.776/143.603) | (305.707/316.129) |
| CNN with Bias (SGB): | - | - |
| CNN without Bias (SGB): | (157.829/143.685) | (593.111/614.993) |

It is important to note in this comparison that the MBGD training method used 40 000 images, while the SGD method only used 1 000 images. To get a more meaningful comparison, the training process for the SGD method is continued until it also used 40 000 images for the training. Figure 44 shows the network predictions after 1 000 repetitions (left) and after 40 000 repetitions (right).

In Figure 44, it can be seen that the predicted NV-centers size is smaller after 40 000 repetitions compared to 1 000 repetitions. The position has shifted approximately 70 nm in the x direction and approximately 40 nm in the y direction. The values of

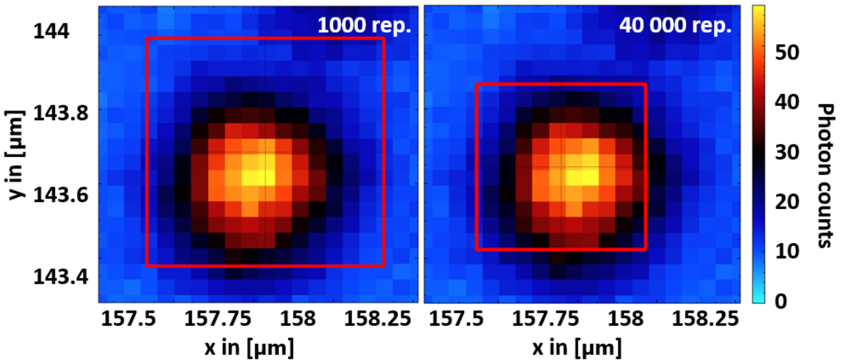


**Figure 44:** The prediction of the CNN using the SGD training method after 1 000 repetitions (left) and 40 000 repetitions (right). The red rectangle indicates the estimated position and size of the NV-center.

the rectangle are listed in Table 9.

**Table 9:** Predictions of the CNN for the position and size of the NV-center using 40 000 repetitions.

| | position (x/ y) [$\mu$m] | size (height/ width) [nm] |
|---|---|---|
| CNN without Bias (SGB): | (157.759/143.645) | (427.778/444.444) |

The comparison of the different training methods and the presence of bias was demonstrated in Figures 43 and 44. It can be seen that the MBGD training method for a neural network without bias leads to the best results. To further confirm this, additional sample images of the test dataset were used. Some of these examples are shown in Figure 45. It can be seen that the CNN makes very good predictions without bias. Figure 45 demonstrates that all NV-centers, except for one, have been found and only one NV-center was predicted in cases where none exists. Also it can be seen that in this example, there is no need for further static modifications of the code as it would have been necessary for the ANN presented in the previous chapter. The training process is entirely dynamic and does not require more code to achieve better results. Moreover, the position and size of the NV-center are also predicted to be near the center. As a result it is now possible to implement an autofocus, using the network, which will be shown in Chapter 8.3.1.
What is also demonstrated in the figure is that the CNN is able to detecting NV-centers with different kcounts. It does not matter whether they have few or many kcounts compared to others. For example, in Figure 45 at the top left, it can be seen that the detected NV-centers have different numbers of measured kcounts. The CNN successfully identified a weakly emitting NV-center, where the height is not correctly focused, right next to a strongly emitting NV-center.

**Filter of the CNN:** The start filters, as described earlier, were not generated through a normal distribution but were initialized manually, like shown in the structure of the CNN. This was done to get a better understanding of how the filters change during the training process. The goal was to identify the features, the network is looking for in the images. To achieve this, the filters used by the network to detect NV-centers have been applied to the sample image. The results are shown in the following Figures 46 to 49 along with the corresponding filters for different CNN architectures and their training methods. The filter values were rounded to

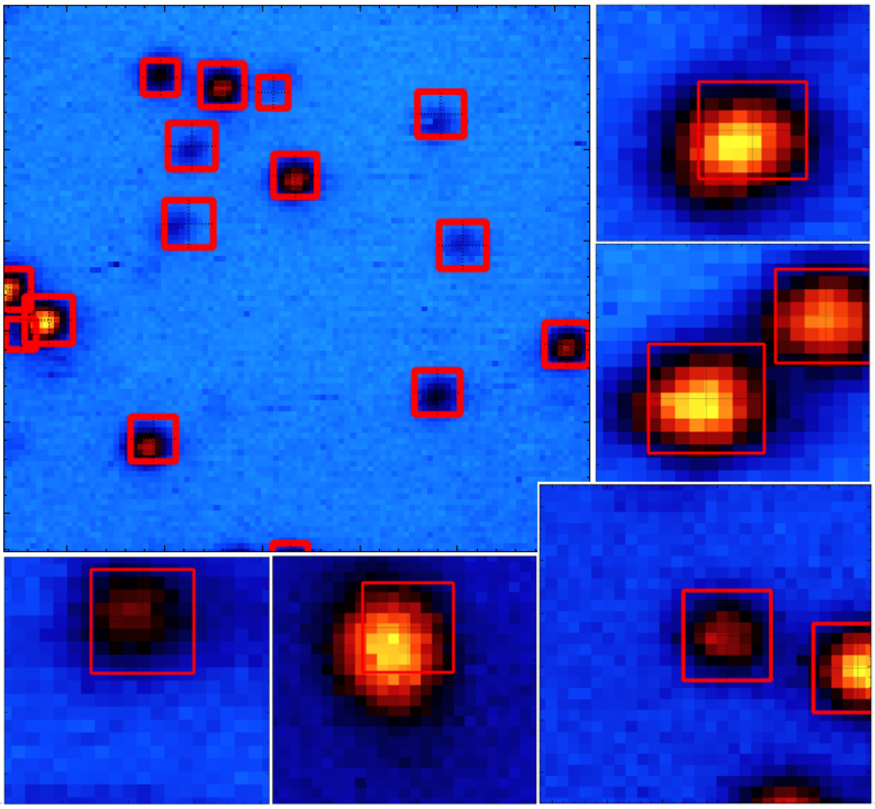
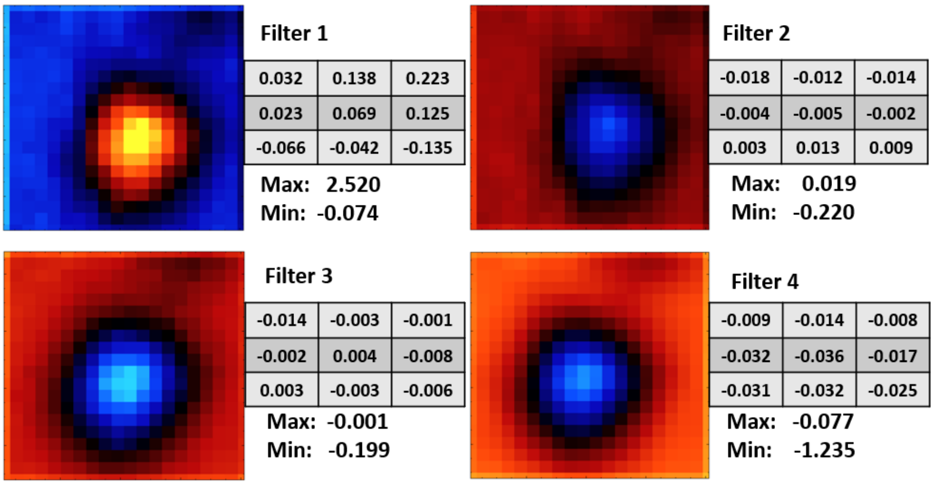


**Figure 45:** Predictions of the network for the position and size of the NV-centers. Therfore, the training method of MBGD with the CNN without bias is used.

three decimal places to get a better overview.
The figures show that the bias makes a big difference in the computed filters. Figures 46 and 47 describe the CNN with bias while Figures 48 and 49 describe the CNN without bias. It can bee seen that a presence of a bias makes the network to look for dips rather than peaks. Additionally, in this case most elements of the filters are negative. On the other hand, when the network has no bias it looks for peaks instead of dips and most elements of the filters are positive. Another difference between having a bias and having no bias is the range between the highest and lowest output value. In the absence of bias, the maximum and minimum values are usually closer together and are mostly negative whereas when a bias is present the difference between the maximum and minimum values is larger and the values are mostly in the positive range.

The fact that the network searches with its filters for dips, rather than peaks, can explain why the network with bias leads to poorer results. These filters result in almost all elements being multiplied by negative numbers, which leads to the image having mostly negative values after applying the filters as shown in Figures 46 and 47. The maximum value of the output image is always in the negative range except for filters that search for peaks (Figure 46 Filter 1 and Figure 47 Filter 2). These filters may be responsible for interpreting sections without NV-centers as NV-centers. In Figure 43 (top left) it can be seen from the green rectangle that the existing NV-center was correctly predicted in terms of its position and size. This could be the result of Filter 1 in Figure 46 which has the largest range between



**Filter 1**

| | | |
|---|---|---|
| 0.032 | 0.138 | 0.223 |
| 0.023 | 0.069 | 0.125 |
| -0.066 | -0.042 | -0.135 |

Max: 2.520
Min: -0.074

**Filter 2**

| | | |
|---|---|---|
| -0.018 | -0.012 | -0.014 |
| -0.004 | -0.005 | -0.002 |
| 0.003 | 0.013 | 0.009 |

Max: 0.019
Min: -0.220

**Filter 3**

| | | |
|---|---|---|
| -0.014 | -0.003 | -0.001 |
| -0.002 | 0.004 | -0.008 |
| 0.003 | -0.003 | -0.006 |

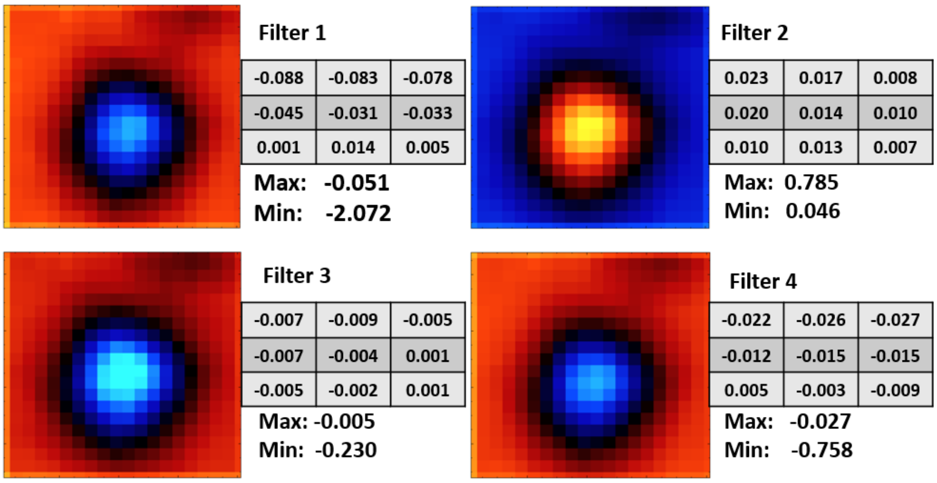
Max: -0.001
Min: -0.199

**Filter 4**

| | | |
|---|---|---|
| -0.009 | -0.014 | -0.008 |
| -0.032 | -0.036 | -0.017 |
| -0.031 | -0.032 | -0.025 |

Max: -0.077
Min: -1.235

**Figure 46:** The filters of the CNN with bias, determined through the MBGD training process, were applied to the example image shown in Figure 43. The colormap shows the output after the specific filter was applied on the example image. Additionally, the image was extended using the padding method to have the same number of pixels after applying the filter. On the right side of each colormap, the corresponding filter as well as the maximum and minimum values of the colormaps elements can be seen.



**Filter 1**

| | | |
|---|---|---|
| -0.088 | -0.083 | -0.078 |
| -0.045 | -0.031 | -0.033 |
| 0.001 | 0.014 | 0.005 |

Max: -0.051
Min: -2.072

**Filter 2**

| | | |
|---|---|---|
| 0.023 | 0.017 | 0.008 |
| 0.020 | 0.014 | 0.010 |
| 0.010 | 0.013 | 0.007 |

Max: 0.785
Min: 0.046

**Filter 3**

| | | |
|---|---|---|
| -0.007 | -0.009 | -0.005 |
| -0.007 | -0.004 | 0.001 |
| -0.005 | -0.002 | 0.001 |

Max: -0.005
Min: -0.230

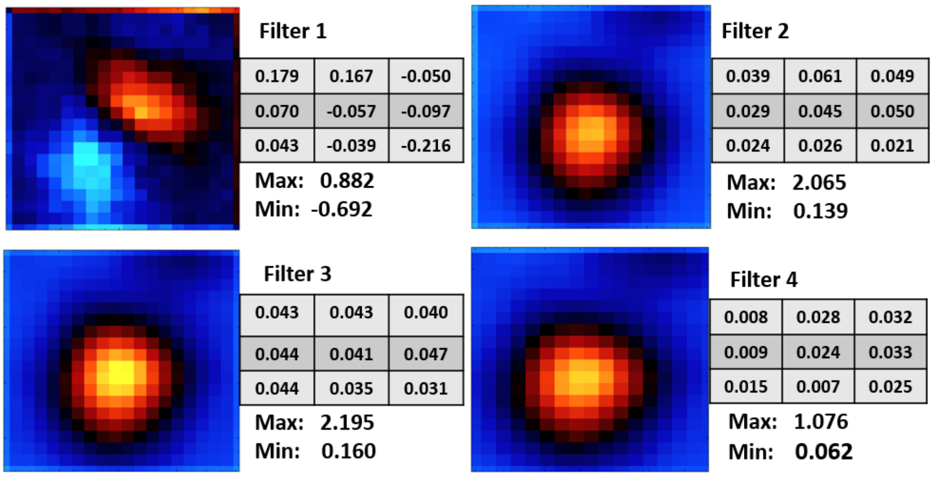
**Filter 4**

| | | |
|---|---|---|
| -0.022 | -0.026 | -0.027 |
| -0.012 | -0.015 | -0.015 |
| 0.005 | -0.003 | -0.009 |

Max: -0.027
Min: -0.758

**Figure 47:** The filters of the CNN with bias, determined through the SGB training process, were applied to the example image shown in Figure 43. The colormap shows the output after the specific filter was applied on the example image. Additionally, the image was extended using the padding method to have the same number of pixels after applying the filter. On the right side of each colormap, the corresponding filter as well as the maximum and minimum values of the colormaps elements can be seen.

maximum and minimum values in the output image, compared to the other filters. In contrast, Filter 2 in Figure 47 leads to a smaller range in the output image resulting in a no NV-center prediction as shown in Figure 43 (top right).
Compared to the filters from Figure 48 and 49, which create output images that



Filter 1

| | | |
|---|---|---|
| 0.179 | 0.167 | -0.050 |
| 0.070 | -0.057 | -0.097 |
| 0.043 | -0.039 | -0.216 |

Max: 0.882
Min: -0.692

Filter 2

| | | |
|---|---|---|
| 0.039 | 0.061 | 0.049 |
| 0.029 | 0.045 | 0.050 |
| 0.024 | 0.026 | 0.021 |

Max: 2.065
Min: 0.139

Filter 3

| | | |
|---|---|---|
| 0.043 | 0.043 | 0.040 |
| 0.044 | 0.041 | 0.047 |
| 0.044 | 0.035 | 0.031 |

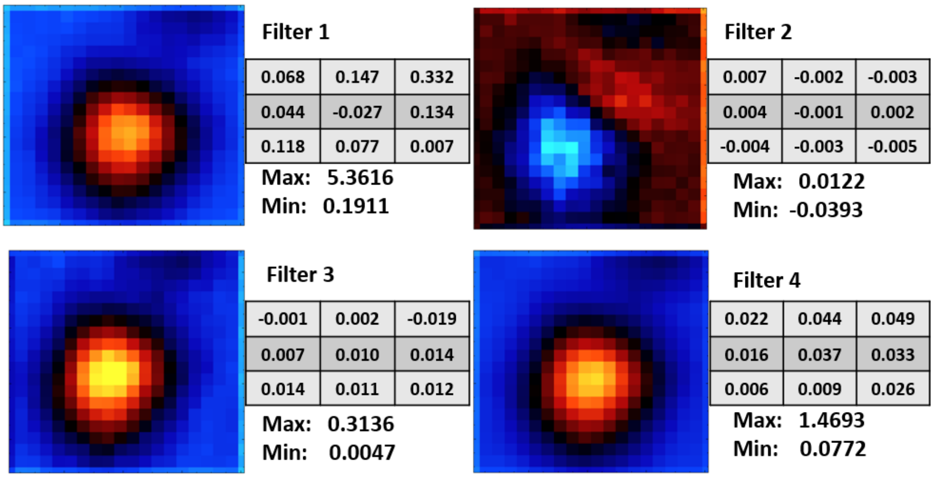
Max: 2.195
Min: 0.160

Filter 4

| | | |
|---|---|---|
| 0.008 | 0.028 | 0.032 |
| 0.009 | 0.024 | 0.033 |
| 0.015 | 0.007 | 0.025 |

Max: 1.076
Min: 0.062

**Figure 48:** The filters of the CNN without bias, determined through the MBGD training process, were applied to the example image shown in Figure 43. The colormap shows the output after the specific filter was applied on the example image. Additionally, the image was extended using the padding method to have the same number of pixels after applying the filter. On the right side of each colormap, the corresponding filter as well as the maximum and minimum values of the colormaps elements can be seen.



Filter 1

| | | |
|---|---|---|
| 0.068 | 0.147 | 0.332 |
| 0.044 | -0.027 | 0.134 |
| 0.118 | 0.077 | 0.007 |

Max: 5.3616
Min: 0.1911

Filter 2

| | | |
|---|---|---|
| 0.007 | -0.002 | -0.003 |
| 0.004 | -0.001 | 0.002 |
| -0.004 | -0.003 | -0.005 |

Max: 0.0122
Min: -0.0393

Filter 3

| | | |
|---|---|---|
| -0.001 | 0.002 | -0.019 |
| 0.007 | 0.010 | 0.014 |
| 0.014 | 0.011 | 0.012 |

Max: 0.3136
Min: 0.0047

Filter 4

| | | |
|---|---|---|
| 0.022 | 0.044 | 0.049 |
| 0.016 | 0.037 | 0.033 |
| 0.006 | 0.009 | 0.026 |

Max: 1.4693
Min: 0.0772

**Figure 49:** The filters of the CNN without bias, determined through the SGD training process, were applied to the example image shown in Figure 43. The colormap shows the output after the specific filter was applied on the example image. Additionally, the image was extended using the padding method to have the same number of pixels after applying the filter. On the right side of each colormap, the corresponding filter as well as the maximum and minimum values of the colormaps elements can be seen.

consist of nearly only positive numbers. In this case, two filters also stand out which search for different characteristics in the image. Filter 1 from Figure 48 leads to both a negative and positive dip or peak. This means that the network is searching for a transition from high to low count rates as indicated by the relatively high values of the filter in the elements $f_{1,1}$, $f_{1,2}$ and $f_{3,3}$ when considering their absolute values compared to other elements. Filter 2 and Filter 3 have a very similar structure, which results in a similar output image. The last filter Filter 4 stretches the output image. When evaluateing the range of the output image caused by the filters, it can be observed that Filter 2 and 3 likely have the most influence. Since these filters are searching for circles, they lead to a good prediction of the position and size of the NV-center. Overall the network without bias appears to be more effective in capturing different features in the image including transitions and circular patterns which may lead to better predictions of NV-center characteristics. However, the influence of these values on the final outcome can only be speculated upon as the weights of the following ANN are challenging to interpret.

Figure 49 shows a similar output of filters compared to Figure 48. Here, it is noticeable that one filter again deviates significantly from the others. Unlike in Figure 48 (Filter 1) this one does not lead to both a peak and a dip but only a dip. The overall structure of the filters remains the same. Elements $f_{1,1}$ and $f_{3,3}$ have the largest absolute values in the filter and their signs also match. The key difference between the filters lies in Element $f_{1,2}$ which goes from a high positive value to a low negative value. These values could be one of the reasons why the filter does not lead to a peak in the output image. The remaining three filters are searching for circular structures with Filter 1 creating a high contrast in the output image. But even in this example it can only be speculated upon as the weights of the ANN are difficult to interpret.

In summary ,considering the filters allow to evaluate why the predictions of this CNN with bias might lead to bad results. The cause appears to be not only related to the classification performed by the ANN but also to the characteristics of the filters themselves.

### 6.3.3 Comparison of the Neural Networks

Chapter 6.3.1 and Chapter 6.3.2 showed two methods for neural networks in terms of their structures and results. Chapter 6.3.1 focused on a simple ANN while Chapter 6.3.2 focused on a CNN, as an extension of the ANN. These chapters discussed the general architecture and the training process. In the case of the CNN, the impact of filters was also demonstrated. Various settings, such as activation functions, the number of hidden layers, the number of neurons per hidden layer, the loss function and the learning rate were varied to get the best results. Additionally, in the case of the CNN also the number of filters, filter sizes, the use of padding or not and pooling were adjusted and tested to optimize the performance.

The previous chapters demonstrated the impact of different settings by changing two main factors. Firstly, different training methods were used and secondly, the use of bias was varied. These chapters highlighted that the presence or absence of bias can have a big influence to the predictions of both ANN and CNN models. The choice of the training method also had a noticeable impact on predictions although its influence was generally smaller compared to the influence of the bias. The comparison between these methods was then visualized in Figure 38 for the ANN and in Figure 43 for the CNN. It can be seen that the MBGD training method and a CNN without bias leads to the best results. This was further tested for additional examples in Figure 40 for the ANN and in Figure 45 for the CNN.

The result shows that a simple ANN is not able to provide the desired outcomes. In comparsion, the CNN produced positive results across various datasets with different settings like the resolution, intensity range, etc.. The CNN also had disadvantages compared to the ANN like the accuracy in predicting the position and size of the NV-centers. The ANN often predicted these more accurately as seen in the figures. Additionally, from Figures 37 and 42 it can be seen that the CNNs loss was on average larger than that of the ANN leading to less precise predictions. Nevertheless, the advantages of the CNN outweighed its disadvantages, making it the preferred choice as the backbone model with the given parameters mentioned in the chapter.

## 6.4 Postprocessing

After the backbone has processed the data, it is necessary to evaluate the results. Since the number of input data points is relatively small (25 values), there is a possibility of detecting structures such as noise or edges as NV-centers, as can be seen in Figure 50 (top left). These structures need to be filtered out in the final step through a postprocessing phase to achieve better results. Various static functions are used to adjust the predictions from the backbone. For each of these functions a threshold is determined which then leads to the best results. For example, this process filters out noise and edges, as well as predictions that are too large or have too few pixels in order to improve the overall outcome. These functions are not only used for filtering ROIs but can also slightly adjust their positions to center them, for example, on the highest point or ignore ROIs that fall outside of the scanning area and can not be fully confirmed as NV-centers. These types of functions require specific thresholds to be set.

To determine the thresholds, the confusion matrix explained in Chapter 2.6 is used. To do this a ground truth (GT) map of the scan must first be created. This means that all NV-centers in this scan with their sizes need to be manually identified and characterized. These scans with the respective GT map form the validation and test dataset, whereby the validation dataset is used in the preprocessor to determine the best thresholds.

To use the confusion matrix, first the TP, FP and FN need to be determined. To do this the program compares the predicted ROIs with the GT elements. In this step, the program goes through all the predictions and checks if a GT element is found nearby. If it matches, that prediction is considered correct and counted as TP. In this case, the GT element is removed from the list and can not be assigned to any other prediction. If no GT element is found nearby, the prediction is incorrect and counted as FP. Once all predictions have been assigned, the program then checks how many GT elements were missed by the AI and count these as FN. With Equation 30 then the F1-score can be determined to get the quality of the network. In the learning mode of the postprocessor, the different thresholds are adjusted and the F1-score for each combination of thresholds is determined. Both, the learning and application mode are shown in Figure 50. On the left side at the top, the output of the backbone with all the predictions for ROIs can be seen. Below this scan is the GT data. To determine the best possible thresholds, the highest F1-score needs to be found. To do this step by step, the thresholds for dirt and edges, too large ROI and ROI with low count as listed below are adjusted. Then the best thresholds are determined and set in the program. After this, the procedure is repeated once more.

**Remove dirt and edges:** In Figure 50 it can be seen from the output of the backbone that many ROIs are located at the edges. This occurs because with the low number of input data points the 5x5 pixel regions can sometimes be identical or very similar to those of NV-centers. To solve this problem, either the backbone or use of preprocessing techniques filter out such ROIs based on factors like the distance between the ROIs which can be modified. In this example, the second solution was chosen. The result is shown in light blue in the following figure. Additionally, the impurity, near the top left corner of the edge, has also been filtered out.

**Remove too large ROI:** The second threshold deals with too large ROIs. In

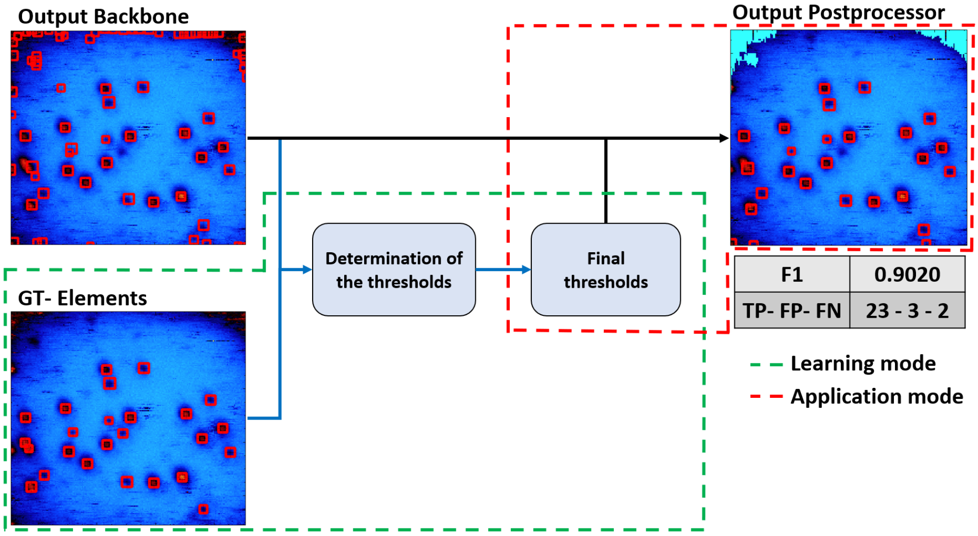


**Figure 50:** The structure of the postprocessor for a sample scan is illustrated below showing both the learning and application modes in their respective boxes. For the learning mode both, the output of the backbone (top left) and the GT elements (bottom left) are required to determine the best thresholds. The application mode only needs the previously determined thresholds to process the output of the backbone and make the final output.

the experiment NV-centers in confocal microscopy have a size ranging from approximately 300 nm to 400 nm. So that, if a predicted ROI is way larger than this 300 nm to 400 nm range, it can be assumed that it is not an NV-center. Using this function provides the ability to filter out NV-centers that are larger than a threshold size determined by the postprocessor.

**Remove low count:** When capturing a scan, there may be some areas with a high point density that leads to predicted ROIs . The causes of this can include short term inaccuracies in the measurement process which generate stipes in the scan. Also in the case of edges, higher point densities can occur. While these should normally be sorted out through the function mentioned earlier, some may fall below the average. To solve these errors, unlike the previous methods, three thresholds are used in the postprocessor. The first threshold describes the size of the neighborhood to be used for calculating the local average. This is necessary because especially in large scans the local average can deviate significantly from the global average. This is clearly visible in Figure 50, especially in the area of the edges which have a higher mean value than the center of the scan. The next two thresholds are related to the ROI. The first one deals with the number of pixels in the ROI that must be higher than a certain value and this value or threshold is given in percentages. The second threshold then deals with the factor by which the mean must be multiplied to be detected it as a ROI. It is important to note that the three required thresholds must be finely tuned to achieve the best possible result.

**Distance between prediction and gt element:** The last threshold is used for the optimal calculation of the metrics. It describes the Euclidean distance between the predicted NV-centers and the manually determined GT elements. What makes this threshold unique is that unlike the previously explained thresholds it is only needed in the learning mode. And in this mode it plays an important role in finetuning the other thresholds to achieve better results.

### 6.4.1 F1-Score

In the previous chapter already all the thresholds in the postprocessing have been discussed. The thresholds used later are determined based on the maximum F1-score and are then set fix in a text file within the program. In Figure 50, the final F1-score for the scan is shown below the output of the postprocessor. In this scan, the AI correctly identified 23 NV-centers, made mistakes in three ROIs and missed two NV-centers. This results in an F1-score of 0.9020. But it is important to note that the postprocessor learned from just a single scan, which makes the thresholds not very generalizable. Therefore, these thresholds were not finalized based on one scan but were adjusted using multiple scans. The six scans with a total number of 104 NV-centers differ in properties such as resolution, NV-center density, intensity ranges, whether they have edges or not and whether they have impurities or not. This diversity was introduced to get the best possible parameters for the thresholds and ensure their stabillity to different scan characteristics.
In addition to the thresholds, described in the previous chapter, there is another threshold. But this threshold is part of the neural networks backbone and not the postprocessor. In the backbone, the first output of each prediction determines whether it is an NV-center or not. In the training data the value was '1' if it is an NV-center. During evaluation by the network a high value close to '1' indicates a high probability while a low value corresponds to a lower probability. The neural network provides a lot of predictions but filters out some of them because the probability of being an NV-center is too low. The threshold at which predictions are filtered out forms the final threshold of this AI which is also varied in the following. Thresholds between 0.6 to 0.99 are then used in the learning mode of the postprocessor. The results can be seen in the following diagram 51 top left. Additionally, the metrics (TP, FP and FN) corresponding to each F1-score were also plotted against the threshold as shown in the bottom left.
The diagram shows that the F1-score increases with the rising threshold in the backbone reaching a maximum of 0.9082 at a threshold between 0.86 and 0.87. Beyond this point, the F1-score in the backbone starts to decrease as the threshold increases. This trend can be explained using the metrics. With a low threshold, the number of predictions from the backbone is very high leading to many predictions that can not be matched with GT elements resulting in a large number of FPs. In Figure 51 bottom left, the number of FPs is plotted in orange. The number constantly decreases as the threshold increased. The other way around, the number of FNs should constanly increase because with a high number of predictions allmost all GT elements should be recognized. But this can not be seen in the green data points in the diagram. The likely cause are the functions in the postprocessor which often classify ROIs, that are close together, as impurities or edges. This leads to a situation where, even though the backbone makes more predictions, fewer ROIs remain as the threshold increases resulting in fewer GT elements being matched. At a threshold of approximately 0.77, this effect is no longer as strong and the curve of FNs follows the expected trend. The problem with FNs also explains the trend of the TPs which then leads to a lower F1-score in the upper diagramm. The jump in the range of 0.75 can also be explained in this way, as it appears that the effect disappears around this threshold.
Additionally, the F1-score was also determined for each individual scan to see how the final thresholds affect different settings. The results are shown in Figure 51 on the right. Scan 1 and Scan 5 represent a standard measurement with an ideal resolution for the network and minimal disturbances. Scan 2 consists of the scan

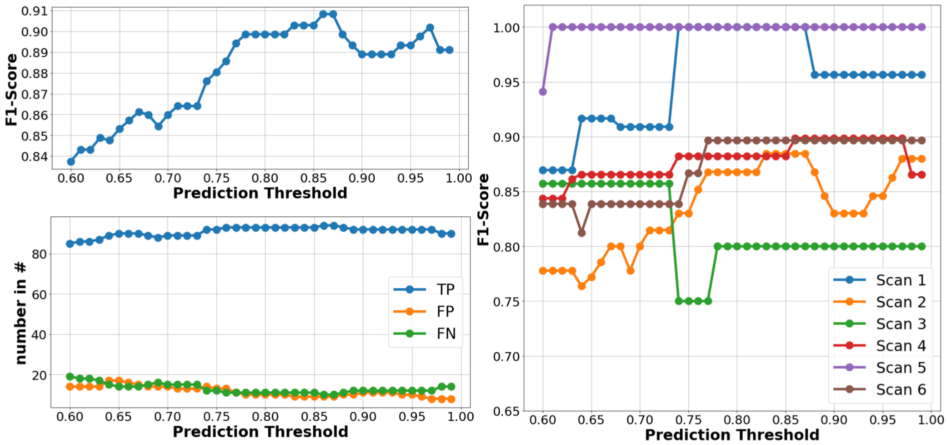


**Figure 51:** The upper left diagram shows the F1-score development at various thresholds of the backbone. The corresponding values of the metrics for TP, FP and FN for each respective F1-score are plotted in the lower left diagram. The right diagram shows the F1-score development for the individual scans.

shown in Figure 50 and is meant to symbolize the impact of impurities and edges in the scan. Scan 3 has a unique feature with a strong intensity difference between different NV-centers. Scan 4 represents a large scan with low resolution and Scan 6 contains many closely spaced NV-centers. In the diagram it is noticeable that the learning process performs well with most parameters but encounters difficulties with the strong intensity difference which could be due to the preprocessor.
The final values for the thresholds determined for this masters thesis and used in the AI later are listed below.

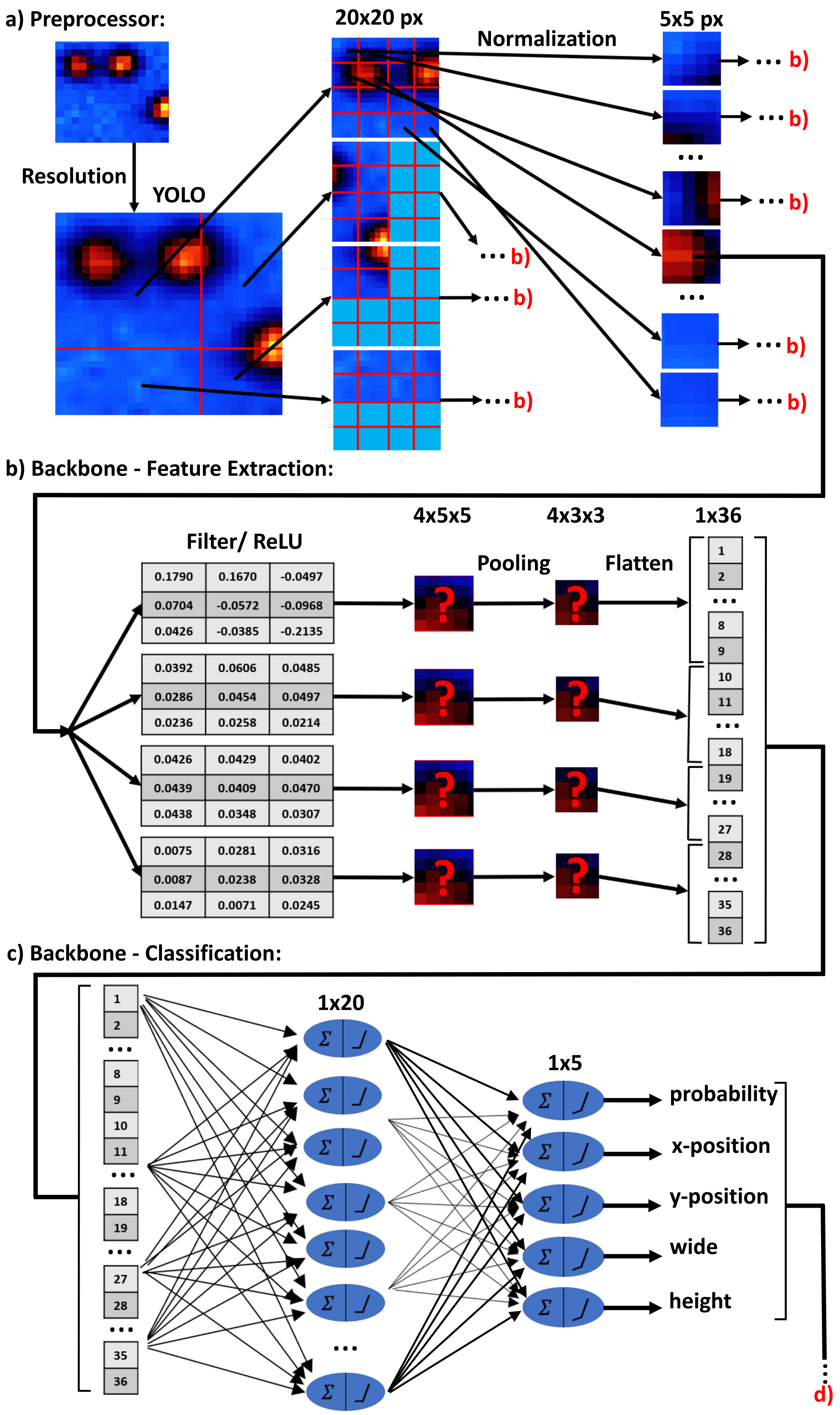
**Table 10:** Final parameter for the thresholds of the postprocessor.

| Prediction Threshold | 0.86 | 0.87 |
|---|---|---|
| Dirt and edges: | 525 | 525 |
| Too large ROI: | 815 | 815 |
| Distance prediction ↔ gt-Element | 475 | 475 |
| Low count (percentage): | 0.04 | 0.04 |
| Low count (surrounding Box): | 1000 | 1000 |
| Low count (multiplication): | 1.1 | 1.1 |

## 6.5 Final Architecture

After the previous chapters deal with the preprocessor, the backbone and the postprocessor this chapter explain the process, as shown in Figure 52, that a scan must go through from input into the Scan of ROI anomaly (SoRa) architecture to the final output with the predictions. In this chapter the training process has been left out for simplicity and because it has already been explained, with the focus shifted towards the evaluation.

As input, a scan with a size of 15x15 pixels and a resolution of 100 nm per pixel was used to show most of the processes. In Figure 52 a), the process of the scan through the preprocessor is shown. First the scans resolution is increased from 100 nm per pixel to 50 nm per pixel resulting in a size change to 29x29 pixels. Next the YOLO algorithm is applied by dividing the scan into 4 sections of 20x20 pixels. To obtain

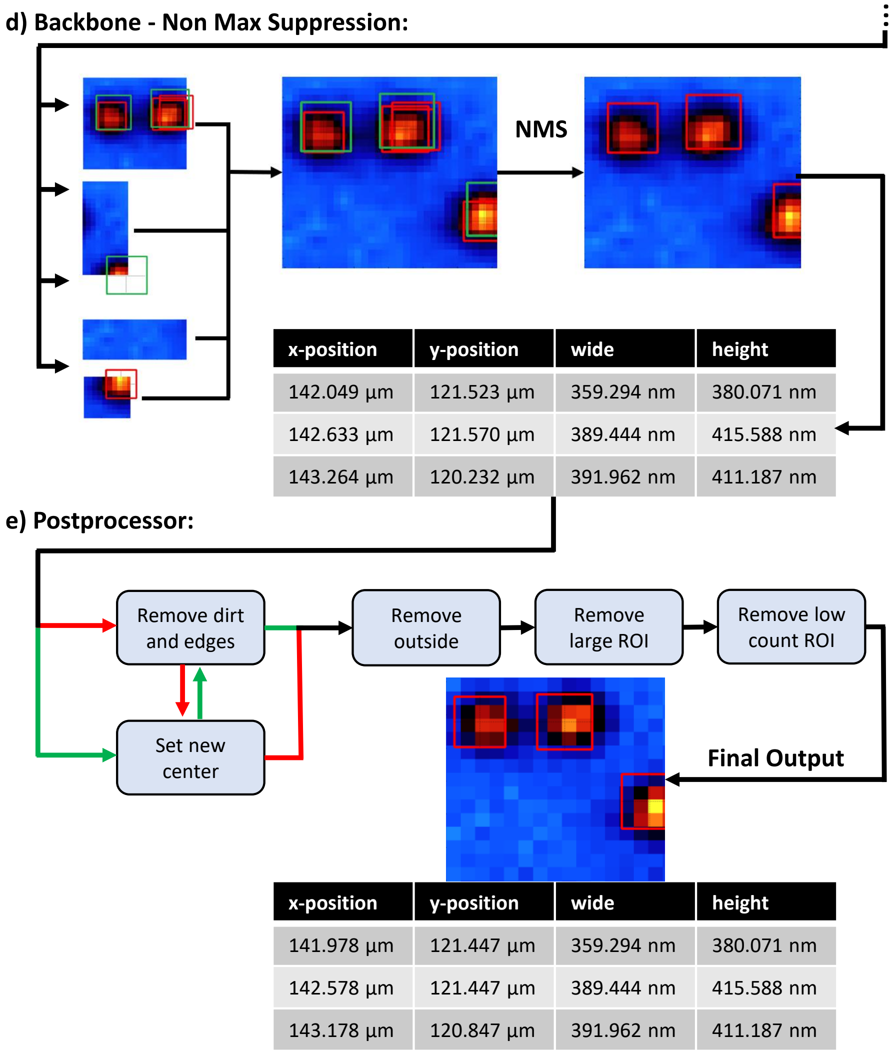


| | | |
|---|---|---|
| 0.1790 | 0.1670 | -0.0497 |
| 0.0704 | -0.0572 | -0.0968 |
| 0.0426 | -0.0385 | -0.2135 |

| | | |
|---|---|---|
| 0.0392 | 0.0606 | 0.0485 |
| 0.0286 | 0.0454 | 0.0497 |
| 0.0236 | 0.0258 | 0.0214 |

| | | |
|---|---|---|
| 0.0426 | 0.0429 | 0.0402 |
| 0.0439 | 0.0409 | 0.0470 |
| 0.0438 | 0.0348 | 0.0307 |

| | | |
|---|---|---|
| 0.0075 | 0.0281 | 0.0316 |
| 0.0087 | 0.0238 | 0.0328 |
| 0.0147 | 0.0071 | 0.0245 |



| x-position | y-position | wide | height |
|---|---|---|---|
| 142.049 µm | 121.523 µm | 359.294 nm | 380.071 nm |
| 142.633 µm | 121.570 µm | 389.444 nm | 415.588 nm |
| 143.264 µm | 120.232 µm | 391.962 nm | 411.187 nm |

| x-position | y-position | wide | height |
|---|---|---|---|
| 141.978 µm | 121.447 µm | 359.294 nm | 380.071 nm |
| 142.578 µm | 121.447 µm | 389.444 nm | 415.588 nm |
| 143.178 µm | 120.847 µm | 391.962 nm | 411.187 nm |

**Figure 52:** Final architecture of the network for detecting NV-centers using an example scan. The functionality of the preprocessor a), the backbone b), c), d) and the postprocessor e) are shown and how these affect the example scan in the process. The backbone process is further divided into feature extraction b), classification c) and NMS d). The final output is then shown at the bottom in the form of the scan and a table with the individual data.

these 4 sections, the scan needs to be expanded to 40x40 pixels with zeros added to the right and bottom sides as indicated by the light blue background in the figure. Then the average is computed without using the additional zeros and the section is divided into 16 segments of 5x5 pixels. These 5x5 pixel segments are then normalized using the previously determined average. These normalized segments then exit as vector the preprocessor.

The backbone, into which this vector is fed, consists of three components: the feature extraction (Figure 52 b)), the classification (Figure 52 c)) and the NMS (Figure 52 d)). It should be mentioned that the neural network shown here corresponds to the CNN presented in Chapter 6.3.2 and not the ANN shown in Chapter 6.3.1. In the first step, the pretrained filters are applied to the input. Zeros are added to

the sides making the dimensions 7x7. This is done to ensure that the dimension remains 5x5 after applying the filter. The resulting 5x5 matrix is then reduced in size using pooling. A 2x2 square is used for pooling and passing over the maximum value. To apply this to a 5x5 matrix, it also needs to be to expand to a dimension of 6x6 by adding zeros. The resulting 3x3 matrix is then converted into a vector. This procedure is performed for all four filters, so the final vector has a dimension of 36. This vector forms the input for the classification section in Figure 52 c). This consists of an ANN with one hidden layer and one output layer. The output layer has a dimension of 5 where the first value represents the probability, the next two values represent the position in the 5x5 section and the last two values represent the size of the ROI. Each 5x5 pixel section has such an output, meaning that the network has predicted 64 ROIs for the 20x20 scan. These ROIs need to be reduced in the next step. First, a threshold value is set to filter out ROIs with a probability lower than this threshold. In this case, the threshold value has been set to 0.86 through the learning process. This means that all ROIs with a probability below 0.86 are filtered out. In the second step, a single NV-center typically spans multiple 5x5 pixel sections which means that many ROIs correspond to the same NV-center. This can be seen in Figure 52 d). Each NV-center always has at least two associated ROIs with different parameters for position and size. The NMS method is now applied to identify ROIs which belong together and group them into a list. From each group the ROI with the highest probability is chosen and the values for position and size are selected. They are then marked in green in the figure while the red rectangels are filtered out. After the backbone the network has recognized all three NV-centers in the input scan but with some slight shifted position. For further adjustments, the data is passed through a postprocessor before the final output.
The postprocessor is shown in its function in Figure 52. The ROIs determined in the backbone are further processed here using thresholds, for example, to filter out larger structures like impurities or edges. The reason for this is that the network always focuses on small sections and in the postprocessor it processes the entire scan for the first time. In the figure, the arrow splits into a red and a green arrow before the first step. This is because the networks performance varies depending on whether there is a lot of impurity or edges in the scan. For edges it is important, for example, that the ROIs are in a line or close to each other. If the ROIs are initially shifted to the highest point in the ROI, the structure changes significantly enough to degrade the prediction. In the other case, if the scan does not have impurities or edges, the network produces better results when the ROIs are shifted first. Therefore, each arrow represents a slightly different order of operations depending on whether the scan contains a lot of impurity and edges or not. The used functions further edit the prediction parameter slightly, so that at the end of the output the position shown in the figure in the scan and in the table is the final output.

In Chapter 6.4.1, the validation dataset was used to determine the best thresholds. To test the final network with all its weights, filters and thresholds, the test dataset was used following the same principle as the validation dataset. This test dataset consists of 5 scans and a total number of 70 NV-centers. To get the final results of the network architecture, the test dataset was evaluated. The metrics including the F1-score have the following form:

| F1 | TP | FP | FN |
|---|---|---|---|
| 0.9209 | 64 | 5 | 6 |

The test dataset shows a very similar F1-score to that of the validation dataset demonstrating that the trained network architecture successfully works on data that was not used for training or optimization purposes.

# 7 Data Analysis

In addition to the object detection another area was modified which is related to ODMR measurements and their analysis. In Chapter 4.1, the general measurement routine was described. The next step, that the AI must be capable of, is recognizing and analyzing the ODMR measurements. The functions that the program must handle includes the single and double lorentzian fits for cw ODMR and a damped sine fit for Rabi-oscillations.

The classical way to achieve this is by using general fitting routines such as the Levenberg-Marquardt method (LM). To include these into the program the Alglib library was chosen which provides these fitting routines in C++[51]. In addition to this, another method inspired by the backpropagation of a neural network was created referred to as BPF. This chapter will focus on the second method and the comparison of both fitting routines.

## 7.1 Backpropagation Fitting

In the Backpropagation Fitting (BPF), the same training process as in a neural network is used, slowly adjusting the function to fit the data points. The parameters to be adjusted in this process are the fitting parameters, such as amplitude, full width at half maximum (FHWM), position of the extremum ($\omega_0$) and offset in the case of the lorentzian function. Each fitting parameter is optimized over the specified number of epochs using its respective learning rate. The loss is then calculated as distance between the measured data point and the point determined by the fitting parameters at the corresponding x position. By using the loss, the function determined by the BPF can be compared to the function determined by the traditional LM method.

To use the fitting process for both methods, the approximate fit parameters must first be determined in a previous step. Without this function both methods may set a wrong minimum as the optimum, which would lead to unusable results in the following measurement processes. The approximate fit parameters are almost always determined from the unchanged measurement data in previous functions. Only in the case of the frequency of the Rabi-oscillation the measurement values are initially Fourier-transformed using Alglib.

In the first step, the BPF is tested. Since the LM method usually possesses the optimal value, the curve determined by the BPF should increasingly approximate it, as the number of epochs is increased. Figure 53 a) precisely shows this behavior for the functions of the simple and double lorentzian as well as the damped sine.

In Figure 53 a) it can be seen that as the number of epochs increases, the function adjusted by the BPF gets closer and closer to the fit parameters determined using the LM. At about one million epochs, the fit parameter values are identical or very close to each other. In the figure this is noticeable by the fact that the green curve (LM) and the red curve are almost on top of each other. To better compare the quality of the fit parameters, the loss was also used. It is calculated as the sum of the differences between the measurement value and the value determined by the fit function at that x position. Table 11 shows these loss values for the BPF with different epochs and those of the LM for a single lorentzian (SL), a double lorentzian (DL) and a damped sine (DS) function.

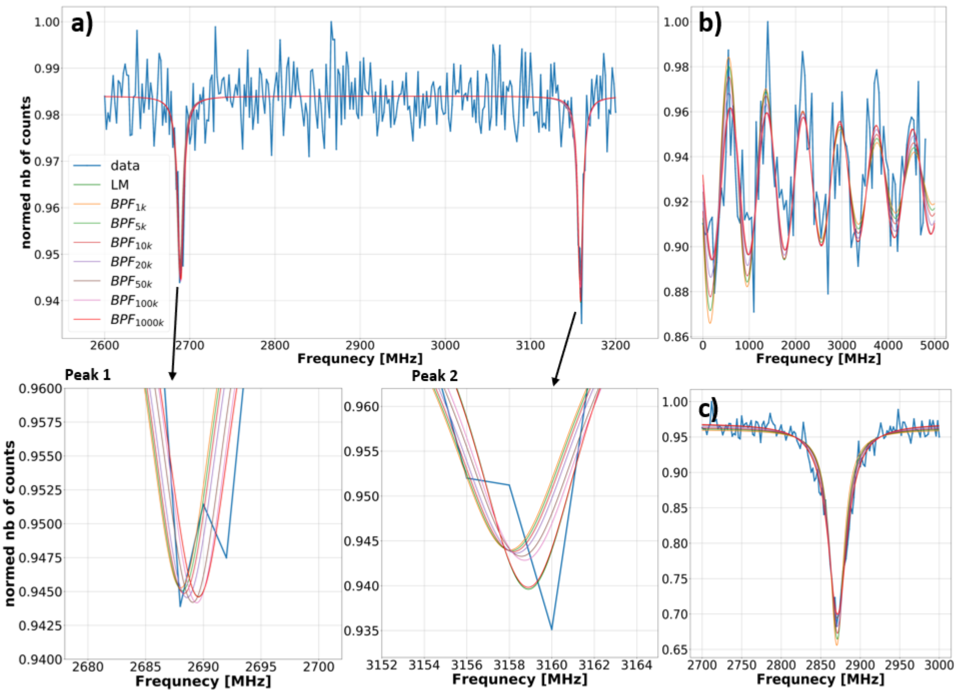


**Figure 53:** Comparison of the fit parameters determined by the LM and the BPF method as fuctions for the double lorentzian a), the damped Sine b) and the single lorentzian. Different epochs were used for the BPF. The legend is shown in a) and applies to the other graphs as well.

**Table 11:** Loss for the LM and BPF methods with different numbers of epochs for three different functions (DL, SL and DS).

| | BDF | | | | | | | LM |
|---|---|---|---|---|---|---|---|---|
| | 1k | 5k | 10k | 20k | 50k | 100k | 1000k | |
| SL | 1.876 | 1.727 | 1.596 | 1.472 | 1.371 | 1.359 | 1.360 | 1.360 |
| DL | 1.347 | 1.349 | 1.351 | 1.355 | 1.362 | 1.366 | 1.371 | 1.371 |
| DS | 1.760 | 1.731 | 1.694 | 1.642 | 1.595 | 1.586 | 1.576 | 1.576 |

For the single lorentzian and damped sine functions, it can be seen that the larger the number of epochs the smaller the loss value becomes. This is expected because the program has more time to reach the optimum values. But there is also an interesting observation with the double lorentzian function where the loss slowly increases. This is likely due to the fact that some of the initial guess parameters for the curve are already quite good but they get worse during the optimization of the other fit parameters.

## 7.2 Weighted Backpropagation Fitting

In the previous chapter it can be seen that the BPF works but it took longer and leads to similar results, compared to the LM method, when the number of epochs is high. But the BPF method has a big advantage: weights. While the LM method tries to determine the best possible combination of fit parameters for the curve, the BPF only determines the best possible value for each fit parameter separately. This means that weights can be introduced allowing the program to focus more on fit

parameters which are important for the measurement. In the case of cw ODMR, the exact value of the the position of the extremum of the dip $\omega_0$ is important but not so much the precise values for the offset or the FHWM. The same goes for Rabi-oscillation, which should measure the $\pi$-pulse time as accurately as possible, the parameters for $\omega$ and $\phi$ become more important then the decay $\tau$ or the offset. The behavior of the weights was then investigated with a fixed number of iterations. It is important to note that for a fair comparison, the number of epochs needed to be adjusted so that the total number of optimizations remained the same between the measurements. For example, if the $\omega$ parameter in a damped sine was improved multiple times within one epoch while the other fit parameters remained constant the resulting curve would naturally fit better. Therefore, multiple improvements to one fit parameter must come at the expense of the other fit parameters to achieve a fair comparison.
To test whether the BPF method is better compared to the LM method different settings were tested. The modified settings included not only weights but also the number of epochs and learning rates. Two different scans were used for each type of measurement to evaluate whether the improvement due to the change is reproducible. These scans were chosen to have significant differences in parameters such as resolution and the number of data points to better evaluate the results. The indicator of how good the values are was once again the loss. By varying the settings over 300 different results were compared. The settings for the best results for each scan along with the corresponding LM value for comparison are shown in the following Table 12. The best loss for each is highlighted in bold.

**Table 12:** Comparison of the loss with different settings of the weights, the learning rate and the number of epochs for two different scans. In addition, the value resulting from the LM was also shown for a better comparison. The loss shown in bold shows the lowest and therefore best value.

| Single Lorentzian | | | | |
|---|---|---|---|---|
| epochs | weights [amp, FHWM, $\omega_0$, offset] | learning rate [amp, FHWM, $\omega_0$, offset] | loss | |
| | | | scan 1 | scan 2 |
| LM | - | - | **0.8523** | 0.3602 |
| 10 000 | [1, 1, 3, 1] | [2, 7, 25, 0.1] | 0.8786 | 1.3586 |
| 10 000 | [1, 1, 1, 1] | [2, 9, 20, 0.1] | 0.8809 | **1.3494** |
| 20 000 | [1, 1, 3, 1] | [2, 7, 25, 0.1] | 0.8750 | 1.3589 |
| 20 000 | [1, 1, 1, 1] | [2, 7, 20, 0.1] | 0.8791 | 1.3583 |
| Double Lorentzian | | | | |
| LM | - | - | 1.3736 | 4.5783 |
| 10 000 | [1, 1, 1, 1] | [2, 9, 20, 0.1] | **1.3486** | 4.6182 |
| 10 000 | [1, 1, 3, 1] | [2, 7, 25, 0.1] | 1.3513 | 4.5776 |
| 20 000 | [1, 1, 1, 1] | [2, 9, 20, 0.1] | 1.3503 | 4.5829 |
| 20 000 | [1, 1, 3, 1] | [2, 9, 20, 0.1] | 1.3536 | **4.5635** |
| Damped Sine | | | | |
| epochs | weights [amp, offset, $\omega$, $\varphi$, $\tau$] | learning rate [amp, offset, $\omega$, $\varphi$, $\tau$] in $10^{-3}$ | loss | |
| | | | scan 1 | scan 2 |
| LM | - | - | **1.5760** | 2.5332 |
| 10 000 | [2, 1, 2, 3, 1] | [2, 10, 0.1, 200, 0.01] | 1.6380 | 2.9574 |
| 10 000 | [1, 1, 3, 2, 1] | [1, 10, 0.1, 200, 0.01] | 1.6965 | 2.5978 |
| 20 000 | [2, 1, 2, 3, 1] | [2, 10, 0.1, 200, 0.01] | 1.5933 | 2.5350 |
| 20 000 | [2, 1, 2, 3, 1] | [1, 10, 0.1, 200, 0.01] | 1.6176 | **2.5332** |

Table 12 shows that the fit parameters determined from the LM method may not necessarily provide the best approximation to the measurement data. While the LM method only has the lowest loss for two scans it has a higher loss for the other scans. But it is important to note that the fact that BPF provides a lower loss is highly dependent on the guess parameters. If these guesses are far from the actual values it may take longer to find or even fail to find them, because of the possibility that only a higher local extremum will be determined instead of a lower one.. This problem is less prominent with the LM method.

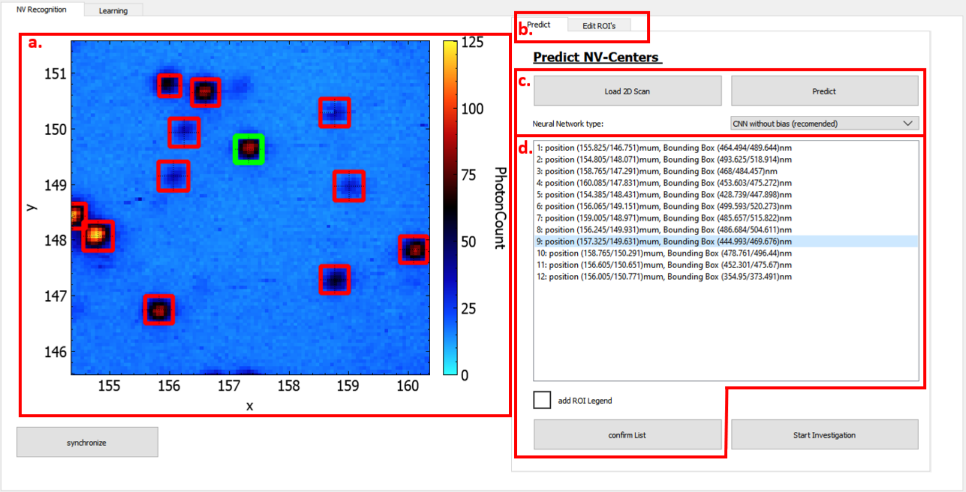
The table also shows, that in the case of the BPF, weighted generally lead to better results than unweighted values. Specifically for the damped sine certain weights lead to lower losses. But it is worth mentioning that using no weights can also lead to good results. This might be due to the presence of good guess values as in this case the program does not need to make significant adjustments. This would explain why the damped sine benefits from weights, as predicting the values for $\omega$ and $\varphi$ is less accurate compared to, for example, the amplitude or the offset.

To get the best possible results a combination of both methods could be used. The LM method is used to test whether it is a local or global extremum by evaluating the loss and then with the BPF the fitting parameters can be more precisely determined using the best settings determined earlier. If these parameters turn out to be worse the values calculated from the LM method can be used instead.

# 8 SoRA

After discussing object detection and the fitting method in previous chapters, this chapter will deal with the Scan of ROI AI (SoRA). This section covers the data generation process for training and validation as well as the corresponding windows for data recognition and analysis. This chapter will not show any code due to it spanning over 10 000 lines. The reason for this high volume of code is that no external libraries were used for the neural network. Instead all functions were written from scratch.

## 8.1 Prediction



Neural Network type: CNN without bias (recomended)

```
1: position (155.825/146.751)mum, Bounding Box (464.494/489.644)nm
2: position (154.805/148.071)mum, Bounding Box (493.625/518.914)nm
3: position (158.765/147.291)mum, Bounding Box (468/484.457)nm
4: position (160.085/147.831)mum, Bounding Box (453.603/475.272)nm
5: position (154.385/148.431)mum, Bounding Box (428.739/447.898)nm
6: position (156.065/149.151)mum, Bounding Box (499.593/520.273)nm
7: position (159.005/148.971)mum, Bounding Box (485.657/515.822)nm
8: position (156.245/149.931)mum, Bounding Box (486.684/504.611)nm
9: position (157.325/149.631)mum, Bounding Box (444.993/469.676)nm
10: position (158.765/150.291)mum, Bounding Box (478.761/496.44)nm
11: position (156.605/150.651)mum, Bounding Box (452.301/475.67)nm
12: position (156.005/150.771)mum, Bounding Box (354.95/373.491)nm
```

**Figure 54:** The main window that appears after SoRA has been opened. This consists of the scan shown in a color map a) with or without the respective predictions, the ROI tab system b), the prediction c) and the list of predictions d).

Figure 54 shows the window that appears after SoRA is opened. This window consists of 4 parts which are briefly presented below.

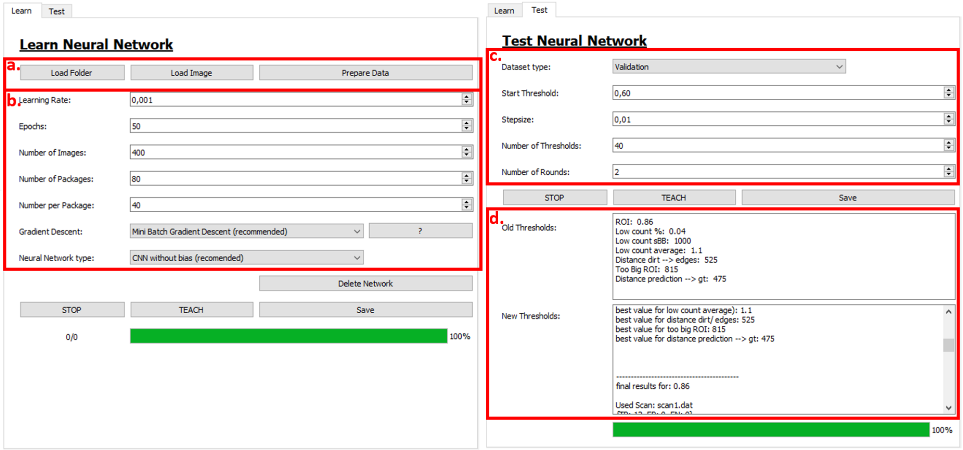
**a) Scan and prediction visualization:** In order to evaluate the quality of the predictions, the measured scan is shown as a colormap in a widget using QCustomPlot. The colormap shows the number of kcounts in a scanned xy plane. In addition, multiple red rectangles and one green rectangle are already shown in the figure. These rectangles represent the visualized output of the network. The center of the rectangle forms the predicted position and the side lengths form the predicted size.

**b) ROI tab system:** The tab system describes the prediction tab which can be seen in the figure and shows the widget objects for the prediction. The second tab called *Edit ROI's* provides the option to delete or add ROIs. But the function of this tab has been simplified to a drop-down menu that appears in the scan with a right click.

**c) Prediction:** When SoRA is opened, the scan which is currently open in the main program is transferred. There is also the option to load and evaluate previous scans. Using the *Predict* button, the scan is evaluated using the neural network and the result is displayed in the QCustomPlot as rectangles. The type of backbone can be selected below. The standard here is the CNN without bias but other network types can also be used for prediction.

**d) List of predictions:** After the scan has been evaluated via object recognition, the output is not only visualized in the form of rectangles in the colormap but also shown in a list. This list can be updated for a further measurement process by adding or changing the ROIs in the colormap by using the *confirm list* button for example for a final measurement process. For a better overview the ROI, that is currently selected, is marked in green in the color map.

## 8.2 Training



Learning Rate: 0,001
Epochs: 50
Number of Images: 400
Number of Packages: 80
Number per Package: 40
Gradient Descent: Mini Batch Gradient Descent (recommended)
Neural Network type: CNN without bias (recomended)
0/0 100%

Dataset type: Validation
Start Threshold: 0,60
Stepsize: 0,01
Number of Thresholds: 40
Number of Rounds: 2

Old Thresholds:

```
ROI:  0.86
Low count %:  0.04
Low count sBB:  1000
Low count average:  1.1
Distance dirt --> edges:  525
Too Big ROI:  815
Distance prediction --> gt:  475
```

New Thresholds:

```
best value for low count average): 1.1
best value for distance dirt/ edges: 525
best value for too big ROI: 815
best value for distance prediction --> gt: 475



--------------------------------------------
final results for: 0.86

Used Scan: scan1.dat
```

100%

**Figure 55:** The tabs for the training process of the backbone (left) and the tab for determining the best thresholds (right). The learning tab is divided into the data loading section for data generation a) and the training process settings b). The test tab is also divided into the settings of the training process c) and the output of the training process d).

In the previous chapters, the training data was briefly discussed. It consists of a 5x5 pixel section which leads to 25 values. In addition to these 25 values, there are 5 more numbers used to classify the NV-center. These additional numbers represent the probability, the position in x and y as well as the height and width of the center. The output of the network is designed to correspond to these 5 values that classify the NV-center.

The program separates the training into two tabs as seen in Figure 55. The left tab is responsible for training the backbone. It mainly consists of two parts: loading data for training data generation which is not important for the training process itself because it just create training data (Figure 55 a)) and settings for the training process b). Training data generation is covered in the following Chapter 8.2.1, so it will be focused on the settings here.

Most settings, such as the learning rate and epochs, have been explained in previous chapters. The other values, such as the number of images, the number of batches and the number of images in a batch depend on the specific training process. In the dropdown menu, it can be selected between MBGD and SGD for the training process. Depending on the chosen method, the program will require different values. For SGD the number of images to be randomly selected is needed while for MBGD

the number of batches used in the training process and the number of images per batch is needed.
In addition to the training method, the backbone architecture using a dropdown menu can also be selected. The possibilities include ANN and CNN each with the option to include or exclude bias. After selecting the backbone architecture, the training process can be started, stopped or saved using the buttons located under section b).

The second tab focuses on optimizing the thresholds introduced in Chapter 6.4. Similar to the first tab, it has adjustable settings for the optimization process (55 c)). In the dropdown menu, it can be chosen between the validation dataset and the test dataset. It is important to note that optimization is only possible with the validation dataset. The test dataset is just for the final evaluation. The other adjustable settings are the start threshold, which describes the threshold used to filter out ROIs already in the backbone, the stepsize, the number of thresholds which should be tested and the number of rounds which describes how often the parameters should be optimized. In each round it takes the optimized values from the previous round and further optimizes them. The optimization process with more then one round helps fine-tune the thresholds for better performance.
The section shown in Figure 55 d)) shows the ongoing optimization process. In the upper part, the thresholds loaded from the file and in the lower part, the optimal parameters corresponding to each threshold are shown. This allows to track the progress of the optimization process and observe how the parameters change with different thresholds.

### 8.2.1 Training dataset

The training data is prepared by opening a new window. The window is accessed using the *Prepare Data* button and is shown in Figure 55. As shown in Figure 55, it can load either a single scan or an entire folder with multiple scans. The scan, or in the case of multiple scans, the first scan is then displayed in the QCustomPlot in Figure 56 a). To create the training data, first the NV-centers need to be characterized.

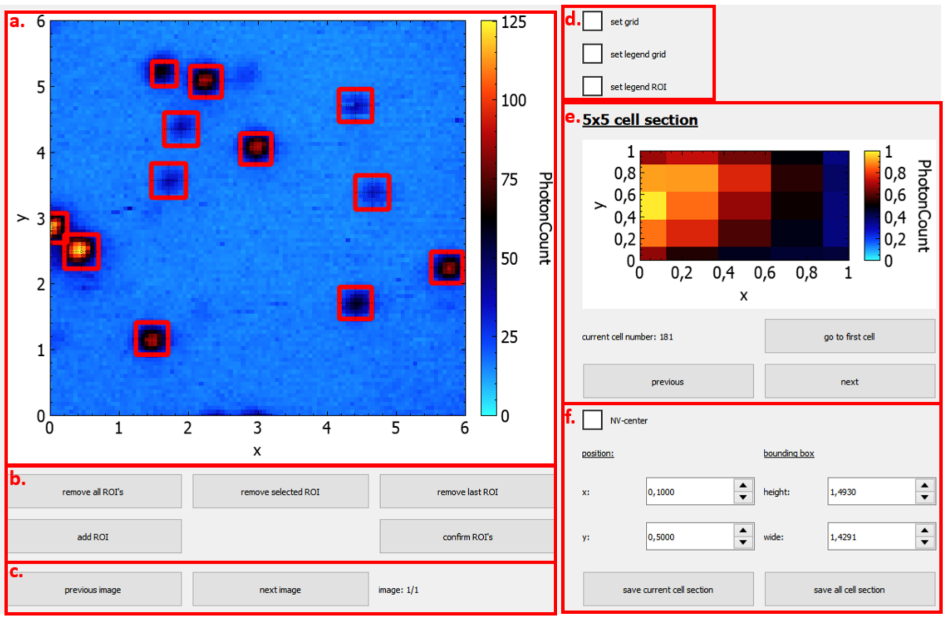


image: 1/1

current cell number: 181

x: 0,1000
y: 0,5000
height: 1,4930
wide: 1,4291

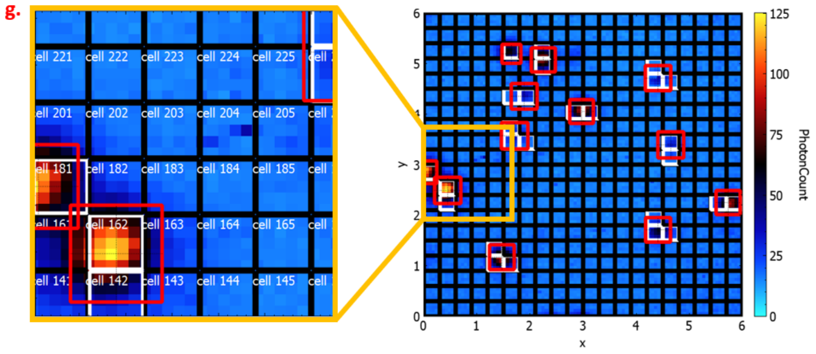


**Figure 56:** Window that opens when the Prepare Data button is used (above). The window is divided into 6 sections which shows the entire scan in a), various settings in b), c), d), a section of the scan in e) and the generation of the regression data in f). The lower part g) shows a close up section with specific settings.

This characterization follows the same principle as on the main page either by using the buttons in section 56 b) or the dropdown menu via the right mouse button. Another option is to use the already trained object recognition. The predicted ROIs are also transferred with the scan, which simplifies the process of creating training data. The final ROIs are then confirmed using the *confirm ROI's* button. Since the training data consists of 5x5 pixel sections, a grid can be overlaid for simplicity as shown in Figure 56 g). The settings shown in Figure 56 d) were changed to put a grid on the scan which shows the 5x5 pixel sections. Additionally, the grid legend was enabled to visualize the number of each cell. The last setting would display the ROI data but it was not enabled. In Figure 56 g) on the right, cells which containing a part of an NV-center are highlighted in white. By holding down the *Ctrl* and *A* keys, additional cells can be selected with the mouse. The selected cell is then shown in another QCustomPlot in Figure 56 e). Using the *next* and *previous* buttons the other white cells can be further be classified. The training data is then generated in Figure 56 f). The position and size data are taken from the corresponding ROI. It should be noted that the position is relative to the selected cell. Once the data is correct it can be saved individually or together using the *save current cell section* or *save all cell sections* buttons.
This setup for training data generation allows a fast creation of many NV-centers for the training data and theoretically enables the integration of an automated training process in the future.
The last section 56 c) is only used when a folder structure is loaded. In this case the buttons can be used to select between scans and go through them one by one.

## 8.3 Routines

The primary task of SoRA is the creation and the use of measurement routines. This allows sections with color centers on the diamond to be automatically analyzed for simplicity or to stay at the position constantly for long measurements using autofocus. This enables a more efficient work. The individual methods that SoRA can do are described below.

### 8.3.1 Autofocus

The most important and used method of SoRA is the autofocus. This is necessary for doing long measurements especially for composite pulses or measuring the $T_1$

time. Due to small fluctuations in enviroment temperature and microwaves during ODMR measurements, the NV-center may drift during a measurement. After just half an hour it may no longer be focused on the NV-center but on its surroundings resulting in a decrease in count rate. Therefore, it is necessary for the experimenter to realign the microscope. For measurements which go over several days, this can be a time-consuming and inefficient process. In this case SoRA can eliminate this issue. By automatically detecting NV-centers SoRA can realign the microscope after a certain time interval. In the case of measurements for composite pulses, the interval has been set to approximately 20 minutes.

To efficiently do this autofocus, two methods were tried. The simplest method used for measurement is the $XY$-adjustment. In this method, xy scans are taken at different heights to enable three-dimensional alignment. Since xz and yz scans are now possible, the second method is then taking three scans (xy, xz and yz) to perform a direct *3D*-adjustment. To test both options, a NV-center was used where the focus was positioned next to the NV-center. Then both methods were used to refocus on this NV-center. The results are shown in Figure 57 and 58.

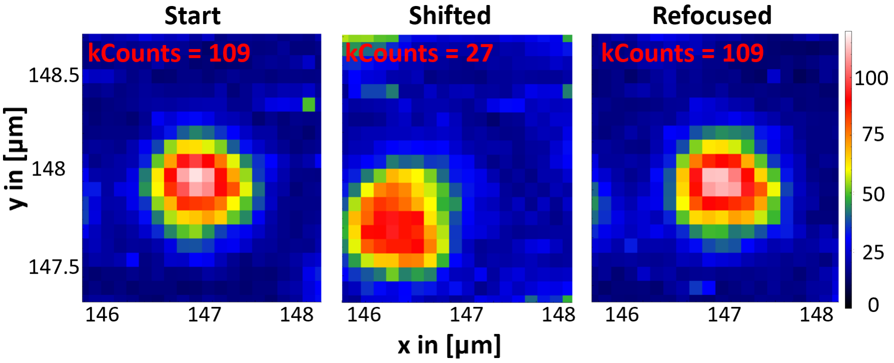


**Figure 57:** Autorefocus after shifting the NV-center using the $XY$-adjustment. The left scan shows the used center, the middle one shows the shifted position that needs to be refocused and the right one shows the result after refocusing.

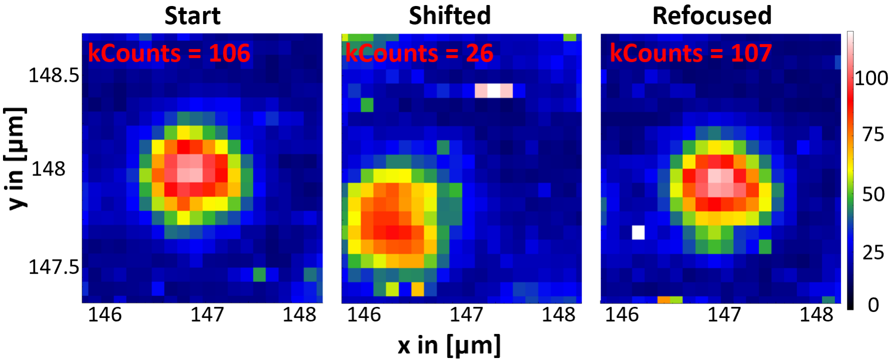
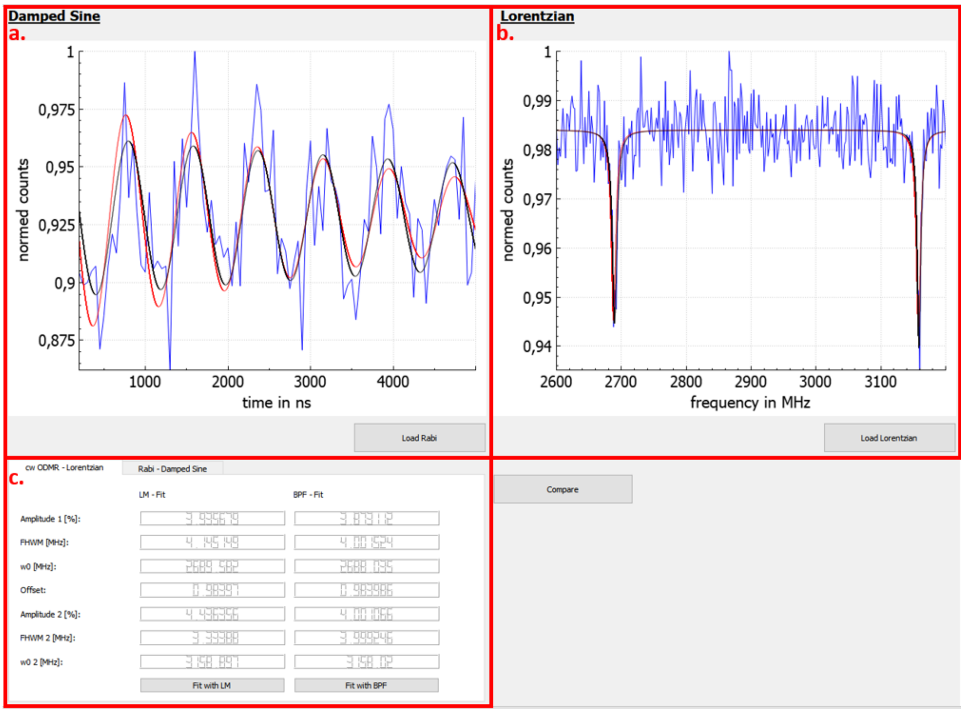


**Figure 58:** Autorefocus after shifting the NV-center using the *3D*-adjustment. The left scan shows the used center, the middle one shows the shifted position that needs to be refocused and the right one shows the result after refocusing.

It can be seen that both adjustment have focused very well on the NV-center.

Figure 57 and 58 show the initial scan, the shifted scan and the results of the two adjustment. In 57 the $XY$-adjustment and in 58 the *3D*-adjustment is shown. The most significant difference lies in the measurement time required for autofocus. In the $XY$-adjustment, it takes 32 seconds whereas in the *3D*-adjustment, it takes 43 seconds. This makes the $XY$-adjustment more relevant for the experiment but both methods can be accessed as functions within the program.

### 8.3.2 ODMR Measurment



| | LM - Fit | BPF - Fit |
|---|---|---|
| Amplitude 1 [%]: | 3.935679 | 3.879112 |
| FHWM [MHz]: | 4.145149 | 4.001524 |
| w0 [MHz]: | 2689.582 | 2688.035 |
| Offset: | 0.98397 | 0.983986 |
| Amplitude 2 [%]: | 4.436356 | 4.001066 |
| FHWM 2 [MHz]: | 3.33388 | 3.999246 |
| w0 2 [MHz]: | 3158.897 | 3158.02 |

**Figure 59:** Window to visualize the evaluation for the damped sine a) and lorentzian b). This consists of the graphs for the corresponding functions and a tab system for the fit parameters in c).

The second set of routines involves ODMR measurements. As described in the fundamentals, simple cw ODMR can be used for example to confirm ROIs as NV-centers or determine the alignment of NV-centers in the diamond. For this purpose, code for the measurement routines for ODMR measurements was written along with automatic movement of the magnets to adjust the orientation of the magnetic field. The measured data can then be analyzed using fit methods to obtain the parallel alignment of the magnetic field with the NV-axis or to perform other measurements such as Rabi-oscillations. The graphical evaluation is shown in Figure 59. This window consists of three parts: the graph for the damped sine a), the graph for the lorentzian b) and two tabs for the fitting functions with the fit parameters c). Both variants of fitting, the LM and the BPF methods, can be used for fitting. The resulting function is then visualized directly in the graph. In Figure 59 the black curve represents the LM method and the red curve represents the BPF method. They can be separately shown or hidden via a dropdown menu. Additionally, in the damped sine plot it can be switch between the Fourier transformation from the dropdown menu or the measured values.

The final section in Figure 59 c) shows the corresponding fit parameters for the different fitting methods in a tab system. The first tab represents both the single and double lorentzian and the second tab is for the damped sine fit.

## 8.4 Specific NV-Center Search

After the resonance frequency and Rabi-oscillation have been determined from the previous ODMR measurements and their evaluation, the time and frequency for a $\pi$-pulse can be determined. With this, a pulsed ODMR scan can be made which for example provides information about the nitrogen or carbon isotope. As mentioned in the fundamentals, a $^{14}$N isotope has a nuclear spin of 1 with a hyperfine splitting of $\sim$ 2.2 MHz while a $^{15}$N isotope has a nuclear spin of $\frac{1}{2}$ with a hyperfine splitting of $\sim$ 3 MHz. Therefore, a $^{15}$N isotope is better suited for a two-level system than a $^{14}$N isotope. But since the $^{15}$N isotrope is much rarer than a $^{14}$N isotope, with a natural abundance of $\sim$0.36%, a targeted search would be very time-consuming. By automating this process through SoRA this search can be simplified.
The same applies to the $^{13}$C isotope which has at least a higher probability than a $^{15}$N isotope. As already described in the fundamentals, a $^{13}$C isotope leads to a further characteristic splitting of the nuclear spin depending on the distance. Since the nuclear spin of the carbon isotope has a longer lifetime than that of nitrogen, it is also interesting for qubit measurements. Since the natural occurance is also low in these cases the automation process through SoRA also simplifies this.

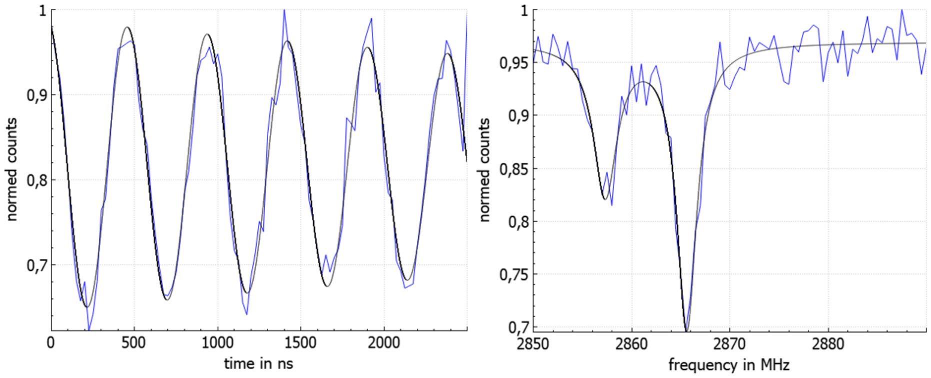


**Figure 60:** Recorded Rabi-oscillation to determine the $\pi$-pulse on the left and the resulting pulsed ODMR measurement on the right. The hyperfine splitting suggests a $^{13}$C isotope near the NV-center.

Figure 60 shows on the right side a measured pulsed ODMR spectrum form a NV-center with a $^{13}$C isotope. On the left side, the measured Rabi-oscillation is shown which was used to determine the $\pi$-pulse. In this case it is 278.33 ns. The measured pulsed ODMR spectrum has two peaks which are separated by a distance of $(8.30 \pm 0.17)$ MHz. From this, the distance between the NV-center and the $^{13}$C isotope can be determined which corresponds to a distance of 2.49 Å. The calculated literature value for this is a hyperfine splitting of 7.3 MHz.[34]

## 8.5 $T_1$ Measurement

Another measurement possibility is to measure the time duration of the lifetime for an NV-center initialized in the ground state $m_s$=0. This longitudinal relaxation time $T_1$ can be determined from the difference between the measurement with $\pi$-pulse and without. Depending on the settings, an accurate measurement of this time can take between 20 to 30 hours which strongly depends on the time stepsize and the number of steps. Figure 61 shows two such measurements each with a measurement time of approx. 22 hours and a step size of 800 ns.

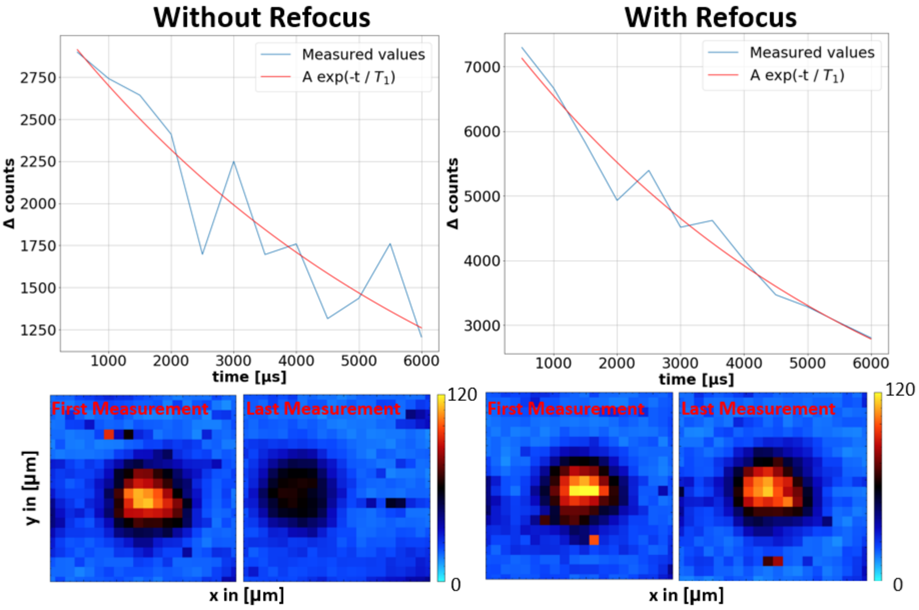


**Figure 61:** $T_1$ measurement with and without the refocus method as well as scans showing the focus before and after the $T_1$ measurement. An exponential function has been fitted to the blue values.

Figure 61 shows in addition to the $T_1$ measurement the influence of the autofocus on the measurement. In the lower part of the figure scans taken before and after the measurement are displayed to show the shifting during it. The first scan was taken before the ODMR measurement and manually adjusted to achieve the highest possible intensity. The second scan was taken directly after the measurement (22 hours) without any manual readjustment. It can be seen that in the case of the autofocus the intensity remained unchanged while without using the autofocus the intensity decreased significantly.
The decrease in intensity can be also seen in the diagrams which show the sum of the differences from 1 000 000 repetitions. Due to the fact that the count rate decreased from 106 kcounts to 47 kcounts without refocusing the signal is significantly weaker. This also explains the fluctuations, as the last measurements only partially capture the decay at the NV-center. To determine the $T_1$ time from this an exponential function is fitted to the measurement data. This leads to the following values for $T_1$ for the measurements:

| | |
|---|---|
| Without Refocus: | (6.56±0.90) ms |
| With Refocus: | (5.85±0.34) ms |

This results in an error range of 13.7% for the measurement without refocus and 5.8% for the measurement with refocus. In conclusion it can be said that the measurement with the refocus method has a significantly smaller error and makes the exponential decay of the curve more clearly visible.

# 9 Summary

The main focus of this masters thesis was the automation of the measurement process through the implementation of artificial intelligence. This was divided into two chapters: object detection, which deals with the search for ROI, and data analysis, which evalute the captured data for NV-centers and their properties.

**Object detection:** The first step towards automation was to teach the program to identify ROIs in a scan. For this purpose, three approaches were tested: a static method and two dynamic methods. The dynamic methods used the principle of neural networks. The thesis dealed with both, ANN and CNN for this task.
To achieve this, a dataset was initially created, which was then divided into training, validation, and test datasets. The neural networks were trained and tested using this dataset. Various parameters were tested, such as the number of hidden layers and neurons per hidden layer, activation functions, loss functions, number of filters, filter sizes, initialization parameters and with or without pooling, padding and bias. Additionally, two learning methods, the SGD and MBGD, were tested and compared. The CNN with the following parameters leads to the best results:

**Table 13:** Parameters for the final architecture of the CNN.

| | |
|---|---|
| Number of hidden layers: | 1 |
| Number of neurons in the hidden layer: | 10 |
| Activation function in the hidden layer: | ReLU |
| Number of neurons in the output layer: | 5 |
| Activation function in the output layer: | Leaky ReLU |
| Number of filters: | 4 |
| Size of the filters: | 3x3 |
| bias: | none |
| Training methode: | MBGD |
| YOLO size: | 5x5 |
| Prediction Threshold | 0.86 |
| Dirt and edges: | 525 |
| Too large ROI: | 815 |
| Distance prediction ↔ gt-Element | 475 |
| Low count (percentage): | 0.04 |
| Low count (surrounding Box): | 1000 |
| Low count (multiplication): | 1.1 |

**Data analysis:** The next part of the masters thesis focused on the data analysis of the ROIs determined through object detection. For this, two algorithms, the LM and the BPF, were integrated into the program, tested, and compared. Through this analysis, the program is then capable of confirming ROIs as NV-centers and examining them for measurement parameters such as orientation, Rabi oscillation, etc.

The final part of this thesis focuses on measurements. In one measurement, the implemented autofocus was used to measure the $T_1$ time. Special attention was given to the effect that autofocus can have on measurements. Another application was found in the search for centers with specific properties, as demonstrated in the measurement of a $^{13}$C isotope located at a distance of approximately 2.49 Åfrom the NV center.

# 10 Outlook

## 10.1 Object Recognition

The achieved accuracy of 0.9209 is a high value for object recognition. But in this case it is important to consider that many special cases were used for this score which pushed the AI to its limits. For individual scans, that do not have extreme conditions, the AI often recognizes all NV-centers, so the actual score for typical scans will likely be better. Because of this it did not make sense to further modify the AI to achieve better results. But of course it is possible that the architecture of object recognition could be improved for limit cases through modifications. This could involve testing different values for the parameters mentioned earlier which might lead to better results. Additionally, the AIs architecture could be modified. For example the output of the backbone could be split into an object part which only indicates probabilities and a regression part which indicates the position and size of the NV-center. The reason for such a split is the activation function. While a sigmoid function is ideal for probabilities a Leaky ReLU function is better for regression data. The AI could also be forced to set weights near zero to zero to get a more precisely focus. Furthermore, the postprocessor could be integrated in the network. For this, the thresholds could be introduced into the training process. But it should be noted that the network would then need access to information about the entire scan. It would be interesting to test whether these methods improve the AIs performance, particularly in extreme cases, where it achieves an accuracy of only 0.9209.

## 10.2 Data Analysis

Another important aspect is the evaluation of the measured ODMR data. Two methods were introduced to automate the evaluation and it was tested if an optimization, inspired by the backpropagation, provides better results than the classical solution. This was done both with and without using weights. The resulting data was then compared, evaluated and reoptimized based on the loss. Since the loss is important for all measurement values, the ROI for the measurement is not necessarily the highest-priority even when weights are used. It would be interesting to explore an alternative method for determining the loss in which the loss in specific regions (for example the dips of a cw ODMR) is given higher importance than in others. This could potentially lead to better parameters and quicker determination in the weighted region. It also could be tested whether it is more sensible to evaluate only specific regions using both the classical algorithm and the new method.
Another entirely different method would be to use a CNN with its own architecture similar to the object recognition method. But this would require a much larger number of measurements because determining the fit parameters is much more challenging and must be more accurate. Since the high number of data was not available the method was quickly discarded.

## 10.3 Measurement Routines

To make the AI do specific measurement routines, it was connected with the measurement instruments in the program in addition to object detection and evaluation. By using signals within the program of the AI, SoRA can interact with devices, such as the AWG, laser, piezo or magnetic stages to create additional routines. This allows an easy implementation of routines for example for a complete characterization including nuclear spin up to pulse sequences and other actions.
Existing routines can also be further improved. For example a three-dimensional

Gaussian function has been implemented to determine the position of the NV-center more precisely. But since this always requires high resolution and the high accuracy is not necessary yet, it would just increase the time required for the AI to make predictions. So it was decided not to include this into the object recognition but it still exists in the code.

# List of Abbreviations

| | |
|---|---|
| **ANN** | Artificial Neural Network |
| **BGD** | Batch Gradient Descent |
| **SGD** | Stochastic Gradient Descent |
| **MBGD** | Mini-Batch Gradient Descent |
| **CNN** | Convolutional Neural Network |
| **SWOD** | Sliding Window Object Detection |
| **R-CNN** | Region-based CNN |
| **Fast R-CNN** | Fast Region-based CNN |
| **Faster R-CNN** | Faster Region-based CNN |
| **YOLO** | You only look once |
| **RPN** | Region Proposal Network |
| **NMS** | Non-Maximum Suppression |
| **TP** | True Positive |
| **FP** | False Positiv |
| **TN** | True Negativ |
| **FN** | False Negativ |
| **NV** | Nitrogen-vacancy |
| **ODMR** | Optical Detected Magnetic Resonance |
| $T_1$ | Longitudinal relaxation time |
| **WPH** | Half-wave plate |
| **WPQ** | Quarter-wave plate |
| **PBS** | Polarizing beam splitter |
| **APD** | Avalanche photodiode |
| **AWG** | Arbitrary waveform generator |
| **PCB** | Diamond holder |
| **ROI** | Region of Interest |
| **GT** | Ground truth |
| **SoRa** | Scan of ROI anomaly |
| **LM** | Levenberg-Marquardt |
| **BPF** | Backpropagation Fitting |
| **FWHM** | Full width at half maximum |
| **SoRA** | Scan of ROI AI |

# References

[1] Bornmann L, Haunschild R, Mutz R. Growth rates of modern science: a latent piecewise growth curve approach to model publication numbers from established and new literature databases. Humanities and Social Sciences Communications. 2021 Oct 7;8(1). doi: https://doi.org/10.1057/s41599-021-00903-w.

[2] Zhang H, Gu M, Jiang XD, Thompson J, Cai H, Paesani S, et al. An optical neural chip for implementing complex-valued neural network. Nature Communications. 2021 Jan 19;12(1):457. doi: https://doi.org/10.1038/s41467-020-20719-7.

[3] Plettenberg P, Bauerhenne B, García ME. Neural network interatomic potential for laser-excited materials. Communications materials. 2023 Aug 15;4(1). doi: https://doi.org/10.1038/s43246-023-00389-w.

[4] Schott S. Comparison of a beta-variational autoencoder and a custom convolutional neural network for surrogate models approximating METRIXS beamline of BESSY II [Bachelor thesis]. [Universität Kassel]; 2023.

[5] Goodfellow I, Bengio Y, Courville A. Deep Learning. Cambridge, Massachusetts: The Mit Press; 2016.

[6] Chi Leung Hui P, editor. Artificial Neural Networks - Recent Advances, New Perspectives and Applications. IntechOpen; 2023.

[7] Salman Khan, Hossein Rahmani, Shah A, M Bennamoun. A guide to convolutional neural networks for computer vision. San Rafael, California: Morgan & Claypool Publishers; 2018.

[8] Understanding Learning Rates and How It Improves Performance in Deep Learning [Internet]. Medium. Towards Data Science; 2018. [cited 2021 Jul 30]. Available from: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10

[9] Intro to optimization in deep learning: Gradient Descent [Internet]. Paperspace Blog; 2018. [cited 2021 Jul 30]. Available from: https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

[10] Ng A. CS229: Machine Learning [Internet]. Ma T, editor. Stanford University; 2022. Available from: https://cs229.stanford.edu/lectures-spring2022/main_notes.pdf

[11] Marc Peter Deisenroth, A Aldo Faisal, Cheng Soon Ong. Mathematics for machine learning. Cambridge, United Kingdom ; New York, Ny: Cambridge University Press; 2020.

[12] Rojas R. Neural Networks. Springer Science & Business Media; 2013.

[13] Prakash J. Back-Propagation is very simple. Who made it Complicated? [Internet]. Medium; 2017 [cited 2021 Aug 5]. Available from: https://medium.com/@14prakash/back-propagation-is-very-simple-who-made-it-complicated-97b794c97e5c

[14] MIT Introduction to Deep Learning | 6.S191 [Internet]. www.youtube.com. Available from: https://www.youtube.com/watch?v=QDX-1M5Nj7s&list=PLtBw6njQRU-rwp5___7C0oIVt26ZgjG9NI&index=2

[15] What is Overfitting? | IBM [Internet]. www.ibm.com. [cited 2021 Aug 6]. Available from: https://www.ibm.com/topics/overfitting#: :text=In %20both%20scenarios%2C%20the%20model

[16] Jiang X, Hadid A, Pang Y, Granger E, Feng X. Deep learning in object detection and recognition. Singapore: Springer Singapore Pte. Limited; 2019.

[17] Kim S. A Beginner's Guide to Convolutional Neural Networks (CNNs) [Internet]. Medium. Towards Data Science; 2019. [cited 2021 Oct 3]. Available from: https://towardsdatascience.com/a-beginners-guide-to-convolutional-neural-networks-cnns-14649dbddce8

[18] Smartphones Galaxy S online kaufen [Internet]. Samsung de. [cited 2021 Oct 4]. Available from: https://www.samsung.com/de/smartphones/galaxy-s21-ultra-5g/

[19] Saha S. A Comprehensive Guide to Convolutional Neural Networks [Internet]. Medium. Towards Data Science; 2018. [cited 2021 Oct 6]. Available from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[20] Jefkine. Backpropagation In Convolutional Neural Networks [Internet]. DeepGrid; 2016. [cited 2021 Nov 9]. Available from: https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

[21] Dumoulin V, Visin F. A guide to convolution arithmetic for deep learning [Internet]. 2018. Available from: https://arxiv.org/pdf/1603.07285.pdf

[22] LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, et al. Backpropagation Applied to Handwritten Zip Code Recognition. Neural Computation. 1989 Dec;1(4):541–51. doi: 10.1162/neco.1989.1.4.541

[23] Bhattacharyya, S, Snasel V, Hassanien AE, Saha S. Deep learning : research and applications. De Gruyter Frontiers in Computational Intelligence. Berlin: De Gruyter; 2020.

[24] Xiang Z, Zhang R, Seeling P. Machine learning for object detection. Computing in Communication Networks. 2020;325–38. doi: 10.1016/B978-0-12-820488-7.00034-7

[25] Selective Search for Object Detection (C++ / Python) | LearnOpenCV [Internet]. 2017. [cited 2021 Nov 18]. Available from: https://learnopencv.com/selective-search-for-object-detection-cpp-python/

[26] Srivastava S, Divekar A V, Anilkumar C, Naik I, Kulkarni V, Pattabiraman V. Comparative analysis of deep learning image detection algorithms. Journal of Big Data. 2021 May 10;8(1), doi: https://doi.org/10.1186/s40537-021-00434-w

[27] Gad A F. Faster R-CNN Explained for Object Detection Tasks [Internet]. Paperspace Blog; 2020. [cited 2021 Nov 18]. Available from: https://blog.paperspace.com/faster-r-cnn-explained-object-detection/

[28] Quinn J, Mceachen J, Fullan M, Mag Gardner, Drummy M. Dive into deep learning : tools for engagement. Thousand Oaks, California: Corwin, A Sage Company; 2020.

[29] Brinkmann N. Objekterkennung in natürlichen Szenen mittels Region-based Convolutional Neural Networks [Internet] [Bachelor Thesis]. [Universität Dortmund]; 2016. [cited 2021 Nov 18]. Available from: https://web.patrec.cs.tu-dortmund.de/pubs/theses/ba_brinkmann.pdf

[30] Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence [Internet]. 2017 Jun 1;39(6):1137–49. Available from: https://ieeexplore.ieee.org/document/7485869, doi: 10.1109/T-PAMI.2016.2577031

[31] Redmon J, Divvala S, Girshick R, Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [Internet]. Las Vegas, NV, USA, 2016, pp. 779-788, 2016; Available from: https://ieeexplore.ieee.org/document/7780460, doi: 10.1109/CVPR.2016.91.

[32] Danzer A. Object detection in high-resolution radar data [Internet] [Dissertation]. [Universität Ulm]; 2022 [cited 2023 Oct 12]. Available from: https://oparu.uni-ulm.de/xmlui/handle/123456789/45995

[33] Lich J. Festkörperbasierte Einzelphotonenquelle in Diamant [Internet] [Diplomarbeit]. [Universität München]; 2008 [cited 2023 Oct 13]. Available from: https://xqp.quantum.physik.uni-muenchen.de/publications/theses_diplom/diplom_lich.html

[34] Neumann P. Towards a room temperature solid state quantum processor — The nitrogen-vacancy center in diamond. [Internet] [Dissertation]. [Universität Stuttgart]; 2011 [cited 2023 Oct 13]. Available from: http://dx.doi.org/10.18419/opus-5067

[35] Lesik M. Engineering von NV-Farbzentren in Diamant für ihre Anwendungen in der Quanteninformation und Magnetometrie [Internet] [Dissertation]. [Université Paris-Sud]; 2015 [cited 2023 Oct 15]. Available from: https://theses.hal.science/tel-01158995

[36] Doherty M W, Manson N B, Delaney P, Jelezko F, Wrachtrup J, Hollenberg L C L. The nitrogen-vacancy colour centre in diamond. Physics Reports. 2013 Jul;528(1):1–45, doi:https://doi.org/10.1016/j.physrep.2013.02.001

[37] Felton S, Edmonds A M, Newton M E, Philip Maurice Martineau, Fisher D A, Twitchen D J. Electron paramagnetic resonance studies of the neutral nitrogen vacancy in diamond. 2008 Feb 15;77(8), doi:10.1103/PhysRevB.77.081201

[38] Jacques V, Neumann P, Beck J, Markham M, Twitchen D, Meijer J, et al. Dynamic Polarization of Single Nuclear Spins by Optical Pumping of Nitrogen-Vacancy Color Centers in Diamond at Room Temperature. Physical Review Letters. 2009 Feb 6;102(5), doi: 10.1103/PhysRevLett.102.057403

[39] Plate R-J. Aufbau eines automatisierten Magnetpositionierungssystems zur präzisen Ausrichtung auf die Quantisierungsachse von Farbzentren. [Internet] [Bachelor thesis]. [Universität Kassel]; 2020 [cited 2023 Sep 10]. Available from: https://www.uni-kassel.de/fb10/institute/physik/forschungsgruppen/licht-materie-wechselwirkung/abschlussarbeiten

[40] Fedotov I V, Doronina-Amitonova L V, Voronin A A, Levchenko A O, Zibrov S A, Sidorov-Biryukov D A, et al. Electron spin manipulation and readout through an optical fiber. Scientific Reports [Internet]. 2014 Jul 16 [cited 2023

Oct 15];4(1):5362. Available from: https://www.nature.com/articles/srep05362, doi: https://doi.org/10.1038/srep05362

[41] Thieme J. Erweiterung des Farbzentrenaufbaus zur flexiblen Magnetresonanzmessung von Farbzentren. [Internet] [Bachelor thesis]. [Universität Kassel]; 2020 [cited 2023 Sep 10]. Available from: https://www.uni-kassel.de/fb10/institute/physik/forschungsgruppen/licht-materie-wechselwirkung/abschlussarbeiten

[42] Kull, H.-J. Laserphysik : physikalische Grundlagen des Laserlichts und seine Wechselwirkung mit Materie. Muenchen: Oldenbourg; 2010.

[43] Marlan Orvil Scully, Muhammad Suhail Zubairy. Quantum optics. Cambridge: Cambridge University Press; 2008.

[44] Singer K, Dawkins S, Crivelli D. Lecture Notes of Nanoscale Quantum Optics I [Internet]. Universität Kassel; 2018.

[45] Nielsen MA, Chuang IL. Quantum computation and quantum information. Cambridge Cambridge University Press; 2019.

[46] Dréau A, Lesik M, Rondin L, Spinicelli P, Arcizet O, Roch JF ., et al. Avoiding power broadening in optically detected magnetic resonance of single NV defects for enhanced dc magnetic field sensitivity. Physical Review B. 2011 Nov 23;84(19), doi: 10.1103/PhysRevB.84.195204

[47] Poulsen A F L, Webb J L, Berg-Sørensen K, Andersen U L, Huck A. Investigation and comparison of measurement schemes in the low frequency biosensing regime using solid-state defect centers. ReasearchGate; 2021. Available from: https://www.researchgate.net/publication/354889678_Investigation_and_comparison_of_measurement_schemes_in_the_low_frequency_biosensing_regime_using_solid-state_defect_centers

[48] Edwin Demuth Becker. High resolution NMR [nuclear magnetic resonance]. 1969.

[49] Torosov BT, Vitanov NV. Smooth composite pulses for high-fidelity quantum information processing. Physical Review A. 2011 May 23;83(5), doi: 10.1103/PhysRevA.83.053420

[50] Mrózek M, Rudnicki D, Kehayias P, Jarmola A, Budker D, Gawlik W. Longitudinal spin relaxation in nitrogen-vacancy ensembles in diamond. EPJ Quantum Technology. 2015 Oct 29;2(1), doi: https://doi.org/10.1140/epjqt/s40507-015-0035-z

[51] ALGLIB - C++/C#/Java numerical analysis library [Internet]. www.alglib.net. [cited 2023 Sep 5]. Available from: https://www.alglib.net/