

GOTTFRIED WILHELM LEIBNIZ UNIVERSITÄT HANNOVER  
INSTITUT FÜR INFORMATIONSVERARBEITUNG

Bachelorarbeit

**Lokalisierung von Windenergieanlagen aus  
Satellitenbildern mittels Convolutional Neural  
Networks**

Hauke Tristan Hinrichs

Matrikelnr. 10007270

Betreuer: Dipl.-Inf. D. Gritzner  
Erstprüfer: Prof. Dr.-Ing. B. Rosenhahn  
Zweitprüfer: Prof. Dr.-Ing. J. Ostermann

Hannover, August 2020

---

Die Richtlinien im *Merkblatt zur Ausführung von Abschlussarbeiten am Institut für Informationsverarbeitung* sind Bestandteil der Aufgabenstellung. Eine anderweitige Verwendung der Arbeit, insbesondere die Bekanntgabe an Dritte, bedarf der Zustimmung des Erstprüfers.



Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Ich stimme der Verwertung meiner Arbeit durch das Institut in den folgenden Punkten zu:

- Aufnahme der gedruckten und der elektronischen Fassung der Arbeit in die Institutsbibliothek,
- Vervielfältigung der gesamten Arbeit oder von Auszügen für Lehrzwecke und
- Wiedergabe der Arbeit durch Bild- und Tonträger

Hannover, den 24.08.2020



# **Abstract**

Forschungen im Bereich der Windenergie benötigen häufig Standortdaten von Windenergieanlagen, um bspw. politisch-ökonomische Handlungsempfehlungen auf Basis einer Stromertragsanalyse der Windflotte eines Landes zu erforschen. Öffentlich einsehbare, genaue und lückenlose Datenbanken existieren teilweise auf nationaler Ebene, stellen aber eine Seltenheit dar. Das Ziel dieser Arbeit ist es zu erforschen inwiefern eine Lokalisierung von Anlagenstandorten aus Satellitenbildern mit einer Objektdetektierung aus der Computer Vision realisierbar ist. Die benötigten Trainingsdaten werden über die Google Maps Static API im Zusammenspiel mit einer Anlagendatenbank aus Deutschland automatisch erzeugt. Es werden verschiedene Modifikationen des Faster R-CNN Modells trainiert und mit gängigen Metriken evaluiert. Eine Hypothese ist, dass bereits das Region Proposal Network ausreicht, die Problemstellung zu lösen. Durch die Ergebnisse kann diese Hypothese widerlegt werden. In Kombination mit dem Classifier der Faster R-CNN Architektur können dagegen zufriedenstellende Ergebnisse erzielt werden. 86% der Anlagenstandorte im Testset werden, mit einer mittleren Abweichung von 13,4 Metern zum tatsächlichen Standort, erkannt. In einer anderen Konfiguration erkennt das Modell 74% der Standorte bei einer mittleren Abweichung von 7,9 Metern. Die Prädiktionen beider Konfigurationen stellen dabei lediglich in 3% der Fälle eine Fehldetektion dar. Im Anschluss wird mithilfe von Daten aus den USA eine eventuelle geographische Abhängigkeit untersucht und durch die Ergebnisse bestätigt.



# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>v</b>
<b>Tabellenverzeichnis</b>	<b>vii</b>
<b>Akronyme</b>	<b>ix</b>
<b>1. Einleitung</b>	<b>1</b>
1.1. Motivation . . . . .	1
1.2. Ziel . . . . .	2
1.3. Aufbau . . . . .	3
<b>2. Theoretische Grundlagen</b>	<b>5</b>
2.1. Artificial Neural Networks . . . . .	5
2.1.1. Perzeptron . . . . .	5
2.1.2. Aktivierungsfunktion . . . . .	6
2.1.3. Mehrlagiges Perzeptron . . . . .	7
2.1.4. Verlustfunktion . . . . .	8
2.1.5. Backpropagation . . . . .	9
2.1.6. Overfitting . . . . .	12
2.2. Convolutional Neural Networks . . . . .	14
2.2.1. Convolution . . . . .	15
2.2.2. Convolutional Layer . . . . .	16
2.2.3. Max Pooling Layer . . . . .	16
2.3. Objektdetektion mit CNNs . . . . .	17
2.4. Faster R-CNN . . . . .	19
2.4.1. Region Proposal Network . . . . .	19
2.4.2. RoI Pooling . . . . .	21
<b>3. Lösungsansatz</b>	<b>23</b>
3.1. Daten . . . . .	23
3.1.1. Datenerhebung (DEU) . . . . .	23
3.1.2. Evaluation der Datenqualität (DEU) . . . . .	25
3.1.3. Datenerhebung (USA) . . . . .	27
3.1.4. Augmentation . . . . .	28
3.1.5. Normalisierung . . . . .	28

---

3.2. Implementierung . . . . .	29
3.2.1. Frameworks und Bibliotheken . . . . .	29
3.2.2. Netzarchitektur . . . . .	29
3.2.3. Modifikation des Faster R-CNN . . . . .	31
<b>4. Evaluation</b>	<b>33</b>
4.1. Metriken . . . . .	33
4.2. Experimente . . . . .	35
4.2.1. Modifiziertes RPN allein . . . . .	35
4.2.2. RPN allein mit variablen Box Größen . . . . .	36
4.2.3. RPN + Classifier mit variablen Box Größen . . . . .	38
4.2.4. RPN + Classifier mit 3 diskreten Box Größen . . . . .	42
4.2.5. Geographische Abhängigkeit am Beispiel der USA . . . . .	44
<b>5. Zusammenfassung und Ausblick</b>	<b>47</b>
<b>Literaturverzeichnis</b>	<b>49</b>
<b>A. Hyperparameter und sonstige Werte für das Training</b>	<b>55</b>

# Abbildungsverzeichnis

1.1.	Windpark in Deutschland . . . . .	2
2.1.	Nervenzelle und Perzeptron . . . . .	6
2.2.	Aktivierungsfunktionen . . . . .	7
2.3.	Mehrlagiges Perzeptron . . . . .	8
2.4.	Backpropagation Kettenregel 1 . . . . .	10
2.5.	Backpropagation Kettenregel 2 . . . . .	11
2.6.	Overfitting . . . . .	13
2.7.	Overfitting - Übertraining . . . . .	13
2.8.	Convolutional Neural Network Beispiel . . . . .	14
2.9.	Convolution Beispiel . . . . .	15
2.10.	Max Pooling Beispiel . . . . .	16
2.11.	Faster R-CNN Architektur . . . . .	19
2.12.	Region Proposal Network Architektur . . . . .	20
2.13.	RoI Pooling Beispiel . . . . .	21
3.1.	Bildvergleich: Sentinel-2 und Google Maps Static API . . . . .	24
3.2.	Fehler in den Trainingsdaten . . . . .	25
3.3.	Verteilung der Fehler in den Trainingsdaten in drei verschiedenen Samples	26
3.4.	Histogramm der Verluste über alle Bilder eines trainierten RPN . . . . .	27
3.5.	Beispiel Windpark USA . . . . .	27
3.6.	Residual Learning . . . . .	29
3.7.	ResNet Convolutional Block . . . . .	30
4.1.	Schematische Darstellung einer WEA . . . . .	34
4.2.	Histogramm der Anlagengrößen . . . . .	35
4.3.	Verlustkurve des modifizierten RPN . . . . .	36
4.4.	Qualitative Ergebnisse: RPN allein mit variablen Box Größen . . . . .	39
4.5.	Qualitative Ergebnisse: RPN + Classifier mit variablen Box Größen . . . . .	41
4.6.	Qualitative Ergebnisse: RPN + Classifier mit 3 diskreten Box Größen . . . . .	43
4.7.	Qualitative Ergebnisse: Geographische Abhängigkeit am Beispiel der USA	45



# Tabellenverzeichnis

4.1. Ergebnisse RPN allein mit variablen Box Größen . . . . .	37
4.2. Ergebnisse RPN + Classifier mit variablen Box Größen . . . . .	40
4.3. Ergebnisse RPN + Classifier mit 3 diskreten Box Größen . . . . .	42
4.4. Ergebnisse RPN + Classifier mit 3 diskreten Box Größen für Daten aus den USA ohne Transfer Learning . . . . .	44
4.5. Ergebnisse RPN + Classifier mit 3 diskreten Box Größen für Daten aus den USA mit Transfer Learning . . . . .	44



# Akronyme

**ANN** Artificial Neural Network.

**AP** Average Precision.

**CL** Convolutional Layer.

**CNN** Convolutional Neural Network.

**FN** False Negative.

**FP** False Positive.

**GTB** Ground Truth Box.

**IoU** Intersection over Union.

**MLP** Mehrlagiges Perzepron.

**MPL** Max Pooling Layer.

**NMS** Non-Maximum Suppression.

**R-CNN** Region-Based Convolutional Neural Network.

**RoI** Region of Interest.

**RPN** Region Proposal Network.

**TN** True Negative.

**TP** True Positive.

**WEA** Windenergieanlage.



# 1. Einleitung

## 1.1. Motivation

Einige Forschungen im Bereich der Windenergie aus den letzten Jahren benötigen Standortdaten von Windergieanlagen (WEA). Darunter fallen bspw. Wind- und Energieertragssimulationen [1][2]. Aufbauend auf diesen Simulationen werden außerdem politisch-ökonomische Handlungsempfehlungen erforscht [3][4]. Weitere Themen sind Flächenanalysen zur Feststellung von Repoweringpotentialen<sup>1</sup> [5], sowie Lebenszyklusbewertungen<sup>2</sup> von Windflotten in Deutschland [6], welche auf Grundlage von Ertragsimulationen und Flächenanalysen erforscht werden. Wie in [2] beschrieben wird, weisen weltweite WEA-Datenbanken große Lücken und Ungenauigkeiten auf. Es wird teilweise mit großem Aufwand versucht diese zu verringern, um den Analysen Aussagekraft zu geben. Eine öffentlich einsehbare, konsistente, vollständige und genaue WEA-Datenbank würde der Forschung helfen. Es gibt nationale Bestrebungen solche Datenbanken aufzubauen. Für die USA existiert eine solche bereits [7]. In Deutschland gibt es seit 2019 das Markstammdatenregister [8]. Ein großes Problem ist allerdings, dass die verpflichtenden Angaben der WEA-Betreiber in das Register nur sehr langsam geprüft werden. Von insgesamt 51.289 Einträgen sind zum Datum dieser Arbeit (17.08.2020) 8.993 geprüft. Zwischen den gewissenhaften Einträgen vieler Betreiber tauchen auch viele fehlerhafte Daten und Standorte auf.

Firmen der Windenergie-Branche, die bereits brauchbare Datenbanken besitzen, haben einen hohen Aufwand diese aktuell zu halten. Kontinuierlich werden alte Anlagen demontiert und es kommen neue hinzu, wie bspw. in der Aufnahme eines deutschen Windparks nach einem Repowering in Abb. 1.1 zu sehen ist. Wünschenswert wäre demnach eine Möglichkeit die Pflege einer bestehenden Datenbank so zu vereinfachen, dass sie möglichst wenig Zeit und Ressourcen in Anspruch nimmt.

---

<sup>1</sup>Repowering ist das Austauschen alter WEA gegen leistungsfähigere.

<sup>2</sup>Was unter Lebenszyklus zu verstehen ist: In Deutschland gewährt das Erneuerbare-Energien-Gesetz (EEG) neuen WEA Förderung über eine Dauer von 20 Jahren. Nach Ablauf dieser Periode stehen Betreiber vor der ökonomisch schwierig zu beantwortenden Frage: Stilllegung, Repowering oder Weiterbetrieb?



Abbildung 1.1.: In gelb: alte Standorte von WEA

## 1.2. Ziel

Aus der Motivation lassen sich zwei Problemstellungen ableiten. Einerseits soll geprüft werden, ob alte WEA Standorte weiterhin korrekt sind. Standorte können falsch sein, indem sie entweder bereits fehlerhaft in der Datenbank sind oder durch den Rückbau einer Anlage fehlerhaft werden. Andererseits gibt es ein Interesse neue Standorte von WEA zu lokalisieren. Diese Probleme könnten mit der Objektklassifizierung, sowie Objektdetektierung aus der Computer Vision adressiert werden. Klassifizierung hat zum Ziel das dominante Objekt eines Bildes zu ermitteln. Bei der Detektierung wird versucht den Umriss eines oder mehrerer Objekte im Bild, sowie seine bzw. ihre Klassen zu ermitteln. Der Umriss nicht exakt der Standort, also der Mittelpunkt der Anlage, der gesucht ist. Eine Möglichkeit ist es den Mittelpunkt des Umrisses zu verwenden, eine andere die Objektdetektierung so zu modifizieren, dass sie statt Umrissen nur noch eine Koordinate im Bild liefert. Mehr hierzu folgt im Abschnitt 3.2.3.

Artificial Neural Networks (ANNs), insbesondere Convolutional Neural Networks (CNNs) in Verbindung mit der Methode des Deep Learning, haben in den letzten Jahren in der Computer Vision zu einer erheblichen Verbesserung der Performanz beigebracht [9]. Anstatt wie zuvor mit per Hand ausgewählten Merkmalen, erkennen diese Netze Objekte anhand von Merkmalen, die sie in einem überwachten Trainingsprozess selbstständig erlernen. Heutzutage finden sich etliche Anwendungen dieser Technik in den verschiedensten Bereichen unseres Lebens wieder.

Das Ziel dieser Arbeit ist es, zu untersuchen ob mit Modellen, basierend auf CNNs, die Standorte von WEA aus Satellitenbildern in Deutschland ermittelt werden können. Dafür steht eine Datenbanken aus Deutschland zur Verfügung. Weiterhin soll im Anschluss die geographische Abhängigkeit des Modells untersucht werden, um abzuschätzen inwieweit das Modell von Deutschland auf andere Regionen übertragbar ist. Dies wird mit der bereits existierenden öffentlichen Datenbank aus den USA [7] realisiert.

## 1.3. Aufbau

Zunächst wird die grundlegende Funktionsweise von ANNs und CNNs erklärt, gefolgt von der Vorstellung des verwendeten Modells zur Objektdetektierung. Anschließend wird erläutert wie vorgegangen wird, um Bilder und Daten für das Training zu sammeln. In Experimenten mit verschiedenen Konfigurationen und Modifikationen des Modells wird dann mithilfe von gängigen Metriken versucht herauszufinden, inwiefern die Detektionsrate optimiert werden kann.



## 2. Theoretische Grundlagen

### 2.1. Artificial Neural Networks

Als ANN wird eine Familie von Modellen des Maschinellen Lernens bezeichnet. Ursprünglich waren sie an natürliche Neuronennetze angelehnt, wie sie bspw. im menschlichen Gehirn zu finden sind. ANNs sind keineswegs eine neue Idee. Erstmals wurden sie 1943 beschrieben. Eine größere Rolle in der Forschung spielten sie allerdings erst wieder ab 1986 [10]. Sie ermöglichen es eine beliebig komplizierte Funktion zu approximieren [11]. Dazu müssen sie in einem überwachten Lernverfahren mit einer Vielzahl an Trainingsdaten konfrontiert werden. Zunächst wird der Aufbau, sowie die Grundbausteine eines ANN betrachtet. Im Anschluss wird beschrieben wie Trainingsdaten beispielhaft aussehen können und wie das Netz mithilfe dieser Daten lernen kann.

#### 2.1.1. Perzeptron

Das Perzeptron stellt die Grundeinheit eines ANN dar. Prinzipiell ist es allein schon ein Beispiel für ein minimales ANN. Es ist einer Nervenzelle (s. Abb. 2.1 links) nachempfunden. Im Gehirn gelangen elektrische Signale über die Dendriten in die Nervenzelle. Davor müssen die Signale noch durch die Synapsen. Diese können das Signal wahlweise verstärken oder abschwächen, was im Grunde den Lernprozess in Gehirn darstellt. In der Nervenzelle werden die Signale aufsummiert und je nach Intensität, die sich ergibt, leitet die Zelle ein Signal über das Axon weiter oder nicht [12]. Dieser Prozess lässt sich fast eins zu eins auf das Perzeptron (s. Abb. 2.1 rechts) übertragen. Die Eingänge sind die Dendriten. Hier kommen beliebige Werte (Signale) an. Die Gewichte an den Kanten sind demnach die Synapsen, die den Wert durch eine Multiplikation erhöhen oder verringern können. Die gewichtete Summe wird durch einen Bias ergänzt und dient als Eingabe für eine Aktivierungsfunktion, welche die Ausgabe erzeugt. Welche Aktivierungsfunktionen an dieser Stelle zum Einsatz kommen können, wird in Abschnitt 2.1.2 erläutert. Berechnen lässt sich die Ausgabe des Perzeptrons folgendermaßen: Die Eingänge seien  $x = (x_1, \dots, x_n)$ , die Gewichte  $w = (w_1, \dots, w_n)$ , der Bias  $b$ , die Aktivierungsfunktion  $f$  und die Ausgabe des Perzeptrons  $y$ .

$$y = f(b + \sum_{i=1}^n x_i w_i) \quad (2.1)$$

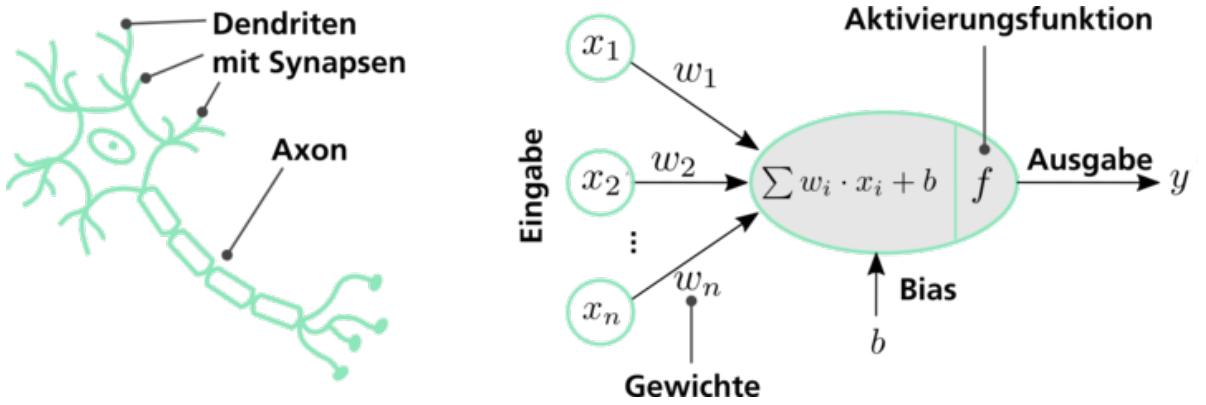


Abbildung 2.1.: Links: Nervenzelle (Neuron), Rechts: Perzeptron

In Formel (2.1) wird demnach das Skalarprodukt von  $x$  und  $w^\top$  berechnet. Die Berechnung kann noch kompakter beschrieben werden. Mittels einer Erweiterung des Eingabevektors um eine Dimension wie folgt  $x = (1, x_1, \dots, x_n)$ , kann der Bias mit in den Gewichtsvektor aufgenommen werden  $w = (b, w_1, \dots, w_n)$ . Wenn im Folgenden von den Gewichten die Rede ist, sind damit immer die Gewichte zusammen mit dem Bias gemeint. Es ergibt sich die kompaktere Schreibweise:

$$y = f(x \cdot w^\top) \quad (2.2)$$

### 2.1.2. Aktivierungsfunktion

Es gibt eine Reihe unterschiedlicher Aktivierungsfunktionen, die in der Praxis Anwendung finden. An dieser Stelle werden nur einige beispielhaft angeführt, sowie ihre Wirkung auf die Ausgabe der Perzepronen erläutert. Prinzipiell ist der Gedanke mit der Aktivierungsfunktion eine Nichtlinearität in die Berechnung einzubringen. Durch diese Nichtlinearität wird dem Netz ermöglicht, nicht nur lineare Funktionen abzubilden. Wird eine lineare Aktivierungsfunktion verwendet, kann jede beliebig komplizierte Netzarchitektur durch eine äquivalente einschichtige Architektur ersetzt werden [13].

Die **Sigmoid Funktion** (s. Abb. 2.2 links):

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.3)$$

Diese Funktion beschränkt die Ausgabe des Perzepronons auf den Bereich  $(0, 1)$ . Der Gradient dieser Funktion wird zu den Rändern hin immer kleiner, wodurch ein Problem beim Trainieren des Netzes auftauchen kann. Genaueres dazu folgt in Abschnitt 2.1.5. Ein Vorteil der Sigmoid Funktion ist, dass ihre Ableitung die Funktion wieder selbst enthält und dadurch einfach zu berechnen ist:  $\sigma'(x) = \sigma(x)(1 - \sigma(x))$

Die **Rectified Linear Unit**, kurz **ReLU** (s. Abb. 2.2 rechts):

$$R(x) = \max(0, x) \quad (2.4)$$

Diese Funktion ist lediglich im Punkt 0 nicht linear. Dennoch hat sie in der Praxis gezeigt, dass mit ihr state-of-the-art Ergebnisse erzielt werden können [14]. Das Problem eines kleiner werdenden Gradienten taucht hier, im Vergleich zur Sigmoid Funktion, nicht auf. Der Gradient ist stets 1 für positive und 0 für negative Werte. Dementsprechend simpel ist die Ableitung der Funktion zu berechnen, was im Trainingsprozess ein enormer Vorteil ist. Aus den oben genannten Gründen ist die ReLU als Aktivierungsfunktion heute sehr weit verbreitet in Deep Learning Architekturen [15].

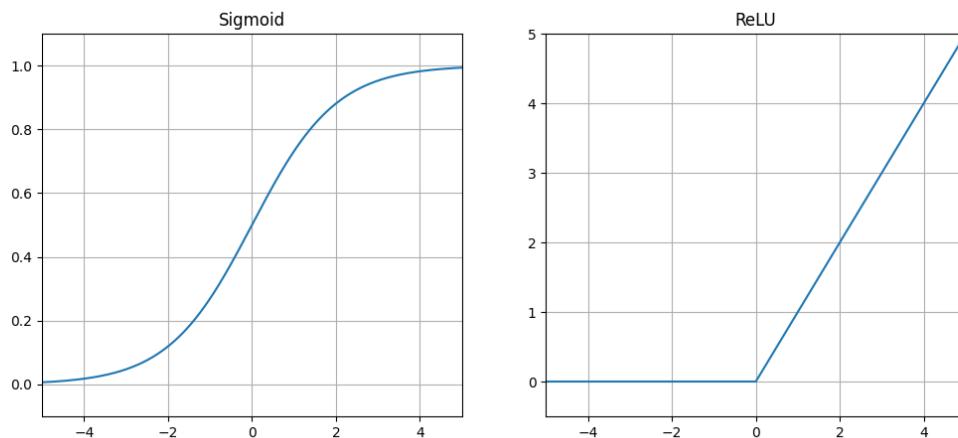


Abbildung 2.2.: Links: Sigmoid Funktion, Rechts: Rectified Linear Unit

### 2.1.3. Mehrlagiges Perzeptron

Durch das parallel, sowie in Reihe schalten von Perzeptronen wird ein Mehrlagiges Perzeptron (MLP), wie bspw. in Abb. 2.3., erzeugt. MLPs sind vorwärtsgerichtete Netze (engl. Feedforward Neural Networks). Das heißt die Ausgabe wird durch ein Durchschalten der Eingaben produziert, einen sog. Forward Pass [10]. Jede Stelle, an der sich die Daten auf dem Weg durch das Netz befinden können, wird als Neuron bezeichnet, in Abb. 2.3 stellen dies die Kreise dar, wird als Neuron bezeichnet. Da in einem MLP jedes Neuron eine Verbindung zu jedem Neuron aus der nächsten Schicht hat, werden die Schichten auch voll verbunden (engl. Fully Connected) genannt. Mathematisch wird es im Vergleich zum Perzeptron minimal komplexer, indem nun die Ausgaben von Neuronen wiederum die Eingaben anderer Neuronen sind. Es kommt demnach in der Berechnung ein weiterer Index dazu. Beispielhaft ist hier die Berechnung der Neuronen aus Abb. 2.3  $y_j$  mit  $j \in \{1, 2, 3\}$ , sowie  $z_k$  mit  $k \in \{1, 2\}$ , dargestellt. Die Gewichte  $w$  sind nun kein

Vektor mehr, sondern eine Matrix. Genau genommen sind es zwei Matrizen für zwei Schichten:  $w^{(1)}$  für die Gewichte der Neuronen  $y_j$ , sowie  $w^{(2)}$  für  $z_k$ .

$$w^{(1)} = \begin{pmatrix} b_1^{(1)} & w_{1,1}^{(1)} & w_{1,2}^{(1)} \\ b_2^{(1)} & w_{2,1}^{(1)} & w_{2,2}^{(1)} \\ b_3^{(1)} & w_{3,1}^{(1)} & w_{3,2}^{(1)} \end{pmatrix}, \quad w^{(2)} = \begin{pmatrix} b_1^{(2)} & w_{1,1}^{(2)} & w_{1,2}^{(2)} & w_{1,3}^{(2)} \\ b_2^{(2)} & w_{2,1}^{(2)} & w_{2,2}^{(2)} & w_{2,3}^{(2)} \end{pmatrix}$$

$f$  ist weiterhin die Aktivierungsfunktion, ebenso wie  $x = (1, x_1, x_2)$  der Eingabevektor.  $w_j^{(i)}$  ist die  $j$ -te Zeile von  $w^{(i)}$ , mit  $i \in 1, 2$ .  $y = (1, y_1, y_2, y_3)$  ist der Ausgabevektor der Neuronen der versteckten Schicht und somit ebenso der Eingabevektor für die Ausgabeschicht.

$$y_j = f(x \cdot w_j^{(1)\top}) \quad (2.5)$$

$$z_k = f(y \cdot w_k^{(2)\top}) \quad (2.6)$$

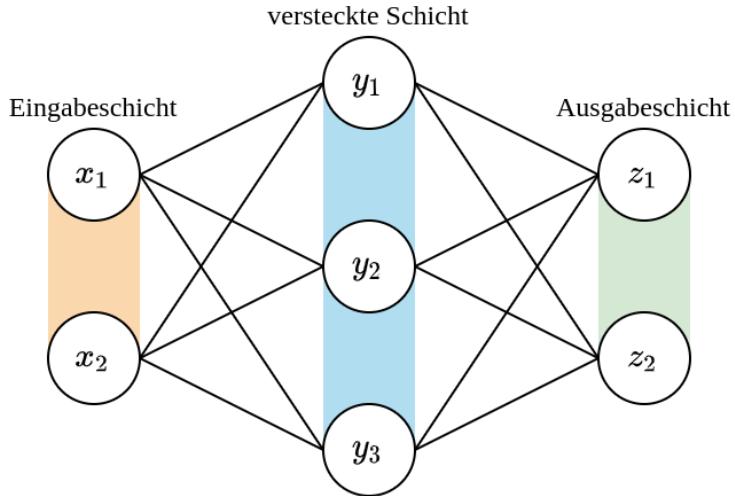


Abbildung 2.3.: Ein Mehrlagiges Perzepton mit einer versteckten Schicht

#### 2.1.4. Verlustfunktion

Man stelle sich vor das MLP aus Abb. 2.3 soll das Klassifizierungsproblem lösen, ob eine WEA von Hersteller A oder B ist. Die Gondel ist bei Anlagen von Hersteller A, im Vergleich zu Gondeln von Hersteller B, deutlich kürzer und hat einen größeren Durchmesser. Die Eingänge für das Netz könnten also Länge und Durchmesser der Gondel sein. Der Ausgang  $z_1$  enthält dann die Wahrscheinlichkeit, dass die Eingänge eine A Gondel beschreiben. Analog enthält  $z_2$  die Wahrscheinlichkeit für eine B Gondel. Die Gewichte des Netzes seien zufällig initialisiert. Wie kann ermittelt werden, wie gut das Netz das

Problem bereits lösen kann? Die Berechnung des Forward Pass mit realen Werten einer Anlage von Hersteller A, liefert vermutlich eine Ausgabe, die fernab der Sollausgabe ist. Diese Entfernung wird durch eine Verlustfunktion (engl. Loss Function) beziffert [16]. Nicht jede Verlustfunktion eignet sich für jedes Problem und ihre Wahl bestimmt entscheidend darüber, ob ein ANN sinnvoll lernen kann oder nicht. Der Trainingsprozess eines ANN ist dann im Grunde das Verändern der Gewichte, sodass der durchschnittliche Verlust  $J$  über alle Trainingsdaten minimal wird. Sei  $\mathcal{L}$  die Verlustfunktion,  $f$  der Forward Pass durch das Netz,  $W$  die Gewichte des Netzes,  $n$  die Anzahl der Trainingsdaten,  $x$  die Eingabe und  $y$  der Sollwert der Ausgabe.  $J$  kann nun wie folgt beschrieben werden.

$$J(W) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(y_i, f(x_i, W)) \quad (2.7)$$

Es folgen Beispiele für Verlustfunktionen. Die **mittlere quadratische Abweichung** (engl. **Mean Squared Error**) findet Anwendung bei Problemen, in denen ein kontinuierlicher Wert prädiziert werden soll (Regressionsprobleme).  $m$  ist in diesem Fall die Dimension des Sollwertes bzw. prädizierten Wertes  $\hat{y}$ .

$$\mathcal{L}_{MSE}(y, \hat{y}) = \frac{1}{m} \sum_{j=1}^m (y_j - \hat{y}_j)^2 \quad (2.8)$$

Die **Kreuzentropie** (engl. **Cross-Entropy Loss**) findet Anwendung bei Problemen, in denen die Zugehörigkeit zu einer Klasse prädiziert werden soll (Klassifizierungsprobleme).  $M$  ist die Anzahl der Klassen.

$$\mathcal{L}_{CE}(y, \hat{y}) = - \sum_{c=1}^M y_c \log \hat{y}_c \quad (2.9)$$

Bei einem binären Klassifizierungsproblem lässt sich der **Binary Cross-Entropy Loss** wie folgt ausdrücken:

$$\mathcal{L}_{BCE}(y, \hat{y}) = -(y \log \hat{y} + (1 - y) \log (1 - \hat{y})) \quad (2.10)$$

### 2.1.5. Backpropagation

Der Backpropagation-Algorithmus ist das Verfahren mit dem heute ein Großteil aller ANNs trainiert wird. Für eine eingehende Erklärung kann [16] konsultiert werden. Zunächst sind die Gewichte des Netzes zufällig initialisiert. Im ersten Schritt passiert ein Forward Pass aller Trainingsdaten durch das Netz. Mit diesem Ergebnis wird der Verlust  $J$  aus Formel 2.7, sowie sein Gradient berechnet. Hierfür muss der Verlust ableitbar sein. Genauer gesagt müssen die partiellen Ableitungen nach den einzelnen Gewichten aus  $W$

definiert sein. Die Berechnung des Gradienten wird mithilfe der Kettenregel ermöglicht. Abschließend werden die Gewichte in Richtung des Gradienten, in Abhängigkeit von einer Lernrate  $\eta$ , aktualisiert.

$$\begin{aligned} W &\leftarrow W - \eta \nabla J(W), \text{ wobei} \\ \nabla J(W) &= \left( \frac{\partial J(W)}{\partial w}, \forall w \in W \right) \end{aligned} \quad (2.11)$$

Die Berechnung der partiellen Ableitungen mittels der Kettenregel wird nun an einem simplen Beispielnetz (s. Abb. 2.4) gezeigt. Zunächst wird die partielle Ableitung nach  $w_2$ , also nur der hintere Teil der Kette, berechnet.

$$\frac{\partial J(W)}{\partial w_2} = \frac{\partial J(W)}{\partial z} * \frac{\partial z}{\partial w_2} \quad (2.12)$$

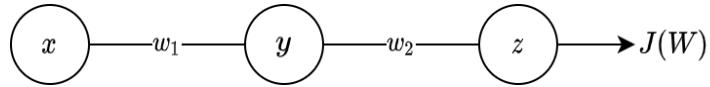


Abbildung 2.4.: Ein simples Netz, um die Berechnung der Kettenregel zu betrachten.

Für die gesamte Kette fehlt nun noch die partielle Ableitung nach  $w_1$ . Hierfür kann die Berechnung von  $\frac{\partial J(W)}{\partial z}$  aus 2.12 wiederverwendet werden, um Rechenleistung einzusparen.

$$\frac{\partial J(W)}{\partial w_1} = \frac{\partial J(W)}{\partial z} * \frac{\partial z}{\partial y} * \frac{\partial y}{\partial w_1} \quad (2.13)$$

Wie verhält sich die Berechnung der partiellen Ableitungen, wenn das Netz mehrere Neuronen pro Schicht hat? Als Beispiel wird das Netz aus Abb. 2.3 herangezogen. In Abb. 2.5 sind in rot die Pfade durch das Netz markiert, welche bei der partiellen Ableitung nach  $w_{11}^1$  eine Rolle spielen. Berechnet wird diese Ableitung indem die Summe der beiden Pfade gebildet wird. Diese werden analog wie in den Gleichungen 2.12 und 2.13 berechnet.

$$\frac{\partial J(W)}{\partial w_{11}^1} = \frac{\partial J(W)}{\partial z_1} * \frac{\partial z_1}{\partial y_1} * \frac{\partial y_1}{\partial w_{11}^1} + \frac{\partial J(W)}{\partial z_2} * \frac{\partial z_2}{\partial y_1} * \frac{\partial y_1}{\partial w_{11}^1} \quad (2.14)$$

Der Algorithmus wird so lange wiederholt bis der Verlust konvergiert, also entweder das globale oder ein lokales Minimum erreicht ist. Nur ein lokales Minimum zu finden, klingt erstmal nicht erstrebenswert. Vermeiden ließe sich dies mit einer konvexen Verlustfunktion, welche die Eigenschaft hat lediglich ein Minimum zu besitzen. In der Praxis verwendet man allerdings selten konvexe Funktionen. In [17] wird argumentiert, dass dies bei großen Netzen kein größeres Problem darstellt und sich trotz lokaler Minima qualitativ gute Konvergenzen ergeben.

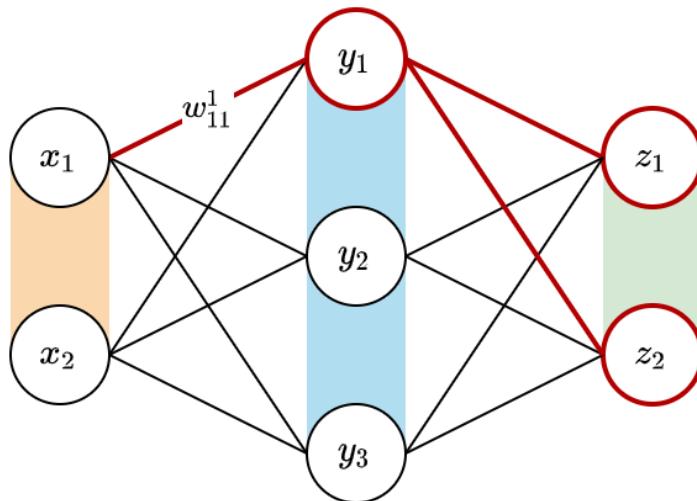


Abbildung 2.5.: In Rot: Neuronen und Gewichte, die bei der Aktualisierung des Gewichts  $w_{11}^1$  entscheidend sind.

Es gibt außerdem noch einige Erweiterungen, bzw. Verbesserungen des Backpropagation-Algorithmus, welche nun aufgegriffen werden.

**Stochastic Gradient Descent:** Wie bereits zuvor erwähnt, werden beim Backpropagation-Algorithmus alle Trainingsdaten, mittels eines Forward Pass durch das Netz geleitet, um den Gradienten zu ermitteln und die Gewichte zu aktualisieren. Dieser Vorgang wird auch Batch Learning genannt. Für das Deep Learning wird eine große Menge an Trainingsdaten benötigt, mit welcher das Batch Learning extrem langsam, sowie specheraufwändig ist. Der Gedanke liegt nahe nur eine Auswahl der Trainingdaten (sog. Mini-Batches) zu wählen, um den wirklichen Gradienten zu approximieren. Dieser Gradient enthält dann ein gewisses Rauschen, welches sich häufig sogar positiv auf das Endergebnis auswirkt. Der größte Vorteil ist aber, dass sich das Training so massiv beschleunigen lässt [18].

**Lernratenoptimierung:** Hat der Verlust Stellen an denen der Gradient sehr klein ist, können diese mit einer konstanten Lernrate auch nur langsam durchquert werden (s. Formel 2.7) [19]. Ein weiteres Problem ist, dass der Algorithmus in einem lokalen Minimum stecken bleiben oder über das globale Minimum hinausschießen kann, je nachdem ob die Lernrate zu klein oder zu groß gewählt ist. Aus diesen Gründen wurden einige Optimierungsverfahren entwickelt, welche die Lernrate dynamisch anpassen. Einen Überblick darüber kann in [20] gefunden werden.

**Vanishing Gradient:** Dies ist ein Problem, welches beim Zurückpropagieren der partiellen Ableitungen durch viele Schichten auftreten kann. Sind die partiellen Ableitungen kleiner als 1, wie es bspw. mit der Sigmoid Aktivierungsfunktion der Fall sein kann (s. Abb. 2.2 und Formel 2.3), nimmt der Gradient bis zu den vorderen Schichten durch die Kettenregel exponentiell ab. Dadurch kann die Gewichtsaktualisierung in den vor-

deren Schichten so klein werden, dass sie nicht mehr wahrnehmbar ist. Dieses Problem kann einerseits mit der Wahl einer anderen Aktivierungsfunktion umgangen werden, andererseits kann eine Netzwerkarchitektur, welche Abkürzungen für den Gradienten implementiert, hilfreich sein [21]. Mehr hierzu folgt in Abschnitt 3.2.2.

**Normalisierung der Eingangswerte:** Der Backpropagation-Algorithmus kann schneller konvergieren, wenn der Durchschnitt der Werte aller Trainingsdaten nahe 0 ist. Um dies zu verstehen betrachten wir die Aktualisierung von Gewichten in der ersten Schicht eines Netzes. Wenn alle Werte des Eingangsvektors positiv sind, haben die Aktualisierungen der Gewichte für ein bestimmtes Neuron alle das gleiche Vorzeichen. Die Gewichte können demnach nur zusammen in eine Richtung aktualisiert werden. Wenn allerdings erreicht werden muss, dass manche Gewichte sich erhöhen und andere sich verringern (bildlich gesprochen muss der Gewichtsvektor seine Richtung ändern) ist dies ineffizient [18]. Aus diesem Grund ist es hilfreich auf die Eingangswerte bspw. die Z-Score Normalisierung anzuwenden. Mehr hierzu folgt in Abschnitt 3.1.5.

**Batch Normalisierung:** Dieses Verfahren dient ebenfalls dazu den Lernprozess des Backpropagation-Algorithmus zu beschleunigen. Im Unterschied zur Normalisierung aller Eingangswerte, werden bei der Batch Normalisierung nur die Werte des Mini-Batches normalisiert. Außerdem kann dies vor einer beliebigen Schicht im Netz passieren. Durch die Verringerung der Abhängigkeit der Gradienten von der Größenordnung der Eingangswerte, wird die Wahl einer höheren Lernrate ermöglicht, ohne in Gefahr zu laufen, dass der Algorithmus divergiert. Dies wirkt sich ebenfalls positiv auf die Trainingsdauer aus [22].

## 2.1.6. Overfitting

Ein generelles Problem, das bei Modellen des Maschinellen Lernens besondere Beachtung erhalten sollte, ist Overfitting (deu. Überanpassung). Von Overfitting wird gesprochen, wenn das Modell sich zu sehr oder exakt an den Daten orientiert mit denen es trainiert wurde, und somit weitere Daten nicht zuverlässig vorhersagen kann [23]. In Abb. 2.6 soll eine Kurve gefunden werden, welche die roten und grünen Trainingsdatenpunkte trennt. In beiden Fällen, also links sowie rechts, wird eine solche Kurve gefunden. Im linken Fall ist sie jedoch selbst an die kleinsten Schwankungen in den Datenpunkten angepasst, so dass man von Overfitting sprechen kann. Im rechten Fall spielen die Schwankungen keine Rolle, sondern lediglich die allgemeine Verteilung der Datenpunkte. Die Kurve generalisiert demnach sinnvoll von den Trainingsdaten auf die Allgemeinheit.

Bei ANNs tritt dieses Problem auf, wenn zu lange trainiert wird. Aus diesem Grund wird neben den Trainingsdaten mit einem weiteren Datensatz, dem Validation Set, gearbeitet. Diese Daten bekommt das ANN während des Trainings nicht zu sehen. Es dient lediglich zur Überprüfung wie gut das ANN auf ungewohnten Daten performt. Kommt es nun zum Overfitting divergieren die Verluste der beiden Datensätze (in Abb. 2.7 ab der markierten Stelle). Um Overfitting zu verhindern wird das sog. Early Stopping angewandt, also der

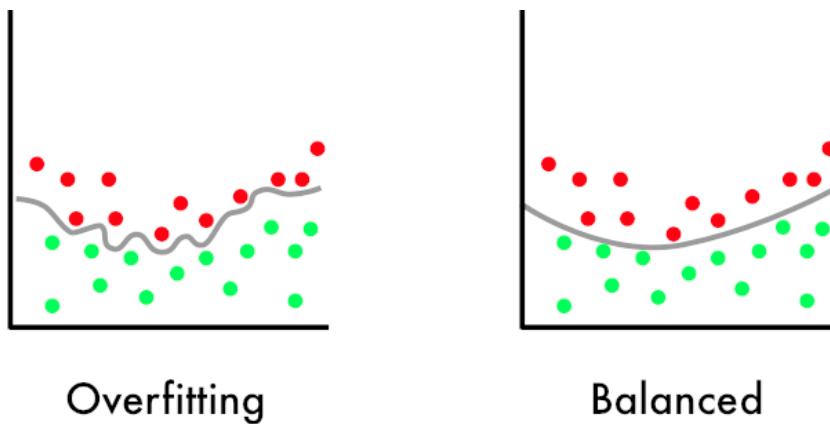


Abbildung 2.6.: Overfitting eines Modells [24]

Abbruch des Trainings. In der Praxis ist es nicht so einfach den richtigen Moment dafür zu finden, da die Kurven in der Realität volatiler sind als in Abb. 2.7. Genaueres hierzu findet sich in [25].

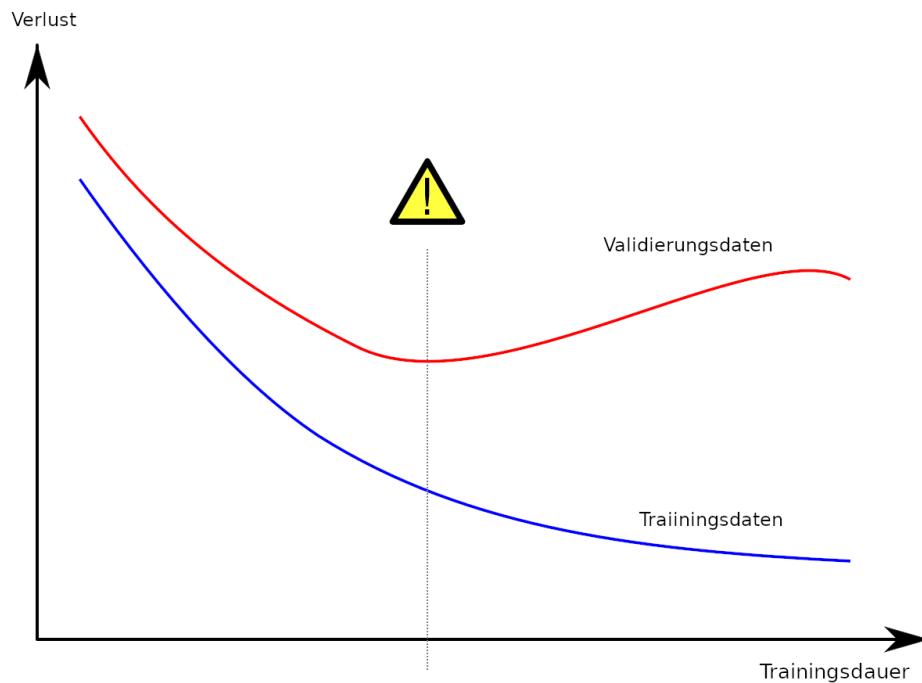


Abbildung 2.7.: Übertraining [26]

## 2.2. Convolutional Neural Networks

CNNs sind wie bereits ANNs aus der Neurobiologie beeinflusst. Hubel und Wiesel fanden 1959 heraus, dass im visuellen Kortex einer Katze einige Neuronen nur aktiviert werden, wenn die Katze vertikale Linien sieht. Andere Neuronen wiederum reagieren lediglich auf horizontale Linien [27]. Weiterhin fanden sie heraus, dass der visuelle Kortex aus Schichten besteht, die aufeinander aufbauen. Wurden in der vorherigen Schicht Linien erkannt gibt es wiederum Neuronen, welche daraus Konturen erkennen, danach Formen, usw.. Außerdem sind es dieselben Neuronen, welche bspw. Linien an unterschiedlichen Stellen des Sichtfeldes erkennen. Aus diesen Erkenntnissen sind einige der Entwurfsentscheidungen von CNNs inspiriert [19].

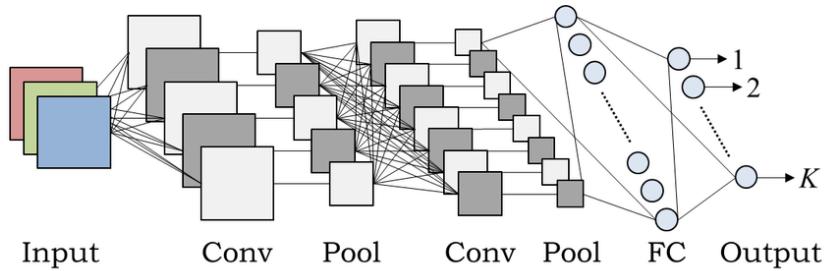


Abbildung 2.8.: Beispiel Architektur eines CNN [28]

Aufgebaut ist ein klassisches CNN, wie es für die Objektklassifizierung von Bildern genutzt wird, grundsätzlich aus einer variablen Anzahl an Blöcken, welche wiederum aus einer variablen Anzahl an Convolutional Layers (CLs), und wahlweise einem Max Pooling Layer (MPL) aufgebaut sind. Nach diesen Blöcken folgt eine variable Anzahl (üblich sind 1 bis 2) Fully Connected Layers, bevor die Ausgabeschicht folgt [19]. Da die CLs, wie gleich noch genauer erläutert wird, immer nur einen Ausschnitt des Bildes betrachten, sind die Fully Connected Layers am Ende des CNN dafür da alle Informationen des Bildes für dessen Klassifizierung heranziehen zu können. In Abb. 2.8 ist diese Architektur beispielhaft zu sehen. Die Eingaben sind die RGB-Kanäle des Bildes. Die Ausgabedimension  $K$  ist in diesem Fall die Anzahl der verschiedenen Klassen, die das Netz prädizieren kann. Die Abb. lässt außerdem erkennen, dass die Blöcke (Conv + Pool) in einem CNN stets 3-dimensional sind. Der Grund dafür wird in den weiteren Abschnitten erläutert.

Warum ist es überhaupt notwendig eine andere Netzarchitektur gegenüber den MLPs zu entwickeln, um Bilder als Eingabe zu verarbeiten? Dafür ist es hilfreich sich bewusst zu machen, was ein Bild als Eingabe für ein MLP bedeutet. Ein RGB-Bild mit dem Format  $200 \times 200$  Pixel liefert bereits  $200 * 200 * 3 = 120.000$  Gewichte pro Neuron in der ersten versteckten Schicht. Für eine komplexe Aufgabe braucht man entsprechend eine große Anzahl an Neuronen über viele Schichten. Die Zahl der Gewichte wird also schnell sehr groß. Davon abgesehen würde ein solches Netz sehr anfällig dafür sein, dass

es beim Training mit Bildern zu Overfitting kommt [19]. Um die Funktionsweise eines CL besser zu verstehen, ist es hilfreich zunächst die Convolution (zu deutsch Faltung) mithilfe von Filtern auf Bildern zu erklären.

## 2.2.1. Convolution

Betrachtet wird das simple Schwarz-Weiß Bild **A** mit den  $2 \times 2$  Filtern **F1** und **F2** aus Abb. 2.9. Eine Convolution wird erreicht, indem ein Filter über das gesamte Bild

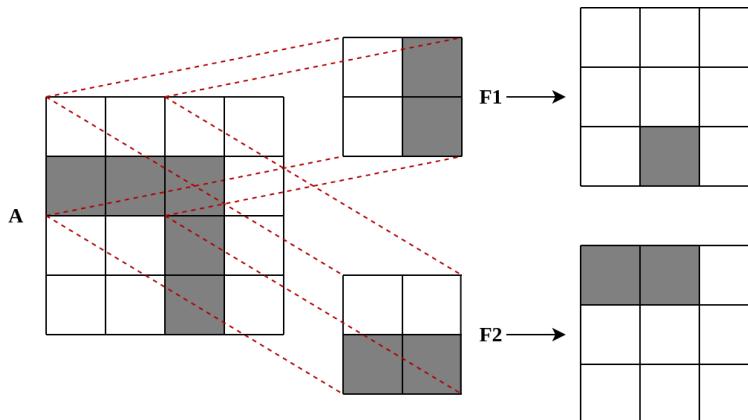


Abbildung 2.9.: Beispielbild A, Filter F1 und F2, sowie die produzierten Feature Maps

geschoben wird und für jede Position eine Ausgabe erzeugt. In Rot ist in Abb. 2.9 die Startposition der Filter links oben im Bild dargestellt. Ein Filter erzeugt ein schwarzes Pixel, wenn der Bildausschnitt mit dem Filter übereinstimmt, ansonsten ein weißes Pixel. Das erzeugte Bild jedes Filters wird als Feature Map bezeichnet. Die Schrittweite (engl. Stride), mit welcher der Filter über das Bild gleitet, ist im Beispiel aus Abb. 2.9 1. Verwendet man eine höhere Schrittweite schrumpft die erzeugte Feature Map und man berücksichtigt weniger Informationen aus dem Ursprungsbild. Weiterhin kann man Filter mit einer anderen Größe verwenden. Würde im Beispiel ein  $3 \times 3$  Filter verwendet werden, wären die erzeugten Feature Maps, bei einer Schrittweite von 1, nur noch  $2 \times 2$ . Um das Schrumpfen der Feature Maps durch die Convolution zu verhindern, kann man das Ursprungsbild mit einem äußerem Rahmen von Hilfspixeln (engl. Padding) ausstatten. Im Fall von  $2 \times 2$  Filtern reicht demnach ein Rahmen von einem Pixel aus, um bei einer Schrittweite von 1, Feature Maps zu produzieren, welche die gleichen Dimensionen wie das Ursprungsbild haben [19].

Eine Convolution auf einem RGB-Bild funktioniert so, dass die Filter, ebenso wie das Bild eine dritte Dimension, die Farbkanäle besitzen. Die Kanäle in den Bildern enthalten für jedes Pixel Werte, ebenso wie die Filter. Um einen Wert Feature Map zu produzieren werden die Produkte aus den Filterwerten und den korrespondierenden Bildwerten aufsummiert.

## 2.2.2. Convolutional Layer

Ein Convolutional Layer erhält ein 3-dimensionales Volumen an Informationen als Eingabe. Dies kann entweder ein Bild oder die Ausgabe einer vorherigen Schicht sein. Aus diesem 3-dimensionalen Volumen produziert das Convolutional Layer durch eine Convolution wiederum ein 3-dimensionales Volumen. Die Tiefe des Ausgabevolumens wird dabei durch die Anzahl an Filtern bestimmt. Die Breite und Höhe ergeben sich, wie bereits erwähnt durch eine Kombination aus der Größe der Filter, der Schrittweite sowie des Paddings.

Wie man in Abb. 2.9 erkennen kann, fungieren Filter F1 und F2 jeweils als vertikaler bzw. horizontaler Kantendetektor von Weiß zu Schwarz. Die Filter in einem CL sind ebenfalls dafür zuständig bestimmte Merkmale (engl. Features) zu erkennen und im Fall einer solchen Erkennung dies mit einer Aktivierung, also einer hohen Summe, zu signalisieren. Dadurch dass die Filter bei der Convolution über das gesamte Bild geschoben werden, sind sie in der Lage Merkmale auf dem gesamten Bild zu erkennen, unabhängig davon wo sie sich befinden. Man hat also in dieser Hinsicht eine deutliche Ähnlichkeit zum visuellen Kortex der Katze hergestellt. Außerdem sind so die Gewichte des Filters über das gesamte Bild geteilt, wodurch sich Gewichte einsparen lassen.

## 2.2.3. Max Pooling Layer

Ein Max Pooling Layer wendet eine Max Pooling Operation auf die Feature Maps an. Bei einer Pooling Operation wird wie bei der Convolution ein Filter bzw. beim Pooling ein Fenster, über die Feature Maps geschoben. In jeder Position wählt das Fenster das Maximum aus, welches dann in eine komprimierte Feature Map übernommen wird. Dadurch soll die Größe der Feature Map verringert werden. Der Gedanke ist, dass wichtige Informationen erhalten bleiben, auch wenn ihre Position im Bild ungenauer wird. Durch die Verkleinerung der Feature Maps wird die Anzahl der Gewichte für folgende Schichten verringert [19]. In Abb. 2.10 ist eine Max Pooling Operation dargestellt. Wie bei der Convolution bestimmt die Größe des Pooling Fensters, sowie die Schrittweite die Größe der Ausgabe Feature Map. In Abb. 2.10 ist die Größe 2 x 2 und die Schrittweite 2.

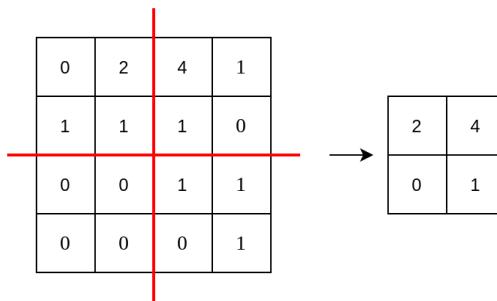


Abbildung 2.10.: Max Pooling Operation mit einem 2 x 2 Fenster, sowie Schrittweite 2.

## 2.3. Objektdetektierung mit CNNs

Wie bereits in der Einleitung erwähnt soll die Lokalisierung von WEA über die Objektdetektierung, mittels CNNs und der Methode des Deep Learning realisiert werden. Bei der Objektklassifizierung wird von den Trainingsdaten lediglich verlangt, dass sie zu jedem Bild das Label des dominierenden Objektes enthalten. In Kontrast dazu wird bei der Objektdetektierung zu jedem Objekt im Bild, welches später vom Netz erkannt werden soll, ein Label, sowie deren Ausmaße benötigt. Diese Ausmaße sind meist in der Form einer rechteckigen Box angegeben, welche im Folgenden Ground Truth Box (GTB) genannt wird.

**Non-Maximum Suppression (NMS):** Ein Problem der Objektdetektierung ist es zu verhindern, dass ein Objekt von mehreren Prädiktionen erkannt wird. Dafür wird seit fast 50 Jahren der Algorithmus der NMS angewandt [29]. Der Pseudocode des Algorithmus ist weiter unten zu sehen. In einer Schleife wird zunächst die Prädiktion mit dem höchsten Detection Score herausgesucht. Dann werden alle anderen Prädiktionen verworfen, die sich mit dieser Prädiktion, mindestens mit dem Schwellwert  $\lambda$ , überlappen. Die Überlappung wird dabei mit der Intersection over Union (IoU) berechnet. Dabei wird die Schnittfläche der Prädiktionen durch die Fläche der Vereinigung beider Prädiktionen geteilt. Die Schleife wird so lange wiederholt, bis alle Prädiktionen verworfen bzw. der neuen Menge an Prädiktionen hinzugefügt sind. Je nachdem wie der Schwellwert gewählt ist, bleiben als Resultat die Prädiktionen mit den höchsten Detection Scores übrig.

---

### Algorithm 1 NMS

---

$B = \{b_1, \dots, b_N\}$ : Menge der prädizierten Boxen  
 $S = \{s_1, \dots, s_N\}$ : Korrespondierende Detection Scores  
 $\lambda$ : IoU Schwellwert

```

1: function NMS( $B, S, \lambda$ )
2:    $B_{res} \leftarrow \emptyset, S_{res} \leftarrow \emptyset$ 
3:   while  $B \neq \emptyset$  do
4:      $m \leftarrow argmax S$ 
5:      $B_{res} \leftarrow B_{res} \cup b_m, S_{res} \leftarrow S_{res} \cup s_m$ 
6:      $B \leftarrow B - b_m, S \leftarrow S - s_m$ 
7:     for  $i = 1$  to  $|B|$  do
8:       if  $iou(b_m, b_i) \geq \lambda$  then
9:          $B \leftarrow B - b_i, S \leftarrow S - s_i$ 
10:    return  $B_{res}, S_{res}$ 
  
```

---

Für die Objektdetektierung wurden in den letzten Jahren diverse Modelle entwickelt, von denen einige kurz vorgestellt werden.

**Region-Based Convolutional Neural Networks (R-CNNs):** Diese Modellfamilie hat die Eigenschaft, dass zunächst Regionen in einem Bild identifiziert werden, welche Objekte enthalten können. In einem zweiten Schritt werden die vorgeschlagenen Regionen klassifiziert. Das erste Modell dieser Art war das R-CNN [30], welches Objektvorschläge mithilfe des Selective Search Algorithmus findet. Wie der Algorithmus funktioniert kann in [31] nachlesen werden. Die Objektvorschläge werden dann einzeln mittels eines CNN klassifiziert. Eine Weiterentwicklung dessen stellt das Fast R-CNN [32] dar. Es ist dadurch schneller, dass die Vorschläge nicht einzeln durch das CNN geleitet werden müssen. Stattdessen werden lediglich einmal die Feature Maps für das gesamte Bild generiert. Die Objektvorschläge auf den Feature Maps werden mithilfe eines Region of Interest (RoI) Pooling auf eine einheitliche Größe gebracht, um klassifiziert werden zu können. Wie genau das RoI Pooling funktioniert wird in Abschnitt 2.4.2 erläutert. Eine letzte Erweiterung in Form des Faster R-CNN verzichtet auf den Selective Search Algorithmus und nutzt stattdessen ein zweites Netz um Objektvorschläge zu prädizieren. Da dieses Modell für diese Arbeit verwendet wird, folgt dazu ein eigener Abschnitt 2.4 mit einer ausführlichen Erklärung der Funktionsweise.

**Single Shot Detectors:** Bei dieser Modellfamilie erfolgt die Objekterkennung nicht, wie bei R-CNNs, in zwei Schritten, sondern, wie der Name bereits vermuten lässt, in einem Schritt. Es ist somit ein einzelnes Netzwerk für die Detektierung, sowie Klassifizierung von Objekten zuständig. Das Ziel ist die Prädiktionsgeschwindigkeit so zu erhöhen, dass sie in Echtzeit ablaufen kann. Das ist bspw. für autonomes Fahren essentiell. Bei dem Modell *You Only Look Once (YOLO)* wird das Bild in ein Raster unterteilt (im Paper [33] ein 7 x 7 Raster). Die Rasterzelle, in der sich das Zentrum eines Objektes befindet, ist dafür zuständig dieses Objekt zu detektieren. Dafür steht jeder Rasterzelle nicht nur ihr eigener Inhalt als Information zu Verfügung, sondern es werden die Informationen des gesamten Bildes in Betracht gezogen. Wie dies im Detail funktioniert kann in [33] nachgelesen werden.

Ein weiteres Modell ist der Single Shot MultBox Detector (SSD). Im Vergleich zu YOLO mit seinem Raster, sind bei diesem Modell mehrere CLs für die Objektdetektierung zuständig. Diese produzieren stufenweise kleiner werdende Feature Maps, welche für jeweils verschiedenen großen Objekten im Bild zuständig sind. Mit dieser Technik werden höhere Genauigkeiten als mit YOLO erzielt, während die Schnelligkeit der Prädiktionen ebenfalls nochmals höher ist. Im Detail kann dies in [34] nachgelesen werden.

Von YOLO existieren mittlerweile mehrere Versionen, in denen das Modell stetig verbessert wurde. Die neuesten Versionen YOLOv3 und YOLOv4 liefern bessere Resultate als SSD [35] [36]. Da die Schnelligkeit für die Problemstellung kein entscheidender Faktor ist, und das Faster R-CNN i.A. eine bessere Genauigkeit als andere Modelle aufweist [37], wird es für diese Arbeit verwendet.

## 2.4. Faster R-CNN

Der Aufbau des Faster R-CNN (s. Abb. 2.11) beginnt mit den *conv layers*, welche eine Kombination aus Convolutional und Max Pooling Layers sind. Eine solche Kombination wird auch auch Fully Convolutional Network genannt, da im Vergleich zu CNNs die Fully Connected Layers am Ende fehlen. Die produzierten Feature Maps dienen dann einem Region Proposal Network (RPN) als Eingabe, welches Regionen (RoI) auf den Feature Maps prädiziert, an denen Objekte im Bild sein könnten. Diese RoI sind rechteckig und können daher mit einer Breite und Höhe, sowie einer Mittelpunktkoordinate eindeutig beschrieben werden. Mittels einem RoI Pooling Layer werden die RoI auf den Feature Maps auf eine einheitliche Größe gebracht. Das ist notwendig, da die Fully Connected Layers des *classifier* eine feste Größe als Eingabe voraussetzen. Der *classifier* ordnet jeder RoI eine Klasse zu [38].

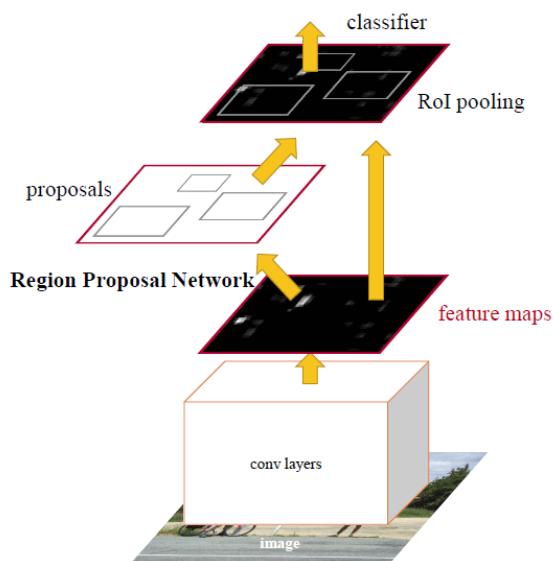


Abbildung 2.11.: Architektur des Faster R-CNN Modells [38]

### 2.4.1. Region Proposal Network

Das RPN ist aus drei CLs aufgebaut. Das *intermediate layer* hat ein *sliding window* (entspricht einer Filtergröße von  $3 \times 3$ ) und, im Fall von Abb. 2.12, 256 Filter. Die Anzahl der Filter ist dabei abhängig vom zu Grunde gelegten Fully Convolutional Network. Hat dessen letztes CL eine Ausgabedimension von 1024, wie es in dieser Arbeit der Fall ist, hat das *intermediate layer* 512 Filter. In jeder Position stellt der Mittelpunkt des *sliding window* einen Anchor dar. Um diesen Anchor werden  $k$  Anchor Boxen gelegt, welche die Vorschläge für Regionen, in denen sich ein Objekt befinden kann, darstellen. Die Zahl  $k$  setzt sich dabei aus den unterschiedlichen Größen und Seitenverhältnissen, welche Objekte im Bild haben können, zusammen. Für eine Standard Objektdetektion wird mit kleinen, mittleren und großen, sowie quadratischen, hochformatigen und querformatigen Anchor Boxen gearbeitet, wodurch sich mit 3 Größen und 3 Seitenverhältnissen  $k = 3 \times 3 = 9$  ergibt. Für jede Anchor Box wird dann im *cls layer*, einer der beiden Ausgabeschichten, ein Objektscore ermittelt. Dieser enthält die Wahrscheinlichkeit, dass sich in der vorgeschlagenen Region ein Objekt oder nur Bildhintergrund befindet. Die zweite Ausgabeschicht, das *reg layer*, liefert regressierte Ausmaße der Anchor Box an das

tatsächliche Objekt im Bild [38].

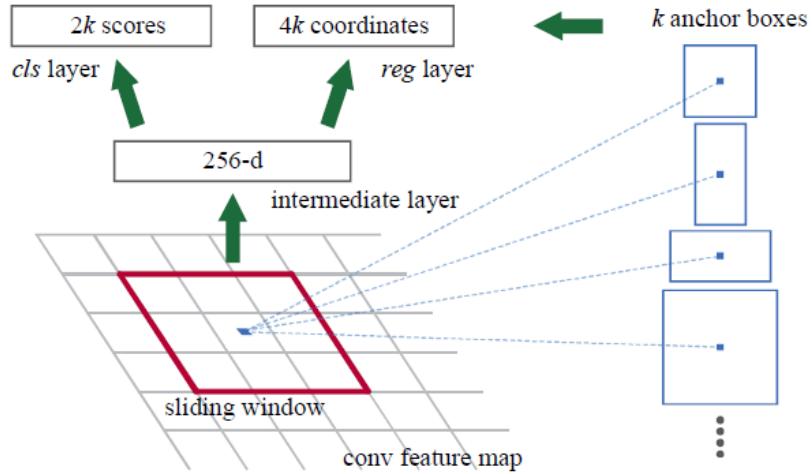


Abbildung 2.12.: Architektur des Region Proposal Network [38]

### Training des RPN

Um zu entscheiden, ob eine Anchor Box ein Objekt enthält, wird die IoU mit den GTBs berechnet. Hat eine Anchor Box eine IoU mit einer GTB von größer als 0,7 wird angenommen, dass diese Anchor Box ein Objekt enthält und sie bekommt ein positives Label. Bei einer IoU von kleiner als 0,3 bekommt sie ein negatives Label. Jegliche Boxen mit einer IoU zwischen 0,3 und 0,7 sind neutral und werden im Trainingsprozess nicht berücksichtigt.

Die Verlustfunktion des RPN setzt sich aus zwei Termen für die beiden Ausgabeschichten (*cls* und *reg layer*) zusammen und lautet wie folgt:

$$\begin{aligned} \mathcal{L}(\{p_i\}, \{t_i\}) = & \frac{1}{N_{cls}} \sum_i \mathcal{L}_{cls}(p_i, p_i^*) \\ & + \lambda \frac{1}{N_{reg}} \sum_i p_i^* \mathcal{L}_{reg}(t_i, t_i^*) \end{aligned} \quad (2.15)$$

$i$  ist der Index der Anchor Box im Mini-Batch, wobei diese Mini-Batches zu 50% positive und zu 50% negative Anchor Boxen enthalten.  $p_i$  ist der Objektscore,  $t_i$  die prädictierten Ausmaße für die Anchor Box.  $p_i^*$  ist das tatsächliche Label der Anchor Box. Der Term des *reg layer*, ist daher für negative Anchor Boxen stets 0, da für diese  $p_i^* = 0$  gilt.  $t_i^*$  sind die die Ausmaße der GTB, welche für das positive Label der Anchor Box gesorgt hat. Normalisiert werden die Terme von  $N_{cls}$ , der Anzahl der Anchor Boxen im Mini-Batch, sowie von  $N_{reg}$ , der Anzahl von Anchor Positionen des *sliding window* (auf 50 x 50 Feature Maps wären dies 2500 Anchors). Der Parameter  $\lambda$  ist ein Gewichtungsfaktor.

In [38] wird gezeigt, dass die beiden Terme ungefähr gleich gewichtet sind, wenn  $\lambda = 10$  ist.  $\mathcal{L}_{cls}$  ist der Binary Cross-Entropy Loss aus Formel 2.10. Der Verlust für den Regressionsterm  $\mathcal{L}_{reg}$  wird wie folgt berechnet:

$$\begin{aligned}\mathcal{L}_{reg}(t, t^*) &= \sum_{j \in \{x, y, w, h\}} smooth_{L_1}(t_j - t_j^*), \text{ wobei} \\ smooth_{L_1}(x) &= \begin{cases} 0.5x^2 & \text{wenn } |x| < 1 \\ |x| - 0.5 & \text{sonst} \end{cases}\end{aligned}\quad (2.16)$$

Die Menge  $\{x, y, w, h\}$  sind die vier Werte, mit denen die Anchor Box beschrieben ist. Durch die Fallunterscheidung des  $smooth_{L_1}$  werden größere Unterschiede zwischen den Werten anders behandelt und der Verlust reagiert weniger anfällig auf Ausreißer [32].

## 2.4.2. RoI Pooling

Wie bereits in Abschnitt 2.4 erwähnt, sorgt das RoI Pooling dafür, dass die verschieden dimensionierten RoI auf eine einheitliche Breite und Höhe (bspw. 7 x 7) gebracht werden. Prinzipiell funktioniert es dabei sehr ähnlich zu einem Max Pooling. Die RoI wird in möglichst gleich große Teile unterteilt, so dass es exakt so viele Teile gibt wie die Zielgröße des RoI Pooling ist. Dann wird aus jedem Teil der größte Wert übernommen. Diese Operation wird für jeden Kanal der Feature Maps einzeln angewandt [32]. In Abb. 2.13 ist ein Beispiel für einen Kanal mit einer RoI Pooling Zielgröße von 2 x 2.

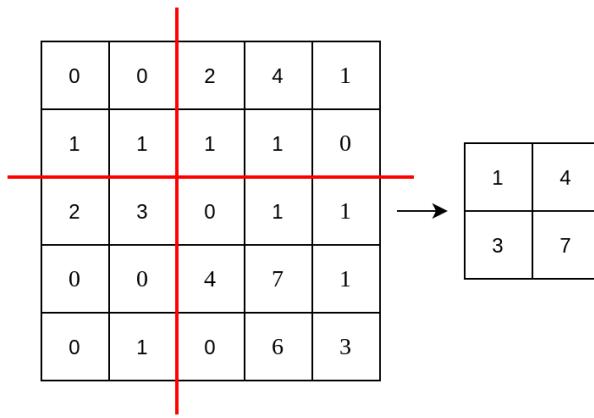


Abbildung 2.13.: RoI Pooling Beispiel mit Zielgröße 2 x 2



# 3. Lösungsansatz

In diesem Kapitel wird zunächst beschrieben welche Vorbereitungen im vornherein nötig sind, um sich dem Problem anzunehmen. Dazu zählt die Vorgehensweise, sowie die sich dabei ergebenden Probleme, bei der Datenerhebung von Windparks aus Deutschland und den USA. Außerdem wird auf die Augmentation und die Normalisierung der Daten eingegangen. Im Anschluss wird ausgeführt welche Bibliotheken, Frameworks und Netzarchitekturen konkret verwendet werden, sowie wie das Faster R-CNN Modell modifiziert wird, um das Problem dieser Arbeit zu adressieren.

## 3.1. Daten

### 3.1.1. Datenerhebung (DEU)

Damit überhaupt Satellitenbilder von WEA gesammelt werden können, wird ein brauchbarer Datensatz mit Standorten benötigt. Zur Bearbeitung dieser Arbeit liegt ein solcher Datensatz für Anlagen aus Deutschland, sowie den USA vor. Im Folgenden wird zunächst beschrieben wie die Daten für deutsche WEA (ca. 26.000) zusammengetragen werden. Der Plan ist es sich den Datenbestand für die Annotierung von Bounding Boxen zunutze zu machen. In diesem ist neben den Standorten auch die Höhe, sowie der Rotordurchmesser der Anlagen enthalten. Die Trainingsdaten werden generiert, indem in Abhängigkeit dieser Parameter verschiedene große Boxen an den Standorten von Anlagen in jedem Bild hinterlegt werden (s. Abb. 3.2). Satellitenbilder können frei verfügbar auf einer Webseite der European Space Agency (ESA) [39] abgerufen werden. Am ehesten für das Problem geeignet sind die Bilder des Satelliten Sentinel-2. Diese weisen allerdings maximal eine Auflösung von 10 Metern pro Pixel auf. Im Vergleich zu Bildern von der Google Static Maps API, mit einer Auflösung von ca. 70 Zentimeter pro Pixel im Beispiel von Abb. 3.1 rechts, ist in den Sentinel-2 Bildern deutlich weniger zu erkennen. Ob es sich bei den Bildern von Google um Satelliten- oder Luftbilder handelt kann nicht sicher gesagt werden. In den Entwicklerhinweisen für die Google Maps Static API steht, dass es sich um Satellitenbilder handelt [40]. Auf Wikipedia wird behauptet, dass es sich bei hochauflösenden Aufnahmen von Städten um Luftbilder und bei den meisten anderen Aufnahmen um Satellitenbilder handelt [41]. Auf jedem Bild von der Google Maps Static API befinden sich unten rechts Informationen über die Quelle der Bilddaten. Für Bilder aus Deutschland stehen hier oft gemeinsam die Quellen *GeoBasis-DE/BKG*, also das Bundesamt für Kartographie (BKG) und Geodäsie, *GeoContent*, sowie *Maxar*



Abbildung 3.1.: Bildvergleich: Sentinel-2 und Google Maps Static API

*Technologies.* Das BKG bietet digitale Orthophotos (verzerrungsfreie, georeferenzierte Luftbilder) mit einer Auflösung von bis zu 20 Zentimeter pro Pixel an [42]. *GeoContent* ist ein deutsches Unternehmen, welches sowohl Luft- als auch Satellitenbilder bereitstellt [43]. *Maxar Technologies* ist ein US-amerikanisches Unternehmen, welches Satellitenbilder bereitstellt [44]. Es lässt sich demnach vermuten, dass es sich bei den Aufnahmen um einen Mix aus Satelliten- und Luftbildern handelt. Zu welcher Kategorie ein bestimmtes Bild gehört kann nicht mit definitiver Sicherheit gesagt werden.

Selbst für einen Menschen ist es schwierig die Anlagen auf dem linken Bild in Abb. 3.1 auszumachen. Weiterhin stellen Wolken ein Problem auf den Sentinel-2 Bildern dar. Zwar ist in den Metainformationen enthalten, zu wie viel Prozent das Bild von Wolken verdeckt ist, selten gibt es jedoch Bilder die komplett frei von ihnen sind. Da die Bilder einen sehr großen Bereich abdecken (ca. 110 x 110 km) ist es problematisch nach hohem Wolkenanteil auszusortieren. Ein Bild mag einen hohen Wolkenanteil haben, jedoch können alle Windparks wolkenfrei sein. Anders herum kann ein Bild einen geringen Anteil von Wolken haben, aber viele Windparks sind von ihnen bedeckt. Aus diesen Gründen wird im weiteren Verlauf mit Bildern der Google Maps Static API gearbeitet. Damit geht natürlich einher, dass die Anlagen auf den Bildern immer eine verschiedene Neigung haben, je nachdem aus welcher Position sie aufgenommen wurden. Dieses Problem soll einerseits durch die Fülle an verschiedenen Bildern und andererseits mit sinnvollen Augmentationen der Bilder adressiert werden. Ein weiteres Problem ist das Fehlen von Metainformationen, nämlich dem Datum der Aufnahme. Das führt dazu, dass ältere Bilder Fehler aufweisen können. In Abb. 3.2 sieht man auf der linken Seite zwar geplante Flächen, nicht jedoch die Anlagen, welche dort mittlerweile stehen. Die Aufnahme ist dafür zu alt. Im nächsten Abschnitt wird versucht diese Fehler zu beifern, um eine Überblick über die Qualität der Daten zu erhalten.

Gesammelt werden 1280 x 1280 Pixel große Bilder, um die Mittelpunkte aller WEA aus



Abbildung 3.2.: Fehlerhafte Annotationen auf älterer Aufnahme

Deutschland, mit der Zoom Stufe 16 (20 ist die stärkste Stufe, s. [40] für mehr Informationen). Die erhaltenen Bilder decken eine Fläche von ca. 920 x 920 Metern ab. Ein Windpark taucht mehrmals in den Bildern auf, da die Standorte aller WEA als Mittelpunkte für die Bilder gewählt werden. Es ist jedoch wichtig eine klare Trennung von Trainings-, Validation- und Testset sicherzustellen. Deshalb wird nach der Zuordnung eines Bildes zu einem Set, mittels einer Umkreissuche jedes weitere Bild des Windparks ermittelt und ebenfalls diesem Set zugeordnet. So ist sichergestellt, dass eine Anlage nicht in zwei Sets auftauchen kann. Es ergeben sich folgende Set Größen:

Training: 17820

Validation: 2970

Test: 2970

### 3.1.2. Evaluation der Datenqualität (DEU)

Die automatische Annotation von Bounding Boxen um die Anlagenstandorte liefert unweigerlich Fehler. Einerseits passiert dies durch zu alte Aufnahmen, andererseits können auch Fehler im WEA-Datenbestand nicht ausgeschlossen werden. Dadurch wird es in den Bildern Boxen geben, in denen keine WEA steht, sowie WEA, die nicht mit einer Box annotiert sind. Um abzuschätzen wie sehr sich diese Fehler in den insgesamt 23.754 Bildern häufen, wird ein Sample der Größe 2000 betrachtet. In diesem gibt es insgesamt 104 Boxen ohne Anlage und 209 Anlagen ohne Box. Dabei verteilen sich

die Fehler insgesamt auf 141 Bilder (7%). Die grüne Datenreihe in Abb. 3.3 stellt die

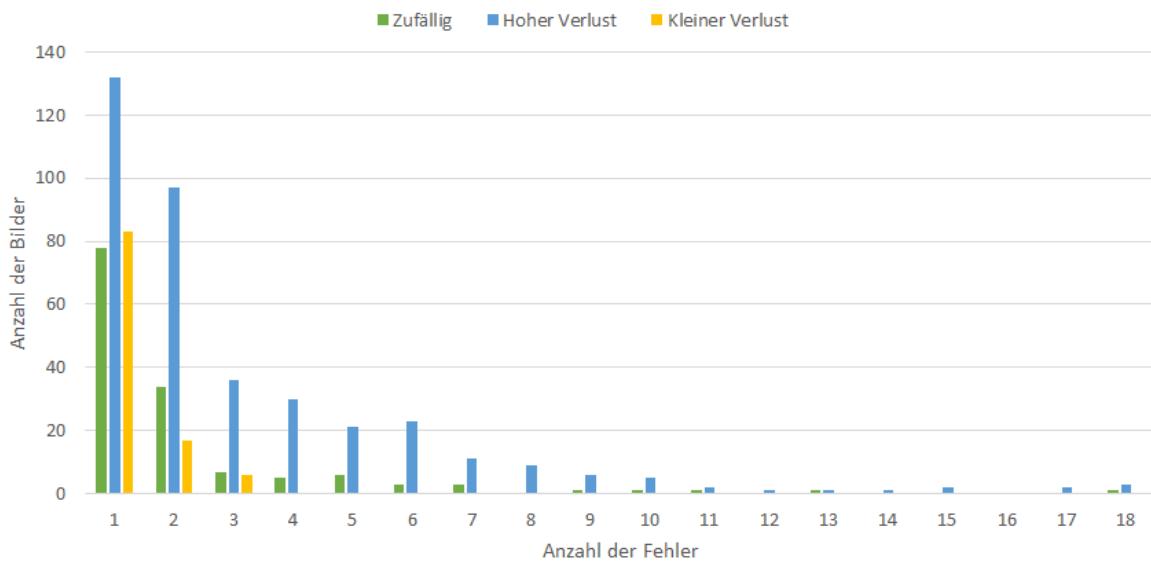


Abbildung 3.3.: Verteilung der Fehler pro Bild in den drei Samples

exakte Verteilung der Fehler pro Bild dieses Samples dar. Häufig tritt nur ein Fehler pro Bild auf, es kann aber auch durchaus zu einer Vielzahl von Fehlern kommen. Im Mittel gibt es pro Bild 4,2 Boxen, das heißt von den ca. 8400 Boxen in den 2000 betrachteten Bildern sind ca. 1,2% fehlerhaft. Die Hypothese ist, dass Bilder mit vielen Fehlern, mithilfe eines bereits trainierten Netzes, in diesem Fall dem RPN alleine, herausgefiltert werden können. Dadurch bestünde die Möglichkeit den Bestand simpler per Hand korrigieren zu können. Zunächst wird für jedes Bild der Verlust errechnet. Das resultierende Histogramm ist in Abb. 3.4 dargestellt. Man kann sehen, dass ein Großteil der Bilder einen Verlust kleiner als 1 hat. Um die Hypothese zu belegen, werden 2000 Bilder mit einem Verlust größer als 3,8, sowie 2000 Bilder mit einem Verlust kleiner als 1 betrachtet. Im hohen Verlust Sample sind insgesamt 590 Boxen ohne Anlage und 636 Anlagen ohne Box. Diese verteilen sich auf 382 Bilder. Die exakte Verteilung der Fehler ist in der blauen Datenreihe in Abb. 3.3 zu sehen. Im Vergleich zum zufälligen Sample sind also in der Tat mehr Bilder mit vielen Fehlern vertreten. Im Sample mit kleinem Verlust befinden sich 3 Boxen ohne Anlage und 133 Anlagen ohne Box. Diese verteilen sich auf 106 Bilder, wobei nur 6 Bilder mehr als zwei Fehler haben. Wie in der gelben Datenreihe in Abb. 3.3 zu sehen ist, hat ein Großteil der Bilder mit Fehlern lediglich einen Fehler. Die Hypothese kann demnach bestätigt werden, wobei auffällig ist, dass sich immer noch relativ viele Anlagen ohne Box in den Bildern mit geringem Verlust befinden. Diese Fehler nehmen offensichtlich weniger Einfluss auf den Verlust eines Bildes. Das könnte daran liegen, dass der Regressionsteil (s. Formel 2.15) lediglich für Anchor Boxen mit positivem Label ungleich 0 ist.

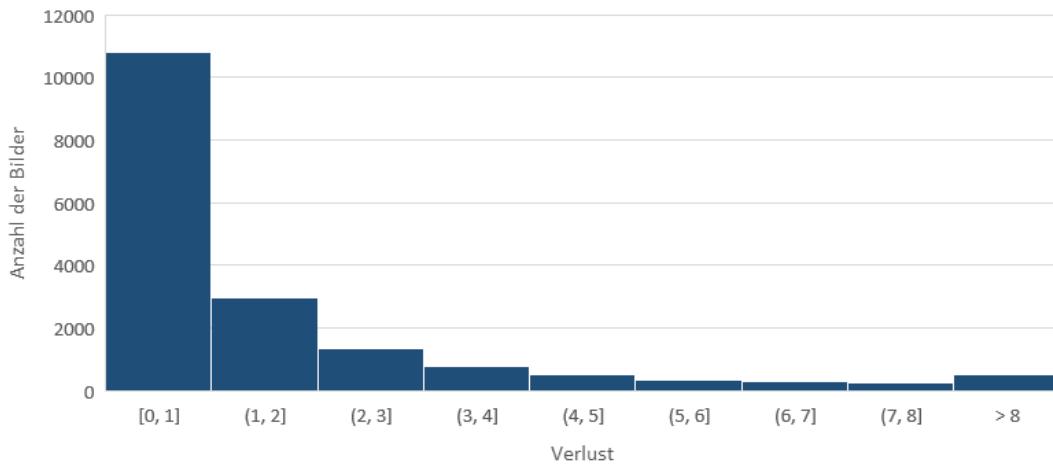


Abbildung 3.4.: Histogramm der Verluste über alle Bilder eines trainierten RPN

### 3.1.3. Datenerhebung (USA)

Die Bilddaten für die Anlagen aus den USA werden gleich erhoben, wie es schon für die Anlagen aus Detuschland in den vorherigen Abschnitten erläutert wurde. Dabei kann auf einen Datenbestand von ca. 65.000 Anlagen zurückgegriffen werden. Allerdings weisen Windparks in den USA Unterschiede zu typischen Windparks aus Deutschland auf, die Probleme mit sich bringen. In Abb. 3.5 ist ein Beispiel eines Windparks in den USA



Abbildung 3.5.: Beispiel Windpark USA; links unten: Anlagen von 2013; rechts: Anlagen von 1982

zu sehen. Die Anlagen rechts im Bild sind von 1982 und im Vergleich zu den Anla-

gen links unten, welche 2013 gebaut wurden, sehr klein. Außerdem sind sie, vermutlich aufgrund der für die Windausbeute günstigen Position in der Hügelstruktur, sehr eng nebeneinander installiert. Das ist insofern ein Problem, dass die Methode jede Anlage als Mittelpunkt eines Bildes zu nehmen, zu einer überproportional häufigen Vertretung eng beieinander stehender WEA in den Bildern führt. Aus diesen Gründen wird bei der Zusammenstellung der Daten für die Anlagen aus den USA darauf verzichtet Bilder von Anlagen zu beziehen, welche vor 2000 installiert wurden. Sie kommen dadurch trotzdem in den Bildern vor, indem neuere Anlagen in der Nähe stehen, sind jedoch nicht überrepräsentiert.

Von den 57.599 potentiellen Bildern von Anlagen (nach Jahr 2000) werden 12.000 zufällig ausgewählt, über die Google Maps Static API bezogen und automatisch annotiert. Nach einer händischen Prüfung, blieben von den 12.000 noch 6.859 Bildern übrig. Aussortiert wurden Bilder die Fehler aufweisen, hauptsächlich weil die Aufnahme zu alt ist. Die Aufteilung in Trainings-, Validation- und Testset folgt dem gleichen Schema wie zuvor bei den Anlagen aus Deutschland. Es ergeben sich folgende Set Größen:

Training: 5144

Validation: 858

Test: 857

### 3.1.4. Augmentation

Es werden einige Augmentationen an den Bildern vorgenommen um einem Overfitting des Modells gegenzusteuern. Die Bilder aus der Google Static Maps API enthalten ein Google Label am unteren Rand des Bildes. Um weiterhin mit quadratischen Bildern zu arbeiten, wird das Label am unteren Rand, sowie der rechte Rand um 50 Pixel abgeschnitten. Dann werden nochmals insgesamt 200 Pixel in der horizontalen, sowie in der vertikalen Richtung vom Bild abgeschnitten. Per Zufall wird entschieden wie viele von diesen 200 Pixeln links, bzw. oben abgeschnitten werden, der Rest entfällt auf die Ränder rechts bzw. unten. Dadurch wird erreicht, dass beim Training nicht jedes Bild exakt im Mittelpunkt eine WEA hat. Nach dieser Operation sind die Bilder noch 1030 x 1030 Pixel groß. Um die Bilder praktikabler für das Training zu machen, ohne dabei den Inhalt weiter abzuschneiden, werden sie auf 800 x 800 Pixel skaliert. Zuletzt werden die Bilder um ein Vielfaches von 90 Grad rotiert und zufällig horizontal, sowie vertikal gespiegelt. Dadurch werden alle möglichen Neigungs- und Schattenwurfsrichtungen der WEA berücksichtigt.

### 3.1.5. Normalisierung

Damit der Backpropagation-Algorithmus effektiv lernen kann, wird auf jeden der RGB-Kanäle einzeln die Z-Score Normalisierung angewandt. Sei  $X$  der Wert des Pixel für einen der Kanäle.  $\mu$  und  $\sigma$  sind der Erwartungswert und die Standardabweichung für

den entsprechenden Kanal. Um diese zu kennen muss man zuvor über alle Bilder und Kanäle iterieren. Welchen Werte diese Größen für den Datensatz dieser Arbeit haben, kann dem Anhang entnommen werden. Der Z-Score berechnet sich dann wie folgt:

$$Z = \frac{X - \mu}{\sigma} \quad (3.1)$$

Durch die Subtraktion des Erwartungswertes werden die Pixelwerte zunächst auf einen neuen Erwartungswert von 0 gebracht. Die Division durch die Standardabweichung erreicht, dass die Werte eine Varianz von 1 haben. Diese Skalierung sorgt dafür, dass die Aktualisierung der Gewichte ausgeglichener wird, was sich ebenfalls positiv auf das Lernverhalten des Backpropagation-Algorithmus auswirkt [18].

## 3.2. Implementierung

### 3.2.1. Frameworks und Bibliotheken

Es wird eine Implementierung des Faster R-CNN mit Tensorflow 1.12.0 und Keras 2.2.4 von Github.com als Grundlage verwendet [45]. Tensorflow ist ein Open-Source Framework, welches das Arbeiten mit mehrdimensionalen Datenfeldern optimiert und daher in vielen Machine und Deep Learning Problemen Anwendung findet. Keras ist eine Open-Source Deep Learning Bibliothek [46], welche unter anderem Tensorflow als Backend nutzen kann, um Standard Anwendungen von Deep Learning Problemen möglichst einfach umsetzbar zu machen. Die Implementierung wurde für diese Arbeit, so angepasst, dass sie mit Tensorflow 2 kompatibel ist.

### 3.2.2. Netzarchitektur

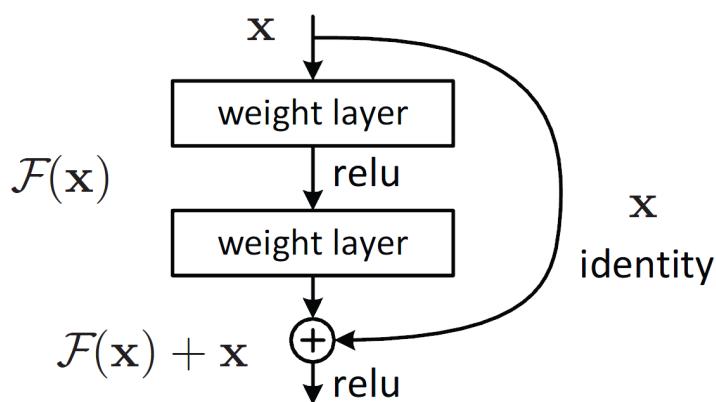


Abbildung 3.6.: Residual Learning: Ein Blockbaustein

Das Faster R-CNN Modell hat, wie bereits in Abschnitt 2.4 beschrieben, als Grundgerüst (engl. Backbone) ein Fully Convolutional Neural Network. Für diese Arbeit wird das *ResNet-50* als Backbone verwendet, welches sich der Technik des Residual Learning bedient. Die Zahl 50 steht für die Anzahl an CLs die das Netzwerk beinhaltet. Tiefere Netze mit mehr CLs bedeuten nicht automatisch bessere Resultate. Tatsächlich wurde das Gegenteil in Experimenten gezeigt [21]. Das liegt daran, dass tiefe Netze schwieriger zu optimieren sind. Theoretisch betrachtet sollten tiefe Netze, im Vergleich zu flacheren, mindestens gleich gute Resultate erzielen können. Das lässt sich einfach belegen indem man sich vorstellt, dass das tiefe Netz alle Schichten mit dem flachen Netz teilt und am Ende lediglich Schichten hinzugefügt sind, welche die Identität abbilden. Somit würden die Netze das gleiche Resultat liefern. Dieser Gedanke wird für das Residual Learning aufgegriffen. Es werden Abkürzungen mit der Identität zwischen Schichten im Netz hinzugefügt. Dafür wird, wie in Abb. 3.6 dargestellt, die Eingabe  $x$  einer Schicht über eine Abkürzung zur Ausgabe  $\mathcal{F}(x)$  einer Schicht hinzugefügt. Sollte nun der Fall eintreten, dass es sinnvoller ist eine Schicht zu überspringen, kann die Ausgabe  $\mathcal{F}(x)$  gegen 0 gehen. Das ist für das

Netz einfacher zu trainieren, als dass  $\mathcal{F}(x)$  die Identität abbildet [21]. Damit ist das Netz in der Lage die volle Tiefe zu gebrauchen, wenn es notwendig ist, und gleichzeitig wie ein flacheres Netz zu agieren, indem Schichten übersprungen werden.

Das ResNet-50 ist aufgebaut aus insgesamt 16 Blöcken, wie sie in Abb. 3.7 dargestellt sind. Dabei hat jeder Block eine Abkürzung zum nächsten (in Abb. 3.7 vom Input zur Addition der Identität und der Ausgabe der CLs). Es sind immer 3 CLs pro Block, die mit Schichten für die Batch Normalisierung und der ReLU Aktivierungsfunktion komplettiert werden. Das verwendete ResNet-50 wird in allen Experimenten mit vortrainierten Gewichten initialisiert, mit denen das Netz in einem Bilderkennungswettbewerb angekommen ist [48]. Damit kann, im Vergleich zu einer zufälligen Initialisierung der Gewichte, ein Teil der Zeit zum Trainieren eingespart werden. Die Gewichte sind abrufbar auf [49].

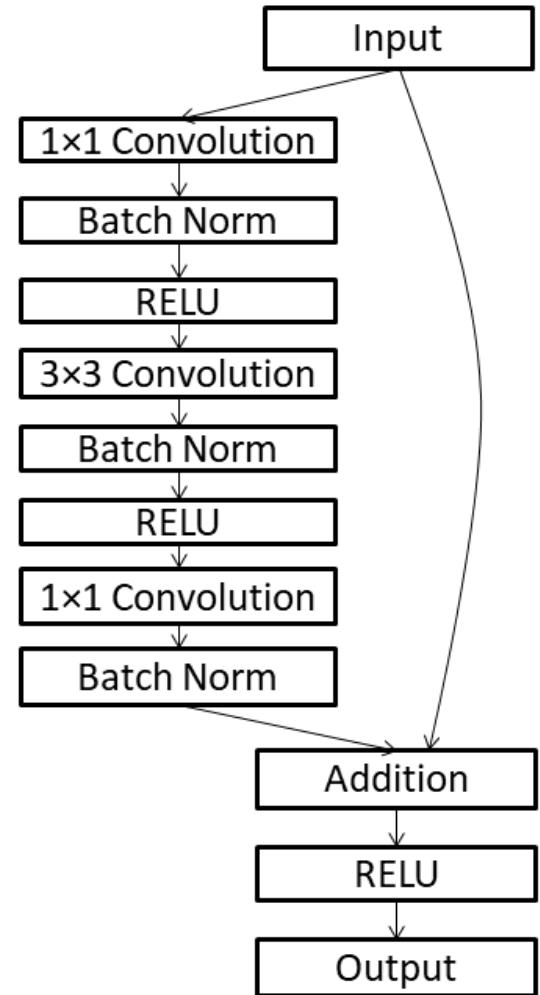


Abbildung 3.7.: Ein Convolutional Block eines ResNet [47]

### 3.2.3. Modifikation des Faster R-CNN

Wie kann das Faster R-CNN modifiziert werden, damit es das zugrunde liegende Problem dieser Arbeit lösen kann? Da nur WEA auf Bildern erkannt werden sollen, handelt es sich um eine binäre Klassenverteilung: WEA oder keine WEA, also Hintergrund. Im Grunde löst das RPN ebenfalls ein binäres Klassifizierungsproblem, indem es Objekte und Hintergrund im Bild prädiziert. Eventuell kann das RPN so trainiert werden, dass die zu erkennenden Objekte nur WEA sind. Der Teil des Classifiers aus der Faster R-CNN Architektur (s. Abb. 2.11) wäre obsolet, da schon die Objektvorschläge des RPN Prädiktionen der Standorte von WEA darstellen würden.

Außerdem ist, wie bereits in der Einleitung erwähnt, lediglich der Mittelpunkt der Anlage von Interesse nicht der gesamte Umriss. Eine Möglichkeit ist es, lediglich den Mittelpunkt der Prädiktionen des RPN, bzw. der gesamten Faster R-CNN Pipeline, zu betrachten. Eine andere ist es das RPN so zu modifizieren, dass es anstatt von Boxen lediglich Koordinaten im Bild prädiziert. Dadurch fallen die Anchor Boxen weg, die Anchors jedoch nicht. Von diesen erhält dann jeder ein Label, ob sich auf diesem eine WEA befindet oder nicht. Das *cls layer* (s. Abb. 2.12) bleibt dafür wie gehabt, allerdings mit  $k = 1$ , da es keine verschiedenen Anchor Box Größen und Seitenverhältnisse gibt. Das *reg layer* hat nicht mehr  $4 * k$  Ausgaben, sondern nur noch 2 für eine Koordinate pro Anchor Position. Die Verlustfunktion verändert sich demnach gegenüber Formel 2.15 nur minimal im Regressionsteil. Der  $smooth_{L_1}$  iteriert nur noch über zwei Werte, ansonsten bleibt alles beim Alten. Was allerdings angepasst werden muss ist wie die Anchors pro Mini-Batch ausgewählt werden. Es gibt viel mehr negative Anchors als positive, da sich auf einem Bild meist nur wenige Anlagen befinden. Deshalb werden pro Mini-Batch zunächst alle positiven Anchors berücksichtigt und dann gleich viele negative Anchors per Zufall ausgewählt. Für den Regressionsterm tragen dann, wie auch in Formel 2.15, nur positive Instanzen zum Verlust bei. Die Trainingsdaten müssen, anstatt mit Boxen um die Anlagen, lediglich mit den Koordinaten der Anlagen annotiert sein.



# 4. Evaluation

In diesem Kapitel werden zunächst die Metriken erläutert, mit denen die verschiedenen Modelle aus den Experimenten evaluiert werden. Danach folgt der allgemeine Experimentaufbau, sowie einzelne Abschnitte zu den Experimenten. In diesen wird zunächst erläutert, inwiefern sich das Modell von den anderen unterscheidet. Weiterhin folgen die Ergebnisse des Modells und eine Diskussion dieser.

## 4.1. Metriken

Zunächst wird geklärt, unter welchen Umständen eine Prädiktion als korrekt bzw. falsch eingestuft wird. Für eine korrekte Prädiktion, also ein True Positive (TP), muss die IoU mit einer GTB über einem Schwellwert  $\lambda_{IoU}$  liegen. Liegt sie unter  $\lambda_{IoU}$  wird die Prädiktion als False Positive (FP) gewertet. Dies gilt ebenso, wenn es keine Überschneidung der Prädiktion mit einer GTB gibt. Ein False Negative (FN) liegt vor, wenn es für eine GTB keine Prädiktion gibt, welche eine IoU mit der GTB über dem Schwellwert hat. True Negatives (TNs) existieren in dieser Problemstellung nicht, da es keine negative Klasse gibt, die explizit detektiert werden soll. Für die Evaluation der Performanz eines Modells werden aus diesen Werten folgende Größen berechnet.

**Precision (P):** Das Verhältnis von korrekten Prädiktionen zu allen Prädiktionen.

$$P = \frac{TP}{TP + FP}$$

**Recall (R):** Das Verhältnis von erkannten Objekten zu allen tatsächlichen Objekten.

$$R = \frac{TP}{TP + FN}$$

**F1-Score (F1):** Das Harmonische Mittel aus Precision und Recall, da diese allein keine hinreichende Aussage über die Qualität des Modells ermöglichen.

$$F1 = \frac{2PR}{P + R}$$

**Average Precision (AP):** Hierzu wird eine Precision-Recall-Kurve gezeichnet, die wie-der ergibt wie sich die Precision zu bestimmten Recall Schwellen verhält. Dafür werden alle

TPs und FPs mit ihrem Confidence Score in eine Liste aufgenommen und absteigend nach dem Score sortiert. Die AP ist die Fläche unter dieser Kurve, und daher vom Recall, sowie der Precision abhängig. Einzelheiten hierzu können in [50] nachgelesen werden. In dieser Arbeit wird die Implementierung der AP aus der *scikit-learn* Bibliothek verwendet [51].

**Durchschnittliche Distanz der Mittelpunkte ( $\bar{D}$ ):** Es wird außerdem die durchschnittliche Distanz der Mittelpunkte von GTB und prädizierter Box in Metern berechnet. Diese ist natürlich nur in Experimenten messbar, in denen ein klassisches RPN<sup>1</sup> ohne bzw. mit Classifier getestet wird. Dabei werden nur die prädizierten Boxen von TPs betrachtet. Da das eigentliche Problem das Ermitteln der Koordinaten der Anlagen ist, stellt diese Größe ein gutes Maß der Genauigkeit dar.

### Unterteilung in Größenkategorien:

Um besser einschätzen zu können wie die Modelle mit den unterschiedlichen Größen der Anlagen umgehen werden die Anlagen in 3 Größenkategorien unterteilt. In Abb. 4.2 sieht man ein Histogramm der Summen aus Nabenhöhe und Rotordurchmesser von allen Anlagen des Datenbestandes aus Deutschland. Welche Größen Nabenhöhe und Rotordurchmesser bei einer WEA darstellen, ist in Abb. 4.1 zu sehen. Die 3 Kategorien sind klein (gelb), mittel (grün) und groß (blau). Für diese wird jeweils die Precision ( $P_k, P_m, P_g$ ), der Recall ( $R_k, R_m, R_g$ ), sowie die durchschnittliche Distanz der Mittelpunkte ( $\bar{D}_k, \bar{D}_m, \bar{D}_g$ ) gesondert gemessen. Für die Precision der einzelnen Kategorien werden Prädiktionen, die sich mit keiner GTB schneiden nicht berücksichtigt. Es wird demnach ein FP einer Kategorie nur zugewandt, wenn sich die prädizierte Box mit der GTB schneidet, die IoU aber unter dem Schwellwert  $\lambda_{IoU}$  liegt.

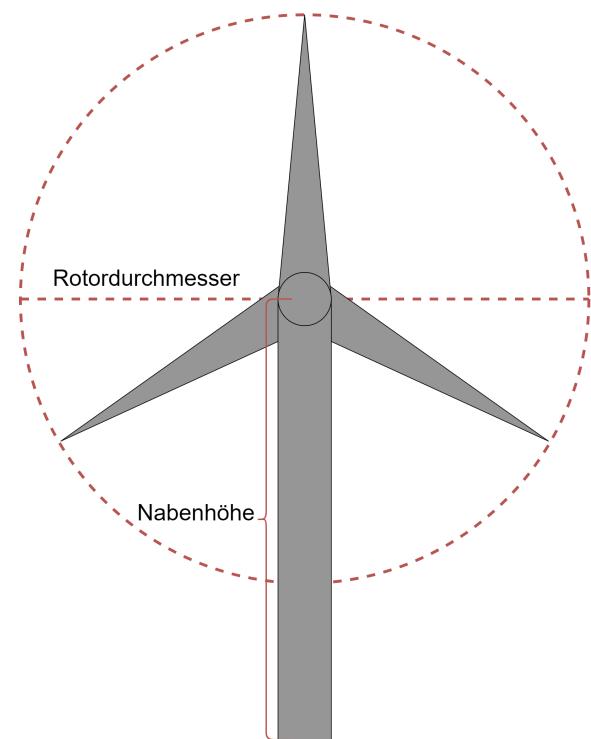


Abbildung 4.1.: Schematische Darstellung einer WEA

<sup>1</sup>Gemeint ist ein RPN das Boxen prädiziert, nicht nur Koordinaten, wie in Abschnitt 3.2.3 beschrieben.

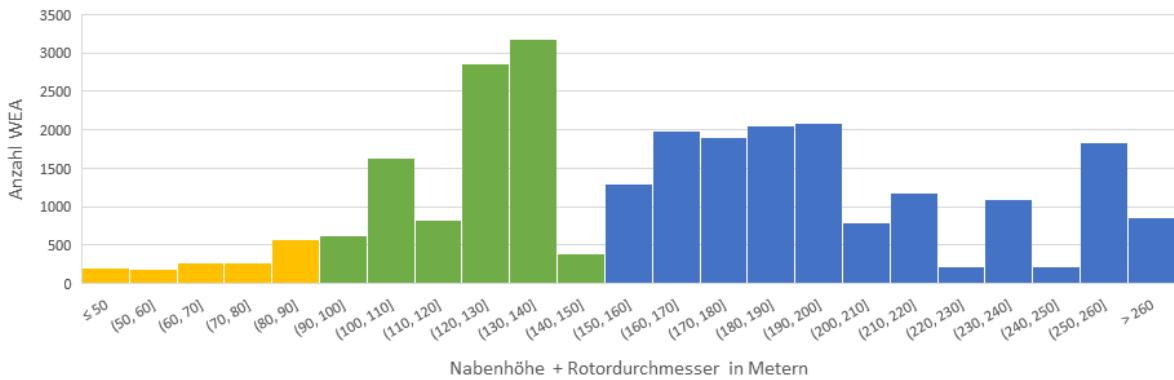


Abbildung 4.2.: Histogramm der Anlagengrößen. Die Größenkategorien sind in gelb, grün und blau markiert.

## 4.2. Experimente

Das RPN wird in allen Experimenten für 30 Epochen<sup>2</sup> trainiert. Der Classifier der Faster R-CNN Pipeline (s. Abschnitt 2.4) wird anschließend, in den entsprechenden Experimenten, zusammen mit dem RPN für 10 Epochen trainiert. Die Wahl aller für das Training relevanten Parameter ist im Anhang zu finden. An dieser Stelle werden nur die entscheidenden Parameter erläutert, die sich in den Modellen unterscheiden. Zu erwähnen ist, dass in allen Experimenten eine strikte NMS, mit einem IoU Schwellwert minimal größer als 0, angewandt wird. Dies kann so gemacht werden, da Anlagen sich nie so nahe kommen, dass sich ihre Boxen überlappen würden. In den Experimenten mit RPN und Classifier wird die sehr strikte NMS lediglich auf die finalen Prädiktionen des Classifiers angewandt, nicht aber auf die Objektvorschläge des RPN, für welche der Schwellwert für die NMS bei 0,3 liegt. Die Annahme, dass Anlagen sich nie sehr nahe stehen, gilt allerdings nicht für Windparks aus den USA. Aus diesem Grund wird für das entsprechende Experiment der Schwellwert für die NMS auf 0,1 erhöht.

### 4.2.1. Modifiziertes RPN allein

Zunächst wird das modifizierte RPN, welches lediglich Koordinaten der Anlagen prädizieren soll (s. Abschnitt 3.2.3), trainiert. Das Netz hat insgesamt 13.279.235 trainierbare Parameter. In Abb. 4.3 sind die Verlustkurven von Trainings- und Validationset zu sehen. Da der Verlust nicht gegen vertretbar kleine Werte konvergiert wird auf einen Test des Modells verzichtet. Um nachzuvollziehen wie es zu solch hohen Verlusten im

<sup>2</sup>Eine Epoche entspricht einer Iteration über alle 17820 Trainingsbilder.

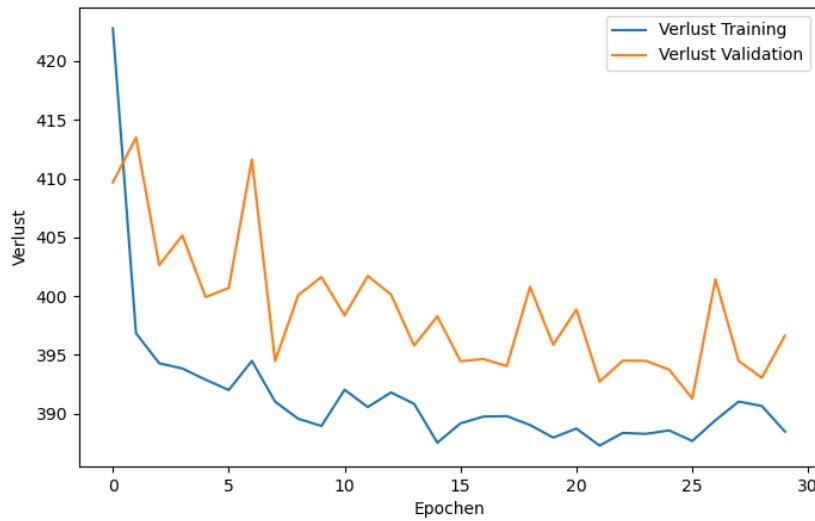


Abbildung 4.3.: Verlustkurve des modifizierten RPN

Training kommen kann, ist eine eingehende Untersuchung der Vorgänge nötig, die im Rahmen dieser Arbeit nicht in Angriff genommen wurden. Stattdessen wird in im Weiteren der Ansatz verfolgt, die Mittelpunkte der Boxen als Prädiktion der Koordinaten der Anlagen anzusehen (s. 3.2.3).

#### 4.2.2. RPN allein mit variablen Box Größen

In diesem Experiment wird ein klassisches RPN trainiert, um herauszufinden ob sich die Hypothese aus Abschnitt 3.2.3 bestätigen lässt, dass das RPN alleine bereits zufriedenstellende Ergebnisse liefert. Das Netz hat insgesamt 13.282.826 trainierbare Parameter. Die Boxen um die Anlagen in den Trainingsbildern nehmen immer relativ wenig Platz im Bild ein und sind stets quadratisch (s. bspw. Abb. 3.2). Aus diesen Gründen sind viele Anchor Box Seitenverhältnisse bzw. Größen im Training des RPN nicht notwendig und würden es unnötig verlangsamen. Deshalb werden in diesem Experiment Anchor Box Größen von 30 x 30 und 50 x 50 Pixel, sowie das Seitenverhältnis 1 x 1 verwendet. Die Größe der Bounding Boxen der Anlagen ist abhängig von der Summe der Nabenhöhe und dem Rotordurchmesser gewählt. Den geringsten Verlust im Validation Set von 1,45 hatte dieses Modell nach 23 Epochen. Für den Test der Performanz wird mit verschiedenen Schwellwerten gearbeitet: Einerseits mit  $\lambda_{obj}$ , also ab wann ein Objektscore des RPN als Prädiktion akzeptiert wird und andererseits mit  $\lambda_{IoU}$ , also ab welcher IoU mit einer GTB eine Prädiktion als TP zählt. Dieser Schwellwert ist im Vergleich zu anderen Objektdetektionsproblemen relativ niedrig angesetzt, da für dieses Problem die Mittelpunkte entscheidend sind, nicht die exakten Umrissse einer Anlage.

In den qualitativen Ergebnissen wird es Beispiele geben, in denen sich die Größen von GTB und prädizierter Box deutlich unterscheiden, die Mittelpunkte jedoch nicht stark voneinander abweichen. Diese Beispiele sind gute Prädiktionen, würden jedoch durch einen zu hohen Schwellwert  $\lambda_{IoU}$  als FP zählen. Außerdem wird mit der mittleren Distanz der Mittelpunkte eine Performanzgröße gemessen, die es bestraft, falls durch einen niedrigeren Schwellwert Prädiktionen als TP zählen, welche eine Anlage nicht mittig einfassen. Die Ergebnisse des Modells sind in Tabelle 4.1 aufgelistet.

$\lambda_{obj}$	$\lambda_{IoU}$	P	R	F1	AP	$\bar{D}$	$P_k$	$P_m$	$P_g$	$R_k$	$R_m$	$R_g$	$\bar{D}_k$	$\bar{D}_m$	$\bar{D}_g$
0,9	0,1	0,36	<b>0,95</b>	0,52	0,65	6,63	0,93	0,99	0,98	<b>0,81</b>	<b>0,95</b>	<b>0,96</b>	6,04	6,47	6,77
0,9	0,2	0,34	0,92	0,50	0,63	5,97	0,66	0,95	0,96	0,58	0,91	0,94	6,03	6,07	5,9
0,9	0,3	0,32	0,84	0,46	0,6	5,71	0,05	0,81	0,95	0,04	0,78	0,93	6,18	5,88	5,61
0,9	0,4	0,27	0,71	0,39	0,52	5,47	0,003	0,51	0,92	0,003	0,49	0,9	8,43	5,67	5,39
0,999	0,1	0,46	0,94	0,62	0,65	6,61	0,93	0,99	0,98	0,8	0,93	0,95	6,05	6,44	6,76
0,999	0,2	0,45	0,91	0,60	0,63	5,96	0,66	0,95	0,96	0,57	0,9	0,93	6,05	6,05	5,89
0,999	0,3	0,41	0,83	0,55	0,59	5,69	0,05	0,81	0,95	0,04	0,77	0,92	6,18	5,87	5,58
0,999	0,4	0,35	0,7	0,46	0,52	5,46	0,003	0,51	0,92	0,003	0,48	0,89	8,43	5,66	5,39
0,99999	0,1	<b>0,6</b>	0,9	<b>0,72</b>	<b>0,65</b>	6,45	<b>0,94</b>	<b>0,99</b>	<b>0,98</b>	0,68	0,87	0,93	6,05	6,29	6,58
0,99999	0,2	0,58	0,87	0,7	0,63	5,88	0,65	0,96	0,96	0,47	0,84	0,91	5,99	5,96	5,82
0,99999	0,3	0,54	0,8	0,65	0,6	5,66	0,03	0,82	0,95	0,02	0,72	0,90	<b>5,36</b>	5,81	5,57
0,99999	0,4	0,46	0,68	0,55	0,52	<b>5,45</b>	0,004	0,51	0,93	0,003	0,45	0,88	8,43	<b>5,65</b>	<b>5,37</b>

Tabelle 4.1.: Ergebnisse RPN allein mit variablen Box Größen

Dieser Ansatz liefert i.A. einen hohen Recall, es wird also ein Großteil der Anlagen auf den Bildern tatsächlich erkannt. Das geht jedoch einher mit einer sehr großen Menge an FPs, die i.A. eine eher geringe Precision nach sich ziehen. Mit Werten zwischen 5,45 und 6,63 Metern erreicht  $\bar{D}$  eine hohe Genauigkeit.

Es ist zu sehen, dass sich die Precision mit einem höheren Schwellwert  $\lambda_{obj}$  zwar verbessern lässt, allerdings dennoch auf einem geringem Niveau bleibt. Die Erhöhung dieses Schwellwerts führt außerdem dazu, dass der Recall ( $R_k$ ,  $R_m$ ,  $R_g$ ) je stärker abnimmt, desto kleiner die Anlagen sind. Bei  $\lambda_{IoU} = 0,1$  ist dieser Effekt am deutlichsten zu sehen. Je kleiner eine Anlage ist, desto unsicherer ist sich demnach das RPN, ob es sich um ein Objekt handelt. Das könnte damit erklärt werden, dass diese Anlagen auf den Bildern schlichtweg zu wenig Platz einnehmen und dadurch zu wenig Informationen liefern, die es dem Netz ermöglichen bestimmte Merkmale zu identifizieren, die auf ein Objekt hindeuten. Ein möglicher Ansatz dieses Problem anzugehen, wäre es bei Anlagen die besonders klein sind Bilder mit einer größeren Auflösung zu wählen. Mit der Google Maps Static API wäre dies über die Erhöhung der Zoom Stufe möglich. Das setzt natürlich voraus, dass die Größen der Anlagen im vornherein bekannt sind und ist daher nur für die Problemstellung geeignet, dass ein bereits bekannter Standort geprüft werden soll. Weiterhin kann beobachtet werden, dass mit einem höheren Schwellwert  $\lambda_{IoU}$  ebenfalls der Recall, sowie die Precision abnehmen. Das war grundsätzlich erwartbar, da durch

höheren  $\lambda_{IoU}$  weniger oft Prädiktionen als positiv gelten. Dieser Effekt hängt erneut davon ab wie groß die Anlagen sind. Während  $P_g$  und  $R_g$  nur sehr gering abnehmen, ist der Unterschied bei  $P_k$  und  $R_k$  umso deutlicher. Das deutet darauf hin, dass die Prädiktionen je stärker von der GTB abweichen, desto kleiner die Anlagen sind. Die Abnahme von  $\overline{D}$ , sowie  $\overline{D}_k$ ,  $\overline{D}_m$  und  $\overline{D}_g$  mit höherem Schwellwert  $\lambda_{IoU}$  liegt mit großer Sicherheit daran, dass nur noch sehr genaue Prädiktionen, welche über dem Schwellwert liegen, als TP zählen und somit zur mittleren Distanz beitragen.

In den qualitativen Ergebnissen (s. Abb. 4.4) lassen sie die Beobachtungen aus den Daten in Tabelle 4.1 wiederfinden. Anlagen in den Bildern werden gefunden und die Mittelpunkte der prädizierten Box und der GTB divergieren selten stark voneinander. Allerdings werden häufig Objekte in den Bildern erkannt, die zwar Objekte darstellen, für die Aufgabenstellung jedoch irrelevant sind. Beispiele in Abb. 4.4 sind Gebäude, LKW auf Straßen (s. oben rechts), Freileitungsmasten (s. unten links) oder auch die Gondel mit Rotorblättern einer aus spitzen Winkel aufgenommenen Anlage (s. unten rechts). Das könnte zum Teil an den Fehlern in den Trainingsdaten liegen. Wahrscheinlicher ist aber vermutlich, dass dies zum größeren Teil durch die Architektur des RPN bedingt ist. Es ist explizit dafür entwickelt interessante Bereiche im Bild zu identifizieren, welche der Classifier in einem zweiten Schritt einordnet und ggf. aussortiert. Insofern liefert das untersuchte RPN nachvollziehbare Ergebnisse.

Zu beachten ist dabei außerdem, dass es sich bei den Testbildern stets um Bilder von Windparks handelt. Diese stehen i.d.R. nicht nahe an anderen Strukturen wie Gebäuden, Straßen, etc.. Viele Möglichkeiten andere Objekte als WEA in den Bildern zu erkennen bleiben dem Netz demnach oft nicht. Wenn ein großer Bereich auf der Suche nach WEA abgescannt werden soll (s. Ziele in Abschnitt 1.2), hätte das Modell vermutlich eine sehr viel schlechtere Precision, da in einem solchen Bereich seltener Anlagen auftauchen, die es erkennen könnte.

Die eher ernüchternden Ergebnisse dieses Experiments widerlegen die Eingangshypothese, dass das RPN bereits alleine ausreichend gute Prädiktionen der Anlagen produzieren kann. Aus diesem Grund wird in einem weiteren Experiment zusätzlich zum RPN der Classifier der Faster R-CNN Pipeline trainiert.

### 4.2.3. RPN + Classifier mit variablen Box Größen

Für dieses Experiment wird das bereits trainierte RPN aus dem vorherigen Experiment verwendet und zusammen mit dem Classifier für 10 Epochen trainiert. Das Netz hat in diesem Fall insgesamt 34.287.120 trainierbare Parameter. Der Classifier wird im Training pro Bild mit insgesamt 15 Objektvorschlägen des RPN konfrontiert. Darunter befinden sich bis zu 7 positive Prädiktionen, sowie bis zu 5 negative Prädiktionen mit dem höchsten Objektscore. Der Rest wird mit zufälligen negativen Prädiktionen aufgefüllt. Dahinter steckt der Gedanke, dass die häufigen Fehldetections aus dem Experiment zuvor



Abbildung 4.4.: Qualitative Ergebnisse: RPN allein mit Schwellwert  $\lambda_{IoU} = 0,9$ . Die Prädiktionen mit Objektscore sind in rot eingezeichnet, die GTBs in blau.

dem Classifier explizit als falsches Beispiel vorgesetzt werden sollen (Hard Negative Mining). Nach der zweiten Epoche wurde der geringste Verlust im Validationset (0,71) erreicht. Schwellwerte für den Objektscore des RPN fallen weg, stattdessen gibt es einen

Schwellwert für den Confidence Score des Classifiers (0,8), ab wann eine Prädiktion als valide angesehen wird. Die Schwellwerte  $\lambda_{IoU}$  bleiben wie zuvor. Die Ergebnisse sind in Tabelle 4.2 aufgelistet.

$\lambda_{IoU}$	$P$	$R$	$F1$	$AP$	$\bar{D}$	$P_k$	$P_m$	$P_g$	$R_k$	$R_m$	$R_g$	$\bar{D}_k$	$\bar{D}_m$	$\bar{D}_g$
0,1	<b>0,98</b>	<b>0,74</b>	<b>0,84</b>	<b>0,99</b>	7,87	<b>1</b>	<b>0,99</b>	<b>1</b>	<b>0,06</b>	<b>0,61</b>	<b>0,82</b>	10,72	10,03	6,84
0,2	0,93	0,7	0,8	0,97	7,13	0,29	0,85	0,99	0,02	0,52	0,82	<b>5,65</b>	8,2	6,71
0,3	0,81	0,61	0,69	0,91	5,91	0	0,53	0,96	0	0,33	0,79	-	<b>5,74</b>	5,96
0,4	0,6	0,45	0,52	0,74	<b>5,26</b>	0	0,17	0,82	0	0,1	0,68	-	6,2	<b>5,17</b>

Tabelle 4.2.: Ergebnisse RPN + Classifier mit variablen Box Größen

Im Vergleich zum ersten Experiment hat dieser Ansatz i.A. eine deutlich bessere Precision, während der Recall geringer ausfällt. Erneut ergeben sich die besten Werte für Recall und Precision bei  $\lambda_{IoU} = 0,1$ , wobei die mittlere Distanz mit 7,87 Metern noch zufriedenstellend niedrig ausfällt. Auffällig ist, dass der Recall der kleinen Anlagen ( $R_k$ ) deutlich geringer ist als zuvor.

Das linke obere Bild in den qualitativen Ergebnisse in Abb. 4.5 zeigt denselben Windpark wie in Abb. 4.4. Daran, sowie an den sehr hohen Precision Werten, ist zu sehen, dass es nur noch äußerst selten zu Fehldetections (FPs) kommt. Außerdem enthalten ca. 10% der Testbilder (s. Abschnitt 3.1.2), aufgrund von Fehlern im Datensatz oder einer zu alten Aufnahme, eine WEA ohne annotierte Box. Erkennt das Modell an dieser Stelle dennoch eine Anlage, wie bspw. in zwei Fällen in Abb. 4.5 rechts oben, wird diese als FP gezählt, obwohl eigentlich ein TP vorliegt. Dadurch liegt die tatsächliche Precision vermutlich sogar noch ein bisschen höher.

Der schlechte Recall für kleine Anlagen hat vermutlich ähnliche Gründe wie auch schon im Experiment zuvor, nämlich die Schwierigkeit markante Features an kleinen Anlagen auszumachen. Dem generell schlechteren Recall von Anlagen aller Größen, könnte zugrunde liegen, dass durch das Hard Negative Mining im Training, der Classifier mit negativen Prädiktionen mit hohem Objektscore konfrontiert wird. Durch Fehler in den Trainingsdaten könnten diese negativen Prädiktionen in Wirklichkeit positiv sein. Eventuell verschlechtert dieser Umstand die Entscheidungssicherheit des Modells und es wird fälschlicherweise Hintergrund, anstatt einer WEA erkannt.

Auffällig ist, dass das Modell in allen Fällen in Abb. 4.5 eine größere Box als die GTB präzisiert. Bei kleinen Anlagen ist dieser Umstand sehr eklatant (s. unten rechts in Abb. 4.5). Dadurch lassen sich die stark sinkende Precision- und Recallwerte mit steigenden Schwellwert  $\lambda_{IoU}$  erklären. Außerdem deutet dies darauf hin, dass der Classifier bei größeren Boxen eine höhere Confidence erzielt, als bei kleineren. Das könnte damit erklärt werden, dass Teile der Nabe und der Rotorblätter, welche teils erst mit einer größeren Box um die Anlage erfasst werden (s. Abb. 4.5 links oben), eventuell wichtige Erkennungsmerkmale für den Classifier sind. Der Gedanke liegt nahe, dass das Netz



Abbildung 4.5.: Qualitative Ergebnisse: RPN + Classifier mit variablen Box Größen. Die Prädiktionen mit Confidence Score sind in rot eingezeichnet, die GTBs in blau.

eventuell besser präzisieren könnte, wenn die Boxen im Training generell größer gewählt werden.

#### 4.2.4. RPN + Classifier mit 3 diskreten Box Größen

Das Netz für dieses Experiment hat insgesamt 34.289.685 trainierbare Parameter. Es wird untersucht welche Auswirkungen es hat, wenn die Größen der Boxen um die Anlagen generell größer gewählt sind. Außerdem sind sie nicht wie zuvor variabel sind, sondern haben diskrete Größen entsprechend der 3 Kategorien aus Abb. 4.2. Im Training werden die Anchor Box Größen exakt auf die 3 verschiedenen Box Größen angepasst. Es ergeben sich die Anchor Box Größen: 40 x 40, 58 x 58 und 70 x 70 Pixel. Trainiert wird wieder zunächst das RPN für 30 Epochen. Getestet wird es diesmal nicht, da sich bereits gezeigt hat, dass die Ergebnisse des RPN allein nicht zufriedenstellend sind. Erneut wurde der geringste Verlust des Validationset (1,4) nach der 23. Epoche erreicht. In den 10 Epochen, in denen das RPN gemeinsam mit dem Classifier trainiert wurde, trat der geringste Verlust im Validationset (0,41) nach der zweiten Epoche auf. Die Schwellwerte für den Test der Performanz sind wie im Experiment zuvor gewählt. Die Ergebnisse sind in Tabelle 4.3 aufgelistet.

$\lambda_{IoU}$	P	R	F1	AP	$\bar{D}$	$P_k$	$P_m$	$P_g$	$R_k$	$R_m$	$R_g$	$\bar{D}_k$	$\bar{D}_m$	$\bar{D}_g$
0,1	<b>0,97</b>	<b>0,86</b>	<b>0,91</b>	<b>0,98</b>	13,36	<b>1</b>	<b>1</b>	<b>1</b>	<b>0,6</b>	<b>0,84</b>	<b>0,89</b>	8,93	12,63	13,99
0,2	0,95	0,84	0,89	0,95	12,84	0,58	0,97	0,98	0,35	0,81	0,88	9,47	11,87	13,55
0,3	0,87	0,77	0,82	0,84	11,01	0,15	0,9	0,91	0,09	0,75	0,82	11,48	10,16	11,56
0,4	0,72	0,64	0,68	0,66	<b>9,16</b>	0	0,68	0,81	0,003	0,57	0,73	<b>8,78</b>	<b>8,61</b>	<b>9,48</b>

Tabelle 4.3.: Ergebnisse RPN + Classifier mit 3 diskreten Box Größen

Im Vergleich zum Experiment zuvor kann hier eine Verbesserung des Recalls, insbesondere für kleine Anlagen ( $R_k$ ), beobachtet werden. Dies ist gut im Vergleich der unteren beiden Ausschnitte der qualitativen Ergebnisse aus Abb. 4.5 und 4.6 zu erkennen. Allerdings geht die Verbesserung des Recalls mit einer deutlichen Verschlechterung der durchschnittlichen Distanzen der Mittelpunkte einher. Teilweise unterscheiden sich die prädictierten Boxen so sehr von den GTBs, dass es die Frage aufwirft, inwieweit noch Merkmale der tatsächlichen Anlage für die Prädiktion verantwortlich sind. Beispiele hierfür sind die untersten Anlagen in den Ausschnitten rechts und links oben in Abb. 4.6. Die prädictierten Boxen streifen die Anlage nur minimal und enthalten hauptsächlich Ackerfläche bzw. Feldwege und Bäume. In diesem Zusammenhang wäre es interessant zu visualisieren, welche Teile im Bild den Ausschlag für eine Entscheidung geben. Im Rahmen dieser Arbeit wurde dies allerdings nicht in Angriff genommen. Sollten tatsächlich auch andere Merkmale wichtig sein, welche nicht die WEA wiederspiegeln, würde dies in gewisser Weise die Grenzen des Ansatzes aufzeigen, automatisch Boxen um die Mittelpunkte der Anlagen zu annotieren.

Es ist nicht ganz eindeutig zu sagen, welcher der beiden Ansätze aus den Experimenten 4.2.3 und 4.2.4 besser geeignet ist, um die beiden Problemstellungen (s. Abschnitt 1.2) zu lösen. Eventuell funktionieren sie am sinnvollsten in Kombination miteinander. Zunächst



Abbildung 4.6.: Qualitative Ergebnisse: RPN + Classifier mit 3 diskreten Box Größen.  
Die Prädiktionen mit Confidence Score sind in rot eingezeichnet, die GTBs in blau.

wird das Modell mit dem höheren Recall (4.2.4) genutzt, um Anlagen zu identifizieren. Im Anschluss versucht das Modell mit höherer Genauigkeit (4.2.3) die Prädiktion der Mittelpunkte zu verbessern. Mit diesem Ansatz wird man beiden Problemstellungen gerecht, also der Überprüfung von bekannten Standorten, sowie der Lokalisierung von bisher unbekannten Standorten.

### 4.2.5. Geographische Abhangigkeit am Beispiel der USA

In diesem Experiment soll herausgefunden werden inwieweit das Modell aus Experiment 4.2.4 geographisch abhangig ist. Dafur wird es zunachst auf dem Datensatz aus den USA (s. Abschnitt 3.1.3) getestet. Im Anschluss wird das gesamte Netz fur 20 Epochen ausschlielich mit den Daten aus den USA nachtrainiert und dann erneut getestet. Dieses Nachtrainieren ist eine rudimentare Form des *Transfer Learning*, genauer gesagt der Domen Adaption. Das Modell erreichte nach der zweiten Epoche den geringsten Validationsverlust von 0,35. Die Ergebnisse sind in den Tabellen 4.4 und 4.5 zu sehen.

$\lambda_{IoU}$	$P$	$R$	$F1$	$AP$	$\bar{D}$	$P_k$	$P_m$	$P_g$	$R_k$	$R_m$	$R_g$	$\bar{D}_k$	$\bar{D}_m$	$\bar{D}_g$
0,1	<b>0,88</b>	<b>0,72</b>	<b>0,79</b>	<b>0,89</b>	15,34	<b>0,63</b>	<b>0,98</b>	<b>0,88</b>	<b>0,09</b>	<b>0,31</b>	<b>0,79</b>	35,43	15,65	15,21
0,2	0,87	0,71	0,78	0,88	14,64	0,05	0,92	0,88	0,01	0,29	0,78	<b>14,6</b>	12,85	14,71
0,3	0,84	0,68	0,75	0,82	13,24	0	0,86	0,84	0	0,27	0,75	-	10,51	13,33
0,4	0,77	0,63	0,69	0,74	<b>11,56</b>	0	0,65	0,78	0	0,2	0,7	-	<b>9,5</b>	<b>11,61</b>

Tabelle 4.4.: Ergebnisse RPN + Classifier mit 3 diskreten Box Groen fur Daten aus den USA ohne Transfer Learning

$\lambda_{IoU}$	$P$	$R$	$F1$	$AP$	$\bar{D}$	$P_k$	$P_m$	$P_g$	$R_k$	$R_m$	$R_g$	$\bar{D}_k$	$\bar{D}_m$	$\bar{D}_g$
0,1	<b>0,85</b>	<b>0,88</b>	<b>0,86</b>	<b>0,88</b>	19,42	<b>0,64</b>	<b>0,93</b>	<b>0,85</b>	<b>0,17</b>	<b>0,75</b>	<b>0,93</b>	30,03	21,96	19
0,2	0,85	0,87	0,86	0,88	18,61	0,28	0,91	0,85	0,08	0,73	0,93	24,3	20,46	18,36
0,3	0,82	0,84	0,83	0,86	17,31	0,13	0,83	0,83	0,03	0,67	0,9	<b>18,09</b>	<b>18,46</b>	17,18
0,4	0,7	0,72	0,71	0,75	<b>15,64</b>	0	0,33	0,75	0	0,27	0,82	-	22,15	<b>15,3</b>

Tabelle 4.5.: Ergebnisse RPN + Classifier mit 3 diskreten Box Groen fur Daten aus den USA mit Transfer Learning

Aus den Ergebnissen in Tab. 4.4 ist zunachst einmal zu erkennen, dass die Werte in allen Belangen schlechter sind, als die Daten aus Tab. 4.3. Das Modell weist demnach eine gewisse geographische Abhangigkeit von Daten aus Deutschland auf. Mithilfe des Transfer Learning wird ein besserer Recall erreicht, die durchschnittliche Distanz der Mittelpunkte verschlechtert sich jedoch deutlich. Diese Methode liefert demnach nicht ausschlielich eine Verbesserung. Eine vollwertige Domen Adaption wurde vermutlich ein besseres Ergebnis liefern, den Rahmen dieser Arbeit jedoch sprengen.

In den qualitativen Ergebnissen in Abb. 4.7 sind die Veranderungen durch das Transfer Learning in zwei Beispielen gegenubergestellt. Die Verbesserung des Recalls, sowie die Verschlechterung von  $\bar{D}$ , finden sich in den beiden oberen Bildern wieder. In den unteren beiden Bildern ist nochmal die Eigenheit mancher US-Windparks abgebildet, Anlagen sehr eng nebeneinander zu platzieren. Man sieht jeweils unten rechts in diesen Bildern, dass sich die Pradiktionen uberlappen, was mit einer sehr strikten NMS

zu einer Nichtdetektion mancher Anlagen führen würde. Das nachtrainierte Modell hat zwar im linken unteren Bild in der Ecke links oben zwei Prädiktionen von kleinen, eng zusammenstehenden Anlagen, allerdings bestehen weiterhin große Probleme mit diesen Windparkstrukturen, zumal die erwähnten Prädiktionen teils mehr als eine Anlage umfassen. Eventuell wäre es hilfreich Windparks dieser Art im vornherein zu identifizieren und dann für eine Prädiktion in die entscheidenden Ausschnitte herein zu zoomen.

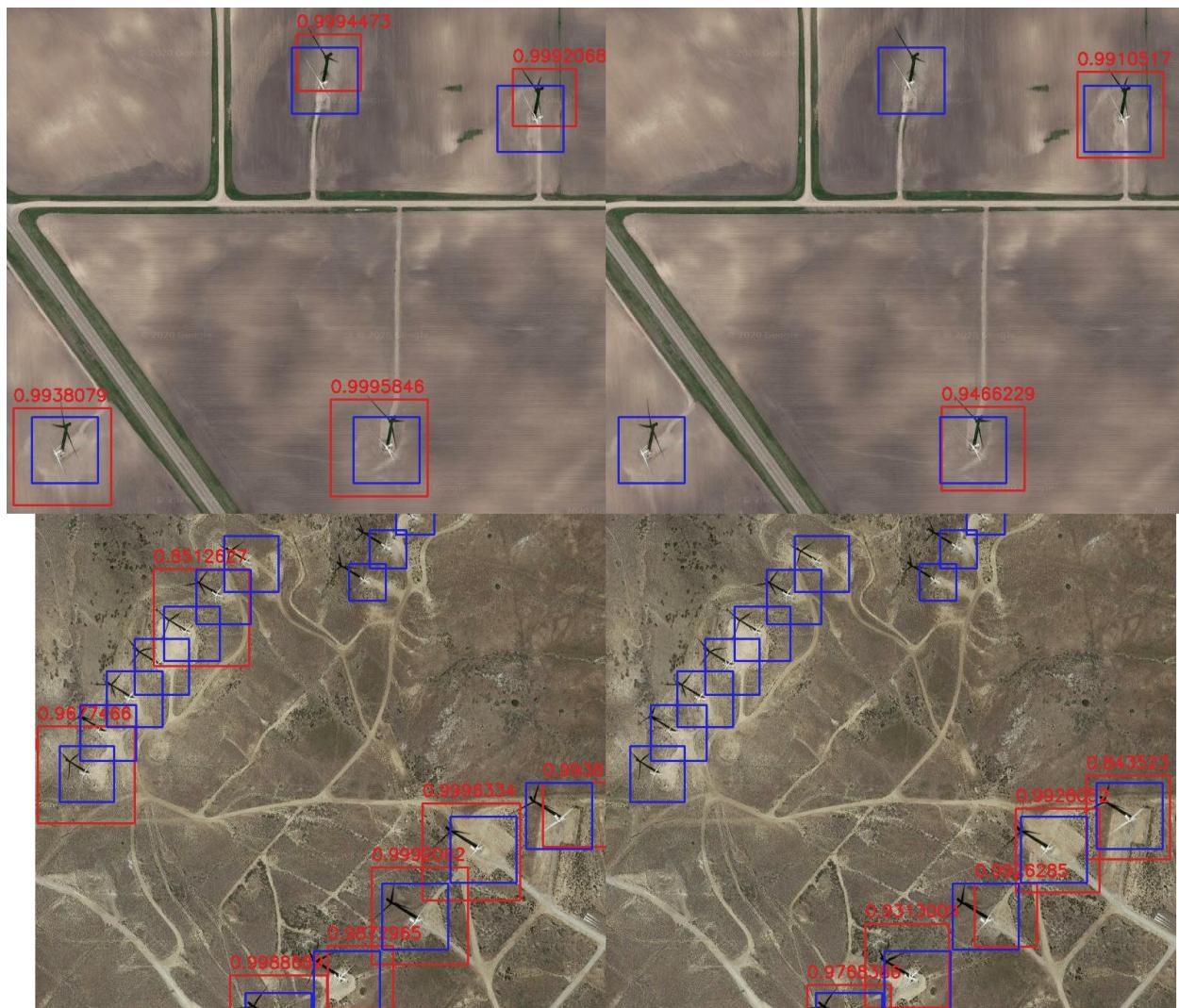


Abbildung 4.7.: Qualitative Ergebnisse: RPN + Classifier mit 3 diskreten Box Größen.  
Die Prädiktionen mit Confidence Score sind in rot eingezeichnet, die GTBs in blau. Rechts: Bilder ohne Nachtraining; Links: Bilder mit Nachtraining



# 5. Zusammenfassung und Ausblick

In dieser Arbeit wird untersucht inwiefern eine Lokalisierung von WEA aus Satellitenbildern mittels einer Objektdetektierung ermöglicht werden kann. Dabei ergeben sich zwei Problemstellungen. Es sollen einerseits bekannte Standorte geprüft, und andererseits neue Standorte ermittelt werden.

Trainings- und Testdaten liefert die Google Maps Static API im Zusammenspiel mit einer WEA-Datenbank aus Deutschland, sowie den USA. Die Standorte von WEA werden dafür, in Abhängigkeit der Größe der Anlage, automatisch für jedes der Bilder annotiert. Durch Fehler in der Datenbank, sowie das unbekannte Alter der Aufnahmen entstehen falsche Annotationen. Deshalb wird die Datenqualität (s. Abschnitt 3.1.2) für die Anlagen aus Deutschland evaluiert. Außerdem wird zur Verbesserung der Qualität ein Ansatz geliefert, der die Bilder nach ihrem Verlust im Training filtert, um solche mit Fehlern zu identifizieren. Für die Objektdetektierung wird das Modell Faster R-CNN mit einem ResNet-50 als Backbone verwendet. Dabei werden die Mittelpunkte der Prädiktionen des Modells als vorgeschlagene Standorte der Anlagen angesehen. Die Modelle werden zunächst lediglich mit Daten aus Deutschland trainiert. Als erstes wird der Ansatz verfolgt lediglich das RPN der Faster R-CNN Pipeline zu verwenden. Damit werden viele Anlagen erkannt ( $R = 0,9$ ), es gibt jedoch auch sehr viele Fehldetektionen von Objekten die keine WEA darstellen ( $P = 0,6$ ). Hier sei nochmal erwähnt, dass die Precision höher ausfällt, da die Trainingsbilder alle mindestens eine WEA abbilden. Für wahllose Satellitenbilder wäre der Wert vermutlich geringer. Aus diesen Gründen wird im Anschluss zusätzlich zum RPN der Classifier trainiert (s. Abschnitt 4.2.3). Mit diesem Ansatz lässt sich eine sehr gute Precision (0,98) erreichen, es gibt demnach nur noch sehr wenige Fehldetektionen. Allerdings verschlechtert sich der Recall zum Ansatz zuvor auf 0,74. Da das Netz stets größere Boxen als die GTBs prädiziert, wird in einem nächsten Experiment (s. Abschnitt 4.2.4) die Größe der GTBs erhöht. Damit kann der Recall auf 0,86 verbessert werden, was vor allem an der besseren Erkennung von kleineren Anlagen liegt. Die Precision sinkt minimal auf 0,97 und die Distanz der Mittelpunkte verschlechtert sich von durchschnittlich 7,87 auf 13,36 Meter. Eine Kombination der beiden Netze wäre eventuell in der Lage die jeweiligen Stärken zu vereinen und den Problemstellungen gerecht zu werden.

Dennoch stellt der Ansatz dieser Arbeit für Deutschland bereits jetzt eine Möglichkeit dar, bspw. die Prüfung des Marktstammdatenregisters oder die Pflege von WEA-Datenbanken in Teilen zu automatisieren. Mit akkurate Datenbanken können darauf aufbauende Analysen und Forschungen (s. Abschnitt 1.1) mit größerer Genauigkeit betrieben werden.

Durch die in allen Belangen schlechteren Ergebnisse des Modells für Daten aus den USA,

kann eine gewisse geographische Abhangigkeit festgestellt werden. Ein rudimentares Transfer Learning in Form eines Nachtrainings mit den Daten aus den USA liefert zwar einen besseren Recall, es verschlechtert allerdings auch die durchschnittliche Distanz der Mittelpunkte. Interessant ware in diesem Zusammenhang die Anwendung einer vollumfanglichen Domanen Adaption, um eine Ubertragbarkeit des Modells auf Regionen mit unterschiedlichen geographischen Gegebenheiten zu erforschen. Außerdem konnte die Entwicklung von Verfahren spannend sein, welche die Probleme mit dicht bestuckten Windparks in den USA und der Nichtdetektion von sehr kleinen Anlagen adressieren. Da in dieser Arbeit das Modell Faster R-CNN mit einem ResNet-50 als Backbone verwendet wurde, ware es auerdem interessant zu erforschen inwieweit andere Modelle und kompaktere Netze mit weniger Aufwand ein ahnlich gutes Ergebnis erreichen.

# Literaturverzeichnis

- [1] I. Staffell and S. Pfenninger. Using bias-corrected reanalysis to simulate current and future wind power output. *Energy*, 114:1224 – 1239, 2016. ISSN 0360-5442. URL <http://www.sciencedirect.com/science/article/pii/S0360544216311811>. Zuletzt abgerufen 14.08.2020.
- [2] I. González-Aparicio, F. Monforti, P. Volker, A. Zucker, F. Careri, T. Huld, and J. Badger. Simulating european wind power generation applying statistical downscaling to reanalysis data. *Applied Energy*, 199:155 – 168, 2017. ISSN 0306-2619. URL <http://www.sciencedirect.com/science/article/pii/S0306261917304622>. Zuletzt abgerufen 22.08.2020.
- [3] R. Green, D. Pudjianto, I. Staffell, and G. Strbac. Market design for long-distance trade in renewable electricity. *The Energy Journal*, 37, 2016. URL <http://www.iaee.org/en/publications/ejarticle.aspx?id=2727>. Zuletzt abgerufen 14.08.2020.
- [4] J.-H. Piel, J. Hamann, A. Koukal, and M. H. Breitner. Promoting the system integration of renewable energies: Toward a decision support system for incentivizing spatially diversified deployment. *Journal of Management Information Systems*, 34 (4):994–1022, 2017. URL <https://doi.org/10.1080/07421222.2017.1394044>. Zuletzt abgerufen 14.08.2020.
- [5] M. de Simón-Martín, A. de la Puente-Gil, D. Borge-Diez, T. Ciria-Garcés, and A. González-Martínez. Wind energy planning for a sustainable transition to a decarbonized generation scenario based on the opportunity cost of the wind energy: Spanish Iberian Peninsula as case study. *Energy Procedia*, 157:1144 – 1163, 2019. ISSN 1876-6102. URL <http://www.sciencedirect.com/science/article/pii/S1876610218312554>. Zuletzt abgerufen 14.08.2020.
- [6] J.-H. Piel, C. Stetter, M. Heumann, M. Westbomke, and M. H. Breitner. Lifetime Extension, Repowering or Decommissioning? Decision Support for Operators of Ageing Wind Turbines. *Journal of Physics: Conference Series*, 1222:12–33, 2019. URL <https://iopscience.iop.org/article/10.1088/1742-6596/1222/1/012033/meta>. Zuletzt abgerufen 14.08.2020.
- [7] U.S. Department of the Interior. The U.S. Wind Turbine Database. URL <https://eerscmap.usgs.gov/uswtdb/>. Zuletzt abgerufen 12.07.2020.
- [8] Bundesnetzagentur für Elektrizität. Marktstammdatenregister. URL <https://>

- //www.marktstammdatenregister.de/MaStRHilfe/subpages/zieleKonzepte.html. Zuletzt abgerufen 22.08.2020.
- [9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks, 2012. URL <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>. Zuletzt abgerufen 21.07.2020.
  - [10] D. Kriesel. *Ein kleiner Überblick über Neuronale Netze*. 2007. URL [http://www.dkriesel.com/science/neural\\_networks](http://www.dkriesel.com/science/neural_networks). Zuletzt abgerufen 13.07.2020.
  - [11] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989. ISSN 0893-6080. URL <http://www.sciencedirect.com/science/article/pii/0893608089900208>. Zuletzt abgerufen 22.08.2020.
  - [12] N. Schaa. Neuronale Netze: Ein Blick in die Black Box. 2020. URL <https://www.informatik-aktuell.de/betrieb/kuenstliche-intelligenz/neuronale-netze-ein-blick-in-die-black-box.html>. Zuletzt abgerufen 22.08.2020.
  - [13] S. S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.
  - [14] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for Activation Functions, 2017. URL <https://arxiv.org/pdf/1710.05941.pdf>. Zuletzt abgerufen 12.07.2020.
  - [15] C. E. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning, 2018. URL <https://arxiv.org/pdf/1811.03378.pdf>. Zuletzt abgerufen 12.07.2020.
  - [16] R. Rojas. *Theorie der neuronalen Netze: Eine Systematische Einführung*. Springer-Verlag, 1993.
  - [17] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature* 521, pages 436–444, 2015. URL [https://www.researchgate.net/publication/277411157\\_Deep\\_Learning/link/55e0cdf908ae2fac471ccf0f/download](https://www.researchgate.net/publication/277411157_Deep_Learning/link/55e0cdf908ae2fac471ccf0f/download). Zuletzt abgerufen 14.07.2020.
  - [18] Y. LeCun, L. Bottou, G. Orr, and K.-R. Müller. Efficient BackProp. In *Neural Networks: Tricks of the Trade, 2nd edn., LNCS 7700*, pages 9–48. Springer, 1998.
  - [19] N. Buduma and N. Locascio. *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*. O'Reilly, 2017.
  - [20] S. Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL <http://arxiv.org/abs/1609.04747>. Zuletzt abgerufen 21.08.2020.
  - [21] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition, 2015. URL <https://arxiv.org/pdf/1512.03385.pdf>. Zuletzt abgerufen

15.07.2020.

- [22] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. URL <https://arxiv.org/abs/1502.03167>. Zuletzt abgerufen 01.08.2020.
- [23] Oxford Dictionary. Definition für Overfitting. URL <https://www.lexico.com/definition/overfitting>. Zuletzt abgerufen 11.08.2020.
- [24] C. Lin. 8 Simple Techniques to Prevent Overfitting. 2020. URL <https://towardsdatascience.com/8-simple-techniques-to-prevent-overfitting-4d443da2ef7d>. Zuletzt abgerufen 12.07.2020.
- [25] L. Prechelt. Early Stopping - But When? In *Neural Networks: Tricks of the Trade*, 2nd edn., LNCS 7700, pages 53–67. Springer, 1998.
- [26] Nicht bekannt. Wikipedia Eintrag Overfitting. URL [https://en.wikipedia.org/wiki/Overfitting#/media/File:Overfitting\\_svg.svg](https://en.wikipedia.org/wiki/Overfitting#/media/File:Overfitting_svg.svg). Zuletzt abgerufen 12.07.2020.
- [27] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology* 148.3, pages 574–591, 1959.
- [28] A. Hidaka and T. Kurita. Consecutive Dimensionality Reduction by Canonical Correlation Analysis for Visualization of Convolutional Neural Networks. *Proceedings of the ISCIE International Symposium on Stochastic Systems Theory and its Applications*, pages 160–167, 2017. doi: 10.5687/ssss.2017.160.
- [29] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. Improving Object Detection With One Line of Code, 2017. URL <https://arxiv.org/pdf/1704.04503.pdf>. Zuletzt abgerufen 07.08.2020.
- [30] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5), 2014. URL <https://arxiv.org/pdf/1311.2524.pdf>. Zuletzt abgerufen 17.07.2020.
- [31] J.R.R. Uijlings, K.E.A. van de Sande, Gevers T., and A.W.M. Smeulders. Selective Search for Object Recognition, 2012. URL <https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013/UijlingsIJCV2013.pdf>. Zuletzt abgerufen 04.08.2020.
- [32] R. Girshick. Fast R-CNN, 2015. URL <https://arxiv.org/pdf/1504.08083.pdf>. Zuletzt abgerufen 17.07.2020.
- [33] J. Redmon, Divvala S., R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection, 2016. URL <https://arxiv.org/pdf/1506.02640.pdf>. Zuletzt abgerufen 04.08.2020.
- [34] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Alexander C. Berg. SSD: Single Shot MultiBox Detector, 2016. URL <https://arxiv.org/>

- pdf/1512.02325.pdf. Zuletzt abgerufen 04.08.2020.
- [35] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. 2018. URL <https://arxiv.org/abs/1804.02767>. Zuletzt abgerufen 15.08.2020.
  - [36] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. 2020. URL <https://arxiv.org/abs/2004.10934>. Zuletzt abgerufen 15.08.2020.
  - [37] J. Hui. Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3), 2018. URL [https://medium.com/@jonathan\\_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359](https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359). Zuletzt abgerufen 15.08.2020.
  - [38] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, 2016. URL <https://arxiv.org/pdf/1506.01497.pdf>. Zuletzt abgerufen 17.07.2020.
  - [39] European Space Agency. Copernicus Open Access Hub, 2014. URL <https://scihub.copernicus.eu/>. Zuletzt abgerufen 22.07.2020.
  - [40] Developer Guide Google Maps. URL <https://developers.google.com/maps/documentation/maps-static/start#MapTypes>. Zuletzt abgerufen 30.07.2020.
  - [41] Wikipedia Eintrag zu Google Maps. URL [https://en.wikipedia.org/wiki/Google\\_Maps#:~:text=Google%20Maps'%20satellite%20view%20is,other%20imagery%20is%20from%20satellites](https://en.wikipedia.org/wiki/Google_Maps#:~:text=Google%20Maps'%20satellite%20view%20is,other%20imagery%20is%20from%20satellites). Zuletzt abgerufen 30.07.2020.
  - [42] Bundesamt für Kartographie und Geodäsie. Digitale Orthophotos. URL <https://gdz.bkg.bund.de/index.php/default/digitale-geodaten/digitale-orthophotos.html>. Zuletzt abgerufen 30.07.2020.
  - [43] Website GeoContent. URL <http://www.geocontent.de/index.php?id=5>. Zuletzt abgerufen 30.07.2020.
  - [44] Website Maxar Technologies. URL <https://www.maxar.com/>. Zuletzt abgerufen 30.07.2020.
  - [45] K. Yoshioka. frcnn-from-scratch-with-keras. URL <https://github.com/kentaro47/frcnn-from-scratch-with-keras>. Zuletzt abgerufen 30.07.2020.
  - [46] F. Chollet et al. Keras, 2015. URL <https://keras.io>. Zuletzt abgerufen 06.08.2020.
  - [47] B. Mandal, N. Sultana, and N. Puhan. Deep Residual Network with Regularized Fisher Framework for Detection of Melanoma. *IET Computer Vision*, 12, 2018. URL [https://www.researchgate.net/publication/326372957\\_Deep\\_Residual\\_Network\\_with\\_Regularized\\_Fisher\\_Framework\\_for\\_Detection\\_of\\_Melanoma](https://www.researchgate.net/publication/326372957_Deep_Residual_Network_with_Regularized_Fisher_Framework_for_Detection_of_Melanoma). Zuletzt abgerufen 01.08.2020.

- [48] O. Russakovsky, Deng J., H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.
- [49] F. Chollet et al. Keras Deep Learning Models, 2017. URL <https://github.com/fchollet/deep-learning-models/releases>. Zuletzt abgerufen 07.08.2020.
- [50] R. J. Tan. Breaking Down Mean Average Precision (mAP), 2019. URL <https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52>. Zuletzt abgerufen 06.08.2020.
- [51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average\\_precision\\_score.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html). Zuletzt abgerufen 11.08.2020.



# A. Hyperparameter und sonstige Werte für das Training

**Optimizer:** Adam      **Lernrate:** 0,001      **Clipnorm:** 0,001

**RPN Stride:** 16

**RoI Pooling Layer Output Größe:** 5 x 5

## Anchor Boxen

Experiment	Größen	Seitenverhältnisse
4.2.2	30, 50	(1,1)
4.2.3	30, 50	(1,1)
4.2.4	40, 58, 70	(1,1)
4.2.5	40, 58, 70	(1,1)

**Bildgröße:** 800 x 800

## Z-Score Normalisierung der RGB-Kanäle

	Deutschland		USA	
Kanal	Durchschnitt	Standardabweichung	Durchschnitt	Standardabweichung
R	85,5	31,77	95,8	26,96
G	90,84	25,56	93,8	23,64
B	71,95	23,68	79,81	23,52