Ball-, Torpfosten- und Roboter-Erkennung im RoboCup durch FCNNs

Maximilian Birkenhagen, Robert Geislinger

Universität Hamburg, Fachbereich Informatik
Vogt-Koelln-Strasse 30, 22527 Hamburg
{6birkenh,6geislin}@informatik.uni-hamburg.de

Zusammenfassung. In dieser Arbeit werden FCNNs für die Erkennung von Bällen, Torpfosten und Robotern im RoboCup genutzt und auftretende Probleme erläutert. Dafür wird die Funktionsweise von FCNNs und mögliche Parameter zur Optimierung dieser erklärt. Bei den durchgeführten Experimenten werden zwei Netzarchitekturen vorgestellt, eine bereits für Ballerkennung erprobte[4] und eine davon inspirierte kleinere Version, und voneinander abgewogen. Weiterhin werden die Optimizer Adam, SGD und Adagrad verglichen und die Problematik von zu geringen Datensätzen im maschinellem Lernen aufgezeigt. Die Erkennung von Bällen und Torpfosten wird erfolgreich durchgeführt, wohingegen die vorgestellten Möglichkeiten zur Erkennung von Robotern noch nicht praxistauglich ist.

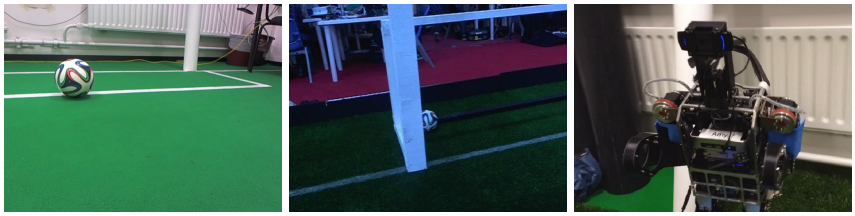
Schlüsselwörter: RoboCup, FCNN, Objekterkennung, Keras, Ball, Torpfosten, Roboter, Heatmap

1 Einleitung *(Maximilian & Robert)*

Die Hamburg Bit-Bits sind ein studentisches Team der Universität Hamburg, welches sich mit Fußball spielenden Robotern im Rahmen des RoboCups beschäftigt [1] [2]. Der RoboCup ist eine Liga in der Fußballspielende Roboter gegeneinander antreten. Das langfristige Ziel für das Jahr 2050 ist ein Sieg gegen den amtierenden Fussballweltmeister der Männer. Die Mitglieder beschäftigen sich mit der Optimierung der Roboter, hinsichtlich des Erfolgs im Fußball spielen, um mit ihrer Arbeit einen Beitrag zur Wissenschaft leisten zu können. Diese Ausarbeitung ist entstanden im Rahmen des Projektes *Projekt RoboCup - Mit humanoiden Robotern Fußball spielen* und nutzt ein Multi-object Fully Convolutional Neural Network (FCNN) für die Erkennung von Fußbällen. Dieses Netz wird im Laufe des Projekts auf die Erkennung von Torpfosten und Robotern trainiert. Beispiele für diese Objekte und deren möglichen Hintergrund sind in Abbildung

[1] Hamburger Bit-Bots. https://robocup.informatik.uni-hamburg.de abgerufen am 16.12.2018
[2] RoboCup https://www.robocup.org abgerufen am 16.12.2018

1 zu sehen. Aufgrund guter Ergebnisse in anderen Arbeiten basieren die Netze in dieser Arbeit auf FCNNs. In dieser Arbeit werden verschiedene Möglichkeiten der Optimierung und Anpassung evaluiert und die Ergebnisse miteinander verglichen. Ziel dieser Arbeit ist eine Verbesserung der schon vorhandenen neuronalen Netzarchitekturen im Bezug auf Erkennungsrate und Netzgröße sowie die Auswirkung anpassbarer Parameter. Im Folgenden wird der Ansatz sowie das Basis-FCNN näher erläutert, auf dessen Grundlage die Vergleiche evaluiert werden und welche Änderungen positive und welche Änderungen negative Effekte bewirken.



(a) Objekt: Ball (b) Objekt: Torpfosten (c) Objekt: Roboter

Abb. 1. Beispiele für zu erkennende Objekte unserer FCNNs [3]

2 Ansatz *(Maximilian & Robert)*

Fully **C**onvolutional **N**eural **N**etworks [2] ermöglichen das Erkennen von Objekten, indem die sie Muster und Regeln überwacht erlernen. Überwachtes Lernen *(eng: supervised learning)* beschreibt das Trainieren von neuronalen Netzen, bei welchem das zu erreichende Ergebnis bereits bekannt ist. Somit können die Berechnungen des neuronalen Netzes zielgerichtet angepasst werden um das gewünschte Ergebnis zu erreichen. Im Vergleich zum unüberwachten Lernen *(eng: unsupervised learning)*, wo das zu erreichende Endergebnis noch nicht bekannt ist, werden große Mengen an aufbereiteten Daten benötigt. Das FCNN wird mit Hilfe eines Trainingsdatensatzes trainiert, welcher Bilder der zu erkennenden Objekte enthält. Nach [3] bestehen solche neuronalen Netze aus vielen einfachen verbundenen Knoten, welche Neuronen genannt werden. Das Konzept künstlicher Neuronen ist ein Modell, was sich an den Neuronen des menschlichen Gehirns anlehnt. Eine Gruppierung solcher Neuronen zu einer Schicht wird dabei als Layer bezeichnet; alle Layer zusammen betrachtet bilden das Netz.

Die Funktionsweise eines Convolutional-Layers, welches einen der Grundbausteine eines FCNNs darstellt, ist schematisch in Abbildung 2 dargestellt. Hierbei wird ein Kernel über das Eingabebild geschoben, der für jeden Matrixwert des Bildes einen Ausgabewert errechnet, welche zusammen eine neue Matrix bilden.

[3] ImageTagger `https://imagetagger.bit-bots.de` abgerufen am 8.12.2018

Die Berechnung geschieht durch einen Filter im Inneren des Kernels, der durch das FCNN selbstständig erlernt wird.

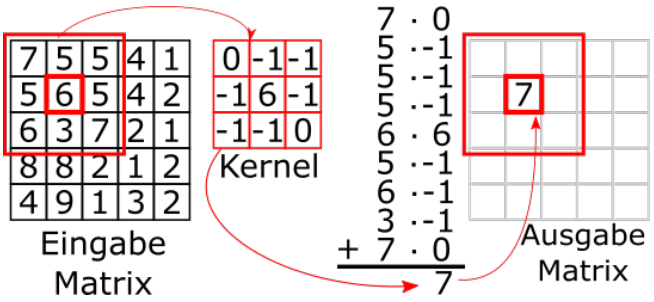


Abb. 2. Funktionsweise eines Convolution-Layers mit unvollständiger Ausgabematrix.

Um einen Vergleich der verschiedenen Parameter und ihrer Auswirkungen auf FCNNs zu evaluieren, wird in der Arbeit auf ein bereits bestehendes Netz zurückgegriffen. Für die Struktur des Basis-Netzes wurde sich an einem Modell orientiert, dass bereits bei den Hamburg Bit-Bots zum Einsatz kommt [4]. Genutzt wird es zur Erkennung von Bällen während des Spiels. Die Netzstruktur ist in Abbildung 3 schematisch dargestellt. Die Convolutional-Layer sorgen

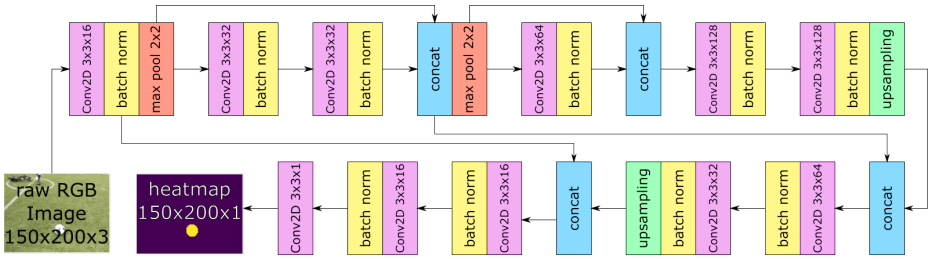


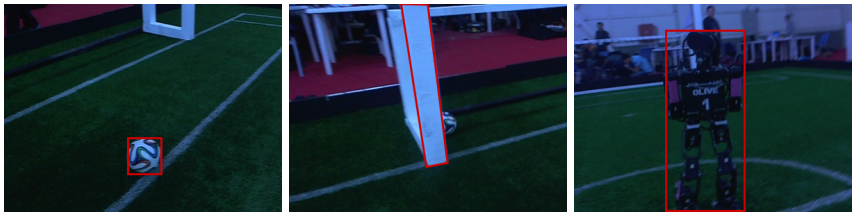
Abb. 3. Struktur des Basisnetz FCNNs basierend auf *Towards Real-Time Ball Localization using cnns* [4]. Conv2D(lila), batch normalization(gelb), max pooling(rot) und concat-layer(blau).

für die eigentliche Erkennung von Mustern und Merkmalen des Inputs. Folgend auf ein Convolutional-Layer ist stets ein BatchNormalization-Layer in der Netzstruktur, um die Matrixwerte zwischen den Layern zu normalisieren. Bei den hier auftretenden MaxPooling-Layern handelt es sich um DownSampling-Layer, welche unter Nutzung eines 2x2-Filters nur den größten Wert der Matrix in eine Matrix kleinere Dimension umrechnet. Hierdurch werden nur wichtige Informationen aus der Eingabematrix herausgefiltert und folgende Berechnungen können durch die verkleinerte Matrix effizienter durchgeführt werden ohne essentielle In-

formationen zu verlieren. Um später die Ursprungsdimension wiederherzustellen werden Upsampling-Layer genutzt, indem sie einzelne Werte der Eingabematrix hochskalieren. Häufig eingesetzt werden ebenfalls Concat-Layer, welche Querverbindungen zwischen LowLevel-Merkmalen der ersten Convolutional-Layer und den HighLevel-Merkmalen der späteren Layer herstellen. Der verwendete Optimizer ist Adam [1] (*eng:Adaptive Moment Estimation*).

Als Ausgabe des Netzes ensteht eine Wahrscheinlichkeitsmatrix mit Werten zwischen 0 und 1. Zeichnet man diese Matrix als Graph, so kann diese als Heatmap interpretiert werden, in der das gesuchte Objekt zu finden ist. Unter einer Heatmap versteht man ein Bild, auf der jener Bereich markiert ist, in welchem die Wahrscheinlichkeit, dass das gesuchte Objekt sich dort befindet, besonders hoch ist. Der Schwellwert θ bestimmt, ob ein Wert der Matrix das gesuchte Objekt abbildet. Dieser Wert wurde nach Beobachtung von Werten zwischen 0,3 und 1 auf 0,7 festgelegt. Das räumt genügend Freiraum für kleinere Fehler des Netzes ein. Die Netzarchitekturen sind in Python mit Hilfe von Keras, einer high-level API für die Tensorflow Engine [4] implementiert. Tensorflow ist eine freie Engine zur Entwicklung von neuronalen Netzen.

Als Quelle für die Datensätze wird die Plattform ImageTagger der Hamburg Bit-Bots verwendet. Hier wurden Daten für Netze im Zusammenhang der Objekterkennung im RoboCup mit Labels versehen. Die Objekte wurden dazu mit Bounding Boxen (Hüllkörper) in rechteckiger Form markiert. Sie kennzeichnen den genauen Standort des zu erkennenden Objektes auf dem Bild. Die Labels wurden manuell für Bälle, Torpfosten, Roboter und andere Objekte gesetzt. Beispiele für die in dieser Arbeit wichtigen Labels sind in Abbildung 4 zu sehen. Für die Verarbeitung und Aufbereitung der Labels wurde der MLHelper [5] genutzt, der im Rahmen dieser Arbeit um Funktionen zur Nutzung von mehreren Labels auf einem Bild erweitert wurde. Mit Hilfe einer Elipsenberechnung wird aus dem Rechteck für die Labels der Bälle ein Kreis berechnet. Somit können die Bälle besser umschlossen werden und das Netz wird störungsfreier ohne die Ballumgebung trainiert. Bei Robotern kann es ein Problem sein, dass sie je



(a) Objekt: Ball (b) Objekt: Torpfosten (c) Objekt: Roboter

Abb. 4. Beispiele für Bounding Boxen (rot) um die Objekte aus dem Imagetagger

[4] Keras `https://keras.io` abgerufen am 1.3.2019
[5] MLHelper GitHub Repository `https://github.com/Daniel451/MLHelper`

nach Modell komplexe geometrische Formen als Außenmaß besitzen. Eine einfache rechteckige Box könnte zu viel Hintergrund der Roboter in die Erkennung einfließen lassen und diese potenziell verschlechtern. Der Datensatz für Bälle [6] verfügt über 22927 Bilder fürs Training und 2177 zum Testen. Der Torpfosten Datensatz verfügt über 7137 und 2538 Bilder, der Roboter Datensatz beinhaltet 992 und 108 Bilder. Für das Training der FCNNs wird der Datensatz dabei pro Epoche einmal durchlaufen. Der Testdatensatz wird nach dem Training einmal durchlaufen und dient der Überprüfung des Lernfortschritts und dem aggregieren der Resultate. Diese Unterteilung ist notwendig, um klare und konkrete Schlüsse ziehen zu können.

Die Bilder selbst werden auf die Dimension 200x150x3 (Breite x Höhe x RGB) herunter skaliert, da nach [4] so auf den Robotern für die Berechnung ein gutes Mittelmaß zwischen Rechenzeit und Erkennungsrate gewährleistet ist.

3 Untersuchbare Parameter *(Maximilian & Robert)*

Der **Optimizer** passt die Gewichte innerhalb des Netzes und die Learning Rate an. Dabei spielt vor allem die Loss-Funktion eine große Rolle, die ein mathematisches Maß für die Güte des Outputs ist.

Die **Aktivierungsfunktion** ist eine Zuordnung, die dem Netzinput jeweils die Aktivität des Netzes zuweist. Je nach Wahl der Funktion wird die Aktivität der Neuronen anders gewichtet. Wichtig im Bereich von FCNNs ist hier, dass man keine lineare Aktivierungsfunktion verwenden sollte, da Convolutional-Layer selbst nur lineare Berechnungen durchführen. Würde man hier eine lineare Funktion zur Aktivierung nutzen, so hätte es denselben Effekt, als würde man nur ein Convolutional-Layer nutzen.

Die **Learning Rate** bestimmt, wie stark die Gewichte der Neuronen in jeder Iteration an den Gradienten der Optimizer angepasst werden. Während eine zu hohe Learning Rate dazu führen kann, dass das gesuchte Minimum der Loss-Funktion nicht erreicht wird, neigt eine zu geringe Learning Rate, dass eher lokale Minima erreicht werden und das globale Minimum nicht erreicht wird.

Durch **Data Augmentation** können kleine Datensätze mit Hilfe von Verfahren wie Spiegelung, Translation oder durch Rauschfilter künstlich erweitert werden, ohne den Datensatz mit neuen Bildern anzureichern. Dieses Vorgehen ist besonders interessant für überwachtes maschinelles Lernen, da hier eine große Menge an Daten benötigt wird, um gute Resultate zu erzielen.

Durch **Änderungen der Netzstruktur** wie dem Verkleinern des Basisnetzes,Entfernen von Layern oder ganz anderen Netzarchitekturen ermöglichen sich vielerlei Möglichkeiten. Umso mehr die Netze verkleinert werden können, desto besser können sie in Echtzeit auf den Robotern ausgeführt werden und Ergebnisse liefern. Sind die Netze jedoch zu klein, könnten die Ergebnisse sich deutlich verschlechtern, da nicht die volle Komplexität des untersuchten Bildraumes widergespiegelt werden kann.

[6] Hamburg Bit-Bots Ball Dataset 2018 `https://robocup.informatik.uni-hamburg.de/en/documents/bit-bots-ball-dataset-2018/`

Bei größeren Netzstrukturen könnte ein eventuelles Overfitting durch Einführen von **Dropout** minimiert werden. Overfitting bezeichnet hierbei das Problem, dass ein neuronales Netz den Trainingsdatensatz auswendig lernt, statt Muster in den Daten zu erkennen. Dadurch erreicht man nur sehr gute Ergebnisse auf den Trainingsdaten, aber auf neuen Daten deutlich schlechtere. Der Dropout sorgt dafür, dass zufällig Neuronen bei der Erkennung nicht in Betracht gezogen werden. Somit werden bestimmten Neuronen nicht bestimmte Bilder während des Trainings zugeordnet, da diese vom Dropout betroffen sein könnten.

4 Ergebnisse *(Maximilian & Robert)*

In diesem Abschnitt werden die Ergebnisse der Experimente anhand der Genauigkeit, der Exaktheit, dem Recall, dem Jaccard-Index sowie der Erkennungsdauer für den Testdatensatz visualisiert und verglichen. Um die Formeln abzukürzen stehen die jeweiligen Abkürzungen für TP: *True Positive*, FP: *False Positive*, TN: *True Negative* und FN: *False Negative*.

Die Präfixe stehen dabei für den Wahrheitsgehalt der Aussage, während der Suffix für *Positive* für ObjektPixel und *Negative* für nicht-ObjektPixel steht.

- **Genauigkeit** *(eng: accuracy)*: Beschreibt das Verhältnis zwischen richtig erkannten Objekten sowie korrekt als nicht-Objekt erkannten Bereichen geteilt durch alle Ergebnisse.

$$\frac{TP + TN}{TP + FP + FN + TN} = Genauigkeit \tag{1}$$

- **Exaktheit** *(eng: precision)*: Gibt an, wie viele als richtig erkannte Labels auch tatsächlich richtig sind. Ist der Wert hoch sind alle gelabelten Bereiche mit hoher Wahrscheinlichkeit auch tatsächlich als Objekt erkannt.

$$\frac{TP}{TP + FP} = Exaktheit \tag{2}$$

- **Recall**: Spiegelt die Sensitivität der Ergebnisse wieder und gibt an, wie viele aller zu erkennenden Bereiche das Netz tatsächlich auch erkannt hat.

$$\frac{TP}{TP + FN} = Recall \tag{3}$$

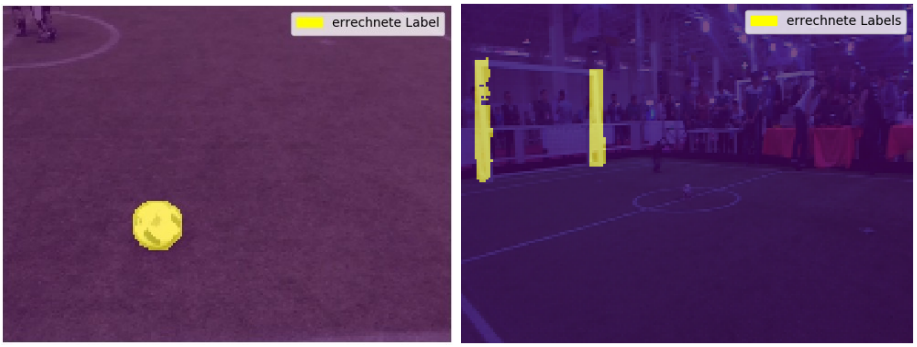
- **Jaccard-Index**: Beschreibt, wie genau die berechneten Bounding Boxen an den richtigen Bounding Boxen liegen. Je mehr die berechnete Bounding Box mit der richtigen Bounding Box überlappt, desto höher ist der Jaccard-Index. Dieser Wert ist vor allem deshalb interessant, da ein hoher Wert genau das Verhalten widerspiegelt, welches von den Netzen als Resultat wünschenswert ist.

$$\frac{TP}{TP + FP + FN} = Jaccard\text{-}Index \tag{4}$$

Die Werte der im Abschnitt 4 aufgeführten Ergebnisse sind auf die erste Nachkommastelle gerundet. Bei der Anzahl der Trainingsepochen für die verschiedenen Experimente wurde unabhängig voneinander trainiert. Demnach wurden für das Training nach 25 Epochen nicht der vorherige Stand des 10 Epochen Trainings weiter verwendet, sondern neu trainiert. Es gab insgesamt vier verschiedene Experimente:

4.1 Basisnetz

Das Basisnetz liefert wie zuvor in Abschnitt 2 beschrieben eine Grundlage, um Vergleiche mit anderen untersuchten Parametern ziehen zu können. Ein errechnetes Ball Label ist beispielhaft in Abbildung 5 (a) zu sehen. Die zugehörigen Ergebnisse sind der Tabelle 1 zu entnehmen.



(a) Objekt: Ball (b) Objekt: Torpfosten

Abb. 5. Beispiele für die errechneten Label vom Basisnetz nach 25 Epochen Training. Die gelben Flächen zeigen die als (a) Ball oder (b)Torpfosten erkannten Bereiche.

4.2 Smallnetz

Dieses Netz, dessen Struktur in Abbildung 6 dargestellt ist, ist eine verkleinerte Version der Basisnetzstruktur aus Abschnitt 4.1. Die Hauptunterschiede zum Basisnetz sind die deutlich verringerte Anzahl der Layer und das Fehlen der Concat-Layer. Das Ziel dieses Netzes ist eine Senkung der Erkennungsdauer bei gleichbleibenden Resultaten. Die Ergebnisse dieses Netzes befinden sich in Tabelle 1.

4.3 Optimizer

In dieser Arbeit werden außerdem drei verschiedene Optimizer miteinander verglichen. Zu überprüfen ist hier, ob das Training bei gleicher Trainingsepochenanzahl zum gleichen Resultat führt. Für jeden der hier getesteten Optimizer, SGD

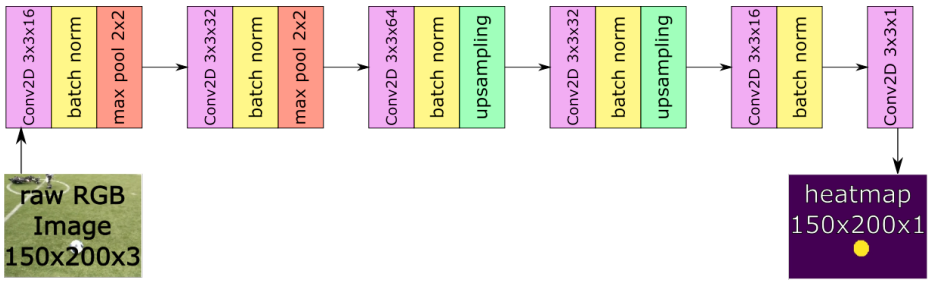


Abb. 6. Struktur des Smallnetz FCNNs. Conv2D(lila), batch normalization(gelb), und max pooling(rot).

(*eng:stochastic gradient descent*), Adagrad und Adam, werden die Standardparameter genutzt. Die Ergebnisse dieses Experiments auf dem Basisnetz sind in Tabelle 2 dargelegt.

4.4 Torpfosten und Roboter

Ebenfalls wird das Basisnetz mit den anderen beiden Datensätzen für Torpfosten und Roboter auf gute Erkennung getestet. Hiermit soll überprüft werden, ob die schon verfügbare und erprobte Netzstruktur ohne Anpassung für diese Anwendungsbereiche nutzbar ist. Ein wesentlicher Unterschied zur Ballerkennung liegt in der Möglichkeit, dass pro Bild mehrere Label existieren und somit mehrere zu erkennende Objekte sich auf einem Bild befinden können. Ein Torpfosten Label ist beispielhaft in Abbildung 5 (b) zu sehen. Die Ergebnisse für dieses Experiment sind in 3 einsehbar.

	Basisnetz-Ball			Smallnetz-Ball		
Anzahl der Epochen	10	25	50	10	25	50
Genauigkeit, ∅	99,8%	99,8%	99,8%	99,8%	99,9%	99,8%
Genauigkeit, 90.-Perzentil	99,9%	100%	100%	100%	100%	100%
Genauigkeit, 99.-Perzentil	99,9%	100%	100%	100%	100%	100%
Exaktheit, ∅	86,7%	86,3%	85,8%	88,0%	91,1%	89,7%
Exaktheit, 90.-Perzentil	100%	100%	100%	100%	100%	100%
Exaktheit, 99.-Perzentil	100%	98,8%	100%	100%	98,8%	100%
Recall, ∅	81,5%	78,4%	73,8%	65,7%	76,3%	73,4%
Recall, 90.-Perzentil	100%	100%	100%	92,5%	98,6%	95,7%
Recall, 99.-Perzentil	100%	100%	100%	100%	100%	100%
Jaccard-Index, ∅	75.4%	71,2%	69,7%	64,4%	72,7%	70,2%
Jaccard-Index, 90.-Perzentil	94,1%	93,5%	93,6%	88,3%	91,4%	89,4%
Jaccard-Index, 99.-Perzentil	98,6%	99,1%	100%	95,7%	97,4%	95,8%
Erkennungsdauer	1563s	1562s	1586s	1389s	1464s	1448s

Tabelle 1. Ergebnisse des Basisnetz und Smallnetz für den Ball-Datensatz mit 10, 25 und 50 Epochen Training.

5 Diskussion *(Maximilian & Robert)*

Die Ergebnisse des Basisnetzes aus Tabelle 1 sind wie erwartet vielversprechend. Bereits bei zehn Trainingsepochen sind sowohl die Werte der Genauigkeit, Exaktheit als auch des Recalls im 99.-Perzentil nahezu 100%. Der vermeintlich niedrige Jaccard-Index bei der Betrachtung des Durchschnitts täuscht. Einige wenige Bilder des Datensatzes werden hier mit komplett falschen Labeln versehen, was den Durchschnitt stark verschlechtert, da ein Ball-Label nur einen

Basisnetz-Ball nach 10 Epochen			
Optimizer	Adam	SGD	Adagrad
Genauigkeit, ∅	99,8%	99,8%	99,9%
Genauigkeit, 90.-Perzentil	99,9%	99,9%	100%
Genauigkeit, 99.-Perzentil	99,9%	100%	100%
Exaktheit, ∅	86,7%	86,4%	87,5%
Exaktheit, 90.-Perzentil	100%	100%	100%
Exaktheit, 99.-Perzentil	100%	100%	100%
Recall, ∅	81,5%	79,5%	72,5%
Recall, 90.-Perzentil	100%	100%	97,9%
Recall, 99.-Perzentil	100%	100%	98%
Jaccard-Index, ∅	75,4%	73,4%	70,0%
Jaccard-Index, 90.-Perzentil	94,1%	93,1%	91,8%
Jaccard-Index, 99.-Perzentil	98,6%	98,6%	98,0%

Tabelle 2. Ergebnisse des Basisnetz-Ball nach 10 Epochen mit den Optimizern Adam, SGD und Adagrad.

Basisnetz							
Objekt	Ball		Torpfosten		Roboter		
Anzahl Trainings-Epochen	10	25	10	25	10	10 Data Augmentation	25
Genauigkeit, ∅	99,8%	99,8%	99,8%	99,3%	72,1%	95,9%	97,5%
Genauigkeit, 90.-Perzentil	99,9%	100%	99,9%	99,9%	78,2%	97,2%	97,9%
Genauigkeit, 99.-Perzentil	99,9%	100%	100%	99,9%	93,4%	97,3%	98,1%
Exaktheit, ∅	86,7%	86,3%	86,4%	96,5%	4,9%	33,3%	0%
Exaktheit, 90.-Perzentil	100%	100%	100%	100%	6,5%	10,9%	0%
Exaktheit, 99.-Perzentil	100%	98,8%	100%	100%	13,7%	27,9%	0%
Recall, ∅	81,5%	78,4%	79,5%	81,9%	49,1%	1,1%	0%
Recall, 90.-Perzentil	100%	100%	100%	97,2%	60,9%	3,7%	0%
Recall, 99.-Perzentil	100%	100%	100%	99,6%	68,9%	5,8%	0%
Jaccard-Index, ∅	75,4%	71,2%	73,4%	79,8%	46,1%	0,8%	0%
Jaccard-Index, 90.-Perzentil	94,1%	93,5%	93,1%	95,2%	63,2%	2,4%	0%
Jaccard-Index, 99.-Perzentil	98,6%	99,1%	98,6%	99,6%	11,9%	4,7%	0%

Tabelle 3. Ergebnisse des Basisnetz nach 10 Epochen für Ball, Torpfosten und Roboter.

sehr kleinen Bereich des Bildes ausmacht. Durch Betrachtung des 99.-Perzentils werden diese wenigen Ausnahmen unterdrückt.

Die Ergebnisse des Smallnetz aus Tabelle 1 sind nicht eindeutig einzuordnen. Durch die deutlich kleinere Netzstruktur als auch die fehlenden Querverweise durch Concat-Layer wurde eine Senkung der Erkennungsdauer erwartet. Zwar sind diese Werte gesunken, aber nur um ca. 200 Sekunden, während die gesamte Netzstruktur auf die Hälfte verkleinert wurde. Ein möglicher Grund hierfür könnte sein, dass alleine das Verarbeiten der Bilder sehr viel Zeit in Anspruch nimmt, sodass große Änderungen an der Anzahl der Layer und die Concat-Layer eher geringe Auswirkungen auf die eigentliche Laufzeit haben. Diese Verkleinerung führt jedoch zu einer deutlichen Senkung der Erkennungsrate im Jaccard-Index. Im Basisnetz betrug dieser Wert bei 50 Epochen im 99.-Perzentil 100%. Im Smallnetz hingegen wurde durch Training maximal 97,4% im 99.-Perzentil erreicht. Die Verschlechterung im 99.-Perzentil ist bei der geringen Verbesserung in der Erkennungsdauer schlechter als erhofft.

Der Einfluss der verschiedenen Optimizer ist in Tabelle 2 zu sehen. Generell sind hier alle Ergebnisse nur sehr wenig voneinander abweichend. SGD ändert nur die Gewichte des Netzes abhängig von der Loss-Funktion. Bei Adam und Adagrad handelt es sich um adaptive Optimizer, die ebenfalls die Learning Rate aufgrund des vorherigen Trainings verändern. Lediglich der durchschnittliche Recall-Wert bei Nutzung von Adagrad ist gegenüber den anderen beiden Optimizern um ca. 9% schlechter ausgefallen. Für die praktische Anwendung an diesem Ball Datensatz gab es somit keine wesentlichen Unterschiede.

Die Erkennung der Torpfosten funktioniert mit dem Basisnetz bereits sehr gut, wie in der Tabelle 3 einzusehen ist. Der Jaccard-Index im 99.-Perzentil nach 25 Epochen Training beträgt hier 98,2%, womit diese nur um ca. 1% geringer sind, als der Wert der Erkennung für Bälle bei gleichlangem Training. Im Gegensatz dazu sind die Recall-Werte im Durchschnitt mit 65,5% bei 10 Epochen für Torpfosten um 15% schlechter als der Wert von 81,5% bei den Bällen. Hier wird deutlich, dass bei einigen untersuchten Bildern nicht alle vorhandenen Torpfosten gefunden werden. Bei 25 Epochen Training liegt dieser Wert mit 77,1% nahe an dem für Bälle von 78,4% und ist somit als gut einzustufen. Mit höherer Epochenanzahl für das Training des Netzes kann die Basisnetzstruktur somit auch für eine zufriedenstellende Erkennung von Torpfosten verwendet werden.

Die Ergebnisse des Basisnetz für die Roboter sind in Tabelle 3 zu sehen. Die Resultate sind hier deutlich schlechter als für die Torpfosten- oder Ballerkennung. Mit nur ca. 12% für den Jaccard-Index im 99.-Perzentil bei 10 Epochen Training ist eine sichere Robotererkennung nicht möglich. Dies liegt vor allem an dem sehr kleinen Datensatz. Mit seinen nur 992 Bildern für Training und 108 Bilder für die Evaluierung, ist eine gute Erkennung nicht gewährleistet. Das Ergebnis nach 25 Epochen verdeutlicht die Aussage: Exaktheit, Recall als auch der Jaccard-Index sind für den Durchschnitt als auch die betrachteten Perzentile auf 0% gesunken. Ein Überprüfen durch Testen des Netzes auf dem Trainingsdatensatz zeigt, dass hier das Netz nur noch die vorhandenen Daten auswendig lernt und ein Overfitting stattfindet. Die Werte steigen hier deutlich in den 90% Bereich für die

Perzentile an. Die Nutzung von Data Augmentation Techniken sorgt hier auch für keine Verbesserung des Resultats, sondern verschlechtern den Trainingserfolg. Nach 10 Epochen Training auf dem augmentierten Datensatz erreicht das Netz für den Jaccard-Index im 99.-Perzentil nur noch 4,7% statt der vorher erreichten 12%.

6 Fazit *(Maximilian & Robert)*

Das Basisnetz liefert für Bälle sehr gute Ergebnisse. Dieselbe Netzstruktur ist zudem nutzbar für die Erkennung von Torpfosten, ohne dass eine weitere Anpassung nötig ist. Für die Erkennung von Robotern kann leider keine genaue Aussage getroffen werden, was die Verwendbarkeit der Basisnetzstruktur betrifft. Hierfür fehlt ein ausreichend großer Datensatz um das Overfitting des Netzes zu vermeiden. Von einer künstlichen Erweiterung der Datensätze durch Data Augmentation ist abzuraten. Dies führte zu einem deutlichen Verschlechtern der Ergebnisse. Die in dieser Arbeit ausprobierte Smallnetzstruktur hat leider nicht den gewünschten Erfolg erbracht, die Basisnetzstruktur unter gleichbleibendem Ergebnis zu verkleinern. Ein Ergebnis von 97,4% im Jaccard-Index des 99.-Perzentils für die Ballerkennung ist aber im Bezug auf die gesunkene Erkennungszeit trotzdem gut. Die Auswahl an verschiedenen Optimizern hat keine großen Unterschiede der Ergebnisse erbracht.

7 Aussicht *(Maximilian & Robert)*

Trotz der hier durchgeführten Experimente sind noch viele der Parameter aus Abschnitt 3 im Rahmen dieser Arbeit nicht untersucht worden. So könnten andere Aktivierungsfunktionen bei gleicher Netzstruktur große Auswirkungen auf die Ergebnisse haben. Ebenfalls könnten andere Learning Rates genutzt werden, um schneller das gewünschte Minimum der Loss-Funktion zu erreichen. Die Bestimmung der Learning Rate ist jedoch wieder vom verwendeten Optimizer abhängig. Während Adam oder Adagrad nur durch anfängliche Learning Rates parametrisiert werden, da sie im Laufe des Trainings automatisch anpassen, kann man für SGD ebenfalls über eine durch den Erfolg während des Trainings flexible Anpassung der Learning Rate nachdenken. Für die Optimizer direkt konnten keine konkreten Schlussfolgerungen über die Unterschiede in ihrer Nutzung gefunden werden. Eine Senkung der Erkennungsdauer durch eine andere Netzstruktur bei gleichbleibenden Ergebnissen wurde ebenfalls nicht erreicht, ist aber durch weiteres experimentieren grundsätzlich möglich. Obwohl die hier vorgestellten Ergebnisse für Bälle und Torpfosten sehr gut sind, wurden die Netze noch nicht praxisnah auf den Robotern ausgeführt.

Die Erkennung von Robotern auf dem Spielfeld ist noch nicht praktikabel. Hier müsste ein wesentlich größerer Datensatz genutzt werden, damit überhaupt überprüft werden kann, ob die vorgestellten Netzarchitekturen auch für diesen Anwendungsfall ausreichen oder mehr Experimentieren erforderlich ist. Generell sind auch die wenigen schon vorhandenen Labels auf dem Roboterdatensatz

noch nicht genügend aufbereitet. Die Roboter besitzen im Vergleich zu Bällen und Torpfosten komplexe geometrische Formen, welche nicht durch einfache Anpassungen wie einer Elipsenberechnung abgeändert werden können. Die manuelle Eingabe der Bounding Boxen im ImageTagger sind nicht genau oder nur mit unverhältnismäßig hohem Aufwand genau um die Roboter festzulegen. Eine Möglichkeit zur Optimierung der Bounding Boxen wäre die Nutzung des Scanline Verfahrens [5]. Man kann davon ausgehen das Roboter sich vom Hintergrund abheben und so eine Anpassung der Boxen möglich ist. Das Label wird also zeilenweise bearbeitet um so ein genauen Umriss des Roboters zu erhalten. Wünschenswert wäre auch eine Möglichkeit die gegnerischen Roboter von den eigenen Mitspielern zu unterscheiden.

Im Anwendungsbereich des RoboCups gibt es neben der Ball-, Torpfosten- und Robotererkennung andere Objekte, deren Erkennung nützlich für die Evaluation der Spielstrategie sein können. Elemente wie das Spielfeld oder der genaue Verlauf der Spielfeldlinien könnten auf mögliche Erkennung untersucht werden. Da es im RoboCup auch zu stetigen Regeländerungen im Bereich der Optik und zugelassenen Farben der Objekte kommen kann, können bereits erreichte Erfolge in Zukunft auch funktionslos werden und müssen somit an die dann aktuellen Regelwerke angepasst werden.

Paper

[1] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[2] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[3] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[4] Daniel Speck, Marc Bestmann, and Pablo Barros. Towards real-time ball localization using cnns. *Robot World Cup XXII. Springer*, 2018.

[5] Chris Wylie, Gordon Romney, David Evans, and Alan Erdahl. Half-tone perspective drawings by computer. In *Proceedings of the November 14-16, 1967, Fall Joint Computer Conference*, AFIPS '67 (Fall), pages 49–58, New York, NY, USA, 1967. ACM.