Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

Visuelle Echtzeit Detektion humanoider Roboter im RoboCup Kontext

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)

eingereicht von: Denis Ring
geboren am: 14.05.1990
geboren in: St. Petersburg/Russia

Gutachter/innen: Prof. Dr. Verena Hafner
Prof. Dr. Hans-Dieter Burkhard

eingereicht am: ..... verteidigt am: .....

Visuelle Objekterkennung spielt eine zentrale Rolle für die Autonomie eines Roboters. Erkennung der Mitspieler (anderer Roboter) im Kontext von RoboCup stellt ein relevantes Modellproblem dar. Die Roboter haben im Bild eine komplexe Erscheinung und wechseln ihre Form abhängig von der Bewegung des Roboters, von dem Blickwinkel, der Beleuchtung und dem Trikot des Roboters. In dieser Arbeit werden zwei vielversprechende Ansätze basierend auf Faltungsnetzwerken (CNN) aus der Literatur untersucht: SPQR-net und auf SqueezeNet basierendes verkleinertes Netzwerk miniSqueezeNet4. Beide Netze werden mit Hilfe der Keras Bibliothek mit Tensorflow Backend in Python implementiert und empirisch getestet. Die empirische Untersuchung erfolgt auf drei Bilddatensätzen: einem eigens dafür erstellten Datensatz, dem Datensatz von SPQR und dem Datensatz aus der Diplomarbeit von Dominik Krienelke.

Inhaltsverzeichnis

1 Einleitung

In dieser Bachelorarbeit wird die Erkennung von Nao Robotern auf dem RoboCup Feld mit Hilfe von CNN behandelt. Es werden die CNN aus [5] und [8] implementiert und auf den Datensätzen des SPQR Teams, des eigenen Datensatzes und dem Datensatz aus [14] trainiert und verglichen. Die Detektion soll auf ganzen Bildern wie in Abb. 1 geschehen.

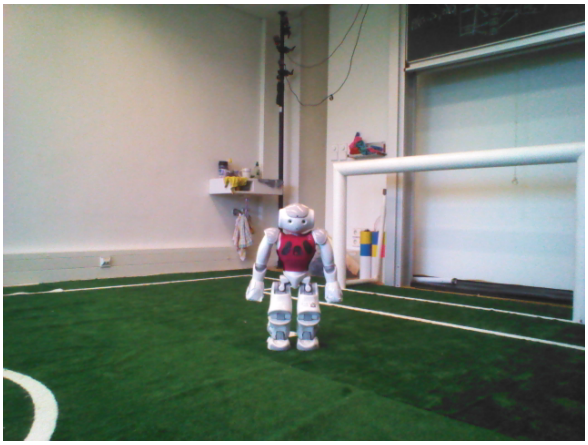


Abbildung 1: Beispielbild, Nao weit entfernt

In der RoboCup Standart Platform League werden die Nao Roboter zum Fussballspielen eingesetzt [27]. Um in den Spielen gute Leistungen zu erzielen, muss man sich einen strategischen Vorsprung verschaffen. Ein solcher Vorsprung wäre die Detektion von Robotern auf dem Feld, sowie eine generelle Erkennung von unterschiedlichen Objekten wie Ball, Torpfosten und Feldlinien. An der Erkennung von Robotern auf dem Feld arbeiten die meisten Teams, jedoch setzt die geringe Rechenleistung der Naos bestimmte Grenzen für den Erkennungsalgorithmus.

Im RoboCup hat sich das B-Human Team als erfolgreich bewährt, sie haben mehrere Weltmeister Titel in der SPL gewonnen. B-Human setzt bei der Erkennung von Robotern auf Farberkennung [23]. Manche andere Teams setzen auf ähnliche Lösungen [10]. Es gibt jedoch auch andere auf Farberkennung basierende Lösungen, wie [19], wo versucht wird, die Orientierung von Robotern auf dem Feld zu bestimmen. Die Anforderungen an die Roboter in der SPL werden jedes Jahr erschwert [27], was nicht generalisierbare Lösungen unattraktiv macht. An dieser Stelle kommen die Neuronalen Netze hervor. Der Vorteil von Neuronalen Netzen ist, dass diese einen allgemeinen Ansatz und hohe Robustheit gegenüber Beleuchtung und anderen äußeren Einflüssen versprechen. Außerdem bieten diese "State-of-the-Art" Lösungen in Objekterkennung und Objektklassifizierung an [16] [26] [21] [22].

Auf Erkennung von Robotern auf dem Feld mit Hilfe von Neuronalen Netzen setzt das SPQR Team. SPQR verwendet CNN nur bei der Klassifizierung von entstandenen Bereichen nach der Segmentierung mit Hilfe der Grünerkennung. Auf ähnliche weise handelt auch [8]. Das Paper von [8] testet das SqueezeNet [12] und das XNOR-Net [20]

und bietet eine Schritt für Schritt Anleitung, wie diese Netze angepasst wurden, um eine Klassifikationsrate von 98% in 1 ms auf dem Nao zu erzielen.

Ein großer Nachteil der verwendeten Methoden von SPQR und [8] ist, dass diese einen Segmentierungsschritt vorher brauchen. An dieser Stelle kommt der Single-Shot-MultiBox-Detector(SSD) hervor [17]. Bei dem SSD Verfahren handelt es sich um die Bestimmung von allen erkennbaren Objekten auf dem Bild in einem Durchlauf, dadurch entfällt der Segmentierungsvorschritt. Das SSD Verfahren wie RCNN gehen einen Schritt weiter, diese verwenden Deconvolution nach der Klassifizierung, um Segmentierungsmasken für erkannte Objekte auf einem Bild Pixelweise zu klassifizieren [22].

Das Ziel von CNNs mit Deconvolution ist es, das Bild mit mehreren Separationsmasken zu generieren [18] [22], um Regionen für alle Objekte von Interesse zu bestimmen, diese können für die Bestimmung der eigenen Position verwendet werden, indem die Ecken auf dem Bild markiert werden.

In der AdultSize League wird von [24] ein Deconvolutionsverfahren mit Segmentierungsmasken verwendet. Das besondere an [24] ist die Weise, in der das Netzwerk trainiert wurde, statt Boxen, wurde für die Markierung der GroundTruth Punkte verwendet, was die Vorarbeit erleichtert, außerdem erkennt das SweatyNet auch Kreuzungen, Bälle und Goalpfosten auf dem Feld.

Die Teams, die in der SPL antreten, haben sich mit der Erkennung von Robotern auf dem Feld beschäftigt, manche versuchen auch die Orientation und Feind/Freund Ermittlung durchzuführen. Die meisten Herangehensweisen können in 2 Gruppen unterteilt werden, die die mit Hilfe der Farberkennung Arbeiten und versuchen "per Hand" die Roboter zu beschreiben, und die, die Neuronale Netze für die Erkennung von Robotern verwenden [8] [5].

In der "per Hand" Roboter Erkennung kann der Algorithmus des B-Human Teams als eine Richtlinie betrachtet werden. Im ersten Schritt ermittelt das B-Human Team den Feldrand, vom Feldrand her überprüft deren Algorithmus die Pixel in vertikalen Linien nach unten, falls diese nicht-grün sind, werden die untersten in diesen Linien als mögliche Endpunkte von Füßen oder Händen markiert und zu einer Region zusammengefasst. Diese Regionen werden nach möglichen Team T-Shirts untersucht, dafür werden Informationen vom GameController verwendet, die die Farben der Teams beinhalten. Je nachdem wie sich diese Farben unterscheiden, werden unterschiedliche Vergleichsmethoden verwendet. Falls eine Teamzuweisung nach dem Vergleich geschehen ist, werden die Position des detektierten Roboters zum Feldmodell hinzugefügt. [23]

Die Teams, die Neuronale Netze verwenden, starten ähnlich den "per Hand" Teams, indem Sie mögliche Regionen mit Robotern mithilfe der Grünerkennung bestimmen, danach werden die erhaltenen Regionen als Eingabe an ein CNN übergegeben und entschieden, ob es sich um einen Roboter handelt oder nicht. Außer Robotererkennung müssen die Roboter auf dem Feld auch die eigene Position bestimmen, den Ball erkennen, sich bewegen und mit anderen Robotern kommunizieren, deswegen darf der Robotererkennungsalgorithmus nicht viel Leistung verbrauchen. [12]

In dieser Arbeit werden das 3 Schichtige CNN des SPQR Teams und das miniSqueeze4 von [8] nachgebaut, um diese auf dem Sliding Window zu verwenden, um die

Klassifizierung von Objekten auf dem Feld zu erzielen. Das Verfahren aus der AdultSize League ist zwar sehr vielversprechend, jedoch läuft es auf i7 und GTX760 in 9ms auf der GPU und 55ms auf der CPU, diese Leistung kann der NAO mit seinem 500 MHz Prozessor nicht bringen.

Der Rest der Arbeit ist wie folgt strukturiert. Im Abschnitt 2 werden einige Grundlagen und Begriffe aus dem Feld der Neuronalen Netze vorgestellt. Im Abschnitt 3 werden die untersuchten Netzwerke vorgestellt, sowie Trainingsverfahren, Metriken, Implementierung und Datensätze diskutiert und vorgestellt. In Abschnitt 4 wird der Aufbau der Experimentellen Auswertung erklärt, sowie Ergebnisse in Form von Tabellen und Bilddaten vorgestellt und diskutiert. Abschnitt 5 beinhaltet die Diskussion der angewandten Trainingsverfahren und den Vergleich zu den Arbeiten des SPQR Teams [5], des miniSqueezeNet4 [8] und der auf HOG Detektion basierten Methode aus [14]. Außerdem wird eine mögliche Erweiterung des Verfahrens vorgestellt.

2 Convolutional Neural Networks (CNN)

Convolutional Neural Networks sind Werkzeuge für Objekterkennung in Bild- und Audiodaten. CNNs bekommen als Eingabe die Bilddaten und geben Klassifikationswahrscheinlichkeiten aus. Die Convolution Neural Networks sind durch Inspiration aus der Biologie entstanden.

In diesem Abschnitt werden mehrere Begriffe aus der Thematik der Neuronalen Netze eingeführt, diese sind primär auf Englisch, da es in diesem Gebiet so üblich ist.

2.1 Aufbau

Der Hauptbestandteil eines CNN ist ein Neuron. Ein Neuron in der 2D Faltungsschicht ist eine HxWxD Faltungsmatrix (synonyme: Kern, Filterkern, Filteroperator, Filtermaske, Faltungskern, Konvolution, convolution kernel oder Feature), die als Eingabe eine HxWxD Matrix erhält und einen 1x1x1 Wert ausgibt. Dabei wird die Faltungsmatrix mit der Eingabematrix Pixelweise multipliziert und die Ergebnisse summiert, die Normierung ist in den Faltungswerten enthalten. Die Grösse HxW der Faltungsmatrix bezeichnet man als das Rezeptive Feld. Neuronen in vollständig verbundenen Schichten nehmen 1D Vektoren an, diese entstehen mit Hilfe der "Flatten" Funktion, in dem das Feature Volumen der Vorschicht, vollständig in einen 1D Vektor transformiert wird.

Faltung:

$$I^*(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{n} I(x+i-a, y+j-a)k(i,j)$$

$I^*(x,y)$ ist das Ergebnispixel, damit ist, a ist der Mittelpunkt der quadratischen Faltung und $k(i,j)$ ist ein Element der Faltung. Um den Mittelpunkt a eindeutig zu definieren, sind ungerade Abmessungen der Faltungsmatrizen notwendig.

Stride ist eine natürliche Zahl die angibt, in welchen Schritten sich der Neuron auf der Eingabe bewegt.

Padding wird für die Erhaltung der Dimension des Ausgabefeldes, gegenüber dem Eingabefeld, verwendet. Bei der Anwendung eines SxSx1 (mit S>1) Neurons auf ein Bild, wird das Ausgabebild kleiner als die Eingabe, Padding macht die Eingabematrix größer, indem es an jeder Seite der Eingabematrix soviele Nullzeilen, bzw. Nullspalten anhängt wie angegeben. [16]

Ein CNN besteht aus mehreren Schichten von Neuronen. Die Anwendung der Faltungsmatrix entlang der Dimensionen der Eingabe produziert eine Feature Map. Alle produzierten Feature Maps werden zu einem Feature Volumen zusammengefasst, auf dieses werden die Neuronen in der nächsten Schicht angewendet. Bei N Feature Maps werden die Neuronen in der nächsten Schicht die Tiefe N haben. Die erste Schicht bekommt als Eingabe das eigentliche Bild, alle weiteren Schichten bekommen die Ergebnisse der Vorschichten. Die Neuronen aus einer Schicht zur nächsten werden gewichtet verbunden, das Bedeutet, zum Beispiel, falls das Neuron 17 aus der Schicht 2 für das Neuron 3 aus der Schicht 3 wichtig ist, dann besteht zwischen den beiden Neuronen eine Verbindung und diese Verbindung hat ein Gewichtswert>0. Jedes Neuron in der nächsten Schicht hat die Tiefe der Anzahl von Neuronen aus der Vorschicht, falls das nicht der Fall ist, dann entstehen mehrere Feature Volumen.

1x1xN Neuronen haben eine besondere Funktion, diese werden für die Reduktion der Dimensionen des Feature Volumens, die Hinzufügung der Tiefe zum Netzwerk oder der Hinzufügung von Nichtlinearitäten verwendet.

Pooling wird für die Reduzierung der Rechenarbeit verwendet, üblicherweise wird Maxpooling verwendet, dabei wird der Maximalwert aus einer SxS Region auf dem Bild ausgewählt und in das Ausgabebild an der entsprechenden position Eingetragen, diese SxS Region wird mit Stride L auf dem Bild weiter fortbewegt, damit das Ausgabebild kleiner ist und trotzdem die wichtigsten Features enthält.

Normalisierung hat als Ziel die absoluten Werte in den Neuronen gering zu halten, indem es negative Werte aus den Feature Maps auf 0 setzt.

Das Ziel von CNN ist die Erkennung zu generalisieren. Wenn die Neuronen zu stark an die Trainingsdaten angepasst wurden, so dass diese nichts außer den Trainingsdaten mehr erkennen können, dann bezeichnet man das als Overfitting.

Die Fully Connected (FC) Schicht berechnet die Wahrscheinlichkeit für ein Objekt im Bild. Im Gegensatz zu vorherigen Schichten, trägt jede Feature Map aus der Vorschicht zu der Entscheidung bei.

$$y_i = f(z_i)$$

mit

$$z_i = \sum_{j=1}^{m_1^{l-1}} w_{i,j}^l y_i^{l-1}$$

l ist die Schicht, $l-1$ ist die FC Schicht, $w_{i,j}^l$ ist das Gewicht zwischen der i-ten Feature Map in Schicht l und der j-ten Feature Map in Schicht $l-1$. $f(z_i)$ ist eine Wahrscheinlichkeitsfunktion wie Softmax oder Sigmoid.

2.2 Training

Das Netzwerk wird beim Training im Vorwärts und Rückwärtspass ausgeführt. Im Vorwärtspass wird eine Klassifizierung durchgeführt, im Rückwärtspass werden die Gewichte gemäß der Abweichung vom Wahrheitswert geändert. Dies wird mehrere Mal auf dem gesamten Datensatz wiederholt, einen Durchlauf auf dem gesamten Datensatz wird als "Epoche" bezeichnet.

Beim Training werden Schichten verwendet, die nur im Rückwärtspass aktiv sind. Dropout Schicht wird verwendet um Overfitting zu vermeiden, dabei wird ein angegebener Prozent an zufällig ausgewählten Neuronen beim Lernen deaktiviert.

Die Aktivierungsschicht sorgt für die Anpassung der Gewichte. Früher wurde für die Aktivierungsschicht die Sigmoid oder tanh Funktionen verwendet, jedoch hatten diese Probleme die Gewichte bei Tiefen Netzwerken richtig anzupassen (verschwindende Gradienten), da der Großteil der Anpassung in den Tieferen Schichten stattfand. Nach der Entwicklung der ReLU Aktivierungsfunktion wird nur diese fast überall verwendet, da sie schneller zu berechnen ist und nicht vom Problem der verschwindenden Gradienten leidet.

Bei CNN werden nur die Features trainiert, alle anderen Parameter müssen bei der Erstellung des Netzwerks angegeben werden. Zu diesen Hyperparametern gehört die Input Größe, die Tiefe des Netzwerks - Schichtanzahl und Anordnung, Dropout Prozent, Lernfunktion (ReLu), sowie die Filter und Pooling Grösse.

3 Erkennung von Robotern mit CNN

Das Ziel ist es eine Erkennung der Nao Roboter auf einem ganzen Bild durchzuführen. Für diese Aufgabe werden 2 vielversprechende CNN, das CNN des SPQR Teams mit Tiefe 3 [5] und miniSqueezeNet4 [8], implementiert, getestet und verglichen. Im Unterschied zu den genannten Arbeiten, wird die Erkennung von Robotern nicht auf vorheriger Segmentierung basieren. Das Sliding Window Verfahren soll für die Erkennung der Roboter auf dem gesamten Bild angewendet werden. Für das Trainieren und Evaluieren werden die Datensätze des SPQR Teams [5], der Datensatz aus [14] und ein im NaoTH Labor selbst erstellter Datensatz verwendet.

3.1 Untersuchte Detektoren

In [5] wurde ein Datensatz für das Trainieren von CNN Modellen erstellt und 3 CNN unterschiedlicher Tiefe trainiert und evaluiert. Beim Aufbau der Netzwerke wurden 3 bis 5 Faltungsschichten verwendet. Der Arbeit zu folge, hat die erste Schicht "Conv1" 64 Filter mit Filtergröße 5x5, die optionale Schicht "Conv2" 128 Filter mit Filtergröße 5x5 und die optionale Schicht "Fully Connected" (Violet in Abb. 2 384 Filter. Aus der Arbeit konnten nicht alle Hyperparameter bestimmt werden, deswegen wurden folgende Hyperparameter aus eigenen Schätzungen bei der Implementation verwendet:

- Bildgröße: 24 x 24, da diese Bildgröße von miniSqueezeNet4 [8] benutzt wird.

- Anzahl Filter der FC Schicht vor Softmax: 64.
- Anzahl Filter der FC Softmax Schicht: 2, da 2 Klassen erkannt werden sollen, ”Hintergrund” und ”Nao”.

Die Schreiber der [5] Arbeit haben betont, dass der eigentliche Beitrag der Arbeit ein Datensatz mit Nao Robotern ist und das Netzwerk nur zum Testen verwendet wurde, obwohl dieses eine Klassifikationsrate von 100% bei 14-22 FPS aufweisen konnte.

Nur die Blau markierten Ebenen in Abb. 2 wurden in dieser Arbeit implementiert. Alle weiteren Referenzen werden sich auf das 3 Schichtige SPQR Netzwerk beziehen oder als ”SPQR_d3” bezeichnet.

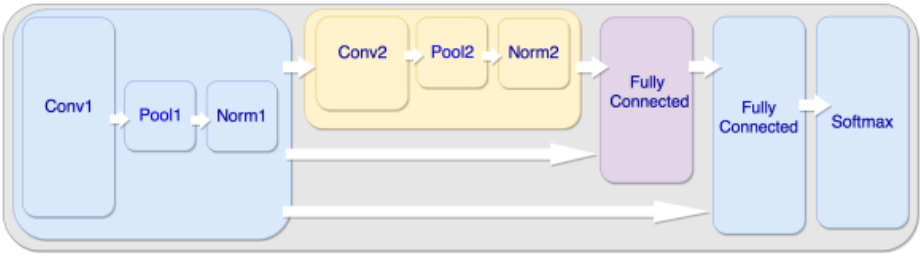
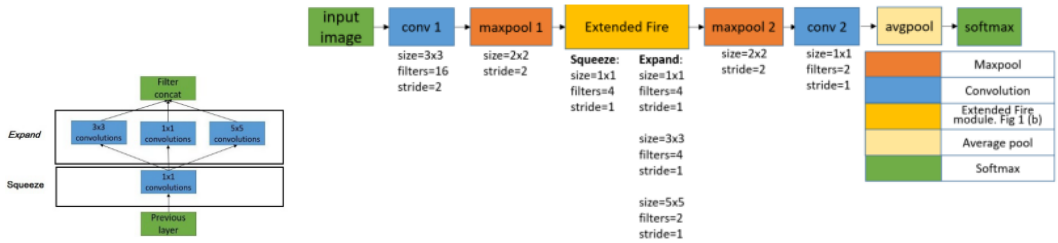


Abbildung 2: Aufbau des SPQR Netzwerkes aus [5]

In [8] wurde eine minimisierte Version des SqueezeNets [12] vorgestellt. Das miniSqueezeNet4 konnte eine Inferenz Zeit von 1 ms und eine Klassifikationsrate von 98,30 % erreichen, wegen dieser Geschwindigkeit eignet sich das miniSqueezeNet4 zum Benutzen in einem Sliding Window Verfahren.



(a) (b)

Abbildung 3: CNN Aufbau aus [8]. (a) Das Extended Fire Modul. (b) Aufbau des miniSqueezeNet4.

In Abb. 2 ist das miniSqueezeNet4 abgebildet, es besteht aus 3 Pooling und 6 Faltungsschichten. Das Netzwerk hat keine Fully Connected Schicht und benutzt eine kleinere Anzahl an Filtern im Vergleich zu dem Netz des SPQR Teams.

3.2 Training und Evaluierung

Dem Trainieren wird viel Aufmerksamkeit zugeteilt, da es die Qualität des Netzwerks deutlich verbessern oder verschlechtern kann. Im RoboCup wird man nicht immer gleiche

Lichtverhältnisse, Hintergrund oder Farben der Roboter T-Shirts haben, deswegen muss eine Möglichkeit existieren das Netzwerk auf neuen Daten schnell und sinnvoll zu trainieren.

Um das Training zu vereinfachen und die Genauigkeit des Netzwerks zu steigern, müssen alle Daten zuerst auf eine Norm gebracht werden, dies geht auch für Bilder, auf die das Netzwerk angewendet werden soll, deswegen soll die Normierung keinen hohen Rechenaufwand darstellen. Für die Normierung der Daten werden zuerst die RGB Kanäle des Bildes auf Werte zwischen 0 und 1 gebracht, indem die Bildmatrix durch 255 geteilt wird. Im zweiten Schritt wird der Durchschnittswert der RGB Kanäle auf 0 gesetzt, indem dieser für jeden Kanal berechnet und von diesem Subtrahiert wird. Ein Teil der Normierung ist die Anpassung der Größe der Bilder, da das Netzwerk einen Eingabetensor bestimmter Größe erwartet.

Es ist möglich vor dem Training die Gewichte der Schichten einzufrieren. Um das Netz an neue Helligkeitsbedingungen anzupassen, können alle Schichten, außer der ersten, eingefroren werden und das Training für wenige Epochen auf neuen Daten durchgeführt [11].

Während des Trainings können die unterschiedlichen Metriken beobachtet werden. Eine besondere Rolle spielt die Verlustmetrik, für die, die Kategorische Kreuzentropie verwendet wurde. Die Verlustmetrik wird beobachtet und falls diese eine Verbesserung auf den Validierungsdaten darstellt, dann werden die Gewichte des Netzes gespeichert. Falls nach einer bestimmten Anzahl an Epochen es zu keiner deutlichen Verbesserung der Verlustmetrik kommt, dann wird der Lernvorgang beendet, um Overfitting zu vermeiden.

Die Gewichte eines neuen Netzwerks werden zufällig gesetzt, deswegen können mehrere Trainingsdurchläufe eines neuen Netzwerks gemacht werden und die besten Gewichte mit kleinstem Verlust auf Validierungsdaten ausgewählt.

Eine weitere, in dieser Arbeit verwendete Trainingsmethode, besteht aus dem Training nicht an binären Daten, sondern an erwarteter Ausgabe. Zum Beispiel, wenn das Netzwerk als Eingabe ein zur Hälfte mit Nao Roboter gefülltes Bild bekommt, dann ist die erwartete Ausgabe 0.5 für Nao Roboter Präsenz im Bild. Dieses Verfahren ist besonders sinnvoll, wenn man bedenkt, dass die zum Training erstellten Daten aus der Sliding Window Funktion stammen, die durchaus zuvor beschriebene Bilder ausgibt.

3.3 Implementierung

Die CNN Klassifikatoren wurden in der Programmiersprache Python 3.6.2 mit Hilfe der Bibliothek Keras [7] umgesetzt. Keras ist eine in Python geschriebene Bibliothek, die schnelle und benutzerfreundliche Bildung von CNN ermöglicht. Für die Berechnungen des CNN kann Keras, die von Google entwikelte API Tensorflow [4], CNTK [25] oder Theano [28] verwenden.

Im Rahmen dieser Arbeit wird das Tensorflow Backend verwendet. Es gibt bereits Erfahrungen im Internet, dass die NAO Tensorflow unterstützen können.

Für die Markierung der Daten wurde ImgLab [3] verwendet, da es bereits in [14] eingesetzt wurde.

3.4 Datensätze

Um eine Klassifizierung mit einem Neuronalen Netzwerk durchzuführen muss dieses erst trainiert werden, dafür werden Trainings und Validierungsdaten benötigt. Wichtig für die verwendeten Daten ist, dass diese in Klassen unterteilt sind, dadurch wird der Trainingsalgorithmus wissen, welche Ausgabe erwartet werden soll und das Netzwerk im Rückwärtspass dementsprechend ändern. Damit ein CNN erfolgreich trainiert werden kann, müssen gewisse Anforderungen für die Bilder und die zu erkennende Objekte erfüllt werden.

Anforderungen an die Bilder kommen aus den sich ändernden Lichtverhältnissen, Kameraparametern und Verschwemmungen wegen der Bewegungen der Roboter. Bestenfalls möchte man während des Spiels, dass die Bilder wie in Abb. 4a aussehen, jedoch ist das oftmals nicht der Fall. Bilder können sich sehr stark in Qualität unterscheiden. Im Fall der Abb. 4b und Abb. 4c sehen die Bilder mehr Rötlich oder Bläulich aus, für Menschen ist das kein Problem, jedoch stellt es eins für die Erkennung dar, falls das CNN nicht dafür vorbereitet ist. Farbveränderungen ist nicht das Einzige womit das CNN arbeiten muss. Die Abb. 4d wurde mit schlecht eingestellten Kameraparametern aufgenommen. Wie man auf der Abb. 4e sieht, wird die Erkennung bei dieser Art von Bildern auch für das menschliche Auge schon schwierig, im Hintergrund können die 2 Roboter und ein Mensch kaum erkannt werden.

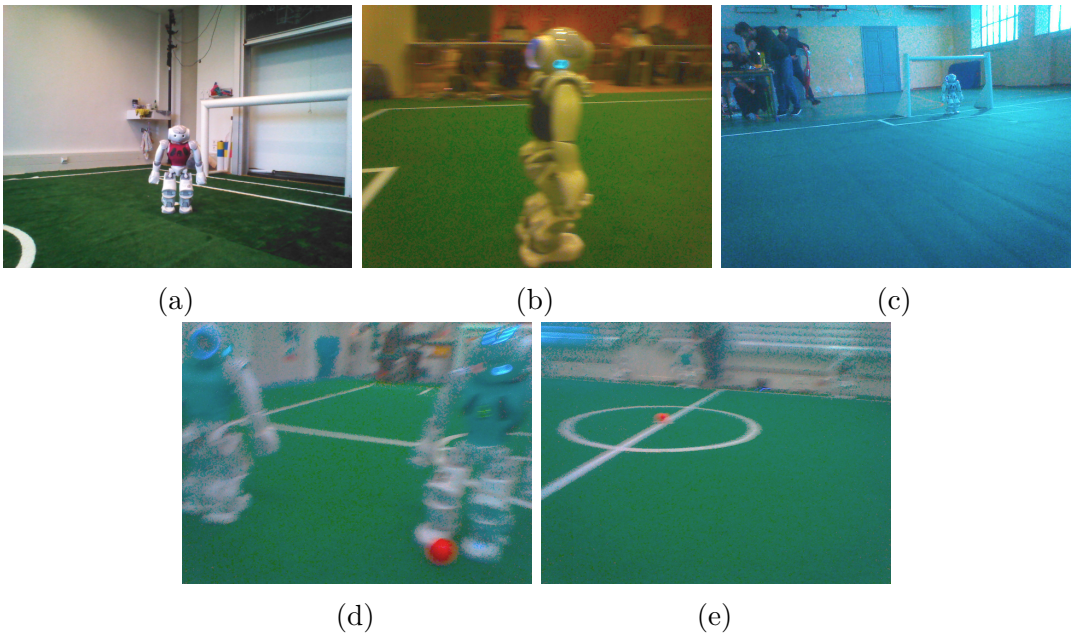
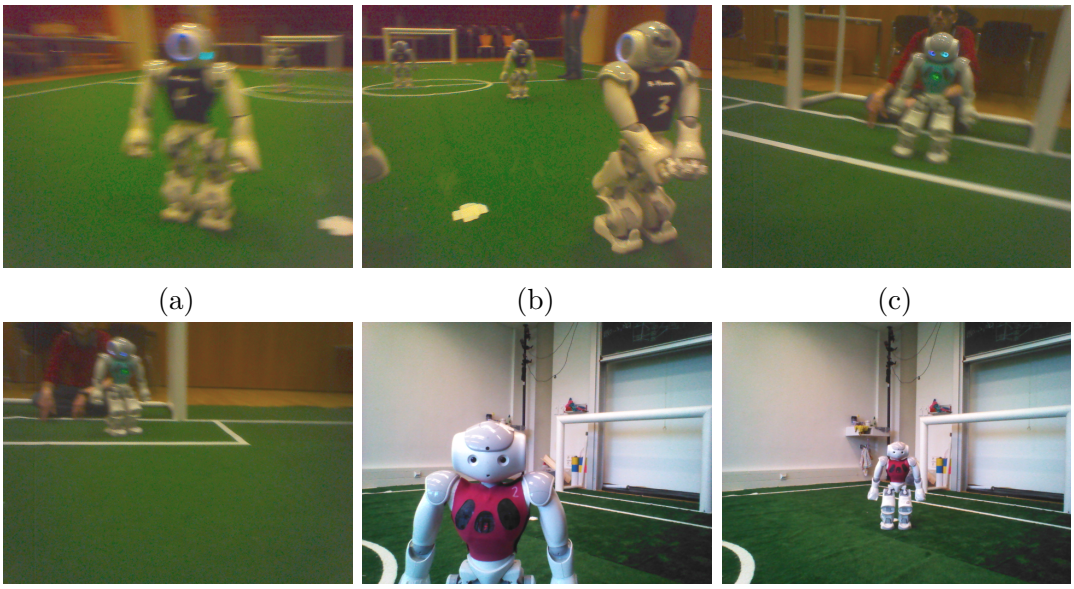


Abbildung 4: (a) Gutes Bild. (b) Rötliches Bild. (c) Bläliches Bild. (d) Bild mit Blur. (e) Bild mit stark verschwommenen NAO im Hintergrund.

Die Qualität der Bilder hat eine große Auswirkung. Blur, Kontrast, Helligkeit, Rauschen und andere Merkmale können die Klassifikation beeinträchtigen, wie dies in [9] und [6] betrachtet werden kann. Um das Netzwerk gegen Veränderungen in der Bild-

qualität robuster zu machen, können die genannten Merkmale in den Bildern beim trainieren verändert werden.

Um Objekte mit CNN besser zu erkennen, müssen die Bilder zum Trainieren des CNNs mit diesen Objekten in unterschiedlichen Haltungen (Abb. 5a, Abb. 5b), Rotierungen (Abb. 5c, Abb. 5d) und Entfernung (Abb. 5e, Abb. 5f) aufgenommen werden.



(a) (b) (c)

(d) (e) (f)

Abbildung 5: (a) Haltung der NAO: Aufrecht, Aufnahme von Vorn. (b) Haltung der NAO: Bereit zum gehen, Aufnahme von Hinten. (c) Bild mit Nao rotiert nach rechts. (d) Bild mit Nao ohne rotierung. (e) Bild mit Nao im Vordergrund. (f) Bild mit Nao im Hintergrund.

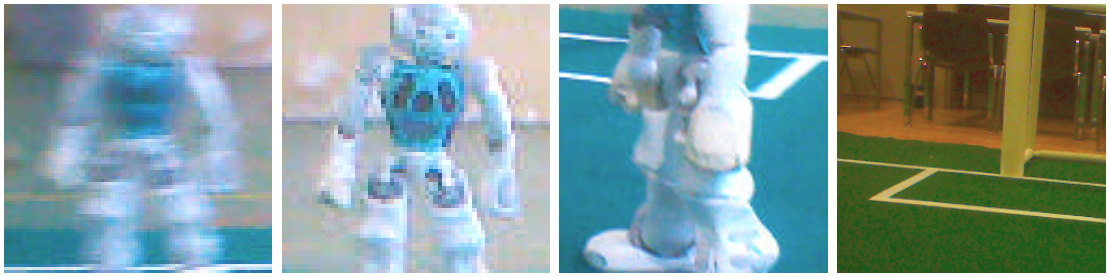
Im Rahmen dieser Arbeit wurde ein eigener Datensatz mit Hilfe des NaoTH Teams erstellt. Außerdem wurden die Datensätze aus einer vorherigen Arbeit [14] und die des SPQR Teams verwendet [5]. Alle Datensätze erhalten nur RGB Bilder aus der NAO Perspektive.

Der Datensatz des SPQR Teams besteht aus ganzen Bildern ohne Markierungen (Abb. 5c), sowie Bildausschnitten unterschiedlicher Grössen, Qualität, Kameraeinstellungen, mit und ohne NAO oder nur Teilen von diesen (Abb. 6a - 6d).

Der Datensatz von [14] enthält .xml Dateien mit Markierungsinformationen für im Datensatz enthaltene ganze Bilder. Dieser Datensatz wurde mit dem Tool ImgLab [3] erstellt. In Abb. 7 ist ein Bild mit Markierungen aus [14] zu sehen. Der eigene Datensatz wurde im NaoTH Lab erstellt und mit Hilfe des imglab Tools auch markiert, da der Parser für [14] Datensatz sowieso gemacht werden musste.

Der eigene Datensatz enthält Videoaufnahmen von einem spielenden Nao (Abb. 8c), sowie Bilder von einem unterschiedlich positionierten NAO (Abb. 8b.), in unterschiedlichen Haltungen (Abb. 8a), auf dem Feld.

Um die Daten für das Trainieren vorzubereiten wurde ein Skript erstellt, dass die



(a) (b) (c) (d)

Abbildung 6: (a) SPQR Ausschnitt mit verschwommenen Nao. (b) SPQR Ausschnitt mit NAO in niedrigeren Auflösung. (c) SPQR Ausschnitt mit nur den Beinen der NAO. (d) SPQR Ausschnitt ohne Nao.

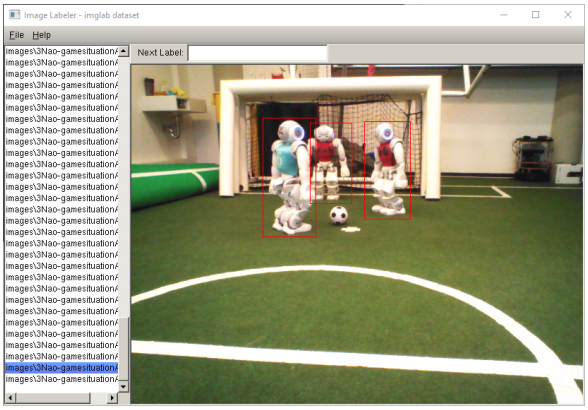
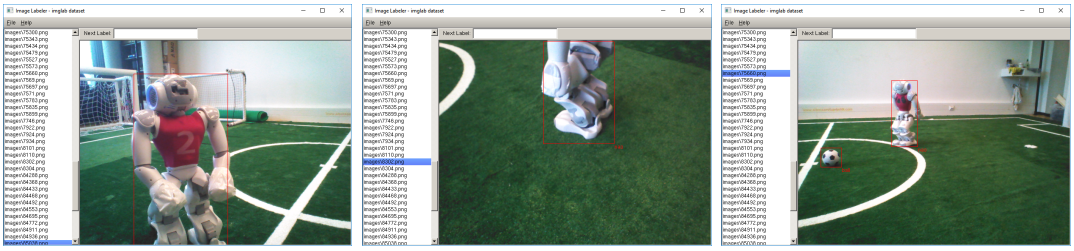


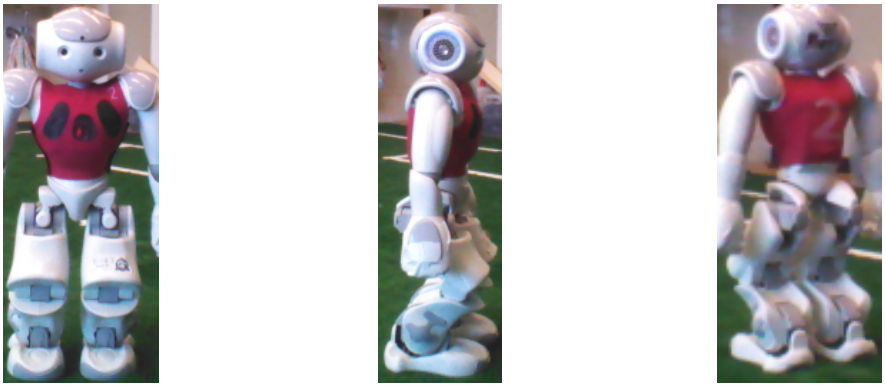
Abbildung 7: [14] Bild mit Markierungen in ImgLab



(a) (b) (c)

Abbildung 8: (a) Bild mit Nao nah an der Kamera in gehender Haltung. (b) Bild mit mit Beinen eines NAO von der unteren Kamera. (c) Bild mit einem spielenden Nao.

Ground Truth direkt ausschneidet (Abb. 9a - 9c), sowie ein Sliding Window, das für die Objekterkennung eingesetzt wird, zum Ausschneiden des Ground Truths verwendet, um die Datengenerierung dem Ablauf der zu verarbeitenden Bilder zu ähneln.



(a) (b) (c)

Abbildung 9: (a) Bild mit NAO von Vorne. (b) Bild mit NAO von einer Seite. (c) Bild mit NAO von Hinten.

Die Sliding Window Funktion nimmt 3 Parameter an, Höhe und Breite des Fensters, sowie die Schrittdistanz - Anzahl Pixel, um die das Fenster verschoben werden soll. Nachdem die Sliding Window Funktion das Bild zerteilt hat, wird für jedes Teilbild entschieden, ob es in die Klassensammlung aufgenommen wird, oder nicht. Die Entscheidung erfolgt indem der Jaccard Index (J(A,B)) zwischen dem Teilbild und der Ground Truth gebildet wird, falls dieser größer als 0.5 ist, dann wird das Bild zur Klasse der Roboter hinzugefügt. Zwischen zwei Teilbildern sollte der Jaccard Index auch nicht über 0.5 liegen, um Overfitting zu vermeiden.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Es könnten mehr Daten erstellt werden, indem die Bilder horizontal gespiegelt werden, um wenige Grad rotiert, das Farbhistogramm und Kontrast verändert werden und Bildrauschen oder Motion Blur eingefügt werden.

Die Skalierung erfolgt, indem die Parameter der Sliding Window Funtion verändert werden, damit kann man aus einem Bild, Bilder unterschiedlicher Auflösung generieren.

In einem anderen Schritt wurde der Hintergrund mit einem Sliding Window ausgeschnitten. Die Auswahl der Bilder erfolgt jedoch teilweise per Hand, da es zu viele gleiche Bilder gibt, um Overfitting zu vermeiden. Um diese Aufgabe zu vereinfachen, kann der Hintergrund jedoch in den Bildern, wie andere Objekte, markiert werden.

3.5 Jaccard Index, Intersection over Union (IOU)

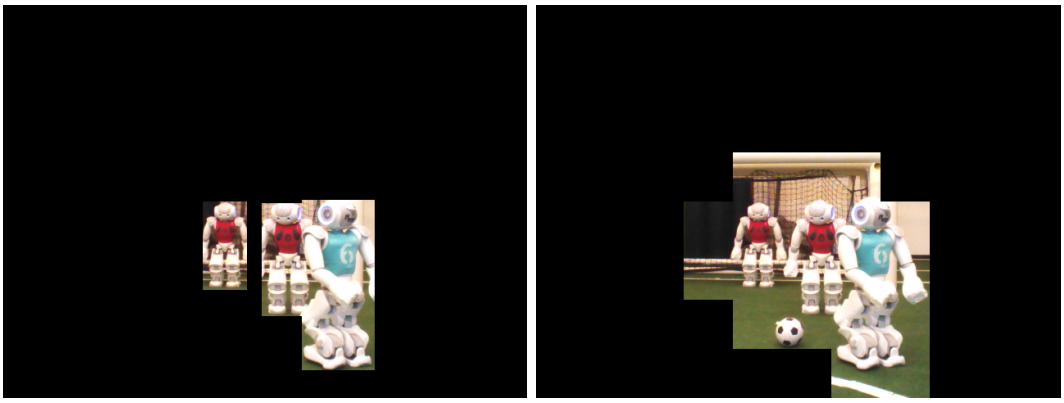
Um die IOU zu bestimmen, werden zwei binäre Masken erstellt, eine Maske m_1 (Abb. 10a) wird aus den Ground Truth Markierungen erstellt, und die Andere wird anhand von

Erkennungen m_2 (Abb. 10a) erstellt. Das IOU wird wie der Jaccard Index berechnet, nur Pixelweise auf den Bildern mit logischen Operatoren, mit der Formel:

$$and_{1,2} = m_1 \wedge m_2$$

$$or_{1,2} = m_1 \vee m_2$$

$$IOU(m_1, m_2) = sum(and_{1,2})/sum(or_{1,2})$$



(a) (b)

Abbildung 10: (a) Angewendete Ground Truth Maske auf das entsprechende Bild. (b) Angewendete Erkennungsmaske auf das entsprechende Bild.

Im Rahmen dieser Arbeit wird IOU auf den gesamten Bildern berechnet, es werden keine einzelnen Detektionen von Robotern durchgeführt, sondern nur Ausschnitte in den Bildern klassifiziert, wie es in Abb. 10 dargestellt ist.

4 Experimentelle Auswertung

Ziel der Arbeit war es eine Robotererkennung mit Hilfe eines CNN durchzuführen. Obwohl die Architektur eines Netzes eine sehr große Rolle spielt, wurde in dieser Arbeit mehr auf geeignete Trainingsverfahren eingegangen, da auf diese Weise kleinere und damit schnellere CNN benutzt werden können.

4.1 Training

Es wurden das SPQR_d3 und das miniSqueezeNet4 auf folgende Weise trainiert.

Das Trainingsverfahren wurde in mehrere Schritte unterteilt. Im ersten Schritt wird das Netzwerk an SPQR [5] Datensatz in maximal 50 Epochen angepasst. Im zweiten Schritt werden alle Schichten außer der ersten Faltungsschicht vom Trainieren ausgeschlossen und nur auf den im NaoTH Labor erstellten Datensätzen in maximal 5 Epochen trainiert. Im Dritten Schritt wird der Trainingsprozess auf der gesamten Datenmenge ausgeführt, in maximal 20 Epochen. Zum Vergleich wurde außerdem ein

Trainingsdurchlauf von Anfang auf dem gesamten Datensatz in maximal 50 Epochen ausgeführt. Außerdem wurde das vorgeschlagene Trainingsverfahren mit Verwendung nicht binärer Erwartungswerte im NaoTH Datensatz ausgeführt, indem die Erwartungswerte auf den Jaccard Index zwischen dem Teilbild und der Ground Truth gesetzt wurden. Für Evaluierungszwecke werden die Gewichte aus dem ersten Schritt, für das Training auf Erwartungswerten, wiederverwendet.

Es ist zu beachten, dass alle im NaoTH Labor erstellten Datensätze vor dem Training mit der Sliding Window Funktion zerteilt worden sind, deswegen sind mehr Bilder mit Hintergrund, statt mit Nao im Datensatz verträten.

Es wurde die kategorische Kreuzentropie als Verlustmetrik für das Trainieren verwendet, mit der Adam [13] Lernrateoptimierung. Der Standartlernwert wurde für ersten Schritt auf 0.001 und für Schritte 2 und 3 auf 0.0003 gesetzt. Die Verwendung einer binären Kreuzentropie(Log Loss) sollte die Ergebnisse nicht beeinflussen.

Für alle NaoTH Trainings und Evaluierungsdaten wurde die Sliding Window Funtkion mit folgenden Parametern ausgeführt: Schritt = 60px, Höhe = 120 px, Breite = 120 px Diese ermöglichten eine Überlappung der Ausschnitte von 50% und zerteilten das 640x480 Große Bild in 63 Teile, 9 Horizontal und 7 Vertikal.

Die finale Evaluation auf Bildern geschieht auf 6 ausgewählten und markierten Bildern aus allen Datensätzen, die die Netzwerke nie beim Trainieren gesehen haben. Diese werden vor der Evaluation mit der Sliding Window Funktion zerteilt und an die Netzwerke geschickt. Erkennungswahrscheinlichkeiten über dem Grenzwert von 0.7 für die NAO Klasse wurden behandelt wie als "Nao" anerkannt.

Während des Trainings und der Evaluierung werden 4 Metriken auf dem Validierungsdatensatz berechnet und beobachtet: Verlust, Kategorisierugsrate (Accuracy), Precision, Recall. Für diese werden die True Positive (TP), die True Negative (TN), False Positive (FP) und False Negative (FN) Werte berechnet werden.

- Accuracy sagt aus, wie genau die Aussage des Netzwerks ist: $A = (TP + TN)/(TP + TN + FP + FN)$.
- Precision sagt aus, wie genau es die True Positives erkennt: $P = TP/(TP + FP)$.
- Recall sagt aus, wie groß der Anteil der Erkennung von allen Positiven ist: $R = TP/(TP + FN)$.

Für die Messung der Qualität des Netzwerkes wird die durchschnittliche Precision (eng. average precision - AP) verwendet. Die AP Metrik wird wie Folgt berechnet:

$$AP = \int_0^1 p(r)dr = \sum_{k=1}^{N} P(k)\Delta r(k)$$

N ist die totale Anzahl an Bildern, $P(k)$ ist die Precision von k Bildern, $\Delta r(k)$ ist die Änderung des Recall Wertes zwischen k und $k-1$.Ein AP Wert von 0.5 für binäre Klassifikation bedeutet, dass jedes 2te Bild richtig erkannt wird, damit wäre eine Zufallsfunktion ein genau so guter Klassifikator. Durch Aufzeichnung von $P(k)$ und

$\Delta r(k)$ Werten entsteht die PR-Kurve, die für die Evaluierung verwendet wird. In der PR-Kurve wird die Precision in abhängig vom Recall eingezeichnet, was eine Aussage über die Qualität eines Klassifikators gibt. Zum Beispiel, um 100% von NAO Robotern als solche zu erfassen, wird man den Grenzwert für die Erkennung verringern müssen und damit wird man mehr Bilder, die keine NAO darstellen, als "NAO" klassifizieren. Die Anzahl an False Positive steigt an - die Precision wird deswegen kleiner.

4.2 Auswertung, Diskussion der Ergebnisse

Auf den PR-Kurven in Abb. 11 kann man erkennen, dass beide Netzwerke einen AP Wert von 0.99 auf SPQR Validierungsdaten erreichen können. Für das SPQR_d3 Netzwerk war die Klassifizierungsrate von 100% angegeben, und das miniSqueezeNet4 weist bessere Ergebnisse als in [8] auf. Der Grund dafür hängt mit der Einfachheit der Trainingsaufgabe zusammen, es werden nur zwei Klassen trainiert und das zu Klassifizierende Objekt hat wenig Variation im Aussehen, zum Beispiel im Vergleich zu Menschen oder Tieren. Außerdem gibt es eine ausreichende Menge an Daten. Der SPQR Datensatz hat insgesamt etwa 16000 Bildausschnitte, davon sind 7000 mit Nao Robotern, im Vergleich hat der CIFAR10 Datensatz [15], dass zum Evaluieren von CNN verwendet wird, 5000 Bilder pro Klasse für 10 Klassen.

Der NaoTH Datensatz mit der Sliding Window Funktion bringen etwa 16000-25000 Teilbilder in das Training, obwohl die Proportion an Bildern mit Nao Robotern deutlich kleiner ist als ohne, da der Großteil der Bilder aus Hintergrund besteht.

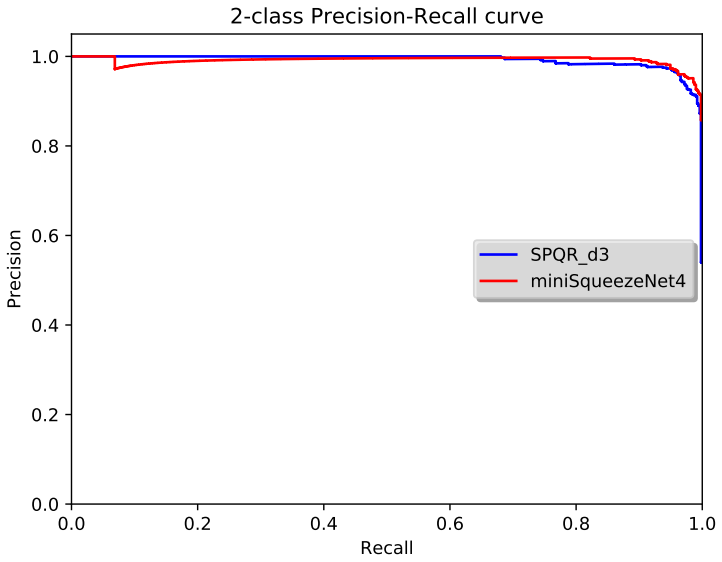


Abbildung 11: PR-Kurven beider Netzwerke auf Validierungsdaten des SPQR Teams nach dem Training auf dem SPQR Datensatz (1. Stufe).

Bei der Evaluierung legt man mehr Wert auf die Verbesserung der Erkennung auf

Iter	AP NaoTH	AP SPQR	max IOU	mean IOU	Zeit in ms
1	0.4294	0.9809	0.9091	0.2986	14.6286
2	0.4457	0.9765	0.9091	0.3001	14.6095
3	0.5141	0.9829	0.9091	0.3514	15.6571
4	0.4744	0.9815	0.9091	0.3220	15.8095
5	0.4886	0.9919	0.6219	0.2757	15.8762
6	0.4114	0.9799	0.9091	0.3008	15.5429
7	0.5121	0.9834	0.7010	0.3089	16.8095
8	0.4442	0.9849	0.9091	0.3430	15.5714
9	0.4595	0.9779	0.7068	0.2573	15.6667
10	0.4451	0.9823	0.9091	0.3157	15.7714
Avg	0.4624	0.9822	0.8393	0.3073	15.5943

Tabelle 1: Evaluierungsergebnisse des SPQR_d3 Netzes nach dem Training auf dem SPQR Datensatz (1. Stufe).

”neuen” Bildern aus eigenem Datensatz und dem Datensatz aus [14].

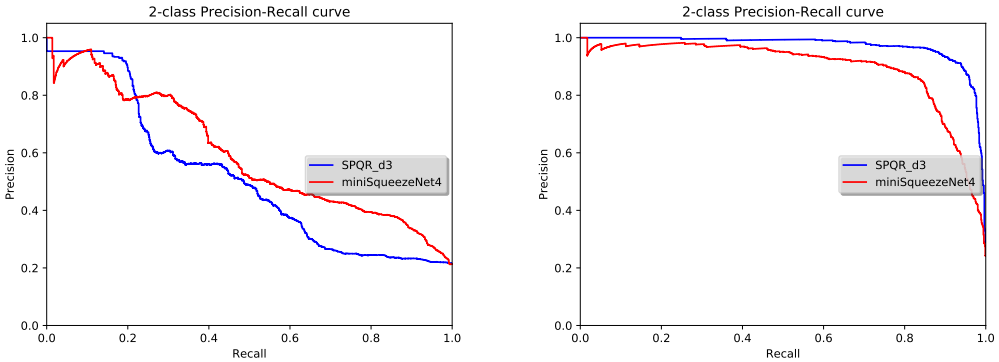
In Abb. 12 sind die PR-Kurven beider Netzwerke auf NaoTH Validierungsdaten nach einzelnen Trainingsverfahren dargestellt. Diese PR-Kurven wurden aus den besten Ergebnissen in jeweiligen Trainingsverfahren ausgewählt. Obwohl das miniSqueezeNet4 nach dem Training auf dem SPQR Datensatz einen besseren AP Wert auf NaoTH Daten aufweisen konnte, hat das SPQR_d3 Netzwerk bessere maximale AP Werte nach einzelnen Trainingsverfahren, die im Durchschnitt bei 0.96 liegen.

Es wurden folgende Metriken auf jeweils 10 Trainingsdurchläufen gemessen und in Tabellen dargestellt:

- AP NaoTH - Average Precision auf Validierungsausschnitten aus eigenem Datensatz und dem Datensatz aus [14].
- AP SPQR - Average Precision auf dem Validierungsdatesatz des SPQR Teams.
- IOU - die gemessene IOU Metrik (Kapitel 3.5) auf 105 Bildern aus eigenem Datensatz.
- Zeit - Durchschnittszeit für die Evaluierung eines ganzen Bildes.

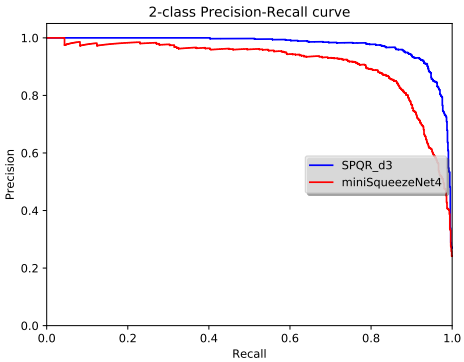
In der Tabelle 1 sind die Ergebnisse des Trainings des SPQR_d3 Netzwerkes auf dem SPQR Datensatz. Der AP SPQR Messwert liegt wie erwartet hoch, da dieser nur auf dem SPQR Validierungsdatensatz gemessen wurde. Der AP NaoTH Wert ist deutlich geringer, das Netzwerk ist nicht in der Lage die Bilder der NaoTH Datensätze richtig zu erkennen. Das miniSqueezeNet4 in Tabelle 2 zeigt ähnliche Ergebnisse auf, nur braucht es 5 Mal weniger Zeit für die Evaluierung eines Bildes.

In den Tabellen 3 und 4 werden die Evaluierungsergebnisse nach dem ganzen Trainingsvorgang in 3 Schritten dargestellt. Die Erkennung auf ”neuen” Bildern ist im Vergleich zum ersten Schritt des Trainings deutlich verbessert geworden. Das SPQR_d3 Netzwerk zeigt bessere Ergebnisse im AP NaoTH Wert als das miniSqueezeNet4 auf.

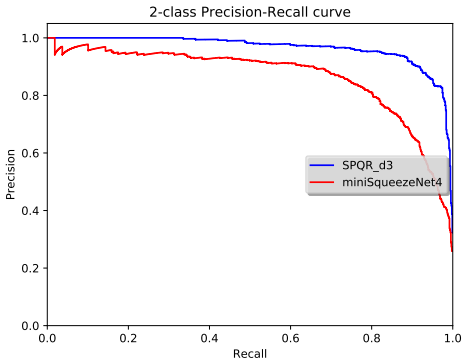


(a) Start

(b) Final



(c) All



(d) Final-Erwartungswerte

Abbildung 12: In diesen Graphen sind die PR-Kurven von beiden Netzwerken auf NaoTH Datensätzen nach unterschiedlichen Trainingsschritten. Start - Training auf SPQR Datensatz. Final - Ablauf von 3 Trainingsschritten. All - Training auf allen Datensätzen ohne Vorschritte. Final-Erwartungswerte - Training mit Jaccard Index Erwartungswerten.

Iter	AP NaoTH	AP SPQR	max IOU	mean IOU	Zeit in ms
1	0.3364	0.9643	0.0000	0.0000	3.2857
2	0.5042	0.8033	0.8120	0.2936	3.2762
3	0.5133	0.9914	0.8621	0.3168	3.2667
4	0.5944	0.9815	0.8081	0.2963	3.1238
5	0.5731	0.9777	0.6050	0.2585	3.1238
6	0.4586	0.9825	0.8168	0.2806	3.1333
7	0.4493	0.9837	0.8120	0.2997	3.2762
8	0.3269	0.9734	0.0000	0.0000	3.1238
9	0.6003	0.9783	0.6870	0.3037	3.1333
10	0.5863	0.9798	0.8120	0.3099	3.1333
Avg	0.4943	0.9616	0.6215	0.2359	3.1876

Tabelle 2: Evaluierungsergebnisse des miniSqueezeNet4 Netzes nach dem Training auf dem SPQR Datensatz (1. Stufe).

Iter	AP NaoTH	AP SPQR	max IOU	mean IOU	Zeit in ms
1	0.9678	0.9708	0.7570	0.4776	14.6857
2	0.9712	0.9682	0.7570	0.4807	14.6952
3	0.9689	0.9839	0.7363	0.4894	15.5333
4	0.9639	0.9777	0.7570	0.4768	14.5143
5	0.9653	0.9740	0.7570	0.4778	15.6190
6	0.9672	0.9564	0.7516	0.4815	15.4952
7	0.9707	0.9804	0.7570	0.4822	15.7143
8	0.9651	0.9747	0.7516	0.4823	15.4952
9	0.9667	0.9700	0.8190	0.4831	15.9143
10	0.9588	0.9659	0.7570	0.4795	15.7524
Avg	0.9666	0.9722	0.7600	0.4811	15.3419

Tabelle 3: Evaluierungsergebnisse des SPQR_d3 Netzes nach der 3. Stufe des Trainings auf binären Erwartungswerten.

Iter	AP NaoTH	AP SPQR	max IOU	mean IOU	Zeit in ms
1	0.6209	0.9556	0.0000	0.0000	3.1238
2	0.7237	0.9631	0.9091	0.4908	3.1238
3	0.8691	0.9965	0.8269	0.4502	3.1333
4	0.8962	0.9717	0.8269	0.5013	3.2381
5	0.8564	0.9735	0.8269	0.4716	3.1429
6	0.8743	0.9832	0.7486	0.4117	3.1714
7	0.8563	0.9894	0.8269	0.4844	3.1238
8	0.7498	0.9716	0.0000	0.0000	3.1238
9	0.8412	0.9769	0.8269	0.4605	3.1238
10	0.8391	0.9832	0.8269	0.4648	3.1238
Avg	0.8127	0.9765	0.6619	0.3735	3.1429

Tabelle 4: Evaluierungsergebnisse des miniSqueezeNet4 Netzes nach der 3. Stufe des Trainings auf binären Erwartungswerten.

Auf Tabellen 5 und 6 werden die Ergebnisse nach dem Training auf dem gesamten Datensatz dargestellt. Das SPQR_d3 Netzwerk zeigt keine deutlichen Unterschiede im Vergleich zum Training in 3 Schritten. Das miniSqueezeNet4 zeigt im Durchschnitt bessere Ergebnisse auf dem NaoTH Datensatz als beim Training in 3 Schritten, jedoch konnte das Netzwerk in 3 von 10 Trainingsvorgängen keine Roboter auf dem eigenen Datensatz erkennen.

Auf Tabellen 7 und 8 werden die Ergebnisse des Trainings mit Erwartungswerten dargestellt. Beide Netzwerke zeigen keine großen Unterschiede in AP Werten im Vergleich zum Training auf binären Werten, jedoch sind die IOU Werte mit diesem Vorgang am höchsten, das bedeutet, dass die Netzwerke besser die Roboter auf den Bildern aus eigenem Datensatz erkennen.

Es ist anzumerken, dass der Trainingsvorgang auf Erwartungswerten im letzten Schritt in fast allen Fällen nach 20 Epochen abgebrochen wurde, obwohl eine Verbesserung des Verlustes auf Validierungsdaten beobachtet werden konnte, die Netzwerke haben damit nicht die bestmöglichen Ergebnisse produziert. Andere Trainingsmethoden wurden in Hälfte der Fälle nach maximal 15 Epochen unterbrochen.

Bei der Evaluierung der Bilder lege ich mehr Wert auf Bilder in den NaoTH Datensätzen, denn diese sollen die Situation simulieren, in der die Netzwerke an eine neue Umgebung angepasst werden müssen. Man möchte auf neuen Bildern gute Ergebnisse erhalten, selbst wenn es auf alten Datensätzen zu Verschlechterung kommt.

In Abb. 13 werden 6 Validierungsbilder nach dem Training im ersten Schritt dargestellt. Die entstandenen Bilder aus dem SPQR Datensatz sehen vielversprechend aus. Das SPQR_d3 hat den liegenden Roboter im 5 Bild erkannt, jedoch hat es ein Stück Hintergrund auf dem letzten Bild auch als einen Roboter anerkannt. Das miniSqueezeNet4 hat dagegen den liegenden Roboter nicht erkannt, dafür aber 2 Roboter auf dem letzten Bild erkannt. Aus den Bildern aus dem NaoTH Labor sehen die Ergebnisse nicht so gut aus. Im ersten Bild wurde sehr viel Hintergrund als Roboter von beiden

Iter	AP NaoTH	AP SPQR	max IOU	mean IOU	Zeit in ms
1	0.9689	0.9789	0.8190	0.4888	15.8095
2	0.9567	0.9610	0.7568	0.4838	15.6381
3	0.9674	0.9905	0.7570	0.4849	15.8762
4	0.9744	0.9858	0.8190	0.4797	15.5524
5	0.9696	0.9828	0.7343	0.4788	15.6571
6	0.9645	0.9892	0.8190	0.4796	15.6000
7	0.9619	0.9807	0.7570	0.4885	15.7619
8	0.9706	0.9731	0.7570	0.4850	15.6571
9	0.9552	0.9790	0.7516	0.4808	15.5524
10	0.9714	0.9887	0.7570	0.4771	16.4476
Avg	0.9661	0.9810	0.7728	0.4827	15.7552

Tabelle 5: Evaluierungsergebnisse des SPQR_d3 Netzes nach dem Training auf allen Datensätzen von Anfang.

Iter	AP NaoTH	AP SPQR	max IOU	mean IOU	Zeit in ms
1	0.8878	0.9763	0.8269	0.4857	3.0857
2	0.8772	0.9816	0.8269	0.4742	3.1333
3	0.9003	0.9915	0.8542	0.4816	3.0952
4	0.8990	0.9758	0.8269	0.4851	3.1238
5	0.8120	0.8963	0.0000	0.0000	4.1714
6	0.7684	0.9805	0.0000	0.0000	3.1238
7	0.8914	0.9827	0.8269	0.4817	3.1810
8	0.8801	0.9829	0.7977	0.4885	3.1238
9	0.7715	0.9774	0.0000	0.0000	3.2762
10	0.9123	0.9656	0.9091	0.4945	3.1333
Avg	0.8600	0.9711	0.5869	0.3391	3.2448

Tabelle 6: Evaluierungsergebnisse des miniSqueezeNet4 Netzes nach dem Training auf allen Datensätzen von Anfang.

Iter	AP NaoTH	AP SPQR	max IOU	mean IOU	Zeit in ms
1	0.9607	0.9801	0.9091	0.5178	15.6095
2	0.9616	0.9779	0.9091	0.5467	15.7524
3	0.9589	0.9875	0.8706	0.5351	16.6381
4	0.9531	0.9851	0.9091	0.5287	15.7333
5	0.9651	0.9827	0.9091	0.5491	15.6667
6	0.9486	0.9708	0.9091	0.5361	17.7905
7	0.9596	0.9801	0.9091	0.5442	15.4476
8	0.9487	0.9740	0.8706	0.5375	16.5905
9	0.9564	0.9417	0.8795	0.5201	15.6952
10	0.9547	0.9691	0.8622	0.5181	15.8476
Avg	0.9567	0.9749	0.8938	0.5334	16.0771

Tabelle 7: Evaluierungsergebnisse des SPQR_d3 Netzes nach der 3. Stufe des Trainings auf Jaccardi Index Erwartungswerten.

Iter	AP NaoTH	AP SPQR	max IOU	mean IOU	Zeit in ms
1	0.6757	0.9727	0.0000	0.0000	3.2762
2	0.5571	0.9214	0.7962	0.3564	2.9810
3	0.8556	0.9968	0.8186	0.4843	3.1238
4	0.8647	0.9801	0.9091	0.4933	3.1810
5	0.8256	0.9773	0.9091	0.4899	3.2762
6	0.8030	0.9764	0.7703	0.4367	3.1238
7	0.8376	0.9855	0.7899	0.4574	3.1238
8	0.7792	0.9666	0.0000	0.0000	3.1810
9	0.8010	0.9821	0.8168	0.4644	3.1619
10	0.8294	0.9866	0.8621	0.4594	3.1238
Avg	0.7829	0.9745	0.6672	0.3642	3.1552

Tabelle 8: Evaluierungsergebnisse des miniSqueezeNet4 Netzes nach der 3. Stufe des Trainings auf Jaccardi Index Erwartungswerten.

Netzwerken erkannt. Bei dem zweiten Bild hat das miniSqueezeNet4 eine bessere Leistung, als das SPQR_d3 gebracht, es hat die Beine des Roboters fast perfekt Ausgeschnitten. Das dritte und vierte Bild sind beide mit extra Hintergrund ausgeschnitten, das miniSqueezeNet4 neigt dazu die Köpfe der Roboter nicht zu erkennen.

In Abb. 14 ist eine deutliche Verbesserung in Erkennung auf dem ersten Bild für beide Netzwerke zu bemerken. Die restliche Bilder zeigen eine deutliche Verschlechterung der Erkennung, dies hängt vielleicht mit einem zu niedrig gewählten Grenzwert von 0.7 zusammen für das SPQR_d3 Netzwerk. Das miniSqueezeNet4 hat in den letzten beiden Bildern keine Roboter erkannt, möglicher weise ist die Generalisierungskapazität des Netzwerks erreicht, da es nur sehr wenige Filter verwendet.

In Abb. 15 kann man erkennen, dass das SPQR_d3 von dem Training am meisten profitiert hat, es klassifiziert weniger Hintergrundteilbilder als "NAO" im Vergleich zu Abb. 13. Das miniSqueezeNet4 weist eine erhöhte Klassifizierung von Hintergrundbildern als Roboter, im Vergleich zu Abb. 13, insbesondere ist die Verschlechterung auf den Bildern 2 bis 4 zu erkennen, wo mehr Hintergrund im Bild erkannt wird. Im zweiten Bild erkennt das miniSqueezeNet4 die Beine des Roboters nicht mehr korrekt.

In Abb. 16 wurden die Netzwerke von Anfang an auf allen Datensätzen trainiert. Das SPQR_d3 zeigt eine Verschlechterung der Erkennung im Vergleich zu Abb. 15, es wird mehr Hintergrund als "NAO" klassifiziert und die Beine werden nicht mehr vollständig erkannt. Das miniSqueezeNet4 zeigt eine bessere Leistung auf dem ersten Bild, jedoch sind die Ergebnisse auf den restlichen Bildern genauso unbrauchbar, wie diese in Abb. 15 waren.

In Abb. 17 wurden die Netze in 3 Schritten trainiert, jedoch wurde für das Training auf den NaoTH Datensätzen keine binären Werte, sondern Erwartungswerte beim Trainieren verwendet. Insbesondere hat davon das SPQR_d3 profitiert, es hat die Roboter auf den neuen Datensatz fast perfekt erkannt. Das miniSqueezeNet4 zeigt auch eine deutliche Verbesserung an, trotzdem erkennt es fälschlicher Weise noch viele Hintergrundteilbilder als "NAO".

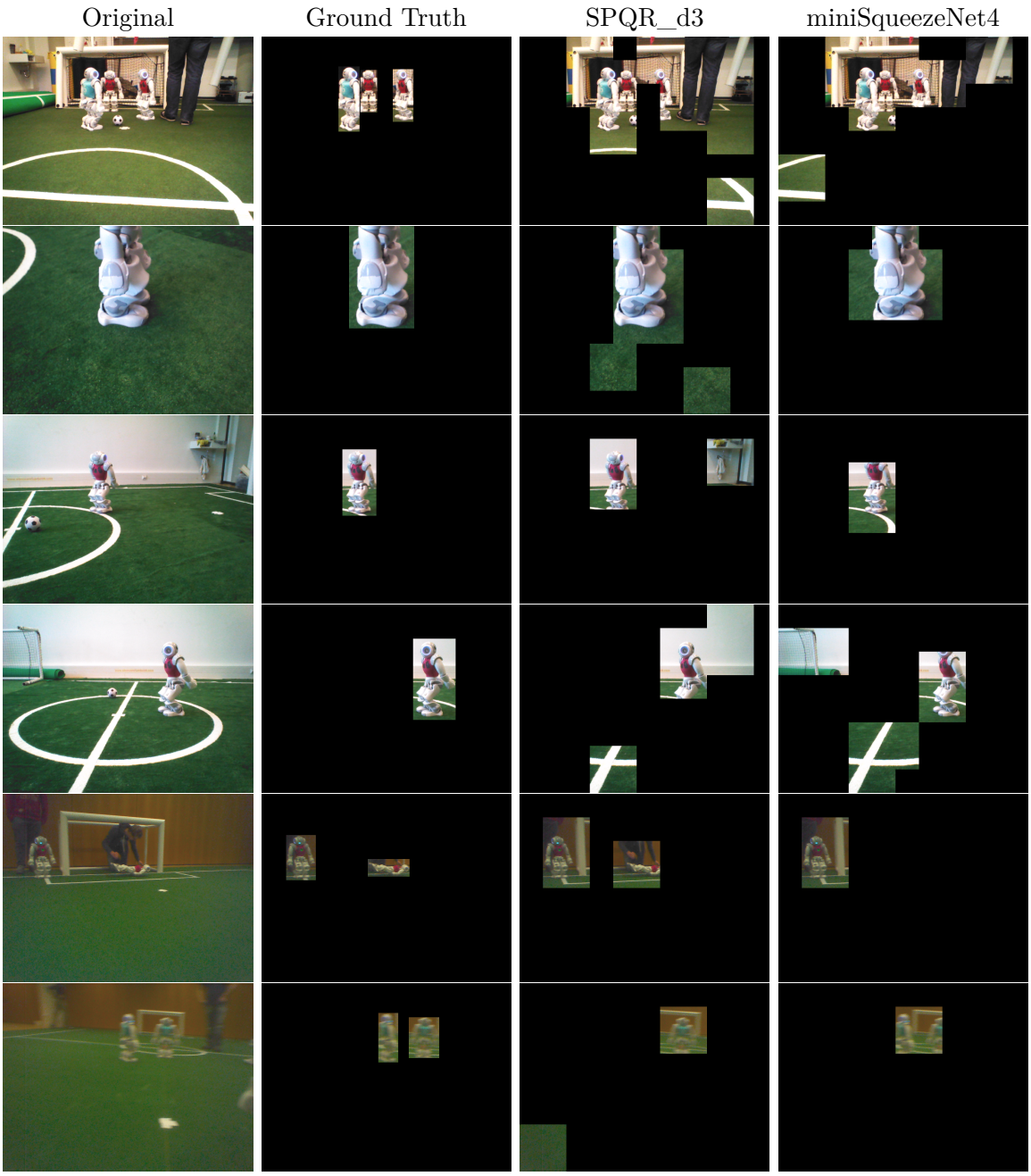


Abbildung 13: Beide Netzwerke wurden nur auf dem SPQR Datensatz trainiert.

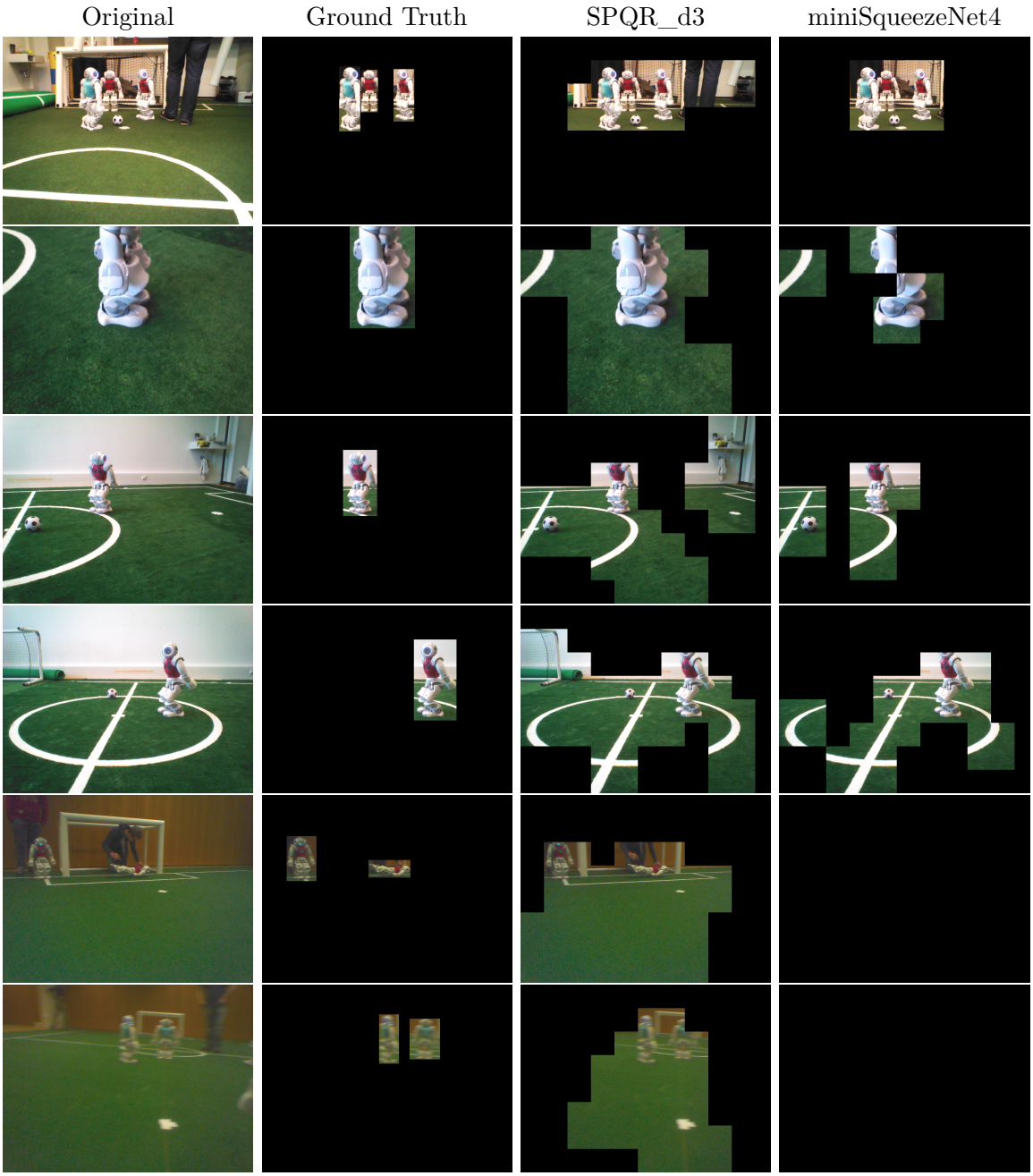


Abbildung 14: Beide Netzwerke wurden mit den Gewichten aus 13 und auf [14] Datensatz trainiert (2. Stufe), mit allen Schichten außer der ersten Faltungsschicht eingefroren.

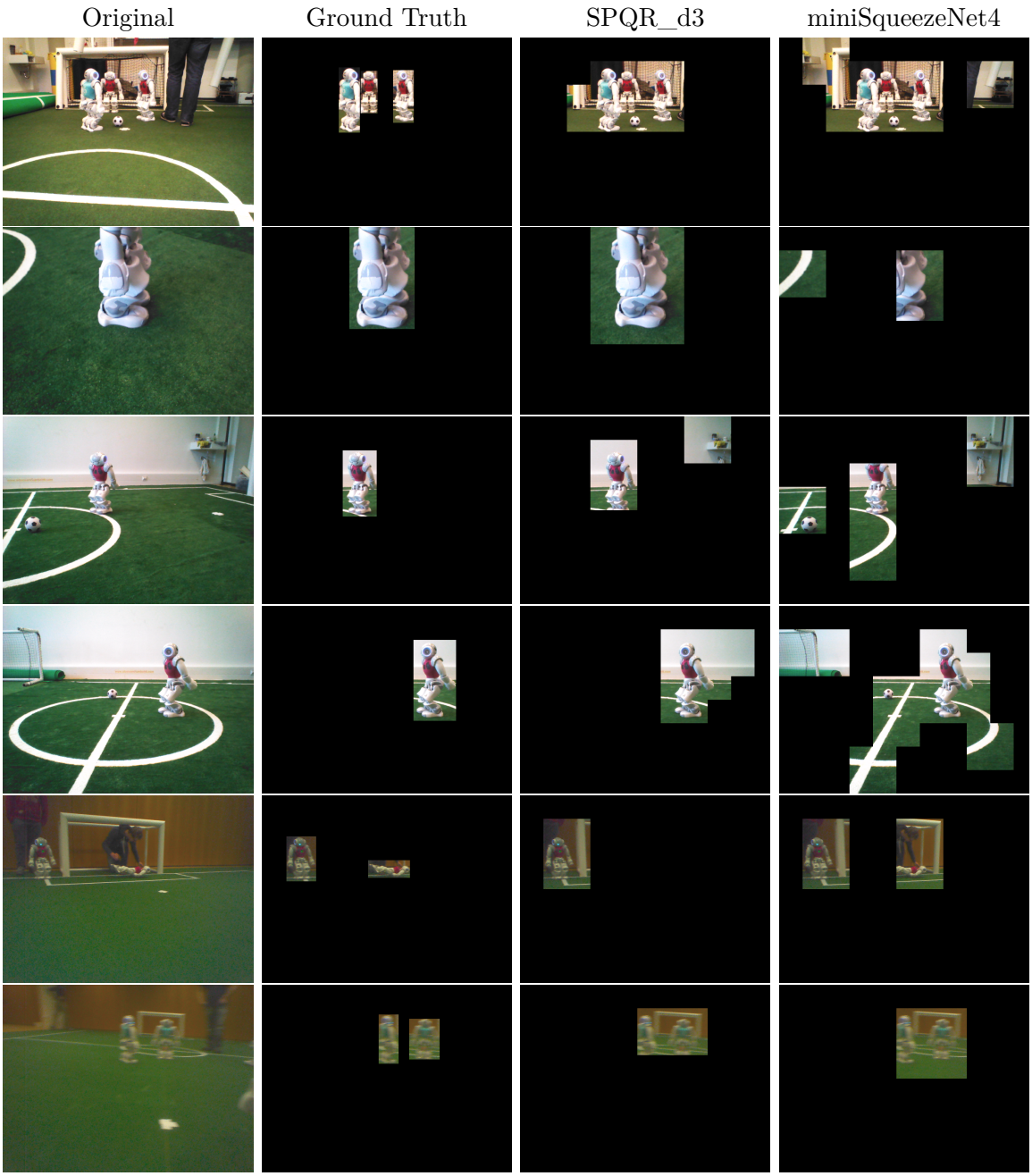


Abbildung 15: Beide Netzwerke wurden mit den Gewichten aus 14 und auf dem SPQR und [14] Datensatz trainiert (3. Stufe).

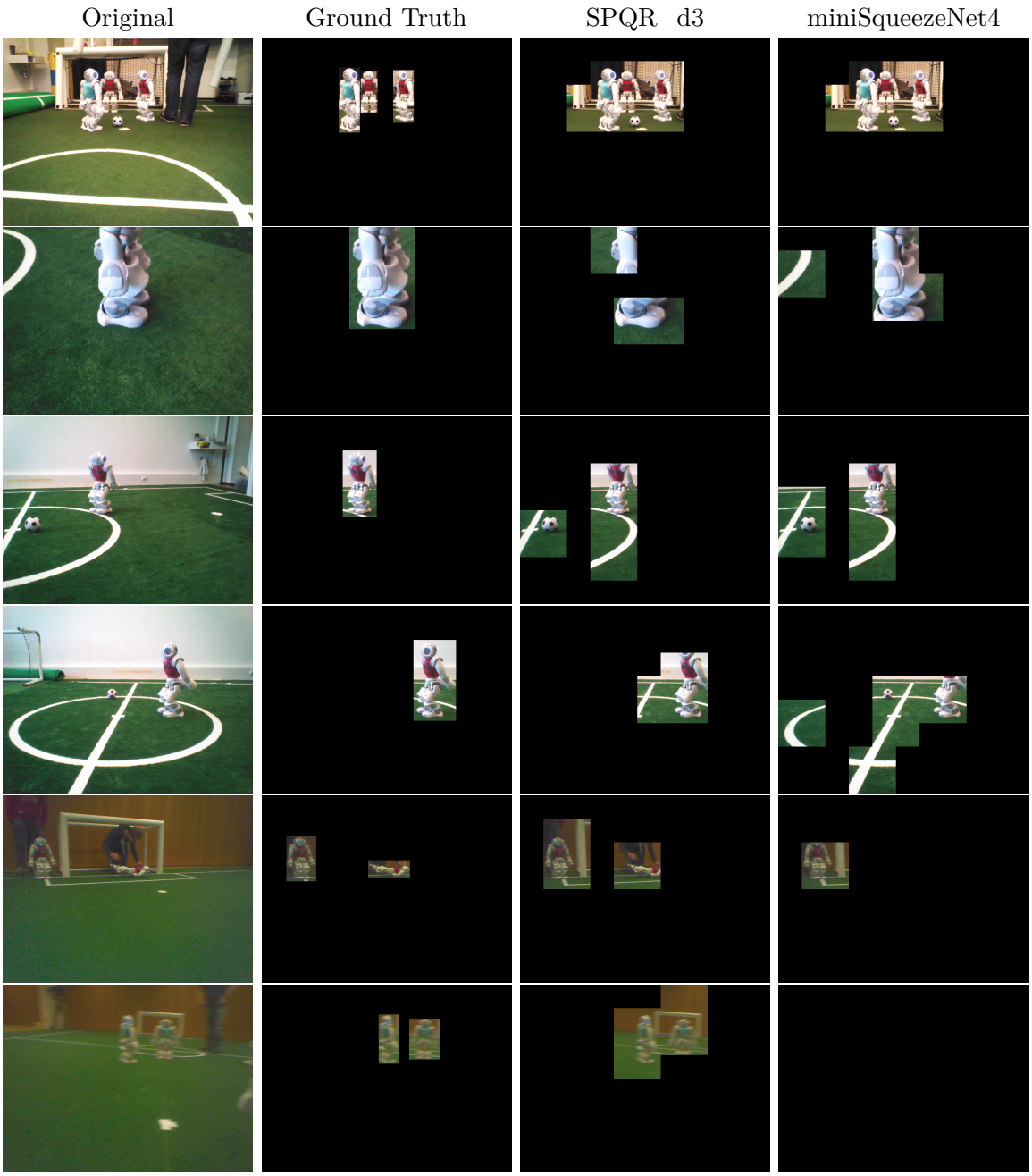


Abbildung 16: Beide Netzwerke wurden auf dem SPQR und den [14] Datensatz trainiert.

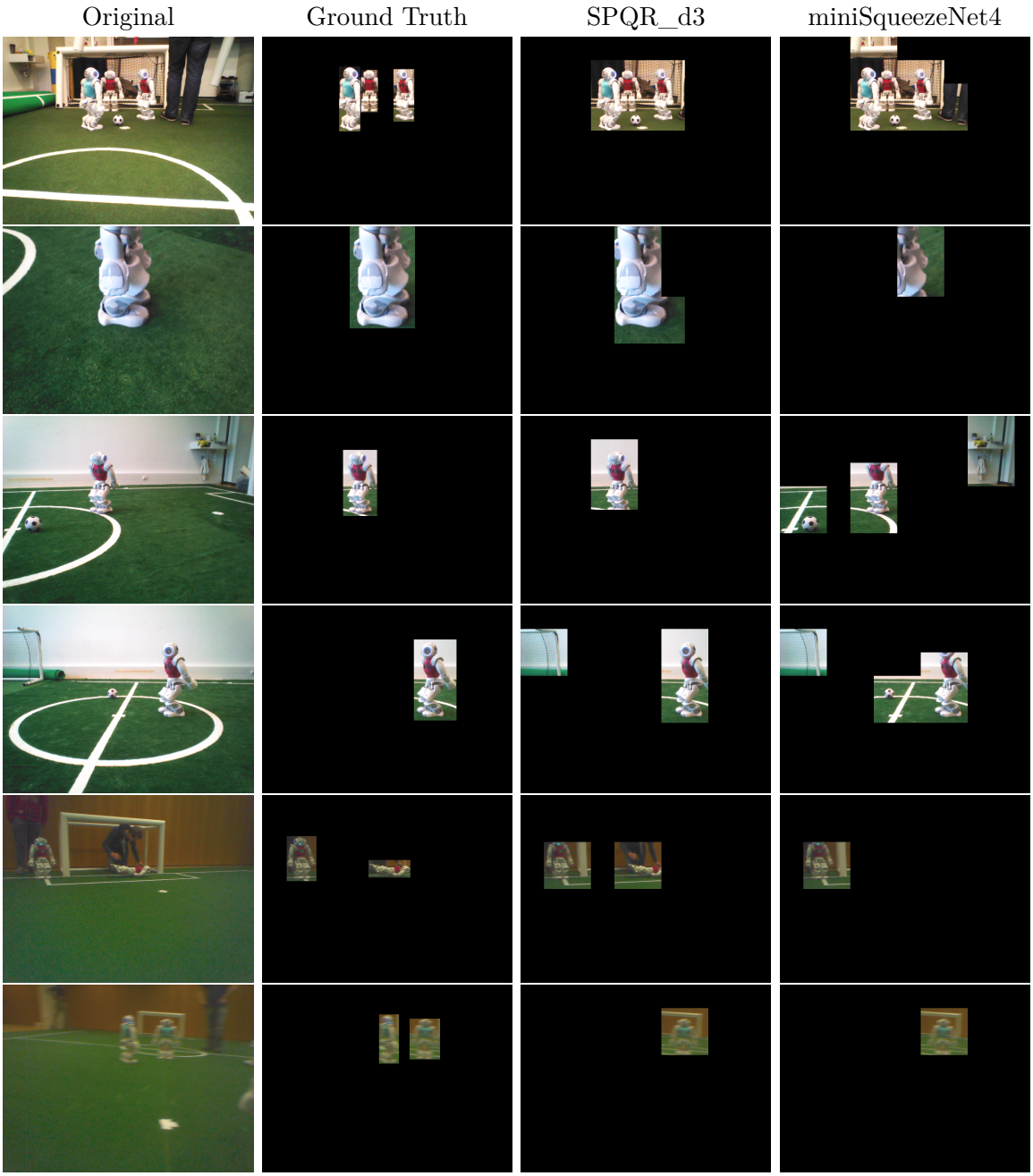


Abbildung 17: Beide Netzwerke wurden auf dem SPQR und den [14] Datensatz trainiert. Für den [14] Datensatz wurden Jaccardi Index Erwartungswerte verwendet (3. Stufe).

5 Zusammenfassung und Diskussion

In dieser Arbeit wurden zwei Faltungsnetzwerke (CNN), das SPQR_d3 und das miniSqueezeNet4, implementiert und empirisch auf einer vorgeschlagenen Trainingsmethode zur Verbesserung der Erkennung im neuen Umfeld evaluiert. Das SPQR_d3 Netzwerk konnte die selben Ergebnisse wie in [5] auf dem SPQR Datensatz aufweisen. Das miniSqueezeNet4 war auf dem SPQR Datensatz besser, als in [8] angegeben.

Im Vergleich zum SPQR_d3, stößt das miniSqueezeNet4 auf die Grenzen des Generalisierungsvermögens, da sehr wenig Faltungsschichten verwendet werden. Selbst wenn das miniSqueezeNet4 ähnliche Ergebnisse wie das SPQR_d3 Netzwerk auf dem SPQR Datensatz aufweisen kann, reicht die Leistung des miniSqueezeNet4 für den NaoTH Datensatz nicht mehr aus. Das SPQR_d3 hat vom Training auf Erwartungswerten eine Verbesserung der Klassifizierung erhalten, insbesondere wurden, wie gewünscht, Anpassungen an neue Umgebung durchgeführt. Das bessere SPQR_d3 Netz kommt aber zum Preis einer 4-5 Mal langsameren Ausführungszeit im Vergleich zu miniSqueezeNet4, so dass es eine erwartete Laufzeit von etwa 240 ms auf dem Nao benötigen wird, das sind nur 4-5 FPS. Das Vergleichsweise gute Generalisierungsvermögen des SPQR_d3 Netzes bedeutet, dass es Raum zur Optimierung gibt, insbesondere könnte die Anzahl der Filter in der Fully Connected Schicht reduziert werden.

5.1 Vergleich zum auf HOG basierten Detektor

Von der Bearbeitungszeit der Bilder ist die auf HD und Full HD angewendete HOG Methode deutlich langsamer als jedes der Netzwerke. Für Full HD braucht die HOG basierte Methode etwa 240 ms auf einem ähnlichen Desktop Rechner. Die Anwengun der HOG Methode auf 100x100 Bilder brauchte aber im Schnitt 3 ms, was der Ausführungszeit des miniSqueezeNet4 ähnlich ist. Die HOG basierte Variante ist vollständiger, indem diese Erkennungen markiert, jedoch ist es unklar, wie diese sich, zum Beispiel, auf dem SPQR Datensatz oder bei einer neuen Umgebung verhalten würde.

5.2 Anwendung auf den Nao Robotern

Für die Anwendung des Verfahrens auf den Nao Robotern muss beachtet werden, dass die verwendeten Kameras der NAO nativ in der YUV422 Kodierung arbeiten, eine Konvertierung ins RGB Format kostet auch Rechenleistung [1], deswegen empfiehlt es sich, die Netzwerke auf der YUV422 Kodierung zu trainieren. YUV422 ist ein Kodierungsschema, dass 4 Bits für die Helligkeit(Luminance - Y) und je 2 Bits für die Farbe (Chrominance - UV) verwendet. Der Y-Kanal wird in einem Byte gespeichert und die UV-Kanäle werden für 2 Pixel in einem anderen Byte gespeichert, damit braucht YUV422 3 Byte um 2 Pixel zu kodieren. Im Vergleich zu den 6 Byte für 2 Pixel im RGB Format, bedeutet das eine Reduktion in Bandbreite von 50%. Natürlich können die YUV422 Bytes mit Hilfe einer Faltungsschicht aufgeteilt werden, indem Faltungen wie Masken behandelt und per Hand gesetzt werden.

5.3 Erweiterung des Verfahrens

Das Training auf markierten Bildern würde es erlauben die Ausschnitte zu Analysieren, indem mehrere Ausgabeschichten zum Netzwerk hinzugefügt werden. Die Konstruktion würde wie folgt aussehen: Das Featurevolumen vor der Klassifikationsschicht, sowie das Ergebnis der Klassifikationsschicht würde an weitere Ausgabeschichten gesendet werden. Das Featurevolumen würde aber vorher mit einer Avg pooling Funktion und Crop in 6 (1 komplette, 3 Horizontal, 2 Vertikal) Boxen (ähnlich SSD) unterteilt werden , so dass 6 Feature Volumen entstehen, die 6 Ausgaben Produzieren, welche eine Aussage über die Position der NAO im Teilbild geben würden. Diese Information könnte verwendet werden, um Regionen mit NAO besser einzugrenzen und die Anzahl an Teilbildern verringern.

5.4 Weiterführende Arbeit

Aus Neugier wurde ein auf SSD300 [17] basiertes vereinfachtes Netzwerk auf markierten Daten trainiert und die Ausführungszeit gemessen. Der Code und die Anleitung können in [2] gefunden werden. Die minimale akzeptable Bildgröße für das Netzwerk war 320x240, da die Pooling, Faltungs und andere für SSD spezifische Schichten die Größe der Feature Maps so verkleinern, dass diese in tieferen Schichten die Größe 0x0 erreichen und damit nicht mehr berechnet werden können. Außerdem hat es eine gemessene Bearbeitungszeit von etwa 200ms auf einem i7-4770K OC 4.2 GHz, was auf dem Nao mehrere Sekunden dauern würde. Ob das Netzwerk für die Verwendung auf den Nao Robotern optimierbar ist, wird im Rahmen dieser Arbeit leider nicht behandelt. Ein Beispiel für die Ausgabe des vereinfachten SSD Netzwerks kann in Abb. 18 angeschaut werden.

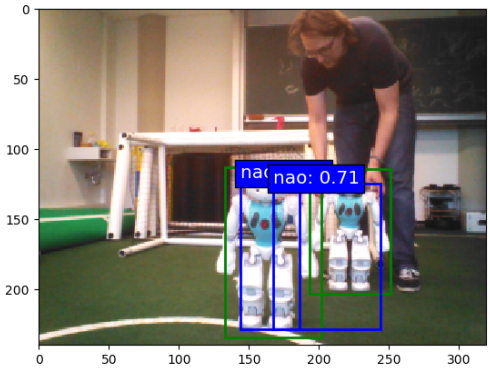


Abbildung 18: Ergebniss eines auf SSD basierenden Netzwerks.

Literatur

[1] Aldebaran documentation. URL: `http://doc.aldebaran.com/2-1/naoqi/vision/alvideodevice.html`, besucht am 07.02.2018.

[2] Auf ssd300 basierende 7 schichtige variation mit anleitung und code. URL: `https://github.com/pierluigiferrari/ssd_keras`, besucht am 07.02.2018.

[3] Imglab. URL: `https://github.com/davisking/dlib/tree/master/tools/imglab`, besucht am 07.02.2018.

[4] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[5] D. Albani, A. Youssef, V. Suriani, D. Nardi, and D.D. Bloisi. A deep learning approach for object recognition with nao soccer robots. Technical report, Department of Computer, Control, and Management Engineering, Sapienza University of Rome, 2016.

[6] Zhuo Chen, Weisi Lin, Shiqi Wang, Long Xu, and Leida Li. Image quality assessment guided deep neural networks training.

[7] François Chollet et al. Keras. `https://github.com/keras-team/keras`, 2015.

[8] Nicolás Cruz, Kenzo Lobos-Tsunekawa, and Javier Ruiz del Solar. Using convolutional neural networks in robots with limited computational resources: Detecting nao robots while playing soccer. Technical report, Advanced Mining Technology Center & Dept. of Elect. Eng., Universidad de Chile, 2017.

[9] Samuel Dodge and Lina Karam. Understanding how image quality affects deep neural networks.

[10] Alexander Fabisch, Tim Laue, and Thomas R¨ofer. Robot recognition and modeling in the robocup standard platform league. In *Proceedings of the Fourth Workshop on Humanoid Soccer Robots in conjunction with the 2009 IEEE-RAS International Conference on Humanoid Robots*, 2010.

[11] Chris Hettinger, Tanner Christensen, Ben Ehlert, Jeffrey Humpherys, Tyler Jarvis, and Sean Wade. Forward thinking: Building and training neural networks one layer at a time.

[12] Forrest N Iandola, Song Han, Moskewicz, Matthew W, Ashraf, Khalid, Dally, William J, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. In *ICLR 2017*, 2016.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[14] Dominik Krienelke. Visuelle detektion humanoider roboter basierend auf histogrammen orientierter gradienten. Master's thesis, Humboldt University of Berlin, 2017.

[15] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector.

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[19] Andre Mühlenbrock and Tim Laue. Vision-based orientation detection of humanoid soccer robots. Technical report, Universit¨at Bremen, Fachbereich 3 – Mathematik und Informatik, 2014.

[20] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542, 2016.

[21] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. Technical report, University of Washington and Allen Institute for AI, 2016.

[22] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, and Microsoft Research. Faster r-cnn: Towards real-time object detection with region proposal networks. Technical report, Microsoft Research, 2016.

[23] Thomas Röfer, Tim Laue, Yannick Bülter, Daniel Krause, Jonas Kuball, Andre Mühlenbrock, Bernd Poppinga, Markus Prinzler, Lukas Post, Enno Roehrig, Rene Schröder, and Felix Thielke. B-human code release 2017. Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany, 2 Universität Bremen, Fachbereich 3, Postfach 330440, 28334 Bremen, Germany, 2017. Kapitel 4.4.

[24] Fabian Schnekenburger, Manuel Scharffenberg, Michael Wülker, Ulrich Hochberg, and Klaus Dorer. Detection and localization of features on a soccer field with feedforward fully convolutional neural networks (fcnn) for the adult-size humanoid robot sweaty. Technical report, Faculty of Mechanical and Process Engineering University of Applied Sciences Offenburg, 2017.

[25] Frank Seide and Amit Agarwal. Cntk: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 2135–2135, New York, NY, USA, 2016. ACM.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition.

[27] RoboCup Technical and Committee. *RoboCup Standard Platform League (NAO) Rule Book*, 2016.

[28] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, den 8. Februar 2018 .....