Dreidimensionale Objektklassifizierung mithilfe der Convolutional Neuronal Networks

Victoria Bibaeva

HAW Hochschule für Angewandte Wissenschaften Hamburg
Hamburg, Deutschland
Email: victoria.bibaeva@haw-hamburg.de

I. Einführung

A. Motivation

Seit vielen Jahrzehnten versucht sich der Mensch, einen Traum zu erfüllen, seine Arbeit von Maschinen erledigen zu lassen. Für einige Aufgaben sind bereits Roboter konzipiert worden, die mit Sensoren und einer künstlichen Intelligenz ausgestattet sind und deswegen autonom mit verschiedenen Objekten der Umgebung interagieren können. Unsere Zukunftsvision basiert darauf, dass solche Roboter zu einem wesentlichen Teil der Industrie sowie des Haushalts sein werden und dabei immer mehr Aufgabengebiete übernehmen. Im Gegensatz zu den Industrierobotern, dessen Umgebung und Position der Objekte eher konstant ist, werden Serviceroboter mit Herausforderungen der ständig ändernden Umgebung konfrontiert.

B. Problemstellung

Um entsprechend handeln zu können, müssen die Roboter ihre Umgebung mit Sensoren erfassen, Objekte lokalisieren und diese erkennen. Soll ein Roboter beispielsweise eine Tasse Kaffee seinem Besitzer holen, so wird er als erstes eine Tasse suchen müssen. Dies erfolgt entweder mithilfe der Objekterkennung, wenn eine bestimmte Tasse gesucht wird, oder mithilfe der Objektklassifizierung, wenn eine beliebige Tasse genügt.

Letztere ist zwar allgemeiner, stellt allerdings viel mehr Herausforderungen dar, weil dabei verschiedene Faktoren ins Spiel kommen, die die Klassifizierung erschweren: Vielfalt der Objekte, diverse mögliche Aussichtspunkte, eventuelle Verdeckung, Einfluss von Umgebung (Licht, Farbe, Hintergrund) usw. Weitere Schwierigkeiten bereiten solche Faktoren wie die Ähnlichkeit zweier Klassen oder große Unterschiede innerhalb einer Klasse.

Im Folgenden werden wir uns mit der Objektklassifizierung im Bereich Servicerobotik beschäftigen. Diese ist nicht nur für Haushaltshilfe oder Hilfe für pflegebedürftige Menschen relevant, sondern findet Anwendung bei Autonavigation, Einparkhilfe, Kameraüberwachung, Tagging von Bilddatenbanken und Markerless Motion Capture.

II. Convolutional Neuronal Networks

Einer der vielversprechendsten Ansätze für Objektklassifizierung nennt sich Convolutional Neuronal Network (dt. "Faltungsnetzwerk"). Dies ist eine Variante von Multilayer Perzeptron mit einer speziellen Architektur, die mit zweidimensionalen Bildern arbeitet. Vorgestellt wurde es von Yann LeCun et al. in 1989 [1], als Inspiration diente ihnen die Sehrinde der Katzen, welche eines der mächtigsten Sehorgane in der Natur besitzen.

Die Idee und erster Versuch, Katzensehrinde für Objektklassifikation als ein künstliches neuronales Netzwerk nachzubilden, stammt von Fukushima und Miyake (1982). LeCun und seine Forschungskollegen haben allerdings die Architektur stark vereinfacht, Back-Propagation zum überwachten Lernen benutzt und dadurch bessere Ergebnisse erzielt. Sie haben viel zur Verbreitung beigetragen und zählen zu den wenigen Experten in diesem Bereich.

LeCun's Systeme für OCR und Handschrifterkennung waren sehr erfolgreich, auch kommerziell — 10 % aller Geldschecks in der USA in den 90er Jahren wurden von seinem System automatisiert gelesen. Google hat Faltungsnetze neuerdings für Gesichts- und Autokennzeichenerkennung bei StreetView-Bildern genutzt, um die Privatsphäre der Menschen zu schützen. Heutzutage dienen diese Netzwerke beispielsweise zur Kundenklassifizierung und -tracking im Supermarkt (nach Alter, Geschlecht), für Videoüberwachung im Flughafen, Handgestenerkennung usw. [2].

Convolutional Networks liefern die besten Ergebnisse bei solchen Problemstellungen wie handschriftliche Zeichenklassifizierung, Gesichtserkennung, Vierbeinererkennung. Was sie besonderes auszeichnet, ist ihre Robustheit, d.h. Unempfindlichkeit gegenüber der Rotation, Translation, Skalierung usw.

A. Netzwerkarchitektur

Wie aus dem Namen hervorgeht, wird in Faltungsnetzwerken eine Technik aus dem Bereich digitale Bildverarbeitung benutzt — die sogenannte "Faltung". Sie eignet sich besonders für Bildglättung, Bildschärfung und als Kantenfilter. Durch diese letzte Eigenschaft hat sich die Faltung auch für die Objektklassifizierung als nützlich erwiesen.

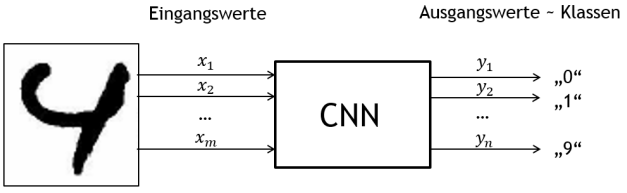

Abbildung 1. Convolutional Neuronal Network (CNN) von außen am Beispiel der Ziffererkennung.

Ein Faltungsoperator erkennt nämlich ein bestimmtes Merkmal (z.B. eine Kante) im Originalbild. Da jedes Objekt aus verschiedenen Merkmalen (Linien, Rechtecke, Kreise usw.) besteht, so ermöglicht die richtige Erkennung dieser

Merkmale demnach die Einordnung des Objektes zu den festgelegten Kategorien.

Objektklassifizierung wird vom Faltungsnetz folgendermaßen durchgeführt, siehe Abbildung 1. Als Input bekommt das Netz ein zweidimensionales Bild, dessen Pixeldaten als Eingangswerte der ersten Neuronenschicht dienen. Nach der Reihe der verdeckten Schichten werden daraus letztendlich die Ausgangswerte berechnet. Die Ausgabeschicht liefert somit eine Antwort auf die Frage, zu welcher der bekannten Klassen das Objekt auf dem Bild gehört. Neuronen jeder Schicht sind nur mit den Neuronen der nächsten Schicht verbunden und bilden dadurch ein Feedforward-Netz.

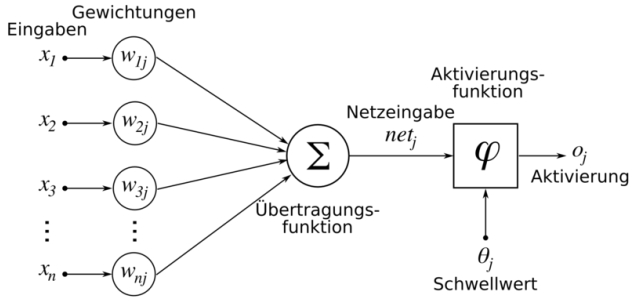

Abbildung 2. Ein Neuron [3].

In Abbildung 2 wird ein allgemeines Neuron dargestellt. Die Eingangswerte $x_1, x_2, \ldots x_n$ (für die Eingabeschicht sind das die Grauwerte des Originalbildes) werden mit den Gewichten $w_{1j}, w_{2j}, \ldots w_{nj}$ des Neurons j multipliziert und zusammenaddiert. Auf das Ergebnis wird eine sogenannte Aktivierungsfunktion angewendet, die am Anfang für alle Neuronen festgelegt wird, um den Ausgangswert o_j zu bestimmen.

Das Neuron kann natürlicherweise als ein Faltungsoperator mit anschließend verwendeter Aktivierungsfunktion angesehen werden. Tatsächlich verfügt ein typischer Faltungsoperator über einen Faltungskern, der auf ein Quellbildabschnitt angewendet wird, um den Zielgrauwert des passenden Zielbildpunktes zu berechnen, siehe Abbildung 3. Der Faltungskern ist eine Zahlenmatrix, deren Elemente mit den Grauwerten dieses Abschnitts pixelweise multipliziert und anschließend addiert werden. Deswegen entspricht der Faltungskern exakt den Gewichten des Neurons.

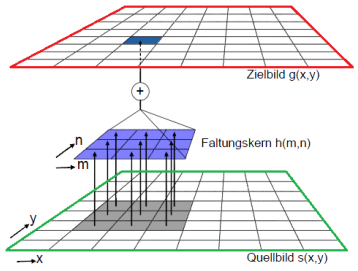

Abbildung 3. Faltung [4].

Um jeden Zielgrauwert mithilfe eines Faltungsoperators zu bestimmen, wandert der Quellbildabschnitt durch das ganze Quellbild. Auf diese Weise entsteht das Zielbild — als Ausgabe einer Schicht der Neuronen mit den jeweils gleichen Gewichten. Das von diesem Faltungsoperator erkannte Merkmal wird im Zielbild hervorgehoben.

Wenn man allerdings mehrere Merkmale im Quellbild erkennen möchte, benötigt man verschiedene Faltungskerne, siehe Abbildung 4. Angewendet auf das Quellbild liefert jeder Faltungskern ein sogenanntes "Feature Map" (dt. Merkmalskarte), das von einer Menge der gleichartigen Neuronen berechnet wird. Alle Feature Maps werden somit von einer Neuronenschicht produziert.

Die Netzwerkarchitektur besteht demzufolge aus mehreren Feature-Phasen, die nacheinander folgende Stufen in der Merkmalerkennung darstellen. Jede Feature-Phase enthält drei Layer:

- *Filter Bank Layer:*

Das Netzwerk lernt hier eine Menge von n Faltungskernen. Diese werden auf die überlappenden Quellbildabschnitte angewendet. Dadurch entstehen n Feature Maps. Ein Feature Map wird von Neuronen berechnet, die dieselbe Parameter (Gewichte und Bias) haben und dazu dienen, auf ein und dasselbe Feature zu reagieren, und zwar unabhängig von der Position im Quellbild. Diese Unabhängigkeit erklärt sich durch Anwendung des Faltungskernes auf überlappende Bildbereiche, was die Merkmalerkennung überall im Bild ermöglicht. Die n Feature Maps bilden das Ergebnis des Filter Bank Layers. Aufgrund der gleichen Gewichte, die zu demselben Faltungskern gehörende Neuronen miteinander teilen, reduziert sich die Anzahl der Netzparameter, die gelernt werden müssen.

- *Non-Linearity Layer:*

Manchmal betrachtet man dieses Layer als Teil des Filter Bank Layers. Hier wird eine nichtlineare Aktivierungsfunktion auf die Ausgabe des Filter Bank Layers punktweise angewendet. Typischerweise ist das eine Sigmoidfunktion (Tangens Hyperbolicus):

$$tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}}$$

Einige Wissenschaftler (siehe z.B. [5]) verwenden auch $f(t) = max(0; t)$, da sie der Überzeugung sind, dass das Training dadurch schneller wird.

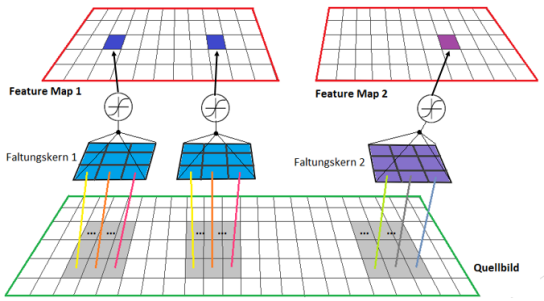

Abbildung 4. Die Entstehung der Feature Maps. Gleiche Gewichte werden mit gleicher Farbe gekennzeichnet.

- *Feature Pooling Layer:*

Auch "Subsampling" genannt. Jedes Feature Map wird nun separat betrachtet. Man berechnet für jeden Pixel den mittleren oder maximalen Wert seiner Nachbarschaft. Das Ergebnis ist ein neues Feature Map mit einer kleineren Auflösung. Auch

wenn Informationen hier verloren gehen, wird das Netzwerk somit robuster gegenüber der kleinen Lageverschiebung des Features.

Nach mehreren Feature-Phasen folgt ein klassischer Multilayer Perzeptron, der den letzten Schritt erledigt, nämlich die Klassifikation selbst. Dort befinden sich einige Full-Connection-Neuronenschichten, bei denen die Neuronen einer Schicht mit den Neuronen aller nachfolgenden Schichten verbunden sind. Die gesamte Netzwerkarchitektur ist in Abbildung 5 zu sehen.

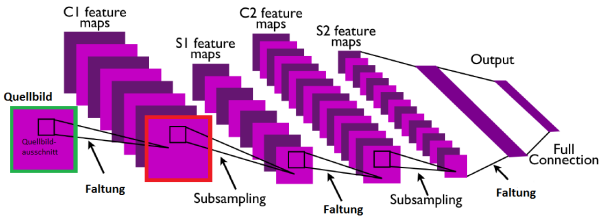

Abbildung 5. Die gesamte Architektur des Faltungsnetzwerkes. Faltung und Subsampling bilden eine Feature-Phase [2].

B. Training

Das Training des Faltungsnetzes erfolgt mithilfe des überwachten Lernens: Es werden Bilder mit der bereits bekannten Zugehörigkeit des Objektes zu einer Klasse geliefert. Die Menge aller Klassen wird im Vorab festgelegt.

Pro Trainingsschritt wird ein Bild als Input an das Netz übergegeben, und die Ausgabe wird mit dem erwarteten Wert verglichen. Nachfolgend werden die Gewichte von der Ausgabeschicht zur Eingabeschicht rückwärts (Backpropagation) mithilfe des Gradientenverfahrens geändert. Dieses Verfahren dient dazu, die Abweichungen zu dem gewünschten Output zu minimieren. Die Gewichte werden nämlich so eingestellt, damit die Differenz zwischen dem gewünschten und tatsächlichen Ausgangswert über alle Trainingsbilder minimal sein soll. Ausgehend von aktuellen Gewichten wird dafür ein kleiner Schritt in Richtung des steilsten Abstiegs (negatives Gradient) der Fehlerfunktion vorgenommen.

Im Gegensatz zu einer einzigen Aktualisierung der Faltungskerne aller Schichten am Ende des Trainings (z.B. indem man einen Mittelwert ausrechnet), bringt dieses schrittweise Training eindeutige Vorteile mit sich, und zwar schnellere Konvergenz, besonderes an großen und redundanten Datenmengen, und Robustheit [1].

In Abbildung 6 sind die gelernten Faltungskerne der ersten Feature-Phase dargestellt. Daran kann man erkennen, dass die kleinsten Merkmale, die das Netz erkennt, linienförmig sind und verschiedene Farbverläufe und Richtungen aufzeigen.

C. Vergleich zu MLP

Was unterscheidet Faltungsnetzwerke von einem allgemeinen Multilayer Perzeptron? Letzterer beherrscht Funktionsapproximation, besitzt allerdings keine Abstraktionsfähigkeit. Demnach wäre es nicht möglich, dass ein Perzeptron, der mit normalen Zeichen trainiert wurde, ein umgedrehtes Zeichen aus dieser Zeichenmenge erkennt. Ohne Preprocessing wird er ebenfalls nicht in der Lage sein, ein skaliertes oder verschobenes Zeichen richtig zu klassifizieren, ganz zu schweigen

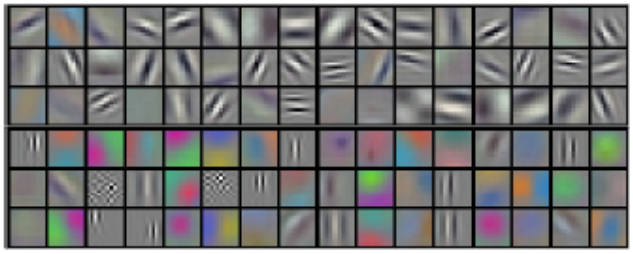

Abbildung 6. Faltungskerne der ersten Feature-Phase [5].

von der Merkmalextraktion, um z.B. die Hunderassen zu unterscheiden.

All diese Aufgabestellungen werden vom Faltungsnetz dagegen erfolgreich erledigt. Nicht nur die Gesichtserkennung und Handgestenerkennung, sondern auch Vierbeinererkennung inklusive Rasse, Gattung oder Art werden von Faltungsnetzen so durchgeführt, dass manch ein Mensch ihre Performanz nicht übertreffen kann. Dank der Robustheit bzgl. geometrischen Transformationen wird die Menge der Trainingsdaten und somit auch die Trainingszeit stark reduziert.

III. Verwandte wissenschaftliche Arbeiten

A. Performanz der Faltungsnetze

Im Bereich Objekterkennung finden regelmäßig diverse Wettbewerbe statt. Einer der namhaftesten nennt sich ILSVRC (ImageNet Large Scale Visual Recognition Challenge, [6]), wird seit 2010 jährlich gehalten und verfügt über das eigene gleichnamige Dataset, das bereits zum Standard-Benchmark geworden ist. Dieses Dataset dient als Grundlage, die bestehenden Algorithmen miteinander zu vergleichen und ihre Performanz zu messen. Im Wettbewerb muss eine der folgenden **Aufgaben** gelöst weden [7], Schwierigkeitsgrad steigend:

1) Objektklassifizierung: Der teilnehmende Algorithmus soll für jedes Bild eine Liste der Objektklassen liefern, die im Bild vorhanden sind;
2) Lokalisierung eines Objektes: Neben der Liste von Objektklassen muss pro Klasse je ein Bounding Box eines Objektes produziert werden;
3) Lokalisierung aller Objekte: Hier müssen alle Objektklassen und alle Bounding Boxes geliefert werden.

Das ILSVRC Dataset besteht aus ca. 1,4 Millionen Bilder und 1000 Kategorien und ist auf den gleichen Prinzipien aufgebaut wie ein anderes Standard-Dataset namens PASCAL VOC [8], das allerdings deutlich kleiner ist.

Das größte bekannte Dataset für Objektklassifizierung ist ImageNet mit ca. 14 Millionen Bildern und mehr als 21.000 Kategorien. In jedem Bild können mehrere Objektkategorien vorkommen, die samt Bounding Boxes aller Objekte von Menschen manuell annotiert sein müssen. Diese Kategorien basieren auf der WordNet Hierarchie (für Details siehe [7]): Jede Kategorie entspricht einem ”Synset”, also einer Menge der Synonyme eines englischen Wortes. Dadurch wird eine Zweideutigkeit der Interpretation für die Bearbeiter vermieden.

Während ImageNet stets weiter wächst und eine noch zu hohe Herausforderung für die Klassifizierungsverfahren darstellt, eignet sich ILSVRC besser als Large Scale Benchmark. Die wichtigsten Kennzahlen der drei Datasets sind in der Tabelle I zusammengefasst.

TABELLE I. KENNZAHLEN DER DREI WICHTIGSTEN DATASETS FÜR DIE OBJEKTKLASSIFIZIERUNG.

Dataset	*Anzahl Klassen*	*Anzahl Bilder*	*Bilder je Klasse, ø*
PASCAL VOC	20	21.738	1.087
ILSVRC	1.000	1.431.167	1.431
ImageNet	21.841	14.197.122	650

Jährlich werden für den Wettbewerb 1000 Kategorien aus den in ImageNet bekannten Synsets ausgewählt, sodass keine Kategorie mit einer anderen überlappend oder hierarchisch abhängig ist. Die dazugehörigen Bilder aus ImageNet dienen für die Teilnehmer als Trainingsset, wobei das Testset bestehend aus 100.000 Bilder mithilfe der Suchmaschinen und Crowdsourcing gesammelt wird. Die Genauigkeit der manuellen Annotation beträgt angeblich 99,7% [7].

Jeder bei der Objektklassifizierung teilnehmende Algorithmus soll pro Testbild bis zu fünf Klassen liefern, die im Bild vorhanden sind. Von diesen fünf soll eine einzige Klasse identifiziert werden, die mit der größten Wahrscheinlichkeit im Bild vorliegt. Der Grund dafür ist, dass jedes Bild zu genau einer Klasse manuell zugeordnet wird, obwohl dort eventuell mehrere Klassen existieren. Um die Ergebnisse der Verfahren auszuwerten, werden daher verschiedene Metriken benutzt [7]:

- "Top-1"-Metrik: Wenn die Klasse X mit der größten Wahrscheinlichkeit nicht der tatsächlichen Klasse Y entspricht, wird dies als Fehler bezeichnet.
- "Top-5"-Metrik: Wenn keine der fünf gelieferten Klassen der tatsächlichen Klasse entspricht, wird dies als Fehler bezeichnet.
- Hierarchische Metrik: wie "Top-1", aber gewichtet. D.h. je weiter die gemeinsame Elternklasse von X und Y in der WordNet Hierarchie steht, desto schwerwiegender der Fehler ist. So wird z.B. eine Fehlerkennung von Hund härter bestraft, wenn der Algorithmus ein Auto zurückgibt, als wenn er zwei Hunderassen verwechselt.

In der Praxis erweist sich die Reihenfolge der Gewinner jedoch als unabhängig von der verwendeten Metrik. Die drei besten Ergebnisse sind für alle drei Aufgaben in der Tabelle II erfasst, die vollständige Aufstellung kann der Quelle [7] entnommen werden. Die richtige Lokalisierung aller Objekte in der Aufgabe 3 wird anhand eines anderen Kriteriums gemessen, und zwar anhand der Genauigkeit bei der Erkennung aller Klassen und zugehörigen Bounding Boxes. Dabei werden fehlende und doppelte Bounding Boxes sowie falsch erkannte Objekte bestraft.

In den ersten zwei Jahren des Wettbewerbs zeigten die auf Support Vector Machines (Team NEC [9]) und Fisher Kernel (Team XRCE [10]) basierende Verfahren die kleinsten Fehlerraten (jeweils 28,2 % und 25,8 %).

Der Wendepunkt für Objektklassifikation kam in 2012, als Faltungsnetzwerke zum ersten Mal im Wettbewerb auftraten und das Ergebnis von ihren Vorgängern mit 16,4 % Fehler fast halbierten (Team SuperVision [5]). Auch bei der Lokalisierung eines Objektes (Aufgabe 2) erzielten sie einen Performanzsprung auf eine Fehlerrate von 34,2 %. Andere Teams nutzten weiterhin Support Vector Machines oder Fisher Kernels, waren allerdings beinahe doppelt so schlecht.

TABELLE II. DIE GEWINNER DES ILSVRC-WETTBEWERBS NACH [7]. BEI DER OBJEKTKLASSIFIZIERUNG UND LOKALISIERUNG EINES OBJEKTES (AUFGABEN 1 UND 2) WIRD DIES ANHAND DER FEHLERRATE GEMESSEN, BEI DER LOKALISIERUNG ALLER OBJEKTE (AUFGABE 3) ANHAND DER ERKENNUNGSGENAUIGKEIT. ALLE ANGABEN SIND IN PROZENT. DIE HERVORGEHOBENEN ZAHLEN BEZIEHEN SICH AUF DAS BESTE RESULTAT DER JEWEILIGEN AUFGABE.

Jahr	*Team*	*Aufgabe 1*	*Aufgabe 2*	*Aufgabe 3*
2010	NEC	**28,2**	-	-
	XRCE	33,6	-	-
	ISIL	44,6	-	-
2011	XRCE	**25,8**	56,5	-
	UvA	31,0	**42,5**	-
	ISI	36,0	-	-
2012	SuperVision	**16,4**	**34,2**	-
	ISI	26,2	53,6	-
	VGG	27,0	50,0	-
2013	Clarifai	**11,7**	-	-
	NUS	13,0	-	-
	ZF	13,5	-	-
	...	...	-	-
	OverFeat	14,2	**30,0**	19,4
	UvA	14,3	-	**22,6**
	VGG	15,2	46,4	-
	NEC	-	-	19,6
2014	GoogLeNet	**6,7**	26,4	-
	VGG	7,3	**25,3**	-
	MSRA	8,1	35,5	35,1
	SYSU	14,4	31,9	-
	UvA	12,1	>60,0	32,0
	NUS	-	-	**37,2**

Im nachfolgenden Jahr präsentierte die Mehrheit der Teams ihre Verfahren, die die Faltungsnetze auf die eine oder andere Weise verwendeten (siehe Teams NUS, ZF in der Tabelle II). Das siegende Team Clarifai [11] schnitt mit 11,7 % der Fehler am besten ab. Dabei haben die neue Visualisierungstechnik (siehe [12] und Abschnitt III.B) und die GPU-Programmierung geholfen, die optimalste Netzwerkarchitektur auszuwählen und das Netz direkt auf GPU zu trainieren, was die Performanz dementsprechend steigerte.

Aus 36 teilnehmenden Teams im Jahr 2014 haben fast alle ihre Verfahren mit den Faltungsnetzen implementiert, einschließlich der Sieger in allen Aufgaben. In der Aufgabe 1 gewann das Team GoogLeNet, ihre Fehlerrate 6,7 % zeigt die vierfache Verbesserung seit 2010 auf. Experimente mit der Faltungsnetzarchitektur und effiziente Implementierung [13] brachten Team VGG auf Platz 1 in der Aufgabe 2. Ihr Netz enthielt drei unterschiedliche Architekturen und 19 Neuronenschichten mit der Aktivierungsfunktion $f(t) = max(0; t)$ (siehe Abschnitt I.A).

Zum ersten Mal dürften die Teilnehmer auch die externen Daten (außerhalb des vorgegebenen Trainingssets) benutzen. Da man das Resultat solcher Teams mit den anderen nicht direkt vergleichen konnte, wurden die drei Aufgeben demgemäß aufgeteilt. Team GoogLeNet erreichte mithilfe der externen Daten die sagenhafte 43,90 % Erkennungsgenauigkeit in der Aufgabe 3, was das Ergebnis vom Sieger-Team NUS (37,20 %, [14]) deutlich übertraf. Die Nutzung der externen Daten in den Aufgaben 1 und 2 brachte jedoch keine Performanzverbesserungen.

Die Analyse der Resultate [7] ermöglicht uns, nach den Fehlerquellen der Algorithmen zu suchen und dadurch die Fehlerquote gezielter zu senken. Bei der Objektklassifikation wurden insgesamt 121 Kategorien zu 100 % richtig erkannt, einige von diesen sind im oberen Teil der Abbildung 7

zu sehen. Die kleinste Trefferquote (in Klammer angezeigt) haben dagegen solchen Klassen wie Wasserflasche (durchsichtig), Haken (metallisch), Velvet (Textur), abwechslungsreiche Szenen wie Restaurant usw., siehe Abbildung 7 unten.

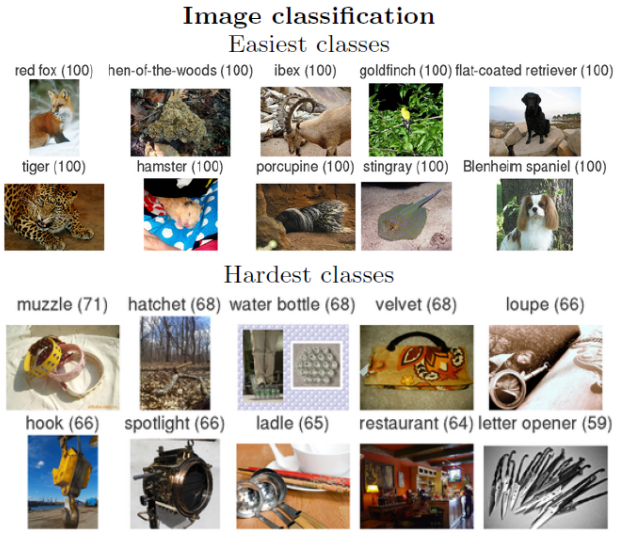

Abbildung 7. Beispiele der leichtesten und schwierigsten Kategorien bei der Objektklassifizierung [7]. In Klammern steht optimistische Trefferquote in %.

Des Weiteren besteht die Möglichkeit, die besten computergestützten Resultate (also vom Team GoogLeNet) mit denjenigen des Menschen zu verglichen. Dazu wurde ein Experiment durchgeführt, im Rahmen dessen die Menschen, die die Testdaten für ILSVRC manuell annotiert haben, analog zum Faltungsnetz für die Aufgabe 1 trainiert wurden [7]. Der Mensch bekam 500 Trainingsbilder und 1500 Testbilder, anschließend wurden seine Ergebnisse mit den geschätzten Ergebnissen des GoogLeNets auf dem gleichen Testset verglichen. Das künstliche Netzwerk zeigte die Fehlerrate 6,8 %, während der Mensch sich lediglich bei 5,1 % der Bilder irrte.

Beide Teilnehmer des Experiments — der Mensch und das Netzwerk — machten gemeinsame Fehler bei mehr als 5 Klassen im Bild, wenn keins der Objekte im eindeutigen Fokus stand, und bei falscher ursprünglicher Annotation der Testbilder. Das Netzwerk war mehr fehleranfällig als der Mensch, wenn die abgebildeten Objekte klein oder dünn waren (21 % aller Fehler), mit einem Filter verzerrt wurden (13 % der Fehler) oder eine abstrakte Form darstellten (z.B. Bildnis, Statue, Spielzeug usw.). Weitere Fehlerquellen sind Nahaufnahmen, unkonventionelle Blickwinkel, Unfähigkeit, einen Text zu verstehen (wenn es im Bild darauf ankommt), massive Verdeckungen usw.

Andererseits gibt es auch Fehler, die der Mensch öfter begeht als das Faltungsnetzwerk. Demnach unterscheiden die Menschen feingranulare Kategorien bedeutend schlechter als das Netzwerk: Bei der Kategorisierung der 120 Hunderassen lag ihre Fehlerquote nämlich bei 37 % im Gegensatz zu 7%. Auch mangelhafte Kenntnis aller ILSVRC Klassen und gewöhnliche falsche Klassifizierung verhindern den Menschen ihren Erfolg.

Im Laufe der weiteren Experimente wurde außerdem folgendes festgestellt: Je weniger Trainingsbilder der Mensch bekommt, desto schlechter seine Trefferquote im Vergleich zum Netzwerk ist [7].

All diese Erkenntnisse sind bemerkenswert: Auch wenn der Mensch eine herausragende Gehirnstruktur besitzt und seit seinem Geburt Jahrzehnte lang lernt, macht er nicht viel weniger Fehler bei der Objektklassifizierung als ein Faltungsnetzwerk, das auf einem um Vielfaches kleinerem Trainingsset und maximal im Laufe der Woche geübt hatte.

B. Visualisierung der Faltungsnetze

Anhand des ILSVRC-Wettbewerbs ist leicht zu sehen, dass die Faltungsnetzwerke verschiedene Aufgaben im Bereich Objektklassifikation sehr gut bewältigen. Dazu tragen solche Faktoren bei wie Verfügbarkeit der umfangreichen Datasets, leistungsfähige und programmierbare GPUs und Regularisierung der vorhandenen Netzmodelle durch Dropout-Technik [15].

Dennoch fehlt den Wissenschaftlern noch das Verständnis, warum Faltungsnetze so viel leisten und wie diese Leistung verbessert werden kann. Einer der neusten Einsätze, der versucht, beide Fragen zu beantworten, wurde in [12] präsentiert. Dies ist eine Visualisierungstechnik, die einen Einblick in die Funktion der mittleren Feature-Phasen und der letzten Full-Connection-Schichten (also des Klassifikators) ermöglicht. Sie basiert auf sogenannten Deconvolutional Neuronal Networks (”Deconvnetze”), die aus denselben Komponenten wie die Faltungsnetze bestehen, nur umgekehrt: Anstatt Pixel auf die Features abzubilden, tun sie das Gegenteil.

Zu jeder Feature-Phase wird ein Deconvnetz-Layer angeschlossen, das aus dem Ergebnis der Phase (also aus dem jeweiligen Feature Map) die Pixel zurückgibt, die es aktiviert haben. Im Deconvnetz-Layer wird das Feature Map mittels einer bestimmten Prozedur ”unpooled”, an die Aktivierungsfunktion $f(t) = max(0; t)$ weitergegeben, um anschließend den transponierten Faltungskern der Phase darauf anzuwenden. Die Outputs des Deconvnetz-Layers am Beispiel der dritten Feature-Phase sowie die entsprechenden Quellbildabschnitte sind in der Abbildung 8 dargestellt.

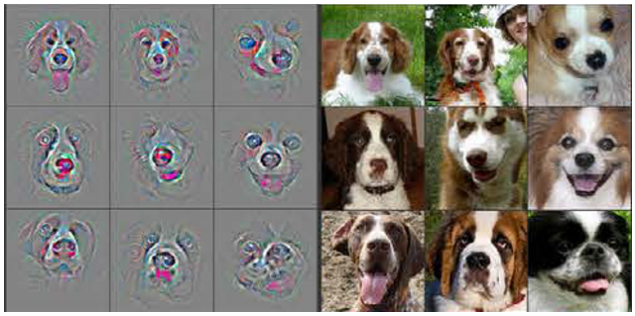

Abbildung 8. Ergebnisse der Visualisierung [12]. Links sind die rekonstruierten Muster, die die höchste Aktivierung im Feature Map der dritten Feature-Phase verursachen. Rechts sind die Bildabschnitte, die dieses Feature Map erzeugten.

Diese Visualisierungsmethode lässt einem die Evolution eines Features während des Trainings betrachten und mögliche Probleme im Modell diagnostizieren. Ausgehend vom Faltungsnetzmodell aus [5] wird dessen Performanz auf ILSRVC mithilfe der Deconvnetze um ein Drittel verbessert.

Das ursprüngliche Modell enthielt 5 Feature-Phasen gefolgt von einem dreilagigen Klassifikator. Die erste Feature-Phase

verwendete Faltungskerne der Größe 11x11 mit dem Schritt 4, die zweite Phase – Faltungskerne 5x5 mit dem Schritt 2, weitere Phasen haben lediglich Größe 3x3 mit dem Schritt 1. Die Pooling-Maske ist stets 3x3 mit dem Schritt 2 Pixel; alle Schritteingaben beziehen sich auf beide Achsen. Jede Schicht des Klassifikators besteht aus 4096 Neuronen, mit 1000 Neuronen in der Ausgabeschicht.

Nach der Visualisierung der Ergebnisse wurde deutlich [12], dass die Faltungskerne der ersten Feature-Phase viele hohe und niedrige, und kaum mittlere Signalfrequenzen enthalten. Dies führt dazu, dass einige Features in späteren Phasen zu verschwommen wirken und dadurch nicht erkannt werden, andere sind zu dominant, was die Kategorisierung beschränkt. Wenn man jedoch die Faltungskerngröße der ersten Phase auf 7x7 und den Schritt auf 2 heruntersetzt, bekommt man eine ausgewogene Frequenzverteilung und vermeidet Aliasing der Features.

Tabelle III präsentiert die Ergebnisse der Auswertung unterschiedlicher Netzwerkarchitekturen. Als Bewertungskriterien dienen die Metriken "Top-1" und "Top-5" aus dem Abschnitt III.A. Demnach verbessert sich die Leistung des vorgeschlagenen Modells A um 2,3 % bzw. 1,7 % gegenüber dem aus [5] stammenden Modell B.

TABELLE III. EXPERIMENTE MIT VERSCHIEDENEN NETZWERKARCHITEKTUREN. DIE ZAHLEN STEHEN FÜR FEHLERRATE IN PROZENT, ANGELEHNT AN [8].

Netzarchitektur	*Top-1*	*Top-5*
Modell aus [5] = *A*	40,7	18,2
A ohne mittleren Schichten	45,4	22,1
A ohne letzten Schichten	44,8	22,4
A ohne mittleren und letzten Schichten	71,3	50,1
Verbessertes Modell = *B*	38,4	16,5
5 Netze des Typs *B*	36,7	15,3
B mit erweiterten mittleren Schichten = *C*	37,5	16,0
B und *C* kombiniert	**36,0**	**14,7**
B mit erweiterten letzten Schichten	38,8	17,0
C mit erweiterten letzten Schichten	38,3	16,9

Weitere Experimente mit den Netzwerkarchitekturen ergaben [12], dass Löschung der Schichten unterschiedliche Auswirkungen haben kann. So führen die Löschung der mittleren Schichten oder der Verzicht auf den Klassifikator (obwohl er die meisten Netzwerkparameter enthält) lediglich zu einer geringen Verschlechterung. Beide verursachen dennoch einen drastischen Rückgang der Leistung um ca. 31 %. Das bestätigt die Existenz einer gewissen Gesamttiefe, die das Netzwerk im Idealfall haben müsste.

Das Hinzufügen zusätzlicher Neuronen in die letzten Schichten bewirkt keine große Leistungsschwankung. Erweiterung der mittleren Schichten im Gegenteil erweist sich als nützlich in der Kombination mit den anderen Modellen und liefert somit die besten Ergebnisse mit bis zu 14,7 % Fehlerquote. Werden Neuronen in all diese Schichten hinzugefügt, kommt das Overfitting vor: Das Trainingsset wird so gut gelernt, dass abweichende Bilder noch mehr Fehler bewirken.

Es wurde außerdem eine Sensitivitätsanalyse durchgeführt, um festzustellen, welche Bildbereiche für eine richtige Klassifizierung zuständig sind. Damit kann man in der ersten Linie prüfen, ob das Netz die Lage des Objektes wirklich identifiziert oder einfach den Umgebungskontext nutzt. Diese Analyse geht folgendermaßen vor: Verschiedene Bildabschnitte werden systematisch mit einem grauen Quadrat verdeckt. Wie erwartet, sinkt die Wahrscheinlichkeit der richtigen Erkennung deutlich, je mehr das Objekt verdeckt ist.

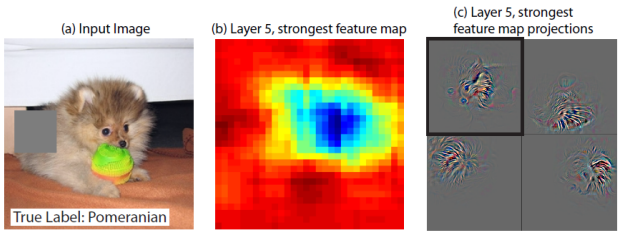

Abbildung 9. Ergebnisse der Sensitivitätsanalyse [12]. Links befindet sich das Originalbild mit einer Verdeckung. In der Mitte – das am stärksten aktivierte Feature Map der letzten Feature-Phase. Blaue Zonen stehen für niedrige, rote – für hohe Aktivierungswerte. Rechts sind die Outputs der Deconvnetz-Schichten.

In der Abbildung 9 wird ein starker Rückgang in der Aktivität des Feature Maps sichtbar, wenn ein Bildbereich aus der Visualisierung verdeckt wird. Dies deutet darauf hin, dass die Visualisierung wirklich wichtige Bildbereiche herauskristallisiert, die mit der richtigen Objektklassifizierung eng verknüpft sind.

IV. EIGENE LÖSUNGSANSÄTZE

Unser Anwendungsfall der Faltungsnetze besteht darin, einen Roboter mit dem von uns entworfenen Faltungsnetz zu trainieren, damit er anhand der 2D-Bilder und Tiefenbilder in Echtzeit bestimmte 3D-Objekte im Sichtfeld der Kamera (z.B. eine Tasse) klassifizieren kann. Dies wird für solche Anwendungsgebiete wie Servicerobotik interessant, aber auch für Obst- und Gemüseernte mit dem Einsatz der Roboter.

Wir sind der Überzeugung, dass das Verwenden der 3D-Information für Objektklassifizierung erhebliche Performanzvorteile mit sich bringt, weil das auch für das menschliche Auge gilt.

V. ZIELE UND ARBEITSPLAN FÜR PROJEKT 1

Im Rahmen des Projekts 1 werden wir uns den Stand der Technik im Bereich Objektklassifizierung insbesondere mit Faltungsnetzen aneignen und die Grundlagen für die nachfolgende Masterarbeit verschaffen. Nach dem Kennenlernen der Trainingsumgebung wird die erste Implementierung mit vorhandenen Trainingsdaten (PASCAL VOC, ImageNet) folgen. Es werden zunächst Kenntnisse gesammelt, wo die Fehlergrenze liegt und welche Konfigurationen zum besten Ergebnis führen könnten. Anschließend wird eine Antwort auf die Frage gesucht, welche Perfromanzvorteile man durch Nutzung von 3D-Daten gewinnen kann. Zu den Risiken bei der Projektdurchführung zählen eventueller Wechsel der Arbeitsumgebung (Lush, Matlab, C++) und die Beschaffung der richtigen Trainingsdaten (Menge, Qualität, Aussagekraft).

Im Rahmen der Masterarbeit untersuchen wir die Auswirkung verschiedener Parameter des Faltungsnetzwerkes, nämlich dessen Architektur, Konfiguration, Trainingsset, Filterparameter usw. Es wird darüber hinaus versucht, die Fehlertoleranz zu verbessern und auf den State-Of-The-Art Benchmarks bessere Ergebnisse zu erzielen. Ferner werden wir ein Experiment durchführen, bei dem die Leistung des Menschen mir der Leistung des entworfenen Netzwerkes verglichen wird.

Auch wenn ein solcher Vergleich ohne weiteres nicht möglich ist, kann die Gleichberechtigung durch geschickte Auswahl der Trainings- und Testbilder trotzdem aussagekräftig sein.

LITERATUR

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," Neural Comput., vol. 1, no. 4, Dec. 1989, pp. 541–551.

[2] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, May 2010, pp. 253–256.

[3] Chrislb, "Schema eines künstlichen neurons," Website, 2005, online erhältlich unter http://de.wikipedia.org/wiki/Datei:ArtificialNeuronModel_deutsch.png; abgerufen am 28. Februar 2015.

[4] A. Meisel, "Faltung," Vorlesungsskript "Robot Vision", 2012.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[6] "Imagenet large scale visual recognition challenge (ilsvrc)," Website, 2015, online erhältlich unter http://www.image-net.org/challenges/LSVRC/; abgerufen am 28. Februar 2015.

[7] O. Russakovsky, J. Deng, 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," 2014.

[8] "The pascal visual object classes homepage," Website, 2015, online erhältlich unter http://pascallin.ecs.soton.ac.uk/challenges/VOC/; abgerufen am 28. Februar 2015.

[9] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang, "Large-scale image classification: Fast feature extraction and svm training," in Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, June 2011, pp. 1689–1696.

[10] F. Perronnin, J. Snchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in Computer Vision ECCV 2010, ser. Lecture Notes in Computer Science, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Springer Berlin Heidelberg, 2010, vol. 6314, pp. 143–156.

[11] M. D. Zeiler, "Hierarchical convolutional deep learning in computer vision," Ph.D. dissertation, Department of Computer Science, New York University, 2014.

[12] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," CoRR, vol. abs/1311.2901, 2013.

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," CoRR, vol. abs/1408.5093, 2014.

[14] M. Lin, Q. Chen, and S. Yan, "Network in network," CoRR, vol. abs/1312.4400, 2014.

[15] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," CoRR, vol. abs/1207.0580, 2012.