technische universität
dortmund

Objekterkennung in natürlichen Szenen mittels Region-based Convolutional Neural Networks

Nils Brinkmann
7. Oktober 2016

Supervisors:
René Grzeszick, M.Sc.
Prof. Dr.-Ing. Gernot A. Fink

Fakultät für Informatik
Technische Universität Dortmund
http://www.cs.uni-dortmund.de

INHALTSVERZEICHNIS

1 EINFÜHRUNG

Kaum eine Methodik hat die Entwicklung von vielfältigen Verfahren im Bereich der künstlichen Intelligenz und Computer Vision so revolutioniert und dabei noch eine solche geschichtliche Entwicklung mit Höhen und Tiefen durchlebt wie die künstlichen Neuronalen Netze. Insgesamt zwei Mal seit der Entwicklung der ersten Vorläufer (Perzeptrons) 1958 riefen Probleme mit den existierenden Methoden ein vermindertes Interesse an Neuronalen Netzen hervor, Zeitperioden die Winter der Neuronalen Netze genannt werden. Heute allerdings führen Methoden, die Neuronale Netze einsetzen, Ranglisten verschiedenster Wettbewerbe in vielen Bereichen der künstlichen Intelligenz an. Die einfache Idee, sehr viele kleine mathematische Bausteine, genannt Neuronen, nach dem Vorbild der Nervenzellen des biologischen Gehirns zu einem Netz zu verschalten, führte zu einem vielseitig einsetzbaren und leistungsfähigen Konstrukt, das viele vorher dagewesene Verfahren abgelöst hat. Der technische Fortschritt im Bezug auf die Leistungsfähigkeit der Prozessoren und Grafikkarten ermöglicht dabei immer größere Netzarchitekturen mit immer mehr Neuronen. Die Verwendung solcher Neuronalen Netze mit vielen Schichten wird Deep Learning genannt und verbessert die Leistungsfähigkeit der Verfahren erheblich.
Einer der Bereiche in dem heute so gut wie alle führenden Verfahren Neuronale Netze einsetzen ist die Computer Vision. Eine der fundamentalen Aufgaben ist dabei die Objektlokalisierung und Klassifikation (kurz: Objektdetektion) in Bildern, auf welcher der Fokus dieser Arbeit liegen soll. Hier werden Neuronale Netze dazu eingesetzt, um aus Bildbereichen Merkmale zu berechnen, die dann als Grundlage für die Klassifikation dienen. Für die Evaluation solcher Verfahren existieren verschiedene Datensätze wie den ImageNet [RDS+15], Scene Understanding Database [XHE+10] oder Pascal Visual Object Classes [EGW+12] Datensätzen, mit welchen die Verfahren trainiert und getestet werden können. Zu einigen Datensätzen existieren jährlich abgehaltene Wettbewerbe um das beste Verfahren in verschiedenen Aufgaben zu bestimmen.
Diese Arbeit beinhaltet eine nähere Betrachtung des Objekterkennungsverfahrens Region-based Convolutional Neural Networks (R-CNNs) aus [GDDM15] von Girshick et al., in dem zunächst mithilfe des Selective Search Algorithmus Bildbereiche extrahiert werden, die Objekte enthalten können. Im Folgenden berechnet ein Convolutional Neural Network (CNN) Merkmale dieser Bildbereiche, die dann mithilfe von Support Vector Machines (SVMs) für die Klassifikation verwendet werden. Ein Merkmal dieses

Verfahrens besteht in dem zweiteiligen Training, bei dem das Neuronale Netz für die Klassifikationsaufgabe, in der keine Objekte sondern das ganze Bild klassifiziert werden, vortrainiert wird und erst dann mithilfe der Datensätze und Annotationen für die Objekterkennung auf die eigentliche Aufgabe fein-getuned wird. Dies ist nötig, da die geringe verfügbare Anzahl an vollständig annotierten Bildern für die Objekterkennung nicht ausreicht, um tiefe Neuronale Netze zu trainieren. Mit diesem Verfahren übertrafen die Autoren konkurrierende Methoden und waren bei der Veröffentlichung 2014 in vielen Wettbewerben führend. Folgende Veröffentlichungen der Autoren wie die Fast R-CNNs [RHGS15b] und Faster R-CNNs [RHGS15a] verbesserten wiederum die Erkennungsleistung und vor allem die Geschwindigkeit des Verfahrens.
Im Anschluss an die detaillierte Vorstellung und Reimplementation mithilfe des Deep-Learning Frameworks Caffe [JSD+14] soll die Leistungsfähigkeit verschiedener Variationen des R-CNN Verfahrens mithilfe der Pascal Visual Object Classes 2007 und Scene Understanding Database 2012 Datensätze analysiert und ein Vergleich zu dem konkurrierenden Verfahren Part-based Models hergestellt werden. Dabei werden Region-Proposal Networks als Alternative zum Selective Search Verfahren, die beiden Neuronalen Netzarchitekturen AlexNet und VGGNet zur Merkmalsextraktion und SoftMax Layer und Support Vector Machines als zwei Möglichkeiten für die Klassifikation evaluiert.

2 AUFGABEN

Innerhalb des Themenbereichs der Computer Vision stellen sich immer wieder ähnliche Aufgaben. Einige der am häufigsten vorkommenden Typen, wie sie auch in Wettbewerben wie der Pascal Visual Object Classes Challenge [EEG+15] vorkommen, sollen hier kurz vorgestellt werden.

2.1 SEGMENTIERUNG

Bei der Segmentierung geht es um die Trennung des Bildes in verschiedene Bereiche, die einzelne Objekte enthalten. Dies kann je nach Aufgabenstellung entweder pixelweise geschehen, sodass jedem Pixel ein Bildbereich, der dasselbe Objekt enthält, zugeordnet wird, oder es werden als Vereinfachung rechteckige Rahmen, welche die Objekte im Bild umschließen, gesucht und ausgegeben (siehe Abbildung 2.1.1). Segmentierungsverfahren können dafür zum Beispiel anhand der Farbe, Textur, Schattierung, Größe und Position zusammengehörende Regionen im Bild ermitteln, die dann als Ergebnis ausgegeben werden.
Schwierig zu segmentieren sind dabei Objekte, die aus mehreren, stark verschiedenen Teilen bestehen (zum Beispiel ein Auto mit seinen Rädern, oder eine Person mit ihrer Kleidung, siehe [USGS13]). Diese Objektteile lassen sich ohne Kontextinformationen oft nur schlecht durch Ähnlichkeiten in Farbe, Textur, etc. zu einem Objekt zusammenfassen und erhöhen deshalb die Komplexität der Segmentierung.
Ein gutes Segmentierungsverfahren zeigt einen möglichst großen Schnitt der berechneten Fläche im Bild und dem Ground Truth im Vergleich zur Vereinigung der beiden Flächen. Als Metrik wird deshalb oft die Intersection over Union (IoU) verwendet.

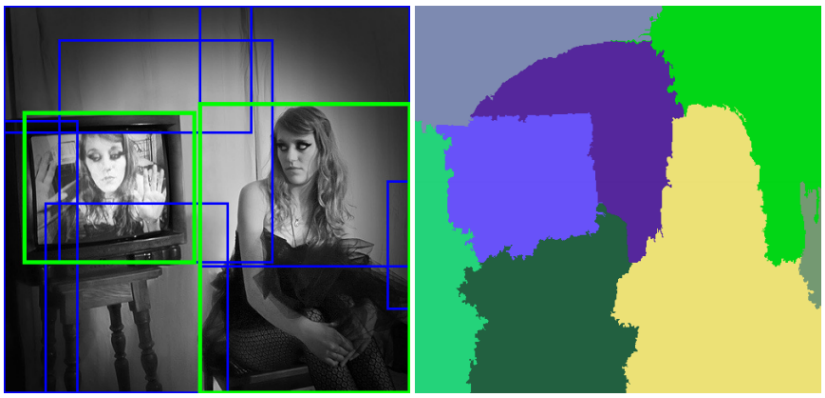


Abbildung 2.1.1: Bounding-Box-basierte und Pixelweise Segmentierung aus [USGS13]

2.2 KLASSIFIKATION

Klassifikation behandelt die Zuordnung von Daten zu einer Klasse, in diesem Fall also die Zuordnung von Bildern zu einer der möglichen Bildklassen. Diese können dabei zum einen beschreiben, was auf dem Bild zu sehen ist (z.B. Person, Auto oder Hund), oder wo die gezeigte Szene aufgenommen wurde (z.B. Wohnzimmer, Wald, Stadt). Oftmals wird dabei nicht nur eine Klassifikation ausgegeben, sondern es werden mehrere Klassifikationen des Bildes inklusive der zugehörigen Pseudowahrscheinlichkeit gefordert. Ein Beispiel dafür ist der Pascal VOC Classification Task [EGW+12], bei dem die Bestimmung der Präsenz von allen Objektklassen gefordert ist. Ziel der Klassifikation ist es also, eine möglichst passende Beschreibung für das Bild zu finden. Neben dem einfachen Klassifikationsfehler ist die Average Precision (AP) eine Metrik solcher Klassifikationsverfahren, die angibt ob und an welcher Stelle in der nach Pseudowahrscheinlichkeit absteigend sortierten Liste der Klassifikationen sich die korrekte Klassifikation befindet [EGW+12]. Die gemittelte Average Precision über alle Objektklassen wird als Mean Average Precision (mAP) bezeichnet.

2.3 DETEKTION

Detektion ist eine Kombination aus Segmentierung und Klassifikation, bei der allen relevanten Objekten im Bild die zugehörigen Klassen zugewiesen wird. Das Ergebnis ist dann die Objektklasse mit dem dazugehörigem rechteckigen Objektrahmen. Die Pixel-genaue Segmentierung mit anschließender Klassifikation wird dahingegen meist

semantische Segmentierung genannt. Ziel ist dabei das Lokalisieren, also Bestimmen der Position und Größe aller relevanten Objekte im Bild und das korrekte Zuordnen zu Ihren Objektklassen.

Die Evaluierung stellt bei der Detektion ein besonderes Problem dar, da sowohl die Position als auch die Korrektheit der Klassifizierung die Leistung des Verfahrens beeinflussen. Daher wird oft ein Ansatz gewählt, in dem erkannte Objekte mit einer Überlappung von mehr als 50% mit einem Ground Truth Objekt diesem zugeordnet werden und die Klassifikationsleistung dann wie bei der Klassifikation mit der (Mean) Average Precision [EGW+12] bewertet wird. Weitere Informationen zur Evaluierung und den verwendeten Metriken sind in 6.3 zu finden. Der Fokus dieser Arbeit wird auf der Detektion von Objekten mithilfe von künstlichen Neuronalen Netzen liegen.

Abbildung 2.3.1: Ergebnis einer Detection aus [GDDM15]

2.4 WEITERE AUFGABEN

Neben den drei Hauptaufgaben Segmentierung, Klassifikation und Detektion werden durch immer speziellere Anforderungen noch weitere Aufgaben mit dazugehörigen Wettbewerbskategorien ins Leben gerufen. Anwendungsbereiche wie Virtual Reality erfordern die Lokalisierung von Köpfen, Händen und Füßen von Personen, wie die Person Layout Competition der Pascal VOC Challenge [EEG+15]. Es können wie in der Action Classification Handlungen von Personen in Bildern bestimmt oder durch die gestiegene Rechenleistung und Performanz der Algorithmen sogar Object Detection in Videos [RDS+15] durchgeführt werden.
Bemerkenswert ist, dass einige dieser Aufgaben durch lediglich neues Trainieren Neuronaler Netze mit den grundsätzlich gleichen Methoden erreicht werden können, die auch für Object Detection eingesetzt werden.

3 NEURONALE NETZE

Ein zentraler Bestandteil vieler aktueller Objekterkennungsverfahren sind (künstliche) Neuronale Netze, also Abstraktionen der biologischen Neuronalen Netze, wie sie in Gehirnen von Mensch und Tier zu finden sind. In ihnen werden viele künstliche Neuronen miteinander vernetzt um eine Eingabe auf eine Ausgabe abzubilden. Im Kontext der Objekterkennung soll mithilfe des Neuronalen Netzes ein Bildbereich auf einen Merkmalsvektor abgebildet werden, mit dem möglichst einfach die zugrundeliegende Objektklasse ausgerechnet werden kann. Grundsätzlich sind Neuronale Netze dazu in der Lage, beliebige Funktionen zu approximieren.

3.1 FUNKTIONSWEISE

Der grundlegende Baustein Neuronaler Netze ist das Neuron. Es besteht aus eingehenden Kanten zu vorhergehenden Neuronen (oder der Eingabe die vom Netz bearbeitet werden soll) mit dazugehörigen Gewichten und einer Aktivierungsfunktion. Die Eingaben x_1 bis x_n und Gewichte w_1 bis w_n werden mithilfe der Übertragungsfunktion verrechnet und mit einem Bias w_0 versehen (siehe Abbildung 3.1.1). Aus diesen Werten wird die Neuroneneingabe y berechnet, die dann mithilfe der Aktivierungsfunktion Θ auf die Ausgabe u abgebildet wird.
Die Übertragungsfunktion ist im Normalfall wie folgt definiert:

$$\Sigma = \sum_{i=0}^{n} x_i \cdot w_i$$

Mit ihr werden also die Eingaben mit den jeweiligen Gewichten versehen und aufsummiert. Grafisch bedeutet ein Gewicht von 0, dass keine Kante zu dem jeweiligen Eingabeneuron existiert. Eine Besonderheit stellt das Gewicht w_0, dass mit einem On-Neuron ($x_0 = 1$) verbunden ist. Mithilfe dieser Konstruktion wird der Funktionswert der Übertragungsfunktion um w_0 verschoben. Diese Betrachtung des Bias vermeidet die Betrachtung von Sonderfällen in der weiteren Berechnung.
Etwas komplexer ist die Wahl einer geeigneten Aktivierungsfunktion, denn sie ist die Quelle der Nichtlinearität der Berechnung die im Netz stattfindet. Einige gebräuchliche

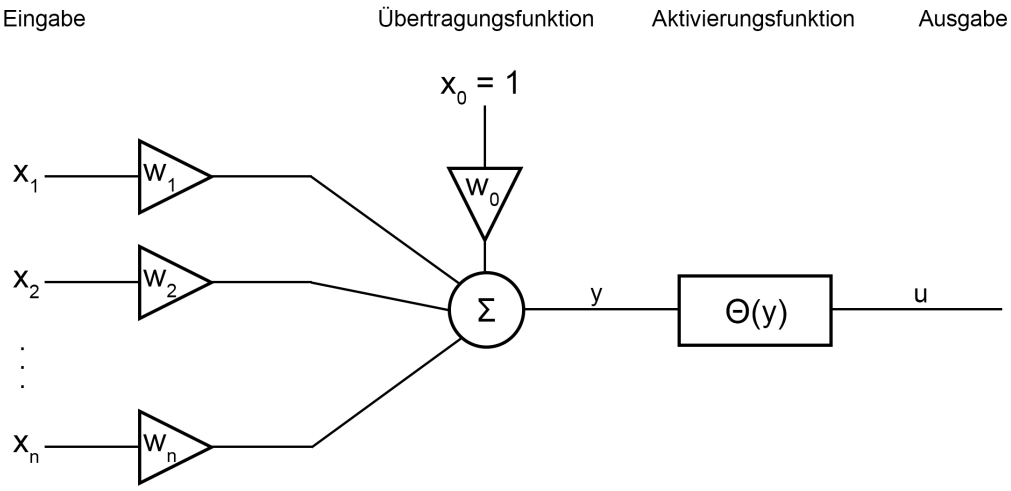


Abbildung 3.1.1: Abbildung eines künstlichen Neurons nach [Nie03]

Aktivierungsfunktionen sind die Schwellwertfunktion, Sigmoidfunktion und Rectified Linear Units (ReLUs):

$$\Theta_{\text{Schwellwert}}(y) = \begin{cases} 1, & \text{falls } y \geqslant 0 \\ 0, & \text{sonst} \end{cases}$$

$$\Theta_{\text{Sigmoid}}(y) = \frac{1}{1 + e^{-\alpha y}}$$

$$\Theta_{\text{ReLU}}(y) = \max(0, x)$$

Die Sigmoidfunktion wurde lange Zeit verwendet, beinhaltet aber das sogenannte Vanishing Gradient Problem. Hier führt der beschränkte Wertebereich $(0, 1)$ der Sigmoidfunktion dazu, dass beim Training mithilfe des Backpropagation Algorithmus (siehe Kapitel 3.3.2) das Training in den vorderen Layern verlangsamt sein kann. Deshalb werden mittlerweile oft die Rectified Linear Units (ReLUs) verwendet, da das Training Neuronaler Netze mit ReLUs im Vergleich zu anderen Verfahren schneller konvergiert (siehe [KSH12]), zu Beginn in einem zufällig initialisierten Netz mit ReLUs im Schnitt nur 50% der Neuronen aktiviert sind und die Funktion ist zudem noch effizient berechenbar (siehe [Fin16]). Neuronale Netze mit ReLUs zeigen deshalb oft bessere Ergebnisse.

3.2 ARCHITEKTUR

Ordnet man die Neuronen in hintereinander liegende Layer an, in dem jedes Neuron aus Layer i nur eingehende Verbindungen von Layer i-1 und ausgehende Verbindungen zu Neuronen in Layer i+1, so erhält man ein einfaches Neuronales Netz (genauer: ein Multi-Layer Perzeptron). Ein solches Netz gliedert sich in ein Eingabelayer, ein oder mehrere versteckte Layer und dem Ausgabelayer, wie in Abbildung 3.2.1 zu sehen ist. Im Eingabelayer in Neuronalen Netzen für die Bildverarbeitung hat ein Neuron jeweils eine Verbindung zu einem der Farbwerte eines Pixel im Bild, die versteckten Layer führen mithilfe ihrer Neuronenverbindungen zu den vorherigem Layer die eigentliche Berechnung durch und das Ergebnis steht dann an den ausgehenden Kanten der Neuronen des Ausgabelayers zur Verfügung. Das Ergebnis wird dabei bei der Detektion meist 1 aus K kodiert, sodass pro Objektklasse (+ Hintergrund) ein Neuron existiert, wobei die Ausgabe dieses Neurons der Pseudowahrscheinlichkeit der Präsenz dieser Klasse entspricht.

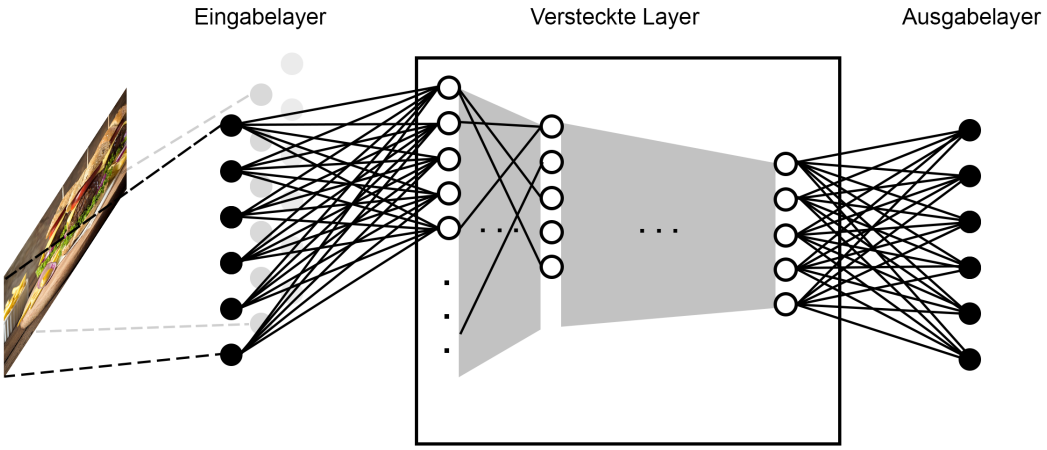


Abbildung 3.2.1: Abbildung eines künstlichen Neuronalen Netzes

Obwohl die Eingabebilder in diesem Anwendungsbereich normalerweise zweidimensional sind, haben Layer oft Neuronen in einer dreidimensionalen Anordnung in sogenannten Slices. Innerhalb eines Layers können mithilfe von mehreren hintereinander liegenden Slices mehrere Merkmale für die Eingabe aus dem vorherigen Layer berechnet werden. Jeder Slice berechnet dann ein Merkmal für die komplette Eingabe

des Layers und innerhalb eines Slices teilen sich die Neuronen viele der Parameter (Gewichte, etc.), was den Speicherbedarf und Lernaufwand erheblich verringert.
Einige Layer-Typen, die für die Objekterkennung von Bedeutung sind, sollen im Folgenden vorgestellt werden.

3.2.1 *Convolutional Layer*

In einem Convolutional Layer hat jedes Neuron einen festen Bereich von Eingabeneuronen, mit denen es verbunden ist, das Receptive Field. Dieses Receptive Field ist zentriert um die Position des Neurons und hat eine feste Größe. Mathematisch entspricht die Berechnung der Ausgabe des Neurons in einem Convolutional Layer einer Faltung (Convolution) der Eingabewerte (Werte der Eingangs-Neuronen) mit einer Maske, die sich aus den Kantengewichten ergibt.
Weitere wichtige Parameter für ein Convolutional Layer sind die Tiefe (Depth), die angibt, wie viele Masken mit den Eingabeneuronen verrechnet werden, um verschiedene Merkmale in einem Layer zu berechnen (also die Anzahl an Slices im Convolutional Layer) und der Stride-Parameter, der den horizontalen und vertikalen räumlichen Abstand zwischen Neuronen im Convolutional Layer im Vergleich zum vorhergehenden Layer angibt. Je nach Wahl der Receptive Field Größe und des Stride-Parameters, können sich die Receptive Fields der Neuronen überlappen oder sogar Abstand zueinander haben.

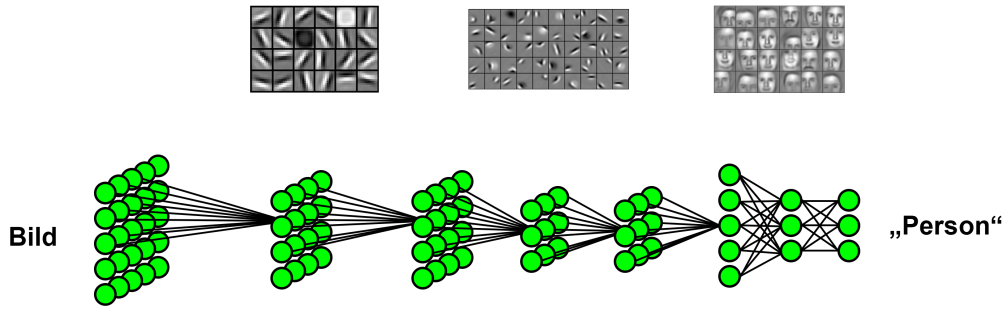


Abbildung 3.2.2: Abbildung einer CNN Architektur, die in tieferen Layern immer komplexere Merkmale berechnet. Grafik in Anlehnung an [Bro14]

Ein wichtiger Aspekt der Convolutional Layer ist das Teilen der Faltungsmaske zwischen allen Neuronen eines Slices in einem Layer. Berechnet eine Maske ein sinnvolles Merkmal für einen Bereich im Bild, ist anzunehmen, dass dies für alle Bildbereiche gilt. Diese Annahme begrenzt die Anzahl an Parametern, die innerhalb des Trainings des Neuronalen Netzes gelernt werden müssen, beschleunigt also den Lernvorgang und beschränkt den Speicherbedarf erheblich, ohne dass die Leistungsfähigkeit des Netzes im Kontext der Objekterkennung erheblich beeinflusst wird (siehe [KSH12]). Zusätzlich lassen sich die Berechnungen in einem Convolutional Layer einfach mathematisch mithilfe einer Matrixaddition und -multiplikation darstellen, die von Grafikkarten sehr effizient berechnen werden können. Dies führt dazu, dass CNN Architekturen erheblich tiefer sein können (mehr Layer enthalten) und die Berechnung trotzdem performant bleibt. Aus diesen Gründen ist das Convolutional Layer in Netzen mit dem Einsatzbereich Bildverarbeitung gut geeignet und es ist erkennbar, warum diese Art Layer namensgebend für eine eigene Klasse Neuronaler Netze ist.

3.2.2 *Pooling Layer*

Pooling Layer fassen Bereiche einer festgelegten Größe in der Eingabe zu einem einzelnen Ausgabewert zusammen. Häufig werden in Neuronalen Netzen Max-Pooling Layer eingesetzt, bei denen aus den zugrundeliegenden Werten in der Eingabe jeweils das Maximum übernommen wird. Dieses Vorgehen reduziert, je nach Größe der Bereiche die zusammengefasst werden, die Anzahl der Dimensionen von der Ausgabe erheblich und sorgt so gleichzeitig für eine Invarianz gegenüber kleinen Verschiebungen von Features. Dadurch wird auch der Rechenaufwand und Speicherbedarf für die dahinter liegenden Layer verringert (siehe [SMB10]).

3.2.3 *Fully Connected Layer*

Eine weitere Art Layer, die vorwiegend in den hinteren Schichten einer CNN-Architektur anzufinden ist, sind die Fully Connected Layer. Sie haben wie in Mehrschichtperzeptronen Verbindungen zu allen Neuronen der vorhergehenden Schicht ohne eine räumliche Beschränkung. Sie nutzen die von den vorhergehenden Layern berechneten Merkmale für bestimmte Bildbereiche, um daraus komplexere Merkmale für die gesamte Eingabe zu berechnen. Aus diesen komplexeren Merkmalen kann dann mithilfe von Support Vector Machines oder SoftMax Classifiern eine Klassifikation bestimmt werden.
Durch die viel größere Anzahl an Kanten in einem Fully Connected Layer im Vergleich zum Convolutional Layer ist auch der Rechenaufwand vergleichsweise groß, weshalb

die Fully Connected Layer eher am Ende einer Netzarchitektur stehen, nachdem die Anzahl der Dimensionen zum Beispiel durch Max Pooling Layer oder einer Stride > 1 in Convolutional Layern schon verringert und die einfacheren Merkmale bereits durch andere Layertypen berechnet wurden.

3.2.4 *SoftMax Layer*

Bei Klassifizierungsproblemen stehen oftmals SoftMax Layer am Ende einer Netzarchitektur, die aus den Neuronenaktivierungen des vorhergehenden Layers Pseudowahrscheinlichkeiten für Klassenzugehörigkeiten berechnen können. Die Eingabe für das SoftMax Layer hat dabei schon die korrekte Dimensionalität mit $1 \times K$ Werten bei K Objektklassen, allerdings können die Werte selber beliebig groß oder klein sein. Mithilfe der SoftMax Funktion (siehe [Biso6])

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad \text{für } j = 1, \ldots, K$$

werden diese Zahlen korrespondierend zu der Höhe ihres vorherigen Wertes in den Wertebereich $[0, 1]$ gebracht, sodass die Werte aller Klassen aufaddiert 1 ergeben. Diese Berechnung ermöglicht eine intuitive Betrachtung der Ausgabe des Neuronalen Netzes ähnlich zu Wahrscheinlichkeiten.

3.3 TRAINING VON NEURONALEN NETZEN

Der zentrale Schritt, der einem Neuronalen Netz seine Fähigkeiten zur Erkennung verleiht, ist das Training. Im Training werden die Gewichte und Bias aller Neuronen im Netz schrittweise so angepasst, das es die Eingaben auf die gewünschten Ausgaben abbildet, es „lernt“. Ausgangspunkt ist dabei ein Netz mit zufällig initialisierten Gewichten. Im überwachten Lernen wird eine Eingabe dann durch das zufällig initialisierte Netz auf eine Ausgabe abgebildet und diese dann mit der gewünschten Ausgabe verglichen. Die Abweichung (Fehler) von der gewünschten zur tatsächlichen Ausgabe des Netzes wird anschließend dazu genutzt, die Gewichte im Netz in kleinen Schritten zu korrigieren. Da hierfür natürlich das Soll-Ergebnis, also die Information über die korrekten Klassifikationen erforderlich ist, handelt es sich bei dem Verfahren um überwachtes Lernen, bei dem eine Menge an annotierten Trainingsdaten zugrunde liegen muss. Der Algorithmus, mit dem das überwachte Lernen durchgeführt wird, heißt Backpropagation. Hiermit wird der entstandene Fehler rückwärts durch das Netz geleitet, um die Gewichte anzupassen.

3.3.1 *Loss Funktion*

Ausgangspunkt für den Backpropagation Algorithmus ist eine Metrik für den Fehler, den das Netz in seinem aktuellen Zustand bei der Berechnung begeht, also die Differenz zwischen Soll- und Ist-Ergebnis. Diese Abweichung wird mithilfe einer Loss-Funktion berechnet, wie zum Beispiel der Euklidischen Loss-Funktion (siehe [Fin16]).

$$E_{\text{Euclid}} = \frac{1}{2N} \sum_{n=1}^{N} \|\hat{f}_n - f_n\|_2^2$$

wobei

$\hat{f} \in [-\infty, +\infty]$ den berechneten Werten (Ist-Werten) und
$f \in [-\infty, +\infty]$ den Zielwerten (Soll-Werten) enspricht.

Sie berechnet den Euklidischen Abstand zwischen dem Soll- und Ist-Ergebnis und ist so ein Maß für den Grad der Abweichung. Eine weitere wichtige Loss-Funktion wird im Zusammenhang mit der Kategorisierung über SoftMax Layer verwendet. Die Multinomiale logistische Loss-Funktion (siehe [Caf16])

$$E_{\text{MultinomialLogistic}} = -\frac{1}{N} \sum_{n=1}^{N} \log(\hat{p}_{n,l_n})$$

wobei

$\hat{p} \in [0, 1]$ dem Vektor mit den berechneten Klassenkonfidenzen und
$l_n \in [0, 1, \ldots, K-1]$ der korrekten Klassifikation für Beispiel n entspricht.

berechnet den Unterschied zwischen der vom Neuronalen Netz berechneten Wahrscheinlichkeitsverteilung und der korrekten Klassifizierung. Der richtigen Klasse wird eine Konfidenz von 1 und allen anderen Klassen eine 0 zugewiesen.

3.3.2 *Backpropagation*

Ziel des Trainings ist es, ein globales Minimum für die Loss-Funktion zu finden. Dafür muss die partielle Ableitung $\delta E/\delta \hat{x}_j$ für das Ausgabelayer gefunden werden. Die Vorgehensweise dafür ist [RHW88] entnommen. Mithilfe der Kettenregel und der

Ableitung der Loss-Funktion E' und der Aktivierungsfunktion Θ' kann die Abhängigkeit zwischen einer Änderung der Eingabe y für das Neuron und dem Fehler E gefunden werden.
Die Eingabe y selbst ist jedoch wieder abhängig von den Ausgaben und Kantengewichten zu den Neuronen des vorherigen Layers, also kann auch der Effekt der Kantengewichte $\delta E/\delta w$ auf den Fehler berechnet werden. Eine einfache Variante, die Gewichte im Netz anzupassen ist es, jedes Gewicht nach folgender Vorschrift um einen zu $\delta E/\delta w$ proportionalen Anteil anzupassen:

$$\Delta w = -\eta \delta E/\delta w$$

wobei η der Lernrate, also der Änderungsgeschwindigkeit der Gewichte entspricht. Diese ist zu Anfangs relativ groß zu wählen, da zu erwarten ist, dass die zufällig gewählten Initialgewichte relativ weit entfernt von den Zielgewichten liegen und muss später im Lernverfahren reduziert werden, um nicht über ein lokales Optimum der Gewichte hinweg zu schreiten.
Variiert werden kann außerdem der Zeitpunkt, zu dem die Gewichte aktualisiert werden. Das Training findet entweder online, das heißt die Gewichte werden nach jeder Betrachtung eines Beispiels aktualisiert, oder Batch-weise statt, wobei die Gewichtsänderungen zunächst über alle Beispiele eines Batches akkumuliert werden und dann nach Betrachtung aller Beispiele eine Aktualisierung durchgeführt wird. Letzteres Vorgehen kann gegebenenfalls robuster sein, da die Gewichtsänderung über mehrere Beispiele gemittelt wird und so Schwankungen vermieden werden.

3.3.3 *Vermeidung von Overfitting*

Ein großes Problem beim Training Neuronaler Netze ist das Fehlen von großen Mengen an korrekt klassifizierten Beispielen, anhand derer man das Netz trainieren kann. Tiefe Netze mit vielen Layern sind in der Lage, die Klassifikation sehr nah an den vorhandenen Trainingsbeispielen auszurichten, wenn nicht ausreichend viele Beispiele für jede Klasse verfügbar sind. Dies führt zu dem sogenannten Overfitting, bei dem zu einem großen Teil nur die Objekte korrekt einer Klasse zugewiesen werden, die sich auch in den Trainingsdaten befanden. Da es die Aufgabe des Neuronalen Netzes für die Klassifikation oder Detektion ist, ein Konzept einer Klasse und nicht nur die Trainingsbeispiele zu erkennen, ist dieses Verhalten nicht erstrebenswert. Um Overfitting zu vermeiden, werden deshalb bei der Architektur und dem Training von Neuronalen Netzen verschiedene Verfahren angewendet.
Zum einen wird versucht, grundsätzlich schon eine möglichst große Menge an Trainingsdaten zu Verfügung zu stellen. Erst durch das Internet und die dadurch verfügbar

gewordenen großen Mengen an Bildern können so tiefe Neuronale Netze wie das AlexNet oder gar VGGNet überhaupt trainiert werden.
Zusätzlich werden die großen Bilddatenbanken möglichst plausibel durch Datenaugmentierungen erweitert. Beispielsweise können in vielen Anwendungsfällen die Bilder mit den dazugehörigen Annotationen gespiegelt oder Bildausschnitte extrahiert und als weitere Trainingsbeispiele angefügt werden. Im Training der Netze wird außerdem ein sogenannter Dropout durchgeführt, bei dem in jeder Iteration ein bestimmter Anteil (meist 50%) der Neuronen zufällig entfernt wird, damit die Gewichte durch das Training nicht genau an die Trainingsbeispiele angepasst werden können [SHK+14]. Es wird in jeder Iteration ein anderes Netzwerklayout trainiert, sodass die so gelernten Merkmale unabhängiger gegenüber den Werten einzelner Neuronen und damit nützlich in Kombination mit vielen Neuronen-Konstellationen sind. Ein Nachteil dieses Verfahrens ist allerdings die erheblich längere erforderliche Trainingsdauer.

3.4 ERFOLGREICHE CNN-ARCHITEKTUREN

Ein Layertyp allein macht noch kein gutes Neuronales Netz aus. Convolutional Layern fehlt die Möglichkeit der Schlussfolgerung auf einer höheren Ebene. Zwar haben Fully-Connected Layer diese Fähigkeit, die Anzahl der zu lernenden Parameter wird mit einem Netz bestehend ausschließlich aus Fully-Connected Layern aber so groß, dass ein Training eines solchen Netzes mittelfristig unmöglich bleiben wird. Auch die Gefahr des Overfittings würde erheblich vergrößert werden.
Deshalb ist eine geeignete Netzarchitektur passend zum Task und den zugrundeliegenden Daten von zentraler Bedeutung bei der Erkennungsleistung eines Neuronalen Netzes. Unter den besonders erfolgreichen Vertretern der CNN-Architekturen der letzten Jahre befinden sich das AlexNet und das VGGNet.

3.4.1 *AlexNet*

Das AlexNet, vorgestellt in [KSH12], ist die erste CNN-Architektur, die eine führende Erkennungsleistung im ILSVRC Wettbewerb gezeigt hat. Es besteht aus 650,000 Neuronen und 60 Millionen Parametern. Eine Besonderheit des Vorgehens von Krizhevsky et al. ist die Aufspaltung des Netzes auf zwei Grafikkarten um die Speicherlimitierung von 3GB der zum Zeitpunkt der Entwicklung verfügbaren Grafikkarten (GTX 580) zu umgehen.
Die Neuronen befinden sich jeweils zur Hälfte auf den beiden Grafikkarten, wobei nur ein Teil der Layer Grafikkarten-übergreifend auf die Neuronen des vorherigen Layers

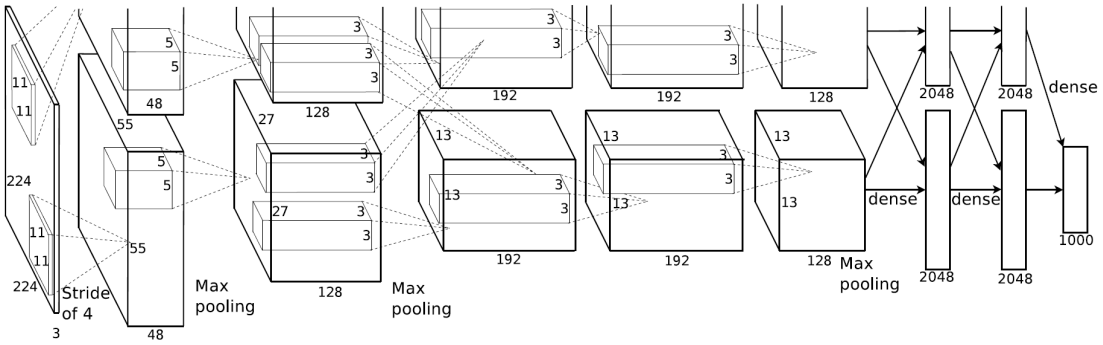


Abbildung 3.4.1: Visualisierung der Architektur des AlexNet aus [KSH12]

zugreift, wie in Abbildung 3.4.1 zu erkennen ist. Dieser Teil ist so klein wie möglich zu halten, da die inter-Grafikkarten Kommunikation erheblich langsamer ist als der Zugriff auf den Grafikkarten-internen Speicher, jedoch groß genug um die Leistung des Netzes nicht stark negativ zu beeinflussen.
Ausgehend vom $224 \times 224 \times 3$ großen Eingabebild wird mithilfe von fünf Convolutional Layern und drei Fully-Connected Layern die Erkennung durchgeführt. Nach Convolutional Layer 1, 2 und 5 befinden sich Max-Pooling Layer, um die Anzahl der Neuronen in den folgenden Layern zu verringern und um die Robustheit gegenüber translatorischen Änderungen zu steigern. Die Pooling Layer weisen dabei eine Überlappung des Pooling-Bereichs auf, was die Erkennungsleistung des Netzes nochmals um circa 0.4% gesteigert hat und Overfitting weiter vermindern soll.
Die Größe des Neuronalen Netzes macht es trotz der Größe der ImageNet Datensätze von 1.2 Millionen Bildern sehr schwierig, Overfitting zu vermeiden. Deshalb wurden neben der Datenaugmentation der Eingabedaten auch ein Dropout innerhalb der ersten beiden Fully-Connected Layer (beides beschrieben in Kapitel 3.3.3) angewendet.
Fisher Vektoren (FV), die vor dem AlexNet das bis dahin beste Ergebnis im ILSVRC 2010 erzielt haben, ergänzen den Bag of Features Ansatz um eine räumliche Komponente. Anstatt lediglich den zugehörigen Zentroid (Klasse) zu einem Feature zu speichern, werden die Distanzen zu den nächsten Zentroiden gespeichert. Mit dem AlexNet übertreffen Krizhevsky et al. die Ergebnisse des FV-Verfahrens um jeweils ca. 8% mit 37.5% Top-1 und 17.0% Top-5 Fehlerraten im ILSVRC 2010 Classification Wettbewerb.

3.4.2 *VGGNet*

Eine weitere sehr erfolgreiche CNN-Architektur ist das VGGNet [SZ14]. Betrachtet wird an dieser Stelle die Variante mit 16 gewichteten Layern. Der grundsätzliche Aufbau ist vergleichbar mit dem des AlexNet, lediglich mit deutlich mehr Layern. Im VGGNet folgen auf 13 Convolutional Layern, zwischen denen teilweise Max-Pooling Layer liegen, drei Fully Connected Layer und ein SoftMax Layer zur Klassifizierung. Der genaue Aufbau ist in Abbildung 3.4.2 zu erkennen.

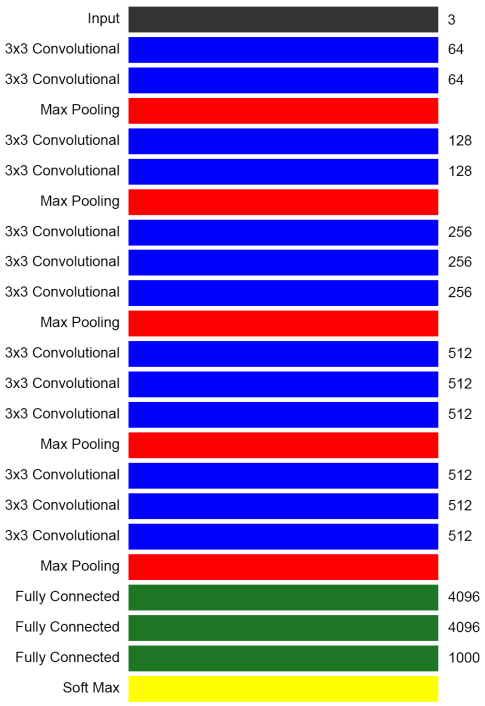


Abbildung 3.4.2: Visualisierung der Architektur des VGGNet Variante D

Neben der Tiefe der Architektur besteht ein wesentlicher Unterschied zum AlexNet in dem Aufbau der Convolutional Layer. Während beim AlexNet noch Convolutional Layer mit großem Receptive Field (wie zum Beispiel 11×11) verwendet wurden, ist das VGG-Net ausschließlich aus Convolutional Layern mit einem kleinen 3×3 Receptive Field aufgebaut. Werden beispielsweise drei Convolutional Layer mit einem Receptive Field der Größe 3×3 hintereinander gelegt, so haben die Neuronen des letzten Layers der Gruppe insgesamt ein Receptive Field der Größe 7×7 bezogen auf die Eingabe des ersten Layers der Gruppe. Gleichzeitig werden jedoch für drei Convolutional Filter mit einem 3×3 großen Receptive Field $3 \times (3 \times 3) = 27$ Parameter benötigt, wohingegen ein einzelnes Layer mit gleich großem Receptive Field $7 \times 7 = 49$ Parameter benötigt hätte. Durch diese Einsparung an Parametern und dem Fortschritt bei der technologischen Entwicklung (wie zum Beispiel den erheblich größeren Speicherkapazitäten moderner Grafikkarten) wird es überhaupt erst möglich, ein so tiefes Netz wie das VGGNet mit seinen 16 Layern mit Gewichten und circa 140 Millionen Parametern einzusetzen.

Grundsätzlich unterscheidet sich die VGGNet Architektur, abgesehen von der Tiefe des Netzes, nicht wesentlich von anderen CNN-Architekturen wie dem AlexNet

oder dem LeNet von LeCun et al. [LBBH98], einem der ersten Convolutional Neural Networks eingesetzt zur Handschriftenerkennung. Dies zeigt, dass eine größere Tiefe des Convolutional Neural Networks einen starken positiven Einfluss auf die Klassifikationsleistung des Verfahrens hat und lässt die Hypothese zu, dass, wenn mit dem technologischen Fortschritt und neuen Verfahren noch tiefere Netzarchitekturen möglich werden, auch die Fehlerraten weiter verbessert werden können.
Mit dieser Architektur erreichen Simonyan et al. Fehlerraten von 23,7% (Top-1) und 6,8% (Top-5) im ILSVRC 2012 Classification Task und übertreffen beispielsweise das AlexNet um 14,4% beziehungsweise 11,4% im gleichen Wettbewerb.

4 REGION-BASED CONVOLUTIONAL NEURAL NETWORKS

Die Aufgabe der Objekterkennung in Bildern stellt eine große Herausforderung für Computer dar. Neben einer Vielzahl von Objektklassen, welche die Verfahren gegebenenfalls in einem einzelnen Bild voneinander unterscheiden müssen, haben die zu erkennenden Objekte oftmals keine einheitliche Größe oder Gestalt. Girshick et al. suchten ausgehend von der Bildklassifikation mithilfe von Convolutional Neural Networks ein Verfahren für die Objekterkennung, das mit diesen schwierigen Bedingungen umgehen kann und entwickelten die Region-based Convolutional Neural Networks (R-CNNs) [GDDM15].
In diesem Verfahren werden auf Basis des Eingabebildes zunächst Bildbereichsvorschläge mit einem Segmentierungsalgorithmus bestimmt, deren Merkmale anschließend separat mithilfe eines Convolutional Neural Network berechnet werden. Auf der Basis der berechneten Merkmale findet dann die eigentliche Klassifikation statt.
Die einzelnen Schritte dieses dreiteiligen Verfahrens sollen im Folgenden näher erläutert werden.

4.1 BILDBEREICHSVORSCHLÄGE

Die Segmentierung des Bildes in seine Objekte stellt die Grundlage des gesamten Verfahrens dar. Da die Objekte im Bild nicht klassifiziert werden können, wenn Sie nicht gefunden werden, ist die Qualität der Bildbereichsvorschläge ausschlaggebend für die Erkennungsleistung des Verfahrens.
Der Selective Search Algorithmus von Uijlings et al. [USGS13] setzt als Ausgangspunkt für eine hierarchische Segmentierung den Algorithmus von Felzenszwalb und Huttenlocher aus [FH04] ein. Dieser Algorithmus verwendet eine Graphstruktur, in der Pixel als Knoten repräsentiert werden und benachbarte Pixel mit Kanten verbunden sind. Die Kanten besitzen Gewichte, welche mit der Unähnlichkeit zwischen den zwei benachbarten Pixeln korrespondieren. Mithilfe dieser Darstellung wird eine Partition des Graphen berechnet, in der die Kantengewichte innerhalb eines Segments klein und die Kantengewichte zwischen zwei Segmenten möglichst groß sind. Unter der Wahl geeigneter Parameter kann dann eine Übersegmentierung gefunden werden, mit der das hierarchische Zusammenfassen ähnlicher Regionen im Selective Search

Algorithmus beginnen kann.
Uijlings et al. stellen dabei drei grundlegende Anforderungen an ihren Segmentierungsalgorithmus:

1. Unabhängigkeit gegenüber der Objektgröße
   Objekte können in Bildern in den verschiedensten Größen auftreten. Je nach Distanz zum Aufnahmegerät kann ein und dasselbe Objekt unabhängig von seiner eigentlichen Größe groß im Vordergrund oder klein im Hintergrund erscheinen. Ein guter Segmentierungsalgorithmus sollte das Bild in seine Objekte unabhängig von deren Größe segmentieren können.
   Deshalb verwenden Uijlings et al. einen hierarchischen Algorithmus der ausgehend von einer Übersegmentierung ähnliche Segmente zusammenfasst. Dieses Vorgehen erfüllt die geforderte Unabhängigkeit zur Objektgröße.

2. Anwenden verschiedener Gruppierungsstrategien
   Verschiedenste Eigenschaften, wie zum Beispiel Ähnlichkeiten in Farbe, Position, Größe, Textur oder Beleuchtung können auf zusammengehörige Segmente hinweisen. Aus diesem Grund kann es keine einzelne Gruppierungsstrategie geben, die in allen Fällen zusammenpassende Segmente des übersegmentierten Bildes korrekt zusammenfasst. Es ist also wichtig, eine Reihe von Ähnlichkeitsmaßen zu berechenen, die auf verschiedenen Merkmalen basieren, auf deren Basis die Gruppierung stattfinden kann.
   Ein Beispiel für ein solches Ähnlichkeitsmaß ist die Größenähnlichkeit, mit der kleine Regionen eher zusammengefasst werden. Als Maß wird der prozentuale Anteil verwendet, den die beiden Regionen zusammen im Bild überdecken: $s_{size}(r_i, r_j) = 1 - \frac{size(r_i)+size(r_j)}{size(im)}$

3. Hohe Geschwindigkeit der Berechnung
   Ein gutes Segmentierungsverfahren muss das Segmentierungsergebnis auch in einer annehmbaren Zeit zur Verfügung stellen können. Gerade wenn der Algorithmus zum Beispiel für die Detektion von Objekten in Bildern noch mit anderen rechenintensiven Verfahren wie einer Verarbeitung des Bildes und der Regionen durch Neuronale Netze kombiniert wird, darf die Segmentierung des Bildes nicht zu viel Zeit in Anspruch nehmen.

Aufbauend auf den verschiedenen Ähnlichkeitsmaßen aus Anforderung 2 und dem übersegmentierten Ausgangsbild wird die hierarchische Gruppierung nach einem einfachen Schema durchgeführt. Es werden iterativ immer zwei Segmente verschmolzen, die das aktuell maximale Ähnlichkeitsmaß untereinander besitzen (siehe Abbildung 4.1.1). Weil dabei mehrere Ähnlichkeitsmaße gleichzeitig Betrachtung finden, ist

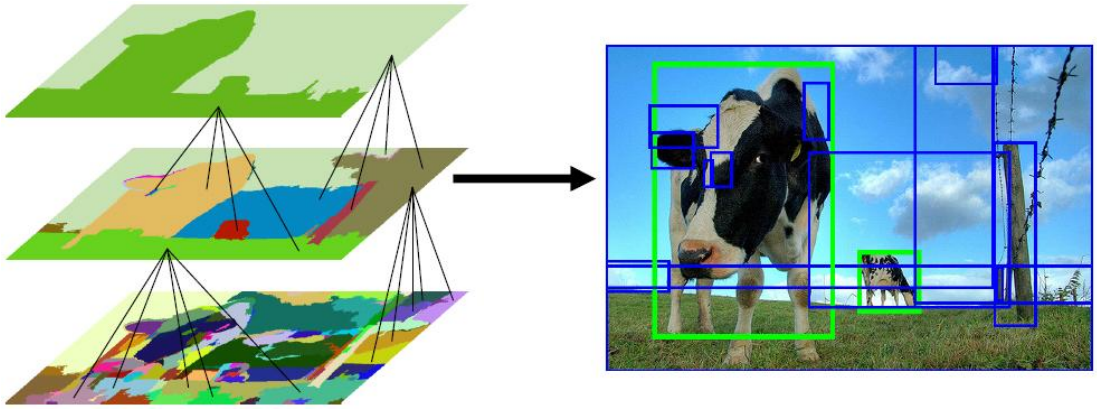


Abbildung 4.1.1: Visualisierung des Selective Search Algorithmus aus [USGS13]

es wichtig, dass diese Metriken einheitlich in einen gemeinsamen Wertebereich (zum Beispiel $[0, 1]$) normiert sind.
Dieses Vorgehen wird wiederholt, bis nur noch ein einziges Segment existiert, welches das gesamte Bild überdeckt. Aus der Hierarchie an Segmentierungen, vom übersegmentierten Anfangsbild bis zum letzten Bild, das nur noch ein Segment enthält, werden dann alle noch nicht enthaltenen Segmente der Ergebnismenge hinzugefügt. Diese enthält damit alle Anfangssegmente und ihre Verschmelzungen bis hin zu dem einzelnen großen Segment am Ende des Verfahrens. Aus dieser Menge werden dann die Bounding Boxen extrahiert. Im R-CNN dienen diese Bounding Boxen als Eingabe für die Merkmalsextraktion.

Eine weitere Möglichkeit Bildbereichsvorschläge zu berechnen sind Region Proposal Networks (RPNs). Neuronale Netze können dazu eingesetzt werden, Bounding Boxen mit einem Objectness-Wert zu berechnen, wie in [RHGS15b] gezeigt wurde. Dieser Wert stellt ein Maß dafür dar, wie wahrscheinlich es ist, dass sich in dem Bildbereich ein Objekt befindet. Eine Anzahl der Bildbereiche mit den höchsten Objectness-Scores dient dann als Basis für die Klassifizierung der abgebildeten Objekte, vergleichbar mit dem Verfahren, das Selective Search für die Bildbereichsvorschläge einsetzt.
Ein Unterschied zwischen Region Proposal Networks und Selective Search besteht in der Art der beiden Verfahren. Während Selective Search heuristische Annahmen über die Beschaffenheit von Objekten mithilfe der Ähnlichkeitsmaße macht, lernt das Region Proposal Network Vordergrund und Hintergrund voneinander zu unterscheiden und Rahmen um Objekte zu finden. Das heißt, dass Region Proposal Networks wie

das Convolutional Neural Network auch mit annotierten Beispielen trainiert werden muss. Selective Search hingegen ist unüberwacht, kommt also ohne Training und damit ohne annotierte Trainingsdaten aus und funktioniert grundsätzlich auch für vorher unbekannte Objektklassen.
Ein Vorteil von RPNs liegt darin, dass sich das Region Proposal Network und das Convolutional Neural Network für die Objektklassifizierung viele Layer und damit Berechnungen teilen können. Deshalb werden für die Bildbereichsvorschläge mithilfe von RPNs weniger zusätzliche Zeit als beim Selective Search Verfahren benötigt, bei dem die Berechnung der Bildbereiche einen bedeutenden Teil der Detektionszeit ausmacht. Da für das Training des Convolutional Neural Network annotierte Trainingsdaten vorhanden sein müssen, stört diese Voraussetzung bei den RPNs eher wenig, zumal das Training des RPN parallel zu dem des CNN stattfinden kann.
Zusammen mit einigen weiteren Modifikationen am Objekterkennungsverfahren können mit RPNs verbesserte Erkennungsraten bei kleinerer Trainings- und Detektionsdauer erreicht werden. Aus diesem Grund wird das Verfahren von Ren et al. „Faster R-CNNs“ genannt. Während mit RPNs cirka 5 Bilder pro Sekunde verarbeitet werden können (mit dem VGGNet auf einer K40 GPU) sind es bei den ursprünglichen R-CNNs die Selective Search einsetzen nur etwa 0,5 Bilder pro Sekunde [RHGS15b]. Damit ist insbesondere Punkt 3 der Anforderungen an den Segmentierungsalgorithmus von Uijlings et al. kritisch zu sehen: Hohe Geschwindigkeit der Berechnung. Wie gezeigt verzögert die Segmentierung mittels Selective Search das gesamte Verfahren deutlich. Die Erkennungsleistung des Objekterkennungsverfahrens mit Selective Search beziehungsweise Region Proposal Networks wird in Kapitel 6.4.1 evaluiert.

4.2 MERKMALSEXTRAKTION

Sind Eingabebild und Bildbereichsvorschläge mit möglichen Objekten bekannt, kann die Merkmalsextraktion erfolgen. Ziel ist es für jedes potenzielle Objekt (jeden Bildbereich aus der Segmentierung) aus dem Bild einen Merkmalsvektor zu berechnen, sodass anhand von diesem Vektor in der Klassifikation die zugehörige Klasse berechnet werden kann. Vektoren von Objekten die zu einer Objektklasse gehören, sollten sich im Merkmalsraum also deutlich von Vektoren von Objekten anderer Klassen unterscheiden lassen.
Bevor der Bildausschnitt durch das Neuronale Netz verarbeitet werden kann, wird zuvor noch eine Transformation der Bildregionen durchgeführt, was die Klassifikation erleichtern soll. Dabei gibt es verschiedene Optionen:

1. Erweiterung
   Eine Möglichkeit der Vortransformation besteht im Erweitern des betrachteten Bildbereichs wie in Bild 1 aus Abbildung 4.2.1 zu sehen. Dieses Vorgehen vermindert ein Abschneiden von Teilen des abgebildeten Objekts, falls der Bildbereich das Objekt nicht perfekt überdeckt. Des Weiteren liefert es der anschließenden Verarbeitung durch das Convolutional Neural Network Kontextinformationen zum Hintergrund des Objekts, was sich je nach Datensatz ebenfalls in einer gesteigerten Erkennungsleistung bemerkbar machen kann.

2. Subtraktion des Mittelwerts
   Bei der Mean-Subtraction gibt es wiederum mehrere Varianten. Es kann der mittlere Farbwert aller Bilder des Trainingsdatensatzes (siehe VGGNet [SZ14]), das mittlere Bild aller Bilder des Trainingsdatensatzes (siehe AlexNet [KSH12]) oder der mittlere Farbwert des aktuellen Eingabebildes vom Eingabebild subtrahiert werden. Durch all diese Verfahren wird der Farbwertbereich der Eingabe näher um 0 zentriert, was das Training des CNNs erleichtern soll.

3. Entfernung des Hintergrunds
   Geschieht die Segmentierung Pixel-weise kann diese Information dazu genutzt werden, lediglich den Bildbereichsanteil durch das CNN verarbeiten zu lassen, der auch zum Objekt gehört (siehe Bild 2 aus Abbildung 4.2.1). Dazu kann der Hintergrund durch die mittlere Farbe ersetzt werden. Mit diesem Verfahren können Störquellen wie andere Objekte im Bildbereich entfernt werden.

Da die Größe der Ausgaben der Convolutional und Pooling Layer abhängig ist von der Größe des Eingabebildes, die Fully Connected Layer jedoch eine feste Eingabegröße haben, muss die Größe des Eingabebildes normalisiert werden. Hierzu werden die horizontale und vertikale Seite des Bildes unabhängig voneinander auf die Größe 227×227 im AlexNet beziehungsweise 224×224 im VGGNet skaliert, wie in Bild 3 aus Abbildung 4.2.1 zu erkennen.
In den Experimenten von Girshick et al. zeigte sich eine Kombination aus einer Vergrößerung des Bildbereichs um 16 Pixel in alle vier Richtungen und Subtraktion des Mittelwerts aller Pixel des Trainingsdatensatzes (siehe [KSH12]) als besonders effektiv. Dieses Vorgehen übertraf andere getestete Kombinationen um 3-5% Mean Average Precision.

Der Bildbereich wird im Anschluss an die Vortransformation durch ein Convolutional Neuronal Network wie zum Beispiel dem AlexNet oder VGGNet (siehe Kapitel 3.4) verarbeitet. Die Layer im Netz lernen einfache bis komplexe Strukturen in Abhän-

Abbildung 4.2.1: Verschiedene Möglichkeiten der Vortransformation von Bildbereichen

gigkeit von der Tiefe und der damit verbundenen Größe des Receptive Fields. Die ersten Convolutional Layer des Netzes lernen Merkmale wie gerichtete Kanten, Ecken oder Punkte des Bildes. Dies ist oben in Abbildung 4.2.2 eines Neuronalen Netzes (für die Objektklasse Gesichter) zu erkennen. Mehrere dieser Merkmale bilden in den darauffolgenden Layern zusammengenommen die Basis für komplexere Merkmale, die im Bildbereich erkannt werden (siehe Abbildung 4.2.2 Mitte). Hier sind unter anderem Filter für Objektteile wie Augen oder Mund im abgebildeten Fall zu finden, die in den Convolutional Layern hinten im Netz dann zu Filtern für vollständige Gesichter zusammengesetzt werden (siehe Abbildung 4.2.2 unten). Dieses Verfahren funktioniert für alle anderen Objektklassen in gleicher Weise.
Aus den Aktivierungen dieser komplexen Filter der einzelnen Objektklassen in den hinteren Convolutional Layern, kann dann mithilfe von Fully Connected Layern wie in Kapitel 3.2.3 beschrieben ein Merkmalsvektor berechnet werden, der zur letztendlichen Klassifikation des abgebildeten Objektes dient. Dieser Vektor hat im Fall des AlexNet und VGGNet 4096 Dimensionen.

Die Performanz der Merkmalsextraktion wird entscheidend durch den Trainingsvorgang beeinflusst. Der von Girshick et al. angestrebte einfache Übergang von der Klassifikation zur Detektion schlägt sich auch hier nieder. Da nicht ausreichend annotierte Bilder für die Detektion verfügbar sind, um tiefe Neuronale Netze wie das

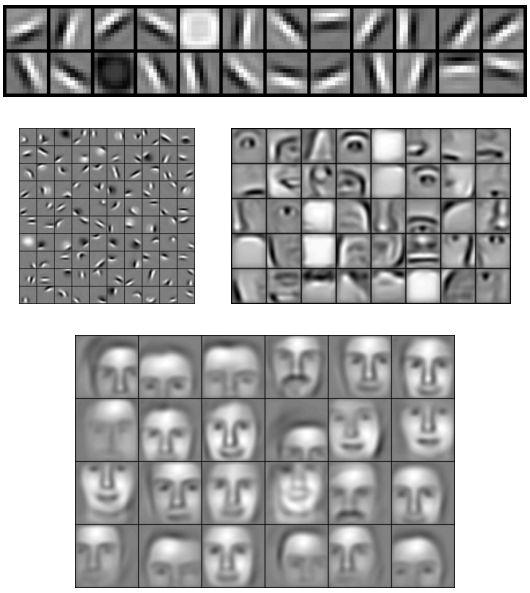


Abbildung 4.2.2: Visualisierung der gelernten Filter von Convolutional Layern vorne (oben/-links) in einem CNN für die Gesichtserkennung und Layern weiter hinten (rechts/unten). Grafiken entnommen aus [LGRN09]

AlexNet oder VGGNet zu trainieren, nutzen die Autoren einen zweiteiligen Trainingsvorgang unter der Hypothese, dass die vom Netz gelernten Merkmale für die Klassifikation auch für die Detektion verwendbar sind. Zunächst wird das Neuronale Netz also für den Klassifikations-Task auf dem ImageNet [RDS+15] trainiert und das so vor-trainierte Netz dann auf den Datensätzen für die Detektion trainiert. Dies wird als Pretraining mit anschließendem Fine-tuning bezeichnet. Es ist wichtig, Bilder aus ähnlichen Domänen zu verwenden, da die gelernten Merkmale für die Bilder aus dem Datensatz für die Klassifikation unter Umständen nicht zu den Bildern aus dem Datensatz für die Detektion passen. Dieses zweistufige Training hat die Anwendung tiefer Convolutional Neural Networks für die Objektdetektion auf Datensätzen mit so wenig annotierten Bildern wie dem PascalVOC [EGW+12] oder SUN [XHE+10] Datensatz überhaupt erst möglich gemacht.
Da das Neuronale Netz bei der Merkmalsextraktion mit der potenziellen Ungenauigkeit der Bildbereichsberechnung umgehen muss, liegt ein weiteres Hauptaugenmerk beim Training des Neuronalen Netzes auf der Definition von Positiv- und Negativbeispielen. Neben den Ground-Truth Annotationen, also den Positivbeispielen für alle Objektklassen, werden Negativbeispiele für das Training der Hintergrund-Klasse

für alle Bildbereiche, die keines der gesuchten Objekte beinhalten, benötigt. Ein Problem besteht allerdings in der Definition von Positiv- und Negativbeispielen, wenn Bildbereichsvorschläge nur eine teilweise Überlappung mit einem Bildbereich aus der Ground-Truth Annotation besitzen. Um diesen Konflikt zu lösen wird ein einfacher Schwellwert angesetzt: Bei einer Überlappung von mehr als 50% mit einer der annotierten Bildbereiche wird der Bildbereichsvorschlag als Positivbeispiel für die entsprechende Klasse der Annotation, sonst als Negativbeispiel gewertet. So kann das Netz darauf trainiert werden auch Objektteile zu erkennen, wenn die Bildbereichsvorschläge etwas unpräzise sind.

4.3 KLASSIFIKATION

Mithilfe des Klassifikators soll für den vom Convolutional Neuronal Network berechneten Merkmalsvektor die korrekte Klasse des abgebildeten Objektes bestimmt werden. Für die 1 aus K Notation werden $N+1$ Pseudowahrscheinlichkeiten berechnet, die angeben, ob ein Objekt der N gesuchten Objektklassen oder Hintergrund (beziehungsweise ein Objekt das zu keiner der gesuchten Klassen gehört) im Bildbereich abgebildet sind. Zwei häufig verwendete Klassifikatoren sind SoftMax Layer (siehe Kapitel 3.2.4) am Ende einer Netzarchitektur oder Support Vector Machines.

Im Gegensatz zum SoftMax Layer sind Support Vector Machines (SVMs) kein Teil des Neuronalen Netzes sondern ein mathematisches Verfahren, bei dem der vom CNN berechnete Merkmalsvektor als ein Punkt in einem Merkmalsraum interpretiert wird, wie in Abbildung 4.3.1 zu erkennen ist. Pro Objektklasse plus Hintergrund wird eine lineare SVM trainiert, die versucht den Merkmalsraum durch eine (Hyper-)Ebene (orange) in zwei Teile aufzuteilen: Zugehörige (grüne) oder nicht zugehörige (rote) Merkmalsvektoren zur betrachteten Objektklasse. Der Abstand des Merkmalsvektors zur Hyperebene korrespondiert dann mit der Pseudowahrscheinlichkeit für die Zugehörigkeit zu der entsprechenden Objektklasse der SVM. Dafür muss die SVM für die Erkennung ihrer jeweiligen Objektklasse trainiert werden. Im Training werden positive und negative Beispiele betrachtet und es wird versucht

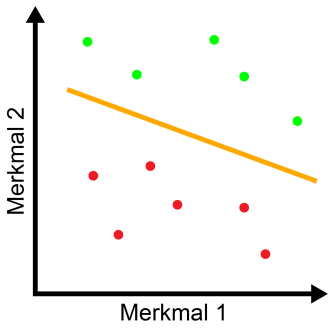


Abbildung 4.3.1: Visualisierung eines zweidimensionalen Merkmalsraums mit Hyperebene

eine Hyperebene zu finden, die den kleinsten Abstand zwischen sich und allen Merkmalsvektoren maximiert. Dieses Optimierungskriterium wird Max-Margin Kriterium genannt.

Wichtig ist, dass zum Trainingszeitpunkt des Neuronalen Netzes zur Klassifikation ein SoftMax Layer eingesetzt wird. Um im Anschluss auf SVMs überzugehen, wird das SoftMax Layer vom fertig trainierten Neuronalen Netz abgeschnitten und die Ausgabe des letzten 4096-dimensionalen Fully Connected Layers als neues Ausgabelayer verwendet. Erst dann können die SVMs trainiert und mit ihnen eine Klassifikation durchgeführt werden. Girshick et al. beschreiben in [GDDM15], dass sich SoftMax Layer und SVMs in ihrer Leistung nicht wesentlich unterscheiden und setzen die SVMs vor allem ein, um eine bessere Vergleichbarkeit mit vorherigen Verfahren, wie zum Beispiel den Part-based Models [FGMR10] herzustellen, die oftmals SVMs verwenden. Der Unterschied in der Erkennungsleistung zwischen SoftMax Layer und SVMs als Klassifikator wird in Kapitel 6.4.1 näher betrachtet.

Die Definition von Positiv- und Negativbeispielen weicht beim Training der SVMs von der im Training des Neuronalen Netzes ab. Hat im Training des Neuronalen Netzes ein Bildbereichsvorschlag mehr als 50% Überlappung mit einem Bildbereich aus der Annotation, wird er als Positiv-, sonst als Negativbeispiel betrachtet. Bei dem Training der SVMs hat sich eine differenziertere Regel als effizienter erwiesen: Bildbereichsvorschläge mit einer Überlappung von mehr als 50% gelten als Positivbeispiele und solche mit Überlappung unter 30% als Negativbeispiele. Alle weiteren Bereiche mit einer Überlappung zwischen 30% und 50% werden verworfen. Girshick et al. [GDDM15] zeigen, dass diese Vorgehensweise das erzielte Ergebnis um 4–5% mAP verbessert.

5 PART-BASED MODELS

Um die Leistungsfähigkeit von Neuronalen Netzen einordnen zu können, ist ein Vergleich mit konkurrierenden Objekterkennungsverfahren wichtig. Ein solches Verfahren sind die Part-based Models (PBMs) von Felsenzwalb et al. [FGMR10]. Die Part-based Models stellten bis zu der Entwicklung der R-CNNs den aktuellen Stand der Forschung in der Objekterkennung dar. Im Gegensatz zu den R-CNNs basiert das Verfahren nicht auf einer Merkmalsberechnung durch Neuronale Netze, sondern es werden im Training Modelle von Objekten und deren Einzelteilen gelernt, die später im Bild erkannt werden sollen.

5.1 FUNKTIONSWEISE

Part-based Models basieren bei der Erkennung auf HOG (Histogram of Oriented Gradients) Deskriptoren [DT05]. Hierbei werden die Richtungen der Gradienten in kleinen rechteckige Bildbereichen (Zellen) betrachtet. Jeder Gradient aus der Zelle wird durch Quantisierung einer von acht Richtungen (im 45° Abstand) zugeordnet und in einem eindimensionalen Histogramm zusammengefasst, welches als Deskriptor für diese Zelle dient. Ein Beispiel der HOG-Berechnung für ein Beispielbild ist in Abbildung 5.1.1 zu sehen.

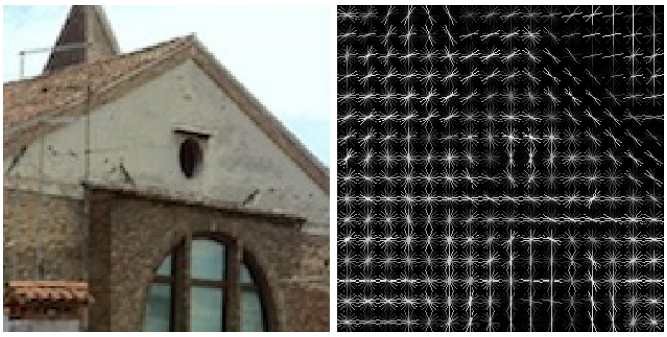


Abbildung 5.1.1: HOG Deskriptoren für ein Beispielbild. Die Zellgröße beträgt 8×8 Pixel. Grafik stammt von [VLF13]

Im Training lernen Part-based Models anhand von annotierten Trainingsdaten ein Modell der zu erkennenden Objektklasse. Die Bewertung übernimmt eine spezielle Form der Support Vector Machines (siehe Kapitel 4.3), genannt Latent SVM [FGMR10], welche die Ähnlichkeit der HOG Repräsentation vom Bildausschnitt zum Modell bewerten. Dabei wird nicht nur eine SVM für das gesamte Objekt als Root-Filter gelernt, sondern zusätzlich auch eine feste Anzahl an SVMs für die einzelnen Objektteile (Part-Filter). Die Verteilung dieser Teile wird beim Training heuristisch über das Objekt initialisiert, wobei ein Teil entweder auf der vertikalen Mittelachse platziert wird oder links und rechts von der Mittelachse zwei Teile achsensymmetrisch verteilt werden. Es werden dann durch eine erschöpfende Suche solche Orte für Teile bevorzugt die besonders viele und starke Kanten besitzen, um die spätere Wiedererkennung zu vereinfachen. Da ein Teil sich allerdings nicht bei jedem Objekt an genau der selben Stelle befindet, werden zusätzlich zu den Teilepositionen und Filtern noch Verschiebungskosten gelernt, die eine zu starke Abweichung der Ist- von der Soll-Position eines Teils relativ zu der Objekt (Root-Filter) Position bestraft. Ein vollständiges Modell einer Objektklasse besteht damit aus Root-Filter, mehreren Part-Filtern und deren Verschiebungskosten, zu sehen in Abbildung 5.1.2.

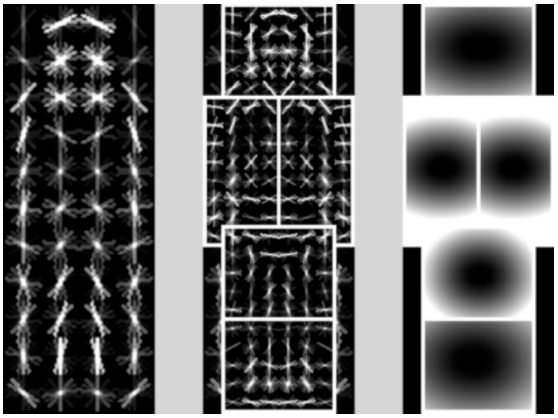


Abbildung 5.1.2: Beispielmodell der Objektklasse Person aus [FGMR10]

Zum Testzeitpunkt werden alle Root- und Part-Filter aller zu erkennenden Objektklassen im Sliding-Window Verfahren über die HOG Darstellung des Testbildes geschoben. Ein Ablaufdiagramm zu diesem Erkennungsverfahren ist in Abbildung 5.1.3 zu finden. Mithilfe der Bewertung der SVMs wird für jeden Bildausschnitt eine Scoremap pro Objektklasse erzeugt. Im Anschluss wird für jede Root-Filter Position in Abhängigkeit von den Part-Filter Aktivierungen und deren Verschiebungskosten eine kombinierte

Aktivierung ausgerechnet. Da dies pro Root-Filter Position geschieht, können auch mehrere Objektinstanzen im Bild erkannt werden. Hohe Ausschläge in der Heatmap der kombinierten Werte korrespondieren dann mit einer hohen Wahrscheinlichkeit, dass an dieser Stelle im Bild eine Instanz der entsprechenden Objektklasse abgebildet ist.

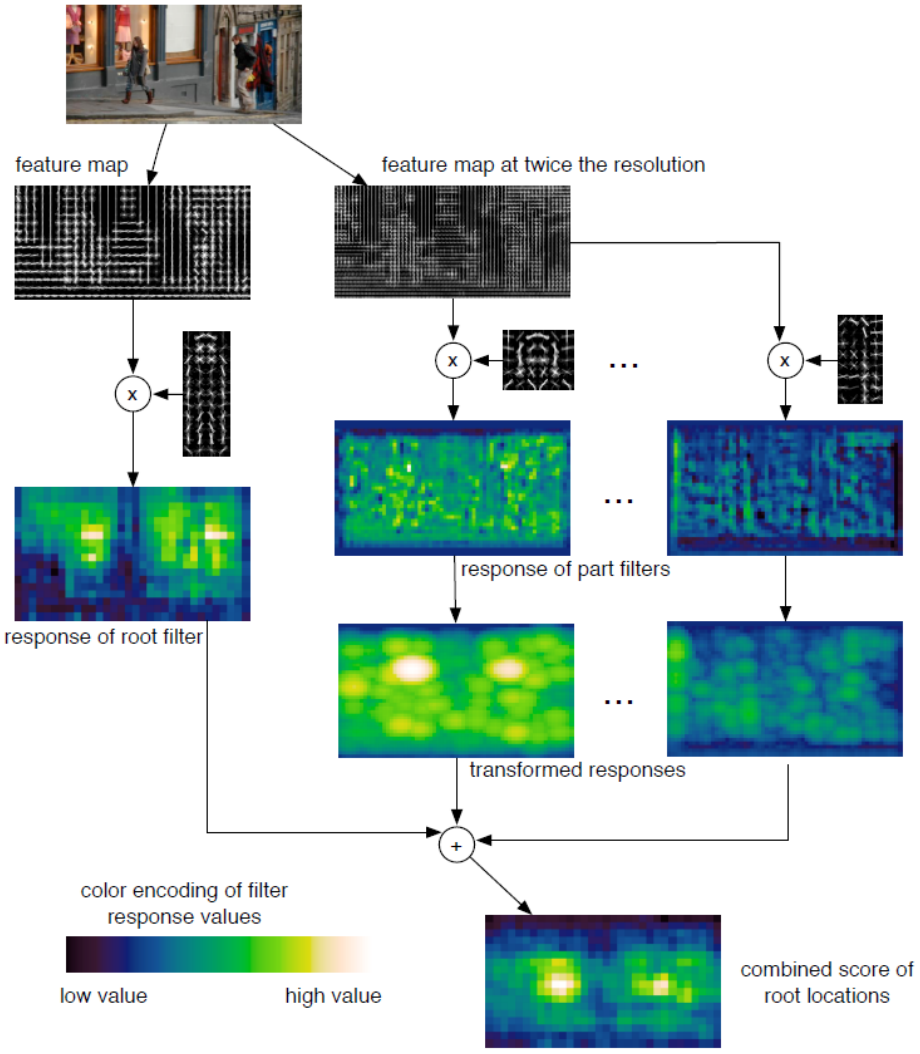


Abbildung 5.1.3: Ablaufdiagramm einer Objekterkennung mit Part-based Models aus [FGMR10]

5.2 VERGLEICH MIT R-CNNS

Region-based Convolutional Neural Networks und Part-based Models unterscheiden sich im Training und Erkennungsverfahren in vielen wesentlichen Punkten. Bereits die Bildbereichsberechnung der R-CNNs findet bei den PBMs keine direkte Entsprechung. Während bei den R-CNNs lediglich für mögliche Objektpositionen die Merkmalsberechnung durchgeführt wird, werden die HOG Deskriptoren der PBMs immer für das gesamte Bild berechnet.
HOG Deskriptoren sind eine einfache Art Merkmal, das im Gegensatz zu den Gewichten in Neuronalen Netzen nicht gelernt wird, sondern von vornherein festgelegt ist. In R-CNNs hingegen werden die anzuwendenden Merkmale sowohl im Training gelernt, sodass sie auf die zu erkennenden Objektklassen zugeschnitten sind, als auch in tiefer liegenden Layern zu komplexeren Merkmalen zusammengefasst, was die Unterscheidung komplexerer Strukturen ermöglicht.
Auch die Klassifikation der beiden Verfahren weicht voneinander ab. Zwar können wie in PBMs auch in Neuronalen Netzen SVMs zur Klassifikation eingesetzt werden, jedoch ist das Klassifikationsverfahren in Neuronalen Netzen erheblich effizienter. Hier muss lediglich für den von jedem Bildbereichsvorschlag berechneten Merkmalsvektor eine Klassifikation mithilfe von einer SVM pro Objektklasse durchgeführt werden. Bei den Part-based Models wird nicht nur für jede Objektposition das Vorkommen von jeder Objektklasse (Root-Filter) überprüft, sondern zusätzlich noch die Vorkommen aller Parts (Part-Filter) aller Objektklassen. Sowohl die Anzahl der zu überprüfenden Objektpositionen als auch die Anzahl der anzuwendenden SVMs pro Objektposition sind bei den Part-based Models also erheblich größer.
Bei den Part-based Models werden stärkere Annahmen über die Beschaffenheit der Objekte gemacht. Nicht nur die berechneten Merkmale werden mit den HOG Deskriptoren festgelegt, statt sie wie in Neuronalen Netzen Datensatz-abhängig zu lernen, sondern auch die Verteilung der Parts erfolgt vertikal achsensymmetrisch, was unter Umständen nicht für alle Objektklassen geeignet ist. Umgekehrt sind die erforderliche Menge an Trainingsdaten bei Neuronalen Netzen und der Trainingsaufwand entsprechend höher. Durch den Einsatz moderner GPUs, welche sich gut für die notwendigen Berechnungen in CNNs eignen, hält sich der erforderliche Zeitaufwand beim Training Neuronaler Netze jedoch in Grenzen.

6 EVALUATION

Mit der Festlegung des Objekterkennungsverfahrens soll nun dessen Leistung auf verschiedenen Datensätzen und Objekterkennungsszenarien analysiert werden. Dazu gehört zum einen die Bestimmung der optimalen Parameterkonfiguration wie zum Beispiel die Wahl der Neuronalen Netzarchitektur oder die Anzahl an Trainingsiterationen. Zum anderen soll ein Vergleich mit den Part-based Models hergestellt werden, indem die Ergebnisse auf den PascalVOC und SUN Datensätzen gegenübergestellt werden.

6.1 IMPLEMENTIERUNGSDETAILS

Als Grundlage für das Testen der verschiedenen Konfigurationen diente die Python-Implementation der Faster-R-CNNs [RHGS15b] von Ren et al., in der das Deep Learning Framework Caffe [JSD+14] eingesetzt wird. Caffe bietet die Möglichkeit, Neuronale Netze mit geringem Aufwand mithilfe von Python und Matlab Interfaces trainieren und testen zu können. Zur Beschreibung der Parameter des Lernvorgangs und der Netzarchitektur werden Googles Protocol Buffers eingesetzt, was Konfigurationsänderungen möglich macht ohne den eigentlichen Programmcode ändern zu müssen. Auch zwischen den GPU und CPU Implementationen kann einfach gewechselt werden. Gleichzeitig ist Caffe nach Aussage der Entwickler wahrscheinlich das schnellste verfügbare Framework für den Einsatz von Convolutional Neural Networks [JSD+14].

Die zwei zentralen Bestandteile eines Neuronalen Netzes in Caffe sind Layer und sogenannte Blobs. Die Layer (siehe 3.2) beinhalten die Layerparameter und Rechenvorschriften, wie mit den Eingabedaten umgegangen werden soll, um die Ausgabe des Layers zu erhalten. Jedes Layer implementiert drei Komponenten: Das Setup, welches beschreibt wie die Gewichte und Kanten eines untrainierten Layers initialisiert werden, den Forward Pass, das heißt der Rechenvorschrift, wie die Eingabe in die Ausgabe des Layers überführt wird und dem Backward Pass für das Training, in dem beschrieben ist wie der entstandene Fehler mit den Layer-Gewichten verrechnet und an das vorherige Layer weitergereicht wird. Die Daten für die Operation sind dafür in Blobs gespeichert. Blobs sind im Fall der Bildverarbeitung normalerweise 4-dimensionale

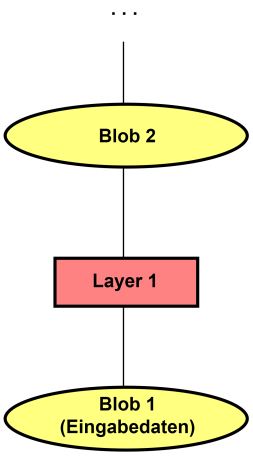


Abbildung 6.1.1: Abbildung der Blob-Layer Interaktion im Caffe Framework

Arrays, wobei die vier Dimensionen ($N \times K \times H \times W$) dabei der Batchgröße N, Anzahl an (Farb-)Kanälen K, Höhe H und Breite W entsprechen. Ein Batch ist eine Menge von Bildern oder anderen Daten, die in einem Schritt verarbeitet werden. Im Training geschieht die Anpassung der Gewichte bei der Backpropagation gemittelt über die Fehler aller Bilder des Batches, um den Einfluss von Sonderfällen bei den Bildern zu mindern. Jedes Layer besitzt einen Bottom Blob, aus dem die Eingabedaten entnommen werden und einen Top Blob, in dem die Ausgabe gespeichert wird (siehe Abbildung 6.1.1). Wird als Bottom Blob eines Layers der Top Blob eines anderen gewählt, erhält man eine Netzarchitektur aus zwei Layern. Mehrfach angewendet können auf diese Weise schnell komplexe und tiefe Neuronale Netze implementiert werden.

6.2 DATENSÄTZE

Für die Evaluation und den Vergleich verschiedener Verfahren werden einheitliche Datensätze benötigt. Diese stellen Trainings- und Testdatensätze mit den notwendigen Annotationen für das überwachte Lernen zur Verfügung. Einige Bilddatensätze für die Objekterkennung sollen im Folgenden vorgestellt werden.

6.2.1 *Pascal Visual Object Classes*

Das **Pascal** **V**isual **O**bject **C**lasses Projekt (PascalVOC) [EEG+15] stellt standardisierte Bilddatensätze und Programmierschnittstellen für die Klassifikation und Objektde-

	Training		Validierung		Test	
	Bilder	Objekte	Bilder	Objekte	Bilder	Objekte
PascalVOC 2007	2.501	6.301	2.510	6.307	4.952	12.032
PascalVOC 2012	5.717	13.609	5.823	13.841	3.422	?*

* Der Test-Datensatz von 2008 bis 2012 ist nicht öffentlich.

Tabelle 6.2.1: Anzahl der Bilder und Objekte in den PascalVOC 2007 und 2012 Datensätzen

tektion in Bildern zur Verfügung. Von 2005 bis 2012 wurden jährlich die PascalVOC Challenges durchgeführt, bei dem das jeweils beste Verfahren in der Segmentierung, Klassifikation und Objektdetektion gesucht wurde. Dazu existieren jährlich wechselnde zusätzliche Aufgaben wie zum Beispiel Aktionsklassifikation, bei der nach der von Personen im Bild durchgeführten Tätigkeit gesucht wurde oder die *Person Layout Taster* Competition, bei der die Teile einer Person (Hand, Fuß, Kopf, usw.) detektiert werden sollten. Eine Übersicht über die Größe der Datensätze von den Jahren 2007 und 2012 findet sich in der Tabelle 6.2.1. Mithilfe dieser Datensätze sollen Verfahren trainiert und getestet werden, die im Klassifikationstask die An- oder Abwesenheit von 20 Objektklassen bestimmen beziehungsweise im Detektionstask aus den selben 20 Klassen Objekte lokalisieren und klassifizieren. Bemerkenswert ist die Häufigkeit des Vorkommens der Objektklasse Person. Von den 5.011 Bildern und 12.608 abgebildeten Objekten aus den Trainings- und Validierungsdatensätzen im PascalVOC 2007 enthalten 2.008 Bilder insgesamt 4.690 Personen. Es sind also über 37% der abgebildeten Objekte Personen.
Die Bilder der PascalVOC Datensätze sind, wie in Abbildung 6.2.1 zu erkennen ist, thematisch und von der Art der Aufnahme unstrukturiert. Es sind sowohl Bilder von einzelnen Objekten oder Objektgruppen als auch von ganze Szenen vorhanden.

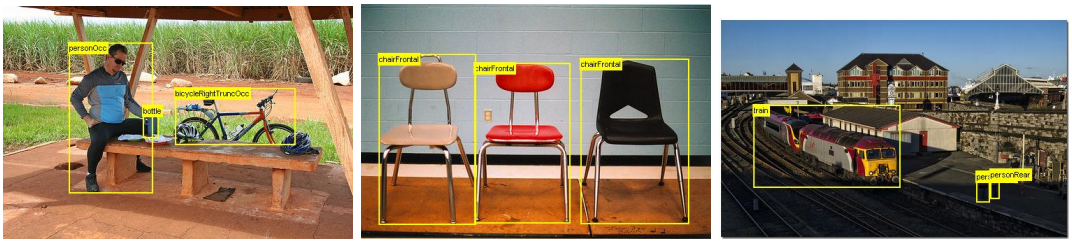


Abbildung 6.2.1: Annotierte Beispielbilder aus dem PascalVOC 2012 Datensatz [EGW+12].

Quelle der Bilder sind der Internetdienst flickr und der Microsoft Research Cambridge (MSRC) Datensatz, die Annotationen wurden händisch hinzugefügt.

6.2.2 *Scene Understanding*

Der **S**cene **Un**derstanding (SUN) Datensatz [XHE+10] hat zum Ziel, Forschern aus den verschiedensten Bereichen eine umfassende Datenbasis an annotierten Bildern zur Verfügung zu stellen. Im Gegensatz zu den Bildern des PascalVOC Datensatzes hat der SUN Datensatz, wie der Name bereits impliziert, einen klaren Fokus auf Bildern von natürlichen Szenen.
Die Bilder sind klassifiziert nach Szenenkategorien aus dem WordNet [Mil95], einem Wörterbuch das versucht alle englischen Wörter nach ihrer Semantik in eine hierarchische Ordnung (Baumstruktur) zu bringen. Jede Szenenkategorie stellt dabei eine Beschreibung des Aufnahmeorts dar. Zusätzlich zur Kategorisierung der Szene werden auch Annotationen zu allen dargestellten Objekten zur Verfügung gestellt. Diese enthalten sowohl eine Segmentierung als auch die entsprechende Objektklasse, welche mithilfe von Crowd-Sourcing erstellt wurden. Jeder kann mit der sogenannten LabelMe Toolbox [RTMF08] an den Annotationen mitwirken.
Der eigentliche SUN Datensatz wird ständig erweitert. Für den Vergleich verschiedener Klassifikations- und Objekterkennungsverfahren wird aus diesem Grund der SUN2012 Datensatz angeboten, welcher statisch ist und zudem nur Bilder aus Szenenkategorien enthält, die mindestens 100 Bilder beinhalten. Eine Übersicht über die Größe der jeweiligen Datensätze findet sich in Tabelle 6.2.2.

	Bilder	Objekte	Szenenkategorien	Objektkategorien
SUN gesamt*	131.067	313.884	908	4.479
SUN2012	16.873	194.953	397	3.005

Tabelle 6.2.2: Anzahl der Bilder und Objekte im SUN Datensatz. * Stand 02.09.2016

Im Beispielbild 3 aus Abbildung 6.2.2 ist zu sehen, dass einzelne Bilder des SUN Datensatzes oftmals unzählige Objekte enthalten. Dies steht im Gegensatz zum PascalVOC Datensatz, dessen Bilder vielfach nur einzelne Objekte oder kleine Objektgruppen abbilden. Hinzu kommt, dass in den PascalVOC Challenges die gesuchte Objektvielfalt mit den 20 wohldefinierten Objektklassen erheblich geringer ist. Diese Eigenschaften erschweren die Objekterkennung auf dem SUN2012 Datensatz erheblich.

Abbildung 6.2.2: Drei Beispielbilder aus dem SUN 2012 Datensatz [XHE+10]

6.2.3 *ImageNet*

Das ImageNet folgt wie der SUN Datensatz der Struktur des WordNet [Mil95]. Der Fokus liegt hier allerdings nicht wie beim SUN Datensatz auf Bildern von Szenen, sondern es sind viele verschiedene Arten von Bildern zu den Wörtern aus dem WordNet enthalten. Zum aktuellen Zeitpunkt enthält das ImageNet 14.197.122 Bilder aus 21.841 Kategorien. 1.034.908 Bilder enthalten neben der Annotationen der Szene auch Annotationen zu den abgebildeten Objekten. Der Datensatz ist damit im Vergleich erheblich größer als beispielsweise der PascalVOC und SUN Datensatz und wird im Kontext der R-CNNs deshalb häufig für das Pretraining (siehe Kapitel 4.2) des Neuronalen Netzes verwendet.
Es existiert wie bei den PascalVOC und SUN Datensätzen ein auf dem ImageNet basierender Wettbewerb für das beste Verfahren in verschiedenen Aufgaben, die **I**mageNet **L**arge **S**cale **V**isual **R**ecognition **C**hallenge (ILSVRC), für welche auch jeweils annotierte Test- und Trainingsdaten veröffentlich werden. Die höhere Anzahl an Bildern ermöglicht dabei auch eine größere zu erkennende Objektvielfalt. In der Objekterkennungsaufgabe müssen insgesamt 1000 verschiedene Objektklassen voneinander unterschieden werden.

6.3 METRIKEN

Um optimale Konfigurationen zu finden und einen Vergleich zwischen verschiedenen Objekterkennungsverfahren herstellen zu können werden einheitliche Metriken benötigt, die geeignete Eigenschaften der von den Verfahren produzierten Ergebnisse bewerten.

6.3.1 *Fehlerrate*

Die vielleicht einfachste Möglichkeit, die Klassifikationsleistung eines Verfahrens zu messen ist die Fehlerrate. Hier wir einfach der Prozentanteil der fehlerhaft klassifizierten Objekte ausgerechnet:

$$\text{Klassifikationsfehler KF} = \frac{\text{\# fehlerhaft klassifizierte Objekte}}{\text{\# Objekte insgesamt}}$$

Diese Metrik ist intuitiv ein geeignetes Maß für die Bewertung von Objekterkennungs- und Klassifikationsverfahren. Ein Verfahren, welches mehr Objekte korrekt klassifiziert ist wahrscheinlich besser als ein Verfahren mit einer höheren Fehlerrate. Ein Problem ergibt sich allerdings, wenn nicht nur eine einzelne Klassifikation als Rückgabe des Verfahrens gewertet wird, sondern eine nach Pseudowahrscheinlichkeit absteigend sortierte Liste an Klassifikationen.

6.3.2 *Intersection over Union*

Die Intersection over Union (IoU) ist eine Metrik für die Leistung eines Segmentierungs- und Detektionsverfahrens. Hier wird die Überdeckung der Fläche des erkannten Segments A mit dem korrekten Segment aus der Ground-Truth Annotation B bestimmt (siehe auch Abbildung 6.3.1):

$$\text{Intersection over Union IoU} = \frac{A \cap B}{A \cup B}$$

Statt Flächeninhalte auszurechnen kann die Berechnung bei Rasterbildern auch mithilfe der Bewertung der Pixel als true/false positives/negatives stattfinden (siehe [EEG+15]):

$$\text{Intersection over Union IoU} = \frac{\text{true positives}}{\text{true positives} + \text{false positives} + \text{false negatives}}$$

Je mehr sich die beiden betrachteten Segmente überdecken, desto kleiner ist der Unterschied zwischen dem Flächeninhalt des Schnitts der beiden Flächen im Vergleich zur Vereinigung.
Diese Metrik eignet sich lediglich für die Bewertung der Genauigkeit der Objektlokalisierung eines Detektionsverfahrens. Da für die Bewertung der Klassifikationsgenauigkeit eine Verbindung zwischen gefundenem Objekt und einem der Objekte aus den Ground-Truth Annotationen aufgebaut werden muss, wird hier oft der folgende

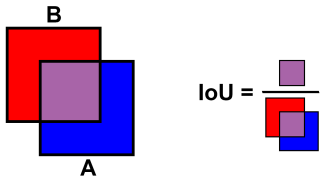


Abbildung 6.3.1: Berechnung der IoU

Ansatz gewählt: Einem gefundenen Objekt wird dasjenige Objekt aus der Ground-Truth Annotation zugeordnet, mit welchem es eine Intersection over Union von mehr als 50% hat. Existiert kein solches Objekt wird davon ausgegangen, dass Hintergrund beziehungsweise ein Objekt aus einer nicht betrachteten Objektklasse erkannt wurde.

6.3.3 *Mean Average Precision*

Je nach Problemstellung ist ein Verfahren, das für alle Objekte die korrekte Klassifikation weit vorne (aber nicht an Stelle 1) in der Rückgabeliste platziert, besser als ein Verfahren, das für einige wenige Objekte die korrekte Klassifikation an Stelle 1 der Rückgabeliste hat, für alle anderen Objekte jedoch gar keine passende Klassifikation findet.
Eine Metrik die genau dies implementiert, ist die (Mean) Average Precision (AP/mAP). Zur Berechnung der Average Precision werden die Precision und Recall Werte eines Ergebnisses benötigt. Die Precision P ist der Anteil korrekt erkannter Objekte unter allen *erkannten* Objekten während der Recall R der korrekt erkannte Anteil unter *allen* Objekten (der Annotation) ist:

$$P = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$
$$R = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Meist wird ein Schwellwert für die Pseudowahrscheinlichkeit verwendet, nach dem entschieden wird ob ein erkanntes Objekt der Rückgabeliste hinzugefügt wird oder nicht. Setzt man diesen Schwellwert niedrig an, werden also mehr Objekte als erkannt eingestuft, wird der Recall steigen, da sich mehr korrekt erkannte Objekte in der Rückgabeliste befinden. Gleichzeitig wird natürlicherweise auch die Precision sinken, da mehr fehlerhaft erkannte Objekte in der Rückgabeliste zu finden sein werden. Gute Objekterkennungsverfahren werden aber auch bei hohem Recall weiterhin eine hohe Precision zeigen. Zeichnet man die Precision-Recall Kurve eines Ergebnisses

(siehe Abbildung 6.3.2), ist die Fläche unterhalb dieser Kurve ein Maß für die Güte der Rückgabe.
Im PascalVOC Wettbewerb wurde bis zum Jahr 2009 folgende Formel zur Berechnung der Average Precision verwendet:

$$\mathrm{AP} = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \max_{\tilde{r}: \tilde{r} \geqslant r} p(\tilde{r})$$

mit $p(\tilde{r}) = \text{Precision @ Recall } \tilde{r}$

Es wird nur der monoton fallende Teil der Kurve an 11 Punkten (0, 0.1, ... 1) abgetastet, um die Auswirkung der „Sägezähne“ zu vermindern, die in sich keine interessante Information beinhalten und durch kleine Variationen in der Reihenfolge der Ergebnisse entstehen (siehe [EEG+15]).
In den Wettbewerben ab 2010 wurde die Art der Glättung etwas abgewandelt, da die Leistung von Verfahren mit geringer Average Precision so schlechter unterschieden werden konnten. In den Folgejahren wurde der Flächeninhalt unter dem monoton fallenden Teil der Kurve aller Datenpunkte verwendet. Auf die Rasterung an 11 Punkten wurde verzichtet.
Wird für jede Objektklasse eine Average Precision ausgerechnet und die Werte gemittelt, erhält man die Mean Average Precision. Diese ist ein Maß für die Leistungsfähigkeit des Verfahrens über alle Objektklassen.

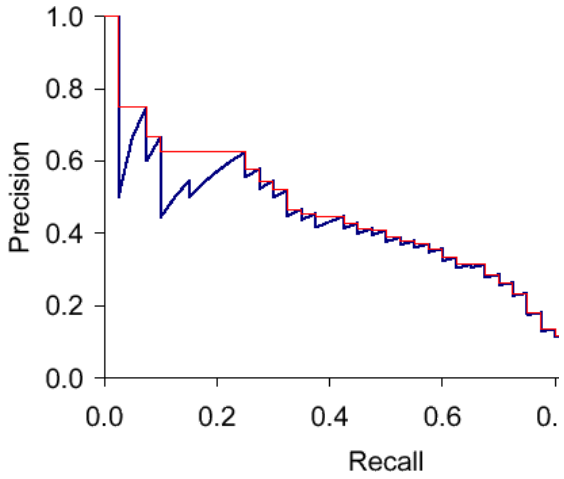


Abbildung 6.3.2: Precision-Recall Kurve aus [MRS08]

6.4 ERGEBNISSE

Im Folgenden soll die vorgestellte Methodik der Region-based Convolutional Neural Networks evaluiert und die auf den PascalVOC und SUN Datenbanken erzielten Ergebnisse diskutiert werden. Da die Wahl der Parameter wie beispielsweise die Lernrate, Anzahl an Iterationen beim Training oder Netzarchitektur essentiell für das erzielte Ergebnis ist, soll zunächst mithilfe einiger Versuche eine geeignete Parameterkonstellation gefunden werden. Verschiedene Variationen der Methodik wie zum Beispiel die Verwendung von Region Proposal Networks anstatt Selective Search zur Bildbereichsberechnung oder die Klassifikation mithilfe von Support Vector Machines anstelle der SoftMax Layer werden gegenübergestellt. Im Anschluss wird ein Vergleich der erzielten Ergebnisse mit dem konkurrierenden Verfahren Part-based Models hergestellt.
Als grundlegendes Verfahren, mit dem alle getesteten Variationen und Parameterkonstellationen im Training verglichen werden, dient die Konfiguration in Tabelle 6.4.1.

	Grundkonfiguration	Alternativen
Bildbereichsberechnung	Selective Search	Region Proposal Networks
Netzarchitektur	VGG16*	AlexNet*
Klassifikator	SoftMax	Support Vector Machines
Datensatz	PascalVOC 2007	SUN 2012

* Vortrainiert auf dem ImageNet, Datensatz von der ILSVRC 2012 1000 Klassen Klassifikationsaufgabe, siehe 6.2.3

Tabelle 6.4.1: Grundlegende Parameterkonfiguration für die Evaluation

6.4.1 *Ergebnisse auf dem PascalVOC 2007 Datensatz*

In den ersten Versuchen soll die Entwicklung des Netzwerkes beim Training unter verschiedenen Lernraten mit steigender Iterationszahl betrachtet werden. Dafür sind in den Graphen aus Abbildung 6.4.1 die Entwicklung der Mean-Average Precision in Abhängigkeit von der Anzahl der Iterationen für eine hohe Lernrate von 0,001 und eine niedrige von 0,0001 abgebildet. Es wurde ein Training des VGG16 Netzes auf dem PascalVOC 2007 Datensatz mit insgesamt 150.000 Iterationen durchgeführt, wobei alle 10.000 Iterationen eine Momentaufnahme des Neuronalen Netzes gespeichert und im Anschluss getestet wurde.

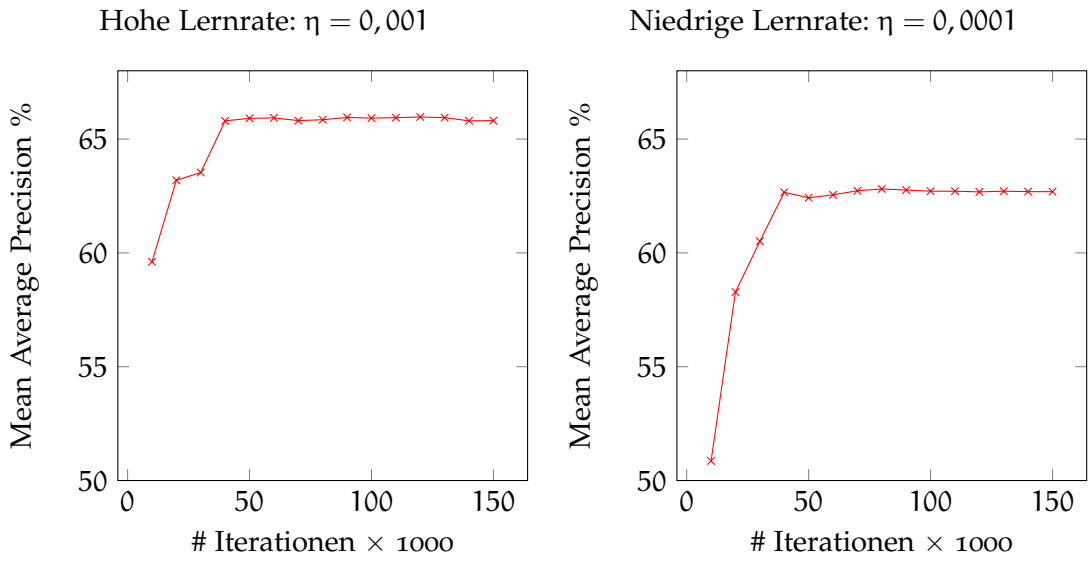


Abbildung 6.4.1: Gegenüberstellung der Mean Average Precision zur Anzahl an Trainings-Iterationen von Trainingsvorgängen mit verschiedenen Lernraten

Beide Kurven haben eine charakteristische Form, bei welcher der Zuwachs an Erkennungsleistung pro Iteration (gemessen an der Mean Average Precision) in den höheren Iterationsstufen abnimmt bis ein Plateau erreicht wird, bei dem mit steigender Iterationszahl keine weitere Verbesserung eintritt. Dies geschieht, da mit dem Backpropagation Algorithmus lokale Optima der Gewichtskonstellation des Neuronalen Netzes gefunden werden. Im linken Graph zur Lernrate von 0,001 verbessert weiteres Training die Erkennungsleistung des Netzes ab Iteration 40.000 nicht mehr wesentlich, diese schwankt in diesem Versuch dann lediglich noch in einem Bereich von $\pm 0,1\%$ um 65,9%.
Der rechte Graph zur niedrigeren Lernrate von 0,0001 zeigt einen ähnlichen Kurvenverlauf. Auffällig ist die deutlich schlechtere Erkennungsleistung bei Iteration 10.000 (50,86% bei $\eta = 0,0001$ gegenüber 59,61% bei $\eta = 0,001$), die direkt aus der niedrigeren Lernrate resultiert. Durch die kleinschrittigeren Anpassungen der Netzgewichte mit der kleineren Lernrate benötigt das Neuronale Netz mehr Iterationen, um passende Merkmale zu lernen. Trotzdem wird auch hier nach etwa 40.000 Trainingsiterationen keine weitere Verbesserung der Erkennungsleistung erreicht. Die beste Mean Average Precision liegt hier bei lediglich 62,6% $\pm 0,1\%$, also über 3% weniger als beim Training mit $\eta = 0,001$.

Interessant ist, dass der linke Graph zu $\eta = 0,001$ bei Iteration 20.000 und 30.000 ebenfalls einen flacheren Kurvenverlauf um 63,5% zeigt. Es ist möglich, dass das Neuronale Netz hier das gleiche lokale Optimum erreicht hat wie das, auf dem das Training mit $\eta = 0,0001$ konvergiert ist und dann mit Iteration 40.000 deutlich um 3% mAP darüber hinweg gesprungen ist.
Weitere Tests mit einer vergrößerten Lernrate von $\eta = 0,01$ beziehungsweise $\eta = 0,002$ führten zu einem fehlgeschlagenen Training. Wahrscheinlich ist, dass der Vanishing Gradient oder Exploding Gradient Effekt eingetreten ist, bei dem der ausgerechnete Gradient während der Backpropagation durch vielfache Multiplikation nach der Kettenregel mit sehr kleinen oder großen Zahlen gegen 0 geht oder sehr groß wird. All dies zeigt, dass die geeignete Wahl der Lernrate von entscheidender Bedeutung für die spätere Erkennungsleistung ist und das Training unter Umständen sehr sensibel auf Änderungen der Lernrate reagieren kann.
Eine weitere Vergrößerung der Anzahl an Trainingsiterationen ist angesichts der erzielten Erkennungsraten nicht erforderlich. Es ist zu erwarten, dass sich durch eine weitere Steigerung der Anzahl an Trainingsiterationen das Overfitting des Neuronalen Netzes auf den Trainingsdatensatz verstärken und sich die Übertragbarkeit der gelernten Merkmale auf den Testdatensatz verschlechtern würde.

Vergleich der Netzarchitekturen VGG16 und AlexNet

Die grundlegende Struktur der beiden Netzarchitekturen des VGG16 und AlexNet sind bis auf die Anzahl der Convolutional Layer sehr ähnlich. Im Folgenden soll der Unterschied der Erkennungsleistung beider Netze untersucht werden. Beide Netze wurden auf dem ILSVRC 2012 Datensatz vortrainiert und auf dem PascalVOC 2007 Trainingsdatensatz fein-getuned. Abbildung 6.4.2 enthält die Ergebnisse der Tests des AlexNet und VGGNet auf dem PascalVOC 2007 Testdatensatz.
Es ist zu erkennen, dass die erreichten Average Precision Werte des VGGNet in allen Klassen über dem AlexNet liegen, das relative Niveau der Werte ist jedoch ähnlich. Die Mean Average Precision von 65,80% des VGG16 ist den 47,13% des AlexNet deutlich überlegen. Die Klasse *bottle* zeigte sich in vielen Tests als besonders schwierig zu erkennen und profitiert am meisten von der größeren Anzahl an Layern des VGGNet, das mit einer Average Precision von 26,98% eine mehr als doppelt so hohe Erkennungsleistung für diese Klasse besitzt. Die geringste Differenz von 9,32% AP besteht in der Klasse *tvmonitor* und ist damit immernoch sehr deutlich.

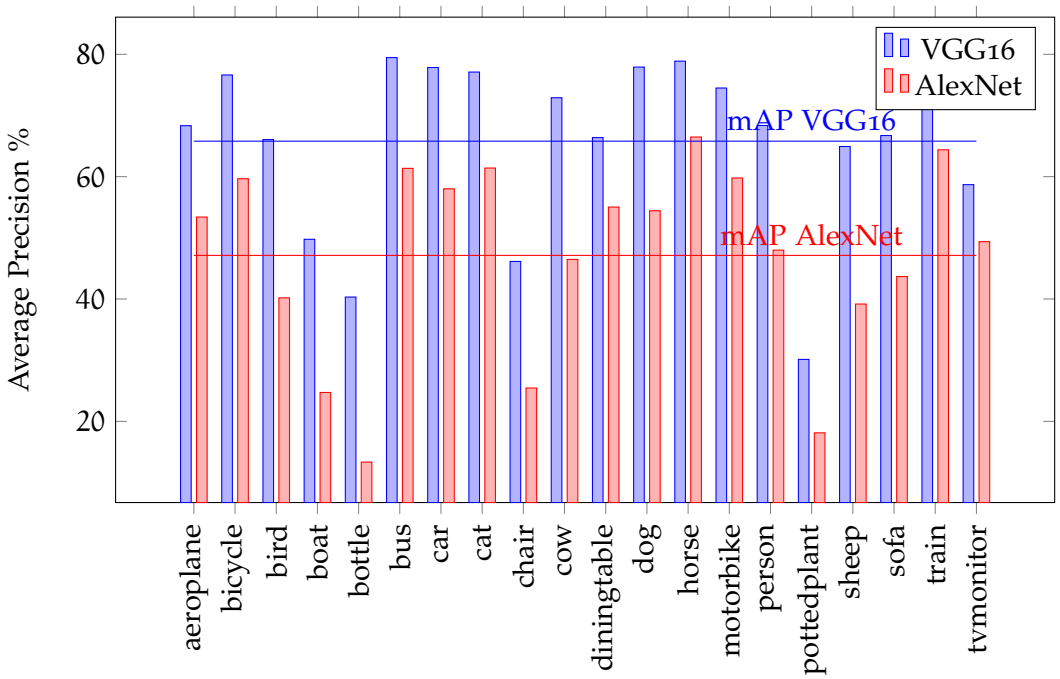


Abbildung 6.4.2: Vergleich der Ergebnisse zwischen dem VGGNet und AlexNet

Es ist zu erwähnen, dass Girshick et al. [GDDM15] mit dem AlexNet durch intensives Suchen nach der besten Parameterkonfiguration bis zu 58,5% mAP auf dem PascalVOC 2007 Datensatz erreicht haben. Zwar zeigt sich auch in den Tests der Autoren das VGGNet mit einer Mean Average Precision von 66,0% in allen Objektklassen als leistungsfähiger, jedoch mit geringerer Differenz als hier dargestellt. Das in diesem Vergleich erzielte Ergebnis von 65,80% mAP mit dem VGGNet entspricht hingegen fast genau dem Ergebnis von 66,0% mAP aus [GDDM15].

Region Proposal Networks und Selective Search für die Bildbereichsberechnung

Mit den Region Proposal Networks wurde ein alternatives Verfahren für die Bildbereichsberechnung entwickelt, dass im Gegensatz von Selective Search nicht auf heuristischen Annahmen zur Objektbeschaffenheit beruht, sondern im Training Merkmale lernt, um Bildbereiche, die Objekte beinhalten, zu bestimmen.
In Abbildung 6.4.3 findet sich eine Gegenüberstellung der Erkennungsleistung pro Klasse des PascalVOC 2007 Datensatzes für beide Optionen. Die grundlegende Parameterkonfiguration entspricht der aus Tabelle 6.4.1 mit einem Austausch der Bildbereichsberechnung mit Selective Search durch RPNs.

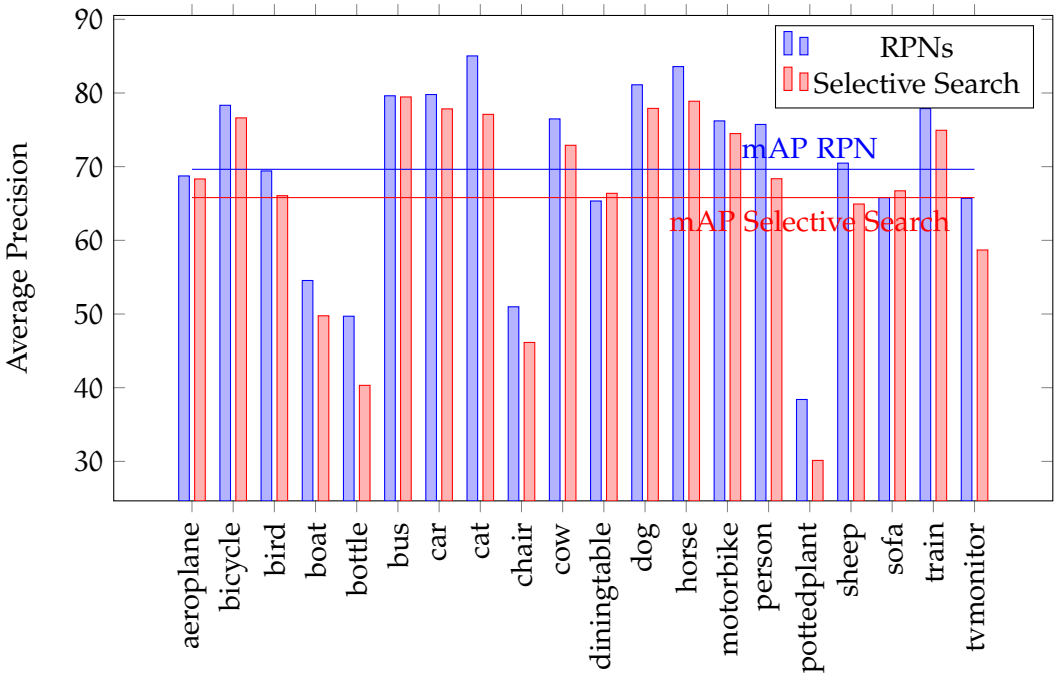


Abbildung 6.4.3: Vergleich der Ergebnisse zwischen zwei verschiedenen Methoden zur Bildbereichsberechnung: Region Proposal Networks und Selective Search

Die Bildbereichsberechnung durch Region Proposal Networks übertrifft das Verfahren mit Selective Search um durchschnittlich 3,84%. Grundsätzlich ist das Niveau der Leistungsfähigkeit von den beiden Verfahren pro Klasse ähnlich, das heißt Objektklassen die von dem Verfahren mit Selective Search gut erkannt wurden, sind generell auch vom Verfahren mit Region Proposal Networks gut erkannt worden. Aus den 20 Objektklassen des PascalVOC 2007 Datensatzes gibt es lediglich zwei, die bei der Verwendung von Selective Search eine höhere Average Precision zeigen als bei RPNs. Dies sind die Klassen *diningtable* und *sofa*, bei denen Selective Search eine um etwa 1% höhere Average Precision aufweist. Bei diesen zwei Klassen funktionieren die Annahmen über einheitliche Farbe, Textur, usw. etwas besser als die gelernten Merkmale der RPNs. Der größte Unterschied bei der Average Precision existiert zwischen den beiden Verfahren bei der Klasse *bottle*. Hier übertrifft das RPN-Verfahren die 40,32% AP von Selective Search um 9,38%, da Flaschen sehr klein sind und zudem schwer durch Farb- und Texturmerkmale zu erkennen sind. Hier sind Region Proposal Networks mit ihren gelernten Merkmalen im Vorteil.

Neben der größeren Leistungsfähigkeit ist auch die deutlich geringere Berechnungsdauer eine Stärke der RPNs. Durch das Teilen der Convolutional Features mit der Merkmalsextraktion kann die Dauer für die Bildbereichsberechnung mit wenigen Millisekunden nahezu vernachlässigt werden. Overhead beim Datenaustausch in der Schnittstelle zwischen Python und C++ erzeugt bereits ohne die eigentliche Berechnung der Selective Search Daten eine mit durchschnittlich 0,25 Sekunden pro Bild um etwa 0,1 Sekunden größere Laufzeit des Verfahrens. Zu dieser Zeit hinzu kommt noch die Berechnung der Bildbereichsvorschläge mit Selective Search, die je nach Implementation und Rechenleistung im Bereich mehrerer Sekunden pro Bild liegt. Damit ist das RPN Verfahren mit 69,64% mAP nicht nur leistungsfähiger sondern auch erheblich schneller als die ursprüngliche R-CNN Methodik mit Selective Search.

Support Vector Machines und SoftMax Layer für die Klassifikation

Wie bei der Bildbereichsberechnung werden auch bei der Klassifikation zwei verschiedene Alternativen betrachtet: Die Klassifikation durch SoftMax Layer und durch Support Vector Machines. In Abbildung 6.4.4 sind die erzielten Ergebnisse auf dem PascalVOC 2007 Datensatz zu sehen. Auch wurde wieder die Konfiguration aus Tabelle 6.4.1 verwendet, einzig der Klassifikator wurde ausgetauscht.
Die Klassifikation mithilfe des SoftMax Layers übertrifft die Support Vector Machines um 1,52%. Auch hier ist das Niveau der Average Precision pro Klasse in etwa identisch. Sowohl die beste (*bus*) als auch die schlechteste Objektklasse (*pottedplant*) sind in beiden Fällen gleich. Die größte Leistungsdifferenz zwischen den beiden Verfahren besteht in den Klassen *cow*, bei der der SoftMax Klassifikator eine um 3,94% höhere Average Precision zeigt als die SVMs und *tvmonitor*, bei der umgekehrt die Support Vector Machines den SoftMax Layer sogar um 5,63% übertrifft. Im Vergleich zur Leistungsdifferenz zwischen RPNs und Selective Search liegen die Ergebnisse von SVMs und dem SoftMax Layer deutlich näher beieinander.

6.4.2 *Ergebnisse auf dem SUN 2012 Datensatz*

Im Folgenden soll das R-CNN Verfahren mit der Konfiguration aus Tabelle 6.4.1 anhand des SUN 2012 Datensatzes evaluiert werden. Aufgrund der Struktur der Objektklassen des SUN Datensatzes wurde zunächst ein Stemming der Annotationen durchgeführt, bei dem Plural- auf Singularformen der Wörter abgebildet und Stichwörter wie *occluded* und *crop* aus den Klassennamen entfernt wurden. Annotationen wie *person sitting* und *person walking* wurden ebenfalls zur Klasse *person* zusammengeführt.

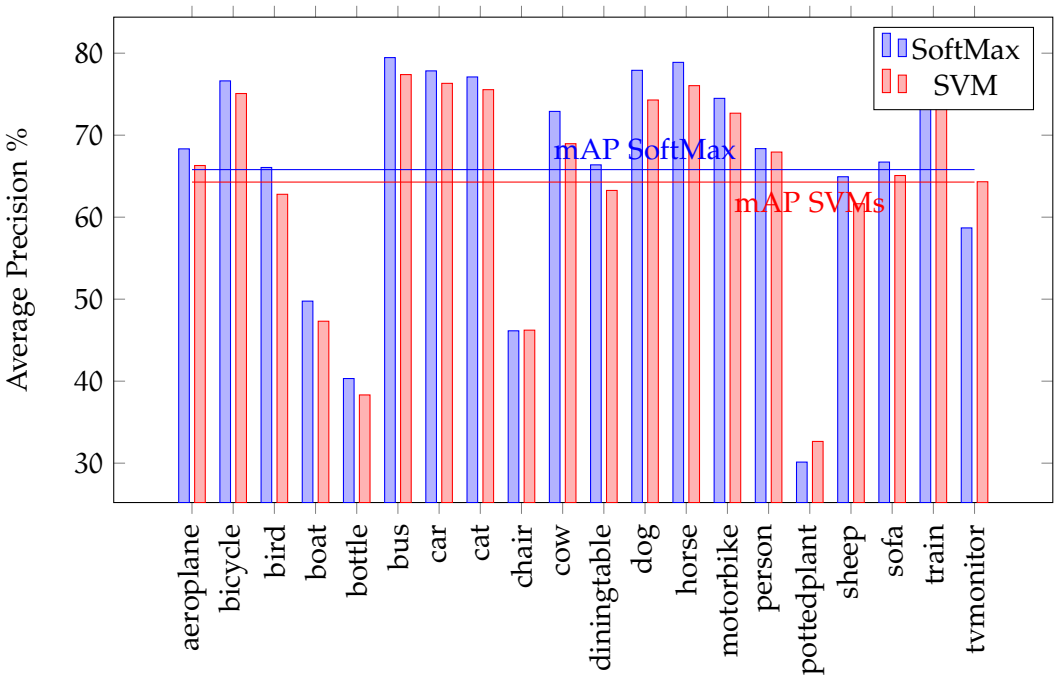


Abbildung 6.4.4: Vergleich der Ergebnisse zwischen zwei verschiedenen Methoden zur Klassifikation: SoftMax Layer und Support Vector Machines

Diese Vorgehensweise vergrößert die Anzahl an Objektbeispielen für viele Klassen und vereinheitlicht die durch das Crowd-Sourcing erstellten Annotationen. Es ist oft nicht erkennbar, in welchen Fällen der Plural von Wörtern verwendet oder die von einer Person ausgeführte Aktion mit bei der Objektklasse angegeben wird. Dies erschwert eine zuverlässige Erkennung der korrekten Objektklasse ohne Stemming.
Um das Training des Neuronalen Netzes auf dem neuen Datensatz zunächst so einfach wie möglich zu gestalten, wurde für die Bildbereichsberechnung, trotz der auf dem PascalVOC gezeigten besseren Leistung, auf Region Proposal Networks verzichtet und Selective Search verwendet. Zur Merkmalsberechnung wurde das VGG16 eingesetzt. Das SoftMax Layer diente als Klassifikator, da es ohnehin im Training benötigt wird und deshalb nicht wie die Support Vector Machines gesondert trainiert werden muss. Zudem zeigte das Verfahren mit der Klassifikation über SoftMax Layer beim PascalVOC Datensatz eine bessere Mean Average Precision.

Test in den 100 am häufigsten vorkommenden Klassen des SUN Datensatzes

Getestet wurde zunächst die Erkennungsleistung in 100 Objektklassen, deren Objekte am häufigsten im Datensatz vorkommen. Die Ergebnisse pro Klasse aus Tabelle 6.4.2 sind dabei nach Häufigkeit des Objektklassen-Vorkommens spaltenweise absteigend sortiert. Die 20 Ergebnisse mit der höchsten Average Precision (aus den 100 häufigsten Klassen) wurden fett gedruckt. Die Mean Average Precision aller Klassen, der 20 besten Klassen (welche in der Tabelle fett gedruckt sind) und der 20 häufigsten Klassen sind in den letzten Zeilen der Tabelle angegeben.

Klasse	AP	Klasse	AP	Klasse	AP	Klasse	AP
wall	**14,89%**	column	0,32%	step	0,01%	jar	0,00%
window	3,66%	ground	2,17%	outlet	0,01%	**field**	**13,84%**
person	**4,30%**	desk lamp	3,18%	streetlight	1,44%	pane	0,03%
tree	**7,42%**	shelf	1,82%	pole	0,94%	palm tree	0,78%
chair	1,03%	**bed**	**22,19%**	**sea water**	**24,56%**	staircase	0,84%
building	**13,73%**	seat	0,78%	plate	0,01%	shoe	0,37%
floor	**16,29%**	sign	0,17%	railing	0,24%	rug	0,67%
ceiling lamp	0,13%	sidewalk	2,80%	worktop	0,22%	clock	0,2%
sky	**52,27%**	flower	0,6%	ball	0,01%	path	1,21%
plant	1,05%	pillow	0,21%	basket	0,07%	tray	0,43%
ceiling	**21,14%**	fence	0,65%	bush	0,77%	magazine	0,00%
cabinet	**4,42%**	armchair	1,86%	fluorescent tube	0,04%	poster	0,24%
door	0,52%	plant pot	0,07%	water	2,54%	stove	0,74%
car	**6,05%**	sconce	0,33%	sofa	3,70%	boat	0,11%
book	0,17%	vase	0,01%	bench	1,39%	drawer	0,61%
table	2,76%	desk lamp	2,14%	paper	0,00%	handrail	0,05%
painting	**7,72%**	rock	0,59%	stool	0,05%	**snowy mountain**	**38,82%**
box	0,11%	screen	2,60%	balcony	0,61%	chandelier	3,86%
curtain	**5,90%**	tree trunk	1,58%	glass	0,01%	swivel chair	2,65%
grass	**4,65%**	night table	1,50%	stand	0,35%	flag	0,02%
picture	0,36%	stone	0,1%	pot	0,08%	cup	0,00%
road	**36,24%**	**skyscraper**	**22,51%**	towel	0,09%	**bookcase**	**7,72%**
cushion	0,66%	house	2,6%	bag	0,06%	bowl	0,03%
bottle	0,00%	mirror	1,15%	sink	0,16%	washbasin	0,27%
mountain	**5,97%**	faucet	0,17%	text	0,27%	coffe table	0,67%
mAP (Gesamt)	3,95%						
mAP (20 beste)	**16,53%**						
mAP (20 häufigste)	8,41%						

Tabelle 6.4.2: Erkennungleistung für die 100 häufigsten Klassen des SUN 2012 Datensatzes

Besonders auffällig ist die im Vergleich zu der auf dem PascalVOC 2007 Datensatz eher geringe erreichte Mean Average Precision von 3,95% für alle 100 Klassen. Selbst die Erkennungsleistung der 20 besten unter den 100 häufigsten Klassen von 16,53% mAP ist weit entfernt von den 65,80% mAP, die auf dem PascalVOC Datensatz erreicht wurden. *Person* befindet sich mit einer AP von nur 4,30% unter den 20 besten Klassen. Aus den 100 häufigsten Klassen besitzen 5 eine AP von 0,00% (*bottle*, *paper*, *jar*, *magazine* und *cup*) und insgesamt 59 Klassen eine AP von unter 1%. Dies ist bedingt durch die besondere Komplexität des SUN Datensatzes.
Es besteht ein klar erkennbarer Zusammenhang zwischen Häufigkeit einer Objektklasse und der erzielten Erkennungsleistung. Unter den 20 häufigsten Klassen befinden sich auch 12 der 20 Klassen mit dem besten Ergebnis. Dies zeigt, dass gerade für das Training von Neuronalen Netzen, die Anzahl an Objektvorkommen eine große Rolle für das Lernen der Objektmerkmale und damit der späteren Erkennungsleistung spielt.

Test in den 20 am häufigsten vorkommenden Klassen des SUN Datensatzes

Analog zum vorhergehenden Test der 100 häufigsten Objektklassen im SUN 2012 Datensatz soll nun die Erkennungsleistung des Verfahrens in den 20 häufigsten Objektklassen überprüft werden. Dafür wurde in Abbildung 6.4.5 die Average Precision pro Klasse für ein auf dem SUN 2012 (20 häufigsten Klassen) trainierten VGG16 Netz in blau dargestellt. Zum Vergleich sind zusätzlich in rot die Ergebnisse für die 20 häufigsten des für 100 häufigste Klassen trainierten Netzes aus Tabelle 6.4.2 angegeben. Die Beschränkung auf 20 Objektklassen beim Training hat sich bei der Erkennungsleistung nicht positiv bemerkbar gemacht. Das für 100 Objektklassen trainierte Netz zeigt insgesamt sogar eine unwesentlich bessere Erkennungsleistung von 8,41% mAP im Vergleich zu den 8,20% mAP des für 20 Objektklassen trainierten Netzes. Dies könnte auf die höhere Anzahl von Objekt-Beispielen im Training von 100 im Gegensatz zu 20 Klassen zurückzuführen sein. Insgesamt ist die Leistung in den verschiedenen Objektklassen der beiden Netze sehr ähnlich.
Auffällig ist, wie beim Test der 100 häufigsten Klassen, die oft eher schlechte Erkennungsleistung. Lediglich die Klasse *sky* wurde mit 59,20% mAP relativ zuverlässig erkannt. Wäre diese Klasse nicht enthalten, beträgt die Mean Average Precision für die restlichen 19 Klassen nur noch 5,52%.

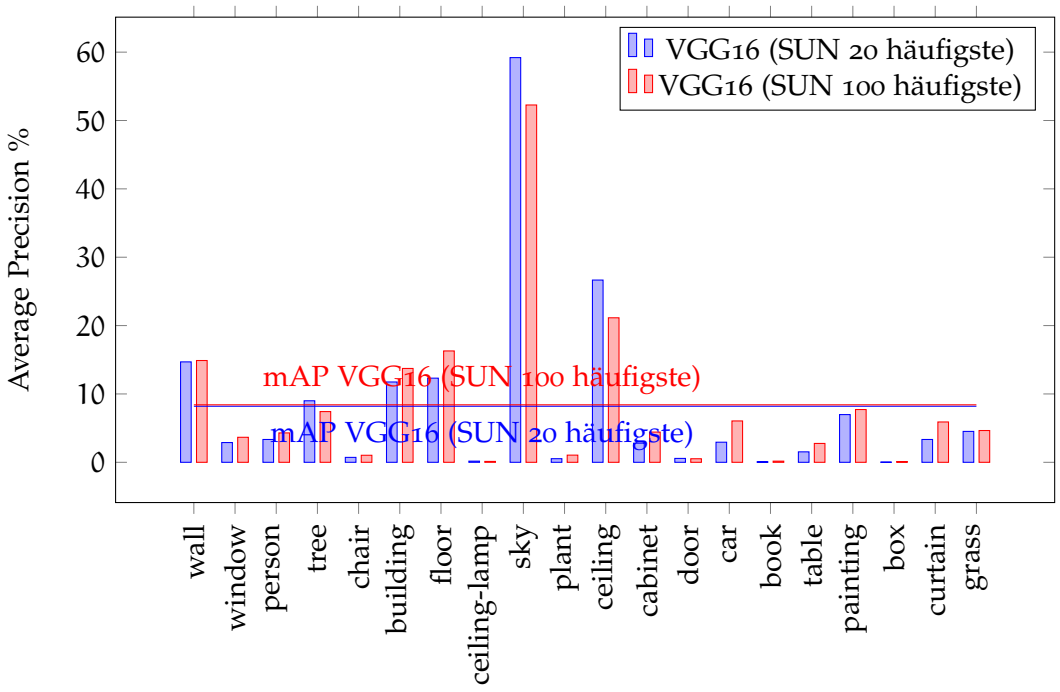


Abbildung 6.4.5: Test der Erkennungsleistung auf dem SUN 2012 Datensatz in den 20 häufigsten Klassen

Test in den 20 Klassen des PascalVOC auf dem SUN Datensatz

In diesem Test wurde die Erkennungsleistung im SUN 2012 Datensatz in den 20 Objektklassen des PascalVOC untersucht. Durch die große Klassenvielfalt des SUN 2012 Datensatzes konnten die Objektklassen des PascalVOC Datensatzes fast vollständig übernommen werden. Lediglich die Klassen *aeroplane*, *diningtable*, *pottedplant* und *tvmonitor* wurden durch die Klassen *airplane*, *table*, *plant* und *television* mit nahezu äquivalenter Bedeutung ausgetauscht. Die PascalVOC Klassen zeichnen sich durch ihre klare Unterscheidbarkeit und Eindeutigkeit aus. Klassenkombinationen wie *picture* und *painting* oder *tree* und *tree trunk* wie sie in den 100 häufigsten Klassen vorkommen, wurden bei der Auswahl der PascalVOC Klassen ausgenommen. Das beste durch Training auf dem SUN 2012 Datensatz erreichte Ergebnis ist in Abbildung 6.4.6 unter dem Namen VGG16 (SUN trainiert) in blau eingezeichet. Hier wurde das auf dem ILSVRC 2012 vortrainierte VGG16 Modell verwendet und ein weiteres Pretraining auf dem PascalVOC 2007 Datensatz (mit den 20 PascalVOC Objektklassen) durchgeführt.

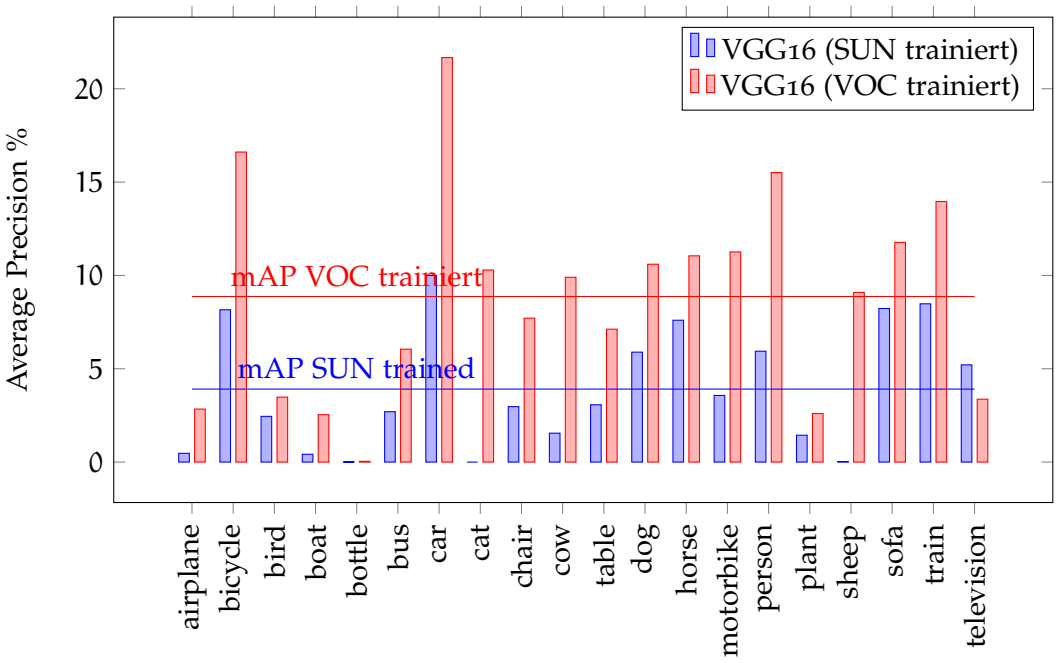


Abbildung 6.4.6: Test der Erkennungsleistung verschiedener Modelle auf dem SUN 2012 Datensatz in den 20 Klassen des PascalVOC

Im Anschluss daran wurde auf dem SUN 2012 Train Datensatz trainiert und auf dem Test Datensatz ausgewertet. Demgegenüber steht der Eintrag VGG16 (VOC trainiert) in rot, bei dem das von Girshick et al. in [GDDM15] verwendete VGG16 Netz (auf dem ILSVRC 2012 vortrainiert und auf dem PascalVOC 2007 Datensatz fine-getuned) ohne weiteres Training direkt auf dem SUN 2012 Test Datensatz ausgewertet wurde. Es ist überraschend, dass das VOC trainierte Model eine mehr als doppelt so hohe Mean Average Precision von 8,87% zeigt wie das auf der SUN trainierte Netz mit 3,91%. Die auf dem PascalVOC 2007 gelernten Merkmale der Objektklassen sind besser auf den SUN 2012 Datensatz generalisierbar, als die auf dem SUN 2012 Train-Datensatz gelernten Merkmale. Lediglich für die Klasse *television* besitzt das R-CNN Model eine schlechtere Average Precision als das auf dem SUN Datensatz trainierte Netz.
Grundsätzlich zeigen beide getesteten Modelle ähnliche Tendenzen hinsichtlich der Erkennungsleistung verschiedener Objektklassen. Während Klassen wie *car* oder *bicycle* verhältnismäßig gut erkannt wurden, funktionieren Klassen wie *bottle* oder *airplane* in beiden Fällen eher schlecht.

Diskussion der Komplexität des SUN Datensatzes

Die im Vergleich zum PascalVOC 2007 eher schwachen erzielten Ergebnisse auf dem SUN 2012 Datensatz sind durch die Besonderheiten der Bilder und Annotationen zu erklären. Bei den 1000 Objektklassen für die ILSVRC 2012 wurde beispielsweise darauf geachtet, dass keine 2 Objektklassen Nachfahren voneinander in der WordNet Baumstruktur sind. Dadurch wird eine semantische Überschneidung der Wortbedeutungen vermieden, was bei den Annotationen zu Verwirrungen und Mehrdeutigkeiten führen könnte. Bei dem SUN 2012 Datensatz (der hauptsächlich für Tests der Szenen-Klassifikation gedacht ist) gelten diese Bedingungen nicht. Beispielsweise sind *chair*, *armchair*, *swivel chair*, *dental chair*, *dental swivel chair* und einige weitere Stuhlarten alle als einzelne Objektklassen im Datensatz enthalten. Diese oft unklare Trennung führt zu Schwierigkeiten beim Lernen von Merkmalen und dem Auseinanderhalten der verschiedenen Objektklassen. Durch die Erstellung der Annotation mittels Crowd-Sourcing und sind diese nicht immer einheitlich. Ein noch umfangreicheres Stemming hätte dieses Problem gegebenenfalls mindern können.
Weiterhin stellt die Größe des Datensatzes eine Herausforderung für das Objekterkennungsverfahren dar. Wird beispielsweise ein Neuronales Netz auf die Erkennung der 20 Objektklassen von der Pascal VOC Challenge 2007 auf der SUN 2012 trainiert (um das oben genannte Problem mit der Semantik der Objektklassen zu umgehen), enthält der insgesamt 5.062 Bilder umfassende Trainings-Datensatz nur 4 Vorkommen der Klasse *cat*. Wie in der Tabelle 6.4.3 zu erkennen ist, haben mehr als die Hälfte der PascalVOC Klassen weniger als 100 Objektvorkommen im SUN 2012 Trainingsdatensatz.

airplane 51	bicycle 44	bird 11	boat 159	bottle 560	bus 55	car 1461	cat 4	chair 2751	cow 31
table 1001	dog 43	horse 24	motorbike 25	person 3890	plant 1975	sheep 9	sofa 239	train 23	television 127

Tabelle 6.4.3: Aufstellung der Objekthäufigkeit im SUN 2012 Trainingsdatensatz

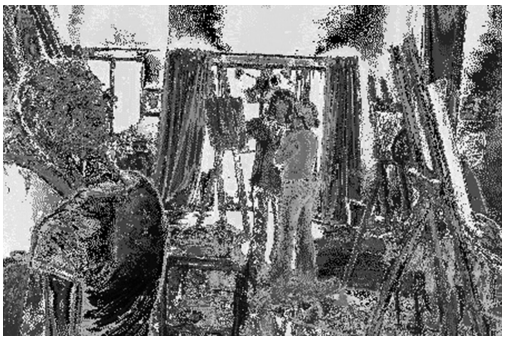
Eine große Rolle spielt im Zuge der Objekterkennung auch die Struktur der Bilder. In Abbildung 6.4.7 sind Ausschnitte aus Bildern des SUN 2012 Datensatzes zu sehen, deren Merkmale Schwierigkeiten bei der Objekterkennung hervorrufen können. Neben Objekten, die trotz starker Verdeckung noch erkannt werden sollen, sind Bilder mit eingefügten Inhalten, Wasserzeichen, schlechter Belichtungssituation, Effekt-versehene Bilder, Bilder mit sehr geringer Auflösung und Bilder Computer-generierter Szenen enthalten. All diese untypischen Bildmerkmale erschweren sowohl das Training als auch die spätere Objekterkennung.

1. Starke Verdeckung (Person)

2. Eingefügte Bildinhalte
(Nicht Teil der Annotation)

3. Wasserzeichen &
Starke Verdeckung (Stühle)

4. Schwierige Belichtungssituation &
Person in Hitzeschutzanzug



5. Effektfilter
(Es sind 3 Personen abgebildet)

6. Schlechte Beleuchtung

Abbildung 6.4.7: Ausschnitte einiger Beispielbilder aus dem SUN 2012 Datensatz, die Probleme bei der Objekterkennung hervorrufen können

6.4.3 *Vergleich der Ergebnisse mit Part-based Models*

Um eine Einordnung der Leistungsfähigkeit des R-CNN Ansatzes durchführen zu können, folgt nun ein Vergleich der erzielten Ergebnisse mit den Part-based Models. Choi et al. [CTW12] verwenden in ihrem Tree-based Context Model Verfahren die Part-based Models aus [FGMR10] und verknüpfen diese mit Kontextinformationen über den Zusammenhang des Vorkommens verschiedener Objektklassen (zum Beispiel dass ein Auto im Bild oft auch auf eine Straße im Bild hindeutet). Auf dem PascalVOC 2007 Datensatz verbessern sie damit die Mean Average Precision des reinen Part-based Model Verfahrens von 26.70% auf 27,90%. Dies sind allerdings noch 35,9% weniger als die Mean Average Precision des R-CNN Verfahrens mit dem VGGNet. In Abbildung 6.4.8 ist ein Vergleich der Average Precision pro Klasse der beiden Verfahren zu sehen. Die Überlegenheit der Erkennungsleistung des R-CNN Verfahrens im PascalVOC 2007 Datensatz ist über alle Klassen hinweg deutlich erkennbar.

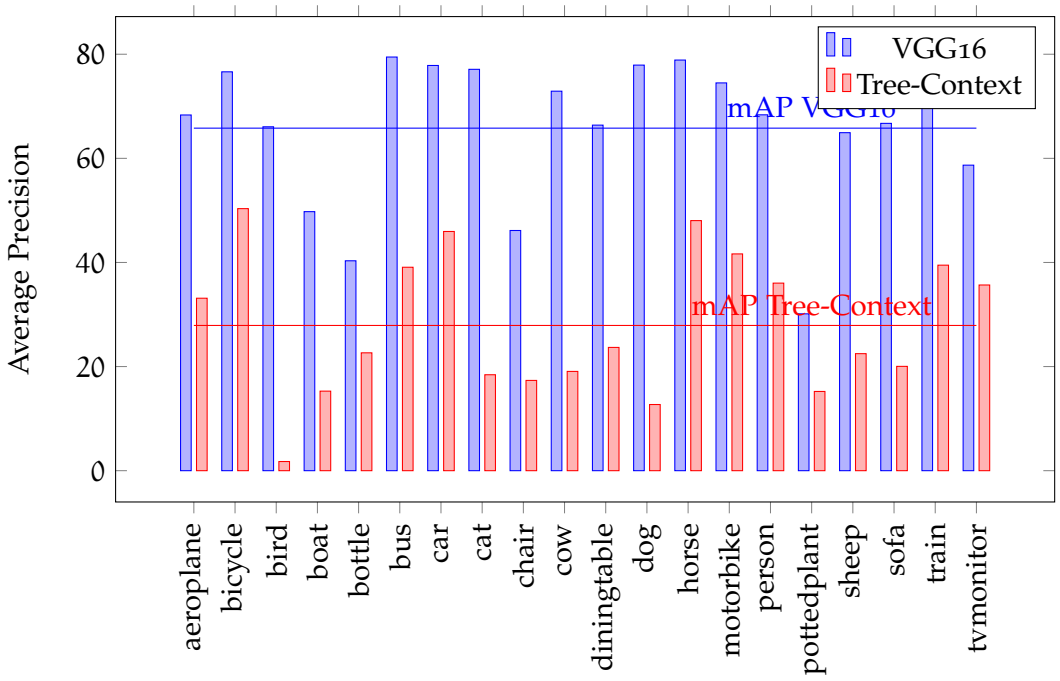


Abbildung 6.4.8: Vergleich der Ergebnisse auf dem PascalVOC 2007 Datensatz zwischen zwei verschiedenen Methoden zur Objekterkennung: Region-based Convolutional Neural Networks und Part-based Models (PBM Ergebnisse aus [CTW12])

Da gerade die Erkennungsleistung in Bildern natürlicher Szenen von der Methodik von Choi et al. profitieren kann, wurde eine Auswertung auf dem SUN 2009 Datensatz durchgeführt auf dem 8,55% mAP in 107 Klassen erreicht wurden. Zwar kann ein Grund für das im Vergleich zu der R-CNN Methodik gute Ergebnis die geringere Modellkomplexität sein, die das Training vereinfacht, jedoch kann dieser Wert nicht direkt mit den 3,95% (100 häufigste Klassen) oder 8,41% mAP (20 häufigste Klassen) des R-CNN Verfahrens auf dem SUN 2012 Datensatz verglichen werden. Es wurde zunächst kein identischer Datensatz (SUN 2009 statt 2012) verwendet und auch die Auswahl der Objektklassen fällt unterschiedlich aus. Choi et al. verwendeten nur Klassen, für die der zugrundeliegende Part-based Model Detektor überhaupt mehr als 4 Objektvorkommen korrekt erkannt hat. Der von Choi et al. durchgeführte Test ist im Vergleich zur Objekterkennung der 100 häufigsten Klassen auf dem SUN 2012 Datensatz als weniger komplex anzusehen. Insgesamt ist allerdings beim Einsatz des R-CNN Verfahrens auf dem SUN 2012 Datensatz im Vergleich zu den Part-based Models auf ähnlichen Daten keine Verbesserung in der Größenordnung, wie sie auf dem PascalVOC Datensatz eingetreten ist, zu erkennen.

7 FAZIT

Für den Task der Objekterkennung in Bildern natürlicher Szenen, für den es sowohl einer präzisen Segmentierung der Bildinhalte als auch einer zuverlässigen Klassifikation der einzelnen Objekte bedarf, stellen die Region-based Convolutional Neural Networks von Girshick et al. einen flexiblen und zugleich leistungsfähigen Lösungsansatz dar. Besonders die erzielten Ergebnisse auf dem Pascal Visual Object Classes 2007 Datensatz bestätigen dies. Interessant ist hier der große Einfluss der Tiefe des Neuronalen Netzes auf die Erkennungsleistung. Obwohl der grundlegende Aufbau des VGGNet und AlexNet bis auf die Anzahl der Convolutional Layer sehr ähnlich ist, zeigt das VGGNet eine deutlich höhere Mean Average Precision aufgrund der komplexeren Merkmale, die durch das tiefere Netz gelernt werden können. Die Leistungsdifferenz beim Einsatz von Region-Proposal Networks anstelle von Selective Search verdeutlicht den Vorteil gelernter Objektmerkmale für die Bildbereichsvorschläge der RPNs gegenüber den heuristischen Annahmen über Objektstrukturen von Selective Search bei gleichzeitig drastischer Verkürzung der Objekterkennungsdauer pro Bild. Der Einsatz eines SoftMax Klassifikators verbesserte das Ergebnis im Vergleich zu Support Vector Machines etwas. Insgesamt besitzt die R-CNN Methodik unter der Verwendung von Region Proposal Networks mit 69,64% mAP auf dem PascalVOC 2007 Datensatz eine gute Leistungsfähigkeit, die zum Zeitpunkt der Veröffentlichung des Faster R-CNN genannten Verfahrens im Frühjahr 2016 dem aktuellen Stand der Forschung entspricht. Ein von der grundsätzlichen Leistungsfähigkeit anderes Bild zeigt sich bei der Evaluierung der R-CNN Methodik auf dem Scene Understanding Datensatz von 2012, bei der die Zuverlässigkeit der Objekterkennung erheblich geringer ausfällt. Mit einer maximal erreichten Mean Average Precision von 8,87% liegt das Ergebnis für die Erkennung der Objektklassen des PascalVOC Datensatzes eine Größenordnung unter dem des PascalVOC 2007. Bemerkenswert ist, dass die beste Erkennungsleistung dabei durch ein Netz erreicht wird, das lediglich auf dem PascalVOC und nicht auf dem SUN Datensatz trainiert worden ist, wohingegen die Verwendung des SUN Trainingsdatensatzes die Erkennungsleistung auf 3,91% verschlechtert. Die Mean Average Precision für die Detektion der 20 oder 100 häufigsten Objektklassen fällt mit 8,41% beziehungsweise 3,95% trotz einer höheren Anzahl an Objektvorkommen für das Training auf dem SUN Datensatz ebenfalls gering aus. Als problematisch stellt sich dabei die Struktur der im SUN Datensatz enthaltenen Bilder und Annotationen heraus. Aufgrund der hohen

Variabilität der Bildinhalte, zum PascalVOC vergleichsweise geringen Qualität einiger Bilder und enge semantische Eingrenzung vieler Objektklassen stellen sich sowohl die Erzeugung qualitativ guter Bildbereichsvorschläge als auch die Berechnung geeigneter Objektmerkmale und die korrekte Klassifikation als große Herausforderung dar.
Besonders ist in diesem Zusammenhang auch der Vergleich zwischen dem R-CNN Verfahren und den Part-based Models. Zwar liegen die Ergebnisse beider Verfahren bei der Evaluation auf dem SUN 2012 Datensatz auf etwa dem selben Niveau, jedoch kann aufgrund der verschiedenen Datensätze und zu erkennenden Objektklassen kein direkter Vergleich zwischen den Werten durchgeführt werden. Auf dem PascalVOC 2007 Datensatz erreichen die Part-based Models 27,90% mAP, wohingegen die Erkennungsleistung der Region-based Convolutional Neural Networks mit 65,80% mAP mehr als doppelt so hoch ist. Hier zeigt sich analog zu der Verbesserung der Region Proposal Networks gegenüber Selective Search die Überlegenheit gelernter Merkmale in Neuronalen Netzen zu heuristischen Ansätzen, aber gleichzeitig auch die Wichtigkeit der für das Training dieser Merkmale erforderlichen großen Datensätze.
Die durch den Einsatz von Neuronalen Netzen erzielten erheblichen Verbesserungen in der Leistungsfähigkeit und Geschwindigkeit gegenüber vorherigen Objekterkennungsverfahren wie den Part-based Models erklären die große mediale Aufmerksamkeit innerhalb und außerhalb der Fachpresse und ermöglichen Applikationen, bei denen Genauigkeit und Geschwindigkeit bedeutende Faktoren sind.

LITERATURVERZEICHNIS

[Bis06] Bishop, Christopher M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006. – ISBN 0387310738

[Bro14] Brown, Larry: *Accelerate Machine Learning with the cuDNN Deep Neural Network Library*. https://devblogs.nvidia.com/parallelforall/accelerate-machine-learning-cudnn-deep-neural-network-library/, 2014. – Accessed: 30.07.2016

[Caf16] *MultinomialLogisticLossLayer Class Template Reference*. http://caffe.berkeleyvision.org/doxygen/classcaffe_1_1MultinomialLogisticLossLayer.html, 2016. – Accessed: 04.08.2016

[CTW12] Choi, Myung J. ; Torralba, Antonio ; Willsky, Alan S.: A tree-based context model for object recognition. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2012)

[DT05] Dalal, Navneet ; Triggs, Bill: Histograms of Oriented Gradients for Human Detection. In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, IEEE Computer Society, 2005. – ISBN 0–7695–2372–2, S. 886–893

[EEG+15] Everingham, Mark ; Eslami, S. M. A. ; Gool, Luc V. ; Williams, Christopher K. I. ; Winn, John ; Zisserman, Andrew: The Pascal Visual Object Classes Challenge: A Retrospective. In: *International Journal of Computer Vision* (2015)

[EGW+12] Everingham, Mark ; Gool, Luc V. ; Williams, Christopher K. I. ; Winn, John ; Zisserman, Andrew: *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012

[FGMR10] Felzenszwalb, Pedro F. ; Girshick, Ross B. ; McAllester, David ; Ramanan, Deva: Object Detection with Discriminatively Trained Part-Based Models. In: *IEEE Transactions on Pattern Analysis Machine Intelligence* 32 (2010). http://dx.doi.org/10.1109/TPAMI.2009.167. – ISSN 0162–8828

[FH04] Felzenszwalb, Pedro F. ; Huttenlocher, Daniel P.: Efficient Graph-Based Image Segmentation. In: *Int. J. Comput. Vision* (2004). http://dx.doi.org/10.1023/B:VISI.0000022288.19776.77

[Fin16] Fink, Gernot A.: *Skriptum zur Vorlesung "Mustererkennung"*. Vorlesungsskript, 2016

[GDDM15] Girshick, Ross B. ; Donahue, Jeff ; Darrell, Trevor ; Malik, Jitendra: Region-based Convolutional Networks for Accurate Object Detection and Segmentation. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2015)

[JSD+14] Jia, Yangqing ; Shelhamer, Evan ; Donahue, Jeff ; Karayev, Sergey ; Long, Jonathan ; Girshick, Ross B. ; Guadarrama, Sergio ; Darrell, Trevor: Caffe: Convolutional Architecture for Fast Feature Embedding. In: *arXiv preprint arXiv:1408.5093* (2014)

[KSH12] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks. Version: 2012. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. In: *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, 1097–1105

[LBBH98] Lecun, Yann ; Bottou, Léon ; Bengio, Yoshua ; Haffner, Patrick: Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE*, 1998, S. 2278–2324

[LGRN09] Lee, Honglak ; Grosse, Roger ; Ranganath, Rajesh ; Ng, Andrew Y.: Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In: *Proceedings of the 26th Annual International Conference on Machine Learning*, ACM, 2009. – ISBN 978–1–60558–516–1, 609–616

[Mil95] Miller, George A.: WordNet: A Lexical Database for English. In: *Commun. ACM* 38 (1995). http://doi.acm.org/10.1145/219717.219748. – ISSN 0001–0782

[MRS08] Manning, Christopher D. ; Raghavan, Prabhakar ; Schütze, Hinrich: *Introduction to Information Retrieval*. Cambridge University Press, 2008. – ISBN 0521865719, 9780521865715

[Nie03] Niemann, Heinrich: *Klassifikation von Mustern*. Bd. 2. Universität Erlangen, 2003 http://www5.informatik.uni-erlangen.de/fileadmin/Persons/NiemannHeinrich/klassifikation-von-mustern/m00-www.pdf

[RDS+15] Russakovsky, Olga ; Deng, Jia ; Su, Hao ; Krause, Jonathan ; Satheesh, Sanjeev ; Ma, Sean ; Huang, Zhiheng ; Karpathy, Andrej ; Khosla, Aditya ; Bernstein, Michael ; Berg, Alexander C. ; Fei-Fei, Li: ImageNet Large Scale Visual Recognition Challenge. In: *International Journal of Computer Vision (IJCV)* (2015)

[RHGS15a] Ren, Shaoqing ; He, Kaiming ; Girshick, Ross ; Sun, Jian: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: *Advances in Neural Information Processing Systems (NIPS)*, Curran Associates, Inc., 2015, S. 91–99

[RHGS15b] Ren, Shaoqing ; He, Kaiming ; Girshick, Ross B. ; Sun, Jian: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. (2015). http://arxiv.org/abs/1506.01497

[RHW88] Rumelhart, David E. ; Hinton, Geoffrey E. ; Williams, Ronald J.: Learning Representations by Back-propagating Errors. In: *Neurocomputing: Foundations of Research*. MIT Press, 1988. – ISBN 0–262–01097–6, S. 696–699

[RTMF08] Russell, Bryan C. ; Torralba, Antonio ; Murphy, Kevin P. ; Freeman, William T.: LabelMe: A Database and Web-Based Tool for Image Annotation. In: *International Journal of Computer Vision* 77 (2008). http://dx.doi.org/10.1007/s11263-007-0090-8. – ISSN 0920–5691

[SHK+14] Srivastava, Nitish ; Hinton, Geoffrey ; Krizhevsky, Alex ; Sutskever, Ilya ; Salakhutdinov, Ruslan: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In: *Journal of Machine Learning Research* 15 (2014). http://dl.acm.org/citation.cfm?id=2627435.2670313. – ISSN 1532–4435

[SMB10] Scherer, Dominik ; Müller, Andreas ; Behnke, Sven: Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In: *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III*, Springer Berlin Heidelberg, 2010. – ISBN 3–642–15824–2, 978–3–642–15824–7, S. 92–101

[SZ14] Simonyan, Karen ; Zisserman, Andrew: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *CoRR* (2014). http://arxiv.org/abs/1409.1556

[USGS13] Uijling, Jasper R. R. ; Sande, Koen E. A. d. ; Gevers, Theo ; Smeulders, Arnold W. M.: Selective Search for Object Recognition. In: *International Journal of Computer Vision* (2013). https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013

[VLF13] *VLFeat: Basic HOG computation*. http://www.vlfeat.org/overview/hog.html, 2013. – Accessed: 08.09.2016

[XHE+10] Xiao, Jianxiong ; Hays, James ; Ehinger, Krista A. ; Oliva, Aude ; Torralba, Antonio: SUN database: Large-scale scene recognition from abbey to zoo. In: *CVPR*, IEEE Computer Society, 2010. – ISBN 978–1–4244–6984–0, 3485-3492

ABBILDUNGSVERZEICHNIS