Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institute for Computer Science
Department VI
Autonomous Intelligent Systems

Bildsegmentation in Objekt-Klassen mit Konvolutionalen Neuronalen Netzen

Bachelorarbeit

Nico Höft

Erstprüfer: Prof. Dr. Sven Behnke

Zweitprüfer: Prof. Dr. Jürgen Gall

Betreuer: Hannes Schulz

Eingereicht: Montag, den 12.5.2014

Erklärung

Hiermit versichere ich, die eingereichte Bachelorarbeit selbständig verfasst und keine anderen als die von mir angegebenen Quellen und Hilfsmittel benutzt zu haben. Wörtlich oder inhaltlich verwendete Quellen wurden entsprechend den anerkannten Regeln wissenschaftlichen Arbeitens (lege artis) zitiert.

Ich erkläre weiterhin, dass die vorliegende Arbeit noch nicht anderweitig als Bachelorarbeit eingereicht wurde.

Ort, Datum

Unterschrift

Zusammenfassung

Bildverstehen ist ein Gebiet, in dem durch die jüngste Entwicklung in Rechenleistung, insbesondere mittels Grafikkarten, neue Verfahren angewandt und größere Datenmengen verarbeitet werden koennen. Ein Teilproblem des Bildverstehens ist die pixelweise Segmentierung von Bildern in Objektklassen.

In dieser Arbeit wird ein Verfahren vorgestellt, welches konvolutionale Neuronale Netze verwendet, um Pixelklassen über lokale Bildinformationen zu bestimmen. Dabei werden Bilder mit ZCA und HOG vorverarbeitet. Multiskale Eingaben und Pooling machen das Verfahren robuster gegen Deformationen. In einem überwachten Pretraining wird das Netz schichtweise gelernt. Das Verfahren wird mit der cuvnet Bibliothek auf GPU implementiert.

Als Neuerung zu vorherigen Arbeiten wird eine online Vorverarbeitung zur Datensatzaugmentation für bessere Generalisierung, Rectifier als Nichtlinearität für schnelleres Lernen und schärfere Ausgaben eingesetzt.

Die Arbeit evaluiert auf den Datensätzen INRIA Graz-02 und NYU Depth v2. Die Genauigkeit der Klassen wird zur Vorarbeit ausgewogener, im Mittel aber kein besseres Ergebnis auf INRIA Graz-02 erzielt. Für RGBD Daten wird eine simple Form von Histogrammen Orientierter Tiefen aus den Tiefeninformationen gewonnen, um diese als weitere Eingabe zu nutzen. Es werden kompetitive Klassifikationsergebnisse auf NYU Depth v2 erreicht. Die Implementierung ermöglicht Vorhersagen für eine hohe Anzahl von Bildern pro Sekunde.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Bildverstehen ist ein wichtiges Themengebiet mit weiten Anwendungsgebieten. Eines der Aufgaben ist die Zuweisung von Bildern zu Klassen. Der nächste Schritt ist die Segmentierung von Bildern in Objekt-Klassen. Dabei ist nicht dem gesamten Bild eine Klasse zuzuweisen, sondern jedem Pixel.

Neuronale Netze werden bereits erfolgreich für die Klassifikation von ganzen Bildern verwendet.

Im Gegensatz zur Klassifikation von Bildern sind für Bildsegmentierung insbesondere lokale Bildinformationen von großer Bedeutung. Globale Bildpositionen sind weniger ausschlaggebend, wenn Objekte beispielsweise über keine typischen Positionen oder Größen verfügen. Ebenfalls erscheint es sinnvoll Informationen aus mehreren Skalen, das heißt aus feinen, als auch gröberen lokalen Relationen, zu extrahieren. In der folgenden Arbeit wird ein Verfahren vorgestellt, welches Konvolutionale Neuronale Netze verwendet (Schulz und Behnke, 2012b). Das Verfahren macht Gebrauch von Multiskalen Eingaben, wiederverwendeten Ausgaben von früheren Schichten, als Form von pretraining (Schulz und Behnke, 2012a), als auch für die Funktionsweise des finalen Netzwerkes. Zuletzt verwendet das Verfahren gelernte paarweise räumliche Klassenrelationen.

In dieser Arbeit wird die Vorverabeitung der originalen durch eine Online Vorverarbeitung ersetzt, wodurch mehr Transformationen und zufällige Kombinationen dieser beim Laden der Eingabe gewählt werden können, um den Datensatz stärker zu augmentieren. Die Transferfunktion wird durch den nichtlinearen Rectifier ersetzt.

Die Arbeit ist wie folgt aufgebaut. Kapitel 2 beschreibt den Aufbau des Konvolutionalen Neuronalen Netzwerk und ferner verwendete Methoden dieser Arbeit. In Kapitel 3 wird diese Arbeit in bezug mit anderen Arbeiten gebracht. Daraufhin werden die Erweiterungen dieser Arbeit in Kapitel 4 erläutert. In Kapitel 5 werden die Experimentellen Ergebnisse beschrieben und in Kapitel 6 werden die Ergebnisse dieser Arbeit diskutiert und ein

Ausblick gegeben.

2 Grundlagen

In diesem Kapitel werden die nötigen Grundlagen der Arbeit behandelt. Dazu wird zuerst auf die grobe Struktur des Neuronalen Netzes eingegangen. Im Anschluss werden Einzelheiten wie die Eingabe, die Konvolution und die einzelnen Schichten der Struktur erläutert. Als nächstes folgt die Beschreibung der Vorverarbeitung der Eingaben, welche u. a. die verwendeten Methoden der Merkmalsgewinnung beschreibt. Danach wird auf die nötigen Elemente zum Lernen wie die Teachermaske, die Lernmethode und Fehlerfunktionen eingegangen. Zuletzt werden die verwendeten Evaluierungsmethoden vorgestellt.

2.1 Konvolutionale Neuronale Netze

Der gewählte Ansatz nutzt ein Konvolutionales Neuronales Netzwerk. Diese Netze sind vergleichbar mit normalen Neuronalen Netzen, mit dem Unterschied, dass aufeinanderfolgende Schichten nicht voll verbunden sind, sondern äquivalent zu einer Konvolution.

Die Konvolutionen verhalten sich entsprechend wie normale Konvolutionsfilter, dessen Koeffizienten mittels überwachtem Training gelernt werden. Dies hat den Vorteil, dass das Netzwerk automatisch Filter lernt, welche lokale Merkmale erkennen. Dies bedeutet allerdings auch, dass Zusammenhänge zwischen weit entfernten Bildpunkten nicht in Betracht gezogen werden.

Die grobe Struktur ist in Abbildung 2.1 dargestellt. Diese Struktur ist eine Erweiterung des Konvolutionalen Netzwerks von LeCun u. a. (1998). Explizit wurde die Struktur um die Multiskalen Eingaben (MS), wiederverwendeten Ausgaben (RO) und den Paarweisen Klassenfilter (PC) erweitert. Außerdem wird keine vollverbundene Schicht zur Ausgabe verwendet, da im Rahmen der Pixelweisen Klassifikation eine bildförmige Ausgabe gewünscht ist.

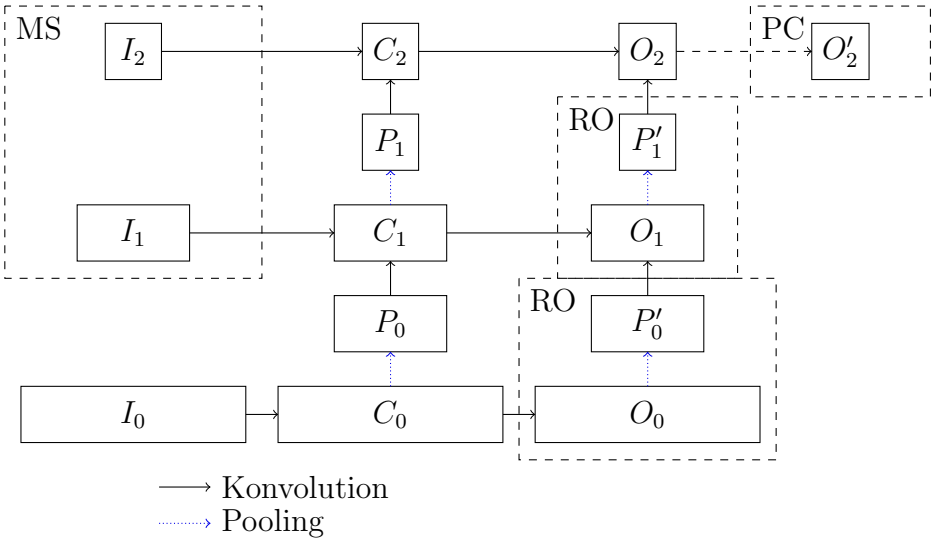


Abbildung 2.1: Struktur des Konvolutionalen Neuronalen Netz. Mit I_i Input, C_i Konvolutionsergebnisse, O_i Output, P_i Pooling Ergebnisse, der i-ten Schicht. MS bezeichnet die Multiskalen Eingaben, RO die Wiederverwendung der Ausgabe niedriger Schichten und PC den paarweisen Klassenfilter.

Das Netzwerk besteht aus drei Schichten, mit je einer Eingabe, Ausgabe und Verbindungen zwischen den Schichten. Zuletzt wird noch eine weitere Konvolution auf der letzten Ausgabe angewendet.

2.1.1 Eingabe

Jede Schicht $i \in \{0, 1, 2\}$ verfügt über eine Eingabe I_i. Die Auflösung der Eingabe wird mit jeder sukzessiven Schicht halbiert, sodass die Eingabe der letzten Schicht I_2 über ein Viertel der originalen Auflösung verfügt. Dies erlaubt dem System Informationen aus verschiedenen Skalen der Eingabe zu extrahieren. Die Eingaben werden, wie in Sektion 2.2 beschrieben, vorverarbeitet.

2.1.2 Konvolutionen

Das Verfahren verwendet Konvolutionen als Hauptoperation der Netzwerkstruktur. Dabei wird bei einer Konvolution auf einer Eingabe, wie in Abbildung 2.2 illustriert, eine Gewichtsmaske pro Merkmalskarte der Eingabe

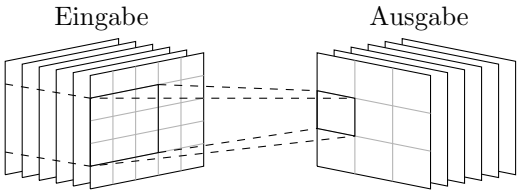


Abbildung 2.2: Illustration einer Konvolution, mit 3×3 Maske, einer 5×5 Eingabe mit sechs Merkmalskarten und entsprechender 3×3 Ausgabe, ebenfalls mit fünf Karten.

mit einer Filtergröße s gelernt. Alle Gewichtsmasken zusammen stellen jeweils einen Filter dar. Die Eingabekarten können einzelne Farbkanäle, vorverarbeite Merkmale oder Ergebnisse vorangehender Konvolutionen sein. Die Summe der Konvolutionsergebnisse über alle Gewichtsmasken, wird als Aktivierung verwendet. Diese Aktivierung würde zunächst eine um $2\lfloor s/2 \rfloor$ kleinere Ausgabeauflösung verfügen, wie in Abbildung 2.2 dargestellt. Daher wird die Eingabe einer Konvolution um einen Rand mit 0 erweitert, um die Dimension der Eingabe zu erhalten und damit Randerscheinungen zu mindern und um auch am äußersten Rand über eine Vorhersage zu verfügen. Auf jede Konvolution erfolgt die Addition eines Bias Terms.

Sei A die Anzahl der Eingabekarten, s eine ungerade Filtergröße, $w_{a,i,j}$ das Gewicht eines Filter für eine Eingabe X an Position i, j der Filtermaske, b der Bias Term, so ist die Ausgabe eines Filters r an der Position x, y definiert mit

$$r(x,y) = b + \sum_{i=-s/2}^{s/2} \sum_{j=-s/2}^{s/2} \sum_{a=1}^{A} w_{a,i,j} \cdot X_{a,x+i,y+j}. \tag{2.1}$$

Die Anzahl der Parameter und damit die Lernkapazität des Netzwerkes kann theoretisch über die Anzahl der Ausgabekarten bzw Filterzahl einer Konvolution frei bestimmt werden. Nur die Ausgabeschicht soll eine Karte pro zu lernender Klasse haben. Das bedeutet, dass für jede Klasse an dieser Stelle ein Filter gelernt wird. Die Filtergröße ist prinzipiell ebenfalls frei wählbar, allerdings haben zu kleine Größen ein zu kleines rezeptives Feld, um genügend weitreichende Relationen erfassen zu können. Zu große Filter hingegen stärken den Einfluss von Randerscheinungen.

2.1.3 Schichten

Wie in Sektion 2.1.1 beschrieben, verfügt jede Schicht i über eine Eingabe I_i.

Die erste Schicht wendet auf I_0 zunächst eine Konvolution an, auf die dann elementweise eine nichtlineare Transferfunktion t angewendet wird, um C_0 zu erhalten. Die Anzahl der Karten ist prinzipiell beliebig und ist der primäre Parameter zur Bestimmung der Lernkapazität. Auf C_0 wird eine weitere Konvolution angewendet, um die Ausgabe O_0 zu erhalten. Diese Konvolution verwendet so viele Filtermasken wie es Klassen im Datensatz gibt. Dies bedeutet, dass das Verfahren für jede Klasse eine Karte ausgibt. In dieser wird der vermutete Grad der Zugehörigkeit eines Bildpunktes zu der jeweiligen Klasse verzeichnet.

Die späteren Schichten $i_{>0}$ sind ähnlich aufgebaut. Lediglich die Eingaben der Konvolutionen zur Berechnung von C_i, O_i sollen um Zwischenergebnisse aus C_{i-1}, O_{i-1} der Vorschichten, erweitert werden. Wie in der originalen Arbeit wird 2×2 Maximum Pooling, auf C_{i-1}, O_{i-1} angewendet, um die räumliche Auflösung zu verringern, die das Verfahren robuster bezüglich kleiner Translationen und Rotationen machen soll (Farabet u. a., 2013). Dies halbiert die Auflösung, sodass die Dimension dieser und der nächsten Schicht kompatibel sind. Max Pooling p ist wie folgt definiert:

$$p(x, y) = \max_{i,j \in \{0,1\}} X_{2x+i,2y+j}. \tag{2.2}$$

Die Pooling Ergebnisse P_i, P_i' werden nach einer Konvolution mit den jeweiligen Zwischenergebnissen der Schicht $i + 1$ zu C_{i+1}, O_{i+1} addiert. Auf C_{i+1} wird die Transferfunktion angewendet. Da die voherige Schicht bereits gelernt wurde, können spätere Schichten auf bereits gelernte Karten zurückgreifen und deren Fehler mittels zusätzlicher Filter auf gröberen Skalen der Eingabe minimieren.

Auf der Ausgabe der letzten Schicht wird noch eine letzte Konvolution, der Paarweise Klassenfilter (PC), eingesetzt. Dieser verwendet Filtermasken mit einem größeren Radius. Als Eingabe bekommt dieser die Karten der letzten Ausgabe und verwendet wieder entsprechend der Anzahl Klassen viele Filter zur Aktivierung. Da die Eingabe-Karten je eine Klasse repräsentieren, lernt dieser Filter paarweise lokale räumliche Klassenrelationen. Beispielsweise könnte gelernt werden, dass Fahrräder selten vom

Himmel umgeben sind.

2.1.4 cuvnet

Zur Implementierung der Netzwerkarchitektur, wird die CUVNET Bibliothek genutzt. Diese Bibliothek ist eine C++ Bibliothek zur Optimierung von Funktionen mittels Gradientenabstiegsverfahren auf der GPU (Schulz, Müller u. a., 2011). Die Bibliothek baut auf der CUV Bibliothek (Schulz, Müller u. a., 2011) auf, welche mittels CUDA die nötigen Grundlagen bereitstellt.

2.2 Vorverarbeitung

Aus Implementationsgründen werden alle Eingaben auf eine einheitliche quadratische Dimension gebracht. Dabei werden drei Skalen der Eingabe verwendet, welche jeweils die Hälfte der Auflösung der Vorschicht verwenden. Zur Eingabe gehören im Sinne dieses Kapitels auch die Ground Truth Masken. Sind in einem gegebenen Datensatz unterschiedliche Formate vorhanden, so werden diese zunächst formaterhaltend auf eine quadratische Maske gelegt ohne Bildinhalte abzuschneiden wie in Abbildung 2.3 illustriert. Die genaue Bestimmung des Hintergrundes wird anhand des Eingabetyps festgelegt. Bei Eingaben für Konvolutionen wird der Inhalt des Bildes am Rand reflektiert, um mögliche Randerscheinungen bei der Konvolution zu verringern. Für Ground Truth Masken wird der Hintergrund als unklassifiziert gesetzt und kann vom Training ausgeschlossen werden. Bilder werden vor der weiteren Verarbeitung mit einem Gaußfilter geglättet, um Rauschen und sehr hohe Frequenzen zu unterdrücken. In Abbildung 2.6 sind Beispiele aller Merkmale auf allen Skalen.

Ferner werden grundsätzlich zwei Methoden der Merkmalsgewinnung verwendet. Erstens ZCA Whitening, welches als eine Form von approximativer Dekorrelation lokaler Bildbereiche dient, und zweitens eine simple Variante des Histogramm Orientierter Gradienten. Aus RGB Daten werden beide Merkmale bestimmt. Steht ein Tiefenkanal zur Verfügung, wird aus diesem analog zum Histogramm Orientiertet Gradienten ein Histogramm Orientierter Tiefen bestimmt.

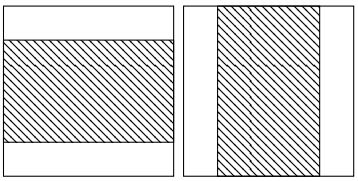


Abbildung 2.3: Beispiel von Eingaben unterschiedlichen Formaten. Die Weiße Fläche zeigt den Hintergrund und das Muster illustriert die daraufgelegte Eingabe.

Während der Trainingsphase kann der gegebene Datensatz über geometrische Bildtransformationen augmentiert werden, um das Netzwerk erstens robuster gegen die verwendeten Transformationen zu machen, als auch die Gefahr von Überanpassung an den Trainingsdatensatz zu senken.

2.2.1 ZCA Whitening

Die Gewinnung der approximativ dekorrelierten Daten erfolgt in zwei Schritten. Erst werden zufällig jeweils P viele $p{\times}p$ große quadratische Bildbereiche x_i aus N RGB Bildern aus dem Trainingsdatensatz als Stichprobe S gewonnen. Jeder Bildbereich x_i stellt eine Zeile in S mit einer Größe von $p{\times}p{\times}3$ dar. Wie in Algorithmus 1 beschrieben, wird die Kovarianz C von S mit $\bar{S}_i$ Mittelwert über Spalte i von S bestimmt

$$C = \frac{1}{P \cdot N} \sum_i \left(S_i - \bar{S}_i\right) \left(S_i - \bar{S}_i\right)^T. \tag{2.3}$$

Mittels Singulärwertzerlegung ($S = U\Sigma V^T$) wird eine Transformationsmatrix Q gewonnen, welche zur Dekorrelation verwendet wird. Dabei werden die Eigenwerte Σ elementweise normiert, sodass alle Merkmalskarten die gleiche Varianz aufweisen.

Mit Q können einzelne Bildbereiche der vorher verwendeten Größe dekorreliert werden. Stattdessen werden nur die Koeffizienten, die den mittleren Pixel festlegen, als $p{\times}p{\times}3$ Konvolutionsfilter genutzt. Die Farbtupel in Q befinden sich jeweils in benachbarten Zeilen bzw Spalten. Das bedeutet, dass sich die relevanten Koeffizienten in den mittleren drei Spalten oder Zeilen befinden. $Q[s,t]$ bezeichne jene Koeffizienten mit s,t Eingabe und Ausgabekanal respektiv. Algorithmus 2 beschreibt die Anwendung.

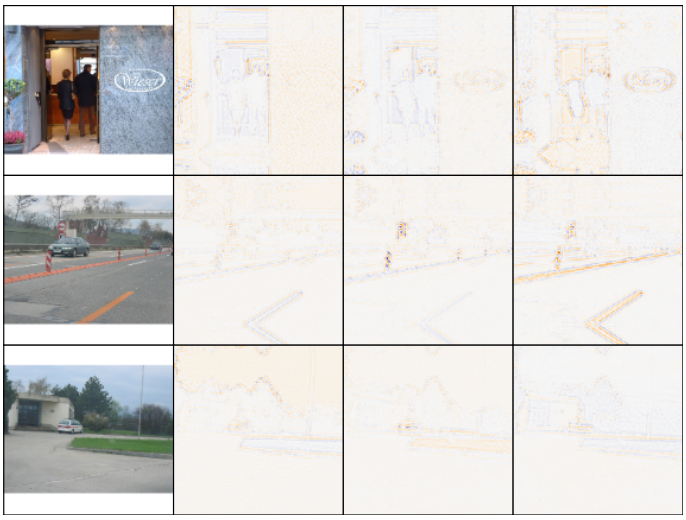


Abbildung 2.4: ZCA transformierte Beispielbilder. Linke Spalte sind die Eingabebilder und die darauffolgenden zeigen die einzelnen Merkmalskarten nach der ZCA Konvolution. Negative Werte werden Blau, positive Werte Rot und 0 wird Weiß dargestellt.

In Abbildung 2.4 sind Beispiele von Dekorrelierten Bildern zu sehen. Eine der auffälligen Eigenschaften dieser Dekorrelation, ist die Unterdrückung von Bereichen mit geringer Variation benachbarter Bildpunkte, wodurch Kanten hervorgehoben werden. Außerdem spiegeln die einzelnen Merkmalskarten nicht mehr jeweils eine der RGB Farben wieder.

2.2.2 Histogramm Orientierter Gradienten

Das zweite Merkmal ist eine simple Form eines Histogramm Orientierter Gradienten und wird in Algorithmus 3 beschrieben. Die RGB Eingabebilder werden erst in Graustufenbilder umgewandelt und mit einem Gaußfilter geglättet. Im Falle eines Tiefenkanals wird dieser nur geglättet. Nach der initialen Glättung der Eingabebilder werden die Gradienten g_x, g_y in x- und y-Richtung mit den Filtermasken $f_x = [-1, 0, 1], f_y = [-1, 0, 1]^T$ bestimmt und aus diesen die Magnitude und Richtung des jeweiligen Gradienten bestimmt. Über die Richtung wird die Magnitude in eine diskrete Menge Richtungen aufgeteilt. Die Interpolation zwischen den aufgeteilten Richtung ist linear respektiv der Richtung. Jeder der Richtungen wird mit einem Gaußfilter geglättet und anschließend normalisiert. Die Normalisie-

Algorithmus 1: Bestimmung der Matrix Q zur konvolutionalen approximativen Dekorrelation

Eingabe : S Stichprobe von Bildpatches
Ausgabe : Q Matrix zur Dekorrelierung, Q^{-1} inverse Transformation

$\mu = \text{Mittelwert}(S)$;
$C_i = Cov(S_i, S_i)$;
$u, s, v = \text{SVD}(C)$;
$s_i^{-1} = \sqrt{s_i}^{-1}$;
if $s_i^{-1} < 0.0000001$ **then**
  $s_i^{-1} = 0$
$Q = u \cdot \text{diag}(s^{-1}) \cdot v^T$;
$Q^{-1} = u \cdot \text{diag}(s) \cdot v^T$;

Algorithmus 2: Approximative lokale Dekorrelation eines Farbbildes

Eingabe : I Farbbild mit $I[c]$ Farbkanal c, Q Dekorrelationsmatrix, $\bar{S}$ Vektor von Mittelwerten mit $S[c]$ Mittelwert von Farbkanal c.
Ausgabe : O dekorreliertes Bild mit $O[c]$ Farbkanal c

$O = 0$;
foreach *Farbkanal* $c \in 0, 1, 2$ **do**
  $I[c] = I[c] - \bar{S}[c]$
foreach *Quellkanal* s **do**
  foreach *Zielkanal* t **do**
    $F = Q_{s,t}$;
    $O[t] = O[t] + convolve(I[s], F)$;

rung sieht eine Normalisierung mit der L_1-norm $= \frac{v}{||v||_1}$, ein abschneiden aller Werte $v = \min(v, 0.2)$ und eine erneute L_1 Normalisierung vor.

Algorithmus 3: Simples Histogramm Orientierter Gradienten

Eingabe : I Graustufenbild, r Anzahl Richtungen
$f_x = [-1, 0, 1], f_y = [-1, 0, 1]^T$;
$g_x = convolve(I, fx), g_y = convolve(I, fy)$;
$m = g_x^2 + g_y^2$;
$a = \text{atan2}(g_x, g_y)$;
foreach *Bildpunkt* x **do**
  m_x und a_x bezeichnen die jeweilige Magnitude und Richtung des Gradienten an Bildpunkt x;
  $bin_x[i]$ bezeichne den i-ten Bin von Bildpunkt x ;
  $b = \lfloor (a_x/\pi \cdot r) \rfloor$;
  $a_1 = \pi/r \cdot b$;
  $a_2 = \pi/r \cdot (b + 1)$;
  $bin_x[b] += m_x \cdot (a_2 - a_x)/(a_2 - a_1)$;
  $bin_x[(b + 1)\%r] += m_x \cdot (a_x - a_1)/(a_2 - a_1)$;
$bin[i] = \text{Gaußfilter}(bin[i])$;
$bin = bin/||bin||_1$;
$bin = \min(bin, 0.2)$;
$bin = bin/||bin||_1$;

2.2.3 Geometrische Transformationen

Während der Trainingsphase werden weitere Transformationen vor der Anwendung der oben beschriebenen Merkmalsgewinnung auf den Eingabebildern angewendet. Die verwendeten Transformationen dienen der Datensatzaugmentierung und sind ausschließlich so gewählt, dass deren Ergebnisse dem Datensatz entsprechende denkbare Bildergänzungen darstellen. Die gewählten Transformationen sind eine horizontale Spiegelung, Skalierung, Rotation und Translation.

Die horizontale Spiegelung dient der Darbietung gespiegelter Merkmale. Ein Fahrrad sieht z. B. von beiden Seiten sehr ähnlich aus, wird also auch von ebenfalls ähnlichen Filtern erkennbar sein, nur gegebenenfalls in gespiegelter oder Spiegelungsunabhängiger Form. Die Skalierung und Ro-

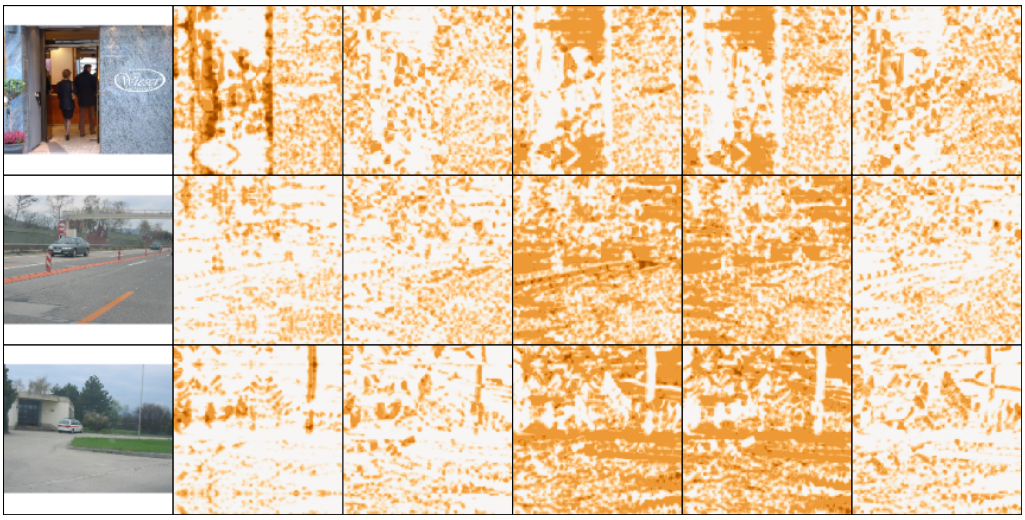


Abbildung 2.5: Beispiel von Histogrammen Orientierter Gradienten. Linke Spalte: Eingabebilder. Die darauffolgenden Spalten sind jeweils ein Bin des Histogramms. Positive Werte werden Rot dargestellt und 0 Weiß.

tation wirken sich vergleichbar zur einer unterschiedlichen Kamerahaltung und Distanz zwischen Kamera und Objekten aus.

Nur die Translation folgt aus einer anderen Motivation. Da der gewählte Ansatz mit Konvolutionalen Filtern arbeitet, verfügt das Verfahren bereits über eine inhärente Robustheit gegenüber Translationen, da ein Konvolutionsfilter auf jeder Position im Bild angewendet wird. Durch die 2×2 Max-Pooling Operationen ergibt sich eine Abhängigkeit bezüglich kleiner Translationen. Befinden sich beispielsweise zwei deutliche Merkmale (große Werte) direkt nebeneinander, so kann die exakte Position z. B. darüber entscheiden ob eines oder beide Merkmale übernommen werden. Da zwei Pooling Operation in Reihe verwendet werden, kann ein Merkmal aus der ersten Schicht aus innerhalb 4×4 verschiedenen Positionen stammen. Letztendlich müssen Filter gelernt, die Robust gegen solche Translationen sind. Die Translationen fördern dies.

Die Transformationen werden beim Laden der Eingabe pro Bild zufällig, in einem vorher festgelegtem Rahmen, bestimmt.

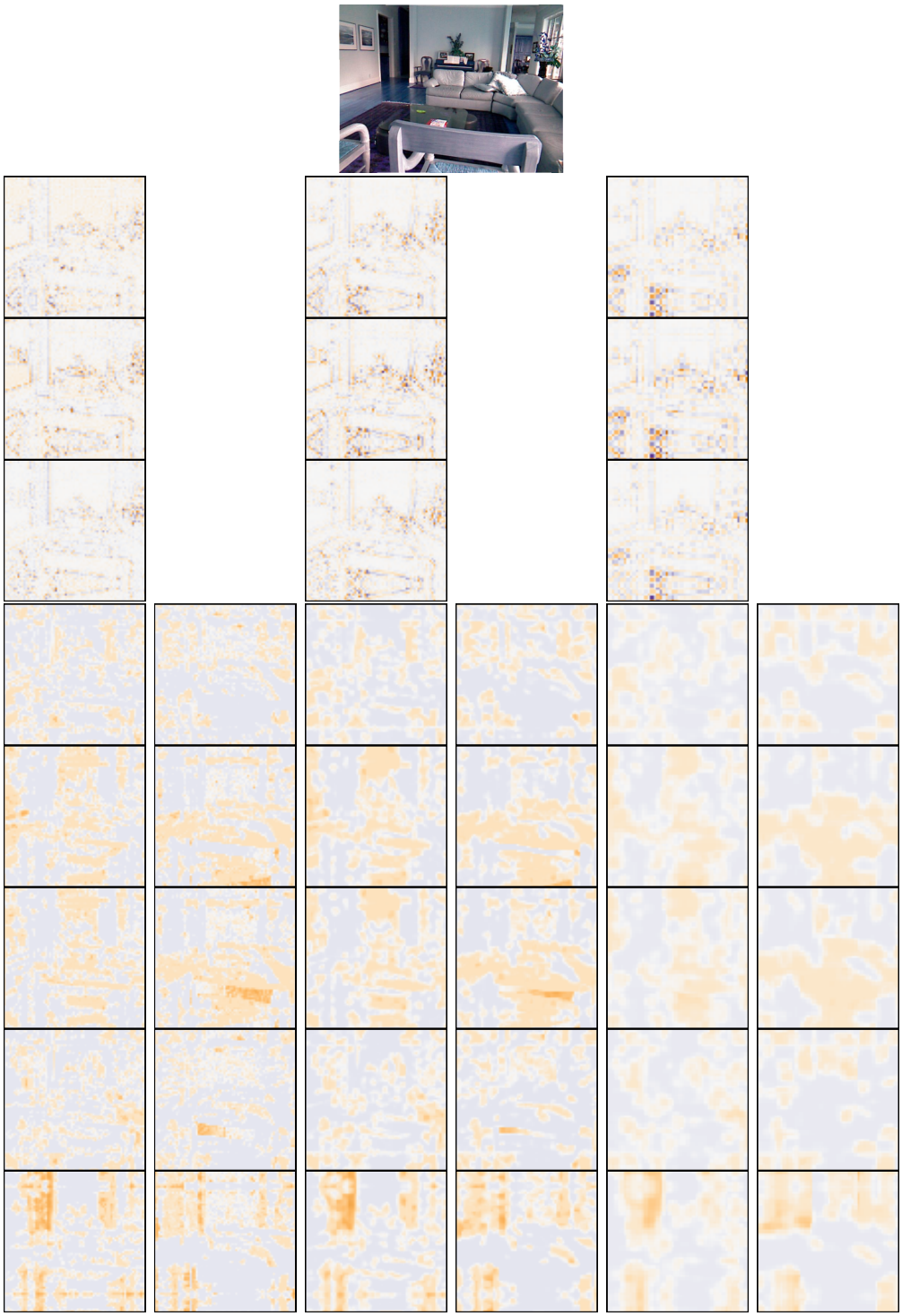


Abbildung 2.6: Illustration aller Merkmale auf allen Skalen.

2.3 Lernen

Die folgenden Abschritte beschreiben die verwendeten Schritte zum Trainieren des Netzwerkes. Begonnen wird mit der Gewinnung der Teacher- und sogenannter Ignoriermasken und daraufhin die verwendeten Fehlerfunktionen, eine Beschreibung des Gradientenabstiegsverfahren und der Verlauf eines Trainingsdurchgangs.

2.3.1 Teacher- und Ignoriermaske

Der Teacher zum lernen wird aus der Ground Truth des jeweiligen Datensatzes abgeleitet. Zunächst wird die Ground Truth auf die Ausgabedimension des Netzwerkes runterskaliert. Die in Sektion 2.2 beschriebenen geometrischen Transformationen werden ebenfalls auf den Teacher angewendet, da die Korrespondenz zwischen der transformierten Eingabe und dem Teacher erhalten bleiben muss. Ferner wird der Teacher kompatibel zur Ausgabe des Netzwerkes aufgebaut. Es wird pro Klasse eine Karte angelegt, sodass in jeder Karte für jeden Bildpunkt mit 1 die Zugehörigkeit bedeutet und ansonsten mit 0 belegt ist.

Zusätzlich wird pro Eingabe eine Ignoriermaske erstellt. Diese Maske wird verwendet, um bestimmte Bildpunkte oder ganze Bereiche vom Training oder der Evaluierung auszuschließen. Dafür wird für jeden Bildpunkt in dieser Maske mit 0 und 1 markiert, ob die Fehlerfunktion an dieser Position grundsätzlich auf Null gesetzt werden soll. Dies hat zur Folge, dass kein Gradient von dieser Position zurückpropagiert und damit nicht zum lernen verwendet wird. Der Implementationseinfachheit halber steht 0 für ignorieren, da die Ignoriermaske so mit dem Ergebnis der Fehlerfunktion multipliziert werden kann, wie in Gleichungen (2.4) und (2.5) beschrieben, um den gewünschten Effekt zu erhalten. Außerdem besteht die Möglichkeit mit dieser Maske Gradienten nach bestimmten Kriterien zu gewichten. Beispielsweise könnte der Fehler einer bestimmten Klasse mit einem Faktor erhöht oder verringert werden und dem Netzwerk damit Prioritäten zuteilen.

Ignoriert werden soll der erweiterte Rand der Eingabe zur Quadratierung der Bilddimension, welcher weder über originalen Bildinhalt, noch über ein Teachersignal verfügt. Außerdem verfügen die verwendeten Da-

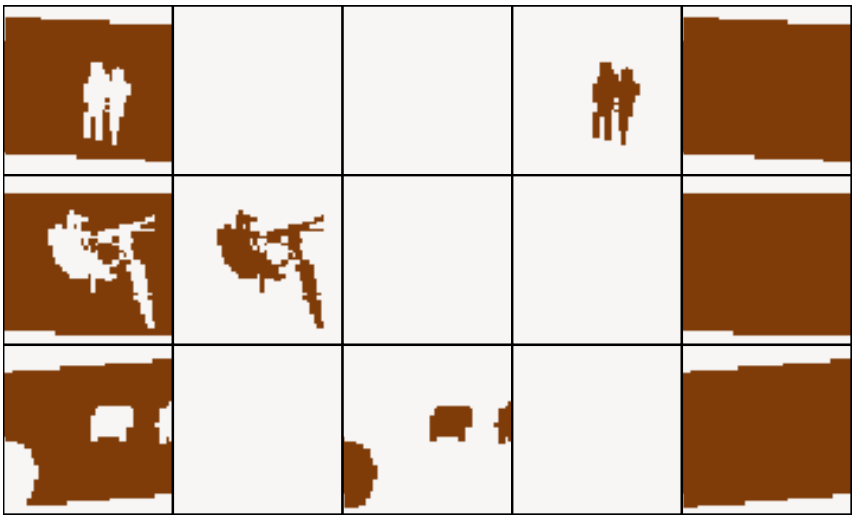


Abbildung 2.7: Beispiele von Teacher und Ignoriermasken nach Vorverarbeitung. Von Links nach Rechts sind die Hintergrundklasse, Fahrrad, Autos, Personen und die Ignoriermaske dargvestellt.

tensätze über nichtklassifizierte Bereiche oder (teilweise zusätzlich) explizit als "Unbekannt" klassifizierte Bereiche. Ersteres tritt z. B. auf, wenn ein Datensatz auf spezifische Klassen beschränkt wurde, wodurch der Rest der Bilder als Hintergrundklasse betrachtet werden kann, wie im Falle vom INRIA Graz-02 Datensatz. Oder aber die Klassifizierung wäre zu schwer für ein Lernverfahren, oder aber schon zu schwer für den Menschen der das Teachersignal bereitzustellen versuchte. Insbesondere direkt zwischen zwei Objekten kann die Pixelgenaue Zuweisung zu einem Objekt schwierig Fallen. In allen Fällen besteht die Möglichkeit diese Bereiche zu ignorieren und von der Evaluierung auszuschließen. Dies wird vom Datensatz abhängig gemacht.

2.3.2 Trainingsablauf

Zum Lernen wird eine Form von Pretraining verwendet (Schulz und Behnke, 2012a). Dabei werden die Schichten aufsteigend nacheinander trainiert. Das heißt, zunächst wird die erste Schicht auf der größten Eingabedimension trainiert. Die weiteren Schichten werden nach und nach hinzugefügt. Während spätere Schichten trainiert werden, werden die Gewichte früherer Schichten weiterhin mittrainiert. Der Fehler wird über die einzelnen Aus-

gabekarten O der aktuell zu trainierenden Schicht, des Teacher Y und der Ignoriermaske I bestimmt. Dabei bezeichne $o_{k,i}, y_{k,i}$ die Ausgabe bzw Teacher von Bildpunkt i zur klasse k und I_i die Ignoriermaske zu Bildpunkt i. In diesem Kontext bezeichne n die Zahl der Bildpunkte. Entweder wird die Multinomiale Logistische Fehlerfunktion E_{ml} mit

$$E_{ml}(Y, O) = -\frac{1}{n} \sum_i \left(\sum_k y_{ik} o_{ik} - \log \sum_k e^{o_{ik}} \right) \cdot I_i \tag{2.4}$$

oder die pixelweise quadrierte ϵ-insensitive Fehlerfunktion E_{ei} mit integrierter Nichtlinearität

$$E_{ei}(Y, O) = \frac{1}{n} \sum_i \left(\max\left(0, \left| y_i - (1 + \exp(o_i))^{-1} \right| - \epsilon \right)^2 \right) \cdot I_i \tag{2.5}$$

verwendet. Die Logistische Fehlerfunktion E_{ml} bevorzugt, dass die gewünschte Klasse lediglich stärker in der Ausgabe vertreten ist, als die konkurrierenden Klassen. Die absoluten Werte sind zweitrangig. Diese Funktion wird verwendet, wenn die beste pixelweise Vorhersage gewünscht ist. Die ϵ-insensitive Fehlerfunktion E_{ei} bestimmt den Fehler pro Klasse. Ist die beste Vorhersage pro Klasse gewünscht, wird diese Funktion verwendet.

Bei den Maximum Pooling Operationen wird der Fehler nur an das maximale Element weitergegeben. Zum Schluss wird der PC trainiert. Auch hier werden weiterhin alle Gewichte mittrainiert.

Als Gradientenabstiegsverfahren werden Rmsprop, eingeführt von Tieleman und Hinton (2012). Rmsprop verwendet einen gleitenden Mittelwert $\tilde{\partial}$ über den Gradienten um den Gradienten zu normieren. Bezeichne $\frac{\partial E}{\partial w}$ den Gradienten eines Gewichtes in der aktuellen Lernphase, so wird der gleitende Mittelwert $\tilde{\partial}(t)$ zum Zeitpunkt t wie folgt angepasst

$$\tilde{\partial}(t) = 0.9\tilde{\partial}(t-1) + 0.1 \left(\frac{\partial E}{\partial w} \right)^2. \tag{2.6}$$

Der Gewichtsanpassung Δw wird nun mit diesem Mittelwert dividiert

$$\Delta w = -\frac{\eta}{\tilde{\partial}} \frac{\partial E}{\partial w}. \tag{2.7}$$

Dies hast zur Folge, dass zeitlich betrachtet aufeinanderfolgende Gradienten, welche sich in ihrer Magnitude nicht zu sehr unterscheiden, auf eine einheitlich Größe normiert werden. Das heißt, die Magnitude des Gradienten wird großteils entfernt und hauptsächlich nur noch das Vorzeichen verwendet, also die Schrittgröße bei der Anpassung konstant gehalten.

2.3.3 Early Stopping

Auf dem Validierungssatz wird Early Stopping als Abbruchkriterium genutzt. Dabei wird alle paar Epochen die Fehlerfunktion auf dem Validierungssatz ermittelt. Der auf dem Validierungssatz gemessene Fehler wird mit einem Boxfilter geglättet. Das heißt der Mittelwert der letzten Ermittlungen wird als Maß herangezogen. Zunächst wird dem Netzwerk eine gewisse Epochenzahl gewährt, bevor das Abbruchkriterium wirksam wird. Bessert sich das Ergebnis innerhalb der Gewährungszeit nach einem bestimmten Kriterium, wird die Gewährungszeit auf die aktuelle Epochenzahl multipliziert mit einem Faktor gesetzt. Das verwendete Kriterium ist die Besserung des Klassifikationsfehlers oder der Fehlerfunktion um einen festgelegten Prozentsatz seit dem letzten Erfüllen des Kriteriums.

2.4 Evaluierung

Um die Güte eines gelernten Netzwerkes zu bestimmen, muss ein Fehlermaß auf der Prediktion und der Ground Truth bestimmt werden. Das zu verwendende Fehlermaß kann vom Datensatz vorgegeben sein.

Die Pixelweise Genauigkeit beschreibt den Anteil der korrekt klassifizierten Bildpunkte. Das bedeutet, dass alle Werte gleichmäßig in die Bewertung einfließen, was zur Folge hat, dass häufiger vertretende Klassen eine größere Auswirkung auf das Fehlermaß haben. Dabei wird die Ignoriermaske herangezogen, um gewünschte Bildbereiche von der Evaluierung auszuschließen. Sei p die Anzahl korrekt klassifizierter Pixel und t die Totale Anzahl Pixel ist die Genauigkeit a_p mit

$$a_p = \frac{p}{t} \tag{2.8}$$

gegeben. Für die Mittlere Genauigkeit a_m über alle Klassen wird für jede Klasse einzeln der Anteil korrekt klassifizierter Punkte ermittelt und der

Durchschnitt über die jeweiligen Anteile bestimmt. Bezeichne p_k die Anzahl korrekt klassifizierten Pixel der Klasse k und t_k die totale Anzahl Pixel dieser Klasse und K die Anzahl der Klassen, dann gilt

$$a_m = \frac{1}{K} \sum_k \frac{p_k}{t_k} i. \tag{2.9}$$

Für INRIA Graz-02 wird für jede Klasse einzeln jeweils ein Schwellwert für eine Binarisierung der Ausgabe der Vorhersage so gewählt, dass die gleiche Precision p wie Recall r erreicht wird. Bezeichne t_p die korrekt erkannten, f_p die falsch erkannten und f_n falsch nichterkannten, dann sind p und r wie folgt definiert:

$$p = \frac{t_p}{t_p + f_p} \tag{2.10}$$

$$r = \frac{t_p}{t_p + f_n} \tag{2.11}$$

Für die Evaluierung der Ergebnisse muss die Vorhersage des Netzwerkes zunächst an das Format der Ground Truth Werte angepasst werden. Das heißt die aufgestockten Randbereiche wieder entfernt und auf die ursprüngliche Auflösung skaliert werden.

3 Verwandte Arbeiten

Kumar und Koller (2010) stellen ein Verfahren zur Zerlegung von Bildern in Regionen vor, welches iterativ Regionen direkt in Abhängigkeit einer Energiefunktion bestimmt. Sie erzielen gute Ergebnisse auf dem Standford Background Datensatz. Die Berechnungsdauer ist eine Schwäche dieses Verfahrens.

Häufig werden Conditional Random Fields eingesetzt (Fulkerson u. a., 2009; Ladický u. a., 2009; Lempitsky u. a., 2011; Nowozin u. a., 2010; Plath u. a., 2009; Schroff u. a., 2008). Die CRFs werden auf verschiedene Arten von Features, wie beispielsweise auf hierachischen Segmentationen, angewendet. Lempitsky u. a. (2011) stellen ein Modell vor, welches Bilder in Form eines Baumes segmentiert. Das Verfahren ist der aktuelle Stand der Technik auf dem Standford Background Datensatz. Die Anwendungsgeschwindigkeit hängt linear von der Anzahl der Klassen ab. Dabei verzeichnen Farabet u. a. (2013) mit ihrem Konvolutionalen Neuronalen Netz, um eine Größenordnung, schnellere Anwendungszeiten. Einen Vorteil der, insbesondere auf GPU, mit dem Verfahren dieser Arbeit ebenfalls erwartet werden kann.

Wenige Publikationen beschäftigen sich mit parameterfreien Verfahren (Liu u. a., 2011; Tighe und Lazebnik, 2010). Die beiden Verfahren versuchen das Problem mittels Korrespondenzen zwischen Eingabebildern und einer Datenbank annotierter Bilder zu lösen. Der Vorteil dieses Vorhabens ist, dass kein Training benötigt wird und die Güte des Verfahrens durch eine Erweiterung der Datenbank erhöht werden kann. Nachteilig ist die Abhängigkeit einer großen Datenbank von Bildern bei der Anwendung des Verfahrens. Neuronale Netze hingegen unterliegen einer Trainingsphase, aber die Ausführung eines gelernten Netz ist sehr effizient.

Unter anderen verwenden Couprie u. a. (2013); Farabet u. a. (2013); Grangier u. a. (2009); Socher u. a. (2011) Neuronale Netze. Dabei unterscheiden sich die Netzstrukturen vom Ansatz der kommenden Arbeit. Gran-

gier u. a. (2009) nutzen eine "Deep-learning" Strategie. Trainiert werden zunächst drei Konvolutions-Schichten und eine lineare Ausgabeschicht. Die Konvolutions-Schichten bestehen aus einer Konvolution und einer Sigmoiden Funktion. Nach dem training werden weitere Konvolutions-Schichten hinzugefügt, die auf den bereits gelernten Schichten antrainiert werden. Das Verfahren dieser Arbeit unterscheidet sich in seiner Architektur. Die Eingabe wird bei unterschiedlichen Skalen verwendet, außerdem wird die Ausgabe jeder Schicht wiederverwendet.

Socher u. a. (2011) verwenden ein Neuronales Netzwerk rekursiv. Dabei wird ein Bild Segmentiert, Bildfeature über die Segmente generiert, welche dann als Eingabe für ein Neuronales Netz dienen. Das Rekursive Neuronale Netz entscheidet, ob benachbarte Regionen vereinigt werden sollen und zu welcher Klasse das Segment am wahrscheinlichsten gehört. Damit versucht das Verfahren natürliche rekursive Zugehörigkeitsrelationen zu erkennen, wie beispielsweise ein Fenster im Regelfall zu einem Gebäude gehört. Dies unterscheidet sich grundlegend vom Neuronalen Netz dieser Arbeit, in der Eingabe und Struktur, da es nicht auf einer vorangehenden Segmentation aufbaut.

Couprie u. a. (2013) verwenden ein Neuronales Netz auf mehreren Eingabeskalen. Im Gegensatz zu dieser Arbeit wird ein Netz auf alle Skalen angewendet. Die kleineren Ausgaben werden auf die Dimension der feinsten hochskaliert und bilden zusammen mit Superpixeln auf dem Eingabebild die Vorhersage. Der mögliche Vorteil der Architektur dieser Arbeit ist, dass jede Skala innerhalb der Netzstruktur zum Ergebnis beiträgt und die Ergebnisse der feineren Skalen als Grundlage der nächst gröberen Schicht dient und dessen Ergebnis verbessern kann. Für die Tiefeninformationen sehen Couprie u. a. (2013) vor, sodass lokale Nachbarschaften über einen Mittelwert von null und Standardabweichung von eins aufweisen. Diese Arbeit wendet auf den Tiefeninformationen eine simple Variante Histogramm Orientierter Gradienten.

Spinello und Arras (2011) stellen Histogramme Orientierter Tiefen vor. Dabei handelt es sich um eine Merkmalsgewinnung auf Tiefeninformationen, mit Analogien zu Histogrammen Orientierter Gradienten (Dalal und Triggs, 2005). Diese Arbeit verwendet eine vereinfachte Variante Histogramme Orientierter Tiefe, da bildförmige Merkmale gewünscht sind, damit diese direkt als weitere Merkmalskarten der Eingabe hinzugefügt werden können.

4 Neuerungen

Tieleman und Hinton (2012) stellen eine Gradientenabstiegsverfahren vor, welches, vergleichbar mit Rprop, versucht die Magnitude des Gradienten unberücksichtigt zu lassen. Vorläufige Experimente zeigten eine darauf resultierten Robustheit gegenüber Fehlerfunktionen, dessen Gradienten deutlich schwanken können.

In neueren Publikationen wurden mit Rectifier als Nichtlineare Transferfunktion gute Ergebnisse erzielt. Diese Funktion verfügt über keine obere Sättigungsgrenze, wodurch Netzwerke schneller trainiert werden können (Glorot u. a., 2011; Krizhevsky u. a., 2012). Daher wird dies in dieser Arbeit verwendet und beobachtet, ob die einseitig fehlende Saturation sich auch auf die Ausgabe auswirkt.

In der originalen Arbeit wurde die Datenaugmentation des Trainingsatzes Offline erledigt und auch nur eine Spiegelung verwendet. Das heißt es wurde einmalig ein neuer Datensatz mit den originalen Bildern und gespiegelten Versionen angelegt. Diese Methode hat den Nachteil, dass für jede gewünschte Kombination von Transformation ein neues Bild angelegt und insbesondere von vornerein festgelegt werden muss. Diese Arbeit verwendet die in Sektion 2.2 beschriebenen Transformationen online, also für jedes zum Lernen eingelesene Bild wird eine zufällige Kombination von Transformationen angewendet. Dies daraus entstehenden Kombinationen sind stufenlos.

Für Datensätze mit Tiefeninformationen werden analog zur HOG Vorverarbeitung eine simple Variante Histogramme Orientierter Tiefen verwendet, da die simplen HOG Merkmale bereits erfolgreich verwendet wurden (Schulz und Behnke, 2012b).

Dank der Entwicklung der Microsoft Kinect, einer Kamera zur einfachen Aufnahme von RGB Bildern und Tiefeninformationen, haben sich neue Möglichkeiten im Bereich Bildverstehen eröffnet. Zur Nutzung innerhalb dieser Netzstruktur wird aus dem Tiefenkanal ebenfalls ein Histogramm Orientierter Gradienten ermittelt.

5 Experimente

In diesem Kapitel wird der experimentelle Teil der Arbeit beschreiben. Zunächst werden die verwendeten Parameter genannt, die zumindest für die meißten Tests verwendet werden. Daraufhin wird ein Blick auf die genutzten Datensätze geworfen. In den darauffolgen Sektionen werden die Experimente auf den jeweiligen Datensätzen beschrieben, diskutiert und zuletzt Ergebnisse auf den Testdaten mit anderen Arbeiten verglichen.

5.1 Parameter

Wenn nicht in der Beschreibung eines Tests ausdrücklich vermerkt, werden folgende Parameter in den Experimenten verwendet.

Die Eingabe wird auf einheitliche quadratische 196×196 in der feinsten Skala und damit in 98×98 und 49×49 auf den jeweiligen folgenden Skalen gebracht. Für Konvolutionen werden jeweils 32 Filter genutzt. Ausnahme sind die Konvolution zur Ausgabe mit vier Filtern, für vier Klassen. Die Filter der Konvolutionen verwenden eine Größe von 7×7, mit Ausnahme des Paarweisen Klassenfilters, der eine Größe von 11×11 bekommt. Für die Berechnung des Histogramms Orientierter Gradienten werden auf RGB Eingaben, als auch für die Tiefeninformationen, fünf Richtungen bestimmt. Das bedeutet für RGB Daten erhält das Netz acht Eingabekarten und für Datensätze mit Tiefeninformationen 13.

Die Filter, die direkt an einer Ausgabe beteiligt sind werden mit Null initialisiert. Alle anderen Filter per Normalverteilung mit einer Standardabweichung von 0.015 zufällig initialisiert. Bias Gewichte werden mit 0.1 initialisiert, um anfänglich Konvolutionsergebnisse überhalb der Sättigung der Transferfunktion zu erhalten.

Für Transformationen wird die Spiegelung mit einer Wahrscheinlichkeit von 50% gewählt, der Skalierungsfaktor wird zufällig zwischen $[0.9, \ldots, 1.1]$ ausgesucht, rotiert wird um einen Wert innerhalb von $[-5°, \ldots, 5°]$ und für

die Translation wird getrennt für beide Richtungen bis zu sieben Bildpunkte angesetzt.

5.2 Datensätze

5.2.1 Inria Graz 02

Der Datensatz INRIA Graz-02 besteht aus 479 Trainings- und 479 Testbildern mit einer Auflösung von 640×480 im Hoch- und Querformat und lediglich drei Klassen (Marszałek und Schmid, 2007). Beispiele in Abbildung 5.1. Die drei Klassen sind Autos, Fahrräder und Personen. Jedes Bild zeigt nur Objekte aus einer Klasse. In Bildern von Fahrrädern befinden sich daher z.B. keine Autos. Es können dabei mehrere Objekte dieser Klasse vorhanden sein. Die Objekte innerhalb der Bilder sind nicht immer im Mittelpunkt, noch in einer ähnlichen Größe oder Position im Bild und können auch stark verdeckt sein. Jedes Bild kann über mehrere Ground Truth Masken verfügen. Wenn immer Objekte klar voneinander getrennt werden konnten, wurde für jede Instanz eine Maske bereitgestellt. Verdeckte Parts eines Objektes sind ebenfalls in den Ground Truth Masken vorhanden, jedoch wird in dieser Arbeit keine Verwendung davon gemacht. Dieser Datensatz verfügt nur über wenige Klassen und keine explizite Hintergrundklassen. Ein sehr großer Anteil des Datensatzes ist unklassifiziert.

Für den Teacher werden alle alle Masken überlagert, die Verdeckung ignoriert und die unklassifizierten Punkte als Klasse “Hintergrund” gesetzt, da diese Bereiche keiner der anderen drei Klassen angehören. Die andere Möglichkeit wäre diese Bereiche zu ignorieren. Das hätte zur Folge, dass das Netz diese Bereiche beliebig klassifizieren kann, was sich negativ auf die Evaluierung auswirken würde.

5.2.2 NYU Depth v2

Zur Evaluation der Tauglichkeit des Verfahrens bei RGB-D Daten wird der von Silberman u. a. (2012) eingeführte Datensatz NYU Depth v2 herangezogen. Dieser stellt 1449 Bilder mit Tiefenfinformationen und dichten Annotationen. Die Aufnahmen zeigen Innenräume mit unkontrollierter Beleuchtung und üblichen Verdeckungen von Gegenständen. Abbildung 5.2

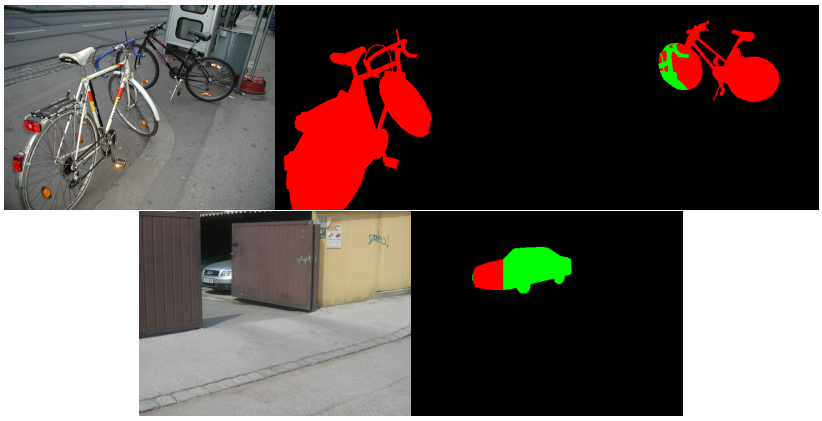


Abbildung 5.1: INRIA Graz-02 Datensatz Beispiele. Obere Zeile: Beispielbild der Klasse Fahrrad mit seinen zwei Masken. Untere Zeile: Ein Beispiel der Klasse der Auto.

zeigt Beispiele. Die Aufnahme wurden mit der Microsoft Kinect erstellt. Diese produziert "Lücken" in den Tiefeninformationen, wenn die Ermittlung der Tiefeninformation nicht möglich war. Diese Lücken müssen mittels einer Vorverarbeitung aufgefüllt werden müssen. NYU Depth v2 liefert bereits kompensierte Tiefeninformationen mit, welche für diese Arbeit genutzt werden. Alle Bilder liegen in 640×480 vor.

Grundsätzlich wurden die Bilder in 894 verschiedene Klassen segmentiert. Darunter befinden sich unter anderen verschiedenfarbige Pfeffersorten. Silberman u. a. führen die vier abstrakten Klassen "Boden", "feste Strukturen", "Möbel" und "Requisiten" ein. Die festen Strukturen beziehen sich auf Wände, Decken oder sonstige unbewegliche Strukturen. Mit Requisiten sind kleinere Gegenstände gemeint, die sich leicht bewegen lassen.

5.3 Inria Graz 02

In dieser Sektion werden die Experimente auf dem Datensatz Inria Graz 02 dokumentiert.

Wie in Sektion 5.2.1 beschrieben zeigt die Ground Truth Maske für Inria Graz 02 Datensatz zum größten Teil "Hintergrund". Um dem Netzwerk eine günstige Startsituation zu verschaffen wurde experimentell eine Initialisierung der Biasgewichte gelernt. Hierfür wurde die Lernrate aller Gewichte

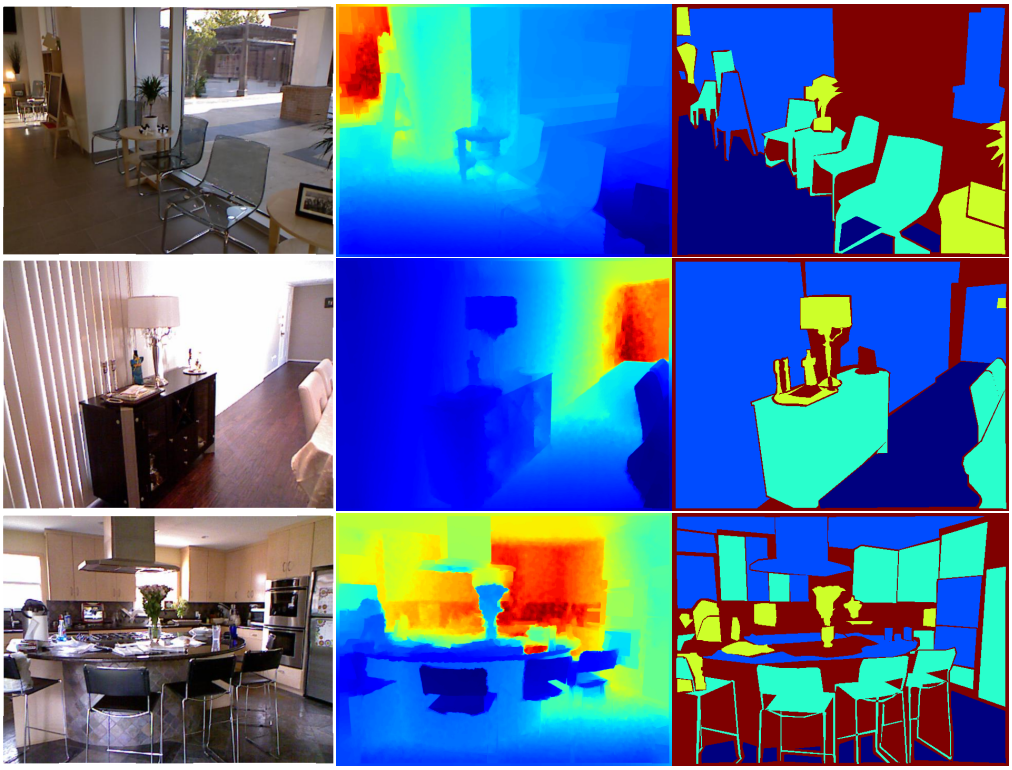


Abbildung 5.2: NYU Depth Dataset v2 Beispiele. Die Spalten von links nach Rechts zeigen die RGB Bilder, Tiefeninformationen, Ground Truth Maske der vier abstrakten Klassen.

abgesehen der Biasgewichte an der Ausgabe auf Null gesetzt und für ein paar Epochen trainiert. Die Gewichte werden für die Initialisierung aller Konvolutionen an der Ausgabe verwendet. Als Gradientenabstiegsverfahren wird Rmsprop mit einer Batchgröße von 16 verwendet. Der originale Trainingsdatensatz wird in 383 Trainings- und 96 Validierungsbilder aufgeteilt. Eine Epoche sind als 20 Batches festgelegt. Ein vollständiger Durchlauf auf dem Trainingsdatensatz sind ein paar Epochen mehr, allerdings werden Statistiken über das Fehlermaß z.B. pro Epoche erhoben, sodass eine kleinere Zahl Batches häufiger Feedback produzieren. Für diesen Datensatz wurden keine Ignoriermasken verwendet.

5.3.1 Vereinfachte Netzstruktur

Der erste Versuch soll die Implementierung testen. Die Netzstruktur wird vereinfacht. Die in Sektion 2.1 beschriebene Multiskalen Eingaben, Wiederverwendeten Ausgaben und der Paarweise Klassenfilter werden nicht verwendet. Auch Pretraining wird nicht verwendet. Das verwendete Gradientenabstiegsverfahren ist Rmsprop mit einer Lernrate von 0.0001. Die intiale Gewährungsphase für Early Stopping beträgt 64 Epochen, verlangt wird eine Verbesserung um 2%, um die Gewährungszeit um einen Faktor von 1.5 zu verlängern. Die Trainingsphase wird nach maximal 2500 Epochen beendet.

In Abbildung 5.3 ist der Verlauf des Fehlers auf den Trainings- und Validierungsdaten, sowie des Klassifikationsfehler, zu sehen. Der Klassifikationsfehler, welcher den Anteil der korrekt klassifizierten Punkte misst, ist ein Fehlermaß, welcher nicht zur Evalierung auf diesem Datensatz genutzt wird. Dennoch ist er während der Trainingsphase interessant zu beobachten. Das Experiment dauerte die vollen 2500 Epochen. Die Fehlerfunktion wird auf dem Trainings- als auch auf dem Validierungssatz bis zum Ende des Trainings reduziert. Dies bedeutet, dass kein merklich negatives Overfitting auf dem Trainingsdatensatz stattfindet. Dies kann bedeuten, dass die Datenaugmentation Overfitting erschwert, die Parameterzahl und damit die Lernkapazität gerade so passend ist, oder wahrscheinlicher, dass die Kapazität des Netzwerkes nicht groß genug ist.

Abbildung 5.4 zeigt 16 von 32 Filtern der Gewichte W_0 der ersten Schicht an der Eingabe, einen 8×8 großen Ausschnitt der Gewichte W_1 der zweiten

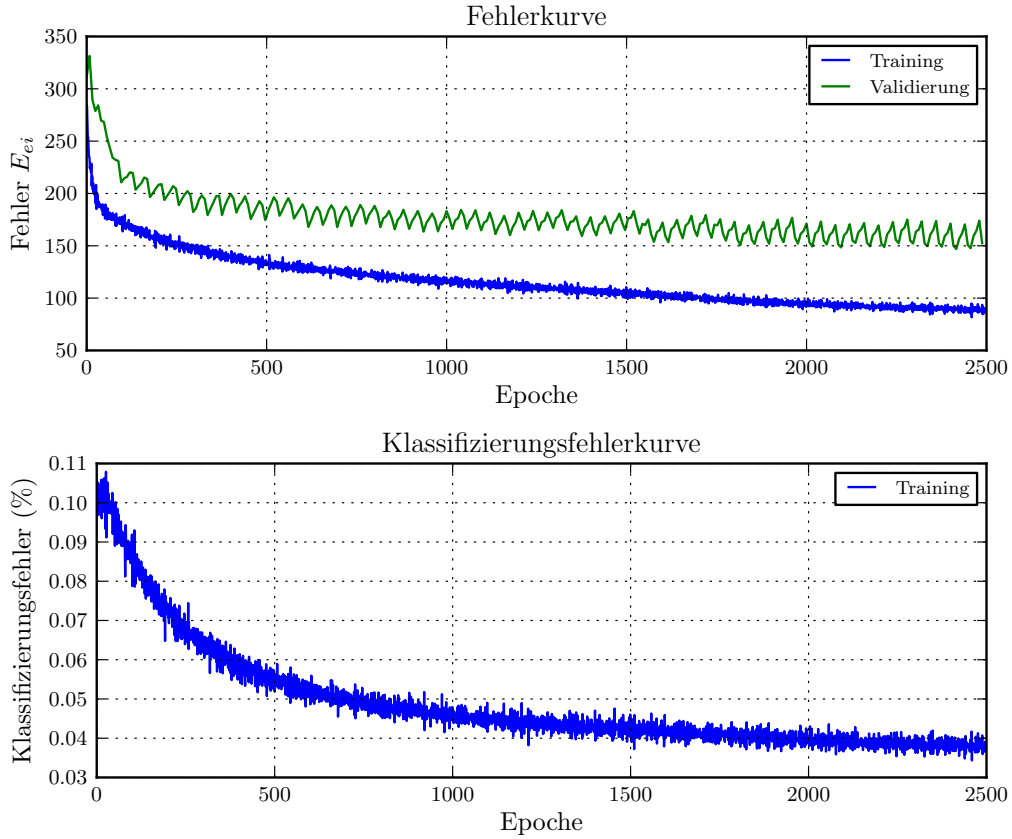
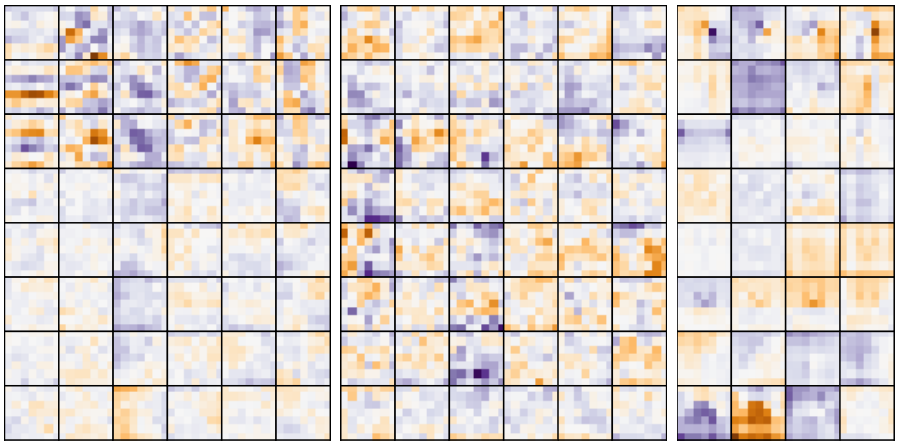


Abbildung 5.3: Verlauf der Fehlerfunktion und Klassifikationsfehlers der vereinfachten Netzstruktur auf INRIA Graz-02.



(a) W_0 **(b)** W_1 **(c)** W_2

Abbildung 5.4: Ein Ausschnitt aller Filter. W_0 Filter auf der Eingabe, W_1 Filter der zweiten Konvolution, W_2 Filter der Ausgabe. Negative Werte werden Blau, Positive Rot und 0 mit Weiß dargestellt.

Konvolution und einen 8×4 großen Ausschnitt der Filter der Ausgabekonvolution. Zunächst sei gesehen, dass eindeutige Strukturen erkennbar sind, ein grundsätzliches Zeichen dafür, dass die Gewichte gelernt wurden. Jede Zeile der Gewichte von W_0 zeigt Masken für jeweils eine Eingabekarte. Die ersten drei Zeilen werden auf die ZCA Karten und die letzten fünf auf den Gradientenkarten angewendet. Da alle Filter innerhalb einer Gewichtsmatrix gemeinsam normiert worden sind, können deren relative Unterschiede beurteilt werden. Dabei fällt auf, dass die Filter auf den Gradienten schwächer ausgeprägt sind, dafür aber mehr Masken auf den Gradienten arbeiten. Auch ist zu sehen, dass die Filtermasken für die ZCA Karten feinere Strukturen zeigen, was daran liegt, dass die ZCA Karten bereits feinere Strukturen zeigen als die Gradienten. Masken späterer Schichten neigen zu gröberen Strukturen und scheinen u.a. darauf abzuzielen die Ausgabe der Vorschicht zu glätten.

Auf dem Testdatensatz erreichte dieses Netzwerk eine Precision von 70.5%, 67.3% und 57.1% auf den jeweiligen Klassen Fahrrad, Auto und Person. Dieses Ergebnis erreichte nicht die Leistung der originalen Arbeit mit allen Features.

5.3.2 Vollständige Architektur

Nach der vereinfachten Struktur soll das Netzwerk mit allen Erweiterungen getestet werden und mit dem Ergebnis der originalen Publikation verglichen werden.

In Abbildung 5.5 ist der Verlauf der Fehlerfunktion und Klassifizierungsfehler zu sehen. Auffällig im Vergleich zur vereinfachten Struktur sind nun drei Sprünge im Verlauf zu sehen. Diese entstehen durch den Wechsel der Lernphase. Dies bedeutet die Ausgabe wird nun von einer untrainierten Schicht bestimmt. Durch die untrainierte Schicht ist erst mal ein schlechteres Ergebnis zu erwarten, allerdings fällt die Fehlerfunktion die ersten beiden Wechsel. Unglücklicherweise beeinflusst die Dimension der Ausgabe die Fehlerfunktion mit einem proportionalem Faktor, weshalb der Fehler automatisch sinkt. Im letzten Wechsel wird die Ausgabedimension nicht geändert und es ist erst ein leichter Anstieg des Fehlers zu verzeichnen. Die erstmalig schlechtere Ausgabe macht sich im Klassifizierungsfehler bei allen Wechseln deutlich bemerkbar. Dieser Faktor der Fehlerfunktion würde bei Verwendung von Gradientenabstiegsverfahren, welche die Magnitude des Gradienten verwenden, Probleme verursachen. Die Normierung von Rmsprop entfernt diesen Faktor.

In Abbildung 5.6 werden ein Teil der Filtermasken auf den drei Skalen der Eingabe dargestellt. Die Filter auf der gröbsten Skala sind vergleichbar mit den Filtern des ersten Tests, mit der vereinfachten Struktur. Interessant ist, dass die Filtermasken auf den Gradienten (ab Zeile vier in der Abbildung) von Skala zu Skala stärker vertreten sind (höhere Magnitude). Eine Betrachtung der Eingabekarten ergab, dass die Standardabweichung der ZCA Karten der gröberen Skalen zunimmt. Die Filter scheinen sich diesem Umstand anzupassen. Es könnte aber besser sein, diesen Umstand garnicht erst auftreten zu lassen.

Die Filter zur Ausgabe einer Schicht zeigen Ähnlichkeiten untereinander und dem Filter der Ausgabeschicht des vereinfachten Netzwerkes. Ein Ausschnitt dieser Filter ist in Abbildung 5.7 zu sehen. Diese Filter zeigen also häufig grobe Strukturen und glätten ihre Eingabe.

Abbildung 5.8 zeigt die Filter zwischen den Schichten und den Paarweisen Klassenfilter. Die Filter in (a), (b) werden genutzt, um die aus der Eingabe gewonnen Merkmale an die nächste Schicht weiterzureichen. Be-

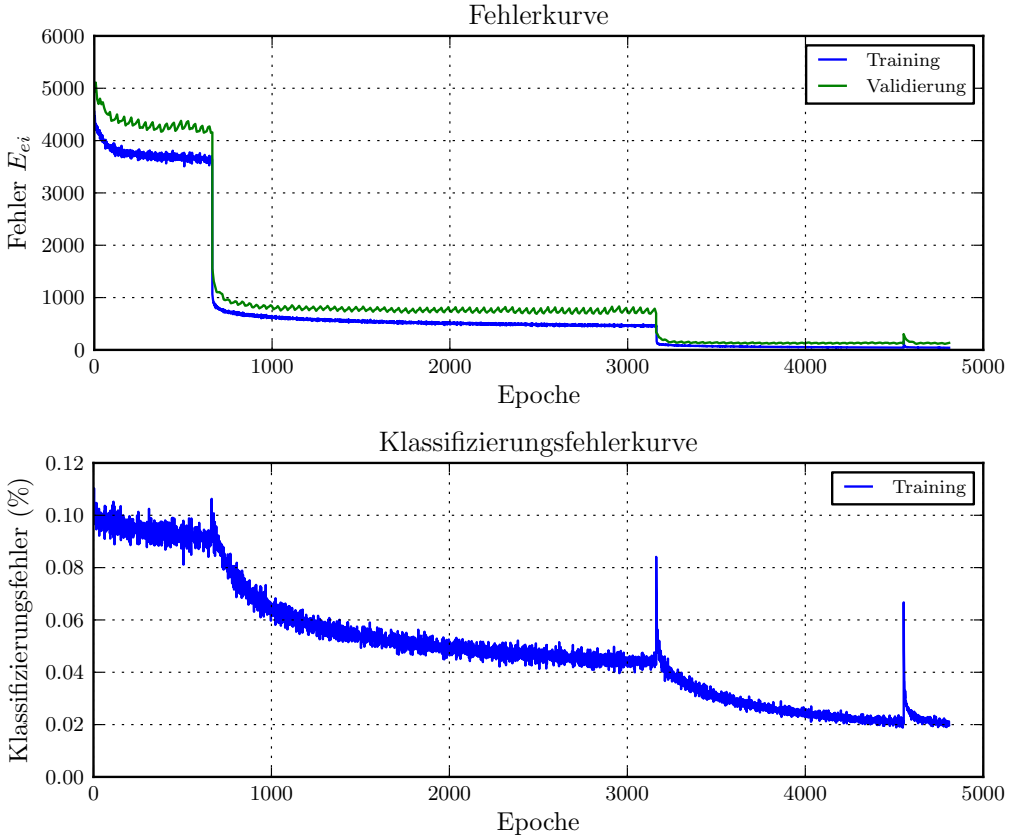
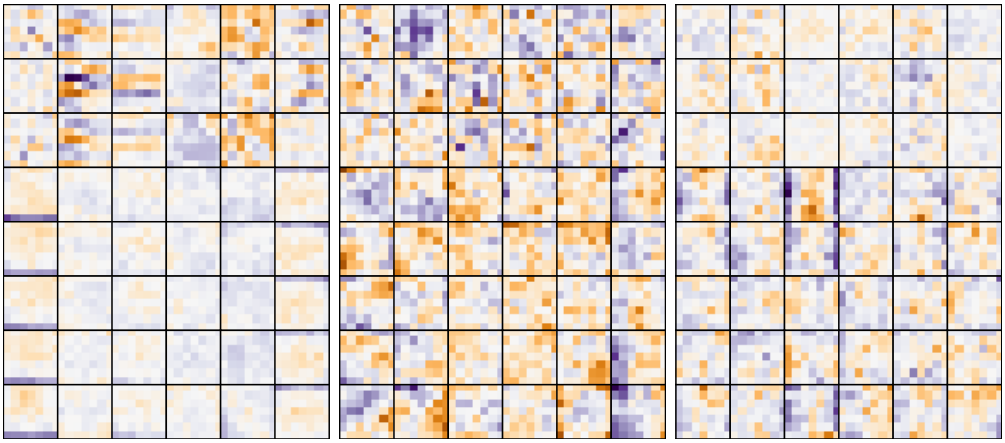
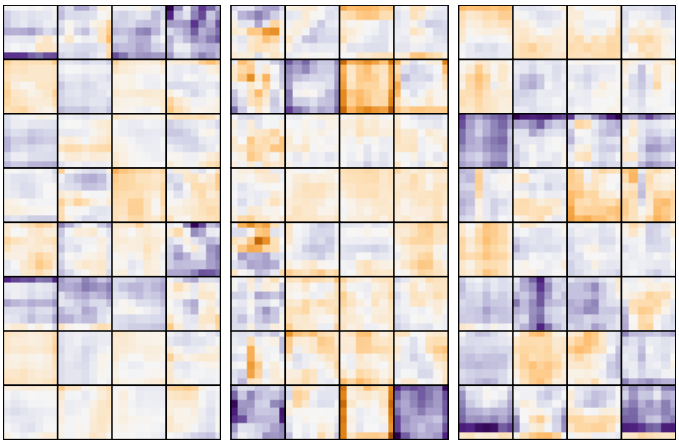


Abbildung 5.5: Verlauf des Fehlers auf Training- und Testdaten, sowie Klassifizierungsfehler auf Trainingsdaten der ersten vollen Architektur auf INRIA Graz-02. Die Sprünge im Verlauf entstehen durch den Wechsel der Lernphase.



(a) Feine Skala **(b)** Mittlere Skala **(c)** Grobe Skala

Abbildung 5.6: Filter auf den Eingabe auf allen drei Skalen. Negative Werte werden Blau, Positive Rot und 0 mit Weiß dargestellt.



(a) Erste Schicht **(b)** Zweite Schicht **(c)** Dritte Schicht

Abbildung 5.7: Filter zur Ausgabe der ersten drei Schichten. Negative Werte werden Blau, Positive Rot und 0 mit Weiß dargestellt.

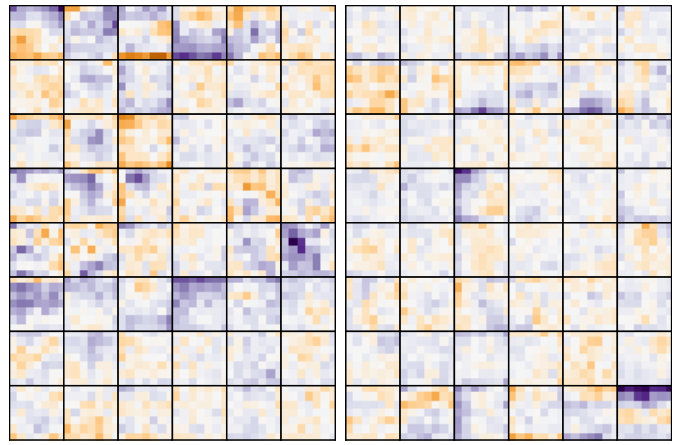
sonders interessant sind die Filter auf den Ausgaben. Die ersten beiden dieser Filter $W_{01,1}$ (c), $W_{12,1}$ (d) sind die jeweiligen Filter auf den Ausgaben einer Schicht, dessen Resultate der Ausgabe der nächsten Schicht hinzuaddiert werden. Der letzte Filter W_3 (e) ist der Paarweise Klassenfilter, ein hochauflösenderer Filter, der nochmal die letzte Ausgabeschicht glätten soll. Jeder der Ausgaben der Schichten wurde zu einer Phase des Trainings als Ausgabe des Netzwerkes genutzt, haben also prinzipiell versucht das Problem eigenständig zu lösen. Dabei zeigen die letzten beiden Filter auffällige Ähnlichkeiten. Dies wird bedeuten, dass die beiden zu grunde liegenden Ausgaben prinzipiell ebenfalls ähnliche Ergebnisse liefern. Beide Filter übernehmen insbesondere den zentralen Punkt des Eingabefensters (mittlerer Punkt der Filter auf der Diagonalen) und filtern die erste Klasse aus den Ausgabe der weiteren Klassen (Die blauen Punkte aus der ersten Zeile der Filter). Ansonsten hat auch dieser Filter eine glättende Wirkung, gut zu sehen z.B. auf den letzten drei Elementen auf der Diagonalen des Paarweisen Klassenfilters, bei denen ein kleiner Radius um den zentralen Punkt ebenso positiv ist, nach außen hin aber abnehmen. Hingegen dazu zeigt der erste dieser Filter ein anderes Verhalten. Dieser übernimmt nicht Ausgabe in einer geglätteten Form wie in der Diagonalen zu sehen wäre. Dies kann bedeuten, dass die Ausgabe dieser Schicht nicht in der Lage war das Problem genügend zu lösen, um sinnvoll direkt weiterverwendet zu werden. Dies ist nicht verwunderlich, da der ersten Schicht weniger Informationen zur Verfügung steht.

Auf dem Testset erreicht dieses Netzwerk mit einer Präzision von 75.2%, 74.2% und 62.7% auf den Klassen Fahrrad, Auto, Personen ein vergleichbares Ergebnis zur originalen Arbeit.

5.3.3 Diskussion

Die erreichten Ergebnisse zeigen zunächst nur ein unwesentliche Verbesserung gegenüber der originalen Arbeit von Schulz und Behnke (2012b). Dieser Umstand kann darauf zurückzuführen sein, dass keine Ignoriermaske und kein softmax auf den Ausgaben zur Weiterverwendung angewendet wird.

Viele Filtermasken wiesen Rauschen auf, die Masken selbst waren also nicht sehr glatt. Glatte Filter wären jedoch wünschenswert.



(a) $W_{01,0}$ **(b)** $W_{12,0}$

(c) $W_{01,1}$ **(d)** $W_{12,1}$ **(e)** Paarweiser Klassenfilter W_3

Abbildung 5.8: Auschnitte der Filter zwischen den ersten beiden Schichten und der vollständige Paarweise Klassenfilter. Negative Werte werden Blau, Positive Rot und 0 mit Weiß dargestellt.

Der Filter auf der ersten Ausgabeschicht zeigte deutlich Unterschiede zu Filtern späterer Ausgaben, was darauf hindeuten konnte, dass die erste und damit noch sehr einfache Schicht nicht in der Lage war das Problem genügend zu lösen. Möglicherweise ist das einzelne Lernen dieser Schicht nicht nötig.

5.4 NYU Depth v2

In diesem Abschnitt werden die Experimente auf dem NYU Depth v2 Datensatz dokumentiert.

Der Datensatz wird einmal zufällig in 796 Trainingsbilder, 73 Validierungsbilder und 580 Testbilder aufgeteilt.

Im Gegensatz zu INRIA Graz-02 wird die Fehlerfunktion E_{ml} verwendet, da die Evaluation auf den Testdaten die Genauigkeit pro Pixel ermittelt, wir also genauso diese maximieren wollen. Aus diesem Grund nutzen wir für den Early Stopper ebenfalls den Klassifikationsfehler auf dem Validierungsset.

5.4.1 Tiefeninformationen

Als erstes wird ein Experiment ohne und mit Verwendung der Tiefeninformationen betrachtet. Den Netzwerken wurden maximal 2500 Epochen pro Lernphase gegeben. Verwendet wurde Rmsprop mit einer Lernrate von 0.0001. Der Early Stopper bekommt zunächst eine Gewährungszeit von 256 Epochen. Es wird eine Besserung von 0.5% verlangt. Bei Erfolg, wird die, zu dem Zeitpunkte erreichte, Epochenzahl multipliziert mit 1.75 zur neuen Gewährungszeit. Leider hat ein Fehler in der Implementation des Early Stoppers dafür gesorgt, dass langsame, aber insgesamt genügend starke Besserungen nicht als Erfolg anerkannt worden sind, wodurch die folgenden Netze kürzer gelernt haben.

Abbildung 5.9 zeigt den Verlauf des Fehlers und des Klassifizierungsfehlers auf Training- und Validierungsdaten des Netzwerks ohne den Tiefeninformationen. Die Trainingsphasen dauerten jeweils 1320, 912, 688, und 256 Epochen.

In Abbildung 5.10 ist der Verlauf von Fehler und Klassfikationsfehler des Netzwerk mit den Tiefeninformationen zu sehen. Die Trainingsphasen dauerten jeweils 2128, 576, 408 und 272 Epochen. Wie schon auf INRIA Graz-02

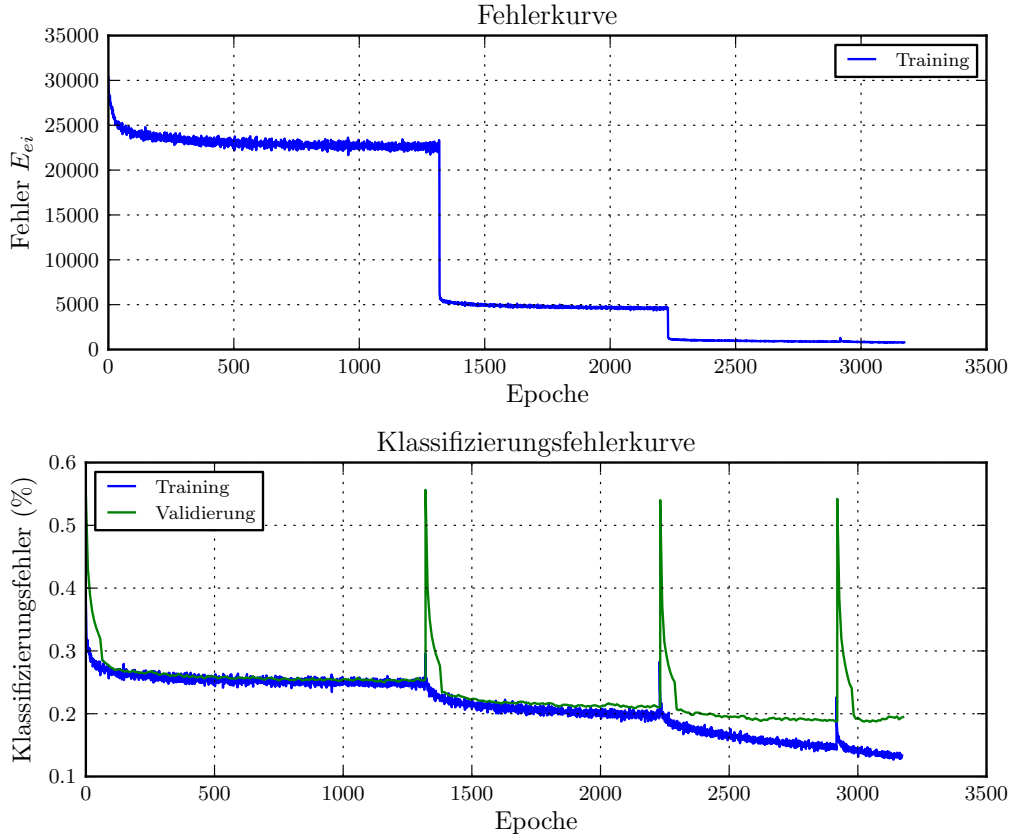


Abbildung 5.9: Verlauf der Fehlerfunktion und Klassifikationsfehler vom ersten Experiment *ohne* Tiefeninformationen auf NYU Depth v2.

zeigt sich kein anhaltend steigender Fehler auf dem Validierungssatz. Mit den Tiefeninformationen wurde der Fehler, und auch der Klassifikationsfehler merklich geringer. In den späteren Phasen ist der Abstand zwischen dem Fehler auf den Trainings- und Validierungsdaten niedriger.

Wie schon bei den ersten Experimenten auf dem INRIA Graz-02 Datensatz, scheinen die Filter auf den gröberen Skalen die zunehmende Standardabweichung der ZCA Karten auszugleichen wie in Abbildung 5.11 zu sehen. Leider sehen die Filter der beiden gröberen Skalen verrauscht aus. Dies kann vermutlich dem frühzeitigen Stop des Netzwerkes angerechnet werden.

Interessant zu beurteilen sind die Filtermasken auf den hinzugekommenen Gradienten der Tiefenwerte, welche in Abbildung 5.12 dargestellt werden (die jeweiligen letzten fünf Zeilen). Die Filter auf der ersten Skala zeigen

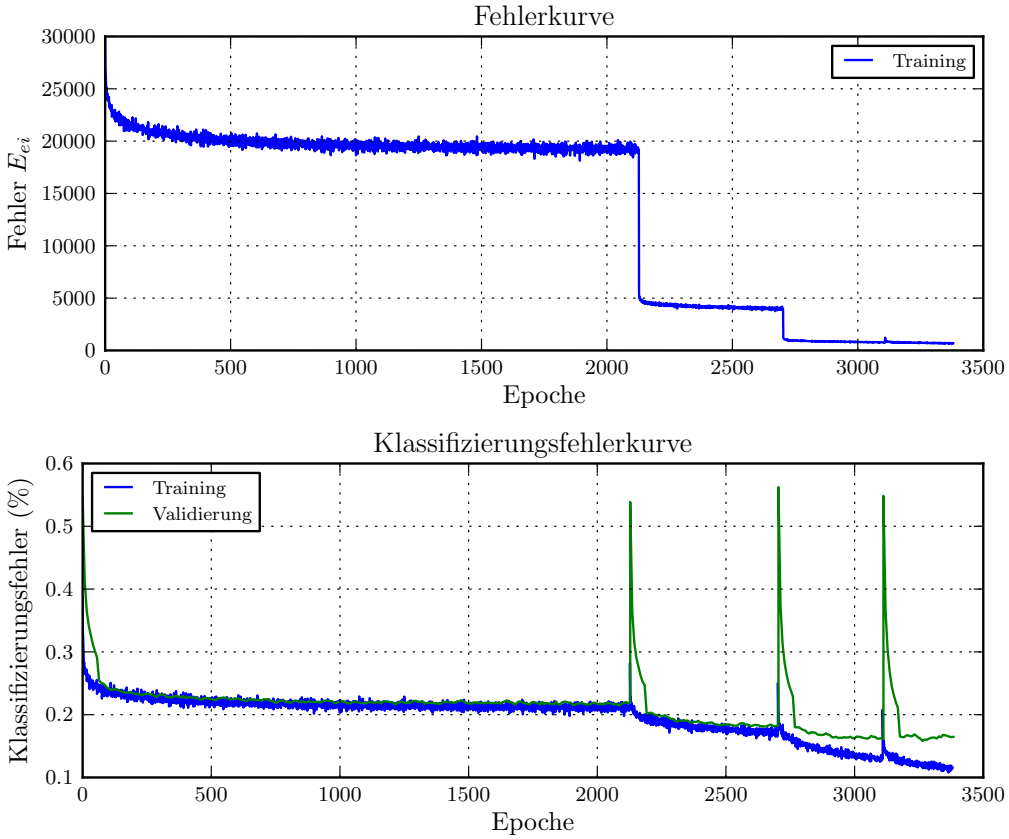
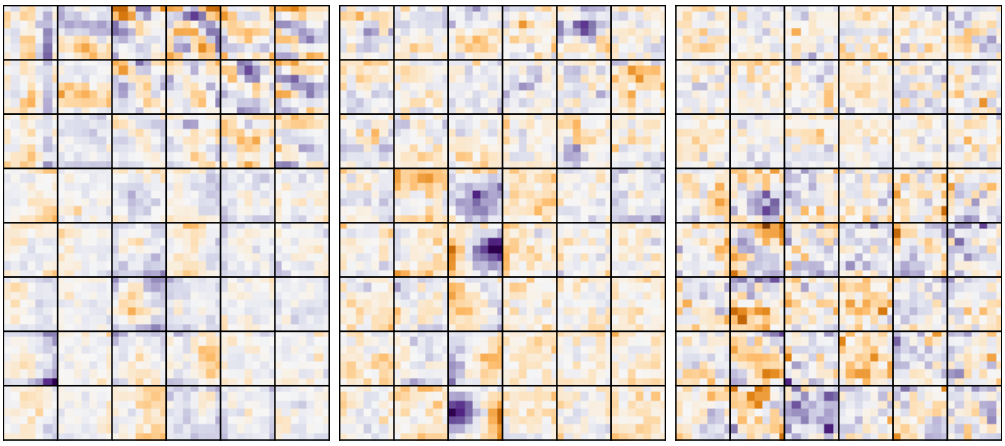


Abbildung 5.10: Verlauf der Fehlerfunktion und Klassifikationsfehler vom ersten Experiment *mit* Tiefeninformationen auf NYU Depth v2.



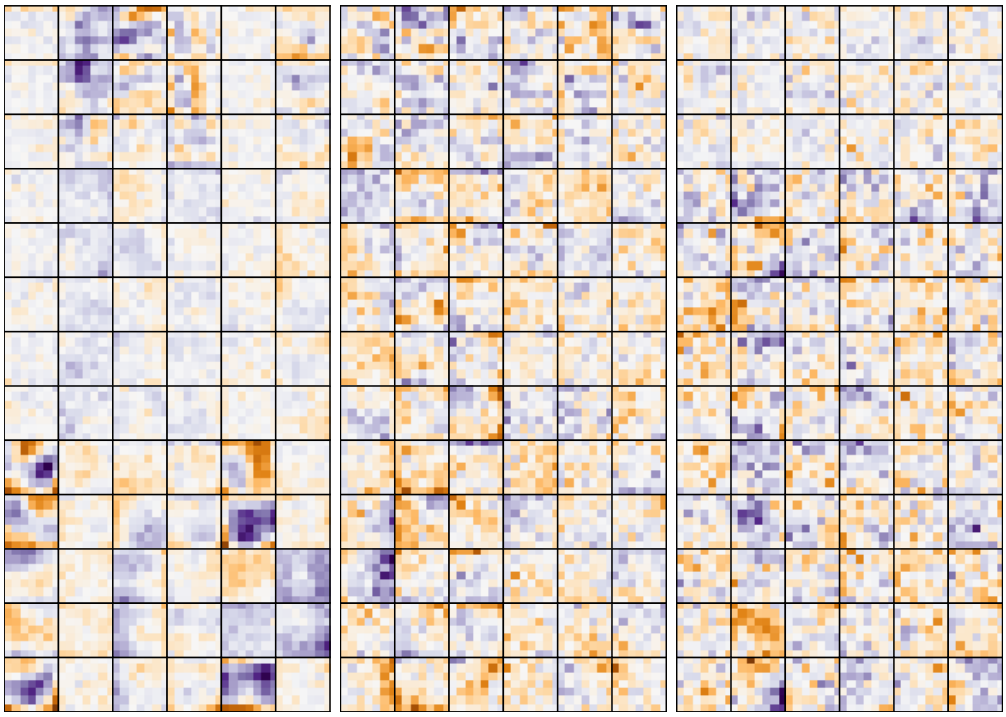
(a) Erste Schicht **(b)** Zweite Schicht **(c)** Dritte Schicht

Abbildung 5.11: Visualisierung der Filter aus dem ersten Experiment *ohne* Verwendung der Tiefeninformationen auf NYU Depth v2. Negative Werte werden Blau, positive Rot und 0 Weiß dargestellt.

relativ hohe Magnituden auf den Tiefeninformationen. Das Netz scheint viel über die Tiefeninformationen zu bestimmen. Auf den späteren Skalen zeigen die Filter Ähnlichkeiten zu den Gradienten. Das Netz macht also auch auf diesen Skalen Gebrauch von den Tiefeninformationen.

In Abbildung 5.13 sind drei Beispiele von Ausgabekarten des Netzwerkes auf Testdaten und die zugehörigen Eingabebilder dargestellt. In der ersten Zeile sind die Eingabebilder, die nächsten vier sind Ausgabe- und Teacherkarten der einzelnen Klassen und die letzte Zeile zeigt die Vorhersage des Netzwerk und Teacher in einer Karte. Zur Visualisierung wurde softmax über die Klassen angewendet. Schön zu sehen ist, dass sich für größere Flächen der ersten beiden Klassen (Boden, Struktur) visuell gute Übereinstimmungen verzeichnen lassen. Kleinere Bereiche Boden hingegen wie z.B. im zweiten Bild hat das Netzwerk nicht erkannt. Auch sonst lassen sich in den meißten Fällen zumindest höhere Aktivierungen an gewünschten Positionen erkennen. Leider sind gerade in der Ausgabekarte der vierten Klasse, also der kleineren Gegenstände häufig hohe Aktivierungen an den richtigen Positionen zu erkennen, werden aber von den anderen Klassen unterdrückt. Dadurch werden allerdings auch falsche Klassifizierungen wie im zweiten Bild in der unteren linken Ecke verhindert.

Interessant ist auch ein Vergleich zwischen dem ersten und dritten Bild.



(a) Feine Skala **(b)** Mittlere Skala **(c)** Grobe Skala

Abbildung 5.12: Visualisierung der Filter auf den Eingaben des ersten Experiments *mit* Verwendung der Tiefeninformationen auf NYU Depth v2. Negative Werte werden Blau, positive Rot und 0 Weiß dargestellt.

Im ersten Bild befindet sich ein Bücherregal, welches, samt den Büchern, als Möbel zu klassifizieren ist, während die Magazine auf dem Tisch in Bild drei als Gegenstände klassifiziert werden sollen. Dabei ist festzustellen, dass die Bücher im Regal eine leichte Aktivierung in der Ausgabe der vierten Klasse aufweisen. Im Sinne der Klassen ist diese Aktivierung eigentlich wünschenswert, da Bücher selbst semantisch der vierten Klassen zugehören.

Ansonsten ist positiv anzumerken, dass die Ausgaben sehr steile Kanten aufweisen. Es sind viele Bereiche ausfindig zu machen, in denen Blaue (niedrige) und Rote (hohe) Aktivierungen dicht aneinander liegen. Eine Eigenschaft, die möglicherweise auf die Rectified Linear Transferfunktion zurückzuführen ist. Diese Eigenschaft muss gegeben sein, wenn auf engem Raum verschiedene Klassen eindeutig zugewiesen werden sollen.

5.4.2 Höhere Eingabedimension

Da die Struktur der verwendeten Netzwerkarchitektur die Eingaben von Schicht zu Schicht halbiert, wurde die Auflösung der Ausgabe der bisherigen Experimente auf 49×49 reduziert. Dafür wird das Teachersignal entsprechend verlustbehaftet skaliert und auch die finale Ausgabe ist darauf beschränkt. Daher soll dieses Experiment eine Eingabeauflösung 252×252 verwenden, sodass die letzte Ausgabe auf 63×63 erhöht wird. Das Experiment soll maximal 1024 Epochen pro Lernphase dauern. Der Early Stopper verlangt eine Besserung um 0.5%, um die Lernzeit um einen Faktor von 1.5 zu verlängern.

Abbildung 5.14 zeigt die Lernkurven dieses Experiments. Die Kurven ähneln sehr den vorherigen Kurven. Abbildung 5.17 zeigt Ausgaben des Netzwerks. In dieser Visualisierung wurde softmax über die Klassendimension angewendet. Die höhere Auflösung der Ausgabe macht sich visuell bemerkbar, da weniger Aliasingeffekte bemerkbar und dadurch glattere Kanten zu sehen sind.

5.4.3 Normierte Eingaben

Wie in voherigen Sektionen beschrieben, zeigten die veschiedenen Merkmalskarten unterschiedliche Standardabweichungen und die HOG Karten

Abbildung 5.13: Ausgaben der letzten Schicht von NYU Depth v2 Testbildern. Zur Visualisierung wurde softmax auf die Klassendimension der Ausgabe angewendet. Dargestellt sind drei Beispiele in je einer Zeile. Die erste Zeile zeigt die jeweiligen RGB Eingabebilder. Die nächsten vier Zeilen zeigen paarweise für jede Klasse (Boden, Struktur, Mobiliar, Requisiten) die Ausgabe des Netzwerk (links) und Teacher (rechts). Die letzte Zeile zeigt die Prediktion und Teacher.

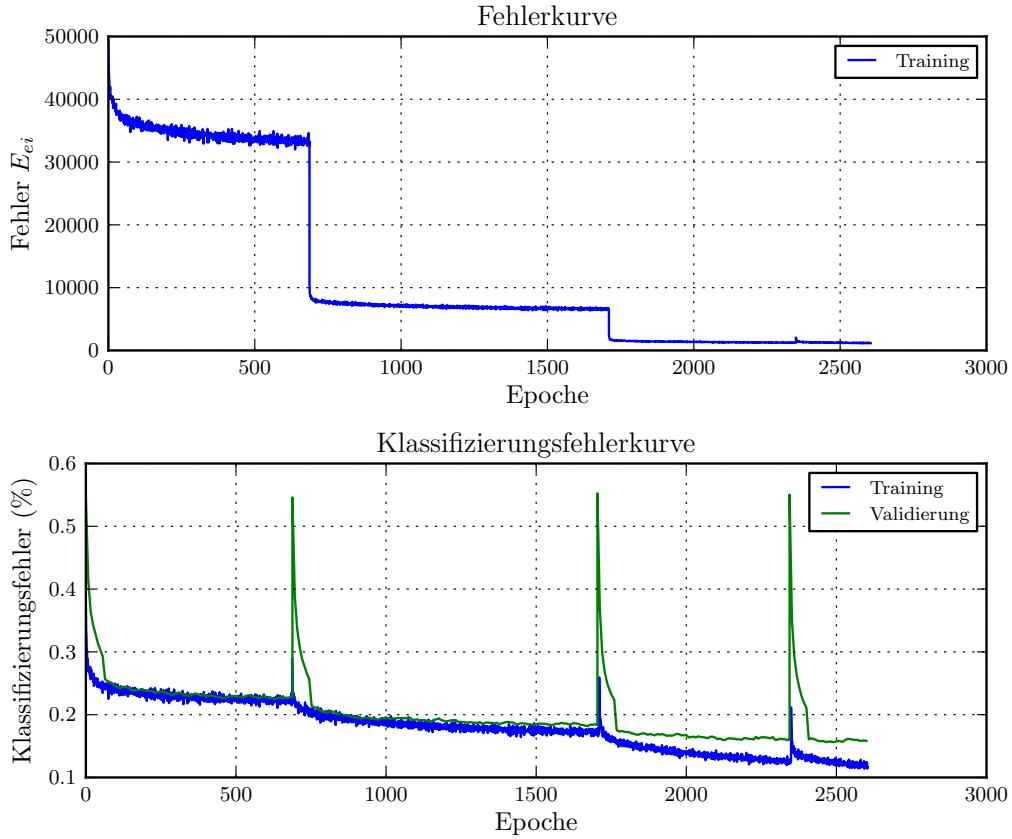
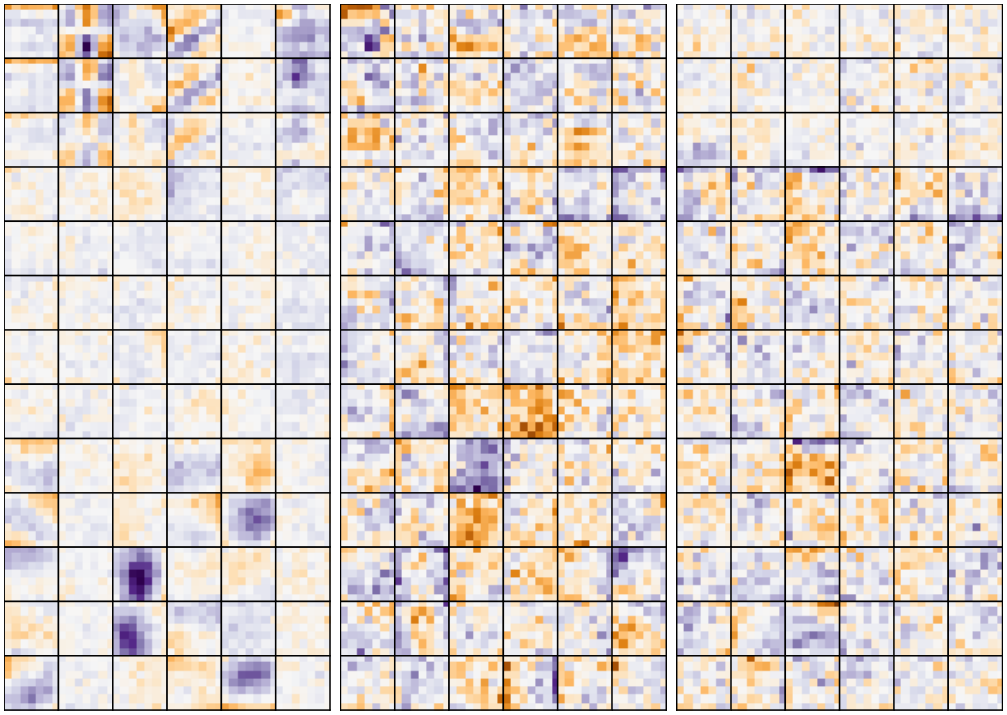
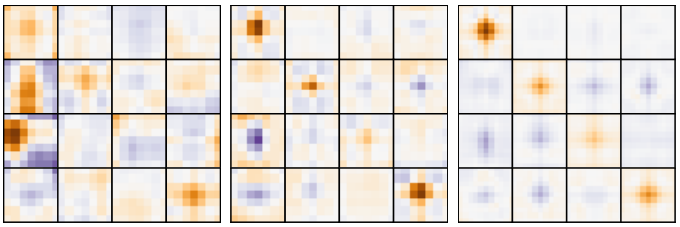


Abbildung 5.14: Verlauf der Fehlerfunktion und Klassifikationsfehler des Experiments mit einer Eingabeauflösung von 252×252 auf NYU Depth v2.



(a) $W_{0,0}$ **(b)** $W_{1,0}$ **(c)** $W_{2,0}$

Abbildung 5.15: Ausschnitt von Filtern auf den Eingaben, eines Netzwerks mit höherer Eingabeauflösung, gelernt auf NYU Depth v2.



(a) $W_{01,1}$ **(b)** $W_{12,1}$ **(c)** W_3

Abbildung 5.16: Filter wiederverwendeter Ausgabe und Paarweiser Klassenfilter des Experiments höhere Eingabeauflösung auf NYU Depth v2.

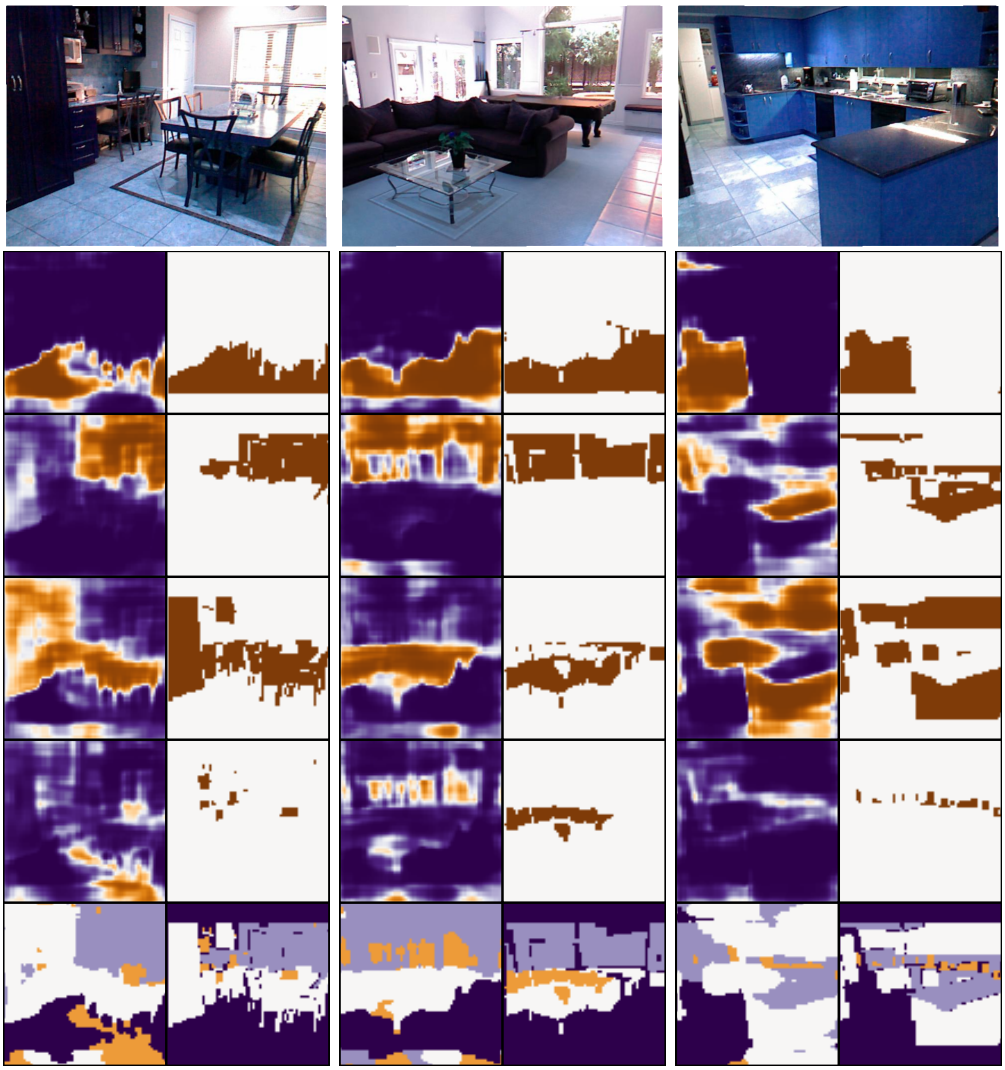


Abbildung 5.17: Ausgaben der letzten Schicht von NYU Depth v2 Testbildern des Experiments mit höherer Ausgabeauflösung. Zur Visualisierung wurde softmax auf die Klassendimension der Ausgabe angewendet. Dargestellt sind drei Beispiele in je einer Zeile. Die erste Zeile zeigt die jeweiligen RGB Eingabebilder. Die nächsten vier Zeilen zeigen paarweise für jede Klasse (Boden, Struktur, Mobiliar, Requisiten) die Ausgabe des Netzwerk (links) und Teacher (rechts). Die letzte Zeile zeigt die Prediktion und Teacher.

einen von Null verschiedenen Mittelwert. Für dieses Experiment wurden der Mittelwert der HOG Karten und Standardabweichungen der gröberen ZCA Karten empirisch ermittelt, um die jeweiligen Merkmale, auch über die verschiedenen Skalen, auf eine einheitliche Standardabweichung und Mittelwert zur normieren.

Für dieses Experiment werden maximal 2000 Epochen angesetzt. Als Lernverfahren wird wieder Rmsprop mit einer Lernrate von 0.0001 angesetzt. Der Early Stopper bekommt zunächst eine Gewährungszeit von 128 Epochen. Es wird eine Besserung von 0.5% verlangt. Bei Erfolg, wird die, zu dem Zeitpunkte erreichte, Epochenzahl multipliziert mit 2 zur neuen Gewährungszeit. Im Gegensatz zu den voherigen Tests wird für die Fehlerfunktion nun der Mittelwert über alle Pixel wie in Gleichung (2.4) verwendet und nicht nur über die Klassen. Da Rmspros einen gleitenden Mittelwert über den Gradienten bildet, sollte sich dies nicht auf das Lernverhalten auswirken. Außerdem wurde den Gewichten an den jeweiligen Ausgabe eine um Faktor 10 verringerte Lernrate erteilt, um einen Fokus auf den Filtern auf den Eingaben zu legen.

Abbildung 5.18 zeigt die Lernkurven des Experiments. Das Netzwerke trainierte jeweils die vollen 2000 Epochen pro Lernphase und hat damit länger gelernt als die Netzwerke letzten Experimente. Im Vergleich zu Abbildungen 5.9 und 5.10 ist der Unterschied der Fehlerfunktion zu sehen. Die aufeinanderfolgenden Phasen zeigen nun ähnlichere Fehler, abgesehen vom initialen Fehlers einer Phase, die nun ebenfalls besser zu sehen sind. Dies hat den Vorteil, dass die Übergänge der Phasen einfacher zu beurteilen sind. Interessant zu sehen ist, dass der Fehler auf den Validierungsdaten in den ersten beiden Phasen mehr mit dem Trainingsset übereinstimmt und erst in den letzten beiden Phasen deutlicher abweicht. Dabei ist anzumerken, dass weiterhin kein anhaltend steigender Fehler auf dem Validierungssatz zu erkennen ist.

Abbildung 5.19 zeigt Filter auf den Ausgaben die sich von vorherigen unterscheiden. Der Paarweise Klassenfilter W_3 zeigt nun leichte horizontale oder vertikale Gewichtungen. In Spalte eins der dritte Filter von oben zeigt eine solche Filtermaske mit vertikaler Neigung. Dieser reduziert beispielsweise die Vorhersage Boden, sobald ein Möbelstück insbesondere über- oder unter dem betrachteten Punkt liegt.

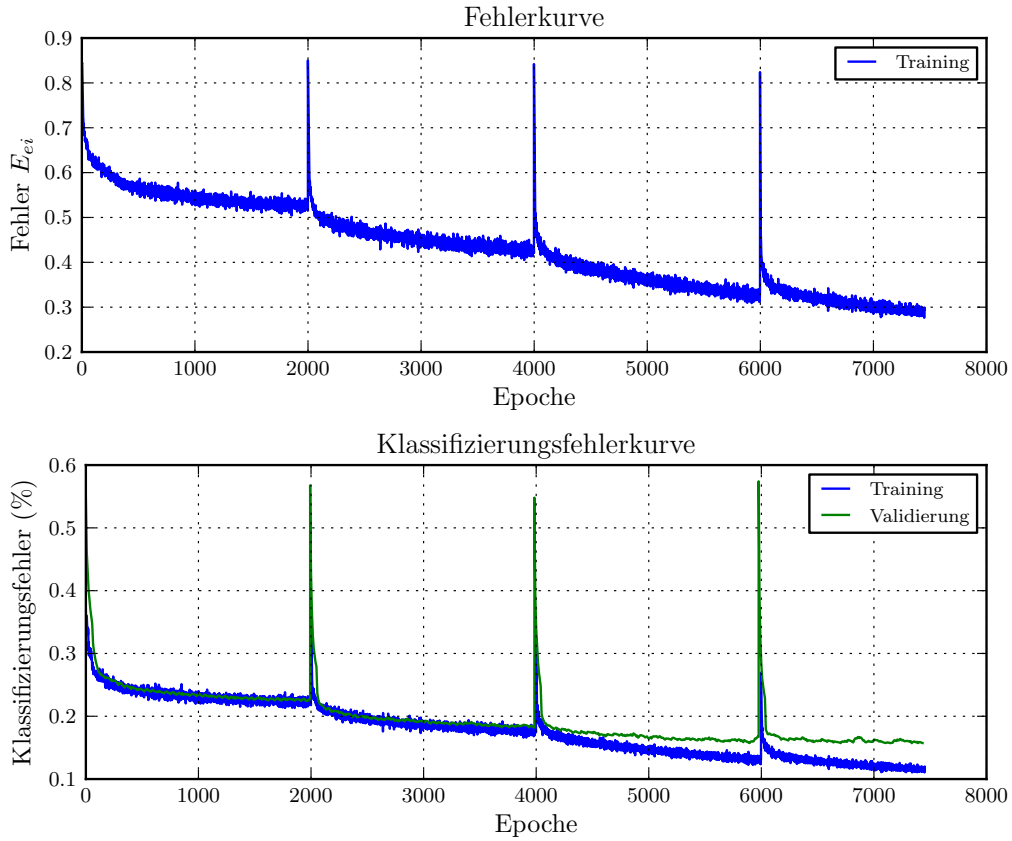
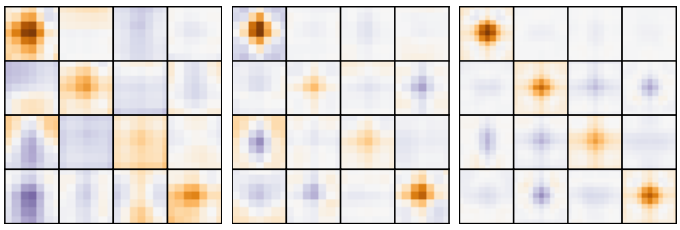


Abbildung 5.18: Verlauf der Fehlerfunktion und Klassifikationsfehler des Experiments auf NYU Depth v2 mit normierten Eingabe.



(a) $W_{01,1}$ **(b)** $W_{12,1}$ **(c)** W_3

Abbildung 5.19: Filter auf Ausgaben des Experiments auf NYU Depth v2 mit normierten Eingabe. Negative Werte Blau, positive Rot und Null mit Weiß dargestellt.

Tabelle 5.1: Klassifikationsergebnisse der verschiedenen Klassen und deren Mittel auf INRIA Graz-02

Netzwerk	Fahrrad	Auto	Person	Mittel
Vereinfachtes Netz (Sektion 5.3.1)	70.5	67.3	57.1	65.0
Vollständiges Netz (Sektion 5.3.2)	75.2	74.2	62.7	**70.7**
Schulz und Behnke (2012b)	**75.6**	**74.8**	61.1	70.5
Fulkerson u. a. (2009)	72.2	72.2	**66.3**	70.2

5.5 Klassifikationsergebnisse

5.5.1 INRIA Graz-02

In Tabelle 5.1 werden die Klassifikationsergebnisse dieser und der verwandten Arbeit der originalen Arbeit. Das Ergebnis zeigt im Mittleren Klassengenauigkeit nur einen minimalen Fortschritt von 0.2% gegenüber der originalen Arbeit. Dabei wird auf der schwächsten klassifizierten Klasse (Person mit 62.7%) ein um 1.7% besseres Ergebnis, wodurch die Genauigkeit über die verschiedenen Klassen etwas ausgeglichen wird.

Es erfordert eine Reevaluierung, unter Verwendung der Ignoriermaske und einem austrainiertem Netz, um die Leistung des Verfahrens auf diesem Datensatz korrekt einschätzen zu können.

5.5.2 NYU Depth v2

In Tabelle 5.2 werden die Ergebnisse dieser Arbeit mit anderen Verglichen. Zunächst wurde in Sektion 5.4.1 gezeigt, dass bei ähnlichen Lernparametern das Netzwerk mit Tiefeninformationen in der Pixelgenauigkeit 3.7% und im Mitel über die Klassen um 6.7% bessere Ergebnisse erzielt. Längere Trainingsphasen und eine höhere Eingabedimension verbesserten die Pixelgenauigkeit nochmals um ˜1%, leider auf Kosten der Klasse Requisiten. Ebenfalls zu beobachten ist, dass das Experiment mit der normierten Eingabe schlechtere Ergebnisse erzielt hat, was möglicherweise der verringerten Lernrate der Filter zu den Ausgaben geschuldet ist.

Leider erreichen wir damit nicht das Ergebnis von Couprie u. a. (2013); Waldvogel (2013) in der Pixel- oder Klassengenauigkeit, allerdings erzielt dieses Verfahren das beste Ergebnis auf den zwei Klassen Möbel und Requi-

Tabelle 5.2: Ergebnisse auf NYU Depth v2. Genauigkeit pro Pixel und Mittlere Genauigkeit über die Klassen.

Netzwerk	Pixel	Klassen
ohne Tiefen. (Sektion 5.4.1)	56.2	56.1
mit Tiefen. (Sektion 5.4.1)	59.9	62.8
Hohe Auflösung (Sektion 5.4.2)	61.0	61.9
Normierte Eingabe (Sektion 5.4.3)	60.5	61.4
Waldvogel (2013)	**68.1**	**65.1**
Couprie u. a. (2013) *ohne* Tiefen.	63.0	59.2
Couprie u. a. (2013) *mit* Tiefen.	64.5	63.5

Tabelle 5.3: Ergebnisse auf NYU Depth v2. Genauigkeit über die einzelnen Klassen.

Netzwerk	Boden	Struktur	Möbel	Requi.
ohne Tiefen. (Sektion 5.4.1)	69.1	57.8	55.7	41.7
mit Tiefen. (Sektion 5.4.1)	78.8	66.6	50.7	**55.0**
Hohe Auflösung (Sektion 5.4.2)	80.1	65.0	**56.6**	50.0
Normierte Eingabe (Sektion 5.4.3)	75.8	66.9	52.9	50.1
Couprie u. a. (2013) *ohne* Tiefen.	68.1	**87.8**	51.1	29.9
Couprie u. a. (2013) *mit* Tiefen.	**87.3**	86.1	45.3	35.5

siten. Für Requisiten werden 14.4% bis 19.6% höhere Genauigkeit gegenüber erreicht. Die ist insbesondere interessant, da diese Klasse aus kleineren Objekten besteht, die besonders durch die geringe Auflösung der Ausgabe leiden müsste. Es wäre interessant zu erfahren, ob die schlechtere Erkennung von kleineren Objekten von Couprie u. a. (2013) an der Anwendung von einem Netz auf alle Skalen liegt und gegen einheitliche Filter pro Skala sprechen würde.

5.6 Geschwindigkeit

Zur Evaluation der Geschwindigkeit wurde die Methode auf einer NVIDIA GeForce GTX Titan, mit 2688 CUDA Kernen, 837 MHz Basis und 876 MHz Boost-Taktung, getestet. Die Vorverarbeitung der Eingaben wird paralleli-

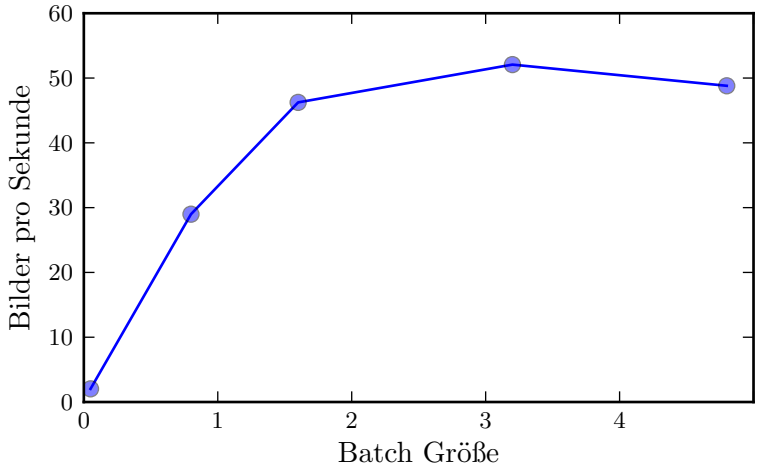


Abbildung 5.20: Illustriert die Relation des mittleren Durchsatz von Bildern pro Sekunde der Vorwärtspropagierung zur verwendeten Batchgröße auf NYU Depth v2. Der Test wurde für Training und reine Vorhersage durchgeführt.

siert auf einem Intel Xeon mit 2.6 GHz durchgeführt. Verarbeitete Eingabe werden einer Queue hinzugefügt, aus welcher das Netzwerk die Eingaben einließt. Dabei werden 24 Threads verwendet, die jeweils ein Bild vorverarbeiten.

In Abbildung 5.20 ist Durchsatz der Vorwärtspropagierung in Bildern pro Sekunde in Abhängigkeit der Batchgröße auf NYU Depth v2, für ein reine Vorhersage, illustriert. In Tabelle 5.4 ist festzustellen, dass die Vorwärtspropagierung beim Training erst bei einer Batchgröße von mindestens 32 erreicht wird. Dies bedeutet, dass eine Batchgröße von 32 zum trainieren verwendetet kann, um eine bessere Approximation des Gradienten zu erhalten, ohne Geschwindigkeit einzubüßen. Im Falle der reinen Vorhersage wird die Sättigung früher erreicht. Dies liegt daran, dass, wie in Tabelle 5.5 beschrieben zu sehen, die Rückwärtspropagierung natürlich ebenfalls Zeit einnimmt. Diese Zeit steht der Vorverarbeitung zusätzlich zur Verfügung um Eingaben vorzubereiten, bevor sie auf die GPU geladen werden müssen. Mit einer Batchgröße von 64 ist es möglich Vorhersagen für 52 Bilder die Sekunde zu bestimmen.

Tabelle 5.4: Durchschnittliche Zeitbeanspruchung beim Training auf NYU Depth v2 der Vorwärtspropagierung (Fprop), Rückwärtspropagierung (Bprop), dem Laden von Bildern auf die GPU (Laden) und der Vorverarbeitung der Eingaben eines Threads, in Abhängigkeit der Batchgröße. Alle Angaben in Sekunden.

Batchgröße	Fprop	Bprop	Laden	Vorverarbeitung
1	0.497	0.825	0.002	1.244
16	0.524	0.900	0.040	0.653
32	0.521	0.833	0.092	0.573
64	0.649	1.458	0.155	0.568
128	0.898	2.743	0.368	0.485

Tabelle 5.5: Durchschnittliche Zeitbeanspruchung reiner Vorhersage auf NYU Depth v2 der Vorwärtspropagierung (Fprop), dem Laden von Bildern auf die GPU (Laden) und der Vorverarbeitung der Eingaben eines Threads, in Abhängigkeit der Batchgröße. Alle Angaben in Sekunden.

Batchgröße	Fprop	Laden	Vorverarbeitung
1	0.490	0.002	1.291
16	0.511	0.041	0.835
32	0.495	0.197	0.624
64	0.589	0.640	0.565
96	1.595	0.372	0.669

6 Diskussion

In dieser Arbeit wurde ein Verfahren von Schulz und Behnke (2012b) vorgestellt und Neuerungen eingebracht. Das originale Verfahren wurde um eine online Vorverarbeitung, Rectifier als Nichtlineare Transferfunktion und Rmsprop als Gradientenabstiegsverfahren erweitert. Um sinnvollen Gebrauch von Datensätzen mit Tiefeninformationen zu machen, wurde ein vereinfachtes Histogramm Orientierter Tiefen für Tiefeninformationen genutzt. Dieses ist analog zu den vereinfachten HOG Merkmalen auf den RGB Eingaben.

Diese Methode wurde einer vereinfachten Netzstruktur, vergleichbar mit einem Konvolutionalen Neuronalen Netz von LeCun u. a. (1998), gegenübergestellt und eine Verbesserung von 4.7% vermerkt. Damit wurde verifiziert, dass die Erweiterung der Netzstrukturerweiterung prinzipiell förderlich ist.

Leider zeigten diese Erweiterungen auf INRIA Graz-02 kaum bessere Ergebnisse im Vergleich zur originalen Arbeit, mit einer geringfügigen Besserung von 0.2% über den Durchschnitt der Klassen. Allerdings wurde die Klasse mit der niedrigsten Genauigkeit der voherigen Arbeit um 1.6% besser klassifiziert, wodurch die Genauigkeit über die Klassen etwas ausgeglichen wird. Dabei wurde bei diesem Test keine Ignoriermaske eingesetzt, was vermuten lässt, dass sich die Ergebnisse auf diesem Datensatz noch bessern lassen.

Daraufhin wurde das Verfahren auf dem NYU Depth v2 Datensatz ohne Tiefeninformationen und unter Berücksichtung der Tiefendaten mittels HOD evaluiert. Mit Tiefeninformationen wurde eine 3.7% höhere Pixelgenauigkeit und 6.7% höhere Klassengenauigkeit erreicht und damit gezeigt, dass Tiefeninformationen erfolgreich genutzt werden können.

In der Pixel- und Klassengenauigkeit blieben die Ergebnisse 7% und 3.1% respektiv unter der Referenz von Waldvogel (2013). Im Vergleich zu einem ähnlichen Neuronalen Ansatz vom Couprie u. a. (2013) zeigte dieses Verfahren ein Ergebnis, indem die Klasse Möbel und Requisiten um 5.5% und 14.5% besseres Ergebnis erreicht.

Außerdem wurde für NYU Depth v2 gezeigt, dass die Implementierung auf GPU erlaubt Vorhersagen mit 52 Bildern pro Sekunde durchzuführen.

Diese Arbeit zeigte somit Potential der verwendeten Netzarchitektur und deckte mögliche Verbesserungsmöglichkeiten auf.

6.1 Ausblick

Es gibt eine Reihe von möglichen Erweiterungen zu den verwendeten Methoden.

Die experimentellen Ergebnisse zeigten nicht ausgelernte Filter. Daher wäre es erstrebenswert zu ermitteln, welche Ergebnisse erreicht werden, wenn als Abbruchkriterium nur der Early Stopper zu tragen kommt und somit erst bei erreichen eines längerfristig unverbessertem Validierungsfehlers gestoppt wird.

Interessant für die wiederverwendeten Ausgaben ist die Anwendung von softmax entsprechend der originalen Arbeit (Schulz und Behnke, 2012b). In dieser Arbeit wird die Ausgabe vor der Nichtlinearität weiterverwendet, obwohl diese nicht entsprechend trainiert wurde, da die Fehlerfunktionen über eine Nichtlinearität verfügen. Abgesehen vom Max Pooling folgen daher lediglich zwei Konvolutionen aufeinander, welche in ihrer Wirkung einem Filter mit größerem rezeptivem Feld entsprechen. Die Folgen dessen sind nicht klar und sollten evaluiert werden, allerdings ist anzunehmen, dass die fehlende Nichtlinearität negative Auswirkungen auf die Klassifikation haben.

In dieser Arbeit wurde gezeigt, dass Histogramme Orienterter Tiefen bessere Ergebnisse erzielen als die Tiefeninformationen wegfallen zu lassen, aber nicht verifiziert, ob dieses Verfahren gegenüber simplen Methoden ebenfalls bessere Ergebnisse liefert. Beispielsweise könnte lediglich der Gradient über die Tiefen getestet werden.

Für Eingaben mit Tiefeninformationen besteht noch die Möglichkeit einer weiteren Transformation. Mittels der Tiefeninformationen können die Eingabebilder im dreidimensionalen Raum rotiert werden. Dafür müssten nach der Rotation vermutlich Lücken gefüllt werden, möglicherweise vergleichbar mit der Aufbereitung der Tiefeninformationen der Kinect Tiefeninformationen. Dabei müsste dann beachtet werden, dass diese Transformation die online Vorverarbeitung teurer macht, weshalb diese Transformation eventuell offline angewendet werden müsste.

Ein Test zeigte positive Ergebnisse bei Verwendung einer größeren Eingabeauflösung. Insbesondere die Auflösung der letzten Schicht könnte von einer deutlich höheren Auflösung profitieren. Dabei besteht z. B. die Möglichkeit die Eingabe in multiple Fenster zu unterteilen, einzeln als Eingabe für das Netzwerk zu nutzen und die vollständige Vorhersage aus den verschiedenen Ausgaben zu rekonstruieren.

Die Augmentation der Trainingsdaten kann für Vorhersage auf den Testdatensatz übernommen werden. Dafür würden die Testdaten mit festgelegten Kombinationen von Transformationen durchgeführt und jeweils eine Vorhersage generiert werden. Auf den Vorhersagen würde die jeweilige Inverse Transformation angewendet werden, um eine gültige Korrespondenz zwischen den jeweiligen Ergebnissen zu erhalten. Über die jeweiligen Vorhersagen ließe sich dann eine, möglicherweise robustere, Vorhersage gewinnen lassen.

Literatur

Couprie, C., C. Farabet, L. Najman und Y. LeCun (2013). „Indoor Semantic Segmentation using depth information“. In: *Computing Research Repository (CORR)* abs/1301.3572 (siehe S. 19, 20, 47, 48, 51).

Dalal, N. und B. Triggs (Juni 2005). „Histograms of oriented gradients for human detection“. In: *Computer Vision and Pattern Recognition (CVPR)*. Bd. 1 (siehe S. 20).

Farabet, C., C. Couprie, L. Najman und Y. LeCun (2013). „Learning Hierarchical Features for Scene Labeling“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* PP.99 (siehe S. 6, 19).

Fulkerson, B., A. Vedaldi und S. Soatto (2009). „Class Segmentation and Object Localization with Superpixel Neighborhoods“. In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE (siehe S. 19, 47).

Glorot, X., A. Bordes und Y. Bengio (2011). „Deep Sparse Rectifier Neural Networks“. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*. Hrsg. von G. J. Gordon und D. B. Dunson. Bd. 15. Journal of Machine Learning Research - Workshop und Conference Proceedings (siehe S. 21).

Grangier, D., L. Bottou und R. Collobert (2009). „Deep Convolutional Networks for Scene Parsing“. In: *ICML 2009 Deep Learning Workshop*. Bd. 114. Citeseer (siehe S. 19).

Krizhevsky, A., I. Sutskever und G. E. Hinton (2012). „ImageNet Classification with Deep Convolutional Neural Networks.“ In: *NIPS*. Hrsg. von P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou und K. Q. Weinberger (siehe S. 21).

Kumar, M. P. und D. Koller (Juni 2010). „Efficiently Selecting Regions for Scene Understanding“. In: (Siehe S. 19).

Ladický, L., C. Russell, P. Kohli und P. H. Torr (Sep. 2009). „Associative Hierarchical CRFs for Object Class Image Segmentation“. In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE (siehe S. 19).

LeCun, Y., L. Bottou, Y. Bengio und P. Haffner (1998). „Gradient-based learning applied to document recognition“. In: Bd. 86. 11. IEEE (siehe S. 3, 51).

Lempitsky, V., A. Vedaldi und A. Zisserman (2011). „Pylon Model for Semantic Segmentation“. In: Hrsg. von J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira und K. Weinberger (siehe S. 19).

Liu, C., J. Yuen und A. Torralba (2011). „Nonparametric Scene Parsing via Label Transfer“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 33.12 (siehe S. 19).

Marszałek, M. und C. Schmid (2007). „Accurate Object Localization with Shape Masks“. In: *Computer Vision and Pattern Recognition (CVPR)*. IEEE (siehe S. 24).

Nowozin, S., P. V. Gehler und C. H. Lampert (2010). „On Parameter Learning in CRF-Based Approaches to Object Class Image Segmentation“. In: *Computer Vision ECCV 2010*. Hrsg. von K. Daniilidis, P. Maragos und N. Paragios. Bd. 6316. Lecture Notes in Computer Science. Springer Berlin Heidelberg (siehe S. 19).

Plath, N., M. Toussaint und S. Nakajima (2009). „Multi-class image segmentation using conditional random fields and global classification“. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. New York, NY, USA: ACM (siehe S. 19).

Schroff, F., A. Criminisi und A. Zisserman (2008). „Object Class Segmentation using Random Forests“. In: *British Machine Vision Conference 2008* (siehe S. 19).

Schulz, H. und S. Behnke (2012a). „Deep Learning“. In: *KI - Künstliche Intelligenz* 26 (4) (siehe S. 1, 15).

— (2012b). „Learning object-class segmentation with convolutional neural networks“. In: *Proceedings of the 11th European Symposium on Artificial Neural Networks (ESANN)* (siehe S. 1, 21, 33, 47, 51, 52).

Schulz, H., A. Müller und S. Behnke (2011). „Exploiting local structure in Boltzmann machines“. In: *Neurocomputing* 74.9 (siehe S. 7).

Silberman, N., D. Hoiem, P. Kohli und R. Fergus (2012). „Indoor Segmentation and Support Inference from RGBD Images“. In: Hrsg. von A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato und C. Schmid. Bd. 7576. Lecture Notes in Computer Science. Springer Berlin Heidelberg (siehe S. 24, 25).

Socher, R., C. C. Lin, A. Y. Ng und C. D. Manning (2011). „Parsing Natural Scenes and Natural Language with Recursive Neural Networks“. In: *Proceedings of the 26th International Conference on Machine Learning (ICML)* (siehe S. 19, 20).

Spinello, L. und K. Arras (Sep. 2011). „People detection in RGB-D data“. In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on* (siehe S. 20).

Tieleman, T. und G. Hinton (2012). „Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude“. In: *COURSERA: Neural Networks for Machine Learning* 4 (siehe S. 16, 21).

Tighe, J. und S. Lazebnik (2010). „Superparsing: scalable nonparametric image parsing with superpixels“. In: Berlin, Heidelberg: Springer-Verlag (siehe S. 19).

Waldvogel, B. (2013). „Accelerating Random Forests on CPUs and GPUs for Object-Class Image Segmentation“. MA. Thesis. Rheinische Friedrich-Wilhelms-Universität Bonn (siehe S. 47, 48, 51).