technische universität
dortmund

Tiefes Multi Task Learning unter Einbezug von Unsicherheit mit Anwendung in der Gesichtsanalyse

Masterarbeit

Keng Hong Chai
8. Juli 2019

Supervisors:
Prof. Dr.-Ing. Gernot A. Fink
Phillip Oberdiek, M.Sc.

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

INHALTSVERZEICHNIS

1 EINLEITUNG

Die automatische Analyse von Gesichtern ist seit langer Zeit ein aktuelles Forschungsthema innerhalb der Mustererkennung und Computer Vision. Schon im Jahr 1964 wurden die ersten Versuche für eine automatische Gesichtserkennung umgesetzt [TND10].

Gesichtsanalyse findet in zahlreichen Bereichen eine Anwendung. Sicherheitskritische Szenarien in denen eine Form der Überwachung notwendig ist, verlangen Gesichtserkennung. Genauso können Systeme, welche zur Prävention von Katastrophen und Paniken eingesetzt werden, die Mimik und Emotion von Gesichtern extrahieren, um vor Gefahren zu warnen [VCE+07, AJSHAR+17]. Mit dem heutigen technologischen Fortschritt im Alltag nimmt die Mensch-Maschinen Interaktion (Robotik, Smart Home, etc.) immer weiter an Bedeutung zu. Insbesondere bei sicherheitskritischen Anwendungen wie z.B. Zugangskontrollen, sind besonders akkurate Systeme nötig, um unberechtigten Personen keinen Zugriff zu ermöglichen. Werden die Gesichter in einer kontrollierten Umgebung aufgenommen, sind bisherige Verfahren nahezu fehlerfrei [DNRW13, LJL14]. Im Allgemeinen können keine Laborbedingungen angenommen werden und Modelle müssen innerhalb des unbeschränktem Falles robust sein. Diese sogenannten in-the-wild Szenarien beinhalten große Varianz in den Daten. Beispielsweise können die Beleuchtung, Skalierung, Position und Ausrichtung der Gesichter innerhalb verschiedener Bilder stark variieren. Abbildung 1.0.1 zeigt einige Beispiele in denen eine erhebliche Variabilität zu sehen ist. Solche Szenarien erschweren die Analyse unmittelbar, sodass im unbeschränkten Fall diese noch keinesfalls gelöst ist [LLWT14, ZLLT14b, GHH+17].

Die Gesichtsanalyse umfasst unterschiedliche Aufgaben, welche für das automatische Verstehen von Gesichtern erforderlich bzw. hilfreich sind. Einige Beispielaufgaben sind exemplarisch in Abbildung 1.0.2 abgebildet. Hierunter fallen die Detektion, das Schätzen der Pose und das Vorhersagen verschiedener Eigenschaften, wie z.B. Geschlecht und Haarfarbe. Ein weiterer wichtiger Aspekt ist die Landmarkenlokalisierung. Landmarken kennzeichnen Schlüsselpunkte innerhalb eines Gesichtes, welche mit einem semantischen Hintergrund belegt sind, wie z.B. Nase und Augen, und erfassen somit unter anderem die Mimik und Verformungen eines Gesichtes. So lassen sich bestimmte Merkmale in einem Gesicht wiederfinden. Sie stellt somit einen wichtigen Schritt für Folgeaufgaben dar. Unter anderem lassen sich diese für das Face

Alignment einsetzen, um Gesichter zu normieren, welches zu besseren Ergebnissen in der Gesichtserkennung führt [WJ18].

Anstelle ein separates Modell für jede einzelne Aufgabe zu trainieren, ist es möglich mit Multi Task Learning (MTL) ein Modell zu lernen, welches auf mehreren Aufgaben gleichzeitig trainiert wird. Dabei existieren zwei wesentliche Gründe für das Verwenden von MTL. Ein klarer Vorteil ist das Verringern der Parameterzahl und der Inferenzzeit. Der geringere Speicherverbrauch ist in ressourcenbeschränkten Umgebungen relevant, in denen mehrere Aufgabe zu lösen sind. Der zweite Vorteil folgt aus der Annahme, dass die betrachteten Aufgaben semantisch verwandt sind. In diesem Fall können zusätzliche Informationen aus den unterschiedlichen Aufgaben verwendet werden und zu besseren Ergebnissen in den jeweiligen Aufgaben führen [Rud17].

Eine große Fragestellung des MTL ist die Kombination der Aufgaben während des Lernprozesses. Im Allgemeinen werden diese heuristisch gewichtet, abhängig von der subjektiven Einschätzung der Schwierigkeit der einzelnen Aufgaben. Das Training des Modells ist jedoch stark abhängig hiervon, da eine ungeeignete Gewichtung zu Problemen während des Trainings führen kann, sodass einige Aufgaben nicht ausreichend gut gelernt werden. Diese Hyperparameter müssen in aufwendigen Experimenten bestimmt werden, welche in der Regel sehr rechenintensiv und zeitaufwändig sind. Kendall et al. stellen ein Vorgehen vor, in dem die homoskedastische Unsicherheiten zur Gewichtung der einzelnen Aufgaben adaptiv während des Optimierungsverfahrens mitgelernt werden [KGC17]. Innerhalb der Szenenanalyse wurden die semantische Segmentierung, Instanzsegmentierung, und Tiefenregression kombiniert, welches in den einzelnen Aufgaben zu verbesserten Ergebnissen führte.

Ziel der vorliegenden Arbeit ist es, unterschiedliche Aufgaben der Gesichtsanalyse in einem MTL Ansatz gemeinsam zu lösen. Statt einer üblichen heuristischen Gewichtung der Aufgaben wird ein aktuelles Verfahren von Kendall et al. verwendet, in dem die homoskedastische Unsicherheit mitgelernt wird. Nach bestem Wissen und Gewissen wurde diese Ansatz bisher noch nicht in der Gesichtsanalyse verwendet. Des Weiteren wird ein synthetischer Hilfstask generiert, welcher einer Approximation der Gesichtssegmentierung darstellt. Zusätzlich wird eine alternative Kostenfunktion, welche in der Landmarkenerkennung neu vorgestellt wurde und zu besseren Ergebnissen führen kann, in diesem Kontext evaluiert [FKA+17].

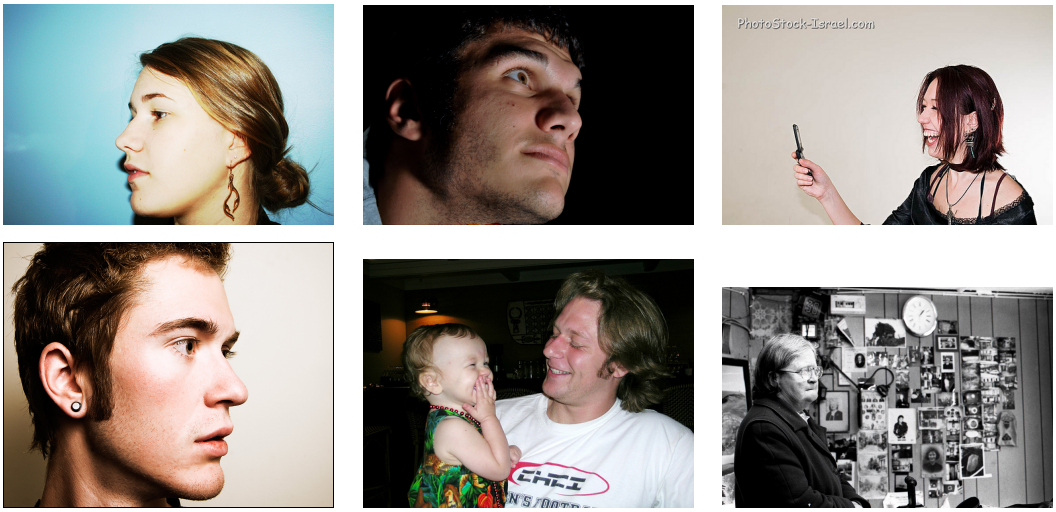


Abbildung 1.0.1: Im unbeschränktem Fall existiert starke Variabilität. Beispielweise kann die Posen, Skalierung der Gesichter, als auch Beleuchtung und Hintergrund innerhalb der Bilder starke Unterschiede vorweisen. Bilder entnommen aus [KWRB11].

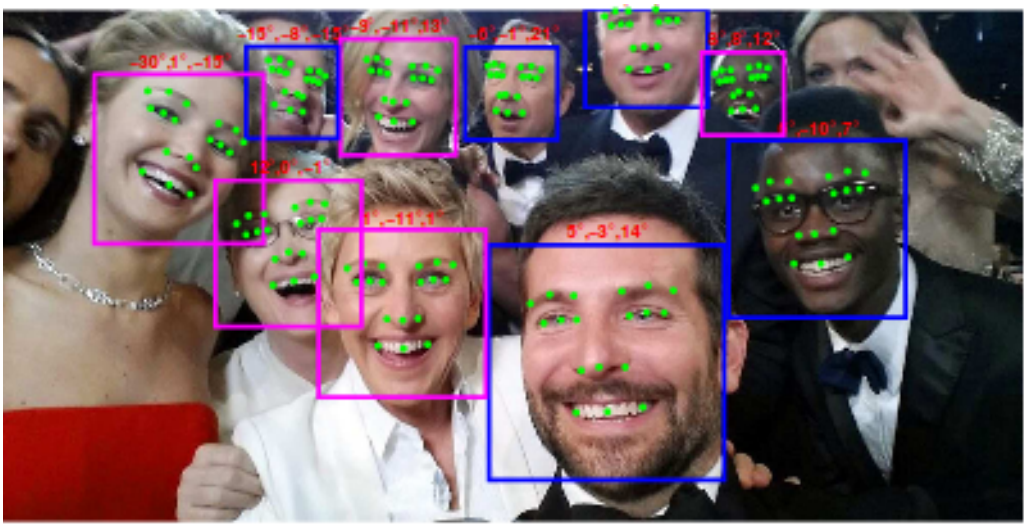


Abbildung 1.0.2: Unterschiedliche Aufgaben in der Gesichtsanalyse. In der Abbildung ist die Detektion, Landmarkenlokalisierung und Geschlechterkennung dargestellt (Farbe der Bounding Box). Zusätzlich gilt es die Pose zu schätzen, welche oberhalb der Boxen dargestellt und in Grad relativ zur Kamera angegeben ist. Abbildung entnommen aus [RPC16].

1.1 GLIEDERUNG DER ARBEIT

Die Arbeit ist wie folgt aufgebaut: Kapitel 2 behandelt die grundlegenden Verfahren, welche für das weitere Verständnis dieser Arbeit erforderlich sind. In Kapitel 3 werden relevante Arbeiten vorgestellt, welche einen größeren Bezug zu dieser Arbeit besitzen. Mit Kapitel 4 wird die verwendete Methode vorgestellt. Innerhalb Kapitel 5 werden die hierzu durchgeführten Experimente präsentiert und die Methode evaluiert. Zuletzt werden in Kapitel 6 die Ergebnisse und die daraus resultierenden Erkenntnisse dieser Arbeit zusammengefasst und Ausblicke auf zukünftige Arbeiten gegeben.

2 GRUNDLAGEN

Dieses Kapitel gibt einen groben Einblick in die grundlegenden Verfahren, welche notwendig sind für das Verständnis dieser Arbeit. Dabei sollen die wesentlichen Grundkonzepte vorgestellt werden. Für ausführliche Erläuterungen sei auf die jeweils angegebene Literatur verwiesen.

In Abschnitt 2.1 wird eine Einführung in das Thema künstliche Neuronale Netze präsentiert, welche essentiell für die Methode dieser Arbeit sind. Abschnitt 2.2 gibt einen Überblick über den Bereich des Multi Task Learnings.

2.1 KÜNSTLICHE NEURONALE NETZE

Künstliche Neuronale Netze sind Modelle, welche der grundlegenden Funktionsweise biologischer neuronaler Netze nachempfunden worden sind. Neuronale Netze bestehen aus vielen verbundenen Neuronen, welche Informationen verarbeiten und untereinander austauschen. Dabei werden die wahren Vorgänge innerhalb des menschlichen Nervensystems auf ein mathematisches Modell abgebildet. Aktuelle künstliche Neuronale Netze entsprechen jedoch nur einer starken Vereinfachung des biologischen Originals und repräsentieren nicht annähernd die realen neurologischen Prozesse (vgl. [GBC16] Seite 14-16). Bei ausreichender Modellkomplexität lassen sich mit neuronalen Netzen beliebige Funktionen approximieren, um komplexe Aufgaben zu lösen.

Obwohl die ersten Arbeiten zu künstlichen neuronalen Netzen schon sehr früh veröffentlicht wurden, waren diese in der Praxis nur von geringer Relevanz. Mit der Weiterentwicklung von GPUs und wichtigen praktischen Erweiterungen von neuronalen Netzen hat sich jedoch der Begriff Deep Learning durchgesetzt. Deep Learning beschreibt dabei das Verwenden von tiefen neuronalen Netzen. Insbesondere im Bereich der Bildverarbeitung hat sich Deep Learning zum aktuellen State-of-the-Art entwickelt.

2.1.1 *Perzeptron*

Basiselement eines neuronalen Netzes sind Neuronen. Die erste konzeptionelle Idee für ein künstliches Neuron lässt sich in das Jahr 1943 zurückverfolgen [MP43]. Neuronen,

wie sie heute in neuronalen Netzen vorkommen, lassen sich am ehesten mit dem von Rosenblatt vorgestelltem Perzeptron vergleichen [Ros58]. Dem Perzeptron unterliegt hierfür als Basis ein lineares Modell. Abbildung 2.1.1 zeigt schematisch ein Perzeptron, welches in blau dargestellt ist. Das Perzeptron berechnet eine Linearkombination von Eingangssignalen x_i [1] und Gewichten w_i und einem konstanten Bias b nach Gleichung 2.1.1:

$$h(\mathbf{x}) = \sum_i w_i x_i + b \tag{2.1.1}$$

Im Anschluss wird eine Aktivierungsfunktion φ auf $h(\mathbf{x})$ angewendet, um das Ausgangssignal des Neurons zu berechnen. Hintergrund ist hierbei, dass die Aktivierungsfunktion die Aktivierung eines Neurons auf bestimmte Eingangsreize modelliert. Durch das Verwenden von nichtlinearen Aktivierungsfunktionen kann die gewichtete Summe $h(\mathbf{x})$ transformiert werden, um unterschiedliche Aufgaben zu lösen. Im ursprünglichen Perzeptron wurde eine Schwellwertfunktion verwendet, mit der ein binärer Klassifikator realisiert werden kann (vgl. [Bis07] Kapitel 4.1.7). Als Entscheidungsregel ergibt sich:

$$f(\mathbf{x}) = \begin{cases} +1, & h(\mathbf{x}) \geqslant 0 \\ -1, & \text{sonst} \end{cases} \tag{2.1.2}$$

Die Gewichte w_i und der Bias b des Perzeptrons sind freie Parameter, welche gelernt werden können. Indem eine differenzierbare Aktivierungs- und Kostenfunktion gewählt wird, kann dafür ein gradientenbasiertes Optimierungsverfahren verwendet werden (s. Unterabschnitt 2.1.6).

2.1.2 *Multi Layer Perzeptron*

Ein einzelnes Neuron ist stark beschränkt in seiner Modellkomplexität. Für komplexere Aufgaben wie z.B. multivariate Regression oder Mehrklassenprobleme sind mehrdimensionale Ausgaben erforderlich. Hierfür werden mehrere Neuronen verwendet, welche in einer Schicht angeordnet sind, wobei keine Verbindungen zwischen Neuronen innerhalb einer Schicht existieren. Bei mehreren Neuronen lässt sich die Berechnungsvorschrift in der Matrixnotation wie folgt ausdrücken:

$$f(\mathbf{x}) = \varphi(\mathbf{W}\mathbf{x}). \tag{2.1.3}$$

1 In der Publikation von Rosenblatt waren die Eingaben x_i binär, spätere Arbeiten von [MP69] verallgemeinern das Perzeptron auf reellwertige Eingaben

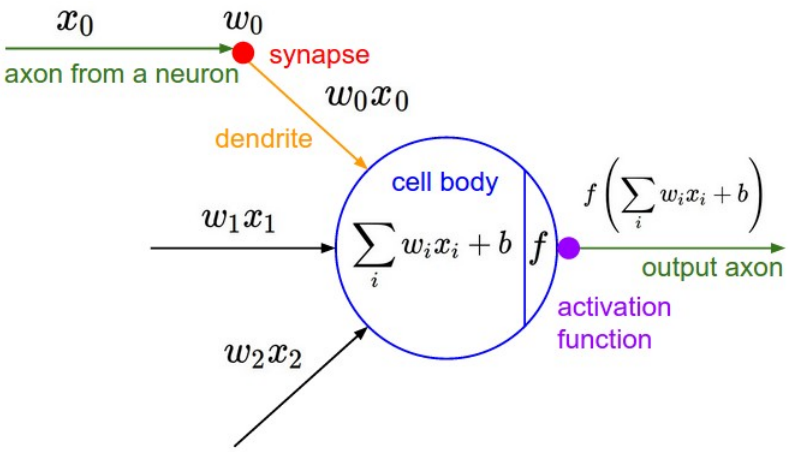


Abbildung 2.1.1: Ein künstliches Neuron (blau) berechnet eine gewichtete Summe der Eingangssignale (eingehende Kanten). Anschließend wird die Ausgabe des Neurons über eine Aktivierungsfunktion berechnet. Abbildung entnommen aus [LJY19].

Die Matrix **W** enthält die Gewichte der einzelnen Neuronen zeilenweise. Indem an den Eingabevektor **x** eine 1 konkateniert wird, können die Biase b_i der einzelnen Neuronen in einer Spalte der Gewichtsmatrix festgehalten werden. Dadurch ist die Berechnung der Ausgabe einer einzelnen Schicht auf eine einzelne Matrixmultiplikation reduzierbar.

Ein neuronales Netz, welches nur aus einer einzelnen Schicht besteht, ist jedoch nur in der Lage linear separierbare Daten zu trennen und ist somit nicht für komplexe Probleme geeignet [MP69]. Diese Einschränkung kann durch das Verwenden von mehreren aufeinanderfolgenden Schichten aufgehoben werden. Indem hierarchisch Schichten von Neuronen verbunden werden, ergibt sich ein sogenanntes Multi Layer Perzeptron (MLP). Ein 3 Schichten MLP ist beispielhaft in Abbildung 2.1.2 zu sehen. Die Ausgabe eines Neurons ist mit den Eingängen aller Neuronen einer Folgeschicht verbunden. Da keine Verbindungen innerhalb einer Schicht, als auch keine zu Vorgängerschichten existieren, nennt man diese Netzarchitektur auch Feed Forward Networks. Das Netz besteht aus einer Eingabeschicht, welche rot untermalt ist. In dieser Schicht wird der Eingabevektor **x** an die nächste Verarbeitungsschicht weitergeleitet. Als nächstes folgt eine sogenannte verborgene Schicht. Schichten zwischen Eingabe- und Ausgabeschicht werden so genannt, weil diese eine latente Repräsentation der eingehenden Daten berechnet. Zuletzt folgt die in grün unterlegte Ausgabeschicht, welche die Netzausgabe berechnet.

Die verbesserte Modellierungsfähigkeit folgt aus der Verwendung von verborgenen Schichten. Mehrere Schichten hintereinander anzuwenden erlaubt es komplexe Strukturen in den Daten zu finden. Jede Schicht berechnet eine nichtlineare Funktion (s. Gleichung 2.1.3). Dadurch entsteht eine Komposition von nichtlinearen Funktionen mit der komplexe Funktionen approximiert werden können. Sei n die Anzahl an Schichten in einem Netz und f^i die Funktion der i-ten Schicht, so ist die Netzausgabe $f(\mathbf{x})$ gegeben mit:

$$f(\mathbf{x}) = f^n(f^{n-1}(...f^2(f^1(\mathbf{x})))) \tag{2.1.4}$$

Die Eingabe wird also sequentiell über die hierarchisch angeordneten Schichten verarbeitet.

Eine wichtige Erkenntnis ist das sogenannte universelle Approximationstheorem (vgl. [GBC16] Kapitel 6.4.1). Bei geeigneter Aktivierungsfunktion und ausreichender Anzahl an Neuronen, kann ein MLP mit drei Schichten (Eingabe, Hidden, Ausgabe) jede beliebige kontinuierliche Funktion approximieren. Eine geeignete nichtlineare Aktivierungsfunktion ist z.B. die Sigmoidfunktion mit:

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \tag{2.1.5}$$

Da diese differenzierbar ist, kann ein Optimierungsverfahren auf Basis des Gradienten der Kostenfunktion verwendet werden (siehe Unterabschnitt 2.1.6). Innerhalb verborgener Schichten wird die Sigmoidfunktion heutzutage nicht mehr verwendet. Die Funktion saturiert schnell, wenn $|z|$ groß ist, womit der Gradient sehr klein wird (vgl. [GBC16] Kapitel 6.3.2) und dies problematisch für gradientenbasierte Lernverfahren ist. In tiefen Netze entsteht dadurch das Vanishing Gradient Problem (siehe Unterabschnitt 2.1.6).

Eine Sigmoidaktivierung wird heutzutage eher in der Ausgabemodellierung von neuronalen Netzen verwendet, um eine binäre Zufallsvariable Y zu modellieren. Es gilt $\sigma(z) \in (0, 1)$, womit die Neuronenausgabe als Bernoulli-verteilte Zufallsvariable interpretiert werden kann (vgl. [GBC16] Kapitel 6.2.2.2). Die Netzausgabe wird als Wahrscheinlichkeit für die Ausprägung der Zufallsvariable mit $P(y = 1) = \sigma(z)$ bzw. $P(y = 0) = 1 - \sigma(z)$ betrachtet.

In verborgenen Schichten wird aktuell in der Regel die Rectified Linear Unit (ReLU) Aktivierung verwendet. Die ReLU Funktion ist definiert als

$$\varphi(z) = \max(0, z), \quad \varphi(z)' = \begin{cases} 1, & z > 0 \\ 0, & \text{sonst} \end{cases}, \tag{2.1.6}$$

wobei $\varphi(z)'$ der Ableitung entspricht. Im positiven Bereich verhält sich die Funktion wie eine lineare Aktivierung. z wird unverändert weitergeleitet und das Neuron ist aktiv. Ist $z \leqslant 0$, wird auf 0 abgebildet und das Neuron ist somit nicht aktiv. ReLU hat gegenüber einer Sigmoidaktivierung einen entscheidenden Vorteil. Die Ableitung der ReLU ist 1, wenn das Neuron aktiv ist und 0 andernfalls. Sie saturiert nicht, wenn das Neuron aktiv ist. Aufgrund dessen kann das Vanishing Gradient Problem mithilfe der ReLU verhindert werden. Das Verwenden der ReLU Aktivierung für die verborgenen Einheiten erlaubt es tiefe Netze zu trainieren.

2.1.3 *Convolutional Neural Networks*

Das universelle Approximationstheorem sagt zwar aus, dass ein Neuronales Netz mit nur einer verborgenen Schicht jede gesuchte kontinuierliche Funktion approximieren kann. Jedoch ist nicht garantiert, dass eine geeignete Parametrisierung gefunden wird mit der die Funktion dargestellt werden kann, da das Training eines mehrschichtigen Netzes einem nicht konvexen Optimierungsproblem entspricht. Des Weiteren ist es möglicherweise nötig exponentiell viele Neuronen zu verwenden. Da in einem MLP jedes Neuron voll verbunden ist mit den Neuronen der Vorgängerschicht, wächst somit auch die Anzahl an lernbaren Gewichten. Das Problem der Überanpassung tritt auf, falls die Zahl an Parametern im Netz zu stark anwächst und somit das Modell nicht mehr auf unbekannte Daten generalisieren kann (vgl. [GBC16] Kapitel 6.4.1).

Stattdessen hat sich der Trend Deep Learning entwickelt, welcher empirisch betrachtet innerhalb vieler Bereiche den State-of-the-Art bildet. Einer der wesentlichen Gründe weshalb tiefe Netze besser funktionieren ist die Aneinanderreihung von mehreren verborgenen Schichten, welche eine Komposition von vielen nichtlinearen Funktionen bildet. Dies erlaubt dem Netz in den verborgenen Schichten komplexe nichtlineare Transformationen der Eingabedaten zu berechnen. Da ein Netz Ende-zu-Ende trainiert werden kann (siehe Unterabschnitt 2.1.6), lernt es dabei die Eingabe x innerhalb der verborgenen Schichten auf eine Merkmalsrepräsentation $\phi(x)$ abzubilden, auf deren Basis das Model eine geeignete Ausgabe berechnen kann. Das automatische Lernen von $\phi(x)$ ist mitunter der Grund, weshalb neuronale Netze Modelle mit manuell gewählten Merkmalen übertreffen (vgl. [GBC16] Kapitel 6).

Tiefe Netze werden in der Regel nicht komplett in Form von MLPs realisiert, da die Anzahl an Parametern sonst zu groß wird. Besonders bei hochdimensionalen Daten wie z.B. Bildern steigt die Modellkomplexität rasant und sind somit nicht effizient verwendbar. Für tiefe Netze hat sich eine besondere Netzarchitektur hervorgehoben, die sogenannten Convolutional Neural Networks (CNN). Diese haben sich beson-

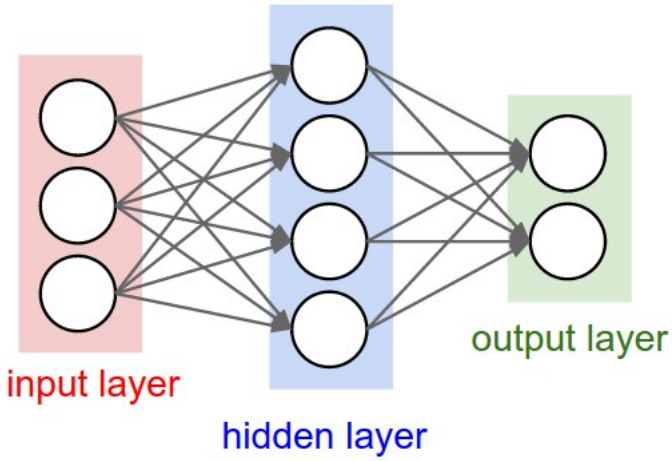


Abbildung 2.1.2: Ein MLP bestehend aus drei Schichten (Eingabeschicht, verborgene Schicht, Ausgabeschicht). Jeder Knoten repräsentiert ein Neuron. Die Ausgabe eines Neurons ist mit allen Neuronen einer Folgeschicht verbunden. Abbildung entnommen aus [LJY19].

ders für Aufgaben im Bereich der Mustererkennung und Computer Vision etabliert [LWL+17, SZ15, PHZ+18]. In den folgenden Unterkapiteln werden einige wichtige Operationen vorgestellt, welche heutzutage in einer CNN Architektur vorkommen.

Convolutional Layer

CNNs sind neuronale Netze, welche in ihrer Architektur Faltungsschichten (Convolutional Layer) verwenden. Faltungsschichten sind besonders effektiv, wenn die Daten eine Rasterstruktur vorweisen, wie z.B. bei Bildern, welche als 2-dimensionale Pixelmatrix dargestellt werden können. Nah beieinander liegende Pixel korrelieren, da sie vermutlich zum selben Objekt gehören. Diese Lokalität innerhalb des Bildes kann von einer Faltungschicht ausgenutzt werden. Daten müssen in der Praxis diskret vorliegen, wenn diese maschinell verarbeitet werden. Beispielsweise werden für ein Bild die Bildpunkte in regulären Intervallen abgetastet. Im Fall von Bildern wird die 2-dimensionale diskrete Faltung betrachtet [2] mit:

$$F(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(i,j) \tag{2.1.7}$$

2 Faltungsschichten sind im Allgemeinem auf n Dimensionen generalisierbar

I notiert dabei die Eingabebildmatrix und K einen 2-dimensionalen Kernel. Eine Faltung ist also eine Überlagerung zwischen der Eingabe und einer Filtermaske. Das Ergebnis $F(i,j)$ bezeichnet eine sogenannte Merkmalskarte (Feature Map), welche auch wieder 2-dimensional und somit als Bild interpretierbar ist. Jeder Punkt der Feature Map ergibt sich als Summe der elementweisen Multiplikation zwischen Bildpunkten eines Ausschnittes von I und Filtergewichten von K, welches im linken Teil der Abbildung 2.1.3 visualisiert ist.

Indem der Filter über das Bild geschoben wird, wird an jedem Punkt eine gewichtete Summe von benachbarten Pixel berechnet (vgl. [GBC16] Kapitel 9.1).

Die Faltung wird konzeptionell über Neuronen ausgewertet. Im rechten Teil der Abbildung 2.1.3 ist dies exemplarisch dargestellt. Bei der Eingabe handelt es sich um ein Bild der Größe 32x32 mit 3 Kanälen (RGB). Jeder Punkt einer Feature Map wird durch ein Neuron repräsentiert. Im Gegensatz zu Neuronen in einem MLP sind diese nicht vollständig verbunden mit der Eingabe. Stattdessen betrachtet ein Neuron jeweils nur einen kleinen Ausschnitt der Eingabe, das sogenannte Rezeptive Feld. Die Pixel innerhalb des Rezeptiven Feldes eines Neurons werden mit der Filtermaske gefaltet (s.o). Innerhalb der Abbildung ist das rezeptive Feld eines Neurons zu sehen. Die Breite und Höhe des Kernels ist in der Regel quadratisch (kxk), wobei das rezeptive Feld sich über die gesamte Tiefe der Eingabe erstreckt. Im Beispiel sind dies drei Kanäle (RGB), womit der Kernel in diesem Fall 3-dimensonal ist. Eine einzelne Faltungsschicht besteht aus d Filtern, von denen jeder eine eigene Feature Map berechnet. Ausgabe einer Faltungsschicht sind also d Feature Maps, ein $dxmxn$ Tensor, wobei d die Tiefe bzw. die Anzahl an Kanälen darstellt, welche wiederum Eingabe einer folgenden Faltungsschicht sein kann. Die Gewichte der Filter sind lernbare Parameter. Da Neuronen, welche zur selben Feature Map gehören, die gleiche Filtermaske verwenden, wird die Anzahl an Gewichten maßgeblich reduziert (Parameter Sharing).

Die Wahl der Filtergröße bestimmt das rezeptive Feld und somit auch den Kontext eines Neurons. Spätere Faltungsschichten bestehen in der Regel aus einer hohen Anzahl an Filtern, um möglichst viele komplexe Merkmale zu lernen. Aus diesem Grund sind in der Praxis nur relativ kleine Kernelgrößen $[1,3,5]$ verwendbar, da während der Faltung die gesamte Tiefe, also alle eingehenden Kanäle, betrachtet wird und das Netz sonst zu viele Parameter enthält. Vor allem bei hochauflösenden Bildern mit großen Objekten ist möglicherweise ein größerer Kontext notwendig. Es ist ausreichend kleinere Filter anzuwenden, da durch mehrere hintereinander folgende Kernel das rezeptive Feld bezüglich des ursprünglichen Eingabebildes sukzessive vergrößert wird [SZ15].

Aufgrund der Equivarianz gegenüber Translation der Faltungsoperation agieren die Neuronen wie Merkmalsdetektoren, welche auf das gesamte Bild angewendet werden.

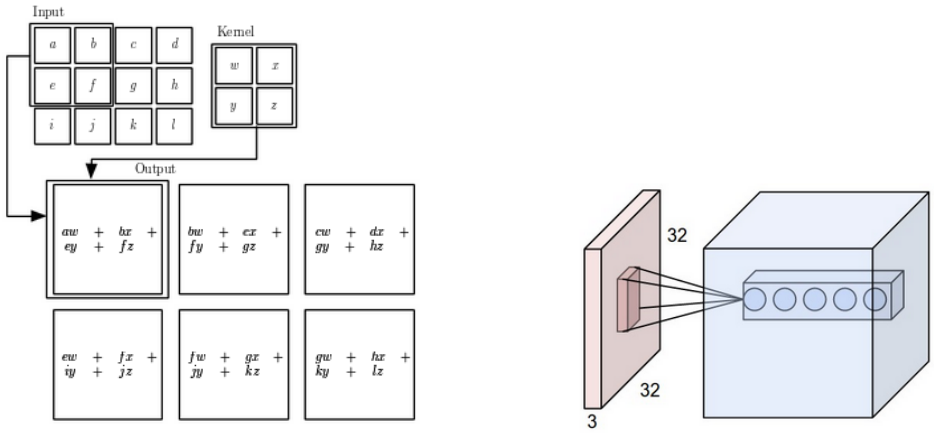


Abbildung 2.1.3: Links: Schematische Darstellung einer Faltungsoperation für ein Eingabbild und einem Kernel. Abbildung entnommen aus ([GBC16] Seite 325).
Rechts: Darstellung einer Faltungsschicht. Die Eingabe der Schicht ist in rot dargestellt, während der blaue Teil die Faltungsschicht visualisiert. Abbildung entnommen aus [LJY19].

Das Aneinanderreihen von Faltungsschichten produziert sukzessiv Feature Maps. In den vorderen Schichten lernt ein CNN elementare Merkmale, wie einzelne Kanten, zu erkennen, während hintere Schichten komplexe und abstrakte Objektstrukturen abbilden [ZF13].

Eine weitere wichtige Form der Faltungsschichten, welche heutzutage verwendet werden, sind die sogenannten transposed convolutions. Diese werden benötigt, um Feature Maps hochzuskalieren [DV16] und werden als lernbares Upsampling betrachtet. Die grobe Idee dieser Faltungsoperation ist, dass eine reguläre Faltung durchgeführt wird, wobei jedoch die Eingabe mit 0 an entsprechenden Stellen vergrößert wird. Dadurch sind kleinere Schrittweiten möglich, welches zu größeren Ausgabe Feature Maps führt [DV16].

Pooling

Die Pooling Operation wird nach einer Faltungsschicht angewendet und dient zur Verkleinerung der Auflösung von Feature Maps. Abbildung 2.1.4 zeigt die Funktionsweise einer Poolingschicht. Analog zur Faltungsschicht wird ein Fenster über eine Feature Map geschoben. Punkte, welche zu einem Fenster gehören, sind mit der gleichen Farbe unterlegt. Diese werden jeweils über eine Aggregatfunktion auf einen einzelnen Wert abgebildet. Am häufigsten wird entweder das Maximum (Max Pooling) oder der

Mittelwert (Average Pooling) gewählt (vgl. [GBC16] Kapitel 9.3). Bei einer Schrittweite und einer Fenstergröße von 2 lässt sich so beispielsweise die Auflösung einer Feature Map halbieren. Im Gegensatz zur Faltungsschicht wird beim Pooling jede einzelne Feature Map separat betrachtet.

Durch das Anwenden von Pooling wird eine leichte Form der Translationsinvarianz eingeführt und ist insbesondere dann interessant, wenn die exakte Position von bestimmten Merkmalen vernachlässigt werden kann. Beispielweise ist für die Klassifikation eher wichtiger, ob ein Merkmal in einem Bild enthalten ist und nicht die konkrete Stelle an der es auftaucht.

Bei einer Poolingschicht besteht Informationsverlust, da eine Nachbarschaft von Punkten auf einen Wert reduziert wird. Jedoch hat dieser einen wesentlichen Vorteil für die Rechenkomplexität. Diese ist durch die kleineren Feature Maps stark reduziert.

2.1.4 *CNN Architektur*

Eine gewöhnliche CNN Architektur besteht aus mehreren Faltungsschichten mit gelegentlich folgenden Poolingschichten. Durch die sequentielle, schichtweise Berechnung werden hierarchisch immer komplexere Merkmale berechnet, welche sich aus den simpleren Merkmalen der vorderen Schichten zusammensetzen. Dieser Teil gilt als Merkmalsextraktor (Feature Encoder), welcher einen Tensor der Form $d \times m \times n$ berechnet, wobei d der Anzahl an Feature Maps und $m \times n$ der Auflösung entspricht. Im Fall von Regression bzw. Klassifikation wird dieser Tensor zu einem 1-D Vektor umgeformt und es folgt ein MLP[3], welches die Ausgabe berechnet. Aufgrund der Vollverbundenheit eines MLP mit der Eingabe befindet sich der Großteil der Parameter eines Netzes in den vollverbundenen Schichten. Um Overfitting zu vermeiden, sind Regularisierungsverfahren notwendig. Beispielsweise kann dafür Dropout benutzt werden.

Dropout

Dropout ist eine bewährte Methode um Netze zu regularisieren und wird hauptsächlich innerhalb einer verborgenen Schicht eines MLPs verwendet. Während des Trainings werden zufällig Aktivierungen von Neuronen auf 0 gesetzt, wobei die Dropoutwahrscheinlichkeit p angibt mit welcher Wahrscheinlichkeit ein Neuron deaktiviert wird [SHK+14]. Während des Trainings entstehen so unterschiedliche Subnetze, in denen einige Neuronen ausgeschaltet sind. Die Idee des Dropouts basiert auf der

3 in diesem Kontext auch einfach Fully Connected Layers genannt.

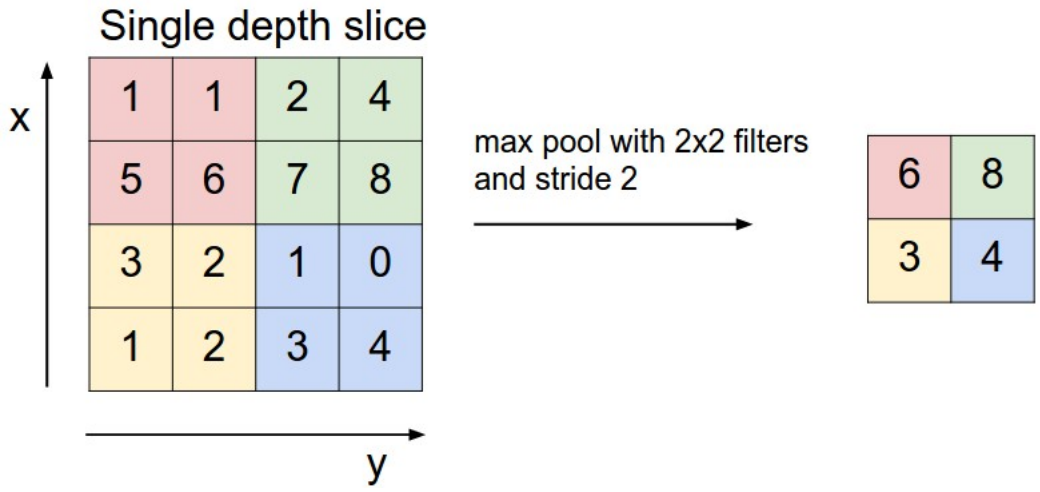


Abbildung 2.1.4: Funktionsweise einer Poolingoperation. Nachbarschaften aus Pixeln werden über eine Statistik zusammengefasst. In diesem Beispiel wird jeweils das Maximum gewählt. Dies wird Max Pooling genannt. Abbildung entnommen aus [LJY19].

Grundlage des Baggings. Mit Bagging wird ein Ensemble von Basismodellen trainiert, welche gemeinsam robuster sind. Das Einführen von Rauschen, durch das Ausschalten von Neuronen, führt zu stabileren verborgenen Einheiten, da das Netz lernen muss Informationen aus besonders vielen Neuronen zu extrahieren, da diese je nach Subnetz ausfallen können. Jedes einzelne Neuron muss einigermaßen unabhängig davon sein, welche Neuronen aktuell aktiv sind. Es wird verhindert, dass das Netz durch einzelne Neuronen die Trainingsdaten auswendig lernt und somit Overfitting auftritt. Falls Dropout während der Optimierung eingesetzt wird, werden die Gewichte in der Schicht in der es angewendet wurde mit der Wahrscheinlichkeit p skaliert. Bei der Evaluierung bleibt Dropout jedoch ausgeschaltet. Es kann gezeigt werden, dass dies annäherungsweise dem Auswerten eines Ensemble der unterschiedlichen Subnetze entspricht (vgl. [GBC16] Kapitel 7.12.).

2.1.5 *Batch Normalization*

Batch Normalisierung (Batch Normalization) ist ein weiteres wichtiges Konzept, welches in aktuellen tiefen Netzen verwendet wird (vgl. [GBC16] Kapitel 8.7.1). Jede

Schicht [4] innerhalb eines Netzes hängt von den Berechnungen der vorherigen Schichten ab. Während des Trainings werden die Gewichte des Netzes verändert und somit auch die Aktivierungen, welche als Eingaben für folgende Schichten verwendet werden. Die Verteilung der eingehenden Aktivierungen für ein Layer ändert sich, worauf sich folgende Schichten anpassen müssen. Dieser Effekt wird auch internal covariate shift genannt. Je tiefer das Netz ist, desto größer ist der Einfluss von kleinen Änderungen in der Verteilung der Aktivierungen auf spätere Schichten. Ist diese zu stark bzw. tritt zu oft auf, kann dies zu Schwierigkeiten im Optimierungsprozess führen. Das Training eines Netzes ist robuster, wenn die Aktivierungen normalisiert sind und innerhalb eines kontrollierten Wertebereichs liegen. Eine Möglichkeit dafür ist das Verwenden von Batch Normalisierung [IS15].

Ein Batch bezeichnet eine Menge von Trainingsbeispielen. Seien $B = \{x^1, x^2, x^3, ..\}$ die Aktivierungen x^i eines Batches. So sind die ersten beiden Momente (Mittelwert μ_B, Varianz σ_B^2) eines Batches gegeben mit:

$$\mu_B = \frac{1}{|B|}\sum^{|B|} x_i, \qquad \sigma_B^2 = \frac{1}{|B|}\sum^{|B|}(x_i - \mu_B)^2 \tag{2.1.8}$$

Die normalisierten Aktivierungen ergeben sich durch:

$$\hat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{2.1.9}$$

ϵ ist eine kleine Konstante für numerische Stabilität, um eine Division durch 0 zu verhindern. Die normalisierten Aktivierungen $\hat{x_i}$ sind also standardisiert mit einem Mittelwert von 0 und einer Varianz von 1. Durch das Standardisieren kann es jedoch vorkommen, dass eine Schicht an Modellierungskomplexität verliert (vgl. [GBC16] Kapitel 8.7.1). Aus diesem Grund werden die normalisierten Aktivierungen noch nach

$$y_i = \gamma\hat{x_i} + \beta \tag{2.1.10}$$

skaliert und verschoben. Die Skalierungs- γ und Verschiebungsfaktoren β sind lernbare Parameter. Da die $\hat{x_i}$ standardisiert sind, hilft die Reparametrisierung beim Optimieren des Netzes.

Batch Normalisierung hat auch einen regularisienden Effekt auf das Modell. Da jeder Batch aus zufällig gezogenen Beispielen besteht, entsprechen die Batchstatistiken nur Schätzungen, welche so Rauschen in die Eingabe eines Layers einführen, Das Netz

4 außer der Eingabeschicht

muss innerhalb der Layer lernen gegen diese Form von Rauschen robust zu sein. Die Hauptmotivation für Batch Normalisierung ist dennoch das stabilere und schnellere Lernen in tiefen Netzen (vgl. [GBC16] Kapitel 8.7.1).

2.1.6 *Training von neuronalen Netzen*

Neuronale Netze können verwendet werden, um Klassifikations- und Regressionsprobleme zu modellieren. Damit ein neuronales Netz korrekte Vorhersagen treffen kann, müssen die Parameter des Modells auf das spezifische Problem angepasst werden. Es handelt sich also um einen Fall des überwachten Lernens. Hierfür wird ein Trainingsdatensatz benötigt, welcher mit der gesuchten Ausgabe annotiert ist. Sei $D_{train} = \{(x^1, y^1), (x^2, y^2), ...\}$ eine Menge von Trainingsbeispielen, wobei x^i den Eingabedaten und y^i der dazugehören Zielvariable entspricht. Im Fall von Klassifikation ist y^i der Klassenindex. Bei Regression gilt $y^i \in \mathbb{R}^n$ (vgl. [HTF01] Kapitel 2). Mithilfe der Trainingsdaten sollen Muster in den Daten gelernt werden, damit das Modell auf unbekannten Eingabedaten korrekte Vorhersagen treffen kann.

Der aktuelle Trainingsfortschritt eines Modells kann mit einer Kostenfunktion L (loss function) quantifiziert werden. Sie gibt den Fehler zwischen der Ausgabe des Netzes $\hat{y}$ und der annotierten Zielvariablen y an. So lässt sich das Lernen der Modellgewichte w auf folgendes Optimierungsproblem

$$\min_w \frac{1}{|D|} \sum_D L(y^i, \hat{y}^i) \tag{2.1.11}$$

zurückführen. Die Gewichte werden verändert, sodass die Kostenfunktion minimiert wird und somit der Fehler verringert wird. Im Allgemeinen sind analytische Verfahren nicht praxisrelevant, da es sich hierbei um ein nichtkonvexes Optimierungsproblem handelt [CHM+14]. Für L wird stattdessen eine differenzierbare Funktion gewählt. Dies erlaubt es ein iteratives Optimierungsverfahren auf Basis des Gradienten der Kostenfunktion zu verwenden. Mit dem sogenannten Gradientenabstieg (Gradient Descent) werden die Gewichte schrittweise mithilfe des Gradienten aktualisiert. Komplexere Optimierungsmethoden für neuronale Netze, wie z.B. Newton's Method, welche die Hesse Matrix betrachten, sind schwierig anzuwenden und ist ein aktuelles Thema in der Forschung [WLL18]. In der Praxis werden tiefe Netze aufgrund der Einfachheit und den hiermit erzielten Ergebnissen mithilfe eines iterativen Gradientenabstieges trainiert. Der Gradient besteht aus den partiellen Ableitungen des aktuellen Punktes der Fehlerfunktion bezüglich der Modellparameter w_i und zeigt in die Richtung der größten Steigung der Kostenfunktion. Um ein Minimum zu erreichen wird

der negative Gradient verwendet (vgl. [GBC16] Kapitel 4.3). Es ergibt sich folgende iterative Aktualisierungsregel:

$$w_{i,j} \leftarrow w_i - \eta \frac{\partial L}{\partial w_{i,j}} \tag{2.1.12}$$

η bezeichnet die Lernrate, die Schrittweite mit der in Richtung des negativen Gradienten optimiert wird. Eine zu kleine Lernrate führt zu langen Trainingszeiten. Ist diese jedoch zu groß, divergiert das Verfahren. Es sei an dieser Stelle erwähnt, dass aufgrund der Nichtkonvexität ein Optimum nicht dem globalen Optimum entsprechen muss. In der Praxis werden lokale Minima gefunden, welche in der Regel zu ausreichend genauen Modellen führen (vgl. [GBC16] Kapitel 5.9).

Da neuronale Netze aus einer großen Anzahl an Gewichten bestehen, benötigen diese viele Trainingsdaten. Für einen einzigen Optimierungsschritt muss Gleichung 2.1.12 auf dem gesamten Trainingsdatensatz ausgewertet werden, um darauf den Gradienten zu bestimmen. Bei großen Datensätzen ist dies nicht praktikabel. Stattdessen wird das Trainingsverfahren batchweise durchgeführt, welches einer Approximation des Gradienten des kompletten Trainingsdatensatz entspricht. Diese Abwandlung wird Stochastic Gradient Descent genannt (vgl. [GBC16] Kapitel 5.9). Da der Gradient eines Batches schneller berechnet werden kann, sind in der Praxis so mehr Gradientenschritte möglich und das Training somit weitaus schneller.

Das Berechnen der partiellen Ableitungen ist für die Gewichte der Neuronen innerhalb der Ausgabeschicht ohne weiteres möglich. Mit der Kettenregel ergibt sich:

$$\frac{\partial L}{\partial w_{i,j}} = \frac{\partial L}{\partial \varphi_j} \frac{\partial \varphi}{\partial h_j} \frac{\partial h_j}{\partial w_{i,j}} \tag{2.1.13}$$

Für die Gewichte der verborgenen Schichten ist das nicht der Fall. Hierfür wird ein Verfahren namens Backpropagation benötigt.

Backpropagation

Das Trainieren von tiefen Netzen mithilfe des Gradientenabstieges erfordert das Bilden der partiellen Ableitungen bezüglich der Gewichte in den verborgenen Schichten. Da diese jedoch nicht direkt an der Ausgabe beteiligt sind, ist die partielle Ableitung nicht ohne weiteres effizient berechenbar. Beim Anwenden der Kettenregel für verborgene Einheiten in einer Schicht l zeigt sich, dass die Gradienten der Folgeschichten benötigt werden.

$$\frac{\partial L}{\partial w^l_{i,j}} = \frac{\partial L}{\partial f^n} \frac{\partial f^n}{\partial f^{n-1}} \cdots \frac{\partial \varphi^l_j}{\partial h^l_j} \frac{\partial h^l_j}{\partial w^l_{i,j}} \tag{2.1.14}$$

Eine Möglichkeit um diese effizient zu bestimmen ist die Methode der Backpropagation [RHW86]. Sie basiert auf der Idee, dass der globale Fehler am Ende des Netzes rückwärts durch das Netz bis zum Anfang weitergeleitet werden kann. Währenddessen wird an den einzelnen Schichten ein lokaler Fehler berechnet, welcher sich aus dem Fehler der nächsten Schicht zusammensetzt. Betrachte Hilfsvariable δ_j, welche den lokalen Fehler des Neurons j aus den Fehlern folgender Schichten rekursiv berechnet. Für die Ausgabeschicht gilt Gleichung 2.1.13. Innerhalb verborgener Schichten setzt sich δ_j als Summe aus den Fehlern folgender Schichten nach

$$\delta_j^{(l)} = \frac{\partial L}{\partial \varphi_j^{(l)}} \frac{\partial \varphi_j^{(l)}}{\partial h_j^{(l)}} = (\sum_k w_{j,k}^{(l+1)} \delta_k^{(l+1)}) \varphi_j'^{(l)} \tag{2.1.15}$$

zusammen. Die partiellen Ableitungen in Abhängigkeit von δ sind somit folglich aus der Kettenregel gegeben mit

$$\frac{\partial L}{\partial w_{i,j}^{(l)}} = \frac{\partial L}{\partial \varphi_j^{(l)}} \frac{\partial \varphi_j^{(l)}}{\partial h_j^{(l)}} \frac{\partial h_j^{(l)}}{\partial w_{i,j}^{(l)}} = \delta_k^l h_i^{(l-1)}. \tag{2.1.16}$$

Die partiellen Ableitungen des Netzes werden also rekursiv berechnet, indem der Fehler von der Ausgabeschicht bis zur Eingabeschicht rückwärts durch das Netz fließt.

Die Aktualisierungsregel für die Gewichte bei einem Gradientenabstieg mit Backpropagation zum Berechnen der partiellen Ableitungen lässt sich mithilfe der Hilfsvariablen δ ausdrücken durch:

$$w_{i,j}^{(l)} \leftarrow w_{i,j}^{(l)} - \eta \frac{\partial L}{\partial w_{i,j}^{(l)}} \tag{2.1.17}$$

Vanishing Gradient

Das Vanishing Gradient Problem ist eine Schwierigkeit, welches bei der Optimierung von tiefen Netzen auftreten kann. Dieses beschreibt das Phänomenen, dass die Gradienten für die Gewichte in den vorderen Schichten zunehmend kleiner werden und

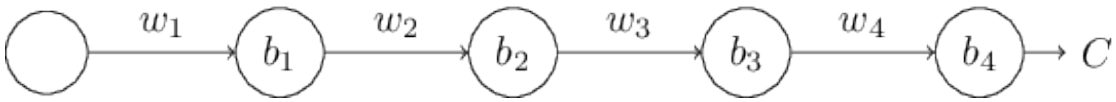


Abbildung 2.1.5: Veranschaulichung des Vanishing Gradient Problems. Abbildung entnommen aus [Nie18].

gegen 0 gehen. Dadurch ist somit keine Aktualisierung der Gewichte mehr möglich (s. Gleichung 2.1.12). Die Ursache kann an einem einfachen Beispiel erläutert werden (s. Abbildung 2.1.5). In der Abbildung ist ein Netz aus vier Schichten mit je einem Neuron dargestellt. Für die partielle Ableitung w_1 ergibt sich nach der Kettenregel:

$$\frac{\partial C}{\partial w_1} = \frac{\partial C}{\partial \varphi_4}\frac{\partial \varphi_4}{\partial h_4}\frac{\partial hi_4}{\partial \varphi_3}\dots\frac{\partial \varphi_1}{\partial h_1}\frac{\partial h_1}{\partial w_1} \tag{2.1.18}$$

Angenommen die Neuronen besitzen eine Sigmoidaktivierung, dann lässt sich oben genannte Gleichung vereinfachen zu:

$$\frac{\partial C}{\partial w_1} = \sigma'(h_4)w_4\sigma'(h_3)w_3\sigma'(h_2)w_2\sigma'(h_1)\varphi_0 \tag{2.1.19}$$

Es gilt $\sigma'(x) \leqslant 0.25 \ \forall x$. Durch jede weitere Schicht gelangt ein weiterer Faktor mit $\sigma'(x)$ in die partielle Ableitung, sodass bei tiefen Netzen die Gradienten für die vorderen Gewichte verschwindend klein werden. Dies führt dazu, dass während des Gradientenabstiegs insbesondere die vorderen Gewichte nie aktualisiert werden (s. Gleichung 2.1.12).

2.2 MULTI TASK LEARNING

Die Idee des Multi Task Learnings (MTL) hat sich in vielen Anwendungsgebieten, wie z.B. Computer Vision und Natural Language Processing, etabliert [Rud17]. MTL ist ein Spezialfall des maschinellen Lernens und tritt dann auf, wenn ein Modell mit mehreren Aufgaben zur selben trainiert wird.

Zusätzliche Aufgaben ermöglichen einem Modell weitere Informationen aus verwandten Aufgaben miteinzubeziehen. Unter der Annahme, dass diese semantisch ähnlich sind, können domänenspezifische Informationen aus den Aufgaben extrahiert werden, welche in einer einzelnen betrachteten Aufgabe möglicherweise nicht enthalten wären. Diese zusätzlichen Informationen können bei einer geeigneten Gewichtung der jeweiligen Kostenfunktionen zu einem Modell führen, welches bessere Leistungen als ein Single Task Model in den jeweiligen Aufgaben erzielt. MTL Modelle bestehen in der Regel aus geteilten Parametern, welche von allen Aufgaben verwendet werden und darauf folgenden taskspezifischen Parametern. Einer der Gründe, weshalb MTL zu besseren Ergebnissen führen kann, ist das Konzept des induktiven Bias. Innerhalb des gemeinsamen Teiles werden Parameter bevorzugt, welche auf unterschiedliche Aufgaben generalisieren können.

Ein weiterer Vorteil besteht im Effizienzgewinn, da ein großer Anteil an Parametern zwischen Aufgaben geteilt wird. Zusätzlich ist es nicht mehr notwendig für jede

Aufgabe ein eigenes Modell zu trainieren bzw. auszuwerten. Alle Ausgaben für die unterschiedlichen Aufgaben ergeben sich aus der einmaligen Auswertung des MTL Modells. Die einfachste Art MTL umzusetzen ist die Verwendung von neuronalen Netzen.

2.2.1 *MTL mit Deep Learning*

Die generelle Idee einer gemeinsamen Repräsentation, welche zwischen allen Aufgaben geteilt wird und taskspezifischen Ausgabemodellen, lässt sich mithilfe eines neuronalen Netzes realisieren. Abbildung 2.2.1 visualisiert schematisch ein Netz hierfür. Zu Beginn wird die Eingabe durch eine Reihe von geteilten Schichten auf eine gemeinsame Merkmalsrepräsentation abgebildet. Im Falle von Bildern ist die Verwendung von mehreren Faltungsschichten sinnvoll (s. Abschnitt 2.1). Dieser Teil gilt deswegen auch als Feature Encoder. Darauf folgt für jede betrachtete Aufgabe ein separater Ausgabezweig, welcher aus eigenen aufgabenspezifischen Schichten besteht. Dies wird aufgrund des geteilten Feature Encoders auch Hard Parameter Sharing genannt. Im Gegensatz dazu steht das Soft Parameter Sharing. Jede Task besitzt hier einen eigenen Feature Encoder, wobei diese jedoch die gleiche Architektur verwenden. Mit einem Regularisierungsterm wird versucht die Parameter der Feature Encoder ähnlich zu halten.

Damit das MTL Modell für jede der betrachteten Aufgaben gute Vorhersagen treffen kann, werden die einzelnen Kostenfunktion der unterschiedlichen Aufgaben gleichzeitig optimiert. Dafür werden diese in einer Gesamtkostenfunktion vereint:

$$L_{gesamt} = \sum_i w_i L_i \tag{2.2.1}$$

Hierbei bezeichnet L_i den Loss des i-ten Tasks. Die Gewichte der Linearkombination müssen in der Praxis heuristisch gewählt oder in einer aufwändigen Hyperparametersuche bestimmt werden. Da das Training des Modells selbst schon viel Zeit in Anspruch nimmt, führt die Suche zu erheblichen Mehraufwand. Jedoch ist eine geeignete Wahl der Gewichte für das Training entscheidend [KGC17].

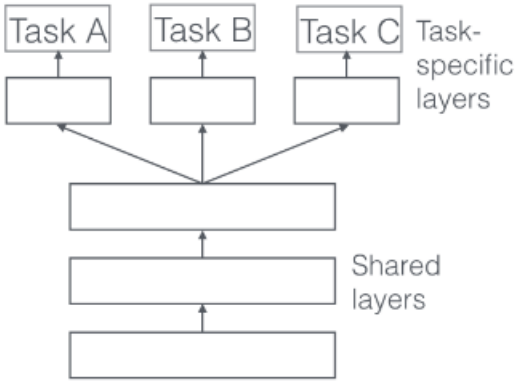


Abbildung 2.2.1: Hard Paramter Sharing mithilfe eines Neuronalen Netzes für das MTL. Abbildung entnommen aus [Rud17].

3 VERWANDTE ARBEITEN

Nachdem im vorherigen Kapitel die wesentlichen Grundlagen für die Arbeit vorgestellt wurden, wird dieses Kapitel eine Auswahl von Arbeiten behandeln, welche einen besonderen Bezug zur vorliegenden Arbeit besitzen. In der Methodik wird ein MTL Modell umgesetzt, welches unterschiedliche Aufgaben der Gesichtsanalyse realisiert. Das Kapitel ist nach den betrachteten Aufgaben strukturiert, welche innerhalb dieser Arbeit behandelt werden. Zu Beginn wird in Abschnitt 3.1 ein Einblick in bestehende Methoden für die Landmarkenlokalisierung gegeben. Abschnitt 3.2 behandelt die Detektion von Gesichtern. In Abschnitt 3.3 werden aktuelle Methoden zur semantischen Segmentierung vorgestellt. Mit Abschnitt 3.4 wird ein Auszug an Arbeiten präsentiert, welche für die Attributerkennung vorgestellt wurden. Zum Schluss werden innerhalb Abschnitt 3.5 relevante Arbeiten aus dem Bereich des Multi Task Learnings diskutiert, auf deren Ideen die Methodik dieser Arbeit basiert.

3.1 LANDMARKENLOKALISIERUNG

Landmarken werden als bestimmte Schlüsselpunkte innerhalb eines Bildes definiert. Für die Gesichtsanalyse werden Landmarken benutzt um besondere Punkte des Gesichtes zu markieren. Beispielsweise können somit wichtige Bereiche eines Gesichtes, wie z.B. Augen, Nase, usw., erfasst werden (s. Abbildung 3.1.1). Durch das Verhältnis zwischen den Punkten beinhalten diese Information über die Pose, Mimik, und Verformungen eines Gesichtes. Zahlreiche Aufgaben profitieren von einer präzisen Lokalisierung der Hauptmerkmale eines Gesichtes [WJ18, JT16]. Deshalb stellt die Landmarkendetektion einen essentiellen Schritt für Folgeaufgaben dar [WJ18]. Beispielsweise können gefundene Landmarken zur Normalisierung der Ausrichtung verwendet werden, welches einen positiven Effekt für die Gesichtserkennung hat [WJ18, JT16].

Die Landmarkenerkennung befasst sich mit dem automatischen Lokalisieren von vorher definierten Schlüsselpunkten, welche über ihre Koordinaten gegeben sind. Existierende Methoden lassen sich in zwei Ansätze unterteilen.

Abbildung 3.1.1: Visualisierung von wichtigen Punkten in einem Gesicht. In diesem Beispiel sind 5 Landmarken (grün) definiert, welche mit den Augen, der Nase und den Mundspitzen korrespondieren. Bild und Annotation entnommen aus dem CelebA Datensatz [LLWT14].

3.1.1 *Active Appearance Model*

Frühere Methoden basieren auf dem sogenannten Active Appearance Model. Es handelt sich hierbei um ein Verfahren mit dem versucht wird die Textur und Form eines Objektes (z.B. Gesichter) explizit durch ein parametrisiertes Modell zu erfassen [CET98]. Dafür wird ein shape model und ein appearance model gelernt. Durch das Anpassen der Koeffizienten des Modells können somit Verformungen und Aussehen des Gesichtes ausgedrückt werden.

Das Modell wird in einer Trainingsphase gelernt. Dafür werden Gesichtsbilder mit annotierten Landmarken benötigt. Sei $\mathbf{Y}$ die Datenmatrix, welche die Annotation aller Bilder enthält, wobei die Landmarken je Bild zeilenweise notiert sind. Auf dieser wird eine Hauptkomponentenanalyse durchgeführt. Bei der Hauptkomponentenanalyse handelt es sich um eine Methode zur Berechnung einer Transformation der Daten in einen neuen Vektorraum. Die bestimmten orthogonalen Basisvektoren $\mathbf{v_i}$ verlaufen entlang der beobachteten Varianz von $\mathbf{Y}$ (vgl. [Bis07] Kaptitel 12.1). Da durch die Landmarken die Form eines Gesichtes widergespiegelt wird, modellieren die $\mathbf{v_i}$ somit die Verformungen der Gesichter im Datensatz. Die Landmarken $\mathbf{y}$ für ein Bild lassen sich aus diesen nach

$$\mathbf{y}' = \bar{\mathbf{y}} + \sum_{i}^{k} p_i \mathbf{v_i} \tag{3.1.1}$$

rekonstruieren, wobei $\bar{\mathbf{y}}$ den Mittelwert bezeichnet. Eine Gesichtsform besteht also aus einer mittleren Form und einer Linearkombination der Verformungen, wobei die Koeffizienten p_i den Anteil dieser bestimmen. Dieselbe Idee wird für das appearance model verwendet mit dem Unterschied, dass die Hauptkomponentenanalyse auf den mithilfe von $\mathbf{y}'$ und einer Warpingfunktion W normalisierten Eingabebildern ausgeführt wird. Es ergibt sich analog für das appearance model:

$$I(W(\mathbf{y}')) = \bar{\mathbf{A}} + \sum_{i}^{k} \lambda_i \mathbf{a_i} \tag{3.1.2}$$

Um Landmarken für neue Testbilder vorherzusagen, werden die Koeffizienten p und λ des Active Appearance Model auf die jeweiligen Gesichter angepasst. Dies kann über folgendes Optimierungsproblem gelöst werden:

$$\min_{p,\lambda} \|(\bar{\mathbf{A}} + \sum_{i}^{k} \lambda_i \mathbf{a_i}) - I(W(p))\|^2 \tag{3.1.3}$$

Mit Gleichung 3.1.1 können darauf die Landmarken ermittelt werden [WJ18]. Diese Modelle sind jedoch nicht robust gegenüber Variabilität in den Bildern (Verdeckung, Beleuchtung, etc.). In diesen Fällen sind die Ergebnisse nicht mehr zufriedenstellend.

3.1.2 *Regression*

Das Lokalisieren von Landmarken lässt sich als Regressionsproblem formulieren. Es ist eine Funktion $f(x) = \mathbf{y}$ gesucht, welche von einem Eingabebild x auf die Koordinaten der Landmarken $\mathbf{y}$ abbildet, womit es sich also um einen Fall von multivariater Regression handelt. Hierfür haben sich CNNs (s. Unterabschnitt 2.1.3) durchgesetzt, welche insbesondere bei Bilddaten zu hervorragenden Ergebnissen führen [WJ18]. Eine der ersten Arbeiten hierzu basiert auf einem kaskadierendem Ansatz von mehreren CNNs [SWT13]. Die CNNs bestehen aus mehreren Faltungs- und Poolingschichten gefolgt von einem MLP, welches auf Basis der berechneten Merkmale die Koordinaten der Landmarken bestimmt. Die Idee des Ansatzes ist in Abbildung 3.1.2 zu sehen, welche aus drei Schritten besteht. Zu Beginn werden drei globale CNNs verwendet, welche jeweils einen unterschiedlichen aber überlappenden Ausschnitt des Gesichtes sehen. Die Eingabe der CNNs ist in Gelb unterlegt. Jedes der drei CNNs berechnet jeweils initiale Positionen für die einzelnen Landmarken, welche gemittelt werden (rote Punkte). In den folgenden Schritten werden diese weiter verfeinert. Hierfür werden weitere CNNs verwendet, welche jeweils nur für genau eine Landmarke zuständig

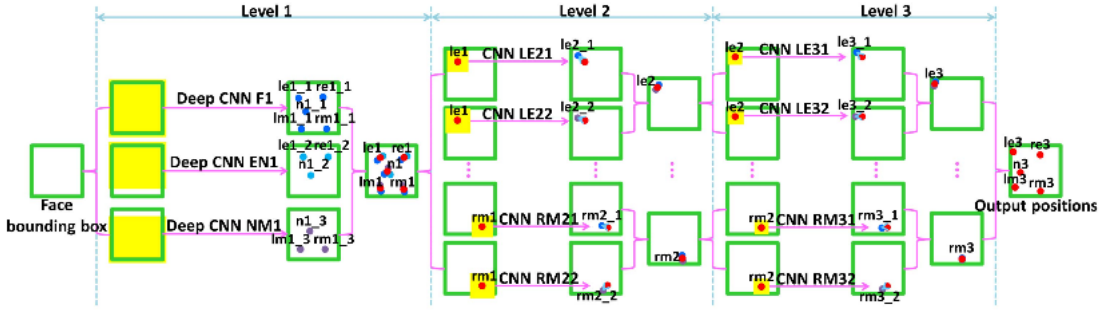


Abbildung 3.1.2: Ein kaskadierender Ansatz für die Landmarkenerkennung auf Basis von CNNs. Anfangs werden global initiale Positionen geschätzt. Folgende CNNs sagen jeweils nur eine Landmarke voraus. Sie betrachten nur noch einen lokalen Ausschnitt um die Landmarke, welche sie vorhersagen sollen. Die lokalen Ausschnitte ergeben sich aus den Schätzungen des vorherigen Schrittes. Abbildung entnommen aus [SWT13].

sind. Die CNNs betrachten als Eingabe jeweils nur einen lokalen Bildausschnitt um die Landmarke, welche sie vorhersagen sollen. Diese sind aus dem vorherigen Schritt entnehmbar (gelb unterlegt). Jede Landmarke wird von zwei verschiedenen CNNs vorhergesagt, welche wiederum gemittelt werden. Im letzten Schritt wird dies noch ein weiteres Mal durchgeführt, welches die finalen Ausgabe für die Landmarken ergibt.

Ein aktueller Trend im Bereich des Deep Learnings ist das Verwenden von Fully Convolutional Networks. Diese Architektur verwendet nur Faltungs- bzw. Poolingschichten. Das MLP am Ende des Netzes wird vermieden, da diese im allgemeinen einen Großteil der Parameter in einem Netz ausmachen. Mit dieser Architektur ist es möglich Ausgaben zu modellieren, welche dieselbe Struktur wie die Eingabe besitzen. Somit kann beispielsweise eine Vorhersage auf Pixelebene durchgeführt werden [LSD14, NYD16, BT16]. Dieses Konzept wurde z.B. in [MRR18] zur Landmarkendetektion verwendet. Anstelle eines MLPs, welches die Koordinaten der Landmarken ausgibt, wird die Ausgabe durch eine Heatmap modelliert, welche dieselbe Auflösung wie das Eingabebild besitzt (s. Abbildung 3.1.3). Hohe Werte in der Heatmap sind in rötlichen Tönen dargestellt, welche auf eine mögliche Landmarke hinweisen. Dadurch das kein MLP verwendet wird ist die Methode unabhängig von der Auflösung der Eingabebilder, da die Eingabeschicht des MLPs die Dimension der Merkmale, welche von den Faltungsschichten extrahiert wird, festlegt. Zusätzlich ist in der Methode keine besondere Behandlung von Bildern, in denen mehrere Gesichter vorkommen, notwendig.

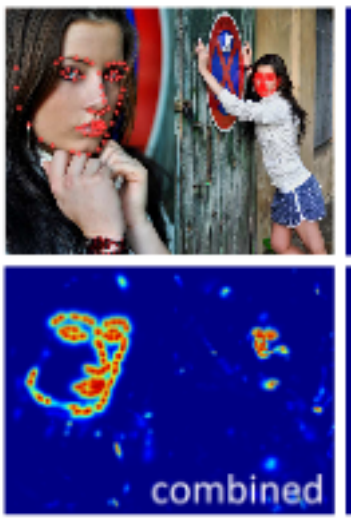


Abbildung 3.1.3: Das Detektieren von Landmarken mit Fully Convolutional Networks kann durch Heatmap Regression modelliert werden. Abbildung entnommen aus [MRR18].

Ein größeres Problem bei der Landmarkenerkennung stellt die Variabilität innerhalb der Bildstile dar. Oft existieren starke Unterschiede in der Beleuchtung und im Kontrast innerhalb eines Datensatzes. Im Weiteren können sowohl Farb- als auch Grauwertbilder vorkommen. Dong et al. verwenden ein Generative Adversarial Network (GAN) um dieses Problem zu behandeln [DYOY18]. Generative Adversarial Networks sind ein aktuelles Forschungsthema innerhalb des tiefen Lernens. Es handelt sich dabei um generative Modelle, die das Ziel verfolgen neue realistische Samples zu erzeugen, welche ähnlich zu den Trainingsdaten sind. GANs bestehen aus einem Generator und einem Diskriminator. Während des Trainings werden diese gegenseitig trainiert. Der Generator erzeugt neue Samples während der Diskriminator versuchen muss zwischen realen und generierten Daten zu unterscheiden. Während des Trainings lernt so der Generator immer realistischere Samples zu erzeugen. Jedoch ist das Training von GANs sehr instabil (vgl. [GBC16] Kapitel 20.10.4). Das GAN wird benutzt um Gesichtsbilder mit unterschiedlichen Stilen zu augmentieren. Diese zusätzliche Varianz in den Daten führt zu besseren Detektionsergebnissen.

Eine weitere aktuelle Arbeit schlägt eine neue Kostenfunktion für die Landmarkendetektion vor [FKA+17]. In der Regel wird für eine Regression der L2 Loss gewählt. Unter der Annahme, dass die Residuen normalverteilt sind mit einem Erwartungswert von 0, entspricht das Minimieren des quadratischen Fehlers einer Maximum

Likelihood Schätzung (vgl. [Bis07] Kapitel 3.1.1). Problematisch sind jedoch Ausreißer. Während der Optimierung wird ein Modell sich eher auf Beispiele mit größeren Fehlern fokussieren, kleinere Fehler werden wegen des Quadrates weniger stark bestraft. Der sogenannte Wing Loss versucht dies zu umgehen, indem für kleinere Fehler sich die Kostenfunktion nichtlinear verhält und für größere linear (s. Unterabschnitt 4.1.2).

3.2 GESICHTSDETEKTION

Bei der Gesichtsdetektion handelt es sich um einen Spezialfall der Objektdetektion. Im Folgenden werden einige Ansätze diskutiert, welche sich innerhalb der Objektdetektion etabliert haben. Diese können natürlich auf den Spezialfall Gesichter konkretisiert werden. Die Aufgabe ist es innerhalb eines Bildes die Positionen der enthaltenen Gesichtern zu finden. Für die Detektion werden normalerweise Bounding Boxes verwendet, welche die Gesichter durch ein Rechteck eingrenzen. Ansätze für die Detektion lassen sich grob in drei Methoden unterteilen.

- **Sliding Window:**

  Die ersten Verfahren basieren auf einem naiven Sliding Window Ansatz. Dafür wird ein Fenster über das Bild geschoben, womit unterschiedliche Bildausschnitte definiert werden. Für jeden Bildausschnitt entscheidet ein Klassifikator, ob ein gesuchtes Objekt in dem Fenster enthalten ist. Aufgrund der möglichen Varianz in der Skalierung der Objekte müssen in der Regel verschiedene Fenstergrößen benutzt werden. Zusätzlich müssen mit den Fenstern alle Position abgedeckt werden, welches mit zum Teil überlappenden Ausschnitten realisiert wird. Aufgrund der hohen Anzahl an zu überprüfenden Regionen sind diese Methoden ziemlich rechenintensiv [LOW+18].

- **Region Proposal:**

  Anstelle anzunehmen, dass die Objekte an jeder Stelle im Bild vorkommen können, werden hier Bildausschnitte betrachtet, welche besonders auffällig sind. Das Vorschlagen von Regionen in denen das Objekt enthalten sein könnte, kann entweder als externer Algorithmus (z.B. Selective Search) realisiert werden oder innerhalb des Klassifikators mittrainiert werden [LOW+18].

- **Einstufige Verfahren**:

  Trotz der Reduzierung der Bildausschnitte durch das Region Proposal, ist die Rechenzeit für ein Bild immer noch erhöht, da der Klassifikator mehrere Ausschnitte betrachten muss. Einen anderen Ansatz verfolgen einstufige Verfahren.

Beispielsweise werden in YOLO die Bilder in Zellen unterteilt. Für jede Zelle wird eine feste Anzahl von Bounding Boxen vorhergesagt. Im Gegensatz zur Klassifikation von übersegmentierten Bildausschnitten, ist dies also ein Regressionsproblem, in dem das komplette Bild nur einmal betrachtet werden muss. Dies spiegelt sich in einer starken Reduzierung der Inferenzlaufzeit wider [RDGF15].

3.3 SEMANTISCHE SEGMENTIERUNG

Für Anwendungen, wie z.B. autonomes Fahren, ist das Verstehen der kompletten Szene eines Bildes notwendig, womit eine Objektdetektion nicht präzise genug ist. Jeder Pixel eines Bildes muss somit interpretierbar sein. In der semantischen Segmentierung ist das Ziel eine Klassifikation auf Pixelebene durchzuführen, welches einer der Hauptaspekte für das Szenenverständnis ist. Diese Aufgabe ist unter anderem sehr schwierig und weiterhin ungelöst, da gegenüber der Klassifikation auf Bildebene eine höhere Anforderung an das Modell gestellt wird [GOO+17]. Jedoch schaffen es aktuelle Ansätze auf Basis von tiefen CNNs erwähnenswerte Ergebnisse zu erreichen. Hierfür werden Fully Convolutional Networks verwendet, welche nur aus Faltungs- und eventuell Poolingschichten bestehen. Die Ausgabe wird nicht mithilfe eines MLP berechnet, sondern durch eine Feature Map modelliert. Diese besitzt dieselbe Auflösung wie das Eingabebild, sodass jeder Punkt der Feature Map die Klassifikationsentscheidung bezüglich des Pixels im Eingabebild darstellt.

Saito et al. stellen eine Methode für das Erfassen von 3-dimensionalen Gesichtsmodellen vor, in dem ein entscheidender Schritt die Gesichtssegmentierung darstellt [SLL16]. Bei der Gesichtssegmentierung werden zwei Klassen betrachtet. Ein Pixel gehört entweder zu einem Gesicht bzw. zum Hintergrund. Dafür verwenden die Autoren eine Encoder-Decoder Netzarchitektur. Zu Beginn folgt eine Sequenz von Faltungs- und Poolingschichten, welche die Auflösung der Feature Maps reduziert (Encoder). Dieser Schritt ist notwendig um das effektive rezeptive Feld zu vergrößern (s. Abbildung 2.1.3). Da jedoch eine Klassifikation auf Pixelebene gesucht ist, muss die Auflösung der folgenden Feature Maps wieder vergrößert werden. Dafür werden Schichten verwendet, welche eine transposed convolution berechnen. Diese entsprechen einem lernbaren Upsampling (s. Unterabschnitt 2.1.3).

Annotationen für semantische Segmentierung zu sammeln ist ein aufwändiger Prozess. Alle Pixel innerhalb eines Bildes müssen mühselig annotiert werden. Aus diesem Grund fokussieren sich Arbeiten auch auf Fälle in denen nur schwach annotierte Daten vorhanden sind. Beispielsweise wurde in [PCMY15] eine Annotation auf Bildebene verwendet, während innerhalb von [RBFL15] eine Punktannotation

verwendet wird. Bei einer Punktanotation sind die einzelnen Objekte nicht auf den Pixel genau annotiert sind. Stattdessen sind die einzelnen Objekte durch einen Punkt im Zentrum des Objektes markiert. Die Autoren zeigen, dass bei starker Reduzierung des Annotationsaufwandes noch akzeptable Ergebnisse möglich sind. Für die Datensätze, welche in dieser Arbeit betrachtet werden, sind keine Pixelannotationen für die Gesichtssegmentierung vorhanden. In dieser Arbeit wird die Annotation für die Segmentierung aus der vorhandenen Annotation synthetisiert. Dies kann als eine Form des schwachen überwachten Lernens interpretiert werden.

3.4 ATTRIBUTERKENNUNG

Gesichtsattribute bezeichnen bestimmte Eigenschaften, welche ein Gesicht aufweist. Sie können von visuellen Attributen (z.B. Haarfarbe, Hutträger, etc.) bis zu Informationen über die Mimik (Lachen) reichen. Ein akkurates Erkennen dieser ist somit ein wesentlicher Bestandteil für das automatische Verstehen von Gesichtern. Es ist klar, dass diese zusätzlichen Informationen bei weiteren Aufgaben, wie z.B. Erkennung, Identifikation oder auch Verifikation, behilflich sein können [ZGH+18]. Aktuelle Methoden basieren auf CNNs, welche die Fähigkeit ausdrucksstarke Merkmale zu lernen ausnutzen. Das Erkennen von Attributen ist ein Fall von Klassifikation, wobei jedoch mehrere Klassen für ein Gesicht zutreffen können.

Liu et al. [LLWT14] verwenden ein Verfahren, welches sich in drei Teile unterteilen lässt. Jedes der drei Teile besteht aus einem eigenen Modell, welches separat trainiert werden muss. Der erste Schritt besteht aus dem Lokalisieren des Gesichtes innerhalb des Bildes. Dafür wird ein CNN verwendet, welches jedoch nur mithilfe von Attributen auf Bildebene trainiert wird. Das dazugehörige MLP wird zur Auswertung verworfen. Ohne Bounding Box Annotation erreicht das Modell akkurate Lokalisierungen, indem die letzten Feature Maps betrachtet werden. Diese besitzen an den Stellen an der ein Gesicht vorkommt hohe Aktivierungen.

Auf Basis dieser gefundenen Bildausschnitte werden im nächsten Schritt mithilfe eines weiteren CNNs diskriminative Merkmale gelernt. Dieses CNN wird ähnlich wie oben mit Attributannotationen trainiert, wobei jedoch nur die Ausgabeschicht des MLPs verworfen wird. Die verbliebene Schicht des MLPs berechnet eine Merkmalsrepräsentation, welches als Eingabe für den 3. Teil dient. Die hier berechneten Merkmalsvektoren werden verwendet um für jedes Attribut eine eigene Support Vector Machine (SVM) zu lernen. Eine SVM ist ein linearer Klassifikator, welcher ein binäres Klassifikationsproblem löst. Hierbei wird eine Trennfunktion in Form einer Hyperebene gelernt, welche die beiden Klassen trennt, sodass der Abstand zwischen

den Trainingsbeispielen der zwei Klassen zu dieser maximiert wird (vgl. [HTF01] Kapitel 12).

Die Ausgabemodellierung über einen externen Klassifikator zu realisieren hat den Nachteil, dass diese separat voneinander optimiert werden müssen. Insbesondere führt dies zu einer längeren Trainingszeit. Solche mehrstufigen Ansätze werden heutzutage kaum noch benutzt. Stattdessen werden Netze Ende-zu-Ende trainiert, wobei die Vorhersage durch die Ausgabeschicht des Netzes modelliert wird. Dies hat zur Folge, dass gleichzeitig Merkmalsextraktion und Klassifikation/Vorhersage optimiert werden.

Günther et al. verwenden statt der Attribut SVMs ein tiefes Netz, welches Ende-zu-Ende trainiert wird [GRB16]. In dieser wird eine aktuelle, erfolgreiche und oft verwendete Architektur verwendet, das ResNet (s. Abschnitt 4.2). Die Ausgabe wird durch eine vollverbundene Schicht modelliert, welche ein Ausgabeneuron für jedes Attribut verwendet.

3.5 MULTI TASK LEARNING

Der folgende Abschnitt behandelt Arbeiten aus dem Bereich des MTL mit starkem Bezug zur Methodik dieser Arbeit. Eine der ersten Arbeiten, welches ein neuronales Netz für das MTL betrachtet, ist die Arbeit von Caruana [Car93]. In dieser wurde ein MLP mit einer verborgenen Schicht verwendet, welche für alle Aufgaben geteilt wurde. Innerhalb dieser lernt das MLP eine geteilte Repräsentation der Daten, welche auf mehrere Tasks generalisieren kann. Der hierdurch eingeführte induktive Bias erlaubt es dem Netz zusätzliche Informationen aus den Trainingssignalen verwandter Aufgaben zu benutzen (s. Abschnitt 2.2).

Die Idee des MTLs findet sich auch in aktuellen Arbeiten im Bereich der Gesichtsanalyse wieder, welche jedoch eher ein CNN als Grundlage verwenden. Zhang et al. [ZLLT14a] stellen ein MTL Modell für die Landmarkendetektion vor, welches zusätzlich unterschiedliche Attribute betrachtet. Diese beinhalten das Lachen, das Tragen einer Brille, das Geschlecht und die Pose. Es ist offensichtlich, dass Informationen aus den Attributen die Ergebnisse der Landmarkenerkennung positiv beeinflussen können. Beispielsweise hat das Attribut Lachen einen Einfluss auf die Landmarken im Mundbereich, während die Pose das Verhältnis der Landmarken untereinander bestimmt. In deren Arbeit wurde ein CNN benutzt (s. Abbildung 3.5.1), wobei alle vier Faltungsschichten und eine vollverbundene Schicht zwischen den Aufgaben geteilt wurden. Der hieraus resultierende Merkmalsvektor ist derselbe für jede Aufgabe. Darauf folgt für jede Task eine weitere vollverbundene Schicht, welche als jeweilige Ausgabeschicht dient. Es wird eine strikte Unterteilung zwischen der Hauptaufgabe

(Landmarken) und den Hilfsaufgaben (Attribute) durchgeführt. Das Model wird mit folgendem MTL Loss trainiert

$$L_{gesamt} = L_{haupt} + \sum \lambda_i L_i, \tag{3.5.1}$$

wobei L_{haupt} der Kostenfunktion für die Landmarkenregression und L_i der einzelnen Attribute entspricht. λ_i stellt dabei die Gewichtung der Hilfsaufgaben dar. Statt diese vorher heuristisch festzulegen können diese mitgelernt werden. Ein Problem dabei sind die unterschiedlichen Lernverhalten während der Optimierung. Da die Aufgaben unterschiedlich komplex sind, benötigen diese auch unterschiedlich lange zur Konvergenz. Um ein Overfitting der einfacheren Aufgaben zu vermeiden, wird ein sogenanntes early stopping für die einzelnen Hilfsaufgaben verwendet. Hierbei wird das Training einer Aufgabe i abgebrochen, falls ein bestimmtes Kriterium dafür erreicht ist. Dieses ist gegeben mit:

$$\frac{k \cdot med^t_{j=t-k}(E^i_{train}(j))}{\sum^t_{j=t-k} E^i_{train}(j) - k \cdot med^t_{j=t-k}(E^i_{train}(j))} \frac{E^i_{val}(t) - \min_{j=1\dots t}(E^i_{train}(j))}{\lambda^i \min_{j=1\dots t}(E^i_{train}(j))} > \epsilon. \tag{3.5.2}$$

$E^i_{train}(t)$ und $E^i_{val}(t)$ bezeichnen die Trainings- bzw. die Validierungskosten von Aufgabe i zur aktuellen Trainingsiteration t und k einen Hyperparameter, welcher bestimmt wie viele vorangegangene Trainingsiterationen zur Evaluierung des Kriteriums betrachtet werden. Ist das Kriterium für eine Aufgabe größer als ein manuell gewählter Schwellwert ϵ, so wird das Training für diese abgebrochen. Der erste Term misst die relative Verbesserung des Trainingsfehlers, während der zweite Term das Verhältnis des Validierungsfehlers zum Trainingsfehler modelliert. In der Arbeit wurde gezeigt, dass das Miteinbeziehen zusätzlicher Attribute eine Verbesserung der Landmarkenerkennung zur Folge hat, wobei jedoch kein Einblick in die Qualität der Attributinferenz gegeben wird. In der Methode wird zwar das Wählen der λ_i vermieden, jedoch müssen k und ϵ passend gewählt werden.

Dieselben Autoren liefern mit [ZLLT14b] eine Erweiterung des Ansatzes. Es werden 22 Attribute betrachtet statt der vorherigen vier. Zusätzlich wird die Korrelation zwischen den Tasks explizit durch eine lernbare Kovarianzmatrix modelliert. Im Weiteren wird angenommen, dass die λ_i Normalverteilungen folgen mit festen Varianzen von 1, wobei die Mittelwerte mitoptimiert werden. Dies führt zu einem komplexen 3-stufigen Optimierungsverfahren, in dem die a-Posteriori Wahrscheinlichkeit der Parametergruppen gegeben der Eingabe, der Annotation und der Kosten sequentiell optimiert werden.

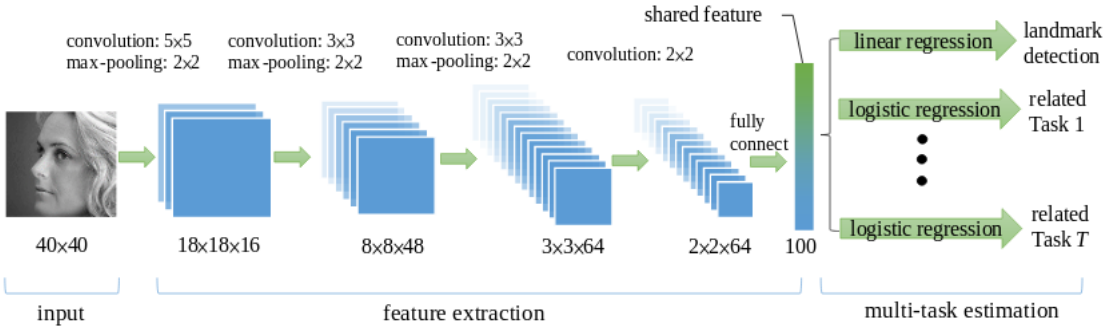


Abbildung 3.5.1: MTL Architektur für die Vorhersage von Landmarken und Gesichtsattributen. Zu Beginn folgt eine Reihe von geteilten Schichten, worauf für jede Aufgabe eine eigene vollverbundene Schicht verwendet wird. Abbildung entnommen von [ZLLT14a].

[RPC16] ist ein aktuelles Verfahren, welches innerhalb eines Bildes Gesichter detektiert und das jeweilige Geschlecht, die Pose und die Position der Landmarken bestimmt. Die Pose ist dabei durch die Abweichung der drei Achsen (Roll-Nick-Gier-Winkel) relativ zur Kamera gegeben. In der Arbeit wird keine Annahme über die Anzahl an Gesichtern je Bild getroffen. Um Bilder mit mehreren Personen zu behandeln wird ein Region Proposal Ansatz verwendet. Hierfür wird der Selective Search Algorithmus [USGS13] angewendet, welcher bis zu 2000 Bildbereiche je Bild vorschlägt [RPC16] und somit zu einer erheblichen Erhöhung der Inferenzlaufzeit führt. Für die Gesichtsdetektion ist somit nur ein Klassifikator notwendig, welcher für einen Bildausschnitt entscheidet ob dieser ein Gesicht beinhaltet. Die grundlegende Idee der verwendeten Architektur wird auch in dieser Arbeit verwendet, welche in Abschnitt 4.2 näher beschrieben wird. Als Gesamkostenfunktion wird eine naiv gewichtete Summe der einzelnen Kostenfunktionen benutzt. Dies sind Hyperparameter, welche entsprechend der subjektiven Einschätzung der Schwierigkeit einer Aufgabe eingestellt werden müssen.

Das manuelle Wählen der Taskgewichte ist nicht trivial, jedoch kann eine ungeeignete Gewichtung zu suboptimalen Ergebnissen führen. Kendall et al. stellen ein Verfahren für das Lernen eines MTL Modells für die Szenenanalyse vor [KGC17], welches ohne eine heuristische Gewichtung auskommt. In dem Modell wird die semantische Segmentierung, Instanzsegmentierung und Tiefenregression kombiniert. Hierfür wird ein probabilistischer Ansatz zur Modellierung der Gesamtkostenfunktion gewählt. Dabei wird explizit für jede Aufgabe ein Koeffizient zur Gewichtung seiner

Kosten mitgelernt, welcher als eine Form der homoskedastischen Unsicherheit interpretierbar ist. Dadurch kann implizit die unterschiedliche Skalierung der einzelnen Kostenfunktionen behandelt werden. Die Idee des Ansatzes ist Bestandteil der Methodik dieser Arbeit. Aus diesem Grund wird die Gesamtkostenfunktion und deren Herleitung in Abschnitt 4.1 näher erläutert.

Obwohl die Formulierung dessen relativ aktuell ist, haben schon einige Arbeiten die lernbare Unsicherheitsgewichtung miteinbezogen. Bischke et al. betrachten die semantische Segmentierung von Gebäuden in hochauflösenden Satellitenbildern [BHF+17]. Ein Problem hierbei sind die Gebäudegrenzen für die es schwierig ist pixelgenaue Vorhersagen zu treffen. Um dies zu behandeln wird ein weiterer Task eingeführt, welcher einen größeren Fokus auf Gebäudegrenzen legt. Als Gesamtkostenfunktion wird die oben genannte Unsicherheitsgewichtung benutzt. Jedoch ist nur die Annotation für die semantische Segmentierung vorhanden. Bei dem zweiten Task handelt es sich um eine synthetisch generierte Hilfsaufgabe, welche aus der vorhandenen Annotation erstellt wird. Dafür wird Distance Transform durchgeführt, welche den Abstand eines Pixels zur Objektgrenze ermittelt. Dies gibt dem Modell zusätzlichen Anreiz präzisere Vorhersagen für die Pixel an den Gebäudegrenzen zu treffen. Da das Annotieren für die Gesichtssegmentierung ein zeitintensiver Prozess und insbesondere in den hier betrachteten Datensätzen nicht vorhanden ist, wird ähnlich wie in [BHF+17] eine künstliche Annotation dafür erzeugt.

In [LK18] wird die Unsicherheitsgewichtung für die Szenenanalyse von Straßenbildern verwendet. Hierbei werden Tiefenregression und semantische Segmentierung betrachtet. Dabei werden zusätzlich die Wetterbedingung und die aktuelle Uhrzeit vorhergesagt. Zwar haben diese Aufgaben keinen direkt Einfluss auf die oberen Aufgaben, da diese jedoch einen Effekt auf die Beleuchtung haben, können diese möglicherweise die Leistung in den anderen Aufgaben verbessern. Tiefe CNNs benötigen eine große Anzahl an Trainingsdaten. Für das MTL Szenario sind Annotationen für die weiteren Aufgaben nötig, welche bei aktuellen Datensätzen oft nicht ausreichend verfügbar sind. Diese im Nachhinein zu bestimmen ist des Weiteren nicht immer möglich. Aus diesem Grund werden die Bilder über eine Simulation generiert, in der die Szenen nach Belieben konfiguriert werden können. Diese Szenenparameter können direkt als Annotation für verschiedene Aufgaben entnommen werden. Bei der Simulation handelt es sich um Material aus dem Videospiel GTA V, welches nahezu fotorealistische Szenen rendert. Im Weiteren wird eine wesentliche Erweiterung der Gesamtkostenfunktion von Kendall et al. [KGC17] eingeführt, welche mitunter einen wichtigen Randfall während der Optimierung abdeckt (s. Abschnitt 4.1).

4 METHODIK

Im vorherigen Kapitel wurden die unterschiedliche Aufgaben innerhalb der Gesichtsanalyse vorgestellt, welche in dieser Arbeit betrachtet werden. Zusätzlich wurden Arbeiten behandelt, welche das MTL im Bereich der Gesichtsanalyse anwenden. Diese haben jedoch den Nachteil, dass die Optimierung dieser komplex ist oder eine manuelle Gewichtung der einzelnen Aufgaben verwenden. In dieser Arbeit wird ein aktuelles Verfahren verwendet, welches das automatische Lernen der Taskunsicherheiten zur Gewichtung der einzelnen Kostenfunktionen heranzieht [KGC17]. Nach bestem Wissen und Gewissen ist diese Arbeit eine der ersten, welche die Unsicherheitsgewichtung aus [KGC17] im Bereich der Gesichtsanalyse anwendet.

Im folgenden Kapitel wird die Methodik präsentiert, die in dieser Arbeit realisiert wurde. Dafür wird zu Beginn in Abschnitt 4.1 die Unsicherheitsgewichtung der Gesamtkostenfunktion erläutert, welche für das Training des Modells benutzt wird. In Abschnitt 4.2 wird die verwendete Modellarchitektur beschrieben. Es handelt sich hierbei um ein tiefes CNN, in dem vordere und hintere Schichten verknüpft werden, um lokale Informationen und semantisch abstrakte Merkmale zu kombinieren.

4.1 TRAINING DES MTL MODELLS

Damit das Netz für alle Aufgaben adäquate Vorhersagen treffen kann, müssen geeignete Gewichte für das Netz gefunden werden. Diese werden in einer vorherigen Trainingsphase optimiert, indem eine Kostenfunktion minimiert wird (s. Unterabschnitt 2.1.6). Einerseits muss eine geteilte Merkmalsrepräsentation, welche auf alle Aufgaben generalisiert, als auch diskriminative Berechnungszweige für die einzelnen Aufgaben gelernt werden. Dafür wird eine Gesamtkostenfunktion verwendet, welche den Trainingsfortschritt aller betrachteten Aufgaben misst. Da eine heuristische Gewichtung dieser unvorteilhaft ist (s. Abschnitt 3.5), wird hierfür der in [KGC17] vorgestellte Ansatz gewählt, welcher die gelernten Unsicherheiten des Modells bezüglich der unterschiedlichen Aufgaben verwendet.

4.1.1 *Gesamtkostenfunktion unter Verwendung von Unsicherheiten*

Eine aktuell oft verwendete Ansicht mit der Formen von Unsicherheiten charakterisiert werden, ist das Unterscheiden zwischen epistemischer und aleatorischer Unsicherheit. Innerhalb der epistemischen wird die Unsicherheit des Modells bezüglich der Verteilung der Daten beschrieben, welche aufgrund von Datenmangel entsteht. In der Regel wird dies durch Verteilungen über die Modellparameter realisiert [KG17]. Bei der aleatorischen Unsicherheit handelt es sich um Unsicherheit, welche durch Rauschen innerhalb der beobachteten Daten entsteht, das ohne weiteres Wissen über diese nicht erklärt werden kann. Ein Beispiel dafür wären Bilder in denen Teile eines Objektes verdeckt sind. Weiterhin kann die aleatorische Unsicherheit in zwei Arten unterteilt werden. Die heteroskedastische Unsicherheit ist abhängig von der Eingabe und wird durch die Modellausgabe ausgedrückt, z.B. indem die Varianz dieser zusätzlich ausgegeben wird. Im Gegensatz dazu ist die homoskedastische Unsicherheit unabhängig von der Eingabe. Es handelt sich hierbei um eine konstante Größe, welche für alle Eingaben gleich bleibt. Diese unterscheidet sich jedoch für verschiedene Aufgaben und kann somit als Unsicherheit des Models bezüglich einer bestimmten Aufgabe angesehen werden. Im Folgenden wird gezeigt, wie sich diese über eine probabilistische Herleitung in eine Gesamtkostenfunktion integrieren lassen und automatisch mitgelernt werden können.

Herleitung

Sei $f^{\mathbf{W}}(\mathbf{x})$ die Ausgabe des Netzes, wobei $\mathbf{W}$ der Gewichte des Netzes und $\mathbf{x}$ der Eingabe entsprechen. Zusätzlich modellieren die Zufallsvariablen y_i die betrachteten Aufgaben i. So kann ein probabilistisches Modell mit folgender MTL Likelihood aufgestellt werden:

$$p(y_1, y_2, ..., y_k | f^{\mathbf{W}}(\mathbf{x})) = \prod_i^k (p(y_i | f^{\mathbf{W}}(\mathbf{x})). \tag{4.1.1}$$

Dabei wird die vereinfachende Annahme getroffen, dass die y_i stochastisch unabhängig voneinander sind, sodass die gemeinsame Likelihood als Produkt der einzelnen Likelihood Funktionen der jeweiligen Aufgaben ausgedrückt werden kann. Im Fall von Regression wird die Likelihood üblicherweise mit einer Gaussverteilung modelliert. Es ergibt sich

$$p(y_i | f^{\mathbf{W}}(\mathbf{x})) = \mathcal{N}(f^{\mathbf{W}}(\mathbf{x}), \sigma_i^2), \tag{4.1.2}$$

wobei σ_i^2 die Varianz von y_i ist und somit der homoskedastischen Unsicherheit einer Aufgabe i entspricht. Bei einem Klassifikationsproblem wird die Likelihood über eine Softmax[1] Funktion dargestellt. Sei z ein Vektor, welcher für jede Klasse einen Eintrag besitzt, so ist die Softmax Funktion definiert als $\rho_i(z) := \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_k)}$. Die hieraus resultierenden Werte sind normalisiert und werden als Pseudowahrscheinlichkeiten gesehen, da diese positiv sind und die Summe dieser 1 ergibt. Indem ein zusätzlicher positiver Wert σ_i^2 eingeführt wird mit der die Netzausgabe skaliert wird, ergibt sich damit eine sogenannte Boltzmann Verteilung mit:

$$p(y|f^{\mathbf{W}}(\mathbf{x})) = \rho(\frac{1}{\sigma^2} f^{\mathbf{W}}(\mathbf{x})), \tag{4.1.3}$$

wobei σ_i^2 die Unsicherheit der Verteilung von y_i widerspiegelt. Dies lässt sich somit als homoskedastische Unsicherheit interpretieren.

Für die Optimierung des Modells werden die Modellparameter $\mathbf{W}$ und die Unsicherheitsparameter σ_i^2 angepasst, sodass die Likelihood maximiert wird (s. Gleichung 4.1.1). Es wird also diejenige Parameterkonfiguration gewählt, unter der das Modell die beobachteten Daten am ehesten erzeugt hätte. Bei einer Maximum Likelihood Schätzung wird des öfteren die logarithmierte Likelihood betrachtet, welche aufgrund der Faktorisierung umformuliert werden kann zu:

$$\log \prod_i^k (p(y_i|f^{\mathbf{W}}(\mathbf{x})) = \sum_i^k \log p(y_i|f^{\mathbf{W}}(\mathbf{x})). \tag{4.1.4}$$

Diese kann somit als Summe der aufgabenspezifischen logarithmierten Likelihood Funktionen berechnet werden. Anstelle diese zu maximieren wird die negative logarithmierte Likelihood minimiert, womit sich folgende Gesamtkostenfunktion ergibt

$$-\log \prod_i^k (p(y_i|f^{\mathbf{W}}(\mathbf{x})) = \sum_i^k -\log p(y_i|f^{\mathbf{W}}(\mathbf{x})). \tag{4.1.5}$$

Bei einer Regression von y_i bei angenommener Normalverteilung kann die negative logarithmierte Likelihood mit

$$-\log p(y_i|f^{\mathbf{W}}(\mathbf{x})) \propto \frac{1}{2\sigma_i^2} L_i(\mathbf{W}) + \log \sigma_i \tag{4.1.6}$$

1 Im Zweiklassenfall reduziert sich diese auf die Sigmoid Funktion.

approximiert werden, wobei L_i den Fehler zwischen der Modellausgabe und der Annotation für diese Aufgabe misst. Dafür ergibt sich der L2 Loss, welcher gegeben ist mit $L_i = \|y_i - f^{\mathbf{W}}(\mathbf{x})\|_2^2$.

Ist y_i diskret, so handelt es sich um Klassifikation, in der die logarithmierte Likelihood mit

$$-\log p(y_i|f^{\mathbf{W}}(\mathbf{x})) \propto \frac{1}{\sigma_i^2} L_i + \log \sigma_i \tag{4.1.7}$$

approximiert werden kann. In diesem Fall wird die Kreuzentropie als Kostenfunktion verwendet, welche definiert ist als

$$L_i = -\sum_k y_k \log(\hat{y}_k), \tag{4.1.8}$$

wobei k den Klassenindex bezeichnet und $\hat{y}_k$ die vom Modell vorhergesagte Pseudowahrscheinlichkeit für die $k-$te Klasse entspricht.

Die Gesamtkostenfunktion ist differenzierbar, da sie nach Gleichung 4.1.5 eine Summe von differenzierbaren Termen ist. Somit können sowohl die Netzparameter $\mathbf{W}$ als auch die Unsicherheiten σ_i gleichzeitig mit einem Gradientenabstieg gelernt werden (s. Unterabschnitt 2.1.6).

In beiden Fällen dienen die σ_i dabei zur Gewichtung der Fehler L_i (s. Gleichung 4.1.6 und Gleichung 4.1.7). Bei einer niedrigen Unsicherheit in einer Aufgabe nimmt der Fehler in dieser einen größeren Einfluss auf das Training ein. Im Gegensatz dazu führt eine hohe Unsicherheit dazu, dass der Fehler einer Aufgabe weniger stark gewichtet wird. Es wird also das Verhältnis der Kostenfunktionen der verschiedenen Aufgaben während des Trainings dynamisch angepasst. Ein weiterer Vorteil ist dabei, dass zusätzlich die Skalierung der einzelnen Kostenfunktionen behandelt wird.

Eine Möglichkeit den Loss beliebig zu verringern ist das triviale Vergrößern von σ_i, welches jedoch dazu führt, dass der Taskloss verschwindend klein wird. Dadurch würde das Lernen der Netzgewichte vernachlässigt werden, womit effektiv kein Training des Modells stattfindet. Dies wird durch den hinteren Term $\log(\sigma_i)$ verhindert. Er dient als Regularisierung, welcher zu hohe Werte für σ_i bestraft.

Das Vorhersagen von σ_i kann zu numerischen Problemen führen. Um eine Division durch 0 zu vermeiden, wird stattdessen $s = \log \sigma^2$ vorhergesagt. Ist das Modell sehr sicher in einer Aufgabe, sodass $\sigma_i << 1$ gilt und zusätzlich somit der Fehler L_i klein ist, führt der Regularisierungsterm dazu, dass dieser Teilterm der Gesamtkostenfunktion negativ wird. In den ersten Experimenten ist der Fall aufgetreten, dass dadurch die Gesamtkostenfunktion negativ werden kann, welches ein unerwünschtes Verhalten bei der Optimierung ist. Aus diesem Grund wird ein angepasster Regularisierungsterm

benutzt, welcher von Liebel und Körner verwendet wurde [LK18]. Hierfür wird der Regularisierungsterm verändert, sodass dieser nicht negativ werden kann. Durch eine Verschiebung des Logarithmuses mit $\log(\sigma_i^2 + 1) \geqslant 0$ ist garantiert, dass dieser positiv ist.

4.1.2 *Wing Loss*

Der Wing Loss ist eine aktuelle Kostenfunktion, welche für die Landmarkendetektion eingesetzt werden kann (s. Abschnitt 3.1). Zwar hat der L2 Loss einen theoretischen Hintergrund, dieser ist jedoch aufgrund des Quadrierens des Fehlers anfällig gegenüber Ausreißern. Zusätzlich werden Trainingsbeispiele mit einem kleineren Fehler eher vernachlässigt.

Der Wing Loss ist definiert als:

$$L_{wing}(y, \hat{y}) := \begin{cases} w \ln(1 + \frac{||y-\hat{y}||}{\epsilon}), & ||y - \hat{y}|| < w \\ ||y - \hat{y}|| - (w - w \ln(1 + \frac{w}{\epsilon})), & \text{sonst.} \end{cases}$$

Die Kostenfunktion besteht aus zwei wesentlichen Teilen. In dem Fall, wo der absolute Fehler kleiner als ein Hyperparameter w ist, wird eine nichtlineare Funktion verwendet, womit der Gradient in diesem Fall verstärkt wird. Ist der absolute Fehler groß, wird dieser gedämpft, indem eine lineare statt einer quadratischen Funktion verwendet wird. Der hintere Term verbindet den nichtlinearen und linearen Teil der Kostenfunktion, sodass die Funktion stetig und somit differenzierbar ist. Der Hyperparameter ϵ bestimmt dabei die Krümmung des nichtlinearen Teiles.

Der Wing Loss bezieht seinen Namen aus seiner Form. Abbildung 4.1.1 zeigt diesen exemplarisch für verschiedene Hyperparameter, wobei die Flügelform klar erkennbar ist. Für den Wing Loss werden die Hyperparameter $w = 10$ und $\epsilon = 2$ gewählt, welche sich als geeignet herausgestellt haben [FKA+17].

4.2 MODELLARCHITEKTUR

Die verwendete Architektur basiert auf der Arbeit von [RPC16], welche ein sogenanntes ResNet als Basis besitzt. Innerhalb dieser wird das Paradigma des Hard Parameter Sharings für das MTL verwendet (s. Abschnitt 2.2). Das Netz lässt sich in geteilte Schichten, welche zur Merkmalsberechnung dienen (s. Unterabschnitt 4.2.2) und in taskspezifische Modellteile (Unterabschnitt 4.2.3) aufteilen. Bevor die Architektur

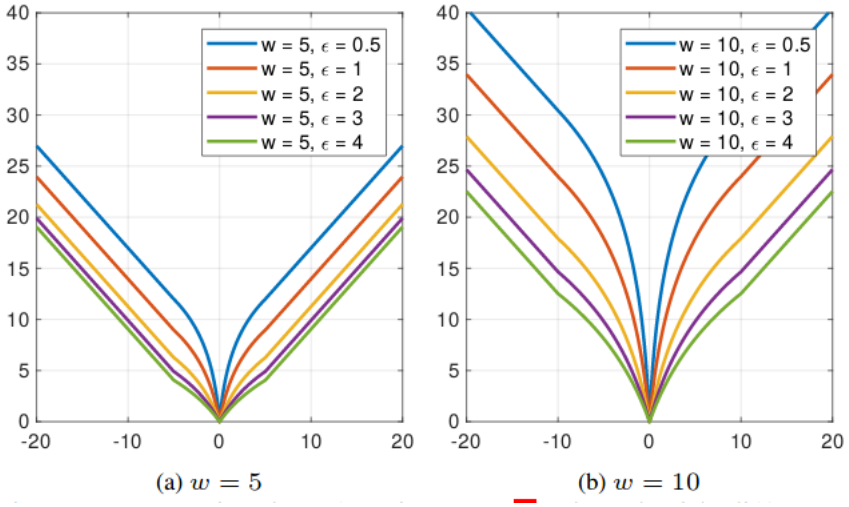


Abbildung 4.1.1: Wing Loss für unterschiedliche Hyperparameterkonfigurationen. Abbildung entnommen aus [FKA+17].

im Ganzen erläutert werden kann, wird im Folgenden das Konzept des ResNets vorgestellt.

4.2.1 *ResNet*

Das ResNet ist eine tiefe CNN Architektur, welche für die Bildanalyse vorgestellt wurde [HZRS15a]. Seit der Einführung hat sich diese zu einem bewährten und aktuell oft eingesetzten Modell entwickelt. In dieser wird ein akutes Problem innerhalb des Lernens von tiefen Netzen behandelt. Die Tiefe des Netzes ist entscheidend um diskriminative und repräsentative Merkmale zu lernen. Das willkürliche Aneinanderreihen von vielen Faltungsschichten kann jedoch zu schlechteren Ergebnissen führen. Angenommen es gibt ein ausreichend gutes Netz mit einer Tiefe n, so dürften zusätzliche Faltungsschichten die Ergebnisse nicht negativ beeinflussen. Während des Trainings sollte das Netz zumindest in der Lage sein innerhalb der überflüssigen Schicht die Identitätsfunktion zu lernen, um dieselben Modellergebnisse zu erreichen. Dies ist im Allgemeinen nicht der Fall, welches sich auf Faktoren der nichtkonvexen Optimierung von neuronalen Netzen zurückführen lässt und somit schwerer zu optimieren sind.

Um dies zu umgehen wird das Konzept des Residual Lernens eingeführt. Dafür wird die ursprüngliche Abbildung einer Folge von Faltungsschichten umformuliert.

Sei x die Eingabe der betrachteten Faltungsschichten und $h(x)$ die Feature Maps, welche innerhalb dieser berechnet werden, so ergibt sich für die neue Abbildung $F(x)$:

$$F(x) = h(x) + x. \tag{4.2.1}$$

Innerhalb der Schichten lernt das Netz die Residuen $h(x)$, worauf die Eingabe x aufaddiert wird. Diese sogenannte Residual Abbildung erlaubt es dem Netz die Eingabe direkt weiterzuleiten, falls die Schichten für das Modell nicht von Bedeutung ist. Das Netz muss somit nicht mehr die Identitätsfunktion lernen, stattdessen reicht es die Residuen $h(x)$ gegen 0 laufen zu lassen, welches einfacher zu optimieren ist.

Dies wird in Form von sogenannten Residual Blöcken realisiert, welche aus mehreren Faltungsschichten bestehen. Abbildung 4.2.1 visualisiert exemplarisch einen Block. In einem Feed Forward Netz lässt sich die Residual Abbildung durch eine zusätzliche direkte Verbindung zwischen Eingabe und Ausgabe eines Blockes herstellen. Diese leitet die Eingabe x unverändert an die Ausgabe des Blockes, wo diese auf die innerhalb des Blockes berechneten Feature Maps $h(x)$ addiert werden.

Das ResNet existiert in verschiedenen Varianten, welche sich hauptsächlich in der Anzahl an lernbaren Schichten unterscheiden (ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152). In dieser Arbeit wird das ResNet-18 mit 18 und das ResNet-101 mit 101 lernbaren Schichten betrachtet. Der generelle Aufbau der Architekturen ist in Tabelle 4.2.1 veranschaulicht.

Das ResNet-18 verwendet in den Blöcken zwei aufeinanderfolgende Faltungsschichten mit jeweils einer Kernelgröße von 3 (Abbildung 4.2.1 links). Im rechten Teil der Abbildung sind die Blöcke dargestellt, welche von den noch tieferen Netzen, wie z.B. ResNet-101, verwendet werden. Dies sind sogenannte Bottleneck Blöcke, welche nötig sind um die Rechenkomplexität gering zu halten. Dafür wird eine Faltungsschicht mit einer Fenstergröße von 1×1 benutzt um die Anzahl an Feature Maps zu reduzieren. Da eine Faltung über alle eingehenden Feature Maps berechnet wird, kann durch die Wahl der Filteranzahl die Dimensionsreduktion gesteuert werden (s. Unterabschnitt 2.1.3). Darauf folgt eine Faltungsschicht mit einem 3×3 Kernel. Würde diese auf der ursprünglichen Dimension ausgeführt werden, wäre die Laufzeitkomplexität stark erhöht. Am Ende eines Blockes werden 1×1 Faltungen angewendet um die Anzahl an Feature Maps wieder zu erhöhen.

Beide Architekturen sind grundsätzlich ähnlich aufgebaut, wobei der wesentliche Unterschied in der Anzahl und Struktur der verwendeten Blöcke (s.o.) und der Filteranzahl liegt. Aufgrund dessen wird sich die folgende Beschreibung an der Architektur des ResNet-101 orientieren. Bei der Eingabe handelt es sich um 224×224 RGB Bilder. Zu Beginn wird eine 7×7 Faltungsschicht mit 64 Filtern und einer Schrittweite von 2 verwendet. Dies sorgt dafür, dass die Auflösung der berechneten Feature Maps

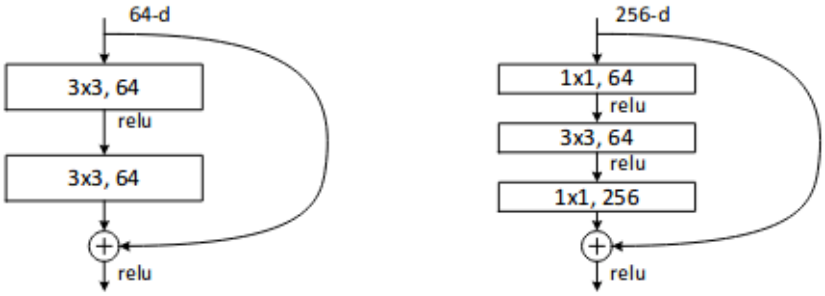


Abbildung 4.2.1: Residual Blöcke werden durch eine direkte Verbindung zwischen Eingabe und Ausgabe eines Blockes realisiert. Für die sehr tiefen Versionen des ResNets werden Bottleneck Blöcke verwendet (rechts). Abbildung entnommen aus [HZRS15a].

halbiert wird. Die großen Kernel 7×7 sind am Anfang des Netzes noch praktikabel, da die Eingabe drei Kanäle (RGB) besitzt. Darauf folgt ein Max Pooling, womit die Auflösung ein weiteres Mal halbiert wird. Diese frühe Verkleinerung der Auflösung sorgt dafür, dass das ResNet effizient anwendbar ist. Die Berechnungslaufzeit des ResNets ist vergleichbar mit der beliebten VGG Architektur, obwohl dieses um einiges tiefer ist. Nach der Pooling Schicht folgt der Kern der Architektur. Dieser besteht aus vier Sequenzen der oben erwähnten Bottleneck Blöcken. Die erste Sequenz besteht aus drei Blöcken, welche jeweils $64, 64, 256$ Filter verwenden. In der nächsten Blockfolge werden vier Blöcke mit jeweils $128, 128, 512$ Filtern angewendet, wobei die allererste Schicht ein Downsampling durch eine Schrittweite von 2 realisiert. Es ergeben sich somit Feature Maps mit einer Auflösung von 28×28. Die 3. Folge von Blöcken beinhaltet 23 Blöcke. Auch hier wird wieder in der ersten Schicht ein Downsampling wie oben erreicht, wodurch eine Auflösung von 14×14 entsteht. Die Blöcke enthalten jeweils $256, 256, 1024$ Filter. Der letzte Teil besteht aus drei weiteren Blöcken mit jeweils $512, 512, 2048$ Filtern, wobei die erste Schicht ein weiteres Mal die Auflösung der Feature Maps halbiert. Das Erhöhen der Filteranzahl in den späteren Schichten dient dazu möglichst viele komplexe Merkmale abzudecken. Nach jeder Faltungsschicht wird Batch Normalisierung angewendet (s. Unterabschnitt 2.1.5).

Am Ende des Netzes wird ein globales Average Pooling ausgeführt, wodurch das ResNet theoretisch unabhängig von der Eingabegröße ist. Jede Feature Map wird auf ihren globalen Mittelwert reduziert, wodurch sich bei 2048 Filtern ein 2048-dimensionaler

Merkmalsvektor ergibt. Die vollverbundene Schicht, welche zur Klassifikation für den ImageNet Datensatz dient wird nicht benötigt und kann verworfen werden.

4.2.2 *Geteilte Schichten*

Basis der Modellarchitektur ist das ResNet-101, welches auf dem ImageNet Datensatz vortrainiert wird. Anstelle einer zufälligen Initialisierung werden oft vortrainierte Netze zur Initialisierung gewählt. Selbst wenn die Aufgaben sich stark unterscheiden, kann das Vortrainieren zu einer schnelleren Trainingskonvergenz führen. Die Intuition dabei ist, dass die ersten Faltungsschichten, welche in der Regel einfache Kantendetektoren sind, dadurch eine einigermaßen passende Konfiguration besitzen (vgl. [GBC16] Kapitel 15.2). Diese sollten zumindest nicht schlechter als zufällig initialisierte Gewichte sein.

Das komplette CNN für die Merkmalsberechnung ist in Abbildung 4.2.2 zu sehen, wobei die Schichten des ResNet-101 in orange dargestellt sind. Zusätzlich werden Feature Maps aus unterschiedlichen Hierarchiestufen miteinander verknüpft. Hintergrund dabei ist, dass Merkmale in den vorderen Schichten geometrische Informationen enthalten, während hintere Schichten semantisch abstrakte Merkmale berechnen. Dies erlaubt es dem Netz low Level und high Level Merkmale zu kombinieren, welches insbesondere von besonderer Bedeutung ist, wenn verschiedene Aufgaben betrachtet werden, die unterschiedliche Informationen benötigen. Beispielsweise sind für die Landmarkenerkennung präzise lokale Informationen vonnöten, während für z.B. für die Geschlechtervorhersage eher abstraktere Merkmale gebraucht werden.

Dafür werden die Feature Maps am Ende der vier Blocksequenzen betrachtet. Um diese zu kombinieren und weitere hilfreiche Informationen aus diesen zu extrahieren werden abwechselnd Faltungen angewendet (blaue Schichten) und die daraus resultierenden Feature Maps mit denen aus der nächsten Sequenz addiert. Bei den Faltungen handelt es sich um Blöcke, welche aus einer 3×3 und einer 1×1 Faltungsschicht bestehen. Innerhalb der 3×3 Faltungen wird analog zum ResNet eine Schrittweite von 2 gewählt, damit die Auflösungen der Feature Maps übereinstimmen. Die 1×1 Faltungen dienen zur Erhöhung der Anzahl an Feature Maps, womit eine elementweise Addition möglich ist. Nach jeder Faltungsschicht wird Batch Normalisierung durchgeführt. Alle verborgenen Einheiten verwenden eine ReLU Aktivierung. Am Ende wird ein globales Average Pooling ausgeführt, welches zu einem 2048-dimensionalen Merkmalsvektor führt.

Name	Ausgabegröße	Resnet-18	Resnet-101
Conv1	112×112	$7 \times 7, 64$, stride 2	$7 \times 7, 64$, stride 2
Blocksequenz 1	56×56	3×3 Max Pooling	3×3 Max Pooling
		$\begin{pmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{pmatrix} \times 2$	$\begin{pmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{pmatrix} \times 3$
Blocksequenz 2	28×28	$\begin{pmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{pmatrix} \times 2$	$\begin{pmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{pmatrix} \times 4$
Blocksequenz 3	14×14	$\begin{pmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{pmatrix} \times 2$	$\begin{pmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{pmatrix} \times 23$
Blocksequenz 4	7×7	$\begin{pmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{pmatrix} \times 2$	$\begin{pmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{pmatrix} \times 3$
GAP	1×1	7×7 Average Pooling	7×7 Average Pooling

Tabelle 4.2.1: Architektur des Resnet-18 und Resnet-101. Tabelle inspiriert nach [HZRS15a].

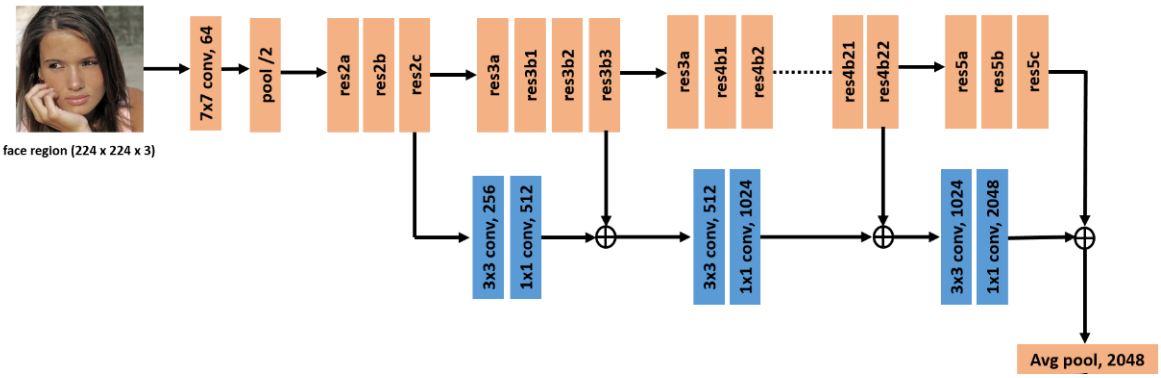


Abbildung 4.2.2: Visualisierung der Modellarchitektur. In orange sind die Schichten des ResNets dargestellt, während in blau die zusätzlichen Schichten gekennzeichnet sind. Die Verknüpfung wird durch eine elementweise Addition erreicht (Additionssymbol). Abbildung entnommen aus [RPC16].

4.2.3 *Aufgabenspezifische Schichten*

Nach den geteilten Schichten folgen aufgabenspezifische separate Berechnungszweige, welche als Eingabe die am Ende des Encoders berechnete Merkmalsrepräsentation erhalten. In der ursprünglichen Arbeit, welche die Architektur eingeführt hat, wurde ein Region Proposal mithilfe des Selective Search Algorithmus durchgeführt. Dies erlaubt dem Verfahren Bilder mit mehr als nur einem Gesicht zu betrachten. Durch das Region Proposal kommt es jedoch zu einer erheblichen Erhöhung der Trainingsdauer, da jeder Bildausschnitt einzeln vom Netz verarbeitet werden muss. Aufgrund dessen wird an dieser Stelle die vereinfachende Annahme getroffen, dass pro Bild nur ein Gesicht annotiert ist . Dies ist natürlich keine Restriktion der in dieser Arbeit verwendeten Methode. Jederzeit ließe sich ein externes Modul zur Generierung von möglichen Bildbereichen mit Gesichtern einsetzen, welches jedoch aus Rechenkomplexität vernachlässigt wird.

- **Landmarkenerkennung:**

  Hierfür müssen die Koordinaten der Landmarken innerhalb des Bildes vorhergesagt werden. Bei n Landmarken sind somit $2n$ Werte nötig. Dafür wird ein MLP mit zwei vollverbundenen Schichten verwendet. Die Ausgabe wird mithilfe von $2n$ Ausgabeneuronen modelliert, welche eine lineare Aktivierung besitzen. Nach der ersten Schicht wird Batch Normalisierung durchgeführt. Dropout wird

hier nicht angewendet. Als Fehlerfunktion wird der L2 Loss bzw. der Wing Loss verwendet.

- **Detektion:**

  Da kein Region Proposal verwendet wird, ändert sich hier die Problemstellung. Statt einer Klassifikation des betrachteten Bildausschnittes, gilt es das Gesicht innerhalb des Bildes zu lokalisieren. Dafür werden Bounding Boxes vorhergesagt. Eine Bounding Box ist mit genau vier Werten eindeutig definiert. Die Architektur für die Detektion wird analog zur Landmarkenerkennung gewählt, wobei nur vier Ausgabeneuronen benötigt werden. Für den Taskloss wird der L2 Loss gewählt.

- **Attribute:**

  Innerhalb der Arbeit werden hauptsächlich binäre Attribute betrachtet. Sind diese binär kann die Ausgabe mit einer Sigmoidaktivierung ausgedrückt werden. Bei einem multinomial/multinoulli verteiltem Attribut kann die Softmax Funktion verwendet werden. Falls die Anzahl an annotierten Attribute sehr hoch ist, können diese nicht mehr in eigenen Berechnungzweigen bestehend aus vollverbundenen Schichten modelliert werden. Dies würde jegliche Speicheranforderungen bei weitem übersteigen. In diesem Fall werden alle Attribute in einem einzigen MLP modelliert. Dabei wird für jedes Attribut ein Ausgabenneuron verwendet, wobei dann eine elementweise Sigmoidaktivierung angewendet wird. Die ersten Experimente haben früh gezeigt, dass dieser Teil zu Overfitting neigt. Deswegen wird hier zusätzlich Dropout zur Regularisierung verwendet (s. Unterabschnitt 2.1.3). Bei einem binären Attribut kann die Kreuzentropie auf den Zweiklassenfall reduziert werden. Es ergibt sich die binäre Kreuzentropie mit $L = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$.

- **Semantische Segmentierung:**

  In keiner der hier verwendeten Datensätze ist eine Annotation für die semantische Segmentierung von Gesichtern vorhanden. Diese zu sammeln ist ein aufwändiger und teurer Prozess. Aus diesem Grund wird diese automatisch aus den vorhandenen Informationen generiert. Dafür wird die konvexe Hülle der Landmarken berechnet, welches in Abbildung 4.2.3 visualisiert ist. Zur Segmentierungsmaske gehören die Pixel, welche innerhalb der konvexen Hülle liegen. Dies stellt somit nur eine Approximation der tatsächlichen Gesichtspixel dar. Für diese Aufgabe ist die räumliche Auflösung enorm wichtig. Aufgrund dessen werden hier die Feature Maps vor dem globalen Average Pooling als Eingabe

Abbildung 4.2.3: Die konvexe Hülle der Landmarken bietet eine vereinfachende Approximation der Annotation für eine semantische Segmentierung.

verwendet. Innerhalb der oben genannten Architektur wird die Auflösung der Feature Maps insgesamt fünf mal halbiert. Aus diesem Grund werden fünf lernbare Upsampling Schichten verwendet, welche wieder die ursprüngliche Auflösung von 224×224 Pixeln herstellen (s. Unterabschnitt 2.1.3). Jede Schicht verwendet 64 Kernel außer der letzten, die aus genau einem Filter besteht, welche die Ausgabe der semantische Segmentierung berechnet. Alle Filter verwenden eine Kernelgröße von 3 und eine Schrittweite von 2, um die Auflösung analog zum Downsampling zu verdoppeln. Bei der Segmentierung werden zwei Klassen betrachtet, womit wiederum die binäre Kreuzentropie verwendet wird. Frühe Experimente haben jedoch gezeigt, dass dies nicht ausreicht. Innerhalb eines Bildes überwiegt die Anzahl der Hintergrundpixel um ein wesentliches. Das Modell lernt aus diesem Grund für alle Pixel nur noch die Hintergrundklasse vorherzusagen. Um dies zu vermeiden wird die Gesichtsklasse mit einem Faktor $C > 1$ stärker gewichtet. Es ergibt sich $L = Cy \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$.

Das Modell wird mit der Unsicherheitsgewichtung der Gesamtkostenfunktion trainiert, wobei die passenden Kostenfunktionen für die Aufgaben eingesetzt werden. Dafür wird ein Gradientenabstieg mit Backpropagation verwendet (s. Unterabschnitt 2.1.6). Bei einem regulären Gradientenabstieg kann es zu Schwierigkeiten bei der Konvergenz kommen, da eine feste Lernrate benutzt wird. Aus diesem Grund haben sich einige Erweiterungen durchgesetzt, welche zusätzlich Statistiken über Gradienten aus vorherigen Iterationen in das Parameterupdate einfließen lassen. In dieser Arbeit wird der Adam (Adaptive Moment) Algorithmus verwendet, mit dem effektiv eine dynamische Lernrate für jeden Parameter realisiert wird [KB15]. Dafür

werden der erste und der nicht zentrierte zweite Moment der Gradienten adaptiv nach jedem Batch mit

$$m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t \odot g_t$$

neu bestimmt, wobei β_1, β_2 Momentum genannt werden und $\odot$ die elementweise Multiplikation darstellt. Indem $\beta_1, \beta_2 \in [0, 1)$ gewählt wird, werden ältere Gradienten exponentiell schwächer gewichtet. Da die Momente zu Beginn mit 0 initialisiert werden, sind diese verzerrt (biased). Aus diesem Grund werden diese noch korrigiert mit

$$\hat{m_t} \leftarrow \frac{m_t}{1-\beta_1^t}$$
$$\hat{v_t} \leftarrow \frac{v_t}{1-\beta_2^t}.$$

Mit den Momenten wird die Lernrate skaliert, sodass das Parameterupdate des regulären Gradientenabstiegs (s. Unterabschnitt 2.1.6) aktualisiert wird zu

$$w_{i,j} = w_{i,j} - \eta \frac{\hat{s_t}}{\hat{r_t} + \epsilon}, \tag{4.2.2}$$

wobei ϵ eine kleine Konstante ist, welche für numerische Stabilität benötigt wird.

5 EVALUIERUNG

Nachdem im vorherigen Kapitel die Methodik dieser Arbeit erläutert wurde, werden in diesem Kapitel die Experimente aufgeführt mit denen die Qualität des Verfahrens getestet wurde. Dafür werden Datensätze von Gesichtsbildern verwendet, welche innerhalb von natürlichen Szenen aufgenommen wurden.

Abschnitt 5.1 wird die Datensätze und die Details zu den dazugehörigen Aufgaben präsentieren. Innerhalb von Abschnitt 5.2 werden die Metriken beschrieben mit denen die Modellleistungen in den unterschiedlichen Aufgaben bewertet werden. In Abschnitt 5.3 wird der generelle Versuchsaufbau erläutert. Daraufhin werden die Ergebnisse in Abschnitt 5.4 vorgestellt und diskutiert.

5.1 DATENSÄTZE

Im folgendem Abschnitt werden die evaluierten Datensätze präsentiert. Diese enthalten Gesichter aus dem Alltag und besitzen damit eine große Varianz im visuellen Erscheinungsbild. Insbesondere existieren starke Unterschiede in der Pose, Mimik und den Gesichtsattributen. Hiermit wird versucht ein reales Szenario nachzubilden, wodurch die Generalisierungsfähigkeit des Models getestet werden kann. Zusätzlich enthalten die Datensätze Annotation für verschiedene Aufgaben, womit diese für das MTL eingesetzt werden können.

5.1.1 *Multi Task Facial Landmark (MTFL)*

Bei dem Multi Task Facial Landmark (MTFL) Datensatz [ZLLT14a] handelt es sich um Bilder von Personen, welche aus dem Internet extrahiert wurden. Insgesamt besteht der Datensatz aus 10.000 Trainingsbildern. Abbildung 5.1.1 zeigt einige Beispiele. Auffällig ist hierbei die extreme Varianz in den Daten. Mitunter sind die Bilder unter unterschiedlichen Aufnahmebedingungen entstanden. Beispielsweise finden sich im Datensatz sogar gemalte Porträts wieder. Das Evaluierungsprotokoll aus [ZLLT14a] verwendet ca. 3000 Bilder aus einem weiteren Datensatz (s. Unterabschnitt 5.1.3).

Auf jedem Bild ist genau ein Gesicht annotiert, wobei jedes Gesicht mit genau fünf Landmarken annotiert ist (beide Augen, Nase, beide Mundwinkel). Falls eine Land-

Abbildung 5.1.1: In der Abbildung sind beispielhaft einige Bilder aus dem MTFL Datensatz [ZLLT14a] dargestellt, welche die Variabilität in den Daten zeigt.

marke nicht sichtbar bzw. verdeckt ist, so werden diese aus dem Kontext geschätzt. Zusätzlich sind jeweils die binären Attribute Geschlecht, Lachen und das Vorhandensein einer Brille vorhanden. Die Pose des Gesichtes ist im Winkel zur Kamera gegeben, wobei diese auf fünf Werte diskretisiert wurde ($0°, \pm 30°, \pm 60°$).

5.1.2 *CelebA und MAFL*

Mit dem CelebA Datensatz ist ein umfangreicher Datensatz gegeben, der aus über 200000 Bildern besteht [LLWT14]. Auf jedem Bild ist einer von 10177 Berühmtheiten enthalten. Dieser Datensatz wurde primär für die Analyse unterschiedlichster Gesichtsattribute erstellt. Für jede Person sind 40 binäre Attribute annotiert, welche verschiedene Eigenschaften beschreiben. Einige davon sind beispielhaft in Abbildung 5.1.2 dargestellt. Diese enthalten umfangreiche Informationen über visuelle Eigenschaften der Personen.

Für den Datensatz ist eine feste Aufteilung in Trainings-, Validierungs- und Testdatensatz gegeben. Obwohl dieser Datensatz bisher eher in der Attributanalyse verwendet wurde, sind zusätzlich Annotationen für die Landmarken und Bounding Boxen der Gesichter vorhanden, womit dieser für das vorgestellte MTL Modell benutzt werden kann. Nach bestem Wissen und Gewissen des Autors sind bisher auf diesem Protokoll keine Ergebnisse für die Landmarkendetektion präsentiert worden. Die einzigen Resultate wurden auf einer kleineren Untermenge veröffentlicht. Mit dem Multi Attribute Facial Landmark (MAFL) ist ein Evaluierungsprotokoll auf Basis des

Abbildung 5.1.2: Diese Abbildung zeigt einige auf dem CelebA Datensatz definierte Attribute mit einigen Beispielbildern. Abbildung entnommen aus [LLWT].

CelebA Datensatzes gegeben, welcher aus 20000 zufällig gezogenen Bildern besteht [ZLLT14b]. Zusätzlich wurden zufällig 1000 Bilder für den Testfall ausgesucht.

5.1.3 *Annotated Facial Landmarks in the wild (AFLW)*

Der Annotated Facial Landmarks in the Wild (AFLW) Datensatz ist ein Benchmark für die Landmarkendetektion, welcher von der Arbeitsgruppe Learning, Recogition & Surveilance der TU Graz gepflegt wird [KWRB11]. Insgesamt sind auf 21997 Alltagsbildern 25993 Gesichter abgebildet. Da ein Region Proposal aus Gründen der Rechenkomplexität vernachlässigt wird, sind Bilder mit mehreren Gesichtern problematisch. Deswegen werden die Bilder herausgefiltert, welche nur ein annotiertes Gesicht enthalten. An dieser Stelle sei angemerkt, dass hierdurch nicht garantiert ist, dass auf jedem Bild auch nur ein Gesicht abgebildet ist. Insgesamt verbleiben somit 18956 Bilder.

Eine Besonderheit des Datensatzes ist, dass jedes Gesicht mit einer Bounding Box und bis zu 21 Landmarken annotiert ist (s. Abbildung 5.1.3), wobei dabei nur die für das menschliche Auge sichtbaren Landmarken annotiert wurden.

Zusätzlich sind einige Attribute, wie z.B. das Geschlecht annotiert. Im weiteren ist die Pose der Gesichter relativ zur Kamera gegeben. Diese ist über drei Achsen (Roll-Nick-Gier-Winkel) angegeben (s. Abbildung 5.1.4). Die Pose eines Gesichtes wurde mithilfe eines mittleren 3D Modells und der zum Gesicht gehörenden Landmarken geschätzt. Des Weiteren ist die Form der Gesichter mithilfe einer Ellipse umrahmt. Diese ist ein präziserer Indikator für die Gesichtspixel als die konvexe Hülle der Landmarken, welches beispielhaft in Abbildung 5.1.5 zu sehen ist.

5.2 METRIKEN

In diesem Abschnitt werden die Metriken erläutert mit denen die Modelleistung in den unterschiedlichen Aufgaben gemessen wird. Eine Standardmetrik für die Landmarkendetektion ist der Normalized Mean Error (NME), welcher die Genauigkeit der vorhergesagten Landmarken misst. Dieser wird mit

$$\mathrm{NME} = \frac{1}{N}\frac{1}{m}\frac{\|(\hat{y} - y)\|_2}{d_{iod}} \tag{5.2.1}$$

berechnet, wobei N die Anzahl an zu testenden Bildern und m die Anzahl an annotierten Landmarken bezeichnen. Es wird also die euklidische Distanz bestimmt, welche zusätzlich mit der Anzahl an definierten Landmarken normalisiert wird. Da

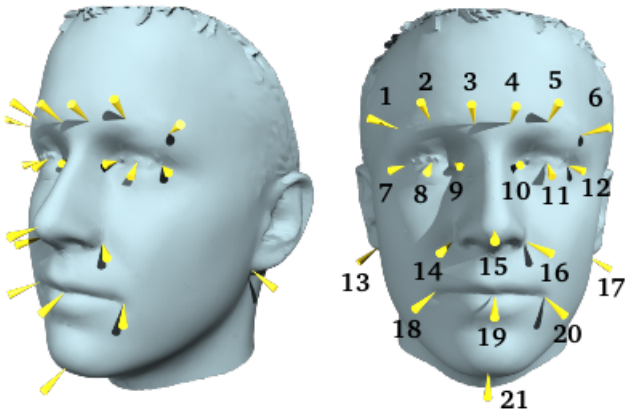


Abbildung 5.1.3: Auf dem AFLW Datensatz sind 21 Landmarken definiert, wobei diese jedoch nur annotiert worden sind, falls diese sichtbar sind. Abbildung entnommen aus [KWRB11].

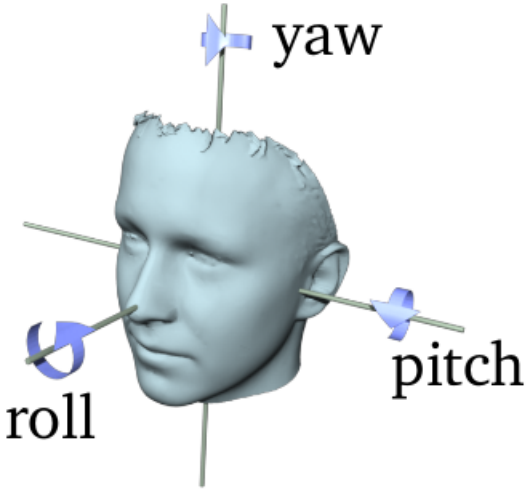


Abbildung 5.1.4: Die Pose eines Gesichtes ist über die Roll-Nick-Gier-Winkel relativ zur Kamera angegeben. Abbildung entnommen aus [KWRB11].

Abbildung 5.1.5: Innerhalb des AFLW Datensatzes ist eine Schätzung der Gesichtsform mit einer Ellipse vorhanden. Diese bietet eine bessere Schätzung für die Annotation der semantischen Segmentierung. Ursprungsbild entnommen aus [KWRB11].

die Skalierung des Gesichtes eine unmittelbare Rolle auf die Metrik hat, wird diese im Weiteren mit der sogenannten Inter-ocular-Distanz (d_{iod}) normalisiert. Dies ist der Abstand zwischen den Augen, welche durch die gegebenen Landmarken berechnet wird, womit die Metrik einigermaßen invariant gegenüber unterschiedlichen Skalierungen ist.

Die Failure Rate (FR) gibt den Anteil an Bildern wieder, für die geeignete Landmarken vorhergesagt wurden. Ein Fehler diesbezüglich tritt auf, falls für ein Bild der NME größer als 10% ist (s. Gleichung 5.2.2).

$$FR = \frac{\sum_i NME(i) > 0.1}{N} \tag{5.2.2}$$

Für jedes Attribut wird die Klassifikationsgenauigkeit bestimmt. Diese ist gegeben mit

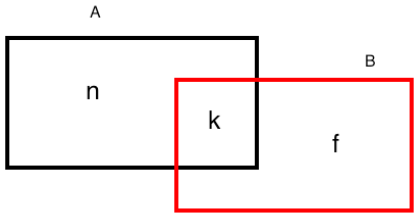
$$Acc_j^{att} = \frac{\sum_i \mathbb{1}_{y_{ij}=\hat{y}_{ij}}}{N}, \tag{5.2.3}$$

wobei die Identikatorfunktion angibt, ob für ein Bild i das Attribut j korrekt vorhergesagt wurde. Dabei wird die Ausgabe der Sigmoidfunktion mithilfe eines Schwellwertes von 0.5 auf die finale Klassifikation abgebildet.

Für die Detektion und Segmentierung muss zusätzlich die räumliche Präzision einer Vorhersage bewertet werden. Elementar dafür ist das Intersection over Union (IoU)

Maß. Seien A die annotierten Pixel und B die vorhergesagten Pixel eines Objektes, so ist diese in Gleichung 5.2.4 dargestellt.

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} = \frac{|k|}{|n| + |f| - |k|} \tag{5.2.4}$$



Mit der Intersection over Union Metrik wird somit das Verhältnis zwischen den korrekt erkannten Pixeln (Schnitt) und der Vereinigung (korrekt + falsch + nicht erkannte Pixel) berechnet. Für ein korrekt detektiertes bzw. segmentiertes Objekt gilt öfters dabei ein IoU Wert von mindestens 0.5 als ausreichend. Ein Gesicht wird somit als korrekt detektiert bzw. segmentiert betrachtet, falls der IoU Wert bei mindestens 0.5 liegt. Folglich kann die Detektionsgenauigkeit und die Segmentierungsgenauigkeit angegeben werden mit

$$\text{Acc}^k = \frac{\sum_i \text{IoU}(i) > 0.5}{N}, \qquad k \in \{\text{Detektion, Segmentierung}\}. \tag{5.2.5}$$

Die Pose ist im AFLW Datensatz über den Winkel (in Radiant) angegeben. Für die Pose wird der absolute Fehler für jede der drei Achsen einzeln berechnet nach

$$\text{Error}_{abs} = \frac{\|\hat{y} - y\|_1}{N}. \tag{5.2.6}$$

5.3 VERSUCHSAUFBAU

Der folgende Abschnitt beschreibt einige Hintergrundinformationen und den generellen Aufbau und die Struktur der durchgeführten Experimente. Auf den Datensätzen in denen keine feste Einteilung in Trainings-, Validierungs- und Testdatensatz vorhanden ist, werden zufällig 20% der Trainingsdaten für die Validierung gezogen. Da innerhalb des AFLW Datensatzes Bilder vorkommen in denen mehrere Personen annotiert sind, werden nur die Bilder verwendet, welche mit nur einem Gesicht annotiert sind. Im Weiteren ist auf dem AFLW Benchmark kein Testset definiert. Aus diesem Grund wird hierfür eine Aufteilung von 70%, 20%, 10% gewählt.

Alle Bilder werden für die Eingabe des Netzes auf 224×224 Pixel skaliert, wobei angenommen wird, dass es sich um RGB Bilder handelt. Es kann jedoch vorkommen,

dass innerhalb der Datensätze Grauwertbilder enthalten sind. In diesem Fall werden diese zu RGB Bildern konvertiert, indem die Grauwertstufen für die drei Kanäle kopiert werden.

Da der MTFL Datensatz im Vergleich zu den anderen Datensätzen klein ist, wird für diesen Datensatz anstelle des Resnet-101 das Resnet-18 für die Basis der Modellarchitektur verwendet (Abschnitt 4.2). Weil dieses weniger Kernel in den Faltungsschichten verwendet, müssen die Faltungsschichten, welche nicht zum ResNet gehören, ebenso angepasst werden, damit eine elementweise Addition durchgeführt werden kann. Die Schichten, welche zum ResNet gehören, werden durch ein Vortraining auf dem ImageNet Datensatz initialisiert, während die restlichen Schichten über die Standardimplementierung von PyTorch initialisiert werden. Diese verwendet die sogenannte He Initialisierung [HZRS15b]. In dieser werden die Gewichte zufällig aus einer Gleichverteilung $\mathcal{U}(-a, a)$ gezogen, wobei a berechnet wird mit

$$a = \sqrt{\frac{6}{\text{fan\_in}}} \tag{5.3.1}$$

und fan_in der Anzahl an eingehenden Verbindungen entspricht.

Die initialen Unsicherheiten werden mit $s = 1$ festgelegt, sodass $\sigma^2 = e$ gilt. Da $s = \log(\sigma^2)$ gilt, ist somit garantiert, dass die Varianzen immer positiv bleiben. Diese werden im selben Schritt wie die Netzgewichte aktualisiert, sodass dasselbe Optimierungsverfahren benutzt werden kann. Dafür wird der Adam Algorithmus mit den Standardwerten für die Momentumsparameter $(0.9, 0.999)$ eingesetzt.

Für die Kostenfunktion der semantischen Segmentierung ist eine stärkere Gewichtung der positiven Gesichtsklasse notwendig. Die Gesichtspixel werden zusätzlich mit einem Wert von 2.5 gewichtet, welches sich als ausreichend herausgestellt hat. Außerdem muss die Kostenfunktion der Landmarkendetektion für den AFLW Benchmark angepasst werden, da nicht garantiert ist, dass alle 21 Landmarken annotiert sind. Es ergibt sich

$$L_{lm} = \|v(\hat{y} - y)\|_2^2, \tag{5.3.2}$$

wobei $v \in \{0, 1\}^{42}$ kennzeichnet, ob eine bestimmte Landmarke annotiert ist. Falls eine Landmarke nicht annotiert ist, wird diese somit für die Kostenfunktion ignoriert. Zusätzlich wird eine weitere Hilfsaufgabe modelliert, welche die Sichtbarkeit der Landmarken vorhersagen soll. Es handelt sich dabei um ein Klassifikationsproblem, welches mit vollverbundenen Schichten und einer Sigmoidausgabe realisiert wird. Mit einem weiterem Unsicherheitsgewicht lässt sich dies problemlos in das Verfahren integrieren.

Das Training wird mit einer konstanten Lernrate durchgeführt, ohne eine Form von Scheduling der Lernrate. Es wird eine Lernrate von 0.001 verwendet solange nicht anders angegeben. Des Weiteren werden die Modelle mit einer festen Anzahl von Epochen trainiert ohne Early Stopping. Aus Laufzeitgründen wird nur nach jeder vierten Epoche validiert. Für den Testdatensatz werden jeweils das Modell mit dem besten Validierungsloss und das der letzten Epoche evaluiert.

5.4 ERGEBNISSE

In diesem Abschnitt werden die durchgeführten Experimente vorgestellt. Zu Beginn werden die Ergebnisse auf den einzelnen Datensätzen präsentiert, wobei insbesondere ein Vergleich zu dem State-of-the-Art gegeben wird. Daraufhin werden weitere Faktoren evaluiert, welche unabhängig von bestimmten Datensätzen sind. Um einen Eindruck über die Genauigkeit der Ergebnisse für die Landmarken-, Gesichtsdetektion und semantische Segmentierung zu geben, werden zum Schluss qualitative Ergebnisse präsentiert.

5.4.1 *MTFL*

Innerhalb der ersten Versuche wurden zuerst flachere Architekturen getestet. Dabei wurde ein CNN mit jeweils vier (CNN-4) und acht Faltungsschichten (CNN-8) für den geteilten Feature Encoder verwendet.

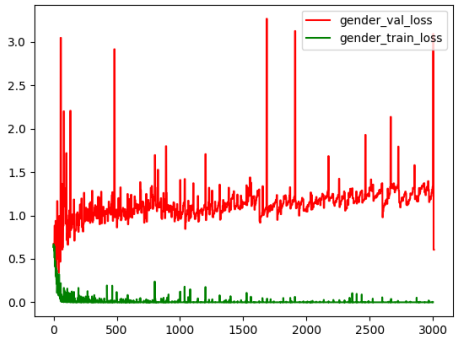
Abbildung 5.4.1 zeigt sowohl den Verlauf des Trainingslosses (grün) als auch des Validierungslosses (rot) einiger Aufgaben für das CNN-8 Modell. Auffällig ist hierbei, dass schon bei den kleineren Netzen für einige bestimmte Aufgaben Overfitting auftritt. Im Fall der Attributerkennung neigt das Modell zu Overfitting, da das Vorhersagen einzelner Attribute im Vergleich zur Landmarkenerkennung wesentlich einfacher zu lösen ist. Dies spiegelt sich mitunter auch in den gelernten Unsicherheiten wider, welche exemplarisch in Abbildung 5.4.2 dargestellt sind. Die Unsicherheiten, die für die Attribute gelernt werden, sinken direkt auf sehr kleine Werte, während die Unsicherheit bezüglich der Landmarkenerkennung anfangs wächst und im Laufe des Trainings wieder absinkt. Ähnliche Verläufe lassen sich auf den anderen Datensätzen und den weiteren Aufgaben beobachten. Dabei fällt auf, dass dieser Trend bei Aufgaben auftritt, welche einen L2 Loss verwenden. Dies zeigt, dass die Unsicherheiten die Skalierung der Kostenfunktionen miteinbeziehen und somit ein optimales Verhältnis zwischen den unterschiedlichen Aufgaben adaptiv lernen. Bezüglich der Landmarkendetektion kann kein Overfitting beobachtet werden. Die Differenzen zwischen

dem Trainings- und Validierungsloss sind dabei nicht signifikant und lassen sich auf die Tatsache zurückführen, dass eine Regression der Koordinaten mit einem L2 Loss durchgeführt wird. Diese hat dadurch einen größeren Dynamik-/Wertebereich als die (binäre) Kreuzentropie, womit der relativ kleine Unterschied erklärt werden kann. Aus diesem Grund wird im Weiteren für die Schichten, welche für die Attributerkennung zuständig sind, Dropout als Regularisierung verwendet. Für die weiteren Aufgaben, in denen eine präzise Lokalisierung gefordert ist, wird auf Dropout verzichtet.

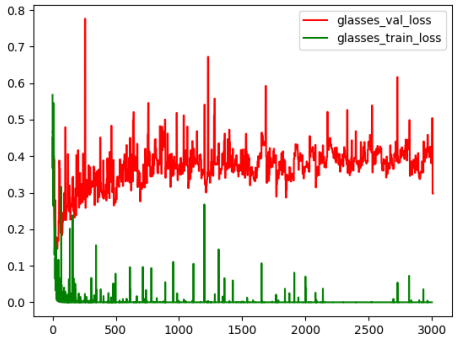
Innerhalb von Tabelle 5.4.1 sind die Ergebnisse auf dem MTFL Datensatz aufgelistet, wobei in der ersten Zeile der bisherige veröffentlichte State-of-the-Art dargestellt ist [ZLLT14a]. Da die Autoren jedoch keine Angaben zur Qualität der Attributerkennung ihres Modells angeben, ist kein Vergleich in diesen möglich. Die Pfeile dienen dabei zur Veranschaulichung und verdeutlichen für eine Metrik ob kleinere bzw. größere Werte besser sind.

Für das kleinste Modell (CNN-4) waren die erzielten Ergebnisse für die Landmarkendetektion wesentlich schlechter als die bisher veröffentlichten Resultate, welches sich mitunter aus der geringen Anzahl an Faltungsschichten im gemeinsamen Feature Encoder begründen lässt. Dieser bietet keine ausreichende Modellkomplexität, womit keine geeigneten Merkmale gelernt werden können. Dies ist erkennbar daran, dass das Verdoppeln der Faltungsschichten (CNN-8) unmittelbar zu besseren Ergebnissen führt. Mitunter erreicht dieses Modell bessere Resultate als die bisherigen State-of-the-Art Ergebnisse. Es kann eine Verbesserung des NME beobachtet werden, während die FR deutlich verbessert wurde. Das Reduzieren der FR um 3.4% deutet darauf hin, dass das Modell besser generalisiert, da für 3.4% mehr Bilder im Testdatensatz geeignete Landmarken vorhergesagt werden können. Zusätzlich ist eine Verbesserung in der Erkennung der Attribute ersichtlich. In sowohl CNN-4 als auch CNN-8 zeigen die Ergebnisse für die semantische Segmentierung, dass die Modelle keine ausreichend genaue Vorhersagen treffen können. Es kann zwar eine leichte Steigerung der Segmentierungsgenauigkeit beobachtet werden, jedoch sind die Resultate für die semantische Segmentierung mit 5.58% bei weitem nicht akkurat.

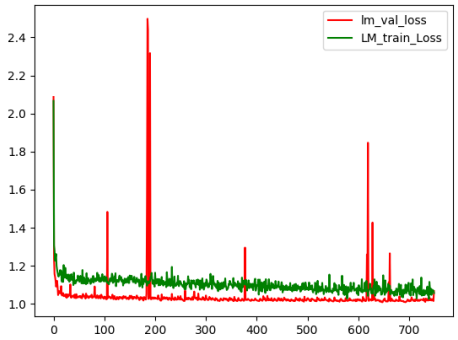
Des Weiteren wurde ein Experiment mit dem in Abschnitt 4.2 vorgestellten Modell durchgeführt, wobei aufgrund der Datenknappheit in diesem Datensatz das ResNet-18 als Basis genommen wurde, welches somit um einiges tiefer als die bisher verwendeten Netze ist. Die erzielten Ergebnisse sind in der 3. Zeile zu sehen. Dabei ist klar zu erkennen, dass die Zahlen gegenüber des CNN-8 weiterhin beachtlich verbessert wurden. Auch die Segmentierungsgenauigkeit kann auf 97.20% gesteigert werden, womit das Modell geeignete Vorhersagen für die semantische Segmentierung von Gesichtern treffen kann. Für einen Eindruck der Gesichtssegmentierung sei auf die qualitativen Ergebnisse in folgendem Unterabschnitt 5.4.7 verwiesen. Dies verdeutlicht weiterhin,



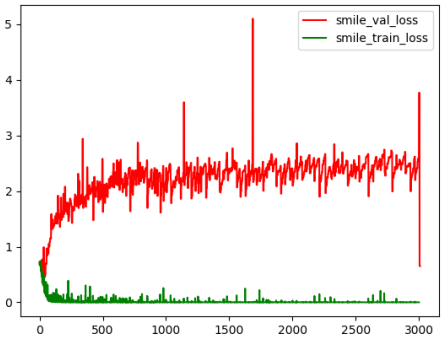
(a) Attribut: Geschlecht



(b) Attribut: Brille

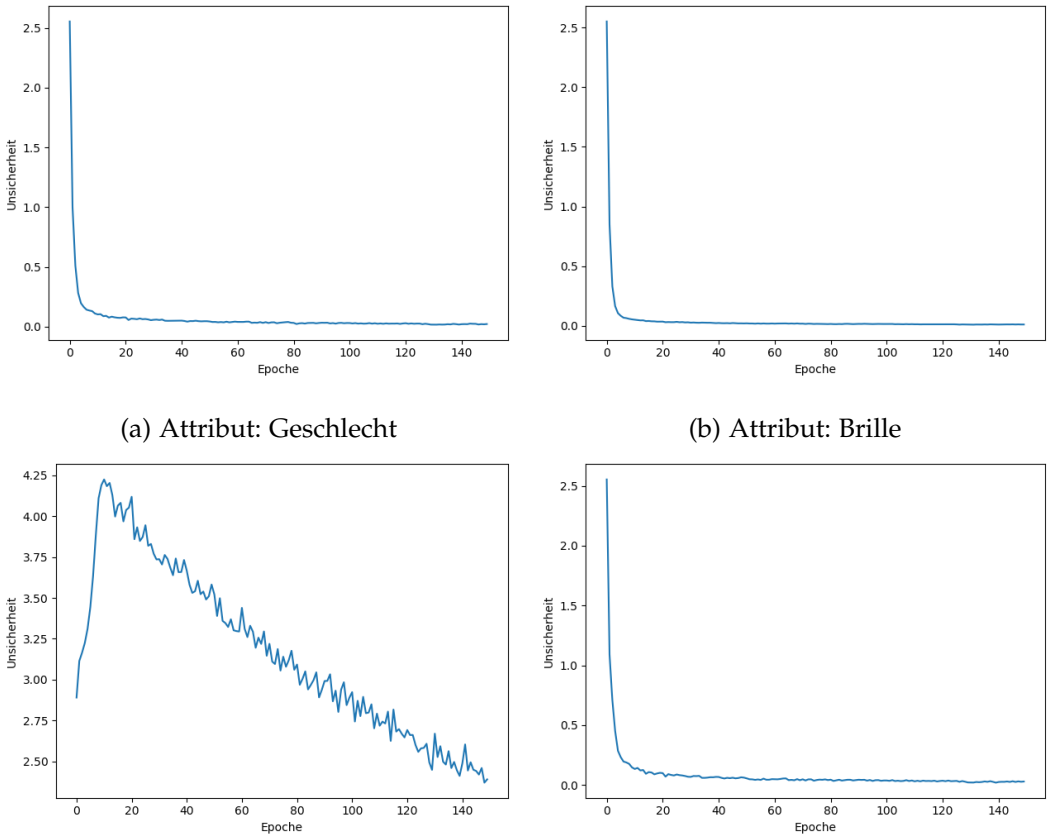


(c) Landmarkendetektion



(d) Attribut: Lachen

Abbildung 5.4.1: Vergleich zwischen dem Trainings- (grün) und Validierungsloss (rot) im Laufe des Trainings. Hierbei ist erkennbar, dass die Attributaufgaben zu Overfitting neigen. Der geringe Unterschied bei der Landmarkendetektion lässt sich auf den größeren Dynamikbereich des L2 Loss zurückführen.



(a) Attribut: Geschlecht

(b) Attribut: Brille

(c) Landmarkendetektion

(d) Attribut: Lachen

Abbildung 5.4.2: In der Grafik ist beispielhaft die Entwicklung der gelernten Unsicherheiten dargestellt.

dass für die pixelgenaue Segmentierung eine gewisse Tiefe des Modells benötigt wird. Im Vergleich zum aktuellen State-of-the-Art konnte mit diesem Modell für die Landmarkenlokalisierung eine deutliche Verbesserung der Ergebnisse erreicht werden. Dabei wurde der NME um 3.7% verringert, welches fast der Hälfte des ursprünglichen Fehlers entspricht. Des Weiteren konnte die FR von 25% auf 6.11% reduziert werden, welches eine signifikante Steigerung der Präzision des Modells darstellt, da dadurch für 18,89% mehr Testbilder akkurate Landmarken vorhergesagt wurden. Es kann also beobachtet werden, dass der tiefere Feature Encoder zu besseren Ergebnissen führt. Da die Resultate für die Attribute auch unmittelbar verbessert wurden, obwohl diese zu leichtem Overfitting neigen, kann die Annahme getroffen werden, dass das MTL Paradigma im Bereich von tiefen CNNs als implizite Regularisierung dient, womit Merkmalsrepräsentationen gelernt werden, welche auf unterschiedliche Aufgaben generalisieren. Zusätzlich erlaubt das Kombinieren von Feature Maps unterschiedlicher Hierarchiestufen innerhalb der Architektur Informationen weiter zu leiten, welche für bestimmte Aufgaben von Vorteil sind. Die dazugehörigen Faltungsschichten, welche auf den verknüpften Feature Maps angewendet werden, extrahieren weitere Informationen, die aus der gemeinsamen Betrachtung von low level und abstrakten Merkmalen entstehen, welches sich in den verbesserten Ergebnissen widerspiegelt. Das Vorhersagen ob eine Person auf einem Bild eine Brille trägt ist mit einer Genauigkeit von 97.20% fast fehlerfrei. Für das Attribut Lachen kann die geringere Genauigkeit zum Teil möglicherweise auf die Problematik zurückgeführt werden, dass das Empfinden davon subjektiv sein kann. Eine Interpretation für die fehlende Genauigkeit beim Erkennen des Geschlechts ist der Mangel an Trainingsbildern im Datensatz, da in den folgenden Datensätzen mit mehr Daten diese zu ausreichenden Ergebnissen führt (s. Unterabschnitt 5.4.3 und Unterabschnitt 5.4.2). Da die Pose in fünf Ausprägungen quantifiziert wurde, ist diese generell ungenau, welches die durchwachsenen Ergebnisse erklären könnte.

Die Versuche zeigen, dass das dynamische Lernen der Unsicherheiten für die Gewichtung der Gesamtkostenfunktion geeignet ist, um MTL durchzuführen. Somit lässt sich eine aufwändige Hyperparametersuche bzw. komplexe mehrstufige Optimierungsverfahren vermeiden (s. Abschnitt 3.5). Insbesondere konnte mit der in dieser Arbeit verwendeten Methode der bisher veröffentlichte State-of-the-Art erheblich übertroffen werden.

Methode	NME ↓	FR↓	Seg.↑	Geschlecht ↑	Lachen↑	Brille↑	Pose↑
[ZLLT14a]	8.0	25.0	-	-	-	-	-
CNN-4	9.92	34.96	0.33	69.18	64.64	92.32	61.64
CNN-8	7.73	21.60	5.58	74.62	70.22	96.46	64.17
MTL Modell-18	4.30	6.11	97.20	81.47	79.33	97.80	73.66

Tabelle 5.4.1: Ergebnisse der verschiedenen Modelle auf dem MTFL Datensatz. Die Pfeile verweisen darauf, ob hohe bzw. niedrigere Werte für eine Metrik besser ist.

5.4.2 *CelebA*

Auf dem CelebA Datensatz haben sich bisherige Arbeiten nur mit der Attributerkennnung beschäftigt. Somit sind für die Landmarkendetektion keine Referenzwerte vorhanden, diese sind jedoch auf dem MAFL Protokoll, welcher Teil des nächsten Abschnittes ist, verfügbar.

Tabelle 5.4.2 gibt einen Überblick über die Ergebnisse auf dem Datensatz. Dabei wurden zwei Versionen des Modells betrachtet. Es wurde wieder die Variante mit dem ResNet-18 als Basis (2. Zeile) und die Architektur mit dem weitaus tieferen ResNet-101 als Grundlage (3. Zeile) verwendet. Hierbei ist zu erkennen, dass die tiefere Version zu deutlich genaueren Ergebnissen führt. Sowohl für die semantische Segmentierung als auch Gesichtsdetektion können hiermit adäquate Resultate erreicht werden, wobei diese nahezu fehlerfrei sind. Im Weiteren ist auch zu sehen, dass für die Landmarkendetektion mit einem NME von 2.76% und einer FR von 1.72% geeignete Vorhersagen erzielt werden. Ein westlicher Grund für die guten Ergebnisse ist hierbei der große Datensatz, welcher für das Training des Netzes ausgenutzt werden kann. Dadurch ist das Netz mit seiner erhöhten Modellkomplexität in der Lage komplexe Strukturen und Muster in den Daten zu finden, ohne das dabei Overfitting auftritt.

Die Verbesserung kann insbesondere bei der Attributvorhersage beobachtet werden (s. Tabelle 5.4.3). In der Tabelle sind die Attribute in den Zeilen aufgelistet, wobei die Spalten die verschiedenen Resultate der Attribute enthalten. Die letzte Zeile gibt jeweils

Methode	NME↓	FR↓	Seg.↑	Det. ↑
MTL Modell-18	7.01	12.52	71.77	98.59
MTL Modell-101	2.76	1.72	99.32	99.65

Tabelle 5.4.2: Ergebnisse auf dem CelebA Datensatz.

das Mittel über alle 40 Attribute an. Für die flachere Variante (MTL Modell-18) konnten schlechtere Ergebnisse beobachtet werden, welches sich auf die nicht ausreichende Modellkomplexität zurückführen lässt. Insgesamt ist eine Verbesserung von 11.71% bezüglich der mittleren Attributgenauigkeit zu sehen. Mit der tieferen Variante konnten die Ergebnisse der Autoren Liu et al [LLWT14], welche den Datensatz eingeführt haben, übertroffen werden. Im Vergleich zum aktuellen State-of-the-Art [GRB16] liegen die Ergebnisse jedoch unter denen mit einer Differenz von knapp 3.10% An dieser Stelle sei betont, dass der Fokus in der Arbeit von [GRB16] auf der Attributerkennung lag. Des Weiteren wurden unterschiedliche Formen der Datenaugmentierung verwendet. Zusätzlich wurden die Bilder mithilfe der annotierten Landmarken normalisiert, sodass die Gesichter dieselbe Skalierung als auch Ausrichtung besitzen. Diese Faktoren sind eine Begründung für die besseren Ergebnisse.

Für die meisten Attribute können hinreichend gute Genauigkeiten erzielt werden. Unter den Attributen existieren jedoch einige, welche schwerer vorherzusagen sind. Insbesondere subjektive Eigenschaften ("Attractive", "Big Lips", etc.) als auch komplexere geometrische Abhängigkeiten ("Arched Eyebrows", "Oval Face", "Pointy Nose") schneiden dabei schlechter ab, welches sich mit den Ergebnissen der anderen Arbeiten deckt.

5.4.3 *MAFL*

In diesem Abschnitt werden die erzielten Ergebnisse auf dem MAFL Protokoll aufgeführt. Innerhalb von Tabelle 5.4.4 sind zwei Experimente dargestellt, wobei sich diese in der verwendeten Lernrate unterscheiden. Für einen Überblick in die Qualität der Attributerkennung sei auf Tabelle 5.4.5 verwiesen. Das beste Ergebnis ist in der letzten Zeile zu sehen, wobei wieder eine Lernrate von 0.001 verwendet wurde. Im Experiment, welches in der vorletzten Zeile abgebildet ist, wurde die Lernrate um einen Faktor 10 erhöht. Das Vergrößern der Lernrate auf 0.01 führte jedoch zu schlechteren Ergebnissen bezüglich der Landmarkendetektion. Im Hinblick auf die Attributerkennung kann jedoch eine leichte Verbesserung der gemittelten Genauigkeit beobachtet werden. Auch auf diesem Datensatz können akkurate Bounding Boxen innerhalb der Bilder lokalisiert werden. Für die semantische Segmentierung ergeben sich bezüglich der synthetischen Annotation genaue Ergebnisse.

Die Referenzwerte aus [ZLLT14b] sind in der ersten Zeile zu sehen. Hierbei ist eine deutliche Steigerung der Ergebnisse zu diesen erkennbar. Der NME konnte um 4.47% reduziert werden, womit dieser effektiv halbiert wurde. Weiterhin konnte die FR von 24.9% auf 2.4% verbessert werden. Demnach ist eine Verbesserung von 22.5

Attribut	[LLWT14]	[GRB16]	MTL Modell-18	MTL Modell-101
5 o Clock Shadow	91	94.87	90.01	92.47
Arched Eyebrows	79	94.21	71.47	78.63
Attractive	81	82.86	57.85	77.82
Bags under Eyes	79	85.31	79.74	81.64
Bald	88	99.09	97.88	98.67
Bangs	95	96.10	84.43	94.27
Big Lips	68	72.70	67.30	69.97
Big Nose	78	84.38	78.80	77.86
Black Hair	88	90.36	72.84	85.94
Blond Hair	95	96.20	86.67	93.82
Blurry	84	96.37	94.94	95.04
Brown Hair	80	88.40	82.03	85.58
Bushy Eyebrows	90	92.46	87.05	89.55
Chubby	91	95.74	94.79	94.65
Double Chin	92	96.43	95.43	95.68
Eyeglasses	99	99.68	93.54	98.66
Goatee	95	97.55	95.52	96.40
Gray Hair	97	98.39	96.81	97.16
Heavy Makeup	90	92.10	59.50	87.84
High Cheekbones	88	87.77	51.82	84.16
Male	94	98.50	61.35	95.14
Mouth slightly open	92	94.06	50.49	90.47
Mustache	95	97.11	96.13	96.30
Narrow Eyes	81	87.71	85.13	85.20
No Beard	95	96.45	14.63	93.66
Oval Face	66	77.31	70.44	73.38
Pale Skin	91	97.05	95.70	95.80
Pointy Nose	72	76.90	71.43	72.65
Receding Hairline	89	93.67	91.51	91.97
Rosy Cheeks	90	95.16	92.83	93.44
Sideburns	96	97.84	95.36	96.27
Smiling	91	92.96	49.97	89.63
Straight Hair	76	85.26	79.01	80.71
Wavy Hair	80	86.25	63.60	79.22
Wearing Earrings	82	90.98	79.34	85.47
Wearing Hat	99	99.10	95.80	98.37
Wearing Lipstick	93	93.96	47.81	90.51
Wearing Necklace	88	89.27	86.21	85.98
Wearing Necktie	79	97.29	92.99	96.19
Young	86	88.98	24.29	84.79
∅	87	91.67	77.06	88.77

Tabelle 5.4.3: Genauigkeiten auf dem CelebA Datensatz für die Attributerkennung. Diese sind in Prozent angegeben.

Prozentpunkten auf dem Testdatensatz gegeben, welches einer relativen Verbesserung von knapp 90% entspricht. Die aktuellere Arbeit [SSG+18] betrachtet dieses Protokoll im Kontext von Autoencodern. Mit den dort vorgestellten Deforming Autoencodern wird explizit innerhalb der latenten Repräsentation im Bottleneck des Netzes das Bild in Textur und Verformung zerlegt. Aus diesen wird das Usprungsbild wieder rekonstruiert. Um die Qualität der latenten Repräsentationen zu evaluieren, geben die Autoren den NME auf dem MAFL Datensatz an, welches den aktuellen State-of-the-Art hierauf bietet (2. Zeile). Auch im Vergleich zu dem State-of-the-Art konnte eine Verbesserung von bis 1.93% erzielt werden. Es sei an dieser Stelle erwähnt, dass mit den Autoencodern ein unüberwachtes Lernen der Repräsentationen durchgeführt wird. Für die Inferenz der Landmarken wurde ein weiterer Regressor in Form eines MLPs benutzt, welcher auf den latenten Repräsentationen des Autoencoders trainiert wurde.

5.4.4 *AFLW*

Der folgende Abschnitt gibt einen Einblick in die Qualität des Verfahrens auf dem AFLW Datensatz. Tabelle 5.4.6 zeigt einen Überblick einiger Experimente. Die Ergebnisse der Referenzarbeit, welche die in dieser Arbeit verwendete Architektur vorgestellt hat, sind in der ersten Zeile zu sehen (Hyperface). Es sei jedoch erwähnt, dass ein kompletter direkter Vergleich nicht ohne weiteres möglich ist, da aus Gründen der Vereinfachung das Region Proposal vernachlässigt wurde. Des Weiteren ist nicht klar wie die Normalisierungskonstante für den NME gewählt worden ist. Da die Landmarken der Augen nicht garantiert annotiert sind, ist es möglich, dass die inter-ocular-Distanz nicht bestimmt werden kann. Aus diesem Grund wird in der Literatur für den ALFW Benchmark die Gesichtsgröße verwendet, welche aus der Bounding Box entnommen werden kann. Oft ist dabei jedoch nicht klar, ob hierbei die Breite oder Höhe verwendet wurde. In dieser Arbeit wurde sich für die Breite der Bounding Box

Method	NME↓	FR↓	Seg.↑	Det.↑
Zhang et al. [ZLLT14b]	7.95	24.9	-	-
DAE [SSG+18]	5.45	-	-	-
MTL Modell-101 ($\eta = 0.01$)	4.68	5.70	97.50	99.0
MTL Modell-101 ($\eta = 0.001$)	3.52	2.40	99.10	99.10

Tabelle 5.4.4: Ergebnisse auf dem MAFL Protokoll.

Attribut	MTL Modell-101 ($\eta = 0.01$)	MTL Modell-101 ($\eta = 0.001$)
5 o Clock Shadow	89.10	90.90
Arched Eyebrows	74.40	80.30
Attractive	75.70	80.10
Bags under Eyes	81.20	80.00
Bald	97.20	97.30
Bangs	92.10	94.10
Big Lips	72.20	75.10
Big Nose	78.00	79.90
Black Hair	81.80	88.10
Blond Hair	91.40	92.90
Blurry	94.10	94.40
Brown Hair	80.60	81.30
Bushy Eyebrows	86.30	88.80
Chubby	93.00	94.20
Double Chin	94.10	94.50
Eyeglasses	96.10	97.70
Goatee	94.70	94.30
Gray Hair	95.50	95.30
Heavy Makeup	86.20	87.50
High Cheekbones	81.30	85.40
Male	92.20	95.60
Mouth slightly open	83.90	89.20
Mustache	94.40	96.10
Narrow Eyes	88.00	88.50
No Beard	88.00	92.50
Oval Face	71.90	75.10
Pale Skin	95.50	95.60
Pointy Nose	71.80	71.70
Receding Hairline	91.40	92.20
Rosy Cheeks	94.50	94.50
Sideburns	95.90	95.70
Smiling	87.60	88.50
Straight Hair	79.10	79.50
Wavy Hair	76.20	80.90
Wearing Earrings	81.10	85.90
Wearing Hat	97.10	98.20
Wearing Lipstick	87.30	89.50
Wearing Necklace	86.70	88.70
Wearing Necktie	93.50	95.40
Young	79.80	85.40
∅	86.78	88.76

Tabelle 5.4.5: Ergebnisse auf dem MAFL Protokoll für die Attributerkennung.

als Skalierungsnormalisierung entschieden. Der Fehler bezüglich der Posenschätzung ist als Mittelwert über die drei Achsen (s. Abbildung 5.1.4) angegeben. Im Gegensatz zur Arbeit von [RPC16] in der zufällig 1000 Bilder aus dem Datensatz zur Evaluierung gewählt wurden, werden hier 10% verwendet, welches somit um die 1800 Testbilder sind, womit eine bessere Schätzung der Fehlermetriken bestimmt werden kann.

In der zweiten Zeile sind die Ergebnisse eines Experimentes zu sehen in der die bisherige Lernrate von 0.001 verwendet wurde. Dabei ist ein zum State-of-the-Art vergleichsweise schlechterer NME zu beobachten. Bezüglich der Segmentierungs- und Detektionsgenauigkeit konnten relativ präzise Vorhersagen getroffen werden, wobei die Segmentierung jedoch schlechter als die Detektion funktioniert. Die Spalte Sichtbarkeit zeigt die Genauigkeit mit der das Modell vorhersagen kann, ob bestimmte Landmarken innerhalb des Bildes sichtbar sind. und wird berechnet mit

$$\mathrm{Acc}^{\mathrm{sichtb.}} = \frac{1}{N} \sum_{i}^{N} \frac{\sum_{j=1}^{m} \mathbb{1}_{y_{ij}=\hat{y}_{ij}}}{m}, \tag{5.4.1}$$

wobei m der Anzahl an annotierten Landmarken für ein bestimmtes Gesicht in Bild i entspricht und j eine konkrete Landmarke kennzeichnet. Da die Angabe, Landmarken nur dann zu annotieren falls diese sichtbar sind, vor allem bei 21 definierten Landmarken stark subjektiv sein kann, sind mit einer Genauigkeit von 89.61% hier akzeptable Ergebnisse zu sehen. Fälschlicherweise wurde in diesem Experiment bei der Berechnung der Pose diese nicht von der Darstellung in Grad in die Einheit Radiant umgerechnet, womit in diesem Fall die Zahlen somit nicht verglichen werden können.

Eine Verbesserung der Ergebnisse konnte durch das Anpassen der Lernrate erzielt werden. Die Resultate bei einer Verringerung der Lernrate auf 0.0001 sind in der 3. Zeile zu sehen. Es ist erkennbar, dass dieses Modell in den Aufgaben deutlich besser abschneidet als das Modell des vorherigen erwähnten Experimentes. Hierbei ist eine deutliche Steigerung der Leistung innerhalb der Landmarkendetektion von bis zu 1.25% bemerkbar. Eine zu große Lernrate führt dazu, dass zu große Änderungen an den Gewichten des Netzes durchgeführt werden, welches in der Theorie die Trainingsdauer reduzieren kann. Dies führt jedoch dazu, dass während des Trainings sehr große Schritte in der Optimierungslandschaft vorgenommen werden. Falls diese zu groß sind kann es vorkommen, dass das Modell Schwierigkeiten hat sich einem lokalen Optimum zu nähern. In geringfügiger Weise wird dies mithilfe des Adam Optimierungsalgorithmus behandelt, da für jeden Parameter eine individuelle Lernrate realisiert wird. Es sei an dieser Stelle erwähnt, dass die Lernrate auch direkt die Entwicklung der Varianzen beeinflusst. Diese Faktoren begründen möglicherweise die besseren Ergebnisse.

Im Vergleich zum State-of-the-Art in [RPC16] können ähnliche Werte für die Landmarkendetektion erreicht werden. Zwar ist ein leicht geringerer Fehler beobachtbar, wobei der Unterschied jedoch nicht signifikant ist. Bezüglich der Pose sind schlechtere Resultate zu sehen. Insgesamt liegt hier eine größere Differenz von 0.98 vor.

In der Arbeit von [RPC16] wurde eine naive Gewichtung der Gesamtkostenfunktion vorgenommen. Dabei wurden diese nach der subjektiven Einschätzung des Schwierigkeitsgrades gewählt. Diese Hyperparameter sind angeben mit

$$\begin{aligned} w_{detektion} &= 1 \\ w_{landmarken} &= 5 \\ w_{sichtbarkeit} &= 0.5 \\ w_{pose} &= 5 \\ w_{geschlecht} &= 2. \end{aligned}$$

Um das dynamische Lernen der Unsicherheiten in Relation zur heuristisch gewählten Gesamtkostenfunktion zu setzen, wurde ein Experiment durchgeführt, welches eine naive Kombination der einzelnen Kostenfunktionen ($\sum_i w_i L_i$) mit den oben genannten Hyperparametern verwendet. Die Ergebnisse hiervon sind in der letzten Zeile von Tabelle 5.4.6 vermerkt. Das dynamische Lernen der Unsicherheiten führt im Gegensatz dazu zu einer Verbesserung in dem Großteil der Aufgaben, wobei das Ergebnis für die Pose geringfügig um 0.21 schlechter ausfällt.

Insgesamt kann also somit behauptet werden, dass das dynamische Anpassen der Varianzen eine geeignete Methode ist mit der MTL innerhalb der Gesichtsanalyse angewendet werden kann. Zusätzlich wird dadurch eine aufwändige und kostenintensive Suche geeigneter Hyperparameter vermieden. Der direkte Vergleich zeigte außerdem, dass dadurch eine Verbesserung der Ergebnisse möglich ist, welches sich aus der direk-

Methode	NME↓	FR ↓	Seg.↑	Det.↑	Sichtb.↑	Pose↓	Geschlecht↑
HyperFace [RPC16]	2.93	-	-	-	-	4.95	-
MTL Modell ($\eta = 0.001$)	4.10	6.17	85.86	91.24	89.61	(14.41)	80.91
MTL Modell ($\eta = 0.0001$)	2.85	3.16	95.99	95.41	91.06	5.93	84.44
MTL Modell (Naiver Loss)	3.26	3.9	95.2	95.46	89,98	5.72	82.01

Tabelle 5.4.6: Ergebnisse auf dem AFLW Datensatz.

ten Integration in das Training begründen lässt. Da die Varianzen adaptiv mitgelernt werden, helfen diese bei der Skalierung der unterschiedlichen Kostenfunktionen.

5.4.5 *Einfluss des Wing Losses*

In diesem Abschnitt wird der Effekt des in Unterabschnitt 4.1.2 beschriebenen Wing Losses evaluiert. Dafür wurden Experimente durchgeführt in denen für die Landmarkenerkennung der Wing Loss eingesetzt wird. Die restlichen Rahmenbedingungen der Experimente werden wie bisher beibehalten. Da das Training auf dem CelebA Benchmark aufgrund der Größe des Datensatzes lange dauert, die Ergebnisse bezüglich der Landmarkendetektion schon beachtlich sind und mit dem MAFL Datensatz eine Untermenge dessen gegeben ist, wird auf diesen im Folgenden verzichtet.

Innerhalb von Tabelle 5.4.7 sind die Experimente aufgelistet, welche mit dem Wing Loss durchgeführt wurden. Für den direkten Vergleich sind in den jeweils nachfolgenden Zeilen die besten erreichten Ergebnisse mit dem L2 Loss dargestellt. Sowohl auf dem MTFL und MAFL Benchmark führte der Wing Loss zu deutlich schlechteren Ergebnissen. Bezüglich des MTFL Datensatzes konnte eine Verschlechterung des NME von 1.61 beobachtet werden, wobei die FR mit einer Differenz von 5.68% erheblich schlechter ausfällt. Auch auf dem MAFL Datensatz wurden mit dem Wing Loss schlechtere Ergebnisse mit einem Unterschied von 0.63% (NME) und 1.70% (FR) erzielt. Die Ergebnisse auf dem AFLW Benchmark sind in den letzten beiden Zeilen zu sehen. Hierfür wurden mit dem Wing Loss ähnliche Resultate erreicht wie mit der regulären L2 Kostenfunktion.

Die Experimente zeigen jedoch, dass die Integration des Wing Losses in den MTL Kontext zu keiner Verbesserung in der Landmarkendetektion geführt hat. Dabei ist erkennbar, dass der Wing Loss auf dem kleineren Datensatz deutlich schlechtere Ergebnisse produziert. Da auf den anderen Datensätzen der Unterschied zu den Vergleichsexperimenten geringer ist, liegt die Vermutung nahe, dass in diesem Fall das Modell öfters zu größeren Fehlern tendiert. Diese Hypothese wird unterstützt durch die hohe FR. Die Motivation des Wing Losses war der Fokus auf kleinere Fehler, sodass selbst bei einigermaßen akkuraten Landmarken, der Fehler für das Netz weiterhin relevant bleibt, um noch präzisere Vorhersagen zu treffen. Aufgrund des linearen statt quadratischen Teils der Kostenfunktion im Falle eines größeren Fehlers hat das Modell Schwierigkeiten größere Fehler im Training auszugleichen, welches sich auf dem Testdatensatz negativ bemerkbar macht. Auf den größeren Datensätzen scheint der Effekt dieses Problem weniger stark zu sein, sodass die Diskrepanz zwischen dem Ergebnissen des Wing Losses und des L2 Losses kleiner ist. Dies beruht auf der

Tatsache, dass die Landmarkendetektion mit den größeren Datensätzen einfacher zu lernen ist.

5.4.6 *Einfluss der semantischen Segmentierung*

Die Hauptmotivation des MTL ist das gleichzeitige Betrachten von semantisch verwandten Aufgaben, welches dem Modell erlaubt zusätzliche domänenspezifische Informationen zu extrahieren. Mit dem zusätzlichen Informationsgewinn ist das Modell theoretisch in der Lage auf den einzelnen Aufgaben bessere Resultate zu erzielen, da der Austausch zwischen den Aufgaben das Training positiv beeinflussen kann.

In diesem Abschnitt soll der Einfluss der synthetischen Hilfsaufgabe der semantischen Segmentierung näher betrachtet. Dafür wurden zusätzliche Experimente durchgeführt, in denen jeweils die semantische Segmentierung ausgelassen wurde, während die restlichen Faktoren der Experimente beibehalten wurden. Tabelle 5.4.8 zeigt den Vergleich zwischen den Experimenten mit der semantischen Segmentierung (+) und denen exklusive dieser (-), wobei die Experimente nach Datensätzen getrennt sind. Für den MTFL Datensatz zeigt sich, dass die semantische Segmentierung zu einer deutlichen Verbesserung in der Landmarkenerkennung führt. Bezüglich der Attributerkennung bleiben die Ergebnisse konstant. Eine mögliche Hypothese für die besseren Resultate ist die durch die Segmentierung eingeführte implizite Regularisierung des Feature Encoders, womit effektiv eine Einschränkung/Nebenbedingung für das Modell während des Trainings aufgestellt wird.

Auf dem MAFL Datensatz verhält sich die Landmarkendetektion ähnlich zueinander, wobei kein signifikanter Unterschied festzustellen ist. Jedoch lässt sich bei dem Experiment mit der semantischen Segmentierung eine Steigerung der Genauigkeit bei den Attributen erkennen.

Datensatz	NME↓	FR ↓
MTFL (Wing Loss)	5.91	11.79
MTFL (L2 Loss)	4.30	6.11
MAFL (Wing Loss)	4.15	4.10
MAFL (L2 Loss)	3.52	2.40
AFLW (Wing Loss)	2.81	3.22
AFLW (L2 Loss)	2.85	3.16

Tabelle 5.4.7: Vergleich zwischen den Ergebnissen mit dem Wing Loss und dem L2 Loss.

Bezüglich des AFLW Benchmarks führt das Nichtverwenden der semantischen Segmentierung zu einer erblichen Verschlechterung in der Landmarkenlokalisierung. Die Ellipse, welche mithilfe eines 3D Gesichtsmodells und der annotierten Landmarken bestimmt wurden, ergeben eine gute approximative Schätzung der Segmentierungsmaske, die für geometrische Aufgaben nützlich sind. Des Weiteren ist zu sehen, dass der Hilfstask bei der Schätzung der Pose von Vorteil ist. Jedoch kann gleichzeitig eine schlechtere Klassifikationsgenauigkeit bezüglich der Attribute (Geschlecht) beobachtet werden.

Zusammenfassend kann also gesagt werden, dass die geometrischen Informationen, welche in der semantischen Segmentierung enthalten sind, bei Lokalisierungsaufgaben einen positiven Einfluss ausüben. Das Modell ist weiterhin in der Lage, gegeben der schwach annotierten Gesichter, angemessen Gesicht von Hintergrund unterscheiden zu können, wobei dieses begrenzt wird durch die geometrische Form (konvexe Hülle, Ellipse) der automatisch annotierten Daten.

5.4.7 *Qualitative Ergebnisse*

Der folgende Abschnitt dient dazu einen Eindruck in die Ergebnisse für die Aufgaben zu geben, in denen mit Hilfe quantitativer Resultate eine subjektive Einschätzung der Präzision der Vorhersagen nur schwer möglich ist. Zu Beginn wird eine Auswahl von qualitativen Ergebnissen in der Landmarkenerkennung präsentiert. Abbildung 5.4.3 zeigt Bilder aus dem MAFL Datensatz für die geeignete Landmarken vorhergesagt werden können (NME $< 10\%$). Hierbei ist die Annotation in grün und die vom Modell vorhergesagten Landmarken in rot dargestellt. Es ist erkennbar, dass das gelernte Modell robust gegenüber Varianz im Ausdruck, Skalierung und Mimik der Gesichter

Datensatz	NME↓	FR ↓	∅Attribute ↑	Pose ↓
MTFL (-)	5.16	7.38	83.32	-
MTFL (+)	4.30	6.11	83.07	-
MAFL (-)	3.43	2.80	87.44	-
MAFL (+)	3.52	2.40	88.76	-
AFLW (-)	3.41	4.54	86.55	6.11
AFLW (+)	2.85	3.16	84.44	5.72

Tabelle 5.4.8: Vergleich zwischen Experimenten mit semantischer Segmentierung (+) als Hilfsausgabe und den Experimenten ohne die semantische Segmentierung.

ist. Zusätzlich generalisiert das Verfahren auf das in-the-wild Szenario, sodass unterschiedliche Hintergründe und Situationen, in denen die Bilder aufgenommen wurden, zu keiner Beeinträchtigung der Ergebnisse führen.

Innerhalb von Abbildung 5.4.4 sind einige Fehlerfälle visualisiert, in denen ein NME größer als 10% beobachtet wurde. Dabei fällt auf, dass eine extreme Rotation der Pose zu ungenauen Landmarken führt (c-d). Zusätzlich sind Gesichter, welche zum Teil verdeckt sind, fehleranfällig (a-b). Des Weiteren ist der NME ein strengeres Maß bei kleineren Gesichtern, da dadurch die inter-ocular-Distanz kleiner ausfällt, womit schon aus leichten Abweichungen höhere Fehler resultieren.

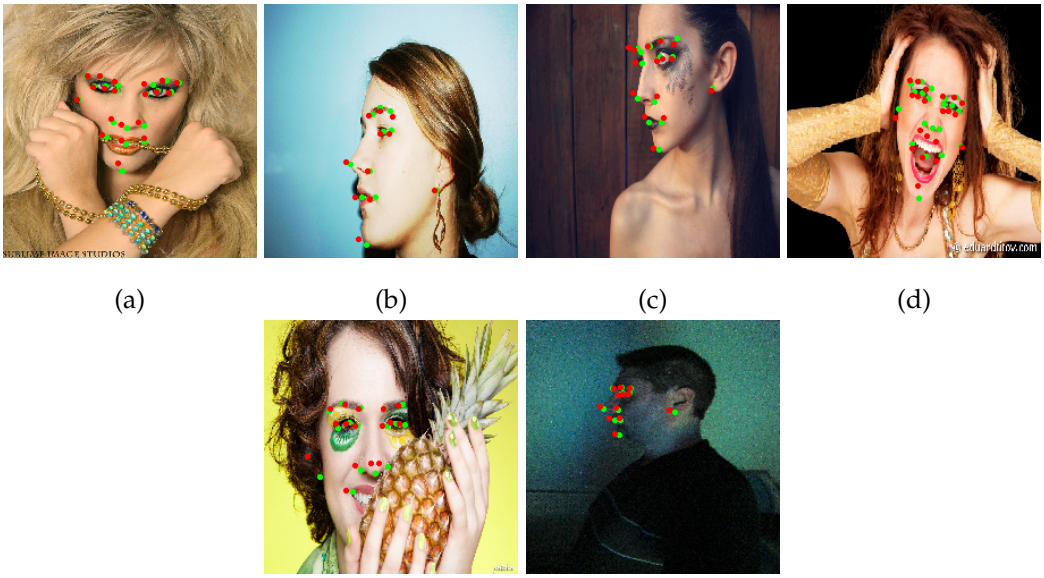
Für die qualitativen Ergebnisse bezüglich der Landmarkendetektion auf dem AFLW Datensatz sind jeweils nur die Landmarken, welche annotiert wurden, visualisiert. In Abbildung 5.4.5 sind Bilder gezeigt für die genaue Landmarken vorhergesagt werden. Hierbei ist erkennbar, dass das Modell selbst bei kompletter Verschiebung der Pose noch präzise Vorhersagen getroffen werden können (b, c, f). Im weiteren ist das Verfahren auf diesem Datensatz besser in der Lage mit zum Teil verdeckten Gesichtern umzugehen (e). Auch hier ist das Modell robust gegenüber starken Unterschieden in der Mimik einer Person (d).

Die Fälle, in denen ein erhöhter Fehler zu beobachten ist, sind in Abbildung 5.4.6 dargestellt. Dabei ist klar erkennbar, dass ein Großteil der Fehler auf die Tatsache zurückgeführt werden kann, dass durch das Auswahlkriterium der Bilder nicht garantiert ist, dass auf jedem Bild nur eine Person zu sehen ist. Interessanterweise fokussiert sich das Modell in den meisten Fällen (a-d) auf eines der zu sehenden Gesichter, wobei im letzten Bild (f) jedoch das Modell nicht in der Lage ist akkurate Landmarken für irgendein Gesicht zu erzielen. Im vorletztem Bild (e) führt die schlechte Auflösung des Bildes und das komplette Verdecken des Gesichtes zu Problemen, womit das Modell keine geeigneten Landmarken vorhersagen kann. Die meisten Fehler sind

Abbildung 5.4.3: Qualitative Ergebnisse für die Landmarkendetektion auf dem MAFL Datensatz. Hierbei ist die Annotation (grün) den Vorhersagen (rot) gegenübergestellt.

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

Abbildung 5.4.4: Testbilder des MAFL Datensatzes für die nach der FR keine adäquaten Landmarken vorhergesagt wurden.



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

(e) (f)

Abbildung 5.4.5: Qualitative Ergebnisse für die Landmarkendetektion auf dem AFLW Datensatz.

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

(e) (f)

Abbildung 5.4.6: Fehlerfälle bezüglich der Landmarkenerkennung auf dem AFLW Datensatz.

somit Sonderfälle oder Bilder mit mehreren Gesichtern, welche durch eine geeignete Behandlung, wie z.B. Region Proposal, umgangen werden können.

Abbildung 5.4.7 zeigt einige Resultate für die Gesichtsdetektion. In den Bildern ist jeweils die annotierte Bounding Box in blau darstellt, während die Vorhersage in rot angegeben ist. Die obere Reihe zeigt Beispiele aus dem MAFL Datensatz. Hierbei ist zu sehen, das eine akkurate Lokalisierung der Gesichter möglich ist. Auch auf dem AFLW Datensatz ist eine präzise Detektion von Gesichtern möglich (untere Reihe). Die Bilder zeigen, dass die Detektion auf unterschiedliche Kontexte generalisieren kann und robust gegenüber verschiedene Aufnahmebedingungen (Winkel, Skalierung, etc.) ist.

Mit Abbildung 5.4.8 sind einige Resultate für die semantische Segmentierung dargestellt. Dabei ist die Segmentierungsmaske, welche anzeigt ob ein Pixel zum Gesicht gehört, in weiß dargestellt. Links ist jeweils die vom Modell vorhergesagte Segmentierung und rechts die synthetische Annotation zu sehen. Die obere Reihe visualisiert Beispiele für den MAFL Datensatz. In der unteren Reihe sind Ergebnisse auf dem AFLW Benchmark dargestellt. Es ist erkennbar, dass das MTL Modell in der Lage ist, gegeben der schwach annotierten Daten, grob das Gesicht zu segmentieren. Besonders

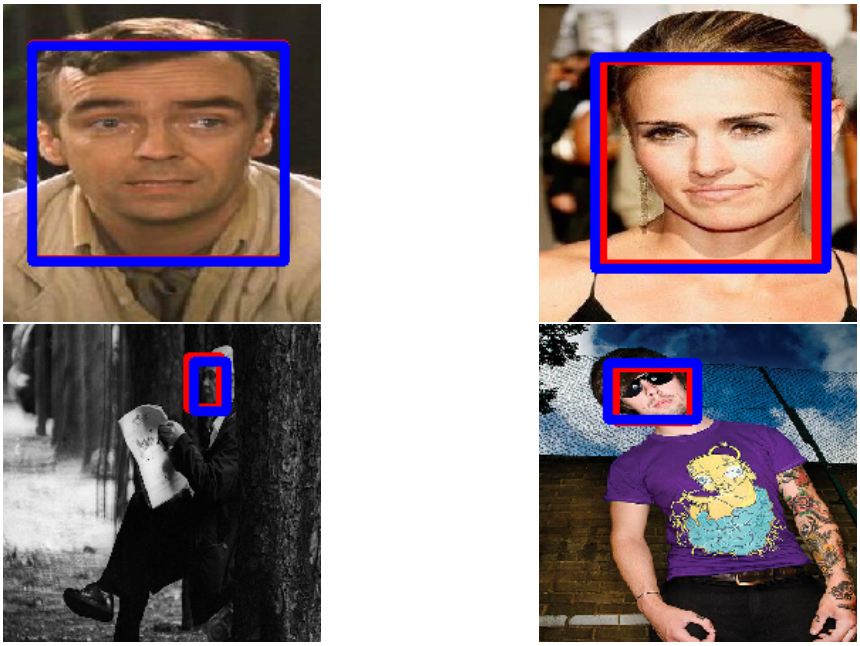


Abbildung 5.4.7: Qualitative Ergebnisse für die Gesichtsdetektion. Die obere Reihe enthält Bilder aus dem MAFL Datensatz, während die untere Reihe Bilder aus dem AFLW Benchmark darstellen.

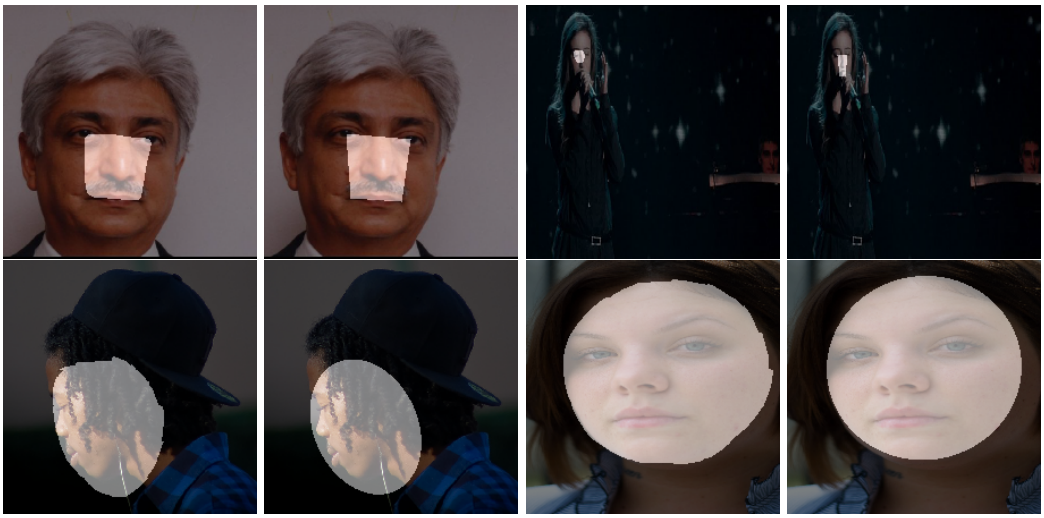


Abbildung 5.4.8: Qualitative Ergebnisse für die semantische Segmentierung. In der oberen Reihe sind aus dem MAFL Datensatz dargestellt, während die untere Reihe Bilder aus dem AFLW Benchmark visualisiert.

auf dem AFLW Datensatz sind die Vorhersagen aufgrund der geeigneteren Annotation eine angemessene Segmentierung der Gesichtspixel.

6 FAZIT

Das automatische Verarbeiten und Verstehen von Gesichtern umfasst zahlreiche Teilaufgaben. Mit Kapitel 3 wurde eine Auswahl von Problemen vorgestellt, welche in einen MTL Ansatz zusammen gelöst werden. Dabei wurde eine grundlegende Schwierigkeit im MTL diskutiert. Das Verhältnis der einzelnen Aufgaben zueinander ist bei der Optimierung ein wichtiger Faktor und wird öfters entweder heuristisch gewählt oder in einer aufwändigen Hyperparametersuche bestimmt. Stattdessen wurde mit dieser Arbeit ein aktuelles Verfahren für das tiefe MTL betrachtet, welches explizit die Unsicherheiten der Methode bezüglich der betrachteten Aufgaben modelliert und in das Training integriert. Diese Unsicherheiten werden während des Trainings dynamisch angepasst, womit das Modell mit einer Gesamtkostenfunktion trainiert wird, in der zusätzlich eine geeignete Beziehung zwischen den Aufgaben abhängig vom aktuellen Trainingsstand bezüglich einer Aufgabe gelernt wird. Statt eines mehrstufigen Optimierungsprozess, wie in [ZLLT14b], lässt sich die hier verwendete Methodik direkt in das Training integrieren. Nach bestem Wissen und Gewissen wurde dieses Verfahren in dieser Arbeit zum ersten Mal in der Gesichtsanalyse angewendet.

In der Evaluierung wurde gezeigt, dass hiermit der aktuelle State-of-the-Art auf dem MTFL [ZLLT14a] und MAFL [ZLLT14b] Benchmark überboten werden konnte. Auf dem vollständigen CelebA Datensatz konnten zwar die ursprünglichen Ergebnisse der Originalarbeit verbessert werden, jedoch lag die Erkennungsleistung der Attribute unter denen des aktuellen State-of-the-Art [GRB16]. Hierbei muss jedoch erwähnt werden, dass in der Arbeit von [GRB16] nur die Attributklassifikation betrachtet wurde. Des Weiteren sind dort unterschiedliche Formen der Datenaugmentierung verwendet worden, welches den Unterschied womöglich erklärt.

Auf dem AFLW Datensatz konnten ähnliche Ergebnisse wie in [RPC16] erzielt werden, wobei angemerkt werden muss, dass das verwendete Evaluierungsprotokoll anders aussah. In [RPC16] wurde für die Gewichtung der Aufgaben eine heuristische Gewichtung gewählt. Hierfür wurde analog ein Experiment durchgeführt mit den dort angegeben statischen Hyperparametern. Es zeigte sich, dass die hier verwendete dynamische Unsicherheitsgewichtung im Gegensatz dazu zu besseren Ergebnissen geführt hat. Zusammenfassend kann also festgestellt werden, dass mit der in dieser Arbeit verwendeten Methodik der Bedarf an sensiblen Hyperparametern reduziert wird, während gleichzeitig eine Verbesserung der Ergebnisse erreicht wird.

Mit der semantischen Segmentierung wurde analog zu [BHF+17] ein synthetischer Hilfstask in das MTL Modell eingebracht. Dieser wurde schwach überwacht mit einer automatischen Approximation der Annotation gelernt. Dabei zeigt sich in der qualitativen Analyse, dass das Modell in der Lage ist Gesicht und Hintergrund zu unterscheiden. Im weiteren zeigen die eigens dafür durchgeführten Experimente in denen die semantische Segmentierung weggelassen wurde, dass die in der semantischen Segmentierung enthaltenen geometrischen Information zu einer Verbesserung und in einem Fall zu ähnlichen Ergebnissen führen.

Zusätzlich wurde der Wing Loss, welcher empirisch gesehen nach [FKA+17] eine bessere Detektion von Landmarken erlaubt, im MTL Kontext betrachtet. Hierbei zeigten die Experimente jedoch, dass das Verwenden des Wing Losses zu keiner Verbesserung in der Landmarkendetektion geführt hat. Insbesondere auf dem MTFL und MAFL Datensatz wurden mit dem Wing Loss schlechtere Ergebnisse als mit dem L2 Loss erzielt. Auf dem AFLW Benchmark waren die Resultate ähnlich zueinander. Der Wing Loss ergab keine Verbesserung in der hier verwendeten Methodik.

Innerhalb dieser Arbeit wurde keine explizite Behandlung von Bildern mit mehreren Gesichtern durchgeführt. Dies ist jedoch keine direkte Einschränkung der hier verwendeten Methode. Beispielsweise kann wie in [RPC16] ein Region Proposal angewendet werden. Da dieses doch zu einer erhöhten Laufzeitkomplexität führt, wäre ein möglicher Ausblick das Kombinieren mit einem schnellerem Multi-Objekt Detektionsverfahren, wie z.B. das YOLO Framework (s. Abschnitt 3.2). Dafür könnte zusätzlich für jede Zelle die Menge an Vorhersagen erweitert werden, um zusätzlich zur Bounding Box unterschiedliche Aufgaben, wie z.B. Attribut- und Landmarkenerkennung zu integrieren.

Zusätzlich wurden jeweils in den einzelnen Experimenten nur Datensätze verwendet, welche hinsichtlich der Aufgaben vollständig annotiert waren. Jedoch tritt in der Praxis eher der Fall auf, dass für verschiedene Datensätze unterschiedliche Informationen vorliegen. Diese können auf trivialer Weise nur dann zusammen betrachtet werden, falls diese vollständig mit denselben Aufgaben annotiert wurden. Falls möglich, können diese unter erheblichem Aufwand nachträglich erstellt werden. Eine weitere Option ist eine Form des dynamischen Lernens, in dem innerhalb der Gesamtkostenfunktion bestimmte Aufgaben ignoriert werden, falls diese für ein Trainingsbeispiel nicht vorhanden sind. Interessant wäre zu untersuchen, wie sich dies mit der Unsicherheitsgewichtung kombinieren lässt.

LITERATURVERZEICHNIS

[AJSHAR+17] Abdullah, Nurul A. ; Jamri Saidi, Md ; Hidayah Ab Rahman, Nurul ; Chuah, Chai W. ; A Hamid, isredza r.: Face recognition for criminal identification: An implementation of principal component analysis for face recognition, 2017, S. 020002

[BHF+17] Bischke, Benjamin ; Helber, Patrick ; Folz, Joachim ; Borth, Damian ; Dengel, Andreas: Multi-Task Learning for Segmentation of Building Footprints with Deep Neural Networks. In: *CoRR* abs/1709.05932 (2017). http://arxiv.org/abs/1709.05932

[Bis07] Bishop, Christopher M.: *Pattern recognition and machine learning, 5th Edition*. Springer, 2007 (Information science and statistics). http://www.worldcat.org/oclc/71008143. – ISBN 9780387310732

[BT16] Bulat, Adrian ; Tzimiropoulos, Georgios: Human pose estimation via Convolutional Part Heatmap Regression. In: *CoRR* abs/1609.01743 (2016). http://arxiv.org/abs/1609.01743

[Car93] Caruana, Rich: Multitask Learning: A Knowledge-based Source of Inductive Bias. In: *Proceedings of the Tenth International Conference on International Conference on Machine Learning*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1993 (ICML'93). – ISBN 1–55860–307–7, 41–48

[CET98] Cootes, Timothy F. ; Edwards, Gareth J. ; Taylor, Christopher J.: Active Appearance Models. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Springer, 1998, S. 484–498

[CHM+14] Choromanska, Anna ; Henaff, Mikael ; Mathieu, Michaël ; Arous, Gérard Ben ; LeCun, Yann: The Loss Surface of Multilayer Networks. In: *CoRR* abs/1412.0233 (2014). http://arxiv.org/abs/1412.0233

[DNRW13] De Marsico, M. ; Nappi, M. ; Riccio, D. ; Wechsler, H.: Robust Face Recognition for Uncontrolled Pose and Illumination Changes. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 43 (2013), Jan,

Nr. 1, S. 149–163. http://dx.doi.org/10.1109/TSMCA.2012.2192427. – DOI 10.1109/TSMCA.2012.2192427. – ISSN 2168–2216

[DV16] Dumoulin, Vincent ; Visin, Francesco: A guide to convolution arithmetic for deep learning. In: *arXiv e-prints* (2016), Mar, S. arXiv:1603.07285

[DYOY18] Dong, Xuanyi ; Yan, Yan ; Ouyang, Wanli ; Yang, Yi: Style Aggregated Network for Facial Landmark Detection. In: *CoRR* abs/1803.04108 (2018). http://arxiv.org/abs/1803.04108

[FKA+17] Feng, Zhen-Hua ; Kittler, Josef ; Awais, Muhammad ; Huber, Patrik ; Wu, Xiaojun: Wing Loss for Robust Facial Landmark Localisation with Convolutional Neural Networks. In: *CoRR* abs/1711.06753 (2017). http://arxiv.org/abs/1711.06753

[GBC16] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning*. MIT Press, 2016. – http://www.deeplearningbook.org

[GHH+17] Günther, Manuel ; Hu, Peiyun ; Herrmann, Christian ; Chan, Chi-Ho ; Jiang, Min ; Yang, Shufan ; Dhamija, Akshay R. ; Ramanan, Deva ; Beyerer, Jürgen ; Kittler, Josef ; Jazaery, Mohamad A. ; Nouyed, Mohammad I. ; Stankiewicz, Cezary ; Boult, Terrance E.: Unconstrained Face Detection and Open-Set Face Recognition Challenge. In: *CoRR* abs/1708.02337 (2017). http://arxiv.org/abs/1708.02337

[GOO+17] Garcia-Garcia, Alberto ; Orts-Escolano, Sergio ; Oprea, Sergiu ; Villena-Martinez, Victor ; Rodríguez, José García: A Review on Deep Learning Techniques Applied to Semantic Segmentation. In: *CoRR* abs/1704.06857 (2017). http://arxiv.org/abs/1704.06857

[GRB16] Günther, Manuel ; Rozsa, Andras ; Boult, Terrance E.: AFFACT - Alignment Free Facial Attribute Classification Technique. In: *CoRR* abs/1611.06158 (2016). http://arxiv.org/abs/1611.06158

[HTF01] Hastie, Trevor ; Tibshirani, Robert ; Friedman, Jerome: *The Elements of Statistical Learning*. New York, NY, USA : Springer New York Inc., 2001 (Springer Series in Statistics)

[HZRS15a] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Deep Residual Learning for Image Recognition. In: *CoRR* abs/1512.03385 (2015). http://arxiv.org/abs/1512.03385

[HZRS15b] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In: *CoRR* abs/1502.01852 (2015). http://arxiv.org/abs/1502.01852

[IS15] Ioffe, Sergey ; Szegedy, Christian: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: *CoRR* abs/1502.03167 (2015). http://arxiv.org/abs/1502.03167

[JT16] Jin, Xin ; Tan, Xiaoyang: Face Alignment In-the-Wild: A Survey. In: *CoRR* abs/1608.04188 (2016). http://arxiv.org/abs/1608.04188

[KB15] Kingma, Diederik P. ; Ba, Jimmy: Adam: A Method for Stochastic Optimization. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015

[KG17] Kendall, Alex ; Gal, Yarin: What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In: *CoRR* abs/1703.04977 (2017). http://arxiv.org/abs/1703.04977

[KGC17] Kendall, Alex ; Gal, Yarin ; Cipolla, Roberto: Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In: *CoRR* abs/1705.07115 (2017). http://arxiv.org/abs/1705.07115

[KWRB11] Köstinger, M. ; Wohlhart, P. ; Roth, P. M. ; Bischof, H.: Annotated Facial Landmarks in the Wild: A large-scale, real-world database for facial landmark localization. In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, S. 2144–2151

[LJL14] Liao, Shengcai ; Jain, Anil K. ; Li, Stan Z.: A Fast and Accurate Unconstrained Face Detector. In: *CoRR* abs/1408.1656 (2014). http://arxiv.org/abs/1408.1656

[LJY19] Li, Feifei ; Johnson, Justin ; Yeung, Serena: *CS231n: Convolutional Neural Networks for Visual Recognition*. URL: http://cs231n.stanford.edu/syllabus.html, 2019. – Vorlesung, Zuletzt aufgerufen am 15.04.2019

[LK18] Liebel, Lukas ; Körner, Marco: Auxiliary Tasks in Multi-task Learning. In: *CoRR* abs/1805.06334 (2018). http://arxiv.org/abs/1805.06334

[LLWT] Liu, Ziwei ; Luo, Ping ; Wang, Xiaogang ; Tang, Xiaoou: *Large-scale CelebFaces Attributes (CelebA) Dataset*. http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html Zuletzt aufgerufen am 24.06.2019

[LLWT14] Liu, Ziwei ; Luo, Ping ; Wang, Xiaogang ; Tang, Xiaoou: Deep Learning Face Attributes in the Wild. In: *CoRR* abs/1411.7766 (2014). http://arxiv.org/abs/1411.7766

[LOW+18] Liu, Li ; Ouyang, Wanli ; Wang, Xiaogang ; Fieguth, Paul W. ; Chen, Jie ; Liu, Xinwang ; Pietikäinen, Matti: Deep Learning for Generic Object Detection: A Survey. In: *CoRR* abs/1809.02165 (2018). http://arxiv.org/abs/1809.02165

[LSD14] Long, Jonathan ; Shelhamer, Evan ; Darrell, Trevor: Fully Convolutional Networks for Semantic Segmentation. In: *CoRR* abs/1411.4038 (2014). http://arxiv.org/abs/1411.4038

[LWL+17] Liu, Weibo ; Wang, Zidong ; Liu, Xiaohui ; Zeng, Nianyin ; Liu, Yurong ; Alsaadi, Fuad E.: A survey of deep neural network architectures and their applications. In: *Neurocomputing* 234 (2017), 11 - 26. http://dx.doi.org/https://doi.org/10.1016/j.neucom.2016.12.038. – DOI https://doi.org/10.1016/j.neucom.2016.12.038. – ISSN 0925–2312

[MP43] McCulloch, Warren S. ; Pitts, Walter: A logical calculus of the ideas immanent in nervous activity. In: *The bulletin of mathematical biophysics* 5 (1943), Dec, Nr. 4, 115–133. http://dx.doi.org/10.1007/BF02478259. – DOI 10.1007/BF02478259. – ISSN 1522–9602

[MP69] Minsky, Marvin ; Papert, Seymour: *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA : MIT Press, 1969

[MRR18] Merget, Daniel ; Rock, Matthias ; Rigoll, Gerhard: Robust Facial Landmark Detection via a Fully-Convolutional Local-Global Context Network. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018

[Nie18] Nielsen, Michael A.: *Neural Networks and Deep Learning*. Version: 2018. http://neuralnetworksanddeeplearning.com/chap5.html Zuletzt aufgerufen am 30.04.2019

[NYD16] Newell, Alejandro ; Yang, Kaiyu ; Deng, Jia: Stacked Hourglass Networks for Human Pose Estimation. In: *CoRR* abs/1603.06937 (2016). http://arxiv.org/abs/1603.06937

[PCMY15] Papandreou, George ; Chen, Liang-Chieh ; Murphy, Kevin ; Yuille, Alan L.: Weakly- and Semi-Supervised Learning of a DCNN for Semantic Image Segmentation. In: *CoRR* abs/1502.02734 (2015). http://arxiv.org/abs/1502.02734

[PHZ+18] Pei, Yanting ; Huang, Yaping ; Zou, Qi ; Zang, Hao ; Zhang, Xingyuan ; Wang, Song: Effects of Image Degradations to CNN-based Image Classification. In: *CoRR* abs/1810.05552 (2018). http://arxiv.org/abs/1810.05552

[RBFL15] Russakovsky, Olga ; Bearman, Amy L. ; Ferrari, Vittorio ; Li, Fei-Fei: What's the point: Semantic segmentation with point supervision. In: *CoRR* abs/1506.02106 (2015). http://arxiv.org/abs/1506.02106

[RDGF15] Redmon, Joseph ; Divvala, Santosh K. ; Girshick, Ross B. ; Farhadi, Ali: You Only Look Once: Unified, Real-Time Object Detection. In: *CoRR* abs/1506.02640 (2015). http://arxiv.org/abs/1506.02640

[RHW86] Rumelhart, David E. ; Hinton, Geoffrey E. ; Williams, Ronald J.: Learning representations by back-propagating errors. In: *Nature* 323 (1986), Oktober, 533–. http://dx.doi.org/10.1038/323533a0

[Ros58] Rosenblatt, F.: The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. In: *Psychological Review* (1958), S. 65–386

[RPC16] Ranjan, Rajeev ; Patel, Vishal M. ; Chellappa, Rama: HyperFace: A Deep Multi-task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition. In: *CoRR* abs/1603.01249 (2016). http://arxiv.org/abs/1603.01249

[Rud17] Ruder, Sebastian: An Overview of Multi-Task Learning in Deep Neural Networks. In: *CoRR* abs/1706.05098 (2017). http://arxiv.org/abs/1706.05098

[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), 1929-1958. http://jmlr.org/papers/v15/srivastava14a.html

[SLL16] Saito, Shunsuke ; Li, Tianye ; Li, Hao: Real-Time Facial Segmentation and Performance Capture from RGB Input. In: *CoRR* abs/1604.02647 (2016). http://arxiv.org/abs/1604.02647

[SSG+18] Shu, Zhixin ; Sahasrabudhe, Mihir ; Güler, Riza A. ; Samaras, Dimitris ; Paragios, Nikos ; Kokkinos, Iasonas: Deforming Autoencoders: Unsupervised Disentangling of Shape and Appearance. In: *CoRR* abs/1806.06503 (2018). http://arxiv.org/abs/1806.06503

[SWT13] Sun, Yi ; Wang, Xiaogang ; Tang, Xiaoou: Deep Convolutional Network Cascade for Facial Point Detection. In: *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA : IEEE Computer Society, 2013 (CVPR '13). – ISBN 978–0–7695–4989–7, 3476–3483

[SZ15] Simonyan, K. ; Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *International Conference on Learning Representations*, 2015

[TND10] Thorat, S. B. ; Nayak, S. K. ; Dandale, Jyoti P.: Facial Recognition Technology: An analysis with scope in India. In: *CoRR* abs/1005.4263 (2010). http://arxiv.org/abs/1005.4263

[USGS13] Uijlings, J. R. ; Sande, K. E. ; Gevers, T. ; Smeulders, A. W.: Selective Search for Object Recognition. In: *Int. J. Comput. Vision* 104 (2013), September, Nr. 2, 154–171. http://dx.doi.org/10.1007/s11263-013-0620-5. – DOI 10.1007/s11263–013–0620–5. – ISSN 0920–5691

[VCE+07] Vural, Esra ; Cetin, Mujdat ; Ercil, Aytul ; Littlewort, Gwen ; Bartlett, Marian ; Movellan, Javier: Drowsy Driver Detection Through Facial Movement Analysis. In: *Proceedings of the 2007 IEEE International Conference on Human-computer Interaction*. Berlin, Heidelberg : Springer-Verlag, 2007 (HCI'07). – ISBN 3–540–75772–4, 978–3–540–75772–6, 6–18

[WJ18] Wu, Yue ; Ji, Qiang: Facial Landmark Detection: a Literature Survey. In: *CoRR* abs/1805.05563 (2018). http://arxiv.org/abs/1805.05563

[WLL18] WANG, Chien-Chih ; LOONG TAN, Kent ; LIN, Chih-Jen: Newton Methods for Convolutional Neural Networks. In: *arXiv e-prints* (2018), Nov, S. arXiv:1811.06100

[ZF13] ZEILER, Matthew D. ; FERGUS, Rob: Visualizing and Understanding Convolutional Networks. In: *CoRR* abs/1311.2901 (2013). http://arxiv.org/abs/1311.2901

[ZGH+18] ZHENG, Xin ; GUO, Yanqing ; HUANG, Huaibo ; LI, Yi ; HE, Ran: A Survey to Deep Facial Attribute Analysis. In: *CoRR* abs/1812.10265 (2018). http://arxiv.org/abs/1812.10265

[ZLLT14a] ZHANG, Zhanpeng ; LUO, Ping ; LOY, Chen C. ; TANG, Xiaoou: Facial landmark detection by deep multi-task learning. In: *In ECCV. 94–108*, 2014

[ZLLT14b] ZHANG, Zhanpeng ; LUO, Ping ; LOY, Chen C. ; TANG, Xiaoou: Learning Deep Representation for Face Alignment with Auxiliary Attributes. In: *CoRR* abs/1408.3967 (2014). http://arxiv.org/abs/1408.3967