Universität Bremen, Fachbereich 3: Mathematik und Informatik

Master Thesis

Künstliche Neuronale Netze zur Unterstützung von Bildredaktionen durch Kombination von Bild- und Textverarbeitung

Maira Weidenbach
3170774

27. August 2021

Erstgutachter: Prof. Dr.-Ing. Udo Frese
Zweitgutachter: Dr.-Ing. Robert Porzel

Zusammenfassung

In dieser Arbeit wird ein prototypisches System entworfen und evaluiert, welches für einen gegebenen Text passende Bilder aus einer unsortierten Datenbank suchen kann. Basierend auf zwei Neuronalen Netzen werden Informationen aus rohen Bild- und Textdaten extrahiert und in einen gemeinsamen Vektorraum projiziert. Innerhalb dieses Raumes kann die Ähnlichkeit der Bilder und Texte berechnet werden und darauf basierend die passendsten Bilder zu einem Text vorgeschlagen werden. Trainiert werden die beiden Netze parallel in Abhängigkeit voneinander unter Verwendung eines Contrastive Loss. Bei der Anwendung können die beiden Netze separat verwendet werden und somit eine Abfrage in Echtzeit ermöglichen. Das System wird auf Rezeptdaten der Chefkoch GmbH trainiert und evaluiert. Durch eine Umfrage bei einer potenziellen Zielgruppe kann bestätigt werden, dass in 80% der Fälle zu einem gegebenen Rezept mindestens ein passendes Bild aus der Datenbank gefunden wird.

Abstract

In this work, a prototypical system is designed and evaluated that can search for relevant images for a given text in an unsorted database. Based on two neural networks, information is extracted from raw image and text data and projected into a joint vector space. Within this space, the similarity of images and texts can be calculated and based on this, the most relevant images to a text can be suggested. The two networks are trained in parallel in dependence on each other using a contrastive loss. In the application, the two networks can be used separately, thus enabling real-time querying. The system is trained and evaluated on recipe data from the Chefkoch GmbH. Through a survey of a potential target group, it can be confirmed that in 80% of cases at least one relevant image from the database is retrieved for a given recipe.

Inhaltsverzeichnis

1 Einleitung

„Ein Bild sagt mehr als tausend Worte“ (Kurt Tucholsky).

Bilder sind wichtiger Bestandteil von Artikeln durch die Leser und Leserinnen entscheiden ob sie einen Artikel lesen oder nicht. Ein passendes Bild zu einem Artikel zu finden ist außerdem wichtig, da ein Bild die Beurteilung von Sachverhalten beeinflusst und zum Teil eine stärkere Wirkung erzielen kann als der Text alleine (Kessler et al., 2016). Auch bei Rezepten können Bilder eine wichtige Rolle spielen. Wie beispielsweise bei dieser Zubereitungsanweisung für einen Erdbeer Schoko Quark[1]:

> „Die Erdbeeren putzen und pürieren. Die Schokolade mit dem Puderzucker in der Milch schmelzen und abkühlen lassen und unter den Quark ziehen. Die Sahne steif schlagen und ebenfalls unterheben. Die Creme und das Püree in Gläser schichten und noch 2 Stunden kühl stellen.“

Diese Zubereitungsanweisung beschreibt grundsätzlich sehr genau, was zu tun ist. Außerdem ist sie so formuliert, dass der Leser vermutlich Lust auf der Ergebnis bekommt. Es wird allerdings nicht klar, warum der Quark in einzelne Gläser geschichtet werden soll. Geschmacklich wird es vermutlich keinen Unterschied geben, ob der Quark in kleinen Gläsern oder in einer Schüssel serviert wird. Wenn hingegen das Bild 1.1 dazu präsentiert wird, assoziiert der Leser dieses vermutlich mit Sommer und Leichtigkeit. Die Assoziation könnte eine Inspiration für ein Dessert für eine Gartenparty sein. So entsteht aus einer Visualisierung eine Assoziation, die inspirierend sein kann.

Heutzutage können sich Bildredakteure großer Bilddatenbanken bedienen um geeignete Fotos für ihre Artikel zu finden, wenn kein explizites Foto aufgenommen werden soll. Allein die Internetplattform „Shutterstock“ bietet über 350 Millionen Bilder zum Verkauf an (Shutterstock, 2021). Die Plattform „pixabay“ bietet über zwei Millionen lizenzfreie Bilder an (pixabay, 2021). Die Deutsche Presse Agentur (dpa) gibt an, in ihrem eigenem Bilderarchiv im Jahr 2020 ungefähr 21 Millionen Bilder gespeichert zu haben. Bei dem Tochterunternehmen, picture alliance, seien es sogar rund 85 Millionen (dpa, 2020).

[1] Bestandteil des Datensatzes den das Unternehmen Chefkoch GmbH zur Verfügung stellt.

Abbildung 1.1: Erdbeer Schoko Quark[2]

1.1 Motivation

Es existieren demzufolge schon Millionen an digitalen Bildern, die für bestimmte Zwecke aufgenommen wurden. Um diese Bilder wiederverwenden zu können, müssen sie jedoch abrufbar sein. Die Herausforderung ist es also, in einer Datenbank aus mehreren tausend Bildern ein passendes Bild für einen entsprechenden Artikel zu finden. Die Plattformen „Shutterstock“ und „pixabay“ können beispielsweise mithilfe von Schlüsselwörtern durchsucht werden. Allerdings ist dies nicht immer möglich, denn andere, private Datenbanken sind häufig nicht so gut organisiert. Ein Beispiel ist die hauseigene Bilderdatenbank bei Gruner + Jahr[3] in der Sparte Essen und Trinken, die über Jahre gefüllt wurde. Deren Bilder wurden weder mit Schlüsselwörtern markiert, noch besitzen sie selbstbeschreibende Dateinamen. Ein maschineller Zugriff auf die Bilder ist daher nicht möglich. Sie könnten zwar am Computer angezeigt und einzeln von Menschen durchgegangen werden, jedoch ist dies ineffizient. Selbst wenn ein Mensch jedes Bild für nur eine Sekunde betrachten würde, ergäbe dies eine Viertelstunde für 1.000 Bilder und zweieinhalb Stunden für 10.000 Bilder. Um solche Datenbanken effizienter zu gestalten, wird ein System benötigt, dass die Bilder maschinell durchsuchen kann. Dies könnte eine wertvolle Unterstützung für Bildredakteure bei der Verwendung von archivierten Bildern darstellen. Die Bildersuche (engl. Image Retrieval) ist eine interdisziplinäre Herausforderung, die in der Expertise von Datenbanken, Bild- und Texterkennung und maschinellem Lernen vereint wird (T.karthikeyan and aprabhu, 2014).

[2]Bestandteil des Datensatzes den das Unternehmen Chefkoch GmbH zur Verfügung stellt.
[3]`https://www.guj.de/`

1.2 Beitrag der Arbeit

Deshalb wird in dieser Arbeit ein prototypisches System vorgestellt, das Bildredakteure bei der Durchsuchung von Bilder-Datenbanken unterstützen soll. Durch die Verarbeitung von textuellen Beiträgen sollen adäquate Bilder gefunden und angezeigt werden. Für die Bild- und Texterkennung enthält das System Komponenten des maschinellen Lernens, die neu miteinander verknüpft werden, um maschinelle Suchvorgänge zu ermöglichen. Hierfür werden Neuronale Netze verwendet.

Für diesen Vorgang wird ein prototypisches Such- und Empfehlungssystem konzipiert, welches die Wiederverwertbarkeit von Bildern gewährleistet. Das System baut auf vorhandene Datenbanken auf. Durch die theoretische Fundierung und die Konzeptionierung und Evaluation des Prototyps wird in dieser Arbeit die folgende Fragestellung bearbeitet und beantwortet:

> Wie kann ein System aussehen, das einen Bildredakteur bei der Aufgabe archivierte Bilder aus hauseigenen oder internen unorganisierten Datenbanken wiederzuverwenden unterstützt?

Um diese Frage zu beantworten, werden zunächst die folgenden Teilfragen beantwortet:

1. Was bedeutet Unterstützung in dieser Arbeit?
2. Welche Anforderungen an das System ergeben sich daraus?
3. Wie kann ein solches System konkret aussehen?

Da für dieses System Neuronale Netze genutzt werden, werden für das Training dieser und der Evaluierung des Systems sehr viele Daten benötigt. Daraus ergeben sich die folgenden Fragen, die es zu beantworten gilt:

4. Wie müsste ein Datensatz aussehen, der geeignet wäre ein solches System zu trainieren?
5. Wie kann das System evaluiert werden?

1.3 Aufbau der Arbeit

Im folgenden Kapitel werden die theoretischen Grundlagen erläutert. Es werden die grundlegenden Begrifflichkeiten von Neuronalen Netzen, der Verarbeitung natürlicher Sprache und die Verarbeitung von Bildern eingeführt und weitere, für diese Arbeit relevante, Begriffe definiert. Nachfolgend wird auf die erste Teilfrage und auf ähnliche Arbeiten im Bereich des Suchens und Abrufens von Bildern eingegangen. In Kapitel

4 und 5 wird die Konzeption erarbeitet und auf die Teilfragen 2-3 sowie auf die Implementierung eingegangen. Die Datensätze, auf die sich die 4. Frage bezieht und mit denen das System trainiert und getestet wird, werden in Kapitel 6 vorgestellt. Darauf folgend werden die Experimente zur Evaluation und somit die Bearbeitung der 5. Frage, der Konstruktion, dargestellt und diskutiert. Am Ende der Arbeit wird die Frage, ob das konstruierte Such- und Empfehlungssystem einen Bildredakteur bei der Aufgabe, archivierte Bilder aus unorganisierten Datenbanken wiederzuverwenden, unterstützten kann, beantwortet. Abschließend wird ein Ausblick für weitere mögliche Forschungsansätze gegeben.

1.4 Haftungsausschluss und Geschlechtergerechte Sprache

Die Rezeptbilder und Rezepttexte, die in dieser Arbeit zu Demonstrationszwecken verwendet werden, sind Eigentum der Chefkoch GmbH und wurden Frau Maira Weidenbach im Rahmen dieser Arbeit zur Verfügung gestellt. Alle Rechte an diesem Bild- und Textmaterial gehören und verbleiben somit bei der Chefkoch GmbH.

In dieser Arbeit wird aus Gründen der besseren Lesbarkeit das generische Maskulinum verwendet. Es wird an dieser Stelle darauf hingewiesen, dass weibliche und anderweitige Geschlechteridentitäten dabei ausdrücklich mitgemeint sind.

2 Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen und Zusammenhänge erläutert und in ihren Kontext eingeordnet. Der erste Teil beschreibt die Inhalte zur Bildverarbeitung, der zweite Teil notwendige Informationen zur Textverarbeitung und im dritten Abschnitt wird auf Aspekte des Trainings Neuronaler Netze eingegangen. Zuvor werden Begriffe definiert, die in wissenschaftlichen Arbeiten unterschiedlich verwendet werden. Ohne Beschränkung der Allgemeinheit gelten die folgenden Definitionen im Rahmen dieser Arbeit.

Dezimal- und Tausendertrennzeichen: In dieser Arbeit wird als Dezimaltrennzeichen ein Komma und als Tausendertrennzeichen ein Punkt verwendet. Beispiel: $0{,}25 = \frac{1}{4}$ und $1.000 = 1000$.

Feature-Vektor: Der Feature-Vektor wird in dieser Arbeit als Ausgabevektor der Neuronalen Netze definiert. Die hier beschriebenen und verwendeten Netze extrahieren Informationen aus ihren Eingaben und produzieren immer einen Ausgabevektor der Form $1 \times n$ mit $n \in \mathbb{N}$. Dieser Ausgabevektor wird im weiteren Verlauf Feature-Vektor genannt.

Batch: Ein Batch ist eine echte Teilmenge des gesamten Datensatzes und wird verwendet, um mehrere Daten parallel zu berechnen. Ein Text-Batch ist dementsprechend eine Teilmenge des Text-Datensatzes und ein Bild-Batch eine Teilmenge des Bild-Datensatzes.

Schicht: Eine Schicht beschreibt, in der Vorstellung Neuronaler Netze als biologische Neuronen, eine Reihe von Neuronen auf einer Ebene. In dieser Arbeit wird diese Betrachtungsweise zu Demonstrationszwecken angenommen.

Grundwahrheit: Eine Grundwahrheit ist eine Information, die für den verwendeten Kontext als wahr angenommen wird, ohne dass diese in einem anderen Kontext wahr sein muss oder eine allgemeingültige, einzig wahre Wahrheit darstellt.

In dieser Arbeit werden zwei Qualitätsmaße aus dem Bereich der Informationsabfrage (engl. Information Retrieval) als Metriken verwendet. Zum einen *Trefferquote* (engl. recall) und zum anderen *Genauigkeit* (engl. precision). Die Trefferquote beschreibt die Wahrscheinlichkeit mit der ein positives Ereignis als positiv klassifiziert wird. Die Genauigkeit gibt den Anteil an richtig positiv klassifizierten unter den falsch positiv klassifizierten Ereignissen an (Zweig et al., 2017).

2.1 Digitale Verarbeitung von Bildern

Anders als analoges Filmmaterial, können digitale Bilder auf einem Computer nicht kontinuierlich verarbeitet werden. Das bedeutet sie müssen diskretisiert werden. Die einzelnen Pixel repräsentieren somit nicht nur einen Bildpunkt, sondern einen Mittelwert über eine Region. Je mehr Pixel zur Verfügung stehen, desto höher ist die Auflösung. Mit steigender Auflösung steigt die Verarbeitungsdauer. Um diese gering zu halten, werden nicht mehr Pixel als nötig betrachtet. Das kleinste zu untersuchende Objekt muss daher von mehr als einem Pixel beschrieben werden. Um Objekte analysieren und identifizieren zu können, ist ein mehrstufiger Prozess nötig. Dieser besteht aus einer Kette von Bildverarbeitungsoperationen unter anderem der Mittelung einzelner Pixel, der Kantendetektion, der Analyse der Nachbarschaften und der Segmentierung. Hierfür werden geeignete Filteroperationen oder auch Konvolutionen durchgeführt (Jähne, 2012). Die Idee von mehrstufigen Prozessen und Filtern wurde bereits 1989 von LeCun et al. in einem Neuronalen Netz vereint, welches handgeschriebene US-amerikanische Postleitzahlen erkennt. Dieses Netz besteht aus zwei verdeckten Schichten (engl. Hidden Layer), von denen zwei stark lokal eingeschränkte Schichten sind. Diese greifen die Idee der Filter auf, lokal nach bestimmten Merkmalen zu suchen, unabhängig von der exakten Ortung. Dies geschieht auf Basis geteilter Gewichte an den Verbindungen, was auch als nicht-lineare teilabgetastete (subsample) Konvolution mit einem Kernel bestimmter Größe interpretiert werden kann. Auf diese Art entstehen sogenannte Merkmalskarten. Innerhalb einer Merkmalskarte werden die Gewichte geteilt. Jede Merkmalskarte besitzt eigene Gewichte, wodurch dasselbe Merkmal von einer Merkmalskarte an jeder Stelle gefunden wird. Dadurch bestimmt die Anzahl an Merkmalskarten die Anzahl der betrachteten Merkmale. Die exakten Gewichte unterliegen hierbei dem Trainingsprozess (LeCun et al., 1989). Dies steht im Gegensatz zu den konventionellen Filtern, deren Werte feststehen, wie beispielsweise bei dem Sobel-Kantendetektor (Jähne, 2012).

Ein Convolutional Neural Network (CNN) enthält mindestens eine Konvolutionsschicht, die Merkmalskarten berechnet. Ein weiterer wichtiger Bestandteil von CNNs sind Pooling-Schichten. Dies sind Schichten, welche einzelne Merkmale zusammenfassen und damit die Bildgröße reduzieren. Sobald ein Merkmal gefunden wurde, ist der exakte Ort nicht mehr relevant und kann schädlich sein. Es kommt viel mehr auf die ungefähre Position der Merkmale zueinander an (Lecun et al., 1998). Angenommen auf einem Bild ist ein Baum. In diesem Fall ist es wichtiger zu wissen, dass Blätter meist oberhalb des Stammes sind, anstatt anzunehmen, Blätter müssten immer im oberen drittel des Bildes sein. Durch Pooling-Schichten geht räumliche Auflösung verloren, zudem wird der Rechenaufwand geringer und dadurch kann die Anzahl verschiedener Merkmalskarten erhöht werden. Des Weiteren wird das rezeptive Feld der Neuronen vergrößert, welches den lokalen Sichtbereich auf das

Eingabebild beschreibt (LeCun et al., 1989). Deshalb kann eine Pooling-Schicht helfen, Merkmale höherer Ordnung zu finden. Wenn das Merkmal „Blatt“ ist, muss das rezeptive Feld der Neuronen dieser Schicht alle Pixel beinhalten, die ein Blatt umfassen. Die Merkmalskarten können in einen Feature-Vektor konvertiert werden, welcher die Kodierung des Bildinhaltes abbildet.

2.1.1 Grad-CAM

Ein Nachteil Neuronaler Netzen ist, dass sie für Menschen schwer interpretierbar sind. Sie können Informationen extrahieren und in Zahlen ausdrücken, die ohne Übersetzung jedoch keinen Sinn ergeben. Deswegen gibt es verschiedene Ansätze die Interpretierbarkeit von Netzen zu erhöhen, um zu verstehen, was das Netz genau gelernt hat. Ein Beispiel hierfür ist der Activation Atlas[1], der Einsicht in das gesamte Netz geben soll, welches auf Bildklassifizierung trainiert wurde. Es gibt auch Ansätze, die nicht versuchen das gesamte Netz auf einmal zu interpretieren, sondern die Auswirkung, die das Netz auf ein gegebenes Datum hat. Betrachtet wird hier, auf welche Bereiche in genau diesem Datum das Netz geachtet hat. Daraus lassen sich indirekte Rückschlüsse ziehen. Ein Ansatz hierfür ist das Gradient-weighted Class Activation Mapping (Grad-CAM) (Selvaraju et al., 2020).

Grad-CAM ist eine Technik, die lokale visuelle Erklärungen für einzelne separierbare Klassen liefert. Selvaraju et al. verfolgt den Ansatz, die Ausgabe der letzten Konvolutionsschicht zu verwenden. Dies bietet den größten semantischen Kontext, ohne die räumliche Verteilung ganz zu verlieren. Die Neuronen in dieser Schicht betrachten die klassenspezifische Semantik. In dieser Schicht werden die Gradienten in Bezug auf die einzelne Klassenaktivierung berechnet. Diese werden verwendet, um die Aktivierung der Neuronen in der betrachteten Konvolutionsschicht zu gewichten. Zusätzlich werden Aktivierungen mit negativen Werten ignoriert. Dies hilft, die Indizien, die gegen die Klasse sprechen, von denen zu trennen, die für die Klasse sprechen. Aus diesen Aktivierungen wird eine Heatmap erstellt, welche die gleiche Größe besitzt wie das Bild in der betrachteten Konvolutionsschicht. Wenn diese Heatmap über das originale Bild gelegt werden soll, muss sie dementsprechend auf die gewünschte Größe skaliert werden. Die Autoren betrachten diese Lokalisierungskarten in Bezug auf Bildklassifzierungen, sagen aber auch, dass sie im Prinzip für verschiedene CNNs verwendet werden können (Selvaraju et al., 2020).

[1] `https://distill.pub/2019/activation-atlas/`

2.2 Verarbeitung natürlicher Sprache

Natürliche Sprache ist die Sprache, die Menschen tagtäglich zur Kommunikation untereinander verwenden. Das Themenfeld der Computerlinguistik (CL) beschäftigt sich mit der automatischen, maschinellen Verarbeitung menschlicher, also natürlicher Sprache. Dies beinhaltet sowohl die Produktion natürlicher Sprache, wie beispielsweise von Chatbots als auch die Verarbeitung von eingehender Sprache. Dies ist ein herausforderndes Gebiet, da menschliche Sprache von Natur aus mehrdeutig ist und einem stetigen Wandel unterliegt. Natürliche Sprache besteht aus diskreten Symbolen (einzelnen Buchstaben) deren Bedeutung nicht direkt aus diesen abgeleitet werden kann. Sie tragen vielmehr die Bedeutung, die Menschen ihnen gegeben haben (Goldberg, 2017). Abgesehen davon spielt unter anderem auch der Kontext eine Rolle, um die wahre Bedeutung einer Äußerung zu verstehen Der einfache Satz „Schatz, es ist kalt.“ kann sich auf den reinen Informationsgehalt beziehen, nämlich dass es kalt ist, er kann aber auch einen Vorwurf der folgenden Form bedeuten: „Schatz, wenn du letzte Woche nicht vergessen hättest, den Handwerker anzurufen, würde ich hier jetzt nicht frierend sitzen!“ (Goldberg, 2017). In dieser Arbeit wird ein Teilbereich der natürlichen Sprachverarbeitung behandelt, die maschinelle Extraktion von Informationen aus digital erfassten Sätzen.

2.2.1 Text Vorverarbeitung

Der erste Schritt ist die Textnormalisierung. Dies bezeichnet das Verfahren, einen Text in eine standardisierte Form zu überführen. Eine Ausprägung der Normalisierung ist z. B. die Groß- und Kleinschreibung zu entfernen und alles klein zu schreiben, oder auch Zeichen, die die Betonung einzelner Buchstaben beschreiben wie beispielsweise „é“ zu entfernen und durch ihren Buchstaben „e“ zu ersetzten. Des Weiteren wird die Reduktion von Wörtern auf eine Schreibweise angestrebt, die das Gleiche bedeuten, z. B. „Doktor“ und „Dr.“. Ein weiterer Teil der Normalisierung ist Lemmatisierung. Hierbei geht es darum, die gemeinsame Wurzel zweier mehr oder weniger ähnlichen Wörter zu finden und die Wörter durch ihre Wurzel zu ersetzten (Jurafsky and Martin, 2020). Die gemeinsame Wurzel von „bin“ und „ist“ ist, beispielsweise „sein“. Eine solche Text-Normalisierung ist Aufgaben- und Sprachenspezifisch, da eventuell Informationen verloren gehen können.

Nach der Normalisierung muss entschieden werden, welcher Teil des Textes kodiert werden soll. Eine Möglichkeit ist es, den gesamten Text auf einmal zu kodieren oder ihn in Sätze, Wörter oder Zeichen aufzuteilen. Sätze können beispielsweise an Punkten oder Sonderzeichen wie „?“ und „!“ getrennt werden. Dabei ist zu beachten, dass Abkürzungen wie „z. B.“ oder „Dr.“ Punkte enthalten, die nicht das Satzende anzeigen. Bei Wörtern gibt es eine ähnliche Herausforderung. Eine naive Idee könnte

sein, Wörter durch Leerzeichen zu trennen. Dies funktioniert z. B. nicht bei dem Stadtnamen „New York“, da bei einer Trennung in zwei Wörter ein anderer Sinn entsteht. Die Aufteilung des Textes in sinnvolle Einheiten wird Tokenisierung genannt (Jurafsky and Martin, 2020). Weitere Methoden, zerlegten Text in sinnvolle größere Einheiten zusammenzufassen sind ngrams. Sie verbinden aufeinanderfolgende Token, wobei das n für die Anzahl der verbundenen Token steht. Werden beispielsweise zwei Wörter verbunden, wird dies „bigram“ und bei drei Wörtern „trigram“ genannt. Diese Art von sequenzieller Gruppierung kann helfen, Wörter wie „nicht gut“ zu einer Einheit zu gruppieren. Dies wäre über reine Zeichen, Wörter oder Sätze nicht möglich (Goldberg, 2017).

2.2.2 Text Embedding

Sobald die Wörter und Sätze bereinigt sind, müssen die Text-Token in numerische, differenzierbare Vektoren kodiert werden, damit sie algorithmisch verarbeitet werden können. Dazu wird ein bestimmtes Vokabular V, das aus Wörtern, Zeichen oder Sätzen bestehen kann, im Vorhinein festgelegt. Dabei gilt es laut Goldberg (Goldberg, 2017) verschiedene Dinge zu beachten:

One-Hot Kodierung: Jedem Token wird eine eigene Dimension zugeordnet. Bei einer Vokabellänge von $|V|$, wird die Einheit durch einen Vektor der Länge $|V|$ dargestellt. Diese Vektoren sind dünn-besetzt, da sie nur in der entsprechenden Einheitsdimension eine 1 beinhalten und sonst nur aus Nullen bestehen. Die Token sind durch diese Darstellung unabhängig voneinander.

Dichte Kodierung: Jedes Token wird in einem festgelegten d-dimensionalen Raum eingebettet, wobei d kleiner ist als die Größe des Vokabulars. Die Token haben nicht ihre eigene Dimension, sondern werden mithilfe einer Funktion f auf einen d-dimensionalen Vektor abgebildet. Diese Funktion f kann Teil einer lernbaren Funktion sein, deren Parameter mittrainiert werden. Dabei kann das Training dazu führen, dass ähnliche Wörter auch ähnliche Vektoren haben. Dichte Vektoren haben den Vorteil, dass sie kleiner sind und weniger irrelevante Informationen wie Nullen in einem dünn-besetzten Vektor beinhalten.

Kombination der Einheiten: Um einen gesamten Text durch kodierte Vektor Token darzustellen, müssen diese kombiniert werden. Die normalen Kombinationsarten sind hierbei abhängig davon, ob die Reihenfolge der Token bewahrt bleiben soll oder nicht. Für die Erhaltung der Reihenfolge können die Vektoren konkateniert werden, dabei wird die Dimension immer größer. Wenn die Reihenfolge keine Rolle spielt, können die Vektoren aufaddiert oder gemittelt werden.

Auffüllen und Abschneiden: Für viele Algorithmen ist es wichtig, dass die Kombination der Token immer gleich groß ist. Bei unabhängiger Reihenfolge ist das

nicht problematisch, da die Dimension der Vektoren sich nicht verändert, wenn sie addiert werden oder der Durchschnitt gebildet wird. Wenn die Reihenfolge wichtig ist, ergibt sich aus den einzelnen Vektoren eine fortlaufende Sequenz von Vektoren. Diese Sequenz benötigt immer dieselbe Länge l. Um dies zu erreichen, können kürzere Sequenzen mit neutralen Vektoren aufgefüllt werden oder zu lange Sequenzen abgeschnitten werden. Sowohl das Auffüllen als auch das Abschneiden kann entweder am Anfang oder am Ende der Sequenz stattfinden.

Unbekannte Einheiten und Dropout: Wenn die Tokens aufgrund eines Vokabulars kodiert werden, kann es vorkommen, dass unbekannte Token in der Eingabe vorkommen. Dies können Token sein, die sich nicht in der Vokabelliste befinden. Es gibt zwei Möglichkeiten mit einem solchen Token umzugehen. Entweder kann eine spezielle Repräsentation eingeführt werden oder es kann ignoriert (Dropout) werden. Der Nachteil an einem Dropout ist, dass dadurch Informationen verloren gehen können. Eine spezielle Repräsentation deutet weiterhin darauf hin, dass dort Etwas ist, was jedoch nur nicht kodiert werden konnte. Wenn es sich allerdings um Algorithmen aus dem Bereich des maschinellen Lernens handelt, sollte dafür gesorgt werden, dass diese spezielle Repräsentation auch häufig genug vorkommt, so dass der Algorithmus lernen kann damit umzugehen. Bei lernenden Algorithmen kann das Einfügen der speziellen Repräsentation oder der Dropout helfen, den Algorithmus robuster zu machen.

2.2.3 1-Dimensionale Convolutional Neural Networks

Nach der Normalisierung und der Kodierung in eine Vektorrepräsentation, kann der Text algorithmisch verarbeitet werden. Eine moderne Methode um natürliche Sprache zu verarbeiten, sind Algorithmen, die auf überwachtem maschinellen Lernen basieren. Zu diesen zählen auch Neuronale Netze (Goldberg, 2017). Es gibt einige Ansätze, die Neuronale Netze hierfür verwenden. Eine ausführliche Übersicht ist in (Minaee et al., 2021) zu finden. Die Wahl der Art der Netz-Architektur ist unter anderem von der Wahl der Token abhängig und ob die Reihenfolge wichtig ist. Text besteht aus einzelnen Zeichen, die für den Leser aber erst im Zusammenhang, also in Form von Wörtern, einen Sinn ergeben. Deswegen gibt es viele Ansätze, ganze Wörter zu kodieren wie z. B. in (Karpathy and Fei-Fei, 2015) und (Lu et al., 2021). Durch den Erfolg mit Convolutional Neural Networks (CNN) rohe Signale zu verarbeiten, beispielsweise Bilder auf Pixelbasis, gibt es den Ansatz Texte, in ihrem Roh-Format auf Zeichenbasis, zu verarbeiten (Zhang et al., 2016; Xiao and Cho, 2016). Ein solches CNN kann damit auch als ngram Detektor für Zeichen gesehen werden, für die ngrams die am informativsten für diese Aufgabe sind, ohne dass im vorhinein alle möglichen ngrams spezifiziert werden müssen (Goldberg, 2017).

2.2.4 Rekurrente Neuronale Netze

CNNs beachten die Reihenfolge innerhalb einer Textsequenz. Dennoch werden hauptsächlich lokale Pattern angeschaut. Des Weiteren wird „Zeit“ hier als eine weitere räumliche Dimension betrachtet, also explizit eingebracht und zwar in Form der Sequenz. Aus den Vektoren, die aus den einzelnen Token bestehen, wird eine Sequenz gebildet, die in einer weiteren Dimension der Zeit entspricht. Die gesamte Sequenz wird dann parallel verarbeitet. Laut Elman widerspreche das dem natürlichen Verständnis von Zeit (Elman, 1990). Ein Nachteil ist auch, dass CNNs eine feste Eingabesequenzlänge benötigen. Verbunden damit ist, dass die Länge der Sequenzen begrenzt ist. Elman beschreibt, dass Zeit implizit dargestellt werden kann, indem dem Netz ein Gedächtnis in Form von zusätzlichen Gedächtniszellen gegeben wird. Die Eingabe ist daher immer genau ein Vektor eines Tokens. Die Speicherfunktionalität bildet sich daraus, dass die Neuronen in den tieferen Schichten, also nicht in der Eingabe und Ausgabeschicht, ihre eigene Ausgabe wieder als zusätzliche Eingabe bekommen. Solche Netze werden Rekurrente Neuronale Netze (RNNs) genannt (Elman, 1990).

Ein Nachteil ist jedoch, dass sich ein solches RNN zu Trainingszeiten so verhält, wie ein sehr tiefes Neuronales Netz. Zum Training werden die einzelnen rekurrenten Neuronen entrollt, was ein Problem eines verschwindenden Gradienten auslösen kann. Um dies zu umgehen, wurde von Hochreiter und Schmidhuber die Long Short-Term Memory (LSTM) Architektur eingeführt (Hochreiter and Schmidhuber, 1997). Diese führt Kontrollmechanismen ein, mit denen über die Zeit hinweg Gespeichertes auch wieder vergessen werden kann. Dafür werden die Gedächtniszellen in ein speichernden Teil und ein Arbeitsgedächtnis aufgeteilt. Zu jeder Eingabe wird über Schranken, mathematische Funktionen bestimmt, wie viel von der neuen Eingabe gespeichert und wie viel vergessen wird. Diese Schranken erlauben es den Gradienten, die zu den Speicherzellen gehören, auch über längere Zeit hinweg groß zu bleiben (Hochreiter and Schmidhuber, 1997).

2.2.5 Kombination verschiedener Architekturen

Es gibt viele weitere unterschiedliche Netzarchitekturen oder Weiterentwicklungen von den eben genannten. Diese spielen jedoch für diese Arbeit keine Rolle und werden daher nicht weiter betrachtet. Zusätzlich besteht die Möglichkeit, die verschiedenen Architekturen zu kombinieren und die Eigenschaften der verschiedenen Ausgaben für unterschiedliche Zwecke zu nutzen.

Zhang et al. präsentieren eine Netzarchitektur, die auf 1-Diemsionalen Konvolutionen und vollverknüpften Schichten besteht. Mit dieser Netzarchitektur werden zeichenbasierte Eingaben verarbeitet und Text-Klassifizierungsprobleme gelöst (Zhang et al.,

2016). Die Hauptkomponenten sind dabei temporale (eindimensionale) Konvolutionen. Sie weisen der Zeit also eine explizite Dimension zu. Diese Zeichen-Sequenzen werden durch sechs Konvolutionsschichten und drei vollverknüpfte Schichten verarbeitet. Insbesondere dann, wenn die Größe des Trainingdatensatzes groß ist, funktioniere diese Architektur sehr gut. Xiao und Cho zeigen, dass durch eine Mischung aus Konvolutionsschichten und einer Rekurrenten-Schicht, die Tiefe der Netze verringert werden kann, um vergleichbare Ergebnisse zu erzielen, da eine Recurrente-Schicht besser die Langzeit-Abhängigkeiten verstehen kann. Diese Architektur funktioniere besser, wenn die Anzahl an Klassen größer ist (Xiao and Cho, 2016).

2.3 Training Neuronaler Netze

Das Ziel des Trainings Neuronaler Netze ist es, eine Funktion f zu finden, deren Vorhersage für eine Eingabe genau der Grundwahrheit entspricht, die dieser Eingabe zugeordnet ist. Um die Vorhersage quantifizieren zu können, wird eine Verlustfunktion $L(\hat{y},y) \mapsto \mathbb{R}$ benötigt, die dem Verhältnis der Vorhersage $\hat{y}$ und dem wahren Ergebnis y eine skalare Zahl zuweist. Die Verlustfunktion sollte genau dort ihr Minimum besitzen, wo $\hat{y} = y$ gilt (Goldberg, 2017). Als Verlustfunktion kann im Prinzip eine beliebige Funktion eingesetzt werden, die den eben genannten Kriterien entspricht. Die Wahl der Verlustfunktion ist sehr problemspezifisch. Für diese Arbeit werden nur die beiden Folgenden betrachtet.

2.3.1 Multi-Binäre Kreuzentropie

Die binäre Kreuzentropie Verlustfunktion wird bei binären Klassifizierungsproblemen verwendet, bei denen die Vorhersage als Wahrscheinlichkeitsverteilung interpretiert wird. Hierbei gibt es zwei Klassen, Eine mit dem Wert 0 und die Andere mit dem Wert 1. Der wahre Wert ist deshalb immer $y \in \{0,1\}$. Damit das Ergebnis der Berechnung eine valide Wahrscheinlichkeit darstellt, muss $\hat{y} \in [0,1]$ gelten (Goldberg, 2017). Dies kann unter anderem durch die Sigmoid-Funktion $\sigma(x) = \frac{1}{1+e^{-x}}$ erzwungen werden.

$$L_{\text{bke}}(\hat{y},y) = -y \log(\hat{y}) - (1-y) \log(1-\hat{y}) \tag{2.1}$$

Wenn mehr als zwei Klassen existieren, kann für jede Klasse der binäre Kreuzentropieverlust berechnet und diese dann aufsummiert werden. Dabei wird jede Klasse als zwei Klassen betrachtet. Bei der Klasse A wäre es dann z. B. $A \vee \neg A$ Dadurch wird die Verlustfunktion für L Klassen wie folgt erweitert (Nam et al., 2014):

$$L_{\text{mlbke}}(\hat{y},y) = \sum_{i \in L} (-y_i \log(\hat{y_i}) - (1-y_i) \log(1-\hat{y_i})) \tag{2.2}$$

2.3.2 Contrastive Loss

Eine Verlustfunktion, eine Variante des Contrastive Loss (Kontrastiver Verlust) wurde von Hadsell et al. im Bereich der Dimensionsreduktion vorgestellt (Hadsell et al., 2006). Dabei sollten Daten, die im hoch-dimensionalen Raum ähnlich zueinander sind, auch in einem projizierten nieder-dimensionalen Raum ähnlich zueinander sein. Eine solche Verlustfunktion sorgt dafür, dass ähnliche Punkte zusammengezogen werden, unähnliche sich abstoßen. Denn wenn nur die positiven Paare betrachtet werden würden, würde die Funktion kollabieren (Hadsell et al., 2006). Seitdem wurde diese Art von Verlustfunktion häufig aufgegriffen und für verschiedene Zwecke verwendet. Hier wird das Contrastive Loss von Chen et al. verwendet (Chen et al., 2020). Chen et al. nutzen die Verlustfunktion um visuelle Repräsentationen zu lernen, ohne ein Datensatz der Grundwahrheiten enthält, also für unüberwachtes Lernen (engl. unsupervised learning). In ihrem Ansatz besteht ein Paar immer aus einem Originalbild und einer augmentierten Variante von diesem. Sie erweitern also jeden Batch um einen augmentierten Batch, sodass sie auf eine Batchgröße von $2N$ kommen. Die Verlustfunktion für ein positives Paar (i, j) ist dann definiert durch

$$l_{i,j} = -\log \frac{\mathrm{e}^{\mathrm{sim}(z_i,z_j)/\tau}}{\sum_{k=1,k\neq j}^{2N} \mathrm{e}^{\mathrm{sim}(z_i,z_k)/\tau}}, \tag{2.3}$$

wobei τ einen Temperatur-Parameter darstellt und sim für die Kosinus-Ähnlichkeit steht.

2.3.3 Kosinus-Ähnlichkeit

Die Kosinus-Ähnlichkeit ist ein Maß mit dem die Ähnlichkeit von zwei n-dimensionalen Vektoren $u = (u_1, \ldots, u_n), v = (v_1, \ldots, v_n)$ beschrieben werden kann (Klahold, 2009).

$$\mathrm{Sim}_{\cos}(u,v) = \frac{u \circ v}{\|u\|_2 \cdot \|v\|_2} \tag{2.4}$$

Dabei beschreibt $u \circ v$ das Skalarprodukt zwischen u und v und $\|u\|_2$ die 2er-Norm. Je größer der Wert, umso ähnlicher sind sich zwei Vektoren. Der Vorteil der Kosinus-Ähnlichkeit ist, im Vergleich zum reinen Skalarprodukt, dass über die Länge der Vektoren normalisiert wird und damit das resultierende Ergebnis zwingend in $[-1,1]$ liegt, wobei 1 eine Übereinstimmung der Richtung bedeutet, und -1 die entgegengesetzte (Sartorius, 2019).

2.3.4 Gradientenbasierte Optimierung

Um das Model mithilfe der Verlustfunktionen zu trainieren, muss das folgende Optimierungsprlobem gelöst werden:

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i; \Theta)) \tag{2.5}$$

Bei Neuronalen Netzen ist es verbreitet, dies mithilfe gradientenbasierter Optimierung zu lösen. Das heißt für jedes Eingabedatum x_i mit den aktuellen Parametern den Verlust berechnen und in Abhängigkeit dieses Verlustes die Gradienten der Parameter zu berechnen und diese entgegengesetzt ihres Gradienten anzupassen. Die Art und Weise, wie die Parameter entgegen ihres Gradienten angepasst werden, wird durch die gewählte Optimierungsmethode bestimmt (Goldberg, 2017). In dieser Arbeit soll als Optimierungsmethode der Adam-Algorithmus verwendet werden (Kingma and Ba, 2017), da dieser als sehr robust und effektiv für beliebige Funktionen im hoch-dimensionalen Bereich des maschinellen Lernens gilt (Goldberg, 2017).

In tiefen Neuronalen Netzen können die Gradienten, die aufgrund des Verlustes berechnet wurden, entweder ziemlich stark gegen 0 gehen oder sehr groß werden, wenn sie zurück durch das Netz propagiert werden. Dies macht eine gute Optimierung fast unmöglich. Es gibt viele Ansätze, dies zu kontrollieren oder zu umgehen, allerdings eine allgemeingültige Lösung gibt es noch nicht. Bei den LSTM-Netzen wird dies versucht zu unterbinden, indem die Vergessen-Schranke eingebaut wurde. Ein weiterer Ansatz ist, die Werte innerhalb einer Schicht zu normalisieren, sodass diese nicht zu groß oder klein werden. Batch-Normalisierung und Layer-Normalisierung sind zwei Beispiele hierfür (Ioffe and Szegedy, 2015; Ba et al., 2016).

Bei der Batch-Normalisierung werden die Ausgaben einer Schicht über einen Batch hinweg normalisiert, sodass die nächste Schicht eine normalisierte Eingabe erhält. Es wird also für jedes Neuron über den Batch hinweg der Durchschnitt und die Varianz gebildet und darüber normalisiert, sodass der Durchschnitt 0 und die Standardabweichung 1 beträgt. Zusätzlich werden diese Werte jedoch über lernbare Parameter skaliert und verschoben, um die Aussagekraft der Neuronen nicht zu verlieren. Über einen Batch zu normalisieren ist zwar zur Trainingszeit sehr effektiv; bei der Auswertung jedoch unerwünscht, da die Ausgabe des Netzes allein auf der Eingabe basieren soll und nicht abhängig von einem Batch. Deshalb werden zur Auswertungszeit der gelernte Durchschnitt und die Standardabweichung aller Trainingsdaten verwendet.

Layer-Normalisierung hingegen normalisiert zwar auf einen Durchschnitt von 0 und eine Standardabweichung von 1, allerdings nicht über einen Trainingsbatch, sondern über einzelne Dimensionen innerhalb eines Trainingsdatums. Dies ist insbesondere dann ein Vorteil, wenn die Trainingsdaten Sequenzen sind, die unterschiedlich lang

sein können. Diese werden zwar häufig aufgefüllt, wenn sie zu kurz sind, dennoch würde eine Normalisierung in Batch-Dimension durch die künstlich aufgefüllten Einträge verfälscht werden. Die Layer-Normalisierung lässt sich daher flexibler an die Originallänge der Eingabesequenz anpassen. Ein weiterer Vorteil ist, dass bei dieser Normalisierungsmethode nicht zwischen Training- und Auswertungszeit unterschieden werden muss, da die Normalisierung nur innerhalb einer Eingabe stattfindet (Ba et al., 2016).

3 Verwandte Arbeiten im Bereich der Bildabruf-Systeme

In diesem Kapitel wird auf die Frage eingegangen: Was bedeutet Unterstützung in diesem Kontext? Laut Duden bedeutet unterstützen „sich für jemanden, jemandes Angelegenheiten o. Ä. einsetzen und dazu beitragen, dass jemand, etwas Fortschritte macht, Erfolg hat“ (Duden, 2021b). Wenn für das in dieser Arbeit behandelte Problem das Auffinden eines gesuchten, passenden Bildes als Erfolg interpretiert wird, dann kann Unterstützung bedeuten ein System zu entwerfen, das dazu beiträgt, ein passendes Bild zu finden. Die Suche nach oder auch das Abrufen von Bildern, im englischen Image Retrieval, ist ein Problem zu dem umfassend geforscht wird. Deswegen wird im folgenden ein Überblick über vorhandene Ansätze und Systeme gegeben.

Erste textbasierte Verfahren gibt es schon, seit dem Bilder in Datenbankmanagementsystem verwaltet werden. Die Bilder werden häufig händisch annotiert oder durch Metadaten wie beispielsweise Größe, Auflösung oder Herkunft beschrieben. Mit verschiedenen Anfragesprachen, abhängig vom System, können diese Datenbanken durchsucht werden (Chang and Hsu, 1992). Durch die stetige massive Zunahme von Bilddaten wie beispielsweise Satellitenbilder, die tagtäglich entstehen, ist eine händische Annotation der Bilder in diesen Bereichen häufig nicht mehr möglich. Abgesehen davon weist eine reine, auf händischer Annotation und Metadaten basierende Speicherung weitere Nachteile auf. Unter anderem ist die Inhaltsbeschreibung des Bildes sehr subjektiv. Sogar für ein und dieselbe Person kann die Bedeutung und Interpretation des Inhaltes eines Bildes sich mit der Zeit ändern. Des Weiteren können ein paar wenige Worte nicht die ganze Komplexität des Bildes beschreiben und zusätzlich sind Wörter häufig mehrdeutig. Dies führt unter anderem zu der Entwicklung der inhaltsbasierten Speicherung. Die Idee dabei ist, dass visuelle Inhalte wie beispielsweise Textur und Farbe einer bestimmten Region ein Bild definieren (T.karthikeyan and aprabhu, 2014).

Da der hier betrachtete Fall nicht nur von der Zunahme an Bildmaterial abhängt, sondern auf schon vorhandenen Bilderdatenbanken aufbaut, scheint eine nachträgliche händische Annotation sehr aufwendig. Darüber hinaus bedeutet zum Erfolg beizutragen auch, dass es nicht darum geht, die Aufgabe des Bildredakteurs voll-

ständig zu übernehmen. Deshalb ist ein System, dass genau ein Bild zu einem Text heraussucht, für diese Anwendung nicht geeignet. Dann würde dem Nutzer die Wahl entzogen und der Teil seiner Arbeit erledigt sein. Das System kann daher nicht ein reines Suchsystem sein, sondern eine Mischung aus Such- und Empfehlungssystem, welches dem Nutzer eine Auswahl an Bildern aus der gesamten Datenmenge empfiehlt. Aufgrund dieser Empfehlung kann der Nutzer in letzter Instanz die Entscheidung treffen (Klahold, 2009). Für ein solches Empfehlungssystem fällt eine reine Stichwortsuche weg, da Stichwörter nicht ohne zusätzliche Information in eine Rangfolge gebracht werden können. Eine weitere Möglichkeit auf inhaltsbasierten Datenbanken ein bestimmtes Bild zu suchen, ist eine Abfrage mithilfe eines ähnlichen Bildes zu tätigen (T.karthikeyan and aprabhu, 2014). Darüber hinaus gibt es auch hybride Versionen, in denen Bild und Text gemeinsam als Abfrage verwendet werden wie beispielsweise in (Vo et al., 2019). Auch wenn solche Ansätze sehr vielversprechend sind, besteht das Problem hier darin, ein Bild zu einem bestehenden Text zu finden, das heißt, es existiert noch kein Bild, mit dem gesucht werden könnte. Deswegen beziehen sich die weiteren Ansätze auf reine Textabfragen.

Bei der inhaltsbasierten Beschreibung der Bilder, die mit Textabfragen gesucht werden, kann zusätzlich noch zwischen generativen und diskriminierenden Modellen unterschieden werden. Generative Modelle sind beispielsweise automatische Untertitelungsmodelle, die zu einem Bild einen Text generieren, der dann wiederum mit dem Eingabetext verglichen werden kann. Ein Beispiel hierfür ist ein Modell von Karpathy und Fei-Fei, welches mithilfe eines multimodalen RNN die Regionen eines Eingabebildes mit deskriptiven Teilsätzen oder Objektbeschreibungen charakterisiert (Karpathy and Fei-Fei, 2015). Ein Vorteil einer solchen textuellen Annotation ist, dass sie von Menschen gut verstanden und auf Plausibilität geprüft werden kann. Der Nachteil eines solchen Ansatzes ist jedoch, dass durch eine solche Annotation ein Zwischenschritt einfügt wird, indem die Bilder zuerst textuell annotiert und dann erst gesucht werden. Der Ansatz bildet das eigentliche Problem nicht unmittelbar ab. Diskriminierende Modelle werden direkt auf die spätere Problemstellung trainiert und optimiert (Grangier and Bengio, 2006).

Grangier und Bengio verfolgen den Ansatz mithilfe eines neuronalen Netzes, das von CNNs inspiriert ist, aus lokalen Features einen Text-Vektor zu erstellen. Dieser berechnete Vektor wird dann mit dem originalen Eingabe Text-Vektor verglichen und die daraus resultierende Abweichung wird zum Trainieren verwendet. Darüber hinaus wird aus den Vektoren das Punktprodukt berechnet, aus dem eine Rangordnung erstellt wird. Der Ansatz vereint damit die Repräsentation des Bildes auf der Textebene sowie das Abrufen des Bildes zum gegebenen Satz. Der Zwischenschritt, das Bild zu annotieren und dann durch Stichwortsuche zu suchen, entfällt (Grangier and Bengio, 2006). Allerdings wird das Bild dadurch auf die Text-Ebene übersetzt, nach der Extraktion der Information wird diese Repräsentation also noch einmal

verändert. Dies kann eventuell eine weitere Fehlerquelle darstellen, da es über die reine Informationsextraktion hinausgeht.

Ein bidirektionales Text- und Bildabrufverfahren präsentieren Ma et al. (Ma et al., 2015). Sie konstruieren ein multimodales CNN, welches einen Satz und ein Bild als Eingabe zusammenfügt und eine Bewertung ausgibt, wie das Eingabe-Paar zueinander passt. Sowohl das Bild als auch der Text werden durch ein passendes CNN vorverarbeitet und in gleich große Feature-Vektoren eingebettet. Diese werden dann als multimodale Eingabe in ein weiteres CNN gegeben, welches letztendlich die Bewertung generiert. Das Verfahren kann somit also verwendet werden, um zu berechnen, wie gut ein Bild zu einem Text passt und anders herum, wie gut der Text zu einem Bild passt. Dabei müsste allerdings, wenn genau ein Text mit n-Bildern verglichen wird, diese Berechnung genau n-mal gemacht werden und das Ergebnis gespeichert werden, damit am Ende eine Auswahl in einer Rangordnung empfohlen werden kann. Bei einer großen Datenbank ist dies rechenaufwendig.

Deswegen besteht die Herausforderung nicht nur darin, die Genauigkeit des Ergebnisses zu maximieren, sondern auch die Latenz, mit der das Ergebnis berechnet wird, zu minimieren. Ein Informationsabfragesystem muss den Echtzeitanforderungen der Nutzer gerecht werden (Christopher D Manning and Raghavan, 2008). Auf dies gehen Lu et al. (Lu et al., 2021) mit einem auf einem Fragmentlevel basierenden CNN-Modell ein. Die Grundlage der Bewertung wird durch das Zerlegen des Bildes in Fragmente und der Betrachtung des Textes in unabhängigen Tokens berechnet. Die Latenz halten Lu et al. niedrig, indem sie nach dem Training für jedes Bild mit jedem Wort des trainierten Vokabulars einen Score berechnen und zwischenspeichern. Für die Eingabe wird der Text dann nur noch in Tokens zerlegt, die einzelnen Werte werden im Zwischenspeicher abgefragt und aufsummiert (Lu et al., 2021). Der Vorteil einer solchen Variante ist, dass die zwischengespeicherten Tokens für jedes Bild aus Wörtern bestehen, die für Menschen gut nachvollzogen werden können. Dadurch kann die Transparenz erhöht werden, welche Tokens den Bildern zugeordnet werden und ob es einen Bias gibt. Ein Nachteil bei diesem Ansatz ist allerdings, dass für jedes Bild ein gewisser Teil des Vokabulars zwischengespeichert werden muss, was speicheraufwendig sein kann.

Die beschriebenen Ansätze zeigen deutlich, dass es allgemeine Anforderungen an Bildabruf-Systeme gibt, aber auch, dass die Systeme problemspezifisch sind. Ein Nachteil von allen präsentierten Modellen ist, dass sie davon ausgehen, dass es prägnante, beschreibende Sätze sind, die die Bilder beschreiben oder mit denen die Abfrage gemacht wird. Dies würde jedoch in dem in der Arbeit betrachteten Problem von dem Nutzer fordern, dass er seinen schon geschriebenen Text prägnant zusammenfasst. Dennoch sind der Rechenaufwand und die damit verbundene Zeit sowie eine direkte Problemabbildung wichtige Aspekte. Auf die genauen Anforderun-

gen und die Art, wie die aufgezählten Nachteile umgangen werden können, wird im nächsten Kapitel eingegangen.

4 Konzeption

Das System, das in dieser Arbeit entwickelt wird, ist für einen realen Anwendungsfall ausgelegt. Es geht darum, dass nachdem ein Text verfasst worden ist, für den kein spezielles Foto aufgenommen wurde oder existiert, dieser dennoch mit einem zum Inhalt passenden Bild unterlegt werden kann. Dafür werden aktuell mit einer Stichwortsuche große Bilderdatenbanken von externen Dienstleistern durchsucht oder eine Anfrage bei einer entsprechenden Bilderagentur gestellt. Diese beiden Möglichkeiten sind kein weiterer Bestandteil dieser Arbeit. Hier wird der Fall betrachtet, dass bereits ein Archiv von Bildern in einer Datenbank vorliegt, diese aber nicht maschinell nach bestimmten Kriterien durchsuchbar ist. Das System muss also rohe Bilddaten, ohne Metadaten, Stichwörtern oder Vergleichbarem, sowie natürliche Texte verarbeiten und auf inhaltliche Ähnlichkeit untersuchen können.

In diesem Kapitel werden zunächst die Anforderungen, die an ein Such- und Empfehlungssystem für den beschriebenen Anwendungsfall gestellt werden, spezifiziert. Danach wird die Frage beantwortet, wie ein solches System konkret aussehen kann. Am Ende wird das vorgestellte System mit den Anforderungen verglichen.

4.1 Anforderungen

Im vorherigen Kapitel wurden vorhandene Modelle vorgestellt, die auf das Abrufen von Bildern spezialisiert sind. Sie sind für den oben genannten Anwendungsfall nicht geeignet, da sie davon ausgehen, es gäbe immer einen deskriptiven englischen Satz, der die Bilder beschreibt. Allerdings zeigen sie einige Herausforderungen, die beachtet werden müssen. Unter anderem der erforderliche Speicherplatz oder der Zeitaufwand der Berechnungen (Lu et al., 2021; Grangier and Bengio, 2006). Daraus ergeben sich die folgenden Anforderungen an das System:

Domänenübergreifend Das System muss sowohl rohe Bilddaten als auch Textdaten, die in natürlicher Sprache verfasst sind verarbeiten können.

Problemadressierung Das System muss das Problem möglichst genau abbilden können, also ohne Umwege über Zwischenlösungen, die weitere Fehlerquellen darstellen können.

Skalierbarkeit Das System muss in zwei Richtungen skalierbar sein: In Richtung des Zeitaufwandes bei dem tatsächlichen Abruf. Darüber hinaus muss der Aufwand der Erstellung der Trainingsdaten betrachtet werden.

Kompatibilität Die Verarbeitung der Eingabedaten muss auf die späteren Nutzdaten abgestimmt sein, das bedeutet, die Verarbeitung der Texte muss in der Zielsprache Deutsch möglich sein und farbige Bilder müssen verarbeitet werden können.

Rangfolge Das verwendete Ähnlichkeitsmaß muss in eine Rangfolge gebracht werden können, damit als Empfehlung die ähnlichsten Bilder zum Text präsentiert werden können.

4.2 Digitale Texte und Bilder

Bilder und Texte werden in einer digitalen Repräsentation durch binäre Zahlenwerte dargestellt, die eine verschiedenartige Bedeutung besitzen, oder vielmehr unterschiedlich interpretiert werden. Eine mögliche digitale Repräsentation von Bildern ist beispielsweise die pixelweise Speicherung von RGB-Werten (Rot-Grün-Blau). Texte sind zwar zeichenweise kodiert, z. B. durch die UTF-8 Kodierung, dennoch steht diese Kodierung nicht im Zusammenhang mit RGB-Werten. In der weiteren Argumentation werden diese beiden Kodierungsmethoden weiter exemplarisch verwendet. Wenn Informationen aus Texten und Bildern digital und automatisiert verglichen werden, müssen sie zunächst in einer vergleichbaren digitalen Repräsentation vorliegen. Dadurch ergeben sich drei Fragestellungen auf die in den nächsten Abschnitten eingegangen wird: Erstens, in welchem Zahlenraum soll die Repräsentation stattfinden, zweitens wie ist Ähnlichkeit definiert und drittens mit welcher Funktion sollen Text und Bild in diesen Raum projiziert werden?

4.2.1 Domänenübergreifender Vektorraum

Eine Möglichkeit, einen solchen gemeinsamen Raum zu schaffen, ist, Texte in den Bildraum zu projizieren, von UTF-8 in RGB-Pixel Darstellung zu überführen wie z. B. in der Arbeit von Ramesh et al. in der aus einem deskriptiven Satz ein Bild generiert wird (Ramesh et al., 2021). Die andere Richtung, also aus einem Bild einen deskriptiven Satz zu generieren, also von RGB-Pixel in UTF-8 zu projizieren, ist auch eine Möglichkeit (Karpathy and Fei-Fei, 2015). Der Vorteil solcher Methoden ist, dass sie für Menschen gut zu interpretieren sind. Wenn das Bild eines spielenden Hundes beispielsweise einen Satz generiert, der beschreibt, wie eine Katze schläft, dann kann das Hinweise auf eventuelle Fehlfunktionen der Projektion geben.

Allerdings würde dies bedeuten, dass die notwendigen Informationen aus dem Text oder Bild extrahiert werden müssen und dann zusätzlich wieder übersetzt werden müssen. Dies bietet eine doppelte Fehlerquelle, da Fehler sowohl bei der Extraktion der Information geschehen können, als auch bei der Übersetzung in die jeweilige andere Domäne. Deswegen ist der Ansatz dieser Arbeit, Bild und Text in einen gemeinsamen, neutralen numerischen Raum $\mathbb{I} \subset \mathbb{R}^n$ zu projizieren und in diesem Text und Bild auf Ähnlichkeit zu überprüfen. Ein weiterer Vorteil eines explizit neutralen gemeinsamen Raums ist, dass er nicht definiert ist, um für Menschen verständlich zu sein. Das heißt, er ist nicht durch die Definition und Interpretation der Menschen begrenzt, wie er es beispielsweise durch die eingeschränkte Darstellungsweise von RGB-Werten wäre. Die projizierte Information von Bild oder Text wird als n-dimensionaler Vektor dargestellt: $\mathbf{v}_{\text{text}}, \mathbf{v}_{\text{bild}} \in \mathbb{I}$. Diese Tatsache ist in Bezug auf die Interpretierbarkeit allerdings ein Nachteil, weil ein Vektor $\mathbf{v} \in \mathbb{I}$ keine intuitive Bedeutung hat.

4.2.2 Ähnlichkeit von Text und Bild

Die Bestimmung von Ähnlichkeit in einem hoch-Dimensionen euklidischen Vektorraum, in den die extrahierten Informationen projiziert werden, kann aus zwei verschiedenen Sichtweisen betrachtet werden. Zum einen kann ein Distanzmaß angelegt werden, das bestimmt, wie groß die Distanz ist und dadurch gilt: Je kleiner die Distanz ist, umso ähnlicher sind sich zwei Vektoren. Ein Beispiel hierfür ist die Manhattan-Distanz. Allerdings nimmt der Abstand zweier Vektoren stark zu, wenn die Anzahl an Dimensionen zunimmt. Dies kann umgangen werden, indem anstelle des tatsächlichen Abstandes der Winkel betrachtet wird. Eine mögliche Metrik hierfür ist die Kosinus-Distanz, die sich aus der Kosinus-Ähnlichkeit berechnet: $1 -$ Kosinus-Ähnlichkeit (Sartorius, 2019). Zum anderen kann die Ähnlichkeit direkt zu verwenden, wobei gilt: Je größer der Wert ist, umso ähnlicher sind sich zwei Vektoren. Die Kosinus-Ähnlichkeit wird unter anderem auch eingesetzt, um Dokumente zu Clustern oder deren Ähnlichkeit zu berechnen (Singhal, 2001; Gunawan et al., 2018; Tan et al., 2006). Die Kosinus-Ähnlichkeit eignet sich demnach mathematisch als Ähnlichkeitsmaß für den hier verwendeten Informationsraum und die bereits erfolgreiche Verwendung bei Dokumenten legt nahe, dass sie sich auch für die hier genutzte Art von Daten eignet.

4.3 Domänenspezifische Projektionsfunktion

Als letzter Schritt wird eine Funktion benötigt, die Bild und Text in den Informationsraum projiziert. Neuronale Netze sind zwar durch das Lernverhalten in Gehirnen inspiriert worden und werden konzeptionell in Schichten und Neuronen dargestellt,

letztendlich sind Neuronale Netze jedoch komplexe, mathematische Funktionen (Goldberg, 2017). Neuronale Netze sind mit steigender Rechenleistung immer beliebter geworden und erreichen speziell in der Bildklassifikation sehr gute Ergebnisse (with Code, 2021). Auch in der Textklassifikation werden Neuronale Netze mittlerweile erfolgreich eingesetzt (Minaee et al., 2021). Um ein Text oder ein Bild einer Klasse zuordnen zu können, muss das Netz irgendeine Art der Information aus Text oder Bild extrahieren können. Diese Eigenschaft wird hier benötigt. Deswegen werden für diesen Ansatz Neuronale Netze als Projektionsfunktion verwendet.

Aufgrund der unterschiedlichen Herkunftsdomäne der zwei Eingaben sind zwei Möglichkeiten Neuronale Netze zu verwenden offensichtlich: Ein Netz mit multimodaler Eingabe wie in (Ma et al., 2015), indem sowohl das Bild als auch der Text durch dasselbe Netz verarbeitet werden. Der Vorteil an diesem Ansatz ist, dass ein Netz nach der klassischen Art und Weise trainiert werden kann. Es wird eine Ausgabe berechnet und damit der Fehler, der dann minimiert werden kann. Der Nachteil daran ist allerdings, dass zu jeder Zeit, also auch zur Auswertung, Text und Bild benötigt werden, da sie nur zusammen ausgewertet werden können. Das ist insbesondere nachteilig, wenn ein Text mit einer großen Bilderdatenbank verglichen wird. Dadurch müsste der Text mit jedem Bild durch das Netz ausgewertet werden. Dies kann abhängig vom Netz sehr aufwendig sein. Die zweite Möglichkeit sind zwei separate Netze. Eines, das Bilder verarbeitet und Eines, das Texte verarbeitet. Zur Trainingszeit können diese als ein Netz interpretiert werden und gemeinsam trainiert werden. Zur Auswertung können sie separat verwendet werden. Der Ansatz von zwei verschiedenen domänenspezifischen Netzen wird in dieser Arbeit verfolgt. Im Folgenden wird zunächst auf die jeweiligen Netze eingegangen und danach auf das gemeinsame Training.

4.3.1 Neuronales Netz zur Bildverarbeitung

Wie in Abschnitt 2.1 beschrieben, hat die Verarbeitung von Bildern mit Neuronalen Netzen, im speziellen CNNs, große Erfolge erzielt. Daher wird auch in dieser Arbeit ein Neuronales Netz verwendet, das auf dem Konzept von Konvolutionen aufbaut. Darüber hinaus ist das Training von Neuronalen Netzen sehr abhängig von der Wahl und Anzahl Trainingsdaten (Goldberg, 2017). Da es für spezifische Probleme häufig nur wenige Daten gibt, die zu lösenden Aufgaben aber häufig in einer ähnlichen Domäne liegen wie schon gelöste Probleme, gibt es die Idee, wie der Mensch aus anderen Problemen einer ähnlichen Kategorie zu lernen und das Gelernte zu übertragen. Transferlernen oder auch „learning to learn“ beschreibt den Ansatz aus einer Familie von Aufgaben zu lernen und dieses Wissen zu übertragen. Inspiriert durch den Menschen, der aus wenigen Beispielen häufig sehr gut generalisieren kann und das Gelernte auf andere Aufgaben anwenden kann, gibt es die Idee, dies auch für

Maschinen zu erreichen. Algorithmen könnten also Gelerntes wiederverwenden und als Grundlage für Neues verwenden (Thrun and Pratt, 1998).

Transferlernen sollte trotz vielversprechender Erfolge nur bewusst eingesetzt werden, ansonsten kann dies negative Folgen haben (Pan and Yang, 2010). Das gesuchte Netz muss Informationen über den Inhalt eines Bildes extrahieren können. Dies ist ähnlich zu einem Klassifizierungsproblem, bei dem ein Bild einer bestimmten Klasse zugeordnet wird. Damit eine solche Klassifizierung möglich ist, muss das Netz den Inhalt des Bildes interpretieren können. Ein Netz, das für eine solches Klassifzierungsproblem entworfen wurde, ist das MobileNetV2. Es ist ein Netz mit wenigen Parametern, ausgelegt für leichtere Berechnungen (Sandler et al., 2019). Trotzdem ist es erfolgreich in Bezug auf die ImageNet[1] Klassifizierung, einer der größten frei verfügbaren Datensätze. Es gibt Netze, die eine bessere Klassifzierungsleistung auf dem ImageNet-Datensatz erreichen (with Code, 2021), allerdings werden die gelernten Gewichte verwendet, um sie auf ein anderes Problem zu übertragen. Das bedeutet, eine Klassifizierung die genau auf die ImageNet Klassen optimiert wurde ist nicht nötig. Des Weiteren ist das Bild-Netz ein Teil des Gesamtkonzeptes, das hier prototypisch getestet wird und kein Versuch einen State-of-the-art Bildklassifizierungsalgorithmus zu entwerfen. Außerdem sind die Ressourcen zum Trainieren begrenzt. Das bedeutet eine geringe Anzahl an Parametern, wie es das MobileNetV2 bietet, ist vorteilhaft.

Aus den beschriebenen Gründen wird ein auf ImageNet vortrainiertes MobileNetV2 als Backbone verwendet und um einen Head erweitert, der einen n-dimensionalen Feature-Vektor berechnet. Die Verwendung eines vortrainierten Netzes erfordert, dass die zu verwendenden Daten auf das Format der Daten, mit denen das ursprüngliche Netz trainiert wurde, angepasst werden. Dafür ist eine Vorverarbeitung der Daten notwendig. Die Konstruktion ist in Abbildung 4.1 dargestellt. Das MobileNetV2 wird nach der vorletzten Convolution-Schicht abgeschnitten, und um eine globale durchschnittliche Vereinigung (global average pooling) erweitert, die aus den dreidimensionalen Bild-Feature-Maps einen 1-dimensionalen Vektor berechnet. Zusätzlich wird eine Dropout-Schicht zur Generalisierung sowie zwei Dense-Schichten erweitert. Die letzte Dense-Schicht bildet die Ausgabeschicht und damit den n-dimensionalen Feature-Vektor.

4.3.2 Neuronales Netz zur Textverarbeitung

Aufbauend auf dem Erfolg der Bildverarbeitung, werden Neuronale Netze auch vermehrt bei der Verarbeitung von natürlicher Sprache verwendet (Goldberg, 2017). Deshalb soll auch in dieser Arbeit für die Text Verarbeitung ein Neuronales Netz

[1] `https://www.image-net.org/`

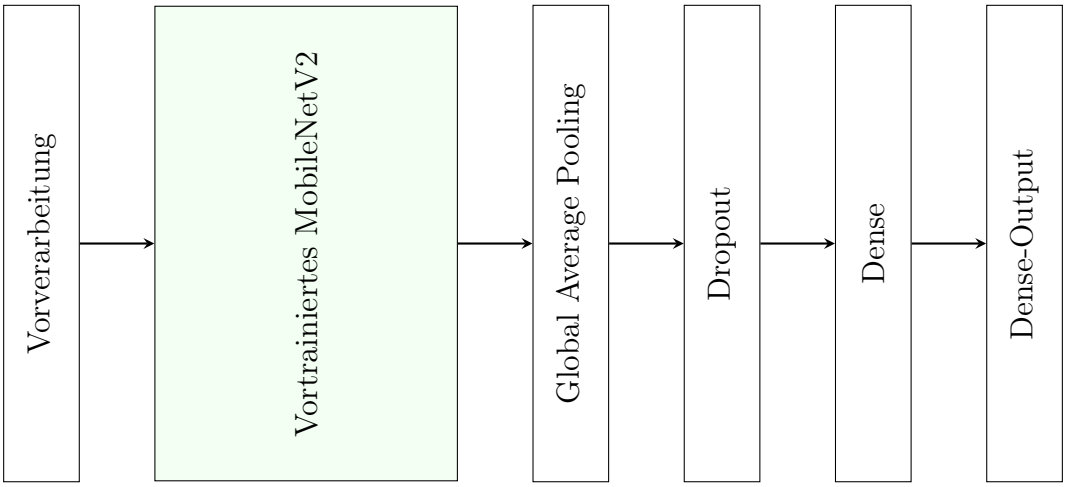


Abbildung 4.1: Schematische Darstellung des Bild-Netzes (eigene Darstellung)

eingesetzt werden. Die Vorverarbeitung der Texte ist allerdings komplexer als die der Bilder, die pixelweise durch ihre Farbwerte repräsentiert werden. Der Text muss zunächst in Token zerlegt werden, die beispielsweise aus Buchstaben oder Wörtern bestehen können. In (Ma et al., 2015) und (Karpathy and Fei-Fei, 2015) werden beispielsweise Wörter als Token verwendet. Dies weist jedoch einige Nachteile auf. Erstens, ist eine Strategie nötig um mit unbekannten oder unerkannten Wörtern umzugehen. Die Wörter können unbekannt sein, wenn sie nicht in dem Vokabular auftauchen oder unerkannt sind, wenn beispielsweise Rechtschreibfehler oder Dialekte die Schreibweise eines Wortes beeinflussen. Zweitens, ist das Vokabular das abgebildet werden müsste, sehr groß. Der Duden schätzt den derzeitigen deutschen Wortschatz auf 300.000 bis 500.000 Wörter (Duden, 2020). Wenn nur eine Teilmenge davon kodiert werden würde, müsste die Teilmenge definiert werden. Drittens muss die Frage beantwortet werden, ob jeder Wort-Vektor unabhängig voneinander ist, oder ob ähnliche Wörter einen ähnlichen Vektor haben. Intuitiver ist die zweite Annahme, dennoch ergibt sich damit ein neues Problem. Die Ähnlichkeit zwischen Wörtern muss dafür definiert sein. Es gibt Modelle, die das abbilden können, wie beispielsweise BERT (Devlin et al., 2019). BERT wurde unter anderem auf dem englischsprachigen Wikipedia trainiert. Abgesehen vom Englischen, ist die Diversität, die Wikipedia bietet, eher kontraproduktiv, wenn es einzig darum geht Kartoffeln von Kohlrabi zu unterscheiden. Möglich wäre in solchen speziellen Bereichen, wie Rezepten, ein eigenes Modell zu trainieren. Andererseits, gibt es keine feste oder einheitliche Definition von Ähnlichkeit in diesem Bereich (Goldberg, 2017). Eine Frühlingszwiebel ist beispielsweise vom Aussehen her, ähnlicher zu einer Gurke als zu einer roten Zwiebel. Aus der botanischen Sicht betrachtet, sind jedoch Frühlingszwiebeln und rote Zwiebeln ähnlicher. Aufgrund der genannten Nachteile und existierender erfolgreicher

Modelle wie beispielsweise (Xiao and Cho, 2016), werden in dieser Arbeit Buchstaben als Token verwendet.

Zusätzlich muss der Text abhängig von der Wahl der Token vorverarbeitet werden. Die Vorverarbeitung enthält die Normalisierung des Textes, wie in Abschnitt 2.2.1 beschrieben, die Augmentierung um die Texte künstlich etwas zu verändern und das Umwandeln der Token in einen entsprechenden Vektor. In Abbildung 4.2 ist eine Skizze des Text-Netzes, das für die Textverarbeitung genutzt wird, abgebildet. Es ist nach dem Vorbild von Xiao und Cho (Xiao and Cho, 2016) aufgebaut. Die eben beschriebene Vorverarbeitung wird um das Auffüllen oder Abschneiden der Texte erweitert, da das Netz eine feste Eingabesequenzlänge benötigt. Das Embedding Layer wandelt den Token-Vektor in einen dichten d-dimensionalen Vektor um. Danach folgen mehrere Konvolutionsblöcke, die anders als bei Xiao und Cho um eine Layer Normalization ergänzt werden. Nach der Konvolution folgt eine Dropout-Schicht, in der einige Werte verworfen werden, um Overfitting entgegenzuwirken. Danach folgt eine LSTM-Schicht und als Ausgabeschicht eine Dense-Schicht welche den n-dimensionalen Feature-Vektor bildet. Die schwarzen Pfeile beschreiben den Informationsfluss, die grünen die Maskierung. Die Embedding-Schicht, berechnet eine Maske, die beschreibt welche Werte zum eigentlichen Text gehören und welche nur aus Auffüllgründen vorhanden sind. Diese Information ist für die Dropout- sowie die LSTM-Schicht wichtig. Das LSTM verarbeitet die sequentiellen Inhalte nicht parallel, sondern sequenziell und kann somit Sequenzen unterschiedlicher Länge verarbeiten. Die Dropout-Schicht verwirft eine bestimmte prozentuale Anzahl an Werten. Deswegen ist es sinnvoll wenn auch diese Schicht tatsächliche Werte verwirft. Da das MaxPooling die Länge der Sequenz halbiert, muss auch die Maske des Embeddings dementsprechend angepasst werden. Daher erhalten die Dropout- und LSTM-Schicht die Maske der Pooling-Schicht.

4.3.3 Übergreifende Verlustfunktion und paralleles Training

Die zwei beschriebenen Netze berechnen jeweils einen n-dimensionalen Feature-Vektor. Es gibt hierbei aber keine explizite Grundwahrheit, gegen die Feature-Vektor verglichen werden kann. Implizit ist der Feature-Vektor des jeweiligen anderen Netz die Grundwahrheit für ein Netz. Der Fehler wird durch die Kosinus-Ähnlichkeit der Beiden berechnet. Das bedeutet, es wird eine Verlustfunktion benötigt die positive Paare zusammenzieht und negative Paare abstößt, ein Contrastive Loss. Es gibt verschiedene Versionen dieser Verlustfunktion, hier wird die in Abschnitt 2.3.2 vorgestellte Variante verwendet. Hierbei gibt es ein positives Paar und $(m - 1)$ negative Paare wobei m der Batch-Größe entspricht. Das passt zu der betrachteten Kombination von Bild- und Textdaten. Ein Batch besteht jeweils aus m Bild-Text Paaren, dadurch gibt es immer ein positives Paar und die negativen Paare bilden

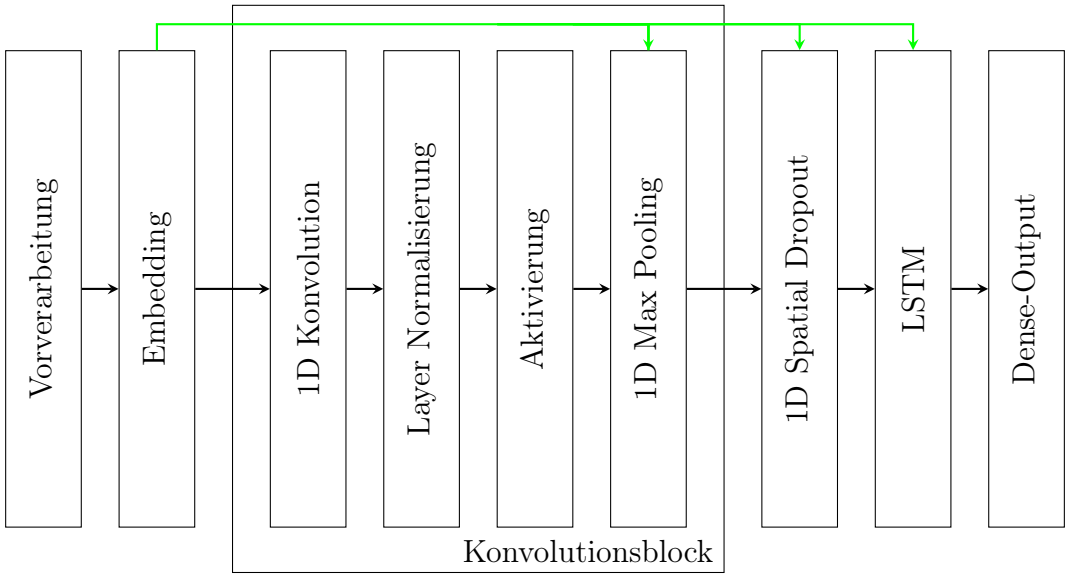


Abbildung 4.2: Schematische Darstellung des Text-Netzes (eigene Darstellung)

die restlichen Kombinationen. Angenommen $t_i \in \text{Text}, b_i \in \text{Bild}$ bilden das positive Paar, so setzen sich die negativen Paare aus $t_i, b_j \in \text{Bild} \setminus b_i$ bzw. $b_i, t_j \in \text{Text} \setminus t_i$ zusammen. Ein Nachteil der Annahme, dass die Beziehung der anderen Bilder beziehungsweise Texte negativ Paare bilden, ist dass dies unter anderem eine falsche Annahme sein kann. Beispielsweise können mehrere Bilder zu einem Text gehören, wenn diese in einem Batch zusammen auftreten, wird mindestens ein Text-Bild Paar fälschlicherweise als negativ Paar gewertet, obwohl es das nicht ist. Des Weiteren sollten sich verschiedene Dessert Rezepte beispielsweise ähnlicher sein als ein Quark- und eine Rinderbratenrezept. Dies kann jedoch in der vorgestellten Verlustfunktion nicht abgebildet werden. Um dem entgegenzuwirken wird ein gewichteter Term w (engl. weights) eingeführt mit dem der Verlust für einen Text-Feature-Vektor l_{t_i} berechnet wird:

$$l_{i,j} = -\log \frac{\mathrm{e}^{\mathrm{sim}(t_i,b_i)/\tau}}{\sum_{j=1,j\neq i}^{N} w_{i,j} \cdot \mathrm{e}^{\mathrm{sim}(t_i,b_j)/\tau}} \tag{4.1}$$

Die Gewichte werden auf Grundlage der Zutaten berechnet. Die Annahme ist, je unterschiedlicher die Zutaten sind, umso unterschiedlicher sind die Rezepte. Die Trainingsdaten werden daher um einen Zutatenvektor erweitert, sodass für jedes negative Paar ein Gewicht berechnet wird, abhängig davon wie viele unterschiedliche Zutaten das betrachtete negative Paar hat. Da die Gewichte sehr datensatzspezifisch sind, werden in den folgenden Experimenten sowohl die gewichtete Variante als auch die ungewichtete Variante getestet.

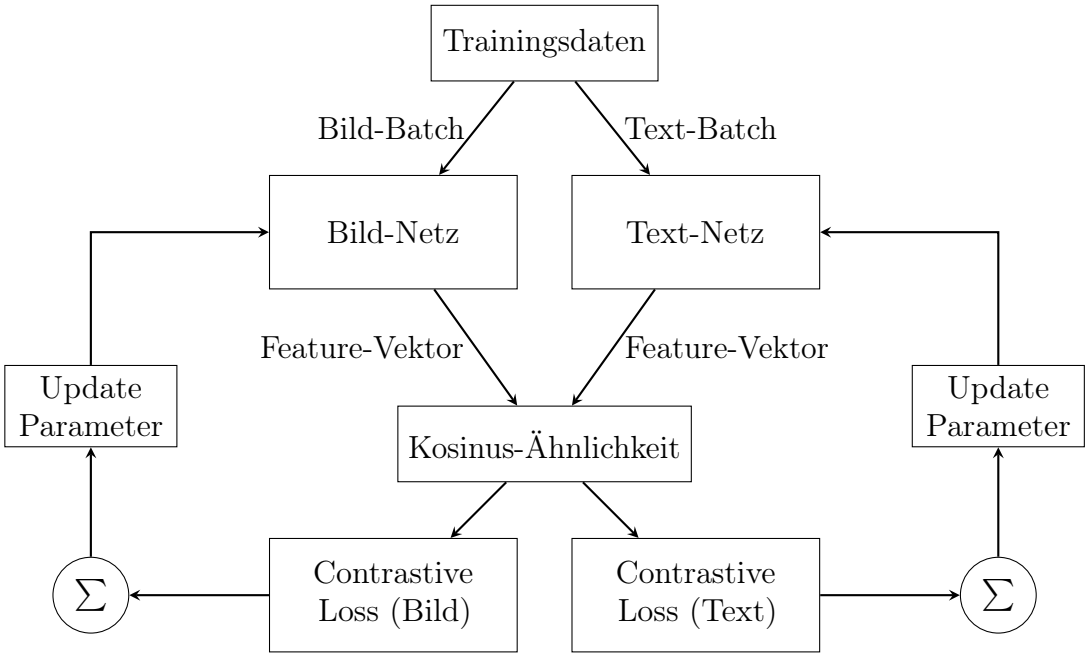


Abbildung 4.3: Schematischer Aufbau des Prototyps (eigene Darstellung)

Eine Schlüsselkomponente des Konzeptes ist, das sich gegenseitige Trainieren der beiden Netze. Wie beschrieben, gibt es keine explizite Grundwahrheit, beide Netze müssen sich aneinander anpassen. Dieser Prozess ist in Abbildung 4.3 dargestellt. Die Trainingsdaten werden in einen Bildbatch und einen Textbatch aufgeteilt und mit dem jeweilig zugehörigen Netz ein Feature-Vektor berechnet. Aus den beiden Feature-Vektoren wird jeweils ein Fehler berechnet und der Durchschnitt über den Batch gebildet. Der so berechnete Fehler wird von dem jeweiligen Optimierer verwendet, um die Parameter der Netze anzupassen. Als Optimierer wird für beide Fälle der Adam Algorithmus (Kingma and Ba, 2017) verwendet, da er als effektiv und robust gilt und daher für einen solchen Prototyp geeignet ist (Goldberg, 2017).

4.4 Zusammenfassung

Das vorgestellte Konzept umfasst zwei Neuronale Netze, die jeweils einen Feature-Vektor der Länge n berechnen. Die Aufteilung in zwei Netze hat zwei wesentliche Vorteile. Zum einen kann dadurch sowohl die Bildinformation sowie auch die Textinformation separat extrahiert werden, da die vorgestellten Netze in ihrer jeweiligen Domäne erfolgreich eingesetzt werden. Dadurch erfüllen sie die Anforderung, dass domänenübergreifend Informationen verarbeitet werden können. Zum anderen kann

dadurch ein Bereich der Skalierbarkeit erfüllt werden. Die Netze werden zur Trainingszeit zusammen trainiert, können jedoch danach unabhängig voneinander verwendet werden. Das kann eingesetzt werden, um unabhängig von einer Abfrage in der Bilderdatenbank für jedes Bild einen Feature-Vektor zu berechnen und diesen zu speichern. Bei der Text-Abfrage muss so nur einmal der Feature-Vektor des Textes berechnet werden und dieser mit den Einträgen in der Datenbank verglichen werden. Bei einer Zunahme an Bildern in der Bilderdatenbank nimmt die Berechnung der Ähnlichkeit bei einer Anfrage nur linear zu. Auf den Aufwand in der Erstellung der Trainingsdaten wird in Kapitel 6 eingegangen.

Eine weitere Anforderung ist die Problemaddressierung, dass das System das Problem möglichst direkt abbildet. Dies wird erreicht, indem eine minimale Vorverarbeitung der Daten gewählt wird und durch die direkte Projektion in einen neutralen Raum, indem Text und Bild auf Ähnlichkeit untersucht werden, ohne das eine Übersetzung notwendig ist. Durch die Wahl der Kosinus-Ähnlichkeit ist zusätzlich gegeben, dass eine Rangfolge in Hinblick auf die Ähnlichkeit gebildet werden kann, da diese auf kontinuierliche Werte zwischen $[-1,1]$ abbildet. Die Kompatibilität zu den späteren Nutzdaten wird erreicht, indem das Text-Netz deutsche Texte und das Bild-Netz farbige Bilder verarbeiten kann. Das postulierte System erfüllt damit, bis auf die Trainingsdaten, die vorgestellten Anforderungen. Im nächsten Kapitel wird die konkrete Implementation vorgestellt und in Kapitel 7 wird das System experimentell überprüft.

5 Implementation

In diesem Kapitel geht es um die konkrete Umsetzung des vorgestellten Konzeptes aus Kapitel 4. Für die Programmierung wird Python[1] in der Version 3.8.10 verwendet. Basierend auf der Programmiersprache Python wird das Framework Tensorflow[2] in der Version 2.5.0 für die Konstruktion und das Training der erstellten Neuronalen Netze genutzt. Zusätzlich wird Keras[3] in der Version 2.4.3 eingesetzt, ein Framework welches auf Tensorflow aufbaut und eine Highlevel-Schnittstelle für bekannte Probleme bietet, wie beispielsweise ein komplexes Neuronales Netz aus verschiedenen Schichten korrekt aufzubauen. Das konkrete Training wird mithilfe der CUDA-Schnittstelle[4] auf einer NVIDIA GeForce RTX 3070 Grafikkarte ausgeführt. Für die Evaluation und Visualisierung der Ergebnisse werden Juypter Notebooks erstellt[5] und Pyplot in der Version 6.1.4 als Interface zur Matplotlib[6] in der Version 3.3.2 verwendet.

Als Erstes wird auf die Generierung der Daten für einen Batch eingegangen. Danach auf das Text-Netz und auf das Bild-Netz. Anschließend wird die Umsetzung der benötigten Verlustfunktion beschrieben und abschließend wird auf die Mechanismen zur Empfehlung der ähnlichsten Bilder eingegangen.

5.1 Data Generator

Die Verwendung eines Data Generators kapselt die Erstellung eines Training- und Validierungs-Batch von dem Trainingsprozess ab. Dadurch sind die Trainingsskripte unabhängig von den tatsächlichen Daten. Für jeden Datensatz gibt es einen separaten Generator. Ein weiterer Vorteil eines solchen Generators ist, dass große Daten, wie z. B. die Bilder zur Trainingszeit, von der Festplatte geladen werden können und nicht im Arbeitsspeicher vorgehalten werden müssen.

[1] https://www.python.org/about/
[2] https://www.tensorflow.org
[3] https://keras.io
[4] https://developer.nvidia.com/cuda-zone
[5] https://jupyter.org
[6] https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html

Darüber hinaus findet die Augmentierung von Text und Bild zur Trainingszeit bei der Erstellung eines Batches statt. Die Texte werden mithilfe des Keyboard-Augmenters der nlpaug-Bibliothek[7] erstellt, der 30% der Wörter augmentiert. Da die Augmentierung zur Trainingszeit stattfindet und für Strings ausgelegt ist, wird der Text erst nach der Augmentierung in einen Text-Vektor umgewandelt. Dies ist zwar rechenintensiver, spart aber Platz im Arbeitsspeicher.

Die Augmentierung der Bilder wird ebenfalls zur Trainingszeit vorgenommen. Dies hat den Vorteil, dass nicht zusätzlich alle augmentierten Bilder im Arbeitsspeicher vorgehalten werden müssen. Die Augmentierung wird mithilfe des Image Moduls von Tensorflow[8] realisiert. Der Vorteil von diesem Modul ist, dass ein ganzer Batch parallel augmentiert, aber dennoch jedes Bild einzeln behandelt wird. In Quellcode 5.1 ist die Augmentierungspipline dargestellt, wobei *bbatch* einen Bild-Batch beschreibt. In Zeile 3 werden die Bilder zufällig entlang der Bild-Breite gespiegelt und in Zeile 4 entlang der Höhe. Wobei dies mit jeweils einer Wahrscheinlichkeit von 50% geschieht. In Zeile 5 wird auf jeden Pixelwert ein zufälliger Wert zwischen $[-0{,}2; 0{,}2)$ addiert, sodass die Gesamthelligkeit des Bildes verändert wird. In Zeile 6 wird der Kontrast der Bilder um einen zufälligen Faktor zwischen $[0{,}5; 2)$ erhöht. Nach der Augmentierung müssen die Bilder normalisiert werden, also auf den Wertebereich von $[-1; 1]$ skaliert und geclipped werden, da die Veränderung des Kontrastes und der Helligkeit einzelne Pixel verändern und dabei den normalen Bereich $[0; 255]$ überschreiten können.

```
1 from tensorflow import image
2
3 bbatch = image.random_flip_left_right(bbatch)
4 bbatch = image.random_flip_up_down(bbatch)
5 bbatch = image.random_brightness(bbatch, 0.2)
6 bbatch = image.random_contrast(bbatch, 0.5, 2.0)
7 normalize_images(bbatch)
```

Quellcode 5.1: Biler Augmentierung

Durch die Verwendung eines Data-Generators wird der Prozess der Batch-Erstellung und die Augmentierung der Daten zentral gekapselt. Die Daten sind somit unabhängig vom Training. Die genaue Art des Trainings wird im folgenden Abschnitt beschrieben.

5.2 Training und Verlustfunktionen

Für das Training werden zwei Trainingsklassen erstellt. Eine für das Trainieren eines Klassifzierungsproblems und die Andere für das Suchen und Abrufen von Bildern.

[7] https://nlpaug.readthedocs.io/en/latest/augmenter/char/keyboard.html
[8] https://www.tensorflow.org/api_docs/python/tf/image

Diese Trennung hilft die spezifischen Tracking- und Trainingsparameter an einer Stelle logisch zusammenzustellen. Für beide Probleme wird ein eigener Trainingsloop geschrieben, der aus einem Trainingsschritt und einem Validierungsschritt besteht. Der Trainingsschritt behandelt genau einen Batch und ist problemspezifisch, weswegen er im Folgenden ausführlicher beschrieben wird.

5.2.1 Bilder Suchen und Abrufen

Bei diesem Problem werden zwei Netze parallel in Abhängigkeit voneinander trainiert. Der Trainingsschritt enthält einen Bild-Batch und einen Text-Batch, die von ihrem zugehörigen Netz verarbeitet werden und jeweils einen Feature-Vektor der Form (batchsize × featurevektor) berechnen. Um mit dem Contrastive Loss aus Abschnitt 2.3.2 den jeweiligen Verlust der Netze zu berechnen, müssen zuvor die Kosinus-Ähnlichkeiten von jedem Bild zu jedem Text und von jedem Text zu jedem Bild berechnet werden. Um rechenintensive Schleifen zu vermeiden, kann dies über eine Matrixmultiplikation realisiert werden. Dies ist schematisch in Abbildung 5.1 dargestellt. Die Matrixmultiplikation ist definiert durch: $\mathbf{A}^{m,n} \times \mathbf{B}^{n,q} = \mathbf{C}^{m,q}$ mit $c_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk}$ und $i = 1, \ldots, m; k = 1, \ldots, q$ (Martin, 2010). Wenn der Bild-Batch transponiert wird, ergibt die Matrixmultiplikation die Matrix von Skalarprodukten der jeweiligen Text- und Bild-Feature-Vektoren aus Abbildung 5.1. Dadurch kann das Skalarprodukt zwischen den Feature-Vektoren effizient berechnet werden. In dieser Form können die Produkte der jeweiligen Normen berechnet werden, unter der Annahme, dass die Normen für die Feature-Vektoren zuerst berechnet werden. Anschließend wird für die Kosinus-Ähnlichkeiten nur noch eine elementweise Division der beiden Matrizen benötigt.

Ein weiterer Vorteil dieser Matrixform der Kosinus-Ähnlichkeiten ist, dass die positiven Paare in der Diagonale der Matrix lokalisiert sind. In Abbildung 5.1 sind diese grün hinterlegt, die negativen Paare sind rot hinterlegt. Die blau-gestrichelte Umrandung deutet die positiven und negativen Paare an, die für die Berechnung des Contrastive Loss von *text1* notwendig sind. Die pink-gestrichelte Umrandung deutet die Werte an, die für *bild1* relevant sind. Die implementierte Verlustfunktion verwendet die Reihen, um den Verlust zu berechnen.

Wie in Abschnitt 4.3.3 beschrieben, wird eine gewichtete und eine ungewichtete Variante des Contrastive Loss getestet. Dabei sind die Gewichte dazu gedacht, dass der Wert der Kosinus-Ähnlichkeit weniger in den Verlust eingeht, wenn es sich um ähnliche Rezepte handelt. Die Ähnlichkeit bezieht sich hierbei auf die Verwendung der Zutaten mit dem Hintergrund, dass beispielsweise Kuchenrezepte (fast) alle Eier und Mehl beinhalten.

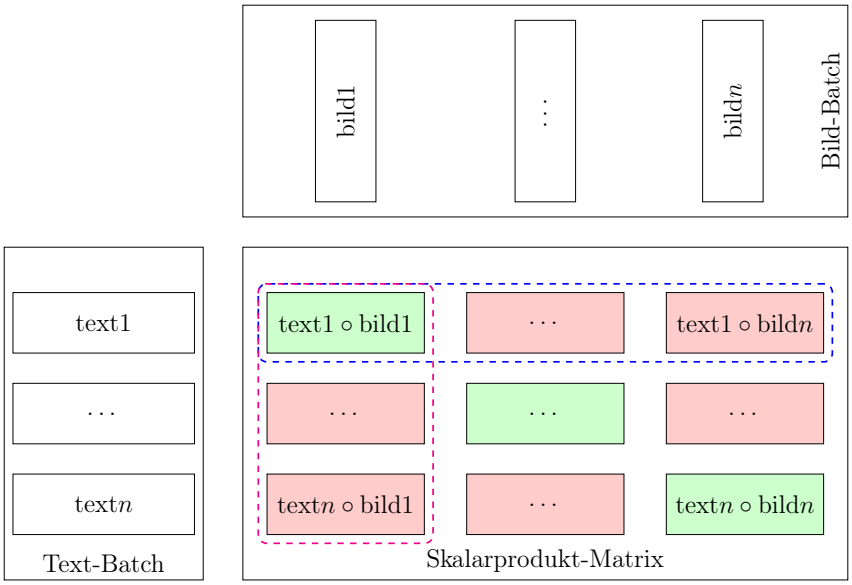


Abbildung 5.1: Berechnung der Skalarprodukte zwischen einem Text- und einem Bild-Batch (eigene Darstellung)

Die Gewichte für die Verlustfunktion berechnen sich aus den Zutaten, die zu einem Rezept gehören. Da ein Bild-Text-Paar immer genau ein Rezept darstellt, gibt es für beide genau einen Zutaten-Vektor. Damit dieser bei den Ähnlichkeiten mit einbezogen werden kann, müssen die Gewichte auch in einer Matrix der Form (Text × Bilder) sein. Das heißt, jeder Zutaten-Vektor muss mit jedem Zutaten-Vektor innerhalb des Batches verrechnet werden. Um Schleifen zu vermeiden, werden die Vektoren transponiert, sodass die letzte Dimension noch mit dem Original übereinstimmt. Dies wird in Zeile 1 von Quellcode 5.2 dargestellt, wobei *zbatch* für Zutaten-Batch steht. Danach werden die transponierten Zutaten-Vektoren vom Original abgezogen (siehe Zeile 2), wobei das Tensorbroadcasting ausgenutzt wird. Die Subtraktion hat zur Folge, dass sich Vektoren bei denen an der gleichen Position eine 1 vorhanden ist, also die dieselbe Zutat enthalten, zu einer 0 ausgleichen. Nur dort wo nur einer der beiden Vektoren an der gleichen Position eine 1 aufweist, ergibt sich eine 1, oder -1. Da danach die Anzahl aller nicht 0er Elemente ermittelt wird, ergibt dies die Anzahl der verschiedenen Zutaten. Die Diagonale ist somit immer 0. Die Anzahl unterschiedlicher Zutaten ergibt das Gewicht. Damit dieses zwischen 0 und 1 liegt, werden die Gewichte am Ende durch das maximale Gewicht dividiert.

```
1 from tensorflow import math
2 zbatch_T = tnp.transpose(zbatch, axes=[1, 0, 2])
3 weights = math.count_nonzero(zbatch - zbatch_T, axis=-1)
```

Quellcode 5.2: Berechnung der Gewichte

Die Verlustfunktion wird zweimal pro Batch aufgerufen, einmal für den Text-Batch und einmal für den Bild-Batch. Grundlage hierfür ist die Kosinus-Ähnlichkeitsmatrix aus Abbildung 5.1 sowie die errechnete Gewichtsmatrix. Die Werte aus der Diagonale bilden die positiven Paare. Nach deren Extraktion wird die Kosinus-Ähnlichkeitsmatrix mit den Gewichten multipliziert. Die Gewichte sind nur für die negativen Paare von Relevanz, da es immer genau ein positives Paar gibt. Dadurch, dass die Gewichte in ihrer Diagonale 0 sind, können die Ähnlichkeitsmatrix und die Gewichtsmatrix elementweise multipliziert werden. Die Nullerdiagonale sorgt für eine Maskierung der positiven Paare. Nach der Multiplikation müssen die Einträge der Reihen aufsummiert werden und dies ergibt die Gesamtsumme der negativen Paare. Wenn die Ergebnisse der positiven und negativen Paare vorliegen, wird der Logarithmus der Division des positiven Paars und des negativen Paares gebildet und der Durchschnitt über die Batch-Größe berechnet. Dies wird in jedem Trainingsschritt sowohl für den Text- als auch für den Bild-Batch berechnet. Auf Grundlage dieses berechneten Verlustes werden die Parameter in den jeweiligen Netzen angepasst.

5.2.2 Hyperparametersuche

Für die Visualisierung des Trainingsprozesses und der Hyperparametersuche wird TensorBoard[9] in der Version 2.5.0 verwendet. Für jedes Problem wird eine Konfigurationsdatei angelegt, in der die zu suchenden Parameter spezifiziert und der Suchraum eingegrenzt wird. Die konkreten Werte der Parameter für einen Suchlauf werden zufällig aus den vorgegebenen Werten ausgewählt. Eine vollständige Suche über den gesamten Parameterraum wäre sehr aufwendig.

5.3 Text-Netz

Das Text-Netz ist nach dem vorgestellten Konzept aus Abschnitt 4.3.2 aufgebaut. Für die Implementierung wird die Funktionale Model-Klasse[10] und die Layer-Klasse[11] von Keras verwendet. In Abbildung 5.2 a) wird dieser Aufbau detailliert gezeigt. Alle blau umrandeten Layer sind direkte Keras Layer. Die orangen Layer sind Custom Layer, die für die Berechnung Keras Layer verwenden, aber notwendig für die Maskierung sind. Die Parameter werden, wie in der Abbildung 5.2 a) zu sehen, gewählt, wobei die kursiv geschriebenen Platzhalter über externe Konfigurationsdateien konfigurierbar sind. Dies ermöglicht eine Flexibilität des Models für eine spätere Hyperparametersuche. Die endgültigen Parameter sind tabellarisch in den nachfolgenden Experimenten in

[9] `https://www.tensorflow.org/tensorboard/`
[10] `https://www.tensorflow.org/api_docs/python/tf/keras/Model`
[11] `https://www.tensorflow.org/api_docs/python/tf/keras/layers`

Kapitel 7 festgelegt. Ein Konvolutionsblock besteht immer aus einem Convolution Layer, Layer Normalization, Activation Layer und einem Pooling Layer, das die Textlänge halbiert. Dadurch muss auch die in dem Embedding Layer berechnete Maske, die angibt welche Werte tatsächlich zu der Sequenz gehören und welche aufgefüllt sind, angepasst werden. Dazu wird das Pooling Layer in ein separates Layer eingebettet. Dieses erhält, im ersten Konvolutionsblock die Maske des Embedding Layers. Dies ist in der Abbildung über die grünen Pfeile angedeutet. Die Maske wird, wie in Quellcode 5.3 beschrieben, berechnet. Für jeden Text wird seine boolsche Maske einzeln betrachtet und die Nachbarn verodert. Dies sorgt dafür, dass Texte mit einer ungeraden Anzahl an Zeichen, um einen Eintrag länger werden. Die Maskierung ist wichtig, damit das LSTM Layer nur die Daten verarbeitet, die tatsächlich zum Eingabetext gehören. Darüber hinaus wird sie auch für das Dropout Layer verwendet, da beim Dropout Werte nicht auf 0 gesetzt, sondern gleichzeitig skaliert werden. Damit nicht über die aufgefüllten Einträge skaliert wird, werden diese vorher alle auf 0 gesetzt. Dafür wird die Maske aus dem Pooling benötigt. Insbesondere wurde 1D Spatial Dropout gewählt, weil diese Art besonders in frühen Konvolutionsschichten besser performen soll (Tompson et al., 2015).

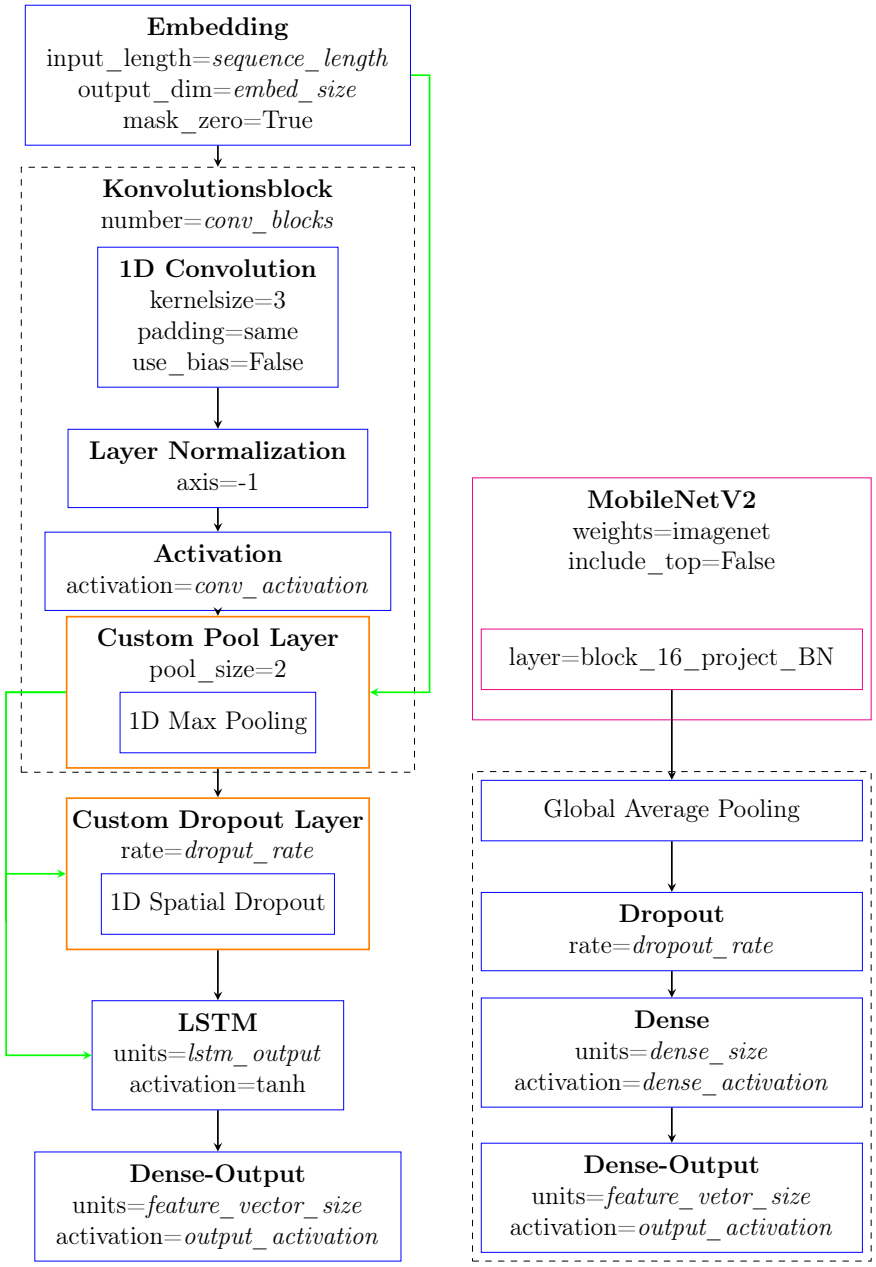
```
1 mask = tf.math.logical_or(mask[:, ::2], mask[:, 1::2])
```

Quellcode 5.3: Poollayer Maskenberechnung

5.4 Bild-Netz

Das Bild-Netz wird wie das Text-Netz auch durch ein Funktionales Keras Model und Layer erstellt und ist schematisch in Abbildung 5.2 b) abgebildet. Im Vergleich zum Text-Netz wird für das Bild-Netz, wie in Abschnitt 4.3.1 beschrieben, zum Transferlernen verwendet. Dafür wird das Netz aus zwei größeren Komponenten zusammen gesetzt. Die erste Komponente ist das Backbone, markiert in Pink, welches aus einem fertigen MobileNetV2[12] besteht, mit Gewichten, die auf dem ImageNet-Datensatz trainiert wurden. Das Backbone hat somit schon gelernt Bilder zu verarbeiten. Die zweite Komponente ist der Head, dieser ist grau-gestrichelt umrandet. Er wird explizit für die eigene Problemstellung entworfen und besteht aus den folgenden vier Schichten: Einem Global Average Pooling, welches für jedes Merkmal über die Höhe und Breite des Bildes den Durchschnitt berechnet und somit einen Vektor unabhängig von der Eingangsbildgröße ausgibt. Zu Generalisierungszwecken wird danach ein Dropout Layer mit einer konfigurierbaren Rate eingefügt sowie zwei Dense Layer verwendet, wovon eines die Ausgabe des Models ist. Diese Layer sind in der Abbildung

[12] https://www.tensorflow.org/api_docs/python/tf/keras/applications/mobilenet_v2



(a) Text-Netz (b) Bild-Netz

Abbildung 5.2: Detaillierter Aufbau der Netze (eigene Darstellung)

blau markiert, weil es vorgefertigte Keras Layer sind, deren Parameter vergeben werden, wie angegeben. Auch hier werden die kursiv dargestellten Platzhalter in der Hyperparametersuche durch konkrete Werte ersetzt. Diese sind tabellarisch in den nachfolgenden Experimenten in Kapitel 7 festgelegt. Bei der Erstellung des Models,

werden die Gewichte des Backbones auf nicht trainierbar gesetzt. Ist eine Anpassung durch ein weiteres Training gewünscht, müssen sie explizit freigegeben werden.

5.5 Bilder Empfehlung

Die Empfehlung der Bilder für einen gegebenen Text wird über eine Datenbankabfrage realisiert. Als Datenbank wird die MongoDB Community Version[13] in der Version 5.0 verwendet. MongoDB ist eine Dokument-Datenbank in der Objekte direkt auf Dokumente abgebildet werden (MongoDB, 2021). Sie wird verwendet, um eine bestehende Bilderdatenbank zu simulieren. Dazu werden die Bilder mit ihrem errechneten Feature-Vektor in der Datenbank abgelegt. Mithilfe einer Aggregation Pipeline wird die Kosinus-Ähnlichkeit zwischen dem Feature-Vektor eines Textes und allen Feature-Vektoren der Bildeinträge berechnet. Die Bilder werden anhand dieses Wertes absteigend sortiert (eine Ähnlichkeit von 1 stellt das Maximum dar). Die zehn ähnlichsten Bilder werden zurückgegeben. Des Weiteren wird sie verwendet um einmalig für alle Texte überprüfen zu können, ob eines der Originalbilder unter den zehn Ähnlichsten ist.

[13] `https://docs.mongodb.com`

6 Datensatz

Eine essenzielle Grundlage für maschinelles Lernen sind die Trainingsdaten. Anhand der Trainingsdaten werden die Parameter in den Netzen eingestellt und optimiert. Deswegen muss der Datensatz, der dafür verwendet wird, die spätere Problemstellung abbilden können. Wenn in der Anwendung beispielsweise verschiedene Essensbilder klassifiziert werden sollen, wird das Ergebnis nicht gut werden, wenn das Netz nur auf Erdbeer Schoko Quarks trainiert wurde. Ein weiterer wichtiger Faktor ist die Größe des Datensatzes. Je größer die Diversität in der entsprechenden Domäne ist, desto besser kann das Netz lernen, über die gesehenen Daten hinaus zu generalisieren. Allerdings sollte hierbei auf Qualität geachtet werden, da zu viel ungewollte Diversität, auch hinderlich sein kann (Goldberg, 2017).

Für diese Arbeit werden zwei Datensätze verwendet, ein künstlich erstellter für die Konzeptionierung und ein komplexerer für die Demonstration einer realen Anwendung. Der Erste ist eine Erweiterung des MNIST[1] Datensatzes, der handgeschriebenen Ziffern beinhaltet. Der zweite Datensatz besteht aus Rezepten die von dem Unternehmen Chefkoch zur Verfügung gestellt wurden. Diese beiden Datensätze werden in den nächsten Abschnitten vorgestellt.

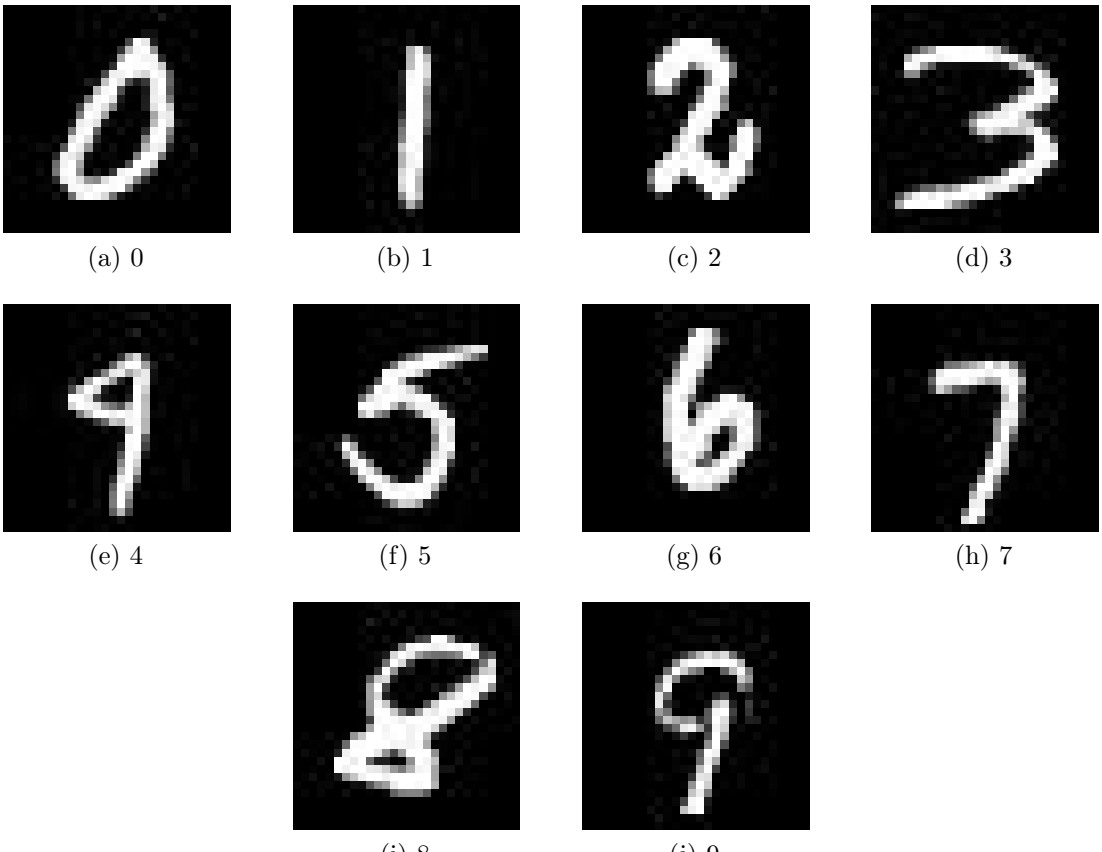
6.1 Erweitertes MNIST

Der MNIST Datensatz ist ein frei verfügbarer Datensatz, der aus einzelnen handgeschriebenen Ziffern zwischen 0-9 besteht sowie einer Grundwahrheit, die versichert, welche Zahl das Bild darstellt. Die Ziffern sind in Graustufen auf den Bildern abgebildet, die Zahl ist weiß und der Hintergrund schwarz. In Abbildung 6.1 sind einige Beispielbilder zu sehen. Alle Bilder haben eine Größe von 28x28 Pixel und die Zahl ist zentriert in der Mitte. Auffällig ist, dass es sich hierbei um die englische Schreibweise der Ziffern handelt, dies ist in Abbildung 6.1 b) an der 1 die aussieht wie die deutsch-geschriebene 7 zu erkennen. Da sich dies jedoch konsistent durch den gesamten Datensatz durchzieht, wirkt sich das nicht weiter aus. Der Datensatz eignet sich gut zum Testen von Klassifizierungsalgorithmen mit minimalen Aufwand von Vorverarbeitung und Formatierung (Yann LeCun, 2021). In dieser Arbeit wird

[1] `http://yann.lecun.com/exdb/mnist/`

dieser allerdings nicht zur Klassifizierung verwendet, sondern für einen künstlichen ImageRetrival Konzeptionsdatensatz. Hierfür werden zusätzlich zu den Bildern, Sätze konstruiert die Zahlen von 0-9 enthalten. Insgesamt wir der Datensatz nach dem 80/20 Prinzip aufgeteilt. Dabei wird darauf geachtet, dass jede Ziffer gleichhäufig vorkommt. Das bedeutet in diesem Fall:

Training	50.400 Bilder und Texte	ca. 80%
Validierung	12.600 Bilder und Texte	ca. 20%



(a) 0 (b) 1 (c) 2 (d) 3

(e) 4 (f) 5 (g) 6 (h) 7

(i) 8 (j) 9

Abbildung 6.1: Beispielbilder des MNIST Datensatz (entnommen aus (Yann LeCun, 2021))

6.1.1 Texte mit Zahlen

Für die Erstellung der Texte werden einzelne Sätze einer Zeichenlänge zwischen 18 und 130 Zeichen (60,64 Zeichen im Durchschnitt) erstellt. Jeder Satz beinhaltet genau

eine Ziffer zwischen 0 und 9. Diese Sätze sollen später vom Text-Netz verarbeitet werden und eine große Ähnlichkeit zu den Bildern mit der entsprechenden Zahl besitzen. Insgesamt gibt es 450 einzigartige Sätze und jeder Satz enthält 1 bis 4 (im Schnitt 1,4) parametrisierbare Aspekte. Bei der Erstellung der Sätze wird darauf geachtet, dass sie grammatikalisch möglichst korrekt sind, also auch an Mehrzahl und Einzahl angepasst. Hierbei werden die „0“ und „1“ als Sonderfälle betrachtet. Die Ziffern stehen am Anfang, in der Mitte und am Ende, um zu verhindern, dass die Lokalisierung der Ziffer eine Rolle spielt. Zusätzlich wurde darauf geachtet, dass der unbestimmte Artikel „ein“/„einer“ vermieden wird. Insgesamt sind die Sätze kurz und einfach gehalten. Die erstellten einzigartigen Sätze werden in einigen Bereichen parametrisiert, sodass die Anzahl an Texten künstlich vergrößert wird und an die Datenmenge der Ziffern aus dem MNIST-Datensatz angepasst werden kann. Ein Beispiel für einen einzigartigen Satz ist: „$N lernt $O die $0-9 zu tanzen.“ Wobei $N für einen Namen, $O für einen Ort und $0-9 für die Ziffern 0-9 steht. Daraus können dann unter anderem die folgenden vollständigen Sätze gebildet werden:

„Frau Mülller lernt auf der Burg die 2 zu tanzen.“
„Bernd lernt im Schwimmbad die 5 zu tanzen.“
„König Karl der Große lernt im Garten die 9 zu tanzen.“

Insgesamt werden Namen, Orte, Essen, Trinken und die Zeit (z. B. Sekunden/Stunden) parametrisierbar gemacht. Die einzigartigen Sätze werden vor der Augmentierung, also dem Ersetzen der Variablen durch ihre Werte, zufällig in Trainings- und Validierungsdaten im Verhältnis von 80/20% aufgeteilt. Dies ist wichtig, damit Training und Validierung sich nicht nur durch ein paar Augmentationen unterscheiden, sondern tatsächlich vom Grundsatz her verschieden sind um eine valide Aussage über die Generalisierung treffen zu können. Auch wenn bei der Augmentierung auf grammatikalische Zusammenhänge geachtet wurde, können am Ende logische Fehler auftauchen, wie z. B. „Das Pentagramm besitzt immer $2-9 Ecken“. $2-9 bedeutet, dass alle Ziffern zwischen 2 und 9 eingesetzt werden können. Dennoch kann dadurch sowohl „Das Pentagramm besitzt immer 5 Ecken“ aber auch „Das Pentagramm besitzt immer 7 Ecken“ generiert werden. Dies ist hier aber kein Problem, da es um das extrahieren der Zahlen geht und nicht um logische Schlussfolgerungen.

6.2 Rezepte von Chefkoch

Der zweite Datensatz ist in Kooperation mit dem Unternehmen Chefkoch[2] entstanden. Das Unternehmen, damals noch pixelhouse media services GbR, wurde 1998 gegründet

[2] `https://www.chefkoch.de`

und bietet mit ca. 350.000 Rezepten eine der größten Koch-Plattformen Europas an. Angefangen mit einer online Rezeptdatenbank, ist Chefkoch mittlerweile auch auf diversen Social Media Kanälen präsent. Gruner+Jahr GmbH, der Mutterkonzern, gibt monatlich die gedruckte Zeitschrift „Chefkoch“ heraus, deren Inhalte sich auf die Plattform beziehen (Chefkoch, 2021). In dieser Arbeit wird von der gesamten Rezeptmenge nur eine deutschsprachige Teilmenge an Rezepten betrachtet.

Der Chefkoch-Datensatz besteht aus einzelnen Rezepten, welche sich zusammensetzen aus: einer Zubereitungsinstruktion, einem Bild, den Zutaten und Kategorien denen das Rezept zugeordnet ist. Die Instruktionen und Bilder sind die Hauptkomponenten des Datensatzes und werden in 6.2.2 und 6.2.3 genauer beschrieben. Das gesamte Set besteht aus 41.162 Rezepten und wird für das Trainieren und Validieren der Netze wie folgt aufgeteilt:

Training	32.000 Rezepte	ca. 78%
Validierung	8.000 Rezepte	ca. 19%
Test	1.162 Rezepte	ca. 3%

Das Test-Set enthält die hauseigenen Rezepte von Chefkoch, die im Rahmen dieser Arbeit zu Demonstrationszwecken verwendet werden. Die restlichen Rezepte werden zufällig auf den Training- und Validierungssatz verteilt. Nach einer einmaligen Aufteilung bleiben diese jedoch für die Vergleichbarkeit über alle Experimente und nachfolgenden Analysen gleich. Ein Rezept besteht immer aus einem Rezepttext und einem Bild. Allerdings gibt es zu manchen Texten mehrere Bilder. Das heißt, in der gesamten Menge an Rezepten kommen manche Texte mehr als einmal vor. Die Anzahl an einzigartigen Rezepttexten beträgt 29.250. Das bedeutet zu jedem Rezepttext gibt es durchschnittlich 1,4 Bilder. Alle nachfolgenden Ausschnitte, Beispiele und Analysen sind dem gesamten Datensatz entnommen, also Trainings-, Validierungs und Testsatz zusammengefasst.

6.2.1 Zutaten und Kategorien

In allen Rezepten zusammen gibt es insgesamt 2213 verschiedene Zutaten, die verwendet werden. Unter anderem zum Beispiel: Butter, Polenta, Paprikaschoten, Trockenfrüchte und viele mehr. Diese werden in Abschnitt 4.3.3 für eine Gewichtung verwendet. Jedes Rezept benötigt mindestens 1, maximal 49 und im Durchschnitt 10,41 Zutaten. Die Kategorien wurden von Chefkoch Mitarbeitenden angelegt und die Rezepte diesen zugeordnet. Jedes Rezept ist 0 bis 34 Kategorien, im Schnitt 7,35 Kategorien, zugeordnet. Insgesamt gibt es 180 verschiedene Kategorien. Die Anzahl wird für diese Arbeit jedoch auf die 35 Kategorien in Tabelle 6.1 reduziert. Die Verteilung gibt an, wie viel Prozent der Rezepte dieser Kategorie zugeordnet

Kategorie (Verteilung)			
Backen	11.281 (38%)	Kinder	2.002 (6%)
Beilage	1.868 (6%)	Kuchen	5.045 (17%)
Braten	3.458 (11%)	Low Carb	3.247 (11%)
Dessert	2.456 (8%)	Nudeln	2.135 (7%)
Europa	3.994 (13%)	Party	3.663 (12%)
Festlich	1.588 (5%)	Rind	2.007 (6%)
Fettarm	1.560 (5%)	Salat	2.218 (7%)
Fisch	1.409 (4%)	Schwein	2.912 (10%)
Fleisch	3.526 (12%)	Snack	3.125 (10%)
Frucht	2.370 (8%)	Sommer	3.542 (12%)
Frühling	1.663 (5%)	Studentenküche	1.671 (5%)
Frühstück	1.211 (4%)	Torte	2.066 (7%)
Geflügel	1.596 (5%)	Vegan	2.361 (8%)
Gemüse	8.971 (30%)	Vegetarisch	12.986 (44%)
Hauptspeise	10.447 (35%)	Vorspeise	2.673 (9%)
Herbst	2.516 (8%)	Weihnachten	1.278 (4%)
Kekse	1.262 (4%)	Winter	2.110 (7%)
Ketogen	1.642 (5%)		

Tabelle 6.1: Die verwendeten Kategorien und ihre Verteilung im Datensatz von Chefkoch

sind. Die Reduzierung ist notwendig, da einige Kategorien inkonsistent zugewiesen oder stark unterrepräsentiert sind.

6.2.2 Rezepttexte

Die Zubereitungsinstruktionen sind die Texte, die im Nachfolgenden als Rezepttexte oder Texte benannt sind. Sie enthalten die Information, wie die Zutaten in ein Gericht umgewandelt werden. In Listing 1 ist ein Beispieltext aus dem Testdatensatz. Es ist die Zubereitungsinstruktion für einen Französischen Kalbsbraten aus der Picardie:

Die Rezepttexte sind zwischen 21 und 6966 Zeichen lang, im Durchschnitt sind es 827,70 Zeichen pro Rezepttext. Der Median beträgt 710 Zeichen, das heißt, die meisten Rezepte sind tatsächlich kürzer als der Durchschnitt, der durch ein paar sehr ausführliche verschoben wird. Zum Vergleich, das oben genannte Rezept besteht aus 906 Zeichen, also länger als das durchschnittliche Rezept. Sonderzeichen und Zeilenumbrüche bleiben für eine bessere Lesbarkeit in den Texten erhalten und werden erst bei der weiteren Verarbeitung eliminiert.

„Das Fleisch salzen, pfeffern und zu einem Braten zusammenschnüren dabei je 1 Zweig Thymian und Salbei dazwischen packen. In einem Bratentopf im heißen Öl sanft und mit etwas Geduld auf allen Seiten anbraten. Die Knochen und die geschälten Zwiebeln darum herum streuen und zugedeckt etwa 70 - 80 Minuten auf kleinem Feuer sachte schmoren lassen. Dabei immer wieder drehen und jedes Mal einen Schuss Fond und etwas Chablis zugießen, damit nichts ansetzt.
Den Braten in Scheiben schneiden und auf einer großen Platte anrichten. Die Sauce durchseihen und abschmecken. Die Zwiebeln mit zarten Böhnchen mischen, die nur kurz blanchiert und in etwas Butter geschwenkt sind.
Für die Röstkartoffeln möglichst gleich große, kleine Kartöffelchen aussuchen, die gar gekocht und in Olivenöl rundum braun gebraten werden - dabei dürfen ruhig einige Salbei-, Thymian- oder Rosmarinzweige mitbraten und ihren Duft abgeben.“

Listing 1: Rezepttext des Französichen Kalbsbraten aus der Picardie, aus dem Datensatz von Chefkoch

6.2.3 Rezeptbilder

Der Datensatz enthält 41.162 Bilder, davon sind ca. 90% von Nutzern der Plattform, die restlichen 10% sind hauseigene professionelle Bilder. Die Bilder können von allen Nutzern auf einer Skala von 0-5 bewertet werden. Die Bewertung der hier verwendeten Bilder liegt zwischen 4 und 5 mit einem Durchschnittswert von 4,43. Die Größe und Form der originalen Bilder variiert stark. Für die Verarbeitung mit dem in dieser Arbeit verwendeten Neuronalen Netz, werden die Bilder auf ein Format von 224x224 Pixel herunter skaliert. Dies passiert in zwei Schritten: als erstes wird das Bild in der größtmöglichen quadratische Form geschnitten und erst im zweiten Schritt wird die Auflösung verringert. Abbildung 6.2 enthält sechs beispielhafte Fotos, die im Test-Datensatz vorkommen. Der Datensatz enthält süße und herzhafte Gerichte, Vorspeisen, Hauptgerichte, Desserts und Getränke wie beispielsweise die Erdbeerbowle. Die Fotos zeigen unterschiedliche Blickrichtungen auf das Essen, wie in c) wo das Essen von oben fotografiert wurde und in d) von der Seite. Auch unterschiedliche Portionsgrößen, wie in b) oder f) in denen eine Portion dargestellt wird, wohingegen in a) der gesamte Kuchen abgebildet ist. Zudem ist auf den Bildern unterschiedlich viel Hintergrund oder Dekoration zu sehen. In b) ist beispielsweise eine Kaffeetasse zu sehen und in c) ein leerer Teller. Das Bild a) mit dem Brownieherz enthält keine weitere Dekoration, es ist nur einzig auf dem Teller abgebildet. Ein weiterer Unterschied im Hintergrund, ist die Farbgestaltung. Die Erdbeerbowle in e) ist auf fast weißem Hintergrund abgebildet, der Französische Kalbsbraten mit

eher dunklem Hintergrund. Außerdem wurde bei den Bildern mit verschiedenen Schärfeefekten gearbeitet, wie beispielsweise in h) und i) wo der Hintergrund und der Vordergrund relativ unscharf sind, die Teekanne im Hintergrund von g) jedoch klar zu erkennen ist.

(a) Brownieherz mit Mascarpone und Himbeeren

(b) Eier im Toastbrot mit Rosmarinbutter

(c) Brokkoli Nudelauflauf

(d) Blumenkohl Bombe

(e) Erdbeerbowle

(f) Hokkaido Kürbissuppe

(g) Erbeer Schoko Quark

(h) Französischer Kalbsbraten

(i) Zimtsterne

Abbildung 6.2: Beispielbilder des Datensatzes

Insgesamt bieten die Bilder eine gewisse Diversität. Beispielsweise durch Beleuchtung, Größe der Objekte oder Anteil an Dekomaterialien und zusätzlich eine gesicherte Qualität, die von den Nutzern der Plattform indirekt durch die Vergabe der Bewertung bestätigt wurde. Die Kategorien werden bereinigt, also große Inkonsistenzen entfernt und nur diejenigen verwendet, die mit mindestens 5% im Datensatz vorkommen. Ebenfalls bieten die Rezepttexte eine gewisse Diversität in ihrer Länge und Schreibweise und bestehen auf jeden Fall aus mindestens einem Satz. Mit dem Datensatz kann also genau das Problem abgebildet werden, dass es einen vorhandenen Text (die Zubereitungsinstruktionen) gibt und mithilfe dieses Textes aus einer Datenbank passende Bilder (Rezeptbilder) gefunden werden können. Als Grundwahrheit für das Trainieren wird hierbei angenommen, dass die Bilder zu ihren zugehörigen Instruktionen passen. Hieraus ergibt sich, dass der Chefkoch Datensatz einen geeigneten Datensatz zur Validierung der vorgestellten Konzeption darstellt.

7 Experimente und Diskussion

In diesem Kapitel wird experimentell überprüft, ob der konzeptionelle Ansatz, der in Kapitel 4 vorgestellt wird, funktioniert und passende Bilder zu einem Text vorschlagen kann. Dafür werden die folgenden vier Experimente durchgeführt. Im ersten wird mithilfe des einfachen erweiterten MNIST-Datensatzes das grundlegende Gesamtkonzept mit der in Abschnitt 2.3.2 beschriebenen Verlustfunktion getestet. In dem zweiten und dritten Experiment werden die zwei Netze separat betrachtet und evaluiert, wie diese auf dem richtigen Chefkoch-Datensatz agieren. Zum Abschluss wird das Gesamtkonzept auf dem Chefkoch-Datensatz bewertet.

7.1 Contrastive Loss und MNIST

Der erweiterte MNIST-Datensatz, der in Abschnitt 6.1 beschrieben wurde, ist ein simples, einfaches Beispiel für Text- und Bildkombinationen. Diese Daten werden parallel verarbeitet, der Text vom Text-Netz und das Bild vom Bild-Netz. Beide berechnen einen Feature-Vektor der Länge $n = 10$. Dies kann hier gut bestimmt werden, da im gesamten Datensatz genau zehn verschiedene Ziffern (0-9) vorkommen. Dies legt nahe, die Dimension des Vektorraumes somit auf $n = 10$ zu beschränken, mit der Idee, dass jede Ziffer eine Dimension darstellt. Als Text-Netz wird das in Abschnitt 5.3 beschriebene Netz verwendet. Für das Bild-Netz wird allerdings ein vereinfachteres Netz, als das in Abschnitt 5.4 vorgestellte verwendet, da dies für farbige und komplexe Bilder konzipiert wurde. Die Ziffern in dem MNIST-Datensatz bestehen nur aus Grauwerten.

Für die Betrachtung der Trainingsparameter sowie für die statistische Auswertung werden Trefferquote und Genauigkeit verwendet. In diesem Experiment wird ein Schwellenwert von 0,95 angelegt, da es um die Bewertung von Ähnlichkeiten geht. Das bedeutet, dass ein Text-Bild-Paar mindestens eine Kosinus-Ähnlichkeit von 0,9 aufweisen muss um als richtig zu gelten. Die Trefferquote sagt also aus, wie häufig ein positives Paar eine Ähnlichkeit größer als 0,9 aufweist. Die Genauigkeit gibt an, wie häufig ein negatives Paar eine Ähnlichkeit unter 0,9 aufweist.

sequence_length	128
embed_size	10
conv_blocks	3
conv_channels	50
dropout_rate	0,4
output_activation	tanh
lstm_output	50
conv_activation	swish
τ	0,7

Tabelle 7.1: Finale Parameterwahl für das MNIST Training

7.1.1 Training

Für das Training werden die Daten so vorbereitet, dass ein Batch immer aus allen Zifferbildern oder vielmehr allen Sätzen mit den entsprechenden Ziffern besteht. Damit wird die Batchgröße automatisch auf zehn festgelegt. Dies stellt sicher, dass jede Ziffer genau einmal in dem Batch vorkommt. Dadurch ist auch sichergestellt, dass alle anderen Ziffern, andere Ziffern sind und somit zurecht genau ein Paar das positive Beispiel bildet und alle anderen Negativbeispiele sind. Deshalb wird hierbei das ungewichtete Contrastive Loss verwendet, da für jedes Beispiel gilt, dass es genau ein positives und neun negative Paare gibt, die auch alle gleich wichtig negativ sind. Eine Unterscheidung, wie negativ die negativen Paare sind, ist dadurch überflüssig. Die Reihenfolge der Ziffern innerhalb eines Batches ist jedoch zufällig. Nach jeder Epoche wird der gesamte Datensatz einmal neu gemischt. Das sorgt dafür, dass die Berechnungen wirklich auf den Daten und nicht auf einer gelernten Reihenfolge beruhen. Als Optimierer wird Adam-Algorithmus mit einer Lernrate von 0,0001 verwendet, sowohl für das Text- als auch das Bild-Netz. Wobei jedes Netz für sich optimiert wird. Die restlichen Parameter werden gewählt wie in Tabelle 7.1 abgebildet.

7.1.2 Auswertung

Die eben beschriebenen Netze werden beide zufällig initialisiert und für 300 Epochen trainiert. Auf dem Validierungssatz wird eine Trefferquote von 87,30% und eine Genauigkeit von 99,55% erreicht. Das bedeutet, obwohl knapp 13% der positiven Paare keine Kosinus-Ähnlichkeit von über 95% aufweisen, weist fast kein negatives Paar eine hohe Ähnlichkeit auf. Um die Auswirkung auf die auf eine bestimmte Textanfrage zu untersuchen, wird ein neuer Text für jede Ziffer entworfen: „Dies ist ein Testsatz, bei dem es einzig um die Zahl $0-9 geht.“ Daraus wird ein Test-Batch ent-

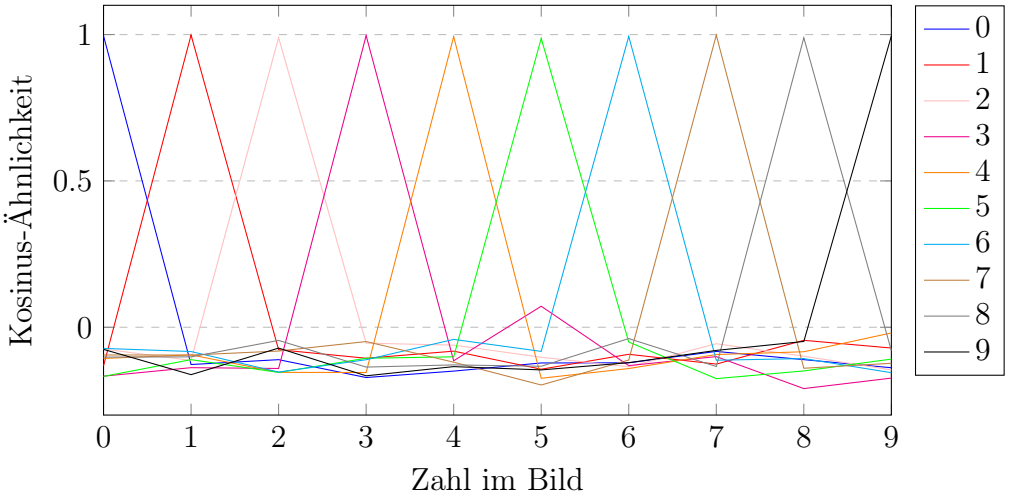


Abbildung 7.1: Kosinus-Ähnlichkeit zwischen Sätzen und Bildern mit gleichen Ziffern

wickelt, der auch bei den Bildern für jede Ziffer ein Bild enthält. Die Feature-Vektoren die sich aus den jeweiligen Batches ergeben, werden auf Ähnlichkeit untersucht. In Abbildung 7.1 sind die Ähnlichkeiten die sich zwischen dem Satz, bei dem $0-9 durch eine Ziffer ersetzt wird und jedem Bild abgebildet. Jede Farbe steht für eine Ziffer im Satz. Dies macht deutlich, dass die Separierung im Vektorraum funktioniert, da die dementsprechenden Ziffern im Text zu den Zahlen eine maximale Ähnlichkeit aufweisen wohingegen sie zu allen anderen orthogonal sind. Die Orthogonalität ist wichtig, da die Feature-Vektoren zehndimensional sind und somit ein maximaler Abstand zu allen anderen nur so möglich ist.

Insgesamt lässt sich sehen, dass das Konzept durchaus sehr gut mit dem einfachen Datensatz funktioniert. Allerdings gibt es einige Nachteile die durch einen solchen künstlichen Datensatz entstehen. Zum einen wurden die Sätze im Training- und Test-Set so augmentiert, dass die Wahrscheinlichkeit sehr groß ist, dass es denselben Satz mit unterschiedlichen Ziffern gibt, wodurch das Netz lernen würde, den Rest des Satzes, abgesehen von der Ziffer, einfach zu ignorieren. Trotz der Tatsache, dass darauf geachtet wurde, in welchem Teil die Ziffer vorkommt, existieren nicht viele Möglichkeiten, wo die Zahl steht. Das heißt, das Netz muss zwar den ganzen Satz verarbeiten, denn sonst würde es mit den Ziffern am Ende nicht funktionieren, dennoch wird es vermutlich keine weiteren Erkenntnisse daraus ziehen können. Zum anderen unterscheiden sich die Bilder der Ziffern sehr von echten, realen Bildern. Nicht nur, dass sie aus Grauwerten gebildet und nicht farbig sind, sondern auch, dass sie perfekt zentriert sind und quasi nur aus Linien und dem Konzept „Zahl“ bestehen. Das wird in keinem realen Anwendungsfall der Fall sein. Um die in Abschnitt 4.3 vorgestellten Netze in Bezug auf Informationsextraktionsfähigkeit bezüglich der

natürlichen Daten verifizieren zu können, werden die beiden Netze zunächst einzeln mit einem vereinfachten Problem betrachtet.

7.2 Klassifizierung auf Kategorie-Basis

Da der Versuch mit dem MNIST-Datensatz gezeigt hat, dass eine ausführliche Interpretation der Ergebnisse schwierig ist, wird zunächst überprüft wie die vorgeschlagenen Neuronalen Netze einzeln in einem Klassifizierungsproblem abschneiden. Ein solches Klassifizierungsproblem kann zeigen, ob die Netze in der Lage sind, die Daten aus dem Chefkoch-Datensatz zu interpretieren und Informationen zu extrahieren. Dabei wird sowohl das Bild-Netz als auch Text-Netz zunächst separat betrachtet. Als Klassen, denen die Netze ihre Eingaben zuordnen werden, dienen die in Abschnitt 6.2.1 vorgestellten Kategorien.

Mit dem entsprechenden Netz (Bild-Netz für Bilder, Text-Netz für Texte) wird jeweils ein Feature-Vektor $v \in \mathbb{R}^{35}$ und $v \in [0,1]$ berechnet und die einzelnen Einträge v_i als Wahrscheinlichkeit für die dementsprechende Kategorie $i \in [0,34]$ interpretiert. Wobei 1 als wahr und 0 als nicht wahr gilt. Die Grundwahrheit wird aus den zugehörigen Kategorien in einen Vektor $k \in \mathbb{R}^{35}$ mit $k_i = 0 \vee k_i = 1$ umgewandelt, der genau dort eine 1 hat, die dem Index der ausgesuchten Kategorien entspricht. Die 35 Kategorien aus Abschnitt 6.2.1 werden dafür als Grundwahrheit angenommen. Als Verlustfunktion wird die Multi-Binäre Kreuzentropie aus Abschnitt 2.3.1 verwendet. Zur Verringerung der Komplexität, werden hierbei die Kategorien als voneinander unabhängige Klassen angenommen, sodass keine, eine oder mehrere unabhängig voneinander wahr sein können. Als Optimierer wird Adam verwendet. Als Metrik wird die Trefferquote und die Genauigkeit mit einem Schwellenwert von 50% betrachtet. Das bedeutet wenn das Netz eine Wahrscheinlichkeit von über 50% berechnet, dass die Ausgabe für diese Kategorie als wahr gewertet wird. Die Trefferquote wird aufgrund des Anwendungsfalles verwendet, da in dieser Arbeit von primärer Bedeutung ist, welche Texte und Bilder mit welcher Kategorie assoziiert werden und nicht mit welcher sie nicht assoziiert werden. Die Genauigkeit gibt darüber hinaus Auskunft ob die Kategorie eher geraten wurde, also immer als wahr angenommen wird oder nicht.

Die Klassifizierung wird im Folgenden zuerst für das Bild-Netz und dann für das Text-Netz betrachtet. Da in beiden Problemen der gleiche Datensatz verwendet wird, nur jeweils entsprechend der Text und im anderen Fall das Bild, bleibt die Verteilung der Kategorien dieselbe. Daher lässt sich für beide eine obere Schranke für den Verlust der Multi-Binären Kreuzentropie aus Abschnitt 2.3.1 berechnen, die auf jeden Fall unterboten werden sollte. Diese obere Schranke ergibt sich aus der

Berechnung mit der Grundwahrheit $g \in G$ und der durchschnittlichen Verteilung a der Kategorien K folgendermaßen:

$$L_{\text{average}} = \sum_{j=0}^{|G|} \sum_{i=0}^{|K|} (-g_{ji} \log(a_i) - (1 - g_{ji}) \log(1 - a_i)) = 11{,}07 \tag{7.1}$$

Das bedeutet, dass die Netze jeweils einen Verlust von 11,07 unterbieten müssen, wenn mehr als die Verteilung gelernt werden soll. Um die Trefferquote und Genauigkeits-Werte einordnen zu können, wird zuerst ein Goldstandard für die Klassifizierung des Datensatzes mit den gegebenen Kategorien erstellt. Wenn nicht anders beschrieben, wird im Folgenden immer der Chefkoch-Datensatz aus dem Abschnitt 6.2 verwendet. Da die Kategorien häufig selbstbeschreibend sind oder vielmehr auch Objekte beschreiben, wie beispielsweise „Kuchen“, wird in den nachfolgenden Abschnitten die Kategorie durch eine kursive Schreibweise: *Kuchen* gegenüber dem Objekt Kuchen abgegrenzt.

7.2.1 Goldstandard

Der Goldstandard bezeichnet ein Verfahren, welches als unübertroffen gilt und deshalb als erstrebenswertes Ergebnis gelten kann (Eckart, 2009). Gerichte zu kochen und zu essen ist für die meisten Menschen vermutlich eine tagtägliche Aufgabe. Deswegen kann davon ausgegangen werden, dass Menschen diese Art der Klassifizierung, also ob es sich um ein Dessert, Gemüse oder Frühstück handelt, sehr gut lösen können. Demzufolge kann eine solche Klassifizierung in diesem Fall als Goldstandard für diese Kategorisierung der Rezepte angesehen werden. Über einen Vergleich der Ergebnisse der Netze mit einem solchen Goldstandard kann eine fundierte Aussage darüber getroffen werden, wie gut oder schlecht das Ergebnis der Netze ausfällt. Um diese Vergleichswerte zu ermitteln, wird der in Anhang A.1 dargestellte Fragebogen mit der Community Edition von Limesurvey[1] erstellt und auf einem privaten Server gehostet.

Der Fragebogen besteht aus einer Auswahl von 141 Rezepten aus dem Test-Satz. Diese wurden so ausgewählt, dass es eine einigermaßen gleichmäßige Verteilung der Kategorien gibt. Daraus werden jeweils 141 Bilder und 141 Zubereitungsanweisungen unabhänig voneinander verwendet, von denen jedes Bild und jeder Text als Frage gewertet wird. Folglich ergibt sich eine gesamte Fragepoolgröße von 282 Fragen. Jedem Teilnehmenden wurden zufällig fünf Bilder und fünf Rezepttexte von den 282 Fragen in zufälliger Reihenfolge angezeigt sowie alle 35 Kategorien. Die Kategorien werden in alphabetischer Reihenfolge angezeigt und behalten diese Reihenfolge

[1] `https://community.limesurvey.org/`

während der gesamten Umfrage bei. Von den Kategorien kann sowohl eine, mehrere oder keine ausgewählt werden. Dadurch wird die Situation, in der sich die Netze befinden, simuliert. Die Umfrage wurde von Studierenden und Mitarbeitenden der Chefkoch GmbH ausgefüllt. Die Studierenden wurden über einen universitätsinternen E-Mail-Verteiler und die Mitarbeitenden über einen hausinternen Newsletter erreicht. Sie wurden darauf hingewiesen, dass die Teilnahme freiwillig ist und zu jeder Zeit abgebrochen werden kann.

Der Fragebogen wurde insgesamt 136 Mal vollständig ausgefüllt. Es wurden keine persönlichen Daten abgefragt oder gespeichert, da dies für die Aufgabe nicht von Bedeutung ist. Demzufolge sind keine weiteren Informationen über die Teilnehmenden bekannt. Da jedem Teilnehmenden fünf Bilder und Texte angezeigt wurden, ist im Schnitt jeder Text und jedes Bild $\frac{136 \cdot 5}{141} = 4{,}82$ Mal klassifiziert worden. Ein Ausschnitt der Ergebnisse ist in Tabelle 7.2 abgebildet und vollständigen Ergebnisse sind in Tabelle A.1 im Anhang abgebildet. Insgesamt ist auffällig, dass die Klassifizierung auf den Texten besser funktioniert als auf den Bildern. Dennoch ist weder die Trefferquote noch die Genauigkeit auf dem Textsatz, mit jeweils knapp 52%, sehr hoch. Die Bilder liegen mit einer Trefferquote von ca. 45% sogar sieben Prozentpunkte unter der Klassifzierungsleistung des Text-Netzes.

Ketogen ist die am schlechtesten klassifizierte Kategorie. Das ist nicht verwunderlich, da es sich um kein verbreitetes Konzept handelt. Weiter ist nicht klar definiert, was genau sehr wenig Kohlenhydrate bedeutet, insbesondere wenn keine Mengenangaben in der Anleitung vorhanden sind. Dennoch ist auffällig, dass die Genauigkeit bei den Bildern sehr viel höher ist, als bei den Texten. Das kann eventuell daran liegen, dass in Texten trotz fehlender Mengenangaben die Zutaten zumindest erwähnt werden und das Fehlen von Kohlenhydraten auffallen konnte. *Fisch* wird trotz des seltenen Vorkommens im Datensatz immer mit einer sehr hohen Genauigkeit erkannt, auch wenn die Trefferquote bei den Bildern nur etwas über 50% beträgt. *Beilage* hingegen, weist zusammen mit *Studentenküche* die geringste Genauigkeit mit ca. 15% auf den Bildern auf. Das Verständnis der Kategorie *Studentenküche* ist vermutlich abhängig davon, ob die Teilnehmenden Studierende sind, was die geringe Genauigkeit erklären könnte. *Beilage* hingegen ist vermutlich recht verständlich, dennoch zeigen die Bilder der Kategorie *Beilage* häufig auch das Hauptgericht zu dem die Beilage serviert wird. Das kann auch die erhöhten Werte auf den Texten erklären. Die Genauigkeit auf *Studentenküche* in den Texten ist hingegen nicht viel höher als auf den Bildern.

7.2.2 Bildklassifizierung

Für die Bildklassifizierung wird das in Abschnitt 4.3.1 vorgestellte Bild-Netz verwendet. Die Eingabe besteht aus den Rezeptbildern in angepasster Größe (224x224

	Texte		Bilder		Verteilung
Kategorie	Treff.	Genau.	Treff.	Genau.	in Prozent
Backen	77,08	81,93	68,51	80,49	37,59
Beilage	41,94	28,26	**25,00**	**15,00**	7,80
Dessert	76,00	27,54	81,03	30,92	9,22
Fettarm	18,18	17,24	20,37	18,97	7,80
Fisch	89,55	**100,00**	54,55	**92,31**	7,80
Fleisch	67,61	36,64	54,10	31,73	9,93
Frucht	58,06	52,17	53,49	35,94	7,80
Gemüse	54,32	87,28	45,10	85,43	40,43
Hauptspeise	80,85	80,57	71,48	85,95	40,43
Herbst	23,73	28,00	11,27	21,62	10,64
Kekse	64,44	78,38	67,24	81,25	7,80
Ketogen	**3,77**	7,14	**9,26**	38,46	7,80
Kuchen	69,81	40,22	58,33	29,79	7,80
Low Carb	18,95	25,35	19,57	30,51	14,18
Rind	42,03	96,67	29,73	73,33	9,93
Schwein	40,98	49,02	21,15	37,93	7,80
Sommer	40,28	24,37	31,51	20,35	10,64
Studentenküche	29,63	19,05	**21,57**	**14,86**	7,80
Torte	82,00	67,21	68,18	77,59	7,80
Vegan	44,44	**80,00**	25,68	**43,18**	10,64
Vegetarisch	50,70	67,92	44,95	60,00	45,39
Weihnachten	63,64	63,64	57,14	72,73	9,93
Winter	33,33	24,66	33,78	32,89	11,35
Gemittelt	52,71	52,56	45,46	47,48	-

Tabelle 7.2: Auswertung des Fragebogens zur Klassifizierung von Bild und Text Daten in Bezug auf die von dem Unternehmen Chefkoch GmbH vorgegebenen Kategorien, welche als Grundwahrheit angenommen wird

Pixel) und neu skaliert zwischen $[-1,1]$. Der Grundwahrheitsvektor wird, wie oben beschrieben, für jedes Bild erstellt. Die Ausgabeschicht des Netzes wird auf eine Größe von $n = 35$ gesetzt. Die Abbildung 7.2 zeigt eine schematische Darstellung. Zur künstlichen Erweiterung des Datensatzes, werden die Bilder während des Trainings einer Augmentierung unterzogen. Die verwendeten Veränderungen sind gängige Methoden (Ruiz et al., 2020) und beinhalten:

- vertikale Spiegelung
- horizontale Spiegelung
- Kontrastierung
- Helligkeit

Spiegelung kann bei der Objekterkennung helfen. Hier ist das als Unterstützung sinnvoll, um das Essen vom Hintergrund zu trennen. Die Veränderung von Kontrast und Helligkeit verursachen ein leichtes Rauschen und tragen damit zur Generalisierung bei.

Das Training des Netzes findet in zwei Schritten statt. Im ersten Schritt wird nur der Head, die für dieses Problem hinzugefügten Schichten, trainiert. Dies ist in Abbildung 7.2 durch die grün gestrichelte Linie angedeutet. In einem zweiten Schritt wird dann ein Teil des Backbones, die vortrainierten Schichten des MobileNetV2, nachtrainiert. Dies ist mit der pink-gestrichelten Linie angedeutet. Die Trennung in zwei Trainingsschritte verhindert, dass die vortrainierten Gewichte durch den anfangs zufällig initialisierten Head irreführend beeinflusst werden. Die Initialisierung der Gewichte der zugefügten Schichten ist zwar zufällig, aber um verschiedene Netzstrukturen im Training zu erproben, werden am Anfang mehrere verschiedene Zufallsinitialisierungen verglichen und die Beste als Initialisierung für alle nachfolgenden verwendet. Der beste Seed hier ist: 1398.

Training

Wie im Implementationskapitel beschrieben sind manche Parameterwerte des Netzes noch nicht festgelegt und es wird untersucht welche Werte sich für die Parameter am besten eignen. Dazu werden zuerst einige Tendenzen mit einer Hyperparametersuche betrachtet, in Abbildung 7.3 ist dies dargestellt. Der Übersichtskeitshalber sind nur die höheren Trefferquote Werte markiert. Die Linien repräsentieren jeweils einen Durchlauf mit den entsprechenden Paramterwerten die auf den Säulen abgebildet sind. Als *unfreez_ layer* wird das Layer des MobileNetV2 bezeichnet, das im zweiten Schritt trainiert wird. Hier lässt sich schon erkennen, dass eine erhöhte *image dense channel* sowie ein tieferes nachtrainieren die Leistung leicht verbessern, aber nicht signifikant sind. Weiteres Trainieren ergibt, dass *block5add* als *unfreezlayer* am besten

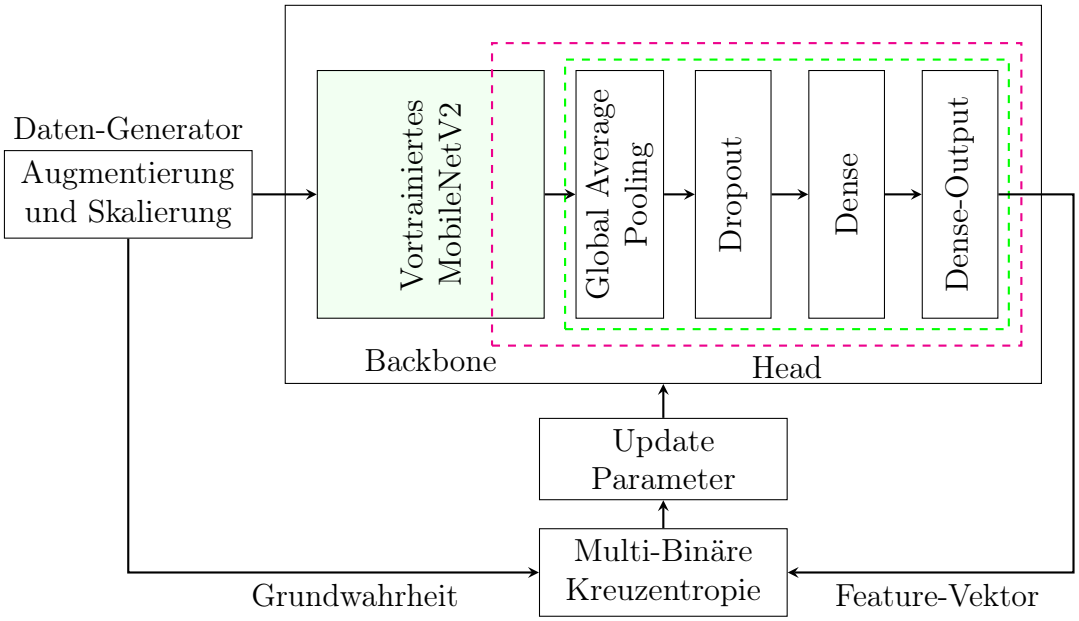


Abbildung 7.2: Schematisches Training der Bildklassifizierung (eigene Darstellung)

funktioniert, dennoch ist der Unterschied nur minimal. Da allerdings die Daten, also die Rezeptbilder, sich vom ImageNet Datensatz unterscheiden, da Essen nur eine Subkategorie darstellt, spricht dies dafür in der Tiefe mit einer niedrigen Lernrate nachzutrainieren.

In Abbildung 7.4 ist zu sehen, dass bei einer Lernrate von 0,001 (blau) im Vergleich zu 0,0001 (rot), die Werte ein bisschen mehr schwanken, und im Endeffekt aber nicht signifikant besser sind. Deswegen wird hier für ein gemäßigtes Training eine Lernrate von 0,0001 verwendet. Die Batchsize wurde von 100 auf 10 heruntergesetzt, da es sonst zu einem starken Overfitting kommen würde. Zusätzlich wurde noch das

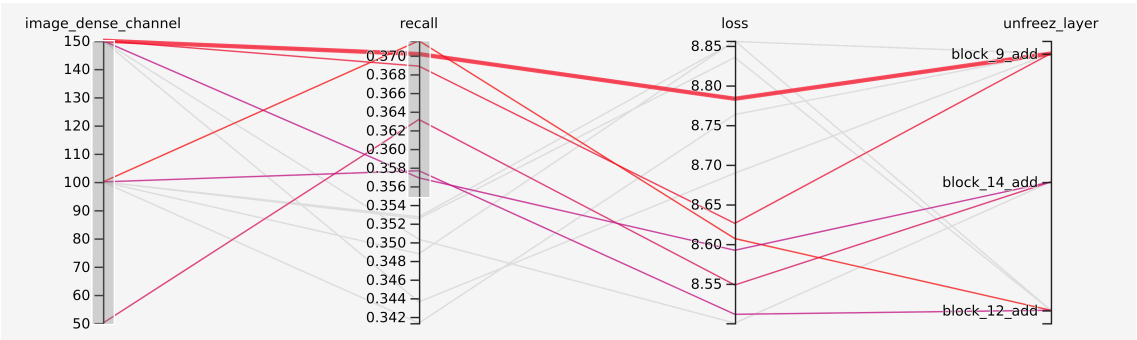


Abbildung 7.3: Ausschnitt aus der Hyperparameter Suche für das Bildnetz

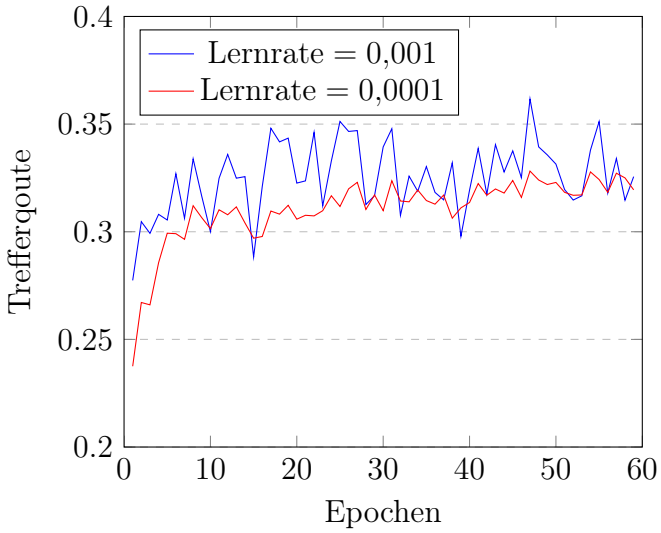


Abbildung 7.4: Einfluss der Learning-Rate auf das Image Netz

Parameter	Wert
dropout_rate	0,3
feature_vektor_size	35
output_activation	sigmoid
Lernrate	0,0001
unfreeze_layer	block_5_add
dense_size	tanh
dense_size	200

Tabelle 7.3: Finale Parameter-Auswahl Bild-Netz

DropoutLayer hinzugefügt um eine Verbesserung der Generalisierung zu erreichen. In Abbildung 7.5 ist abgebildet, wie sich die Größe des ersten DenseLayer nach den Konvolutionen verhält. Abgebildet ist hier in rot eine Schicht mit 200 Neuronen und in blau eine mit 100 Neuronen. Die Verbesserung der Trefferquote setzt sich auch nach dem zweiten Trainingsschritt fort. Dennoch ist auffällig, dass das Netz nur sehr niedrige Werte in der Trefferquote erreicht, unabhängig von der Wahl der Parameter. Auf der Basis dieser Betrachtungen werden, die in Tabelle 7.3 genannten Werte, gewählt. Die Dense-Aktivierung *tanh* wird im Hinblick auf die spätere Problemstellung gewählt, in der auch negative Werte valide sind im Gegensatz zu Klassifizierung.

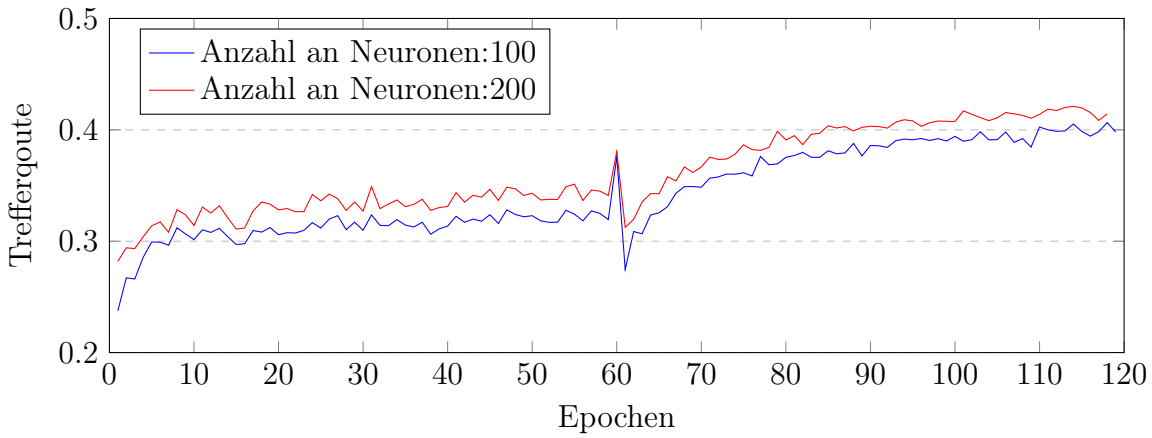


Abbildung 7.5: Vergleich Größe des Dense Layers

Statistische und Qualitative Auswertung

Das oben beschriebene Netz wurde mit einer Batchsize von 10 und 60 Epochen im ersten Schritt, sowie 41 Epochen im zweiten Schritt trainiert. Der Validierungsverlust lag am Ende bei 8,17 ein wenig unter 11,07, dem Verlust der durchschnittlichen Verteilung. Das bestätigt, dass das Netz mehr als nur die Verteilung der Daten gelernt hat. In Tabelle 7.4 sind ausgewählte klassenspezifische Genauigkeit- und Trefferquotenwerte für den Validierungssatz, den Testsatz sowie dessen Teilbereich der Umfrage um einen direkten Vergleich mit dem Goldstandard aus Tabelle 7.2 zu ermöglichen, abgebildet. Die vollständige Tabelle ist in Anhang A.2 abgebildet. Aufgrund der mittelmäßigen Werte werden danach explizit zwei Bilder mithilfe der GradCam-Methode aus Abschnitt 2.1.1 untersucht, um ein besseres Verständnis für die Entscheidung des Netzes für eine bestimmte Klasse zu erlangen.

Hauptspeise wird von den vorgegebenen Kategorien am besten erkannt, sowohl auf dem Validierungs-, als auch auf dem Testsatz. Die Genauigkeit ist relativ hoch was vermuten lässt, dass das Netz einige relative sichere Indizien zu erkennen gelernt hat. *Dessert* hingegen wird im Testsatz nur halb so gut erkannt wie auf dem Validierungssatz. Das kann unter anderem daran liegen, dass das Netz eher klassische Nachspeisen gelernt hat, angerichtete Kleinigkeiten, geschichtete Frucht in Gläsern oder Eis. Generell scheint etwas in Gläsern sehr auf *Dessert* hinzuweisen. Im Testsatz hingegen sind sehr viele Kuchen, Muffins oder Stieleis als Dessert markiert, die nicht erkannt werden. Interessant ist auch, dass *Vegetarisch* sehr viel besser und mit einer sehr viel höheren Genauigkeit erkannt wird als *Vegan*. *Vegetarisch* kommt in dem Datensatz zwar 5,5 Mal so häufig vor wie *Vegan*, allerdings hat die Kategorie eine ca. zehn mal bessere Trefferquote als *Vegan* und damit mehr als durch die reine

Verteilung der Daten zu erwarten wäre. Auch die Genauigkeit ist durchweg niedriger. Es wirkt, als hätte das Netz zwar gelernt *Vegetarisch* zu erkennen, dafür *Vegan* aber eher zu raten. *Weihnachten* wird gut erkannt, im Vergleich zu anderen größeren Konzepten wie z. B. *Sommer*, dies liegt vermutlich an dem Gebäck, was häufig in Sternen- oder Tannenform gebacken wird oder an der Dekoration, die auf den Bildern zum Teil noch sichtbar ist. Durch eine Genauigkeit von 100 % wirkt es so als habe das Netz aussagekräftige Indizien die für *Weihnachten* sprechen gelernt. Allerdings nicht alle, sonst gäbe es eine bessere Trefferquote. Auffällig ist auch, dass *Rind* und *Schwein* auf dem Validierungssatz eine ähnliche Trefferquote haben, die von *Fleisch* aber nur halb so groß ist. Das liegt vermutlich aber auch daran, dass bei ca. 50% der Vorkommnisse von *Rind* oder *Schwein*, *Fleisch* nicht als Kategorie mit angegeben wird.

Im Vergleich zum Goldstandard ist auffällig, dass *Backen* in beiden Fällen zwar eine ähnliche Genauigkeit besitzt, dennoch vom Bild-Netz ein wenig besser erkannt wird. *Dessert* hingegen wird mehr als drei mal so gut von den Menschen erkannt, als von dem Bild-Netz. Allerdings mit einer nicht so hohen Genauigkeit, sodass davon auszugehen ist, dass die Teilnehmenden Vieles als *Dessert* angesehen haben. Vermutlich häufig auch Kuchen- oder Tortenstücke, die in der Grundwahrheit aber nicht als *Dessert* vermerkt sind. Kuchen und Torten, sind scheinbar nicht konsequent als *Dessert* kategorisiert worden. Bei *Kuchen* und *Torten* ist auch auffällig, dass Menschen Torten besser erkannten, auch mit einer sehr viel höheren Genauigkeit, wohingegen das Netz Kuchen deutlich besser als *Torte* erkannte. *Kuchen* kommen hingegen mehr als doppelt so häufig im Datensatz vor wie *Torten*, das könnte die Diskrepanz im Netz erklären, dennoch sehen Kuchen und Torten ähnlich aus sodass die erhöhte Genauigkeit der *Torte* in beiden Fällen im Vergleich zum *Kuchen* auffällig ist. Auffallend ist zudem, dass *Fettarm* vom Netz gar nicht gelernt wird, wohingegen *Low Carb* auf allen Datensätzen zumindest ein wenig erkannt wird, sich im Goldstandard aber *Fettarm* und *Low Carb* mit jeweils ca. 20% Trefferquote die Waage halten.

GradCam ist eine Methode um klassenspezifische Aktivierungen in Bildern zu zeigen. Dies kann helfen, zu verstehen auf welcher Basis ein Netz seine Entscheidung getroffen hat. Um über die reine Statistik hinaus das Netz zu interpretieren, wird die Methodik hier an zwei Beispielen angewendet. Zu den Beispielen werden jeweils die GradCam Bilder angezeigt, deren Kategorie eine Wahrscheinlichkeit von ca. 50% oder mehr aufweist.

Französischer Kalbsbraten: In Abbildung 7.6 sind schwarz-weiß Bilder des Französischen Kalbsbraten abgebildet, überlagert mit drei verschiedenen Heatmaps von drei Voraussagen des Netzes, die eine hohe Wahrscheinlichkeit aufzeigen. Je röter die Färbung auf dem Bild ist, umso höher ist die Aktivierung. Die Grundwahrheit für dieses Bild, alphabetisch sortiert, ist: *Europa*, *Festlich*, *Gemüse*, *Hauptspeise*, *Herbst*,

	Validierung		Test		Umfrage	
Kategorie	Treff.	Genau.	Treff.	Genau.	Treff.	Genau.
Braten	24,88	53,69	19,87	52,54	16,67	50,00
Dessert	43,02	69,86	23,53	52,17	**23,08**	50,00
Fleisch	13,48	40,85	10,06	34,69	14,29	33,33
Frucht	9,23	51,75	5,65	53,85	9,09	50,00
Frühling	15,86	55,38	12,96	77,78	**9,09**	**100,00**
Hauptspeise	**80,05**	78,17	72,82	75,81	78,57	83,02
Kuchen	67,00	72,33	52,76	65,15	54,55	50,00
Low Carb	2,69	32,88	7,64	54,55	**10,53**	**100,00**
Rind	28,32	45,25	27,48	60,00	21,43	75,00
Schwein	24,96	39,14	15,48	40,62	27,27	42,86
Sommer	13,56	53,39	6,78	53,33	**28,57**	**80,00**
Torte	69,43	71,41	40,00	40,00	36,36	66,67
Vegan	7,87	42,02	3,17	10,53	**6,67**	25,00
Vegetarisch	61,39	62,42	68,72	52,54	**77,78**	67,12
Weihnachten	26,06	53,18	18,31	52,00	**42,86**	**100,00**
Gemittelt	41,66	68,50	36,22	63,05	38,07	70,88

Tabelle 7.4: Auswertung des Bild-Netzes zur Klassifizierung von Bild Daten

Rind, *Sommer*. Die Kategorie *Hauptspeise*, abgebildet in a), wurde am sichersten erkannt mit einer Wahrscheinlichkeit von 93%. Hieran lässt sich gut erkennen, dass die größte Aktivierung gleichmäßig über Teile des Fleisches, der Kartoffeln sowie den Tellerrand verteilt ist. Das lässt darauf schließen, dass nicht nur das Essen auf dem Teller wichtig ist, sondern auch der Tellerand und die damit verbundene Abgrenzung des Essens zum Tisch. Dies ergibt insofern Sinn, dasd *Hauptspeise* ein generelleres Konzept ist, für das verschiedene Indizien sprechen, unter anderem auch ein runder Teller auf einem Tisch oder einer Oberfläche. *Braten* wird nur noch mit einer Wahrscheinlichkeit von 56% erkannt, was unter anderem daran liegen kann, dass *Braten* nicht mit in der Grundwarheit aufgeführt wird, obwohl es ein Kalbsbraten Rezept ist. Die Aktivierung in Bild b) ist in gleichmäßiger über das Bild verteilt und umfasst damit auch die Tischkante, aber auch den Rest des Bratens, also der Teil der im Hintergrund liegt. Vermutlich sind die geraden Kanten ein Indiz für *Braten*, dabei wird die Kante des Tisches fälschlicherweise erkannt. Dennoch sind auf Bildern von Braten häufig Scheiben von dem Fleisch abgeschnitten, was Kanten zu einem validen Indiz macht.

Die Kategorien *Schwein* und *Rind* weisen eine ungefähr gleich hohe Wahrscheinlichkeit von ca. 50% auf. Die Aktivierung von *Rind* ist in Bild c) abgebildet und weist die stärkste Aktivierung direkt auf den Bratenscheiben und auffälligerweise auf den

Thymianzweig in der oberen linken Ecke. Braten erhalten häufig Gewürzbeilagen, wie Thymian, Rosmarin oder Zwiebeln. Diese Gewürze sind somit scheinbar ein Indiz, auch wenn sie in diesem Fall nur als Dekoration auf dem Bild zu sehen sind. Die Kategorien *Europa*, *Festlich*, *Sommer*, *Herbst* und *Gemüse* werden nicht erkannt. Die ersten vier, werden generell vom Netz nur sehr selten erkannt, Gemüse hingegen hat mit einer Trefferquote von 73% auf dem Umfragesatz, einen relativ hohen Wert. Kartoffeln sind allerdings auch nicht das, was typischerweise als Gemüse bezeichnet wird. Also wurde das Gemüse (in diesem Rezept sind die grünen Bohnen gemeint, Abschnitt 6.2.2) auf dem Foto einfach weggelassen und konnte somit nicht erkannt werden. An diesem Beispiel lässt sich darüber hinaus die Inkonsequenz von der Kategorie *Fleisch* erkennen, die in diesem Fall nicht als Grundwahrheit angegeben wurde, obwohl Rind definitiv Fleisch ist.

(a) Hauptspeise (b) Braten (c) Rind

Abbildung 7.6: GradCam Heatmaps auf den Französichen Kalbsbraten Bildern

Erdbeer Schoko Quark: Ein zweites Beispiel ist der Erdbeer Schoko Quark in Abbildung 7.7. Das Oringianlbild ist jeweils in schwarz-weiß und mit einer klassenspzefischen Heatmap überlagert. Je röter die Färbung auf dem Bild ist, umso höher ist die Aktivierung. Die Grundwahrheit für das Bild ist: *Dessert*, *Frühling*, *Sommer*, *Vegetarisch*. Die Kategorie *Dessert*, dargestellt in Bild a), hat hier mit einer Wahrscheinlichkeit von 62% die höchste Wahrscheinlichkeit. Hier bestätigt sich die Vermutung von oben, dass Gläser ein wichtiges Indiz für *Dessert* sind. Das lässt sich in diesem Fall daran erkennen, dass der Glasrand unten links eine hohe Aktivierung aufweist, sowie der Bereich zwischen den Gläsern und Bereiche des rechten Glases. Die Aktivierung in Bild b) bei der Kategorie *Sommer* ist recht ähnlich, außer dass hierbei der rote Teil beider Gläser eine höhere Aktivierung aufweist. Dies kann vermutlich an der Kombination von einem rötlichen Dessert in der Kategorie *Sommer* kommen.

Beim *Frühling* hingegen, Bild c), werden sehr viele Bereiche des Bildes aktiviert, besonders stark auch der Hintergrund oben rechts. Was eventuell daran liegt, dass

Frühlingsbilder in diesem Datensatz eher mit einem helleren Hintergrund abgebildet werden. Bei einer Durchsicht der als *Frühling* vom Netz klassifizierten Bilder bestätigt sich, dass diese häufig aus roten Früchten und weißer Sahne bestehen, sowie auch dieses Bild und dadurch die Aktivierung auf dem weißen Hintergrund erklären kann. Dies kann auch erklären, dass die Klassifizierung von *Frühling* im Netz generell nicht gut ist, da es sämtliche Gemüse-Frühlingsbilder, wie beispielsweise grünen Spargel, nicht als solche erkennt. Auch bei diesem Beispiel ist auffällig, dass die Kategorie *Frucht* hier nicht als Grundwahrheit deklariert wurde, obwohl Erdbeeren ja Früchte sind. Diese leichte Inkonsistenz kann ein Grund dafür sein, dass *Frucht* generell schlecht erkannt wird.

(a) Dessert (b) Sommer (c) Frühling

Abbildung 7.7: GradCam Heatmaps auf den Erbeer Schoko Quark Bildern

7.2.3 Textklassifizierung

Für die Textklassifizierung wird das in Abschnitt 4.3.2 konzipierte Text-Netz eingesetzt. Als Eingabe werden die vorverarbeiteten Rezepttexte verwendet und der Grundwahrheitsvektor wird, wie oben beschrieben, für jedes Rezept erstellt. Die Rezepttexte werden normalisiert indem zuerst alle Zeilenumbrüche und Absätze entfernt werden. Des Weiteren werden alle sprachlichen Akzente wie beispielsweise é, ô entfernt oder vielmehr durch e, o. ersetzt werden. Die deutschen Akzente ä, ö, ü bleiben jedoch bestehen, da sie Teil der Sprache sind. Ein paar der Rezepte enthalten Links zu anderen Webseiten, diese werden entfernt. Groß- und Kleinschreibung wird beibehalten, da diese im Deutschen charakteristische Merkmale aufweisen kann (Duden, 2021a). Um den Textsatz künstlich zu vergrößern und robuster gegenüber Tippfehler zu machen, werden zur Trainingszeit synthetische Fehler in Form von Tastatur-Tippfehlern eingeführt (Belinkov and Bisk, 2018). Diese ersetzen zufällige Buchstaben durch Buchstaben deren Tasten nebeneinander liegen.

Der daraus resultierende Text wird anhand des folgenden Vokabulars in einen Text-Vektor aus Zahlen übersetzt:

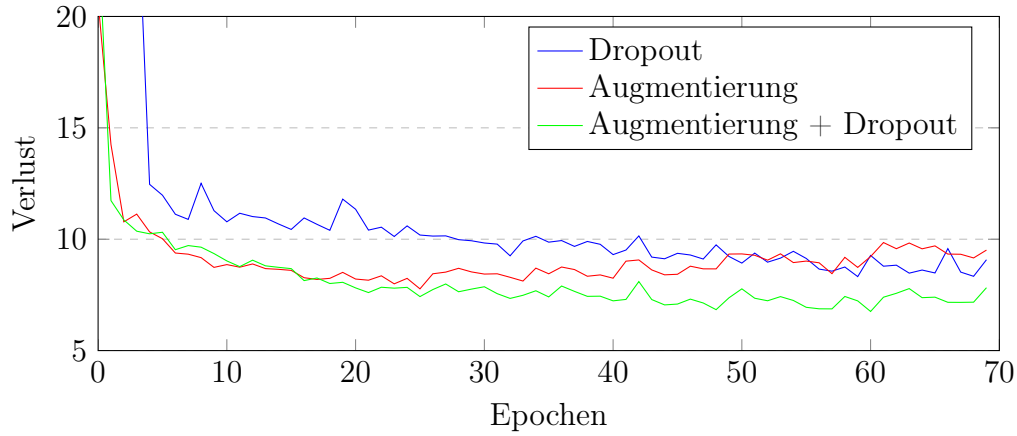
0123456789abcdefghijklmnopqrstuvwxyzäöüß
ABCDEFGHIJKLMNOPQRSTUVWXYZÄÖÜ?!:,. "_-'°Ø%€℃

Die Indizierung beginnt bei 1, Sonderzeichen, die nicht in dem Vokabular vorkommen, werden entfernt. Der sich daraus ergebene Text-Vektor bildet dann die Eingabe für das Netz. Um die Texte durch eine feste Eingabeschicht und den darauffolgenden Konvolutionen verarbeiten zu können, werden die Texte auf eine Länge von 1600 Zeichen aufgefüllt oder gekürzt. Dabei werden immer noch ungefähr 92% der Texte vollständig abgedeckt. Die kürzeren Texte werden mit Nullen aufgefüllt. In der ersten Schicht des Netzes wird dann dementsprechend eine Maske gebildet, sodass das LSTM, welches sequenzielle Daten unterschiedlicher Länge bearbeiten kann, auch nur den tatsächlichen Text analysiert und die aufgefüllten Nullen ignoriert.

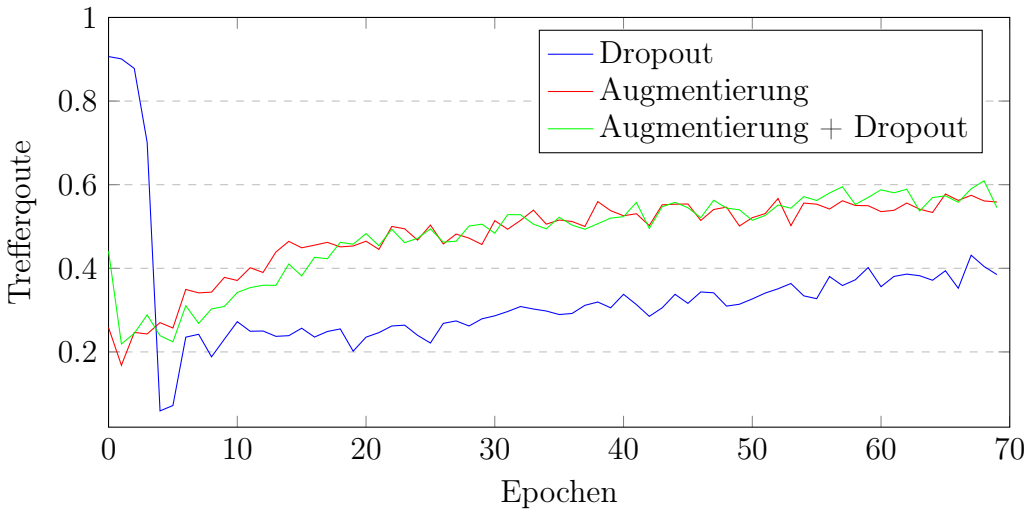
In Abbildung 7.10 ist der Trainingsablauf schematisch dargestellt. Der Datengenerator bereitet die Daten wie beschrieben auf. Diese werden dann vom Text-Netz verarbeitet. Der berechnete Feature-Vektor wird mit der Grundwahrheit verglichen und auf Grundlage des Verlustes sowie der Netz-Struktur werden die Parameter innerhalb des Netzes angepasst. Da hier kein vortrainiertes Netz verwendet wird, und alle Parameter zufällig initialisiert sind, wird alles in einem Schritt trainiert. Wie auch bei dem Bild-Netz werden verschiedene Zufallsinitialisierungen verglichen. Für dieses Netz wird ein Seed von 848 gewählt.

Training

Auch für das Text-Netz wird der Einfluss verschiedener Parameterwerte untersucht. Um den Effekt des Dropout-Layers und der Tippfehler Augmentierung am Text-Validierungssatz zu überprüfen, wird der Vergleich, der in Abbildung 7.8 dargestellt ist, gestartet. die anderen Parameter sind fixiert. Auf demselben Validierungssatz werden drei Läufe verglichen. Einer mit einer Dropout-Rate (in blau) von 0,3, einer Augmentationsrate (rot) von 0,3 und einer mit beidem gleichzeitig aktiv (grün). Auffällig ist, dass pures Dropout zwar einen geringeren Verlust am Ende hat, aber dafür eine deutlich schlechtere Trefferquote als eine reine Augementierung. Die Trefferquote der Kombination von Dropout und Augementierung ist zwar nicht signifikant besser als die bei der reinen Augmentierung, weist aber einen geringeren Verlust auf. Das heißt, dass Dropout zur besseren allgemeinen Generalisierung beiträgt, wohingegen sich Augmentation speziell auf die Verbesserung einiger Klassen auswirkt, woraufhin die Trefferquote höher ist. Da die Kombination am besten abschneidet, wird beides eingesetzt.



(a) Einfluss auf den Verlust



(b) Einfluss auf die Trefferquote

Abbildung 7.8: Text Augmentation und Dropout

Um die Auswirkung der Parameter *conv_ channels*, die Anzahl der Kanäle in den Konvolutionen, des *dropout*, der *lstm_ channel*, die Ausgabelänge des LSTMs und der *Lernrate* zu betrachten, wird eine Hyperparametersuche gestartet. Die Ergebnisse sind in Abbildung 7.9 abgebildet. Die Linien zwischen den Balken deuten immer eine Kombination von Parametern an, die zu einem bestimmten Verlust (Loss) oder auch Trefferquote (Recall) führen. Die Farbe ist anhand der Trefferquote orientiert und geht von rot nach blau, wobei rot die beste Trefferquote ist und blau die schlechteste. Das bedeutet anhand der Farben kann nachverfolgt werden, welche Kombination von Parametern zu den Werten geführt hat. Für dieses Problem funktioniert also eine

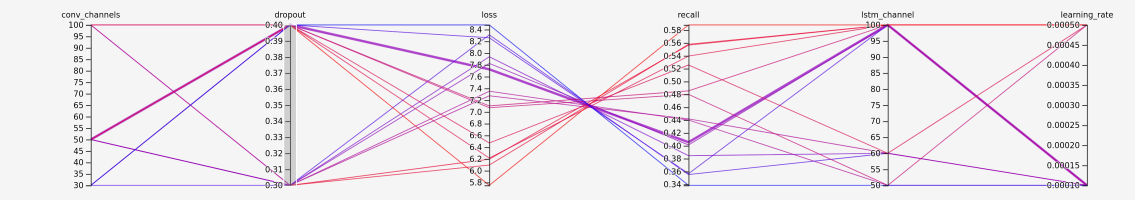


Abbildung 7.9: Hyperparametersuche

Parameter	Wert
sequence_length	1600
embed_size	8
conv_blocks	3
conv_channels	100
dropout_rate	0,5
feature_vektor_size	35
output_activation	sigmoid
conv_activation	swisch
Lernrate	0,0005

Tabelle 7.5: Finale Parameter-Auswahl Text-Netz

Lernrate von 0,0005 besser als 0,0001 und die *conv_ channels* und *lstm_ channel* zeigen die besten Ergebnisse bei einer Größe von 100. Höhere Dropoutraten generalisieren besser. Aufgrund dieser Erkenntnisse und weiteren Versuchen sowie der vorgegangen Überlegungen, werden die Parameterwerte in Tabelle 7.5 gewählt. Mit diesem Netz werden alle nachfolgenden Auswertungen gemacht.

Auswertung

Das gewählte Text-Netz mit den Parameterwerten aus Tabelle 7.5 wurde 200 Epochen lang trainiert. Der Validierungsverlust lag am Ende bei 5,35 was deutlich unter 11,07, dem Verlust der durchschnittlichen Verteilung, liegt. Daraus lässt sich ableiten, dass das Netz auf jeden Fall mehr als nur die Verteilung der Daten gelernt hat. In Tabelle 7.6 sind Ausschnitte der klassenspezifischen Genauigkeit- und Trefferquotenwerte für den Validierungssatz, den Testsatz sowie dessen Teilbereich der Umfrage um einen direkten Vergleich mit dem Goldstandard aus Tabelle 7.2 zu ermöglichen. Die vollständige Tabelle ist in Anhang A.3 abgebildet. Wie auch bei der Bild-Klassifzierung wird nach der statistischen Auswertung eine qualitative Analyse von zwei Texten mit einer Adaption der GradCam-Methode realisiert. Dies hilft weitere Einsicht und Vertrauen in die Entscheidung für bestimmte Klassen zu bekommen.

Auffällig in der Tabelle ist, dass die Kategorie *Fettarm* fast gar nicht erkannt wird, wobei das Netz *Low Carb* und *Ketogen* einigermaßen erkennt. Alle drei sind Konzepte, die die Abwesenheit von etwas aufzeigen, also kein Fett oder keine Kohlenhydrate und ähnlich abstrakt sind. Eine Ursache könnte sein, dass *Fettarm* nur halb so häufig vorkommt wie *Low Carb*, allerdings kommt *Ketogen* genauso häufig vor. Vermutlich ist *Fettarm* zu inkonsequent kategorisiert. Es gibt einige Gerichte die wenig Fett enthalten, wie Couscous-Salate, aber nicht als *Fettarm* kategorisiert sind.

Backen und *Kekse* sind die Kategorien, die das Text-Netz am besten gelernt hat, wobei die Schlüsselwörter „backen“ und „Backofen“ sowie auch Zeitangaben ein Indiz dafür bilden. Diese Werte sind beide besser als die in der Umfrage. Auffallend ist auch, dass die Kategorie *Frühling*, auf dem Umfragesatz besser ist als auf dem Validierungssatz, vor allem da das Bild-Netz diese Kategorie auf dem entsprechenden Bildsatz kaum erkannt hat. Vermutlich liegt es daran, dass für die Kategorisierung von *Frühling*, die Länge des Textes eine Rolle spielt, scheinbar sind kürzere Texte eher ein Indiz für diese Kategorie. Dies wird dadurch unterstützt, dass die Texte im Umfragesatz im Median 100 Zeichen, im Durchschnitt sogar 200 kürzer sind. Dadurch werden vermutlich alle Texte der Frühlingskategorie auch richtig klassifiziert. Auffällig ist auch, dass *Rind* auf allen Sätzen besser klassifiziert wird als *Schwein*, wobei die Kategorie *Schwein* fast doppelt so häufig im gesamten Datensatz vorkommt. Ein Indiz hierfür könnte sein, dass die Texte, die Rindfleisch enthalten, etwas länger sind als der Durchschnitt. In den vorliegenden Rezepten wird Rindfleisch häufig aufwendig zubereitet und die Rezepte enthalten auffällige Kräuter wie z. B. der Tyhmianzweig im Französischen Kalbsbraten Rezept.

Auffallend ist auch, dass *Winter* vom Netz eine etwas höhere Trefferquote aufweist als im Goldstandard, vor allem weil die Genauigkeit deutlich höher ist. Wahrscheinlich ist *Winter* ein größeres Konzept, dass Menschen intuitiv auch ein wenig unterschiedlich verstehen, da es nicht rein auf Essen begrenzt, sondern eher allgemeingültig ist. Das Netz lernt aber nur die Verbindung von Winter und Essen. Die *Studentenküche* hat dafür im Goldstandard eine sehr viel bessere Trefferquote im Vergleich zum Text-Netz, allerdings auch eine niedrigere Genauigkeit. Das ist auch wieder ein größeres Konzept, dass das Netz nicht gelernt hat, das allerdings viele der Teilnehmende an der Umfrage gut kennen, da ein Großteil der Befragten Studierende sind. Sehr interessant ist auch, der Vergleich des Netzes mit dem Goldstandard bei der Kategorie *Ketogen*. Die Teilnehmenden konnten *Ketogen* quasi fast gar nicht richtig kategorisieren, auf denselben Daten erreicht das Netz jedoch eine Trefferquote von 40% und einer Genauigkeit von 100%.

Insgesamt ist es überraschend, dass das Text-Netz auf dem Umfragedatensatz den Teilnehmern der Umfrage bezüglich der Kategorisierung der Texte überlegen ist. Mit einem gemittelten Trefferquote von 63,47 Prozent liegt dieser mit 10,76 Prozentpunkten über dem hier erstellten Goldstandard. Die Genauigkeit ist sogar noch

	Validierung		Test		Umfrage	
Kategorie	Treff.	Genau.	Treff.	Genau.	Treff.	Genau.
Backen	94,74	96,38	89,11	91,91	72,22	96,30
Braten	70,30	66,71	77,93	72,90	72,73	72,73
fettarm	**1,06**	100,00	0,00	0,00	0,00	0,00
Frühling	60,64	75,15	61,54	84,21	**100,00**	**100,00**
Hauptspeise	91,58	87,44	90,04	86,23	90,48	90,48
Kekse	82,91	88,51	83,33	85,11	**100,00**	100,00
ketogen	**19,86**	53,90	22,50	54,55	**40,00**	**100,00**
Low Carb	**25,60**	62,46	29,37	70,00	27,27	100,00
Rind	82,09	75,79	79,17	84,07	77,78	87,50
Schwein	72,77	62,32	65,79	66,23	50,00	57,14
Studentenküche	**2,45**	**52,63**	0,00	0,00	0,00	0,00
Weihnachten	49,05	61,26	37,88	62,50	55,56	83,33
Winter	28,27	59,04	32,50	56,52	**41,67**	**100,00**
Gemittelt	63,88	78,32	63,52	75,85	63,47	83,09

Tabelle 7.6: Auswertung des Text-Netzes zur Klassifizierung von Text Daten

höher mit 83,09% im Vergleich zu den 52,56%. Das kann eventuell daran liegen, dass Menschen ein intuitives, aus Erfahrung gelerntes Verständnis für gewisse Kategorien besitzen, die unabhängig vom Essen sind und diese nicht so gut übertragen können. Das Text-Netz hingegen, wird nicht von weiterem Konzeptionswissen abgelenkt, da es diese Kategorien nur im Bezug auf Essen gelernt hat.

GradCam ist eine Möglichkeit in Bildern klassenbezogene Aktivierungen zu zeigen um zu untersuchen, wovon die Entscheidung des Netzes für eine bestimmte Klasse abhängt. Selvaraju et al. sagen, dass die Methodik theoretisch auf alle Datentypen übertragen werden kann, wenn das zugrunde liegende Netz auf CNNs aufbaut (Selvaraju et al., 2020). Da das für dieses Text-Netz gilt, wird eine Adaption von GradCam auf zwei Beispiel Rezepttexte angewendet um eine bessere Einsicht auf die Entscheidungsgrundlage des Netzes zu bekommen. Je röter die Färbung hinter den Buchstaben ist, desto stärker ist die Aktivierung. Dabei muss beachtet werden, dass die Heatmap auf Basis der letzten Konvolutionsschicht berechnet wird, da diese Schicht den meisten semantischen Inhalt enthält und gleichzeitig noch lokale Strukturen beibehält. Dies bedeutet, dass diese Heatmap relativ grob ist und auf die originale Sequenzlänge hochskaliert wird. Deswegen kann die Lokalisierung etwas grob sein, lässt aber trotzdem wichtige Rückschlüsse auf die Indizien zu, die für die Entscheidung für eine bestimmte Kategorie stehen.

Französischer Kalbsbraten In Listing 2 ist die Heatmap der Kategorien *Hauptspeise* für den Text des Französischen Kalbsbraten Rezeptes abgebildet. *Hauptspeise*

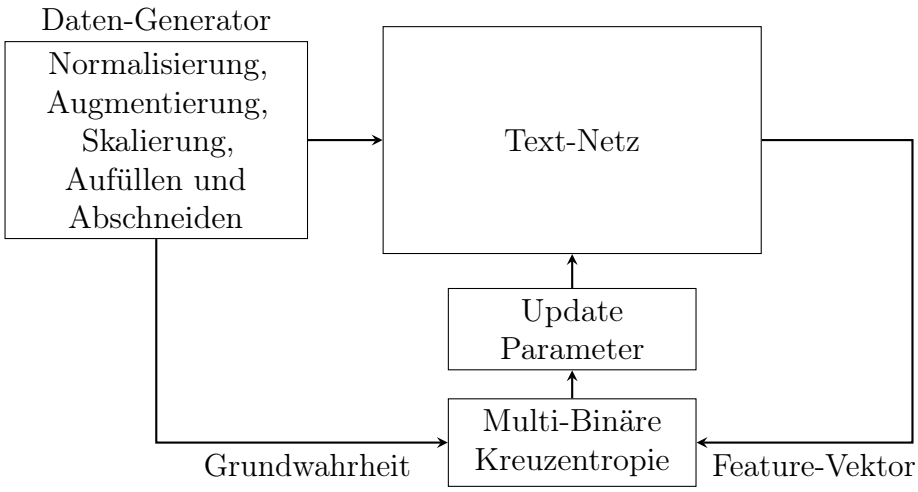


Abbildung 7.10: Schematisches Training der Textklassifizierung (eigene Darstellung)

weist die höchste Wahrscheinlichkeit mit 95% auf dem Text auf. Danach werden noch *Braten* mit 68% und *Rind* mit 50% als richtig klassifiziert. Die anderen Kategorien die in der Grundwahrheit enthalten sind: *Europa*, *Festlich*, *Herbst*, *Sommer* und *Gemüse* wurden vom Text-Netz nicht erkannt. *Gemüse* wird zwar insgesamt vom Text-Netz mit einer Trefferquote von 88% auf dem Validierungssatz gut erkannt, allerdings wird in diesem Rezept das Gemüse, die Bohnen, genau einmal erwähnt, das reicht vermutlich nicht aus. Dies wird auch durch die Umfrage bestätigt, in der keiner der Teilnehmenden diesem Rezepttext der Kategorie *Gemüse* zugeordnet hat. Auffällig ist, dass für die Kategorie *Hauptspeise* „Braten/braten“, „Zwiebeln“, „80 Minuten“, „Butter“ und „Kartoffeln“ die höchste Aktivierung aufweisen. Das bedeutet die Zeitangangabe scheint ein wichtiges Indiz zu sein, was *Hauptspeise* von anderen Kategorien unterscheidet, sowie die Kombination aus Butter, Kartoffeln und Zwiebeln. Der Kalbsbraten scheint ein typisches Hauptgericht bestehend aus Kartoffeln, Fleisch und ein bisschen Gemüse zu sein, zumindest haben alle Teilnehmenden *Hauptspeise* zu diesem Rezept zugeordnet.

Auffällig ist auch die Aktivierung auf jedem Vorkommen des Wortes oder Teilwortes „Braten“, die Aktivierung ist mehr oder weniger stark, hebt sich aber immer ab. Das liegt vermutlich daran, dass in 80% der Fälle im Training, ein Braten auch eine Hauptspeise ist und das Wort „braten“ häufig ein Indiz für *Hauptspeise* ist. Darüber hinaus, weist auch das Wort „Zweig“ eine mittlere Aktivierung auf, was unter anderem dazu passt, dass in Abbildung 7.6 a) auch eine leichte Aktivierung für diese Kategorie auf dem Dekorationszweig in der linken oberen Ecke liegt.

Erdbeer Schoko Quark Die Aktivierung von *Dessert* des Erdbeer Schoko Quarks ist in Listing 3 abgebildet. Der Kategorie *Dessert* wird auf diesem Text mit knapp

Das Fleisch salzen, pfeffern und zu einem Braten zusammenschnüren dabei je 1 Zweig Thymian und Salbei dazwischen packen. In einem Bratentopf im heißen Öl sanft und mit etwas Geduld auf allen Seiten anbraten. Die Knochen und die geschälten Zwiebeln darum herum streuen und zugedeckt etwa 70 - 80 Minuten auf kleinem Feuer sachte schmoren lassen. Dabei immer wieder drehen und jedes Mal einen Schuss Fond und etwas Chablis zugießen, damit nichts ansetzt. Den Braten in Scheiben schneiden und auf einer großen Platte anrichten. Die Sauce durchseihen und abschmecken. Die Zwiebeln mit zarten Böhnchen mischen, die nur kurz blanchiert und in etwas Butter geschwenkt sind. Für die Röstkartoffeln möglichst gleich große, kleine Kartöffelchen aussuchen, die gar gekocht und in Olivenöl rundum braun gebraten werden - dabei dürfen ruhig einige Salbei-, Thymian- oder Rosmarinzweige mitbraten und ihren Duft abgeben.

Listing 2: Heatmap auf dem Rezept des Französichen Kalbsbraten (Hauptspeise)

97% die höchste Wahrscheinlichkeit zugeordnet. In diesem Beispiel ist die stärkste Aktivierung auf dem Anfang des „(Püree) in Gläser“, was die Vermutung bei den Bildern unterstützt, dass Glas ein wichtiges Indiz für *Dessert* ist. Des Weiteren, ist auch „Beeren putzen (und) pürieren“ und „Quark“ ein Indiz, was Sinn ergibt, da diese Zutaten häufig in Nachspeisen der vorliegenden Rezepte vorkommen.

In Listing 4 ist die Heatmap der Kategorie *Frucht* abgebildet, die mit einer Wahrscheinlichkeit von 54% als richtig klassifiziert wird. Hierbei bilden die „Erdbeeren“ ein starkes Indiz für die Kategorie sowie die Angabe „pürieren“ und das „Püree in Gläser“. Dennoch ist die Wahrscheinlichkeit mit 54% nicht sehr hoch, was vermutlich aber auch daran liegt, dass *Frucht* in der Grundwahrheit nicht mit aufgeführt wurde, es dementsprechend beim Training als falsch interpretiert wird und das Netz somit lernt diese Indizien nicht hoch zu bewerten. Dies verdeutlicht sehr gut, dass eine Konsistenz der Kategorien auf dem Trainingssatz wichtig ist, um ein solches Konzept lernen zu können. Zusätzlich fällt auf, dass in der Heatmap für die Kategorie *Frucht* sehr viel mehr Aktivität zu sehen ist, also in der Heatmap für *Dessert*. Das liegt vermutlich daran, dass das Netz sich nur ungefähr halb so sicher ist, dass *Frucht* eine richtige Kategorie für diesen Text ist, im Vergleich zum *Dessert* und deshalb viele Indizien in Erwägung gezogen werden.

Die Erdbeeren putzen und pürieren. Die Schokolade mit dem Puderzucker in der Milch schmelzen und abkühlen lassen und unter den Quark ziehen. Die Sahne steif schlagen und ebenfalls unterheben. Die Creme und das Püree in Gläser schichten und noch 2 Stunden kühl stellen.

Listing 3: Heatmap auf dem Rezepttext des Erdbeere Schoko Quarks (Dessert).

Die Erdbeeren putzen und pürieren. Die Schokolade mit dem Puderzucker in der Milch schmelzen und abkühlen lassen und unter den Quark ziehen. Die Sahne steif schlagen und ebenfalls unterheben. Die Creme und das Püree in Gläser schichten und noch 2 Stunden kühl stellen.

Listing 4: Heatmap auf dem Rezepttext des Erdbeere Schoko Quarks (Frucht).

7.2.4 Diskussion zur Klassifizierung

Insgesamt lässt sich in Bezug auf die statistische Auswertung feststellen, dass größere Konzepte wie die Jahreszeiten oder auch Frühstück, Europa und Studentenküche die Studienteilnehmenden einen wesentlich höhere Trefferquote erreichen als die jeweiligen Netze. Dies liegt vermutlich daran, dass es umfassendere Konzepte sind, die Menschen wesentlich besser kennen und daher besser einzuschätzen können. Allerdings lässt dieses Wissen sie scheinbar häufiger raten, da nicht immer ganz definiert ist, wie sich diese Konzepte jetzt genau auf das Essen beziehen. Denn wenn ein Netz ein solches Konzept gelernt hat, wie z. B. das Text-Netz den Winter (mit ca. 8% höheren Trefferquote als im Goldstandard) dann ist auch die Genauigkeit sehr viel höher. Die konkreten Konzepte wie beispielsweise Gemüse, Kuchen, Hauptspeise werden gut von beiden Netzen erkannt. Geflügel, Rind oder Schwein werden vom Text-Netz sehr viel besser erkannt als vom Bild-Netz, was durchaus Sinn ergibt, da Fleisch in einer Soße optisch sehr ähnlich ist.

Die Analyse einzelner Beispiele mit der GradCam Methode zeigen, dass die Netze für die Kategorien die eine Wahrscheinlichkeit von mindestens 50% aufweisen, erklärbare und logische Indizien zugrunde liegen. Auffällig ist an diesen gewichteten Aktivierungskarten auch, dass die Entscheidung als Eingabe die einzelnen Zeichen anstelle von Wörtern zu verwenden funktionieren kann, da beispielsweise die Zeichen „ffel “ oder „Flei“ ausreichen, um das Wort „Kartoffel “ oder „Fleisch“ zu identifizieren.

Die genauere Analyse der Beispiele hat zudem deutlich gemacht, dass einige Kategorien inkonsequent zugewiesen wurden, und das dies direkten Einfluss auf die Erkennung

dieser Kategorien hat. Ein weiterer Grund für die mittelmäßigen Ergebnisse in der Klassifizierung ist vermutlich die Verwendung der Multi-Binären Kreuzentropie aus Abschnitt 2.3.1, die unter der Anzahl negativer Kategorien einen falschen Bias bekommen kann (Wu et al., 2020). Insgesamt ist jedes Rezept durchschnittlich 7,35 Kategorien von 35 Kategorien zugeordnet, das heißt es sind fast fünf mal mehr negative Kategorien als positive.

Trotz dieser Nachteile, haben die Netze beide mehr als nur die Verteilung gelernt, was dadurch belegbar ist, dass beide ein niedrigeren Verlust aufweisen als sich durch eine durchschnittliche Verteilung ergeben würde. Des Weiteren wurde der Goldstandard aufgesetzt um einen erstrebenswerten Vergleichswert für die Trefferquote und die Genauigkeit zu erhalten. Dabei hat sich ergeben, dass das Text-Netz eine um 10,76 Prozentpunkte bessere Trefferquote erreicht als der Goldstandard. Das Bild-Netz liegt zwar ein wenig unter der Trefferquote vom Goldstandard, weist dafür aber eine höhere Genauigkeit auf. Zusammenfassend ist zu erkennen, dass beide Netze quantitativ in der Nähe der Werte des Goldstandards sind und qualitativ sinnvolle Indizien für eine Kategorie mit einer hohen Wahrscheinlichkeit finden. Daraus lässt sich ableiten, dass sowohl das Text-Netz als auch das Bild-Netz geeignet sind um Informationen aus dem Datensatz zu extrahieren.

7.3 Verknüpfung von Text- und Bildnetz und Contrastive Loss

Als abschließender Versuch werden die vorhergegangenen Experimente vereint und das Gesamtsystem mit einem realen Datensatz getestet. Hierzu wird der Aufbau von 4.3 verwendet. Als Text-Netz wird das in Abschnitt 7.2.3 beschriebene Netz verwendet, mit Ausnahme der letzten Schicht. Die Vorverarbeitung des Textes, sowie dessen Augmentation, findet wie in dem Abschnitt beschrieben statt. Als Bild-Netz wird das in Abschnitt 7.2.2 verwendete Netz genutzt. Auch hier bleibt die Vorverarbeitung und künstliche Augmentation genau wie dem Abschnitt beschrieben. Nur die letzte Schicht wird angepasst, denn die Ausgabeschichten des Bild- und Text-Netzes müssen aufeinander abgestimmt sein, damit eine gemeinsame Verlustberechnung möglich ist. Dies ist zwar auch gegeben, wenn beide einen Feature-Vektor der Länge 35 berechnen, allerdings ist die Länge von 35 nur aufgrund der Anzahl an betrachteten Kategorien festgelegt worden. Die Bindung an Kategorien ist hier nicht mehr notwendig. Dies ist ein weiterer Vorteil dieser Lernmethode, da so die Länge des Feature-Vektors parametrisierbar gemacht und deren Einfluss auf das Gesamtsystem betrachtet werden kann. Des Weiteren wird die Aktivierung der Ausgabeschichten verändert, da die Ausgabewerte nun nicht mehr als Wahrscheinlichkeit interpretiert werden,

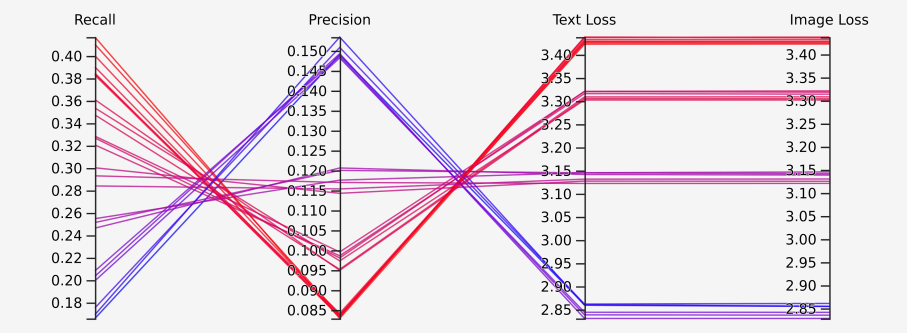


Abbildung 7.11: Zusammenhang von Genauigkeit und Trefferquote im Kontrastive Loss

sondern einen numerischen Vektor in einem n-dimensionalen Raum darstellen. Dieser kann auch negative Werte aufweisen.

Als erstes werden die Trainingsparameter beschrieben und festgelegt. Danach erfolgt eine statistische und qualitative Auswertung. Für die statistische Auswertung sowie bei der Betrachtung der Trainingsparameter werden auch die Trefferquote und Genauigkeit verwendet. In diesem Experiment wird allerdings ein Schwellenwert von 0,95 angelegt wie in Abschnitt 7.1, da es anders als in der Klassifizierung nicht darum geht eine ja/nein Entscheidung bezüglich einer Klasse zu treffen, sondern um die Bewertung von Ähnlichkeiten.

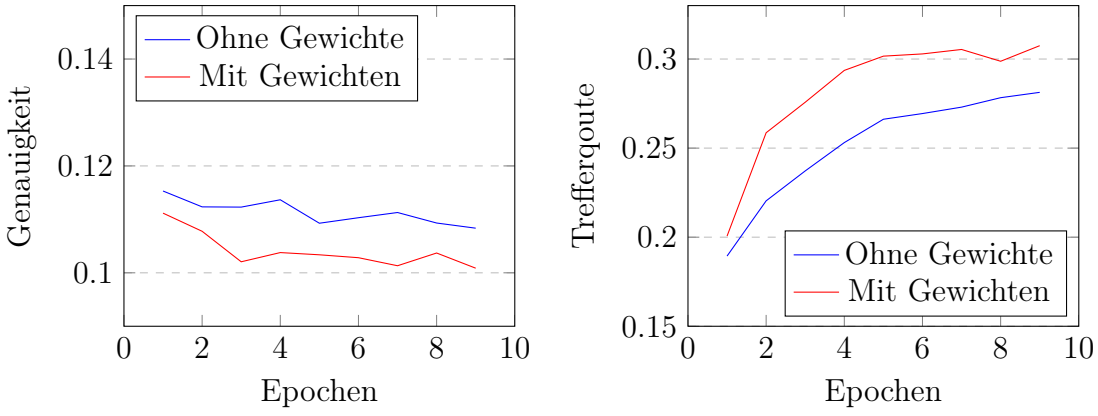
7.3.1 Trainingparameter

Für das Training wurden auch hier unterschiedliche Parameter getestet und der Einfluss verschiedener Werte betrachtet. In dieser Abbildung 7.11 lässt sich sehr gut der Zusammenhang zwischen Genauigkeit, Trefferquote und Verlust sehen, der mit dieser Art von Verlustfunktion verbunden ist. Die Aufteilung der Farben ist auch hier, je besser die Trefferquote ist, umso röter ist die Linie. Je höher die Trefferquote ist, umso niedriger ist hier die Genauigkeit und je höher der Verlust. Dies lässt sich dadurch erklären, dass die Trefferquote größer wird je häufiger eine Kosinus-Ähnlichkeit von über 95% vorliegt. Wenn das Netz aber eine hohe Ähnlichkeit anstrebt, ist die Wahrscheinlichkeit sehr hoch, dass auch andere Paare eine hohe Kosinus-Ähnlichkeit bekommen. Dadurch sinkt die Genauigkeit. Dieser Zusammenhang kann durch den Parameter τ kontrolliert werden, bei der niedrigsten Trefferquote (0,18) ist $\tau = 0{,}4$ bei der höchsten ist $\tau = 1$. Da es hier in dieser Problemstellung durchaus legitim ist, wenn andere Bilder auch passen, wie ein weiteres Bild von einem Schokoladenkuchen, ist eine sehr hohe Genauigkeit nicht nötig. Dennoch sollte ein gutes Gleichgewicht gefunden werden, deswegen wird $\tau = 0{,}7$ gewählt.

Parameter	Wert
feature_vektor_size	80
output_activation	tanh
τ	0,7

Tabelle 7.7: Finale Parameterwerte für das Text-Bild Problem

Des Weiteren wurde der Einfluss der vorgeschlagenen Gewichte überprüft. In Abbildung 7.12 sind die ersten zehn Epochen abgebildet, auf denen zwei identische Netze trainiert wurden. Der einzige Unterschied ist die Verwendung der Gewichte. Mit Gewichten erreicht das Netz tatsächlich schneller eine höhere Trefferquote von etwas über 2%. Dafür ist die Genauigkeit etwas geringerer, wobei das weniger als 1% ausmacht. Dies ist allerdings logisch, da ohne Gewichte alle negativ Paare gleich behandelt werden, was in einen höheren Verlust bedeutet und die Beispiele weiter auseinander drängt. Aber wie eben schon genannt, sollte es zwar ein Gleichgewicht zwischen Genauigkeit und Trefferquote geben, aber der Gewinn von 2% Prozent bei der Trefferquote wird hier höher gewertet, als eine minimal erhöhte Genauigkeit. Außerdem hat sich herausgestellt, dass als Aktivierung der letzten beiden Schichten *tanh* gegenüber *linear* besser eignet. Die Feature-Vektor Länge wird auf 80 festgesetzt. Insgesamt haben sich die Parameter in Tabelle 7.7 am besten erwiesen.



(a) Einfluss auf die Genauigkeit

(b) Einfluss auf die Trefferquote

Abbildung 7.12: Vergleich von der Verwendung von Gewichten gegenüber keiner Gewichte

7.3.2 Statistische und Qualitative Auswertung

Die beiden Netze werden parallel in drei Schritten mit einer Batchsize von 100 trainiert. Im ersten Schritt werden fünf Epochen lang nur die Ausgabeschicht der Netze mit einer Lernrate von 0,001 mit dem Adam Optimierer trainiert. Der Rest der Gewichte wird vorerst festgehalten. In einem zweiten Schritt werden alle Schichten nach den Konvolutionsschichten trainiert, diese sind weiterhin fixiert. Für diesen Schritt wird die Lernrate um einen Faktor von zehn reduziert und weitere sechs Epochen trainiert. Anschließend werden bei dem Text-Netz alle Schichten freigegeben und bei dem Bild-Netz ab *block_9_add* und für weitere 27 Epochen, mit einer um den Faktor zehn verringerten Lernrate (0,00001). Alle weiteren Auswertungen basieren auf Ergebnissen dieser beiden Netze.

Als Erstes werden die statistischen Werte betrachtet. Da diese allerdings nur über die Originalpaare erhoben werden können, von denen bekannt ist, dass sie zueinander gehören. Es ist aber auch möglich, dass die Bilder, die eine sehr hohe Ähnlichkeit aufweisen, sehr passend sind. Das kann jedoch durch die reine Statistik nicht abgebildet oder überprüft werden, da hierzu keine weiteren Informationen oder bekannten Grundwahrheiten existieren. Deswegen wird eine zweite Umfrage erstellt. Am Ende wird, wie bei der Klassifzierung, auf zwei Textanfragen beispielhaft eingegangen und die zehn Bilder, die der Berechnung der Netze zufolge am ähnlichsten sind, vorgestellt und diskutiert.

Statistische Auswertung

Das Training der beiden Netze erreicht eine Trefferquote von 41,43% und eine Genauigkeit von 10,24% auf dem Validierungssatz und eine Trefferquote von 38,99% und Genauigkeit von 11,3% auf dem Umfragesatz. Die Diskrepanz zwischen den beiden Werten lässt ist durch die Art der Bilder erklären. Der Trainings- und Validierungssatz bestehen hauptsächlich aus Bildern die von Nutzern der Plattform bereitgestellt wurden. Während der Test- und Umfragesatz aus professionellen Bildern besteht, die das Essen meistens sehr kunstvoll in Szene setzen. Die Genauigkeit in beiden Fällen, hängt allerdings stark von der gewählten Batchgröße ab, da diese die Anzahl an negativen Paaren bestimmt, die wiederum Auswirkung auf die Genauigkeit haben. Wenn beispielsweise die Batchgröße von 100 auf zehn reduziert wird, steigt die Genauigkeit um das fünffache.

Eine Trefferquote von 41,43% sagt aus, dass in ca. 41% der Fälle, das zugehörige Bild eine Ähnlichkeit von über 95% aufweist. Allerdings ist jedoch nicht nur die Ähnlichkeit der korrekten Paare wichtig zu betrachten, sondern, es ist ebenfalls relevant, ob das Originalbild unter den zehn ähnlichsten Bildern ist. Auf dem Validierungssatz ist das bei 4% der Texte der Fall, beim Testsatz sind es sogar 16%. Da diese Werte dennoch

weit unter den jeweiligen Trefferquoten liegen, bestätigt dies die nicht allzu hohe Genauigkeit, da dies bedeutet, dass in 96% (bzw. 84%) der Fälle andere Bilder eine höhere Ähnlichkeit zu dem Text aufweisen. Allerdings lässt sich z. B. auf dem Bild des Französischen Kalbsbratens in Abbildung 6.2 h) erkennen, dass das Originalbild nicht immer dem Text entspricht. Es sind zwar das Fleisch und die Kartoffeln, aber nicht die Bohnen, also das zugehörige Gemüse abgebildet. Außerdem ist es realistisch, dass bei einem Bilderpool von 8.000 Bildern im Validierungssatz, durchaus mehrere passende Bratenbilder gibt, die vielleicht sogar besser oder gleich gut passen wie das Originalbild. Um dies zu überprüfen und eine Bewertung über eine reine Trefferquote hinaus zu erhalten wird die Folgende zweite Umfrage durchgeführt.

Umfrage zur Qualitativen Bewertung

In dieser Umfrage, abgebildet in Anhang A.2, wird jeweils ein Rezepttext mit jeweils elf Bildern vorgestellt. Zehn dieser Bilder sind die zehn ähnlichsten Bilder, die sich bei der Ähnlichkeitsabfrage mit der Datenbank ergeben. Das Elfte, ist das Bild welches originalerweise zu dem Text gehört. Dies wird dem Teilnehmenden jedoch nicht erklärt um eine Befangenheit in irgendeine Richtung zu vermeiden. Des Weiteren werden die Bilder in unterschiedlicher Reihenfolge angezeigt um Co-Abhängigkeiten in der Reihenfolge und der Bewertung zu verhindern. Dem Teilnehmenden werden aus einem Pool von 100 Rezepten, die zufällig aus dem Umfragepool ausgewählt sind, genau fünf zufällig ausgewählte Rezepte inklusive Bildern angezeigt. Die Bilder werden von den Teilnehmenden auf einer Skala von 1-5 bewertet, je nachdem wie gut sie zu dem präsentierten Rezepttext passen, wobei 1 bedeutet, dass es gar nicht passt und 5 dass es sehr gut passt. Der Vorteil einer solchen Zahlenskala, ist dass es als Intervallskalenniveau interpretiert und ausgewertet werden kann. Die Aufteilung in fünf Stufen ist gängig und erlaubt es dem Teilnehmenden zu differenzieren und eine ambivalente Meinung einzunehmen, ohne sich für eine Seite entscheiden zu müssen (Walter Hussy, 2013). Zusätzlich besteht die Option keine Antwort auszuwählen.

Als Teilnehmende wurden Studierende über ein Nachrichtenportal der Universität sowie weitere Bekannte angefragt. Es wurden jedoch, wie in der Umfrage in Abschnitt 7.2.1 keine persönlichen Daten von Teilnehmenden erhoben und auf die Freiwilligkeit der Umfrage hingewiesen. Die befragte Gruppe ist in diesem Fall für eine solche Bewertung geeignet, da sie potenzielle Konsumenten dieser Rezepte sind, denen das Bild gefallen sollte. Die Umfrage wurde insgesamt 48 mal ausgefüllt, im Durchschnitt wurde so jedes Rezept 2,4 mal bewertet.

Wenn die Anzahl der Bilder betrachtet wird, sind ca. 62% der Bilder als eher nicht passend (2) oder schlechter bewertet und ca. 20% als passt eher gut (4) oder besser bewertet, nur ca. 1% werden mit „keine Antwort“ bewertet. Dadurch ergibt sich eine gesamtdurchschnittliche Bewertung von 2,06. Das bedeutet die Bilder werden generell

eher als unpassend eingestuft. Auf den Originalbildern ergibt sich im Gegensatz dazu eine gesamtdurchschnittliche Bewertung von 4,61. Die Originalbilder werden als eher passend (4) bis sehr passend (5) eingestuft. Ziel des konzipierten Systems ist es jedoch, möglichst passende Bilder vorzuschlagen. Für den betrachteten Fall ist es also nicht relevant wenn auch unpassende Bilder dabei sind. Allerdings ziehen Bilder mit einer Bewertung als gar nicht passend (1) oder eher nicht passend (2), den Durchschnitt stark nach unten. Aus diesem Grund wird zusätzlich untersucht, ob zu jedem Rezept überhaupt ein Bild gefunden wird, das als eher passend (4) oder besser eingestuft wird. Dafür wird die maximale Bewertung die eines der zehn Bilder für ein Rezept bekommt betrachtet. In 54% der Rezepte wird mindestens ein Bild als sehr passend (5) und in knapp 28% der Rezepte als eher passend (4) bewertet. Nur ca. 9% der Rezepte haben kein Bild, das als passend eingestuft wird. Des Weiteren wird die maximale Bewertung der Vorschläge und des Originals verglichen. Bei ungefähr 70% der Rezepten hat eins der vorgeschlagenen Bilder die gleiche oder eine bessere Bewertung als das Originalbild.

Qualitative Analyse an zwei Beispielen

Zur qualitativen Analyse des Verfahrens und den daraus resultierenden Bildvorschlägen werden hier zwei Textanfragen und deren Ergebnis exemplarisch vorgestellt. Zum einen der Französische Kalbsbraten als Hauptgericht und der Erdbeer Schoko Quark als Dessert. Mithilfe des Text-Netzes wird für den gegebenen Text ein Feature-Vektor berechnet und dieser mit der Bilderdatenbank abgeglichen. Die Bilderdatenbank besteht in diesem Fall nur aus den Bildern des Testsatzes, also ca. 3% des gesamten Datensatzes und wird mit dem vorgestellten Bild-Netz berechnet. Die zehn ähnlichsten Bilder, bestimmt durch die Kosinus-Ähnlichkeit ihrer Feature-Vektoren, werden zurück gegeben. Die berechnete Ähnlichkeit, die im Intervall von $[-1,1]$ liegt mit 1 = ähnlich und -1 = unähnlich, steht immer in der jeweiligen Bildunterschrift.

Französischer Kalbsbraten: Die zehn ähnlichsten Bilder zum Rezepttext des Französichen Kalbsbratens aus Listing 1 oder Listing 2 sind in Abbildung 7.13 dargestellt. Auffällig ist, dass es sich bei fast jedem Bild um ein Fleischgericht handelt, das eine Hauptspeise ist. Die Ausnahme bietet Bild c) da dies eine gebackene Süßkartoffel mit gerösteten Wallnüssen ist. Dennoch erfüllt sie die Indizien die die Netze in ihrer Klassifizierungsphase gelernt haben. Die Kartoffel ist von außen sehr dunkel und in fast durchgehende Scheiben geschnitten und die Walnüsse könnten auch etwas angebrannte Kartoffelstücke darstellen. Darüber hinaus, ist oben links in der Ecke etwas grünes, was in diesem Fall eher zur Dekoration gehört, das Netz aber nicht unterscheiden kann wie auch in Bild 7.6 c) ersichtlich ist.

Darüber hinaus ist erkennbar, dass die Hälfte der Bilder Kartoffelbeilagen aufweisen, wie beispielsweise die Salzkartoffeln in a) aber auch der Kartoffelkloß in f) und die

Kartoffelstücke in der Gulaschsuppe in h). Die Bilder b), e), und j) auf der anderen Seite, präsentieren die klare, typische Schnittkante die entsteht wenn von einem Braten ein Stück abgeschnitten wird. In den Bildern a), g) und h) handelt es sich auch nicht um Bratenstücke, aber zusätzlich zu den Kartoffeln weisen sowohl a) als auch h) etwas grünes auf. In g) allerdings lässt sich nur die typische Farbe von Fleisch wiederfinden, dunkel mit ein bisschen frischerem rot. Bei dieser Abfrage wurde das Originalbild wieder gefunden und ist in i) abgebildet.

Insgesamt ist interessant, dass Bild a) die höchste Ähnlichkeit mit dem Text aufweist, obwohl es eigentlich ein Pilz-Fleisch Gulasch mit Salzkartoffeln abbildet. Vermutlich liegt dies an dem Gesamtbild, das Essen ist auf einem runden Teller angerichtet, der in d) beispielsweise nur erahnt werden kann. Es gibt eine Kartoffelbeilage und etwas grünes das dem Rosmarin oder vielleicht auch aufgrund der Bohnen wichtig sein kann. Außerdem hat das Fleisch auch eine typische Fleischfarbe. Es kombiniert also alle Aspekte des Rezepttextes.

Erdbeer Schoko Quark: In Abbildung 7.14 sind die zehn ähnlichsten Bilder zum Erdbeer Schoko Quark abgebildet. Das Originalbild ist nicht dabei, weist aber eine Ähnlichkeit von 96,7% auf. Auffällig ist, dass jedes Dessert in einem Glas oder einer Glasform abgebildet ist, bis auf a) sind sogar alle in Portionsgrößen, also ein Glas pro Portion wie es in dem Rezept vorgeschlagen wird. Dies kommt vermutlich daher, dass sowohl im Text-Netz als auch im Bild-Netz während der Klassifizierung das Wort Glas und der Rand des Glases, starke Indizien waren. Allerdings ist auch auffällig, dass in nur zwei der Bilder, genauer gesagt a) und c) als Früchte wirklich Erdbeeren abgebildet sind, bei den anderen handelt es sich vermutlich um Himbeeren, in j) sind es Weintrauben. Der Unterschied zwischen Himbeeren und Erdbeeren, scheint schwierig zu sein, vermutlich weil die Unterscheidung in beiden Netzen schwierig ist. Bezüglich des Bildes sind die Beeren in der Farbe fast nicht zu unterscheiden und bezüglich des Textes ist in Erdbeere wie auch in Himbeere das Teilwort „eere“ enthalten. In der Heatmap der Text-Klassifizierung in Listing 3, war das Teilwort ein Indiz.

Die Bilder c) und j) passen am wenigsten zu der Beschreibung, da es sich in c) um eine Erdbeer Bowle handelt, also ein Getränk und kein Dessert und in j) grüne Früchte anstelle von roten verwendet werden. Die grüne Färbung kann durch die Tatsache dass Erdbeeren und Himbeeren häufig mit Minze, also etwas grünem, präsentiert werden, wie das in Bild c), d) und i) gezeigt wird. Optisch ist das Bild der Erdbeer Bowle abgesehen davon dass die untere Schicht aus einer Flüssigkeit und nicht aus Sahne besteht, sehr ähnlich zu den anderen. Insgesamt sind zwar Fotos von Bowlen und Cocktails in dem Datensatz vorhanden, allerdings im Vergleich zu Essen sehr wenige und in der Klassifzierungsaufgabe gab es auch keine Kategorie Cocktail oder Getränk mithilfe derer das Netz das Konzept von flüssig und fest hätte lernen können. Deswegen ist es nicht verwunderlich, dass das Bild hier vorgeschlagen wird.

(a) 0,9921

(b) 0,9891

(c) 0,9870

(d) 0,9841

(e) 0,9837

(f) 0,9833

(g) 0,9817

(h) 0,9811

(i) 0,9735

(j) 0,9728

Abbildung 7.13: Die 10 besten Bilder zum Französischen Kalbsbraten Rezept

Unverkennbar ist, dass in allen Bildern die Zutaten geschichtet sind. Sahne oder Quark wechselt sich mit Früchten ab, auf den Bildern b), i) und j) ist auch Schokolade zu sehen. In der Heatmap des Textes ist zu sehen, dass klassenübergreifend jeweils Teile des Wortes „schichten“ leicht aktiviert sind und auch die Heatmaps auf den

Fotos zeigen entlang der Schichten eine erhöhte Aktivierung. Schichten scheinen somit eine wichtige Information zu sein, die in allen Bildern wieder gefunden wird.

(a) 0,9946

(b) 0,9939

(c) 0,9917

(d) 0,9905

(e) 0,9904

(f) 0,9901

(g) 0,9894

(h) 0,9886

(i) 0,9883

(j) 0,9882

Abbildung 7.14: Die 10 besten Bilder zum Erdbeer Schoko Quark Rezept

7.3.3 Diskussion

Insgesamt funktioniert die Unterteilung in große Kategorien sehr gut. Für Rezepte die Fleisch enthalten, werden Bilder mit Fleischgerichten vorgeschlagen, bei Suppen, werden Suppenbilder und bei Desserts werden meist Creme-Desserts oder Kuchen vorgeschlagen. Auch für Aufläufe und Lasagnen sowie Salate werden passende Vorschläge generiert. Hierbei hat die Verwendung der auf den Kategorien vortrainierten Netze unterstützt. Allerdings kommt es auch vor, dass bei einem Geflügelgericht ein Bild mit rotem Fleisch, also Rind oder Schwein, dabei ist oder anstelle eines cremigen Kartoffelsalates, ein Matjessalat präsentiert wird. Dies erklärt unter anderem den durchschnittlichen Wert von 2 also eher nicht passend in der Umfrage, da die Teilnehmenden vermutlich eine ziemlich genaue Vorstellung davon haben, wie Hähnchenschenkel oder Kartoffeln aussehen müssen. Andere Gerichte werden trotz ähnlicher Optik, sehr schnell als unpassend empfunden. Dies ist auch in dem Erdbeer Schoko Quark Vorschlag in Abbilung 7.14 c) zu sehen, wo eine Bowle vorgeschlagen wird die für einen Menschen offensichtlicherweise nicht zu einem Quark-Dessert passt, dennoch von der reinen Optik her sehr ähnlich zu den anderen Bildern ist. Um Details wie verschiedene Beeren erkennen zu können, müsste die Auflösung der verarbeiteten Bilder vermutlich erhöht werden.

Auffällig ist weiterhin, dass das Text-Netz in der Relevanz der gefundenen Indizien nicht unterscheidet. Bei Dessert Rezepten tauchen häufiger Getränke auf, die in Gläsern abgebildet sind, wie zum Beispiel die Erdbeerbowle. Andersherum tauchen bei Getränken Desserts auf. Dies beinhalten häufig Dekoration wie z. B. Schokoladenstäbchen, was darauf schließen lässt, dass Strohhalme oder vielmehr aufgerichtete Stäbe, wiederum ein starkes Indiz für Getränke darstellen. Andererseits ist eine solche Gewichtung der Relevanz der gefundenen Indizien gar nicht unbedingt erwünscht, da so der Bildredakteur die Wahl hat, ob es wichtiger ist, dass Erdbeeren auf dem Bild zu sehen sind, dann könnte er beispielsweise Bild 7.14 a) nehmen, oder ob es wichtiger ist, dass eine Portionsgröße des Nachtisches in kleinem Glässchen mit Dekoration abgebildet wird. Dann kann dieses Bild 7.14 b) verwendet werden.

Zusammengefasst sind die Vorschläge die sich aus den Ähnlichkeiten von Bild und Text ergeben, sinnvoll erklärbar mit dem Wissen, dass dem Netz durch die Trainingsdaten zur Verfügung gestellt wird. Die Differenzierung ist teilweise nicht optimal, wie beispielsweise bei Hähnchenschenkeln und Braten, aber es geht darum, mindestens ein passendes Bild zu finden. Durch die Umfrage konnte bestätigt werden, dass dies für den Großteil der Rezepte (82%) funktioniert. Die Teilnehmenden bewerteten mindestens eines der Bilder mit einer 4 oder 5. In den Fällen ist mindestens ein Bildervorschlag passend. Zusätzlich wird dies durch die beiden gezeigten Beispiele unterstützt.

8 Fazit und Ausblick

In diesem Kapitel werden zunächst die Ergebnisse der Arbeit zusammengefasst und dadurch die Forschungsfrage beantwortet: Wie kann ein System aussehen, das einen Bildredakteur bei der Aufgabe archivierte Bilder aus hauseigenen oder internen unorganisierten Datenbanken wiederzuverwenden unterstützt? Abschließend wird ein Ausblick auf eine Weiterentwicklung und mögliche Verbesserungen des Systems gegeben.

8.1 Zusammenfassung

Diese Arbeit lässt sich gut anhand der in Kapitel 1 definierten Forschungsfragen und der herausgearbeiteten Beantwortung zusammenfassen.

1. Was bedeutet Unterstützung in dieser Arbeit?

In Kapitel 3 wurde zunächst herausgearbeitet, dass Unterstützung in diesem Kontext folgendes bedeutet: Es sollen vom System gewisse Teilmengen an adäquaten Bildern aus einer Gesamtmenge, hier aus einer Datenbank, herausgesucht und dem Bildredakteur vorgeschlagen werden. Diese Unterstützung ermöglicht dem Bildredakteur eine effizientere Suche nach einem passenden Bild zu einem Text.

2. Welche Anforderungen an das System ergeben sich daraus?

Im darauffolgenden Kapitel wurden die Anforderungen spezifiziert, die notwendig sind, damit auf Basis deutscher Textanfragen, farbige Bilder von dem System in realer Zeit abgerufen werden können. Neben der Verfügbarkeit für deutsche Texte und der direkten Adressierung der Problemstellung soll das System eine sehr gute Skalierbarkeit aufweisen. Darüber hinaus wird ein quantifizierbares Ähnlichkeitsmaß benötigt, dass die Bildervorschläge in eine Rangfolge bringen kann.

3. Wie kann ein solches System konkret aussehen?

Die Verarbeitung des rohen Bildmaterials und der natürlichen Texte wird durch die Verwendung von zwei separaten Neuronalen Netzen realisiert, wie in Kapitel 4 beschrieben. Diese extrahieren die Informationen und projizieren sie in einen geteilten neutralen Raum. Hierfür werden die beiden Netze parallel und in Abhängigkeit voneinander trainiert. Innerhalb dieses geteilten Raumes können Text und Bild aufgrund

ihres Informationsgehaltes in Form von Feature-Vektoren auf eine quantifizierbare Ähnlichkeit untersucht werden. Durch diese Quantifizierung ergibt sich automatisch eine Rangfolge, sodass die ähnlichsten Bilder dem Nutzer empfohlen werden können.

4. Wie müsste ein Datensatz aussehen, der geeignet wäre ein solches System zu trainieren?

In Kapitel 6 wurde der verwendete Datensatz vorgestellt und auf die Eignung überprüft. Der Chefkoch-Datensatz besteht aus deutschen Text-Bild Paaren, die genau die spätere Problemstellung abbilden. Zusätzlich ist die Skalierbarkeit bezüglich der Erweiterung des Trainingsatzes gegeben. Dadurch, dass diese Daten schon unabhängig von dem konzipierten System existieren, müssen sie für das Training nicht extra konstruiert werden. Sobald neue Rezepte dazu kommen, können sie in das Training integriert werden.

5. Wie kann das System evaluiert werden?

Um das Konzept zu validieren wurde es schließlich, sowohl mit einem künstlichen Datensatz, sowie mit dem Chefkoch-Datensatz, getestet und evaluiert. In Kapitel 6 werden die verwendeten Datensätze vorgestellt. Neben der statistischen Auswertung wurde für die Evaluierung der Ergebnisse auf dem Chefkoch-Datensatz zusätzlich eine Umfrage entworfen. Diese wurde durch eine potentielle Zielgruppe qualitativ bewertet.

8.2 Fazit

Die gewählte Methode, mit Neuronalen Netzen die Informationen aus Bild und Text zu extrahieren, bringt zwei Nachteile mit sich. Wenn zum Beispiel die Dessertart geändert würde, von kleinen Süßspeisen in Gläsern zu Süßigkeiten verpackt in einfarbigen Plastikkartons, könnten die Netze das nicht abfangen. Ein solches Dessert würde dann vermutlich nie in der Empfehlung auftauchen. Durch das Training lernen die Netze nur Konzepte die auch schon in den Daten vorhanden sind. Eine Lösungsmöglichkeit wäre hier, die Netze regelmäßig auf neueren Daten nach zu trainieren, sodass neue Trends zusätzlich integriert werden. Darüber hinaus benötigt diese Art von maschinellem Lernen eine große Menge an Trainingsdaten. In dem hier verwendeten Datensatz von Chefkoch ist dies besonders bei den Getränken deutlich geworden. Sie sind stark unterrepräsentiert, sodass die Netze nicht gelernt haben, sie von Essen zu differenzieren.

Der Ansatz geht zusätzlich davon aus, dass der verwendeten Datenbank ein weiteres Feld hinzugefügt werden kann, in welchem die Feature-Vektoren abgelegt werden oder dass diese anderswo zwischengespeichert werden können. Ist dies nicht möglich, wäre es trotzdem realisierbar, für alle Bilder während einer Text-Abfrage einen

Feature-Vektor zu berechnen. Allerdings würde dies einen erhöhten Rechenaufwand bedeuten.

Das vorgestellte System bietet wiederum einige Vorteile. Zum einen erlaubt es eine gewisse Flexibilität. Zu Demonstrationszwecken werden immer zehn Bilder vorgeschlagen, dennoch ist die Anzahl frei wählbar. Das Text-Netz ist sprachunabhängig. Es bedarf nur einiger Änderungen in der Vorverarbeitung des Textes. Dadurch, dass diese vom Trainingsprozess getrennt ist, muss dafür nicht das Netz selbst angepasst werden. Des Weiteren können sowohl das Text- als auch das Bild-Netz problemlos ausgetauscht werden. Die einzige Bedingung dabei ist, dass der Feature-Vektor die gleiche Größe aufweist und im selben Raum definiert ist. Dadurch besteht die Möglichkeit, das Bild-Netz und das Text-Netz unabhängig voneinander zu optimieren.

Darüber hinaus ist ein weiterer Vorteil, dass die Trainingsdaten nicht händisch erstellt oder künstlich konstruiert werden müssen. Es kann auf vorhandenen Bilder-Text Paaren aufgebaut werden. In dieser Arbeit wurden Rezepttexte mit Bildern verwendet, die bereits existieren. Die Umfrage hat bestätigt, dass die Originalpaare als sehr passend angesehen werden. Die Originalbilder bekamen im Durchschnitt eine Bewertung von 4,6 in Bezug darauf wie gut sie zum Rezepttext passen. Die hier verwendeten Trainingsdaten bieten zusätzlich die Möglichkeit, durch ihre Kategorien die einzelnen Komponenten, des Systems zu untersuchen. Dadurch kann überprüft werden, ob die Netze ungefähr das lernen, was der Entwickler erwartet.

Außerdem ist für die Nutzbarkeit des Systems wichtig, dass eine Textabfrage in realer Zeit möglich ist, auch wenn die Datenbanken zunehmend wachsen. Dies ist gegeben, wenn die Möglichkeit existiert, im Vorhinein für jedes Bild ein Feature-Vektor zu berechnen und zu speichern. Dann muss nur einmal ein Feature-Vektor für die Textabfrage berechnet werden. Dieser kann mit einfachem, linearen Aufwand, abhängig von der konkreten Anzahl der Bilder in der Datenbank, mit allen Bildern verglichen werden.

Zur Evaluation wurden die Ergebnisse des Trainings auf Grundlage des vorgestellten Konzeptes auf dem Rezeptdatensatz sowohl statistisch als auch qualitativ untersucht. Eine weitergehende umfassende statistische Bewertung ist wünschenswert aber hier nicht realisierbar, da keine weitere Grundwahrheit für ähnliche Bilder, abgesehen von dem Originalbild existiert. In dem Zusammenhang ist jedoch erwähnenswert, dass die Rezepte zwar ein gutes Trainingsbeispiel darstellen, allerdings in Bezug auf die tatsächliche Wiederverwendbarkeit schwierig sind. Denn die Teilnehmenden erwarteten genau das passende Bild zu einem Rezept. Leichte Abweichungen resultierten schnell in einer schlechten Bewertung. Dennoch ergab sich bei der Umfrage auch, dass die Teilnehmenden bei 82% der präsentierten Rezepte mindestens ein vorgeschlagenes Bild als passend empfanden.

Zusammenfassend hat sich der in diesem Ansatz entwickelte Prototyp als erfolgreich erwiesen. So konnte ein System entwickelt werden, welches in 80% der Fälle dazu beigetragen hat, dass ein passendes Bild in der Datenbank gefunden und vorgeschlagen wurde. Damit lässt sich sagen, dass dieses System die Forschungsfrage „Wie kann ein System aussehen, das einen Bildredakteur bei der Aufgabe archivierte Bilder aus hauseigenen oder internen unorganisierten Datenbanken wiederzuverwenden, unterstützt“ beantwortet. Dennoch bietet der Ansatz Potenzial für Optimierungen, um die Qualität der Vorschläge zu erhöhen.

8.3 Ausblick

Um das System in Bezug auf den Chefkoch-Datensatz zu verbessern, könnten verschiedene Aspekte berücksichtigt und ausprobiert werden. Zum einen kann ein anderes Neuronales Netz zur Verarbeitung der Bilder verwendet werden, da dieses in dem Konzeptansatz im Vergleich zum Text-Netz die schlechteren Ergebnisse erzielt. Eventuell hilft es auch, eine höhere Auflösung der Bilder zu verwenden, damit das Netz beispielsweise Himbeeren besser von Erdbeeren unterscheiden kann. Des Weiteren kann der Datensatz künstlich vergrößert werden, indem auch Rezepte aus anderen Sprachen verwendet werden. Diese könnten vorher übersetzt werden. Es könnte auch untersucht werden, ob das System für unterschiedliche Sprachen unterschiedliche Ergebnisse liefert. Das verwendete Text-Netz ist sprachunabhängig, von daher wäre nur eine sprachspezifische Vorverarbeitung notwendig. Dies kann eventuell helfen, weitere indirekte Einsichten in das Verfahren zu gewinnen.

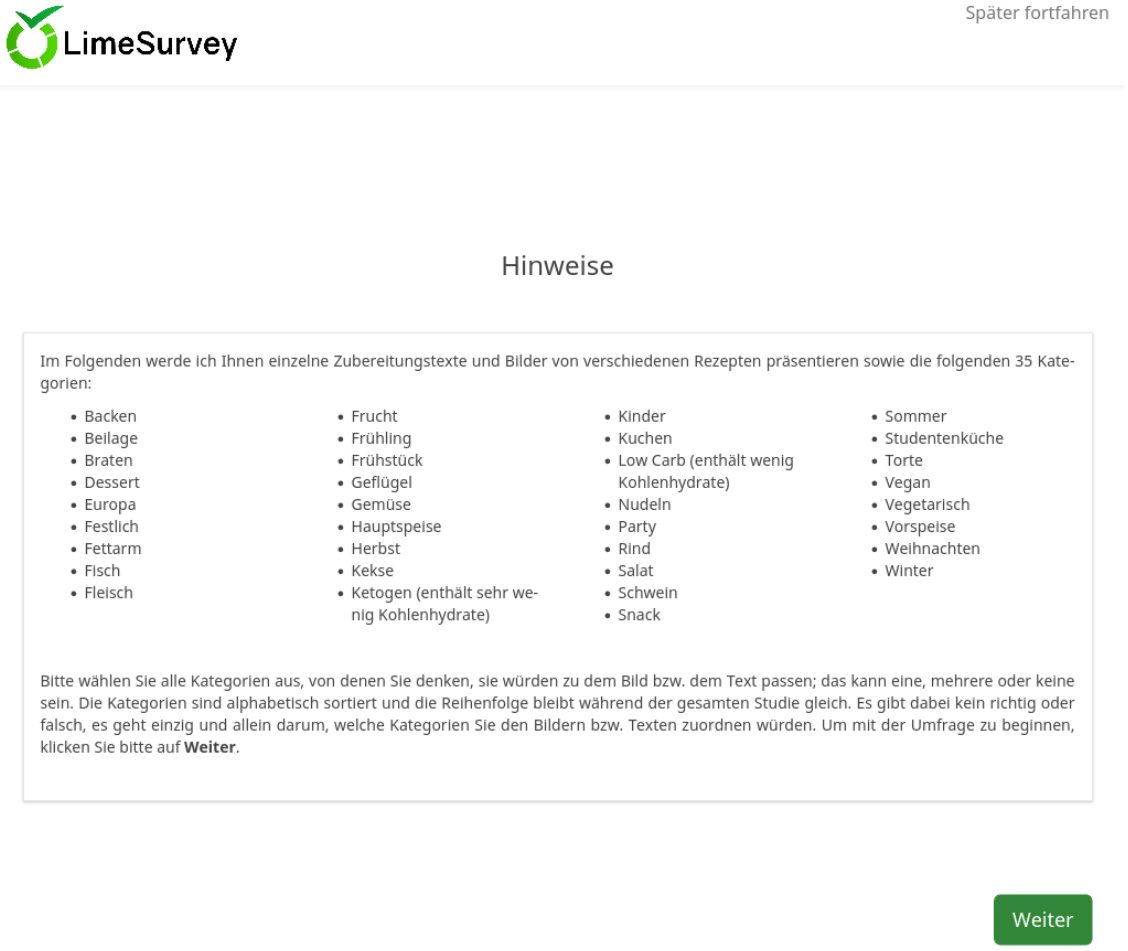
Ein weiterer Ansatz zur Optimierung auf dem bestehenden Datensatz kann auch eine andere Variante des Contrastive Loss sein. In dieser Arbeit wurden die Negativ-Paare aufgrund ihrer Zutatenzusammensetzung gewichtet. Je ähnlicher die Zusammensetzung war, umso weniger wurde das Paar für den Verlust gewertet. Es könnte aber auch anders herum funktionieren. Je unterschiedlicher die Zusammensetzung ist, umso stärker wird das Paar in die Berechnung miteinbezogen. Die Zutaten könnten darüber hinaus mit einer anderen Variante der Multi-Binären Kreuzentropie für das Klassifizierungsproblem verwendet werden. Damit würden die teilweise inkonsistent zugewiesenen Kategorien weniger ins Gewicht fallen.

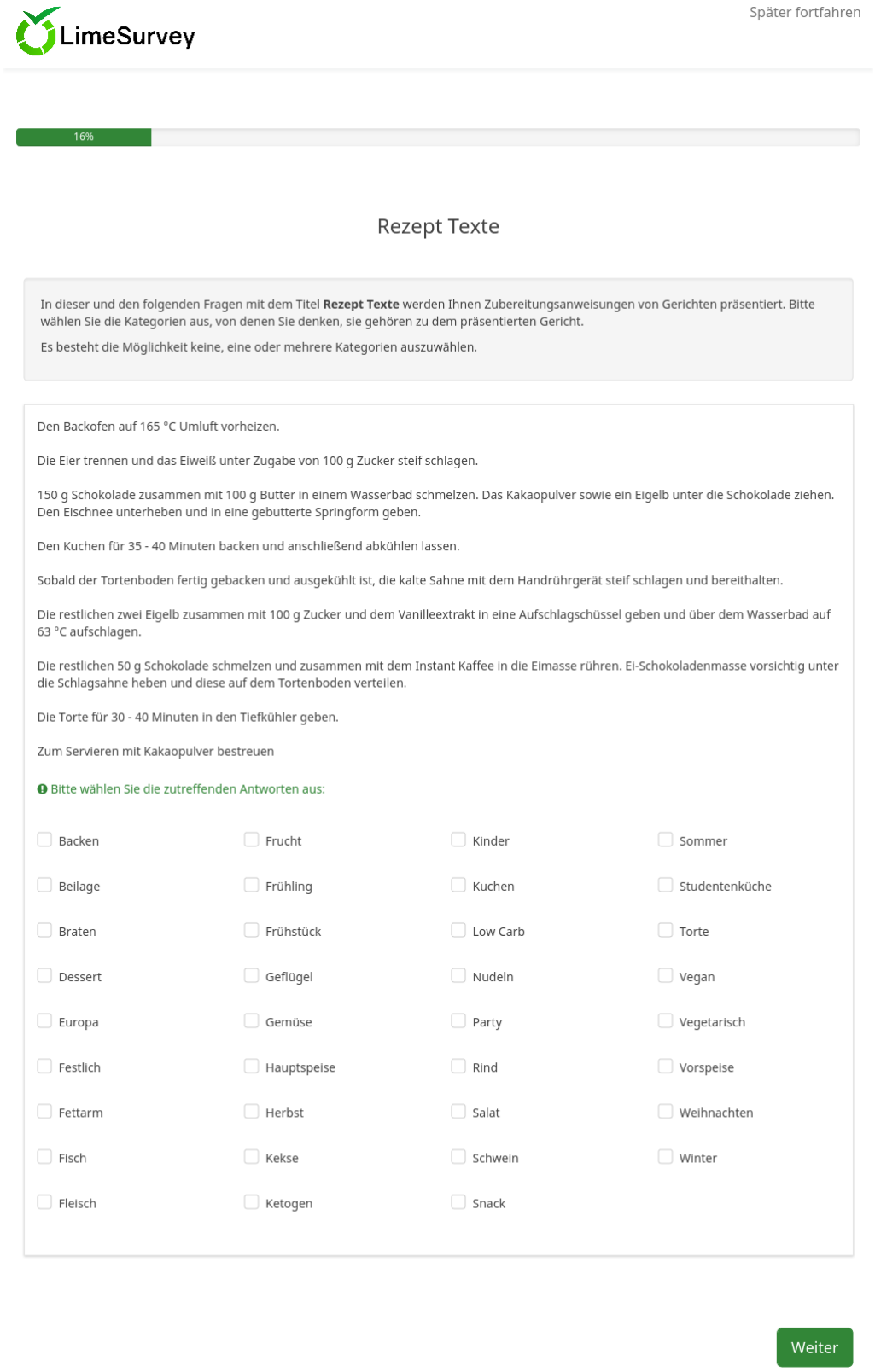
In dieser Arbeit wurde das Problem untersucht, adäquate Bilder für einen gegebenen Text zu finden. Das entwickelte Konzept bietet zusätzlich die Möglichkeit, genau das Gegenteil zu tun. Mit Bildern ein passendes Rezept suchen. Nicht in der Form, dass ein komplett neues Rezept generiert wird, sondern dass es eine Rezept-Datenbank gibt. Die Rezepttexte müssten mit einem Feature-Vektor hinterlegt sein. Dann könnte beispielsweise in einem Restaurant ein Gericht fotografiert werden und die zehn ähnlichsten Rezepttexte werden zum Nachkochen vorgeschlagen.

Über die Chefkoch-Rezepte hinaus könnte das Konzept auch mit Zeitungsartikeln getestet werden, bei denen die Verwendung von Stock-Fotos vermutlich verbreiteter ist. Eventuell funktioniert die Bildverarbeitung auf größeren Konzepten wie Menschen, Landschaft oder Autos besser, da das hier verwendete Bild-Netz auf solchen Konzepten vortrainiert wurde.

A Anhang

A.1 Fragebogen zur Erstellung des Goldstandards


LimeSurvey
Später fortfahren
Hinweise
Im Folgenden werde ich Ihnen einzelne Zubereitungstexte und Bilder von verschiedenen Rezepten präsentieren sowie die folgenden 35 Kategorien:
Backen
Beilage
Braten
Dessert
Europa
Festlich
Fettarm
Fisch
Fleisch
Frucht
Frühling
Frühstück
Geflügel
Gemüse
Hauptspeise
Herbst
Kekse
Ketogen (enthält sehr wenig Kohlenhydrate)
Kinder
Kuchen
Low Carb (enthält wenig Kohlenhydrate)
Nudeln
Party
Rind
Salat
Schwein
Snack
Sommer
Studentenküche
Torte
Vegan
Vegetarisch
Vorspeise
Weihnachten
Winter
Bitte wählen Sie alle Kategorien aus, von denen Sie denken, sie würden zu dem Bild bzw. dem Text passen; das kann eine, mehrere oder keine sein. Die Kategorien sind alphabetisch sortiert und die Reihenfolge bleibt während der gesamten Studie gleich. Es gibt dabei kein richtig oder falsch, es geht einzig und allein darum, welche Kategorien Sie den Bildern bzw. Texten zuordnen würden. Um mit der Umfrage zu beginnen, klicken Sie bitte auf Weiter.
Weiter


LimeSurvey
Später fortfahren
16%
Rezept Texte
In dieser und den folgenden Fragen mit dem Titel **Rezept Texte** werden Ihnen Zubereitungsanweisungen von Gerichten präsentiert. Bitte wählen Sie die Kategorien aus, von denen Sie denken, sie gehören zu dem präsentierten Gericht.
Es besteht die Möglichkeit keine, eine oder mehrere Kategorien auszuwählen.
Den Backofen auf 165 °C Umluft vorheizen.
Die Eier trennen und das Eiweiß unter Zugabe von 100 g Zucker steif schlagen.
150 g Schokolade zusammen mit 100 g Butter in einem Wasserbad schmelzen. Das Kakaopulver sowie ein Eigelb unter die Schokolade ziehen. Den Eischnee unterheben und in eine gebutterte Springform geben.
Den Kuchen für 35 - 40 Minuten backen und anschließend abkühlen lassen.
Sobald der Tortenboden fertig gebacken und ausgekühlt ist, die kalte Sahne mit dem Handrührgerät steif schlagen und bereithalten.
Die restlichen zwei Eigelb zusammen mit 100 g Zucker und dem Vanilleextrakt in eine Aufschlagschüssel geben und über dem Wasserbad auf 63 °C aufschlagen.
Die restlichen 50 g Schokolade schmelzen und zusammen mit dem Instant Kaffee in die Eimasse rühren. Ei-Schokoladenmasse vorsichtig unter die Schlagsahne heben und diese auf dem Tortenboden verteilen.
Die Torte für 30 - 40 Minuten in den Tiefkühler geben.
Zum Servieren mit Kakaopulver bestreuen
Bitte wählen Sie die zutreffenden Antworten aus:
Backen
Beilage
Braten
Dessert
Europa
Festlich
Fettarm
Fisch
Fleisch
Frucht
Frühling
Frühstück
Geflügel
Gemüse
Hauptspeise
Herbst
Kekse
Ketogen
Kinder
Kuchen
Low Carb
Nudeln
Party
Rind
Salat
Schwein
Snack
Sommer
Studentenküche
Torte
Vegan
Vegetarisch
Vorspeise
Weihnachten
Winter
Weiter

LimeSurvey

Später fortfahren

41%

Rezept Bilder

Bitte wählen Sie die zutreffenden Antworten aus:

Backen	Frucht	Kinder	Sommer
Beilage	Frühling	Kuchen	Studentenküche
Braten	Frühstück	Low Carb	Torte
Dessert	Geflügel	Nudeln	Vegan
Europa	Gemüse	Party	Vegetarisch
Festlich	Hauptspeise	Rind	Vorspeise
Fettarm	Herbst	Salat	Weihnachten
Fisch	Kekse	Schwein	Winter
Fleisch	Ketogen	Snack	

Weiter

A.2 Fragebogen zur Qualitativen Auswertung des Text-Bild Problems


LimeSurvey
Zwischengespeicherte Umfrage laden
0%
Rezeptbilder, wie gut passen sie zum Rezept?
Dies ist eine Umfrage zur Bewertung von Rezeptbildern zu einem Rezept.
Vielen Dank, dass Sie an dieser Umfrage teilnehmen. Im Rahmen meiner Masterarbeit an der Universität Bremen im Fachbereich Informatik habe ich zwei Neuronale Netze konzipiert, die Bild- und Textdaten verarbeiten. Das Ziel ist es, für Rezepttexte passende Bilder aus einer Datenbank zu suchen. Für eine qualitative Bewertung meiner Ergebnisse habe ich diese Umfrage erstellt. Die Fragen bestehen aus einem Rezepttext und 11 Bildvorschlägen. Für jedes Bild kann eine Bewertung von 1-5 abgegeben werden, wie gut oder nicht gut dass Bild zu dem Rezept passt. Wobei 1 bedeutet, dass das Bild gar nicht zu dem Rezepttext passt und 5 dass es sehr gut passt.
Die Bearbeitungszeit beträgt ca. 10 Min, insgesamt werden 5 Rezepte angezeigt. In dieser Umfrage werden keine persönlichen Daten erhoben oder gespeichert. Die Teilnahme ist freiwillig und kann zu jedem Zeitpunkt abgebrochen werden.
Sollten Sie weitere Fragen haben oder Schwierigkeiten bei der Bearbeitung auftreten, so stehe ich Ihnen sehr gerne unter maiweide@uni-bremen.de zur Verfügung.
Mit freundlichen Grüßen
Maira Weidenbach
Weiter

LimeSurvey

Später fortfahren

50%

Rezept 3

Bitte bewerten Sie, wie gut die Bilder zu dem vorgestellte Rezept passen:

5 - passt sehr gut, 4 - passt eher gut, 3 - ist okay, 2 - passt eher nicht, 1 - passt gar nicht

Alle Zutaten für den Teig zusammenmischen. Den Teig in die Springform geben und einen Rand von ca. 2 - 3 cm formen. Ich mache das immer mit den Händen, ohne Ausrollen, dadurch wird er zwar nicht überall exakt gleich hoch, aber ich komme damit besser zurecht, als mit dem Ausrollen.

Das Apfelmus in einen kleinen Topf geben und zum Köcheln bringen, ich gebe da schon immer ein wenig Zimt dazu. Währenddessen die Äpfel schälen, vierteln und in kleine Scheiben schneiden.

Die eine Hälfte der geschnittenen Äpfel auf den Boden geben, die andere Hälfte für später aufheben. Das Puddingpulver mit etwas Milch oder Apfelsaft anrühren. Wenn das Apfelmus leicht blubbert, das angerührte Vanillepuddingpulver dazugeben. Gleich kräftig umrühren und wie beim Puddingkochen den Topf von der Platte nehmen.

Dann die Masse auf den Boden geben. Es ist egal ob man die Masse ein wenig abkühlen lässt oder nicht, ich habe da keinen sonderlichen Unterschied festgestellt, wenn die Masse wärmer ist - finde ich - lässt sie sich besser verteilen.

Dann den Rest der geschnittenen Äpfel auf die Masse geben. Nun die Zutaten für die Streusel verkneten (und ja nicht mit dem Zimt sparen) und darauf verteilen. Den Kuchen dann bei ca. 160 °C Umluft für 45 - 50 Minuten in den Backofen geben.

	5 - passt sehr gut	4	3	2	1 - passt gar nicht	Keine Antwort
	○	○	○	○	○	●
	○	○	○	○	○	●

	○	○	○	○	○	●
	○	○	○	○	○	●
	○	○	○	○	○	●
	○	○	○	○	○	●
	○	○	○	○	○	●

	○	○	○	○	○	●
	○	○	○	○	○	●
	○	○	○	○	○	●
	○	○	○	○	○	●

Weiter

A.3 Vollständige Klassenspezifische Auswertung

	Texte		Bilder		Verteilung
Kategorie	Treff.	Genau.	Treff.	Genau.	in Prozent
Backen	77,08	81,93	68,51	80,49	37,59
Beilage	41,94	28,26	25,00	15,00	7,80
Braten	41,94	45,88	35,63	44,93	12,77
Dessert	76,00	27,54	81,03	30,92	9,22
Europa	28,87	20,74	29,59	21,97	13,48
Festlich	54,76	21,50	50,00	24,43	7,80
Fettarm	18,18	17,24	20,37	18,97	7,80
Fisch	89,55	100,00	54,55	92,31	7,80
Fleisch	67,61	36,64	54,10	31,73	9,93
Frucht	58,06	52,17	53,49	35,94	7,80
Frühling	25,49	30,95	27,66	33,33	7,80
Frühstück	52,94	72,97	27,66	30,23	7,80
Geflügel	83,93	97,92	49,06	78,79	9,22
Gemüse	54,32	87,28	45,10	85,43	40,43
Hauptspeise	80,85	80,57	71,48	85,95	40,43
Herbst	23,73	28,00	11,27	21,62	10,64
Kekse	64,44	78,38	67,24	81,25	7,80
Ketogen	3,77	7,14	9,26	38,46	7,80
Kinder	26,39	32,76	24,14	23,33	9,22
Kuchen	69,81	40,22	58,33	29,79	7,80
Low Carb	18,95	25,35	19,57	30,51	14,18
Nudeln	73,77	78,95	70,18	64,52	8,51
Party	43,68	35,19	32,81	17,36	10,64
Rind	42,03	96,67	29,73	73,33	9,93
Salat	56,45	77,78	49,02	73,53	7,80
Schwein	40,98	49,02	21,15	37,93	7,80
Snack	35,16	42,67	35,06	29,35	12,06
Sommer	40,28	24,37	31,51	20,35	10,64
Studentenküche	29,63	19,05	21,57	14,86	7,80
Torte	82,00	67,21	68,18	77,59	7,80
Vegan	44,44	80,00	25,68	43,18	10,64
Vegetarisch	50,70	67,92	44,95	60,00	45,39
Vorspeise	22,45	28,21	27,91	20,69	7,80
Weihnachten	63,64	63,64	57,14	72,73	9,93
Winter	33,33	24,66	33,78	32,89	11,35
Gemittelt	52,71	52,56	45,46	47,48	-

Tabelle A.1: Auswertung Fragebogen zur Klassifizierung von Bild und Text Daten

	Validierung		Test		Umfrage	
Kategorie	Treff.	Genau.	Treff.	Genau.	Treff.	Genau.
Backen	86,73	86,86	74,94	74,94	73,58	82,98
Beilage	8,92	46,67	4,41	42,86	0,00	0,00
Braten	24,88	53,69	19,87	52,54	16,67	50,00
Dessert	43,02	69,86	23,53	52,17	23,08	50,00
Europa	0,00	0,00	0,00	0,00	0,00	0,00
Festlich	1,21	37,50	5,36	75,00	0,00	0,00
Fettarm	0,00	0,00	0,00	0,00	0,00	0,00
Fisch	2,86	64,29	2,13	50,00	0,00	0,00
Fleisch	13,48	40,85	10,06	34,69	14,29	33,33
Frucht	9,23	51,75	5,65	53,85	9,09	50,00
Frühling	15,86	55,38	12,96	77,78	9,09	100,00
Frühstück	8,11	39,13	5,88	50,00	9,09	50,00
Geflügel	10,79	60,29	6,67	44,44	7,69	50,00
Gemüse	68,94	68,76	65,14	72,46	73,21	78,85
Hauptspeise	80,05	78,17	72,82	75,81	78,57	83,02
Herbst	1,36	60,00	0,99	100,00	0,00	0,00
Kekse	50,14	67,67	48,00	54,55	63,64	100,00
Ketogen	1,81	29,63	3,53	50,00	0,00	0,00
Kinder	18,29	56,63	9,52	72,73	7,69	100,00
Kuchen	67,00	72,33	52,76	65,15	54,55	50,00
Low Carb	2,69	32,88	7,64	54,55	10,53	100,00
Nudeln	58,62	67,01	46,91	66,67	33,33	50,00
Party	15,68	49,84	20,57	56,58	26,67	50,00
Rind	28,32	45,25	27,48	60,00	21,43	75,00
Salat	46,03	63,81	37,68	63,41	27,27	50,00
Schwein	24,96	39,14	15,48	40,62	27,27	42,86
Snack	24,45	55,95	30,00	51,00	29,41	62,50
Sommer	13,56	53,39	6,78	53,33	28,57	80,00
Studentenküche	0,00	0,00	0,00	0,00	0,00	0,00
Torte	69,43	71,41	40,00	40,00	36,36	66,67
Vegan	7,87	42,02	3,17	10,53	6,67	25,00
Vegetarisch	61,39	62,42	68,72	52,54	77,78	67,12
Vorspeise	17,82	52,84	18,18	52,94	27,27	75,00
Weihnachten	26,06	53,18	18,31	52,00	42,86	100,00
Winter	0,89	23,81	0,00	0,00	0,00	0,00
Gemittelt	41,66	68,50	36,22	63,05	38,07	70,88

Tabelle A.2: Auswertung des Bild-Netzes zur Klassifizierung von Bild Daten

	Validierung		Test		Umfrage	
Kategorie	Treff.	Genau.	Treff.	Genau.	Treff.	Genau.
Backen	94,74	96,38	89,11	91,91	72,22	96,30
Beilage	26,07	74,84	24,59	65,22	37,50	75,00
Braten	70,30	66,71	77,93	72,90	72,73	72,73
Dessert	70,65	77,31	76,00	70,37	75,00	75,00
Europa	4,77	58,23	8,15	68,75	12,50	100,00
Festlich	16,02	52,24	16,67	47,37	0,00	0,00
kettarm	1,06	100,00	0,00	0,00	0,00	0,00
Fisch	88,45	88,16	89,13	91,11	90,00	100,00
Fleisch	61,20	62,14	55,06	62,14	60,00	46,15
Frucht	33,91	56,45	35,00	75,00	37,50	60,00
Frühling	60,64	75,15	61,54	84,21	100,00	100,00
Frühstück	16,45	66,67	24,24	80,00	60,00	100,00
Geflügel	80,11	91,77	89,09	98,00	88,89	100,00
Gemüse	88,84	78,34	88,61	83,73	93,02	85,11
Hauptspeise	91,58	87,44	90,04	86,23	90,48	90,48
Herbst	33,99	67,97	36,08	67,31	33,33	80,00
Kekse	82,91	88,51	83,33	85,11	100,00	100,00
Ketogen	19,86	53,90	22,50	54,55	40,00	100,00
Kinder	17,27	62,33	23,17	70,37	40,00	100,00
Kuchen	92,00	87,09	89,10	83,23	50,00	75,00
Low Carb	25,60	62,46	29,37	70,00	27,27	100,00
Nudeln	90,39	86,65	87,01	85,90	100,00	76,92
Party	34,08	62,76	39,09	62,60	50,00	54,55
Rind	82,09	75,79	79,17	84,07	77,78	87,50
Salat	83,92	84,62	80,30	81,54	90,00	90,00
Schwein	72,77	62,32	65,79	66,23	50,00	57,14
Snack	45,94	67,35	54,88	64,75	61,54	80,00
Sommer	32,97	64,59	27,43	51,67	50,00	62,50
Studentenküche	2,45	52,63	0,00	0,00	0,00	0,00
Torte	85,46	86,30	78,95	53,57	87,50	77,78
Vegan	40,85	61,64	48,33	51,79	54,55	85,71
Vegetarisch	80,42	77,50	84,04	71,10	78,57	89,19
Vorspeise	42,58	60,00	32,63	38,75	0,00	0,00
Weihnachten	49,05	61,26	37,88	62,50	55,56	83,33
Winter	28,27	59,04	32,50	56,52	41,67	100,00
Gemittelt	63,88	78,32	63,52	75,85	63,47	83,09

Tabelle A.3: Auswertung des Text-Netzes zur Klassifizierung von Text Daten

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). "Layer Normalization." *arXiv:1607.06450 [cs, stat]*.

Belinkov, Y., and Bisk, Y. (2018). "Synthetic and Natural Noise Both Break Neural Machine Translation." *arXiv:1711.02173 [cs]*.

Chang, S.-K., and Hsu, A. (1992). "Image information systems: where do we go from here?" *IEEE Transactions on Knowledge and Data Engineering*, *4*(5), 431–442.

Chefkoch (2021). "Chefkoch presse." `https://www.chefkoch.de/magazin/artikel/7027/Chefkoch/presse.html`, abgerufen am 22.08.21.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). "A Simple Framework for Contrastive Learning of Visual Representations." *arXiv:2002.05709 [cs, stat]*.

Christopher D Manning, H. S., and Raghavan, P. (2008). "Introduction to information retrieval." *Cambridge University Press*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *arXiv:1810.04805 [cs]*.

dpa (2020). "Zahlen und fakten." `https://www.dpa.com/de/unternehmen/zahlen-fakten`, abgerufen am 26.08.21.

Duden (2020). "Umfang des deutschen wortschatzes." `https://www.duden.de/sprachwissen/sprachratgeber/Zum-Umfang-des-deutschen-Wortschatzes`, abgerufen am 22.08.21.

Duden (2021a). "Groß- und kleinschreibung." `https://www.duden.de/sprachwissen/rechtschreibregeln/Gro%C3%9F-%20und%20Kleinschreibung`, abgerufen am 23.08.21.

Duden (2021b). "unterstützen." `https://www.duden.de/rechtschreibung/unterstuetzen_beistehen_helfen_entlasten`, abgerufen am 26.08.21.

Eckart, W. U. (2009). "Umrisse einer Medizin des 20. und frühen 21. Jahrhunderts." In W. U. Eckart (Ed.), *Geschichte der Medizin: Fakten, Konzepte, Haltungen*, Springer-Lehrbuch, 245–326, Berlin, Heidelberg: Springer.

Elman, J. L. (1990). "Finding Structure in Time." *Cognitive Science*, *14*(2), 179–211.

Goldberg, Y. (2017). *Neural Network Methods for Natural Language Processing*. No. 37 in Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers.

Grangier, D., and Bengio, S. (2006). "A Neural Network to Retrieve Images from Text Queries." 10.

Gunawan, D., Sembiring, C. A., and Budiman, M. A. (2018). "The Implementation of Cosine Similarity to Calculate Text Relevance between Two Documents." *Journal of Physics: Conference Series*, *978*, 012120.

Hadsell, R., Chopra, S., and LeCun, Y. (2006). "Dimensionality Reduction by Learning an Invariant Mapping." In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 1735–1742.

Hochreiter, S., and Schmidhuber, J. (1997). "Long Short-Term Memory." *Neural Computation*, *9*(8), 1735–1780.

Ioffe, S., and Szegedy, C. (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." *arXiv:1502.03167 [cs]*.

Jurafsky, D., and Martin, J. H. (2020). *Speech and Language Processing, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, third edtion draft edn.

Jähne, B. (2012). *Digitale Bildverarbeitung*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Karpathy, A., and Fei-Fei, L. (2015). "Deep Visual-Semantic Alignments for Generating Image Descriptions." *arXiv:1412.2306 [cs]*.

Kessler, S. H., Reifegerste, D., and Guenther, L. (2016). "Die Evidenzkraft von Bildern in der Wissenschaftskommunikation." 23.

Kingma, D. P., and Ba, J. (2017). "Adam: A Method for Stochastic Optimization." *arXiv:1412.6980 [cs]*.

Klahold, A. (2009). *Empfehlungssysteme: Recommender Systems ; Grundlagen, Konzepte und Lösungen*. Studium, Wiesbaden: Vieweg + Teubner, 1. aufl edn.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). "Backpropagation Applied to Handwritten Zip Code Recognition." *Neural Computation*, *1*(4), 541–551.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lu, X., Zhao, T., and Lee, K. (2021). "VisualSparta: Sparse Transformer Fragment-level Matching for Large-scale Text-to-Image Search." *arXiv:2101.00265 [cs]*.

Ma, L., Lu, Z., Shang, L., and Li, H. (2015). "Multimodal Convolutional Neural Networks for Matching Image and Sentence." In *2015 IEEE International Conference on Computer Vision (ICCV)*, 2623–2631, Santiago, Chile: IEEE.

Martin, K. (2010). "Lineare algebra." In *Das große Tafelwerk*, 64–70, Berlin, DE: Cornelsen.

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., and Gao, J. (2021). "Deep Learning Based Text Classification: A Comprehensive Review." *arXiv:2004.03705 [cs, stat]*.

MongoDB (2021). "Die datenbank für moderne anwendungen." `https://www.mongodb.com/de-de`, abgerufen am 24.08.21.

Nam, J., Kim, J., Loza Mencía, E., Gurevych, I., and Fürnkranz, J. (2014). "Large-Scale Multi-label Text Classification — Revisiting Neural Networks." In T. Calders, F. Esposito, E. Hüllermeier, and R. Meo (Eds.), *Machine Learning and Knowledge Discovery in Databases*, vol. 8725, 437–452, Berlin, Heidelberg: Springer Berlin Heidelberg, series Title: Lecture Notes in Computer Science.

Pan, S. J., and Yang, Q. (2010). "A Survey on Transfer Learning." *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.

pixabay (2021). "Pixabay webseite." `https://pixabay.com/`, abgerufen am 26.08.21.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). "Zero-Shot Text-to-Image Generation." *arXiv:2102.12092 [cs]*.

Ruiz, D. V., Krinski, B. A., and Todt, E. (2020). "IDA: Improved Data Augmentation Applied to Salient Object Detection." In *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 210–217.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2019). "MobileNetV2: Inverted Residuals and Linear Bottlenecks." *arXiv:1801.04381 [cs]*.

Sartorius, G. (2019). *Erfassen, Verarbeiten und Zuordnen multivariater Messgrößen: Neue Rahmenbedingungen für das Nächste-Nachbarn-Verfahren*. Wiesbaden: Springer Fachmedien Wiesbaden.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2020). "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization." *International Journal of Computer Vision*, *128*(2), 336–359.

Shutterstock (2021). "Shutterstock webseite." `https://www.shutterstock.com/search/*`, abgerufen am 26.08.21.

Singhal, A. (2001). "Modern Information Retrieval: A Brief Overview." 9.

Tan, P.-N., Steinbach, M., and Kumar, V. (2006). "Cluster Analysis: Basic Concepts and Algorithms." In *Introduction to Data Mining*, 487–568, Pearson Education.

Thrun, S., and Pratt, L. (1998). "Learning to Learn: Introduction and Overview." In S. Thrun, and L. Pratt (Eds.), *Learning to Learn*, 3–17, Boston, MA: Springer US.

T.karthikeyan, P. m., and aprabhu, S. n. (2014). "A Survey on Text and Content Based Image Retrieval System for Image Mining." *International Journal of Engineering Research*, *3*(3), 4.

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., and Bregler, C. (2015). "Efficient Object Localization Using Convolutional Networks." *arXiv:1411.4280 [cs]*.

Vo, N., Jiang, L., Sun, C., Murphy, K., Li, L.-J., Fei-Fei, L., and Hays, J. (2019). "Composing Text and Image for Image Retrieval - an Empirical Odyssey." In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6432–6441, Long Beach, CA, USA: IEEE.

Walter Hussy, G. E., Margrit Schreier (2013). *Forschungsmethoden in Psychologie und Sozialwissenschaften für Bachelor*. Springer, Berlin, Heidelberg.

with Code, P. (2021). "Image classification on image net." `https://paperswithcode.com/sota/image-classification-on-imagenet`, abgerufen am 25.08.21.

Wu, T., Huang, Q., Liu, Z., Wang, Y., and Lin, D. (2020). "Distribution-Balanced Loss for Multi-label Classification in Long-Tailed Datasets." In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm (Eds.), *Computer Vision – ECCV 2020*, vol. 12349, 162–178, Cham: Springer International Publishing, series Title: Lecture Notes in Computer Science.

Xiao, Y., and Cho, K. (2016). "Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers." *arXiv:1602.00367 [cs]*.

Yann LeCun, C. J. B., Corinna Cortes (2021). "The mnist database of handwritten digits." `http://yann.lecun.com/exdb/mnist/`, abgerufen am 24.08.21.

Zhang, X., Zhao, J., and LeCun, Y. (2016). "Character-level Convolutional Networks for Text Classification." *arXiv:1509.01626 [cs]*.

Zweig, K. A., Deussen, O., and Krafft, T. D. (2017). "Algorithmen und Meinungsbildung: Eine grundlegende Einführung." *Informatik-Spektrum*, *40*(4), 318–326.

Erklärung zur selbstständigen Abfassung der Masterarbeit

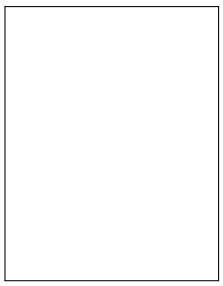
Ich versichere, dass ich die eingereichte Masterarbeit selbstständig und ohne unerlaubte Hilfe verfasst habe. Anderer als der von mir angegebenen Hilfsmittel und Schriften habe ich mich nicht bedient. Alle wörtlich oder sinngemäß den Schriften anderer Autoren entnommenen Stellen habe ich kenntlich gemacht.

Bremen, den 27. August 2021

(Maira Weidenbach)

Matrikelnummer: 3170774
Geburtsdatum: 16.02.1994
Titel der Masterarbeit: Künstliche Neuronale Netze zur Unterstützung von Bildredaktionen durch Kombination von Bild- und Textverarbeitung

Datenträger



Auf dem Datenträger befinden sich:

- Diese Arbeit in Form einer PDF-Datei.
- Der Quellcode zu dieser Arbeit inklusive Demo-Python Notebooks.
- Gespeicherte Gewichte der konstruierten Netze.
- Beispiel Screenshots der Umfragen
- Die Daten aus dem MNIST-Datensatz. Der Datensatz von Chefkoch GmbH darf in dieser Form nicht verbreitet werden und befinden sich daher nicht auf dem Datenträger.