HOCHSCHULE OSNABRÜCK
UNIVERSITY OF APPLIED SCIENCES

Institut für Duale Studiengänge

Prüfungsleistung
im Studiengang Wirtschaftsinformatik

Image Recognition mit TensorFlow

Eingereicht von:	Matthias Fischer (700643) Fabian Hagengers (701292)
Studiengruppe:	15DWF1
Betreuer:	Prof. Dr. Heiko Tapken
Modul:	Big-Data
Abgabedatum:	02.04.2018

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Image Recognition an sich ist zwar keine Neuheit und besteht schon seit dem Ende des 20. Jahrhunderts, hat allerdings vor allem in den letzten Jahren im Kontext von Begriffen wie bspw. Big Data, Machine Learning oder Deep Learning zunehmend an Bedeutung gewonnen. Dies lässt sich auch daran erkennen, dass viele Unternehmen, darunter auch bedeutende Unternehmen wie z. B. Google, Facebook, Microsoft, Apple oder Pinterest, beträchtliche Ressourcen in die Erforschung und Entwicklung von Software zur Bilderkennung investieren. Dies ist nicht zuletzt auch darauf zurückzuführen, dass sich mit Hilfe der Bilderkennung Anwendungen in verschiedenen Bereichen wie gezielter Werbung, angepassten Suchen oder „smarten“ Fotobibliotheken realisieren lassen.[1]

Da die Funktionalität auf der untersten Ebene solcher Bilderkennungssoftware auf sehr komplexen informationstechnischen und mathematischen Algorithmen beruht, nehmen viele Menschen diese als gegeben hin. Ziel dieser Arbeit ist es somit, anhand von sog. Convolutional Neural Networks einen Überblick über die Funktionsweise der Bilderkennung (Image Recognition) und Bildklassifizierung (Image Classification) zu geben und anhand eines praktischen Beispiels in die Software TensorFlow zum Trainieren und Erkennen von Bildern einzuführen.

Dazu wird zunächst darauf eingegangen was genau Image Recognition ist und wie dies in den übergeordneten Kontext von Pattern Recognition, Machine Learning und Big Data einzuordnen ist. Danach wird die Software TensorFlow vorgestellt und darauf eingegangen, wie das Prinzip des Image Retrainings und Image Recognitionings innerhalb dieser funktioniert. Im Anschluss daran wird auf das Convolutional Neural Network als Möglichkeit zur Bildanalyse eingegangen und erläutert, wie dieses aufgebaut ist und funktioniert und wie Image Retraining und Image Recognition mit Hilfe dessen umgesetzt werden können. Darauffolgend beginnt der Praxisteil, in dem zunächst darauf eingegangen wird, wie TensorFlow installiert und eingerichtet wird. Dann wir dargestellt, wie TensorFlow mit Hilfe seiner Python-API verwendet werden kann, um Image Retraining und Image Recognition durchzuführen.

2 Einordnung und Funktionsweise des Image Recognitionings

Image Recognition (deutsch: Bilderkennung) bezeichnet das Extrahieren, Erkennen und Bezeichnen von bspw. Objekten, Orten, Menschen, Schriften oder Handlungen.[2] Das, was Menschen schon als Kinder lernen und als selbstverständlich hinnehmen, ist für Computer

[1] Margaret Rouse [2017]
[2] Margaret Rouse [2017]

ein schwer lösbares Problem, dessen Lösung sehr komplexe und rechenintensive Operationen benötigt. Im Gegensatz zu Menschen sieht der Computer ein Bild lediglich als eine Aneinanderreihung von Pixeln bzw. eine Folge von Bits und kann anhand dieser zunächst nichts erkennen. [3] [4] Eine Analyse des Bildes zum Extrahieren sinnvoller Informationen beherrscht ein Computer zunächst nicht.

Ein erster Schritt des Image Recognitionings ist es somit, dem Computer das Erkennen von Merkmalen beizubringen (**Feature Detection**). [5] Das heißt, ihn anzulernen, anhand der Pixeldaten einfache Merkmale wie z. B. Linien oder Kurven zu erkennen. Dies führt dazu, dass der Computer zwar einfache Merkmale erkennen und extrahieren kann, allerdings besitzt er im Gegensatz zu den Menschen kein weiteres Wissen, es fehlt die Intelligenz. Der Computer muss bei jedem neuen Bild wieder von vorne beginnen („bottom-up“). [6] Menschen sehen Merkmale und merken sich diese automatisch, sie generieren Wissen. Auch der Computer sollte sich dieses Wissen speichern. Dazu müssen Menschen dem Computer zeigen, wie bestimmte Merkmale aussehen und wie er diese erkennen kann (**Feature Learning**). [7]

Nun kann der Computer lediglich die einfachen Merkmale extrahieren, diese allerdings in keinen Kontext setzen und keine komplexeren Muster erkennen. Menschen hingegen besitzen diese Intelligenz bestimmte triviale Merkmale zu Mustern zusammenzusetzen. Dies stellt den nächsten notwendigen Schritt dar: die Mustererkennung (Pattern Recognition). Dies meint, komplexere Merkmale, die sich aus einfachen Merkmalen zusammensetzen, als wiederkehrende Muster in Bildern erkennen zu können. So können zwei horizontale und zwei vertikale Linien (einfache Merkmale), die auf eine bestimmte Weise angeordnet sind, ein Rechteck als Muster ergeben. Ein Muster ist somit eine Menge einfacher Merkmale. Diese Muster können entsprechend der zusammensetzenden Merkmale immer komplexer werden. Für einen Computer würde dies heißen, dass eine bestimmte Anordnung bestimmter Pixel (bzw. eine bestimmte Anordnung von Bits) ein bestimmtes Muster ergeben. [8] Doch selbst wenn der Computer nun Muster erkennen kann, sollte er sich diese auch merken. Menschen erkennen Muster und merken sich diese automatisch, sie speichern diese ab und Generieren somit Wissen. Indem Menschen dem Computer beibringen wie bestimmte Muster in Bildern aussehen und aus welchen Merkmalen sich diese zusammensetzen, kann auch der Computer ein solches Wissen generieren (**Pattern Classification**). [9]

[3] TensorFlow [2017b]
[4] [Burger and Burge, 2015, S. 3]
[5] [Lindeberg, 1998, S. 79–80]
[6] [Burger and Burge, 2015, S. 3]
[7] [Habibi Aghdam and Jahani Heravi, 2017, S. 7–8]
[8] [Burger and Burge, 2015, S. 3]
[9] [Habibi Aghdam and Jahani Heravi, 2017, S. 15–16]

Danach muss dem Computer noch beigebracht werden, statt lediglich Muster ganze Objekte in Bildern zu erkennen (**Object Detection**). Menschen erkennen Objekte anhand verschiedener Muster. Dies muss nun auch der Computer können, das heißt er soll anhand mehrerer zusammengehöriger Muster in einem Bild ein Objekt identifizieren können. Ein Objekt ist somit eine Menge von Mustern. Für einen Computer würde dies heißen, in einem Bild eine bestimmte Anordnung bzw. einen bestimmten Zusammenhang von Mustern (entsprechend von Pixeln bzw. Bits) und anhand dieser somit Objekte zu erkennen. [10] Auch in diesem Falle erkennen Menschen Objekte und merken sich, was diese darstellen. Auch hier muss der Computer ein Wissen entwickeln, das heißt ihm muss von Menschen beigebracht werden, welche Muster sich in welcher Konstellation zu welchem Objekt zusammensetzen (**Object Classification**). [11]

Mit Hilfe dieses Stands können nun mittels Object Detection alle Objekte in einem Bild erkannt und benannt werden. Dahingegen wird beim **Image Recognition** (auch: ***Image Classifiation***) lediglich ein Objekt bzw. eine Klasse in einem Bild erkannt und benannt. Hierbei kann es zwar sein, dass in einem Bild mehrere Objekte bzw. deren Muster erkannt werden, es wird allerdings für jede bekannte Klasse eine Wahrscheinlichkeit berechnet mit der die Klasse in dem Bild vorhanden ist und lediglich die Klasse mit der höchsten Wahrscheinlichkeit wird beim Image Recognitioning benannt (der genaue Prozess wird bei der Erklärung des Covolutional Neural Networks näher erläutert). [12] [13]

An diesem Stand müssten Menschen die Feature Classification, Pattern Classification und Object Classification allerdings manuell vornehmen, das heißt dem Computer genau beibringen wie (auf Pixelebene bzw. Bitebene) ein Merkmal, ein Muster oder ein Objekt aussieht, was sehr aufwendig wäre. Zudem würde die Erkennung solcher nicht mehr funktionieren, wenn Merkmal, Muster oder Objekt ein wenig von ebenjenem Beigebrachten abweicht. An dieser Stelle spielt das **Machine Learning** eine Rolle. Mit Hilfe dessen sollen die Classification-Prozesse automatisiert werden.

Beim maschinellen Lernen wird vor allem zwischen zwei Varianten unterschieden: dem überwachten Lernen (**supervised learning**) und dem unüberwachten Lernen (**unsupervised learning**). In beiden Fällen wird dem Computer ein sog. Training Set an Daten (in diesem Falle Bilder von einzelnen Objekten) bereitgestellt. Beim unüberwachten Lernen werden vom Computer eigenständig Muster in den gegebenen Daten erkannt und datenübergreifend verglichen, um Gemeinsamkeiten und Unterschiede in den Daten zu erkennen und diese ggf. gruppieren oder kategorisieren zu können, ohne genau zu wissen, was die Daten darstellen. Bei der Klassifizierung hingegen wird das überwachte Lernen angewendet.

[10] [Karray et al., 2017, S. 247–253]
[11] [Habibi Aghdam and Jahani Heravi, 2017, S. 277]
[12] Satya Mallick [2016]
[13] Vipul Jain [2018]

Hierbei wird dem Computer für jeden Datensatz mitgeteilt, welche Klasse dieser darstellt (**Labeling**). Auch hier sollen Objekte und deren Muster und Merkmale erkannt werden, können aber durch das Labeling der jeweiligen Klasse zugeordnet werden, sodass durch das Training Set die Muster und Merkmale der Klassen verfeinert, differenziert und generalisiert werden können. Der Mensch muss dem Computer somit nicht mehr beibringen, wie genau die Muster einer Klasse genau aussehen, sondern der Computer erkennt anhand des Training Sets eigenständig ebensolche und weist sie der Klasse zu. Er erlernt die Muster der gegebenen Klassen selbstständig (Machine Learning). [14]

Nun lässt sich Machine Learning allgemein und im Kontext von Image Recognition auch in Beziehung zum Themenbereich **Big Data** setzen. So profitieren Big Data und Machine Learning jeweils voneinander. So muss auf der einen Seite die stetig wachsende Menge an Daten analysiert werden. Hierbei kann Machine Learning von großem Nutzen sein, da Computer die Daten somit automatisiert analysieren können und durch ihre Analyseverfahren durch das Machine Learning stetig selbstständig verbessern. Auf der anderen Seite werden, wie bereits dargestellt, Training Sets von Daten verwendet, um dem Computer Merkmale und Muster der Klassen anzutrainieren und zu optimieren und so die Analyseverfahren zu verbessern. Je größer dabei die Training Sets sind, desto besser werden diese. Durch die stetig wachsende Menge an Daten stehen dem Computer somit immer mehr Daten für die Training Sets zur Verfügung, sodass die Analyseverfahren stetig besser werden. Konkret für Image Recognition bedeutet dies, dass durch Big Data immer mehr Bildmaterial zur Verfügung steht, anhand dessen die Merkmale und Muster der Klassen und die Analyseverfahren zur Bilderkennung stetig optimiert werden. [15] [16]

3 TensorFlow als Framework zur maschinellen Bildverarbeitung

TensorFlow ist ein Open-Source Framework für maschinelles Lernen und soll die Forschung und Entwicklung in diesem Umfeld beschleunigen und verbessern. Mit Hilfe von TensorFlow können im Umfeld von Sprache und Bildverarbeitungsaufgaben neuronale Netze implementiert werden. [17] [18] Neuronale Netze können in TensorFlow mittels gerichteter zyklenfreier Graphen dargestellt werden. Dabei werden die Inputs und Outputs der einzelnen Rechenschritte durch die Kanten und die Verarbeitung des Inputs zu Output durch die Knoten der Graphen repräsentiert. [19] TensorFlow findet dabei in vielen Bereichen

[14] [Silva and Zhao, 2016, S. 71–72]
[15] Molly Galetto [2018]
[16] Marco Varone
[17] Otto Geißler and Nico Litzel [2018]
[18] TensorFlow [2018a]
[19] Otto Geißler and Nico Litzel [2018]

Anwendung wie z. B. Industrie, Wirtschaft, Finanzen oder Medizin für bspw. Sprachübersetzung oder Früherkennung von Hautkrebs und wird entsprechend auch von vielen großen Unternehmen verwendet wie z. B. SAP, ebay, Twitter oder Nvidia. [20] [21]

Ein wichtiger Bereich auf den sich TensorFlow konzentriert ist die Bildverarbeitung. TensorFlow teilt diesen wiederum in zwei Sektionen auf: Image Retraining und Image Recognition. Beim Image Retraining geht es vor allem um das Anlernen des Computers. Dazu wird auch bei TensorFlow ein Training Set verwendet, dass Bilder bestimmter Klassen und deren Labels enthält. Diese werden dann von TensorFlow verarbeitet, sodass sich der Computer die Merkmale und Muster der Klassen aneignen kann. [22] Beim Image Recognition geht hingegen darum, zu Erkennen, welche Klasse auf einem Bild, das man dem Computer als Input gibt abgebildet ist. [23] Für das Image Retraining und das Image Recognitioning bietet TensorFlow sowohl eine Python API als auch eine C++ API. Da TensorFlow eine Open Source Library ist, können diese zudem nach Bedarf angepasst werden. Zur direkten praktischen Anwendung des Image Retrainings und Image Recognitionings bietet TensorFlow zudem fertige Skripte an mit Hilfe derer die Software eigens angelernt werden kann oder beim Input eines Bildes die Wahrscheinlichkeit erkannter Klassen ausgibt. Hierzu kann z. B. ein bereits antrainiertes Model heruntergeladen werden oder Training Sets mit vielen Bildern verschiedener Klassen (diese findet man ebenfalls auf der Webseite von TensorFlow). Wie das Image Retraining mit Hilfe der Python API, von TensorFlow zur Verfügung gestellter Skripte und einem eigenen Training Set funktioniert und wie im Anschluss daran Image Recognitioning mit TensorFlow mittels der Python API, von TensorFlow zur Verfügung gestellter Skripte und einem Input-Bild einer antrainierten Klasse betrieben werden kann, wird im praktischen Teil der Ausarbeitung erläutert. Auf der anderen Seite bietet TensorFlow allerdings auch eine Bibliothek mit vielen Funktionen, um in Python oder C++ seine eigenen Skripte zum Image Retraining und Image Recognitioning zu schreiben und so ein eigenes Convolutional Neural Network zu entwickeln.

4 Convolutional Neural Networks

TensorFlow arbeitet im Bereich Machine Learning allgemein mit künstlichen neuronalen Netzwerken und im Bereich der Bildverarbeitung konkret mit sog. Convolutional Neural Networks (deutsch: *faltendes neuronale Netzwerke*). [24]

[20] TensorFlow [2018a]
[21] Otto Geißler and Nico Litzel [2018]
[22] TensorFlow [2017a]
[23] TensorFlow [2017b]
[24] TensorFlow [2018b]

4.1 Einordnung

Der Begriff der neuronalen Netze kommt eigentlich aus der Biologie und beschreibt eine Ansammlung einzelner Neuronen, die eine Netzarchitektur bilden, die aus verschiedenen Schichten besteht. Im Kontext des Machine Learning spricht man allerdings von künstlichen neuronalen Netzen. [25] Convolutional Neural Networks sind dabei eine spezialisierte Art künstlicher neuronaler Netze. Das Konzept der Convolutional Neural Networks wurde bereits 1989 von LeCun et al. zur Erkennung der menschlichen Handschrift entwickelt. [26]

4.2 Aufbau und Funktionsweise

Convolutional Neural Networks bestehen grundsätzlich aus drei verschiedenen Schichten: convolutional layers, pooling layers und fully connected layers. Dabei wird ein Convolutional Neural Network zunächst aus einer Aneinanderreihung von Convolution-Modulen zusammengesetzt. Ein solches Modul besteht dabei jeweils aus einer Convolution Schicht gefolgt von einer Pooling Schicht. Beim Entwickeln eines Convolutional Neural Networks können beliebig viele Convolution-Module aneinandergereiht werden. Auf das letzte Convolution-Modul folgt dann eine oder mehrere Fully Connected Schicht(en). [27] [28] Die folgende Abbildung zeigt ein beispielhaftes Convolutional Neural Network:

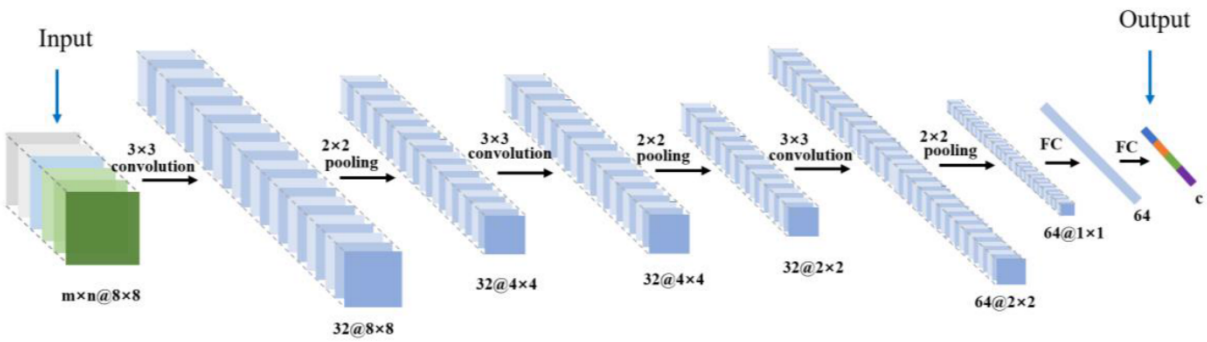


Abbildung 1: Beispielhaftes Convolutional Neural Network. Quelle: [Ji et al., 2018, S. 6]

Innerhalb der Convolution Schichten findet die Convolution (deutsch: *Faltung*) statt. Input einer Convolution Schicht ist ein Bild. Auf dieses Bild werden dann sog. Filter angewendet. [29] Filter dienen dazu Merkmale und Muster bzw. Features in dem Eingabebild zu erkennen. Bei der Entwicklung eines Convolutional Neural Networks kann der Entwickler für jede Convolution festlegen, wie viele Filter angewendet werden sollen. Dabei werden

[25] Julian Moeser [2017]
[26] [Christ, 2017, S. 14]
[27] [Habibi Aghdam and Jahani Heravi, 2017, S. 102]
[28] TensorFlow [2018b]
[29] TensorFlow [2018b]

verschiedene Filter verwendet, um verschiedene Features zu ermitteln. So kann es bspw. einen Filter zum ermitteln der horizontalen Linien in einem Bild geben. Beim Anwenden eines Filters auf ein Bild wird dabei für jeden Pixel des Eingangsbildes eine vom Filter abhängige mathematische Operation durchgeführt. Für jeden Filter wird dabei ein neues Bild generiert, das heißt, das Eingangsbild wird nicht verändert, sondern das Ergebnis der Berechnung jedes Pixels auf ein neues Bild übertragen, das entsprechend die selbe Größe wie das Eingangsbild besitzt. [30] Die folgende Abbildung zeigt ein Eingangsbild vor einer Convolution und die durch die Anwendung verschiedener Filter entstandenen Bilder:

Innerhalb der Pooling Schichten findet das Pooling (deutsch: Bündelung) statt. Eine Pooling Schicht folgt auf eine Convolution Schicht und nimmt die Ausgabebilder dieser (nach Anwendung der Filter) als Input. Beim Pooling wird das sog. Downsampling angewendet. Hierbei wird ein bestimmter Pixelbereich der jeweiligen Eingangsbilder zu einem Pixel zusammengefasst und in einem neuen Bild gespeichert. Dadurch soll die Größe der Bilder verringert und sich auf die wesentlichen Merkmale der Bilder beschränkt werden. Es gibt verschiedene Methoden des Poolings, allerdings hat sich herausgestellt, dass das sog. Max-Pooling die besten Ergebnisse liefert. Bei dieser Methode werden z. B. alle Pixel aller Eingangsbilder in 2 x 2 Pixel-Blöcken betrachtet und der Pixel mit dem höchsten Farbwert in das neue, nun kleinere Bild übertragen. [31] Die folgende Abbildung zeigt, wie das Max-Pooling aus einem Eingabebild der Größe 32 x 32 Pixel ein Ausgabebild der Größe 16 x 16 Pixel macht:

In einer Fully-Connected Schicht werden alle Ausgaben einer Schicht mit allen Ausgaben der nächsten Schicht, das heißt alle erkannten Features einer Schicht mit allen erkannten Features der nächsten Schicht verbunden. Auf diese Weise entstehen kombinierte Features, die für di letztliche Klassifikation von größerer Relevanz sein können. [32] Am Ende der letzten Fully-Connected Schicht wird meist die sog. Softmax Funktion angwendet. Diese vergleicht die aus dem Eingangsbild extrahierten Features mit den Features aller bereits bekannten Klassen. Anhand dieses Abgleichs wird dann für jede bereits bekannte Klasse eine Wahrscheinlichkeit berechnet, dass diese in dem Bild abgebildet ist. Die Summe der Wahrscheinlichkeiten aller Klassen ergibt den Wert 1. [33] [34] [35]

Nun kann ein eigenes Convolutional Neural Network entwickelt werden, wobei beliebig viele Schichten hintereinander geordnet werden können. Gibt man nun ein Bild eines Objektes in das Convolutional Neural Network rein, wird eine erste Convolution + Pooling Phase auf dieses angewendet. Der Output dieser Phase dient dann als Input für die nächste

[30] [Habibi Aghdam and Jahani Heravi, 2017, S. 90–91]
[31] Habibi Aghdam and Jahani Heravi [2017]
[32] Ujjwal Karn [2016]
[33] [Brinkmann, 2017, S. 14–15]
[34] Ujjwal Karn [2016]
[35] TensorFlow [2018b]

Abbildung 2: Anwendung von Filtern auf ein Bild

Phase. Dies geht solange weiter, bis alle definierten Convolution + Pooling Phasen durchlaufen sind. Dann folgt ein oder mehrere Fully Connected Phasen bis letztlich mittels der Softmax Funktion eine Wahrscheinlichkeit für jede bekannte Klasse errechnet wird. Wie bereits erwähnt, kann bei der Entwicklung eines Convolutional Neural Networks die Anzahl der Filter jeder Convolution ausgewählt werden. Welche Filter letztlich angewendet werden entscheidet dieses allerdings selbst. Anhand der aus den vorherigen Phasen extrahierten Features stellt das Network bereits im Verlauf der Convolution + Pooling Phasen

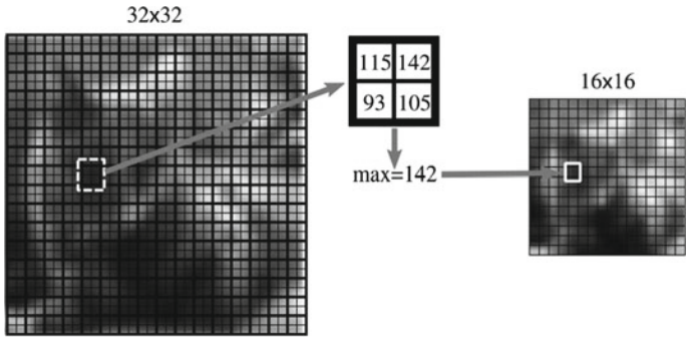


Abbildung 3: Beispiel eines Max-Poolings. Quelle: [Habibi Aghdam and Jahani Heravi, 2017, S. 96]

Vermutungen an, was auf dem Bild abgebildet ist und passt die Filter entsprechend dieser Vermutungen an, um so andere bereits gelernte Features durch Filter, die diese abbilden, zu extrahieren. [36] [37] Die Filter werden dabei mit jeder Convolution + Pooling komplexer. So wird anfangs mit einfachen Filtern nach sog. Low-Level Features (z. B. Gerade, Kurven, Ecken, ...) und später mit komplexeren Filtern nach High-Level Features (z. B. Gesichter, Hände, ...) gesucht. [38] [39] In der folgenden Abbildung ist dargestellt, wie angewendete Filter mit jeder Phase komplexer werden:

4.3 Image Retraining und Image Recognitioning

Um das Convolutional Neural Network anzulernen (Image Retraining), wird ein Training Set an Bildern bereitgestellt, wobei dem Network mitgeteilt wird, welche Klassen auf welchem Bild zu sehen ist. Jedes Bild durchläuft dann die vorgesehenen Convolution + Pooling Phasen, in denen das Network mittels ausgewählter Filter die Features aus dem Bild extrahiert und diese erkannten Features der jeweiligen Klasse zuordnet. Je mehr Bilder derselben Klasse das Network verarbeitet, desto feiner und differenzierter werden die Features der Klasse. Durch Bilder anderer Klassen werden die Features zudem von denen anderer Klassen weiter abgegrenzt. Soll nun Image Recognitioning mit dem angelernten Convolutional Neural Network betrieben werden, kann man diesem ein Bild als Input geben, ohne diesem mitzuteilen, was auf diesem Bild abgebildet. Ist mittels der nun bekannten Features der bekannten Klassen werden entsprechende Filter angewendet und entsprechende Features extrahiert. Am Ende werden dann, wie bereits beschrieben, die extrahierten Features mit denen der bekannten Klassen verglichen und für jede Klasse eine Wahrscheinlichkeit berechnet.

[36] Ujjwal Karn [2016]
[37] [Habibi Aghdam and Jahani Heravi, 2017, S. 90–91]
[38] [Takarli et al., 2016, S. 95]
[39] Ujjwal Karn [2016]

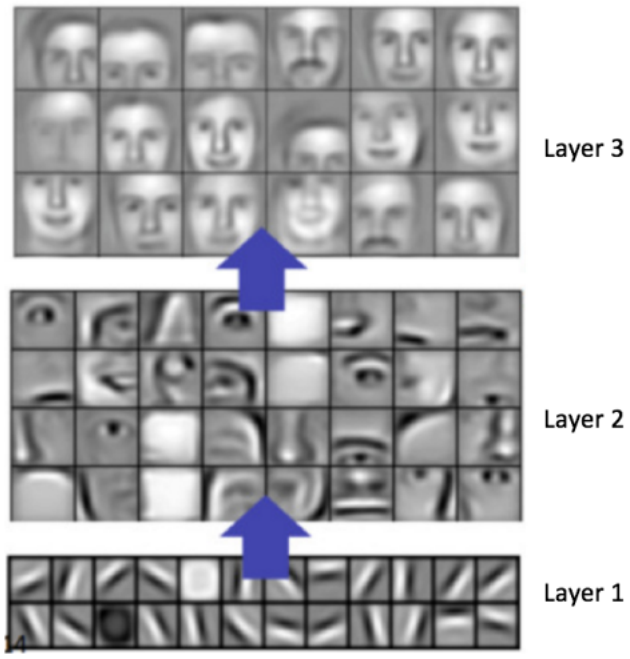


Abbildung 4: Angewendete Filter verschiedener Phasen

5 Image Recognition mit TensorFlow

In den nachfolgenden Kapitel soll mit TensorFlow ein Datenmodell retrainiert werden. Anschließend sollen beispielhaft einige Bilder klassifiziert werden. Um diese Punkte durchzuführen zu können, wird zunächst die Installation von TensorFlow auf einem Ubuntu 16.04 LTS installiert werden.

5.1 Installation

Zunächst müssen alle Voraussetzungen für die Ausführung von TensorFlow vorbereitet werden. Hierfür muss Python[40] 2.7 oder 3.x installiert werden. Unter Ubuntu kann ein Anwendung einfach über die Konsole aus der Packetverwaltung[41] installiert werden. Hierfür muss zunächst die Konsole mit der Tastenkombination „STRG+ALT+T“ geöffnet werden und anschließend der entsprechende Befehl für Python 2.7 oder 3.x ausgeführt werden.

```
1 sudo apt-get install python-pip python-dev # Python 2.7
2 sudo apt-get install python3-pip python3-dev # Python 3.x
```

[40] https://www.python.org/
[41] https://wiki.ubuntuusers.de/Paketverwaltung/

Anschließend kann TensorFlow über die Python-Paketverwaltung installiert werden. Hierfür muss einer der folgenden Befehle, je nachdem ob Python 2.7 oder 3.x verwendet werden soll, ausgeführt werden.

```
pip install tensorflow      # Python 2.7
pip3 install tensorflow     # Python 3.x
```

Nachdem TensorFlow installiert ist, kann die Installation mithilfe einem kleinen Beispielprogramm getestet werden. Hierfür muss zunächst die interaktive Python-Eingabeaufforderung gestartet werden.

```
python
```

Anschließend kann das nachfolgende Programm zur Überprüfung in die Eingabeaufforderungen eingegeben werden.

```
import tensorflow as tf
hallo = tf.constant('Hallo, TensorFlow!')
session = tf.Session()
print(session.run(hallo))
```

Nachdem letzten Befehl sollte das Programm „Hallo, TensorFlow!“ in der Eingabeaufforderung ausgeben.

5.2 Image Retraining

Im nachfolgenden Kapitel soll eine neue Bildklasse angelernt werden. Dafür wird mit dem vor trainierten Inception V3 Modell eine neue Bildkategorie antrainiert. Hierfür wird die beigefügte virtuelle Maschine verwendet. Das Passwort für den angelegten Benutzer der virtuellen Maschine lautet „wert55wert“. Innerhalb des Heimatverzeichnisse befinden sich die folgende Verzeichnis-Struktur.

Abbildung 5: Verzeichnisstruktur Virtuelle Maschine

Alle benötigten Dateien befinden sich in dem „retrain“-Verzeichnis. In diesem Verzeichnis befindet sich einerseits das Python-Programm zum Retraining der neuen Klassen und

ebenfalls die neu anzulernenden Klassen im „images“-Verzeichnis. Der Verzeichnisname beschreibt hierbei den Klassennamen und die darunterliegenden Dateien werden für das Retraining der Klasse verwendet.

Abbildung 6: Verzeichnisstruktur „retrain“

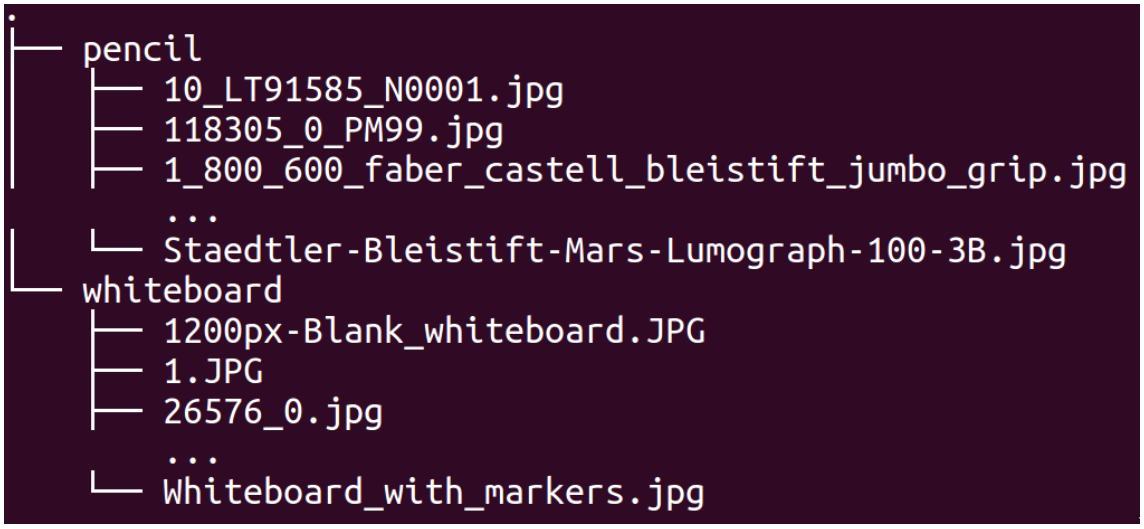


Abbildung 7: Verzeichnisstruktur der Klassen

Anschließend kann das Programm „retrain.py“ mit dem folgenden Befehl ausgeführt werden.

```
1 cd ~/tensorflow/retrain/
2 python retrain.py --image_dir=./images
```

Anschließend lädt das Programm das Inception V3 Modell herunter und führt das Retraining durch.

Die Anwendung erstellt nun für jede vorhandene Datei eine sogenannte Bottleneckdatei. Hierbei handelt es sich um den vorfinalen Layer. Nachdem für jede Bilddatei eine Bottleneckdatei angelegt ist, wird das eigentliche Retraining durchgeführt. Die Ausgabe der Konsole sollte wie in Abbildung 9 dargestellt aussehen.

```
user@user-VirtualBox:~/tensorflow/retrain$ python retrain.py --image_dir=./images
INFO:tensorflow:Looking for images in 'whiteboard'
INFO:tensorflow:Looking for images in 'pencil'
>> Downloading inception-2015-12-05.tgz 100.0%
INFO:tensorflow:Successfully downloaded inception-2015-12-05.tgz 88931400 bytes.
Extracting file from  /tmp/imagenet/inception-2015-12-05.tgz
Model path:  /tmp/imagenet/classify_image_graph_def.pb
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/pencil/Staedtler-Bleistift-Mars-Lumograph-100-3
B.jpg_inception_v3.txt
2018-04-02 20:27:18.855789: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNorm
alization is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_normalization().
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/pencil/STABILO_246GK_SPP_150dpi.jpg_inception_v
3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/pencil/1_800_600_faber_castell_bleistift_jumbo_
grip.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/pencil/faber-castell-bleistift-sparkle-grÕn-118
316-245-800x600px.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/pencil/Bleistift.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/pencil/sharp-pencil-1330506.jpg_inception_v3.tx
```

Abbildung 8: Retraining Konsolenausgaben: Bottleneck-Erstellung

```
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/whiteboard/mob-whiteboard-10-groessen-wae-1-
x300.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/whiteboard/images (1).jpg_inception_v3.txt
INFO:tensorflow:2018-04-02 20:27:49.845449: Step 0: Train accuracy = 100.0%
INFO:tensorflow:2018-04-02 20:27:49.852939: Step 0: Cross entropy = 0.476542
INFO:tensorflow:2018-04-02 20:27:50.283181: Step 0: Validation accuracy = 91.0% (N=100)
INFO:tensorflow:2018-04-02 20:27:51.018774: Step 10: Train accuracy = 100.0%
INFO:tensorflow:2018-04-02 20:27:51.018977: Step 10: Cross entropy = 0.149355
INFO:tensorflow:2018-04-02 20:27:51.114518: Step 10: Validation accuracy = 100.0% (N=100)
INFO:tensorflow:2018-04-02 20:27:51.811864: Step 20: Train accuracy = 100.0%
INFO:tensorflow:2018-04-02 20:27:51.812099: Step 20: Cross entropy = 0.083324
INFO:tensorflow:2018-04-02 20:27:51.902928: Step 20: Validation accuracy = 100.0% (N=100)
INFO:tensorflow:2018-04-02 20:27:52.589885: Step 30: Train accuracy = 100.0%
INFO:tensorflow:2018-04-02 20:27:52.590180: Step 30: Cross entropy = 0.060215
```

Abbildung 9: Retraining Konsolenausgaben: Retraining

Anschließend wird die Graph-Datei erstellt und die Bilderklassen wurden erfolgreich antrainiert.

```
Model path:  /tmp/imagenet/classify_image_graph_def.pb
INFO:tensorflow:Restoring parameters from /tmp/_retrain_checkpoint
INFO:tensorflow:Final test accuracy = 100.0% (N=3)
Model path:  /tmp/imagenet/classify_image_graph_def.pb
INFO:tensorflow:Restoring parameters from /tmp/_retrain_checkpoint
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.
Model path:  /tmp/imagenet/classify_image_graph_def.pb
INFO:tensorflow:Restoring parameters from /tmp/_retrain_checkpoint
INFO:tensorflow:No assets to save.
INFO:tensorflow:No assets to write.
INFO:tensorflow:SavedModel written to: /tmp/saved_models/1/saved_model.pb
```

Abbildung 10: Retraining Konsolenausgaben: Graphdatei-Erstellung

5.3 Image Recognition

Nachdem die neuen Klassen antrainiert wurden, kann nun anschließend mit der Klassifizierung von Bildern begonnen werden. Hierfür werden die Dateien aus dem Verzeichnis „recognition“ benötigt.

```
user@user-VirtualBox:~/tensorflow/recognition$ ll
insgesamt 96
drwxrwxr-x 3 user user  4096 Mär 19 12:09 ./
drwxrwxr-x 5 user user  4096 Mär 19 06:49 ../
drwxrwxr-x 2 user user  4096 Mär 19 06:40 data/
-rw-rw-r-- 1 user user  2758 Mär 18 16:36 img_bleistift.jpg
-rw-rw-r-- 1 user user 54159 Mär 19 12:09 img_lion.jpg
-rw-rw-r-- 1 user user 13991 Mär 18 16:58 img_whiteboard.jpg
-rw-rw-r-- 1 user user  4614 Mär 19 06:41 label_image.py
user@user-VirtualBox:~/tensorflow/recognition$
```

Abbildung 11: Recognition Verzeichnisstruktur

Um nun die neu trainierten Klassen zu verwenden müssen die Parameter bei Programmstart übergeben werden. Zunächst wird die vortrainierten Klassen getestet. Hierfür muss der folgende Befehl ausgeführt werden.

```
1 cd ~/tensorflow/recognition/
2 python label_image.py --image=./img_whiteboard.jpg
```

Die Anwendung sollte das Bild des Whiteboards nicht erkennen, da die Whiteboard Klasse nicht in dem Inception V3 Modell vorhanden ist. Die Ausgabe sollte wie in Abbildung 12 dargestellt aussehen.

```
user@user-VirtualBox:~/tensorflow/recognition$ python label_image.py --image=img_whiteboard.jpg
binder 0.17277507
envelope 0.044833787
fire screen 0.041727718
tray 0.03844816
cleaver 0.026645396
user@user-VirtualBox:~/tensorflow/recognition$
```

Abbildung 12: Recognition: Klasse nicht antrainiert

Anschließend wird mit dem folgenden Befehl ein Bild eines Löwen klassifiziert.

```
1 cd ~/tensorflow/recognition/
2 python label_image.py --image=./img_lion.jpg
```

In diesem Beispiel sollte das neuronale Netz das Bild Klassifizieren können (Konsolenausgabe ist in Abbildung 13 dargestellt).

```
user@user-VirtualBox:~/tensorflow/recognition$ python label_image.py --image=img_lion.jpg
lion 0.92815274
cheetah 0.011404002
leopard 0.0012407672
hartebeest 0.0009785484
cougar 0.00078992196
user@user-VirtualBox:~/tensorflow/recognition$
```

Abbildung 13: Recognition: Klasse antrainiert

Zuletzt wird das in Kapitel 5.2 antrainierte Modell verwendet werden, sodass das Bild des Whiteboard erkannt werden kann. Hierfür müssen die Parameter der „label_image“-Anwendung angepasst werden.

```
1 cd ~/tensorflow/recognition/
2 python label_image.py --graph=/tmp/output_graph.pb --labels=output_labels --input_layer=
      Mul --output_layer=final_result --image=./img_whiteboard.jpg
```

Nachdem dieser Befehl ausgeführt ist, sollte die Konsole das Whiteboard klassifizieren können.

```
user@user-VirtualBox:~/tensorflow/recognition$ py
els.txt --input_layer=Mul --output_layer=final_re

2018-04-02 21:09:00.925198: W tensorflow/core/fra
ted. It will cease to work in GraphDef version 9.
whiteboard 0.99951065
pencil 0.00048932043
user@user-VirtualBox:~/tensorflow/recognition$
user@user-VirtualBox:~/tensorflow/recognition$
```

Abbildung 14: Recognition: Whiteboard erkannt

Der Parameter „–graph“ steht für die Graph-Datei des trainierten neuronalen Netzwerks. Die Labels-Datei steht für die Klassen des Modells. Der Parameter „–input_layers“ definiert den Eingabe-Layer und „–output_layers“ den letzten Layer.

6 Kritische Reflexion

Es ist zu beachten, dass, auch wenn das Gebiet der Convolutional Neural Networks bereits 1989 erforscht wurde, hat das Thema der Bildverarbeitung erst in den letzten Jahren

an großer Bedeutung gewonnen. Entsprechend ist das Thema sehr aktuell und wird momentan weiter erforscht, sodass die Literatur in diesem Themenbereich sich fortwährend ändert und somit die weitere Entwicklung beobachtet werden sollte. Des Weiteren ist anzumerken, dass die Funktionsweise des Convolutional Neural Networks, seiner Schichten und deren Mittel wie bspw. der Filter innerhalb dieser Arbeit nur oberflächlich dargestellt werden kann. So beruhen diese letztlich auf komplexen mathematischen Berechnungen, die bei Vertiefung dieses Themas betrachtet werden sollten. Außerdem ist zu erwähnen, dass das Convolutional Neural Network lediglich eine Variante zur Bildverarbeitung darstellt und hier als solche vorgestellt wird, da diese im vorgestellten Framework TensorFlow Verwendung findet.

7 Fazit

Letztlich lässt sich sagen, dass TensorFlow eine geeignete Möglichkeit für das Image Recognitioning darstellt. Die Verwendung fertiger Skripte macht eine schnelle Anwendung möglich und ist aufgrund der einfachen Verwendung auch schnell erlernbar. Durch die mitgelieferten Funktionen ist auch die eigene Entwicklung eines Convolutional Neural Networks problemlos möglich und dadurch, dass TensorFlow Open-Source ist, können auch bestehende Funktionen nach Belieben angepasst werden. TensorFlow macht es so möglich Machine Learning vor allem mit Hinblick auf Bildverarbeitung trotz komplexer mathematischer Theorie einfach verwenden zu können. Auch wenn das Convolutional Neural Network bereits eine sehr ausgereifte Art der Bildverarbeitung darstellt, wird dennoch weiterhin an neuen und effizienteren Methoden geforscht und auch versucht bestehende Möglichkeiten wie das Convolutional Neural Network stets weiterzuentwickeln, weshalb Interessierte sich in diesem sehr aktuellen Bereich des Image Recognitionings stets weiter informieren sollten.

Literaturverzeichnis

Nils Brinkmann. Objekterkennung in natürlichen szenen mittels region-based convolutional neural networks, 2017. URL http://patrec.cs.tu-dortmund.de/pubs/theses/ba_brinkmann.pdf.

Wilhelm Burger and Mark James Burge. *Digitale Bildverarbeitung: Eine algorithmische Einführung mit Java*. X.media.press. Springer Vieweg, Berlin, 3., vollst. überarb. und erw. aufl. edition, 2015. ISBN 978-3-642-04603-2. doi: 10.1007/978-3-642-04604-9. URL http://ebooks.ciando.com/book/index.cfm/bok_id/319965.

Patrick Ferdinand Christ. Convolutional neural networks for classification and segmentation of medical images, 2017. URL https://mediatum.ub.tum.de/doc/1364397/1364397.pdf.

Hamed Habibi Aghdam and Elnaz Jahani Heravi. *Guide to convolutional neural networks: A Practical Application to Traffic-Sign Detection and Classification*. Springer, Cham, 2017. ISBN 978-3-319-57549-0. doi: \url{10.1007/978-3-319-57550-6}. URL http://dx.doi.org/10.1007/978-3-319-57550-6.

Shunping Ji, Chi Zhang, Anjian Xu, Yun Shi, and Yulin Duan. 3d convolutional neural networks for crop classification with multi-temporal remote sensing images. *Remote Sensing*, 10(2):75, 2018. ISSN 2072-4292. doi: 10.3390/rs10010075.

Julian Moeser. Künstliche neuronale netze – aufbau & funktionsweise, 2017. URL https://jaai.de/kuenstliche-neuronale-netze-aufbau-funktion-291/.

Fakhri Karray, Aurélio Campilho, and Farida Cheriet, editors. *Image analysis and recognition: 14th International Conference, ICIAR 2017, Montreal, QC, Canada, July 5–7, 2017 : proceedings*, volume 10317 of *Lecture Notes in Computer Science*. Springer, Cham, 2017. ISBN 978-3-319-59875-8. URL http://www.springer.com/.

Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, 1998. ISSN 09205691. doi: 10.1023/A:1008045108935.

Marco Varone. Machine learning for big data analytics. URL http://www.expertsystem.com/machine-learning-big-data-analytics/.

Margaret Rouse. image recognition, 2017. URL http://whatis.techtarget.com/definition/image-recognition.

Molly Galetto. Machine learning and big data analytics: the perfect marriage, 2018. URL https://www.ngdata.com/machine-learning-and-big-data-analytics-the-perfect-marriage/.

Otto Geißler and Nico Litzel. So funktioniert google tensorflow, 2018. URL https://www.bigdata-insider.de/so-funktioniert-google-tensorflow-a-669777/.

Satya Mallick. Image recognition and object detection, 2016. URL https://www.learnopencv.com/image-recognition-and-object-detection-part1/.

Thiago Christiano Silva and Liang Zhao. *Machine learning in complex networks*. Springer, Cham and Heidelberg and New York and Dordrecht and London, 2016. ISBN 978-3-319-17289-7. URL http://lib.myilibrary.com/detail.asp?id=893137.

Fariba Takarli, Ali Aghagolzadeh, and Hadi Seyedarabi. Combination of high-level features with low-level features for detection of pedestrian. *Signal, Image and Video Processing*, 10(1):93–101, 2016. ISSN 1863-1703. doi: 10.1007/s11760-014-0706-8.

TensorFlow. Image retraining, 2017a. URL https://www.tensorflow.org/tutorials/image_retraining.

TensorFlow. Image recognition, 2017b. URL https://www.tensorflow.org/tutorials/image_recognition.

TensorFlow. Tensorflow, 2018a. URL https://www.tensorflow.org/.

TensorFlow. A guide to tf layers: Building a convolutional neural network, 2018b. URL https://www.tensorflow.org/tutorials/layers.

Ujjwal Karn. An intuitive explanation of convolutional neural networks, 2016. URL https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.

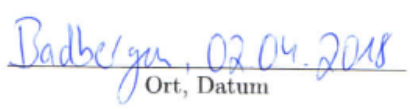
Vipul Jain. Image recognition vs object detection: the difference, 2018. URL https://hackernoon.com/micro-learnings-image-classification-vs-object-detection-the-difference-77110b592343.

Eidesstattliche Erklärung

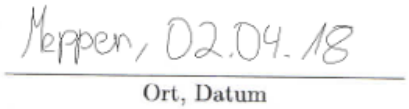
Wir erklären an Eides statt, dass wir unsere Hausarbeit

„Image Recognition mit TensorFlow“

selbstständig und ohne fremde Hilfe angefertigt haben und dass wir alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlehnenden Ausführungen unserer Arbeit besonders gekennzeichnet und die Quellen zitiert haben.


Ort, Datum


Unterschrift


Ort, Datum


Unterschrift