HOCHSCHULE HEILBRONN

COMPUTER & ROBOT VISION (PRÜFUNGSNUMMER: 135031)

BRIEFMARKENERKENNUNG

GEORG JAHN, MARTIN HAAG*

31. Januar 2022

Eingereicht bei PROF. DR. DIETER MAIER

*194985, 194980, gjahn@stud.hs-heilbronn.de,mahaag@stud.hs-heilbronn.de

Inhaltsverzeichnis

Abkürzungs- Fremdwort und Fachbegriffsverzeichnis

NN: neuronales Netz(-werk)

CNN: convolutional neuronal network

Transfer Learning: Technik, bei der ein bestehendes NN auf ein anders, vergleichbares Problem angewandt wird und dadurch nur wenig trainiert werden muss

BGR/RGB-Bild: Bild mit den Farbkanälen in dieser Reihenfolge

Otsu-Methode: Methode zur automatischen Festlegung eines Schwellwertes zur Binarisierung auf Grundlage des Histogrammes

morphologischer Filter: Filter, die im Stande sind, Strukturen von Bildern gezielt zu beeinflussen [1]

PAP: Programmablaufplan

Supervised Learning: Lernvorgang anhand eines Datensatzes, bei dem das Musterergebnis bekannt ist

Relu6: Rectified Linear Unit, eine nicht lineare Aktivierungsfunktion

Adam-Optimizer: eine Methode zur iterativen Anpassung der Gewichte in einem NN

Epoche: ein ganzer Trainingsdurchgang durch alle Trainingsdaten

Forwardpass: Berechnung eines Outputs eines NN anhand eines Inputs

Data Augmentation: Erweiterung eines Datensatzes durch Abänderung bestehender Daten

1 Einleitung

1.1 Motivation

Bildverarbeitung ist kein neues Feld der Forschung mehr und auch neuronale Netzwerke sind bereits seit vielen Jahren Subjekt von ständiger Weiterentwicklung. Da diese beiden Fachgebiete durchaus gut miteinander harmonieren, haben wir uns entschieden, eine Projektaufgabe zu wählen, die diese Beiden Gebiete fusioniert. Im Rahmen dieses Computer & Robot Vision-Projektes haben wir deshalb die Unterscheidung von Briefmarken in gestempelte und ungestempelte automatisiert mithilfe einer "künstlichen Intelligenz" durchgeführt. Wenn auch unsere Arbeit sehr unwahrscheinlich kommerzielle Anwendung findet, ist es doch geeignet, um an einem Tag der offenen Tür oder einem "studieren probieren" Event demonstriert zu werden.

1.2 Ziel der Arbeit

Das Ziel der Arbeit war es, mithilfe klassischer Bildverarbeitung und eines neuronalen Netzwerkes (NN), gestempelte Briefmarken von ungestempelten zu unterscheiden. Ursprünglich war der Plan, mit Bildern von ganzen Briefen zu arbeiten. Dadurch würde der Stempel über den Briefmarken-Rand überlappen, was den Vorteil hätte, dass das Netzwerk deutlich robuster sein könnte. Aufgrund der geltenden Datenschutzgesetze und dem mittlerweile hohen Anteil an digital gestempelten Briefen waren wir jedoch nicht in der Lage, einen ausreichend großen Datensatz zusammen zu tragen. Deshalb haben wir uns entschieden eine Briefmarkensammlung zu digitalisieren. Dafür wurden die einzelnen Briefmarken auf einen dunklen Untergrund gelegt und mit einem Handy fotografiert. Dieser selbst generierte Datensatz ist zwar auch nicht wirklich groß genug um aussagekräftige beziehungsweise robuste Trainingserfolge mit einem NN zu erzielen, für einen "Proof of Concept" reicht die Bildersammlung allerdings aus.

1.3 Vorgehensweise

Unser Vorgehen teilt sich in zwei Teilschritte auf:

- Klassische Bildverarbeitung:
  Die Roh-Bilder sollen so vorverarbeitet werden, dass sie als Input für das NN dienen können. Dafür wird ein quadratisches Bildverhältnis benötigt. Für diesen Schritt sollen Funktionen und Werkzeuge aus der OpenCV-Programmbibliothek verwendet werden. Dazu zählen die helligkeitsbasierte Binarisierung, die morphologischen Filterung, die Drehung des Bildes und schlussendlich die Skalierung auf ein durch das NN bestimmte Seitenverhältnis. Das Endergebnis dieser Bildverarbeitungsschritte kristallisiert sich ein rechteckiger Bildausschnitt heraus, der nur noch die Briefmarke enthält.

- Klassifizierung mit einem NN:
  In diesem Bereich der Arbeit wird der vorverarbeitete Datensatz für das Training und die Evaluierung des NN noch einmal weiter aufbereitet, denn Neuronale Netze erwarten als Eingabe immer ein Bild mit konstanter Größe. Außerdem wird hier das NN selbst initialisiert. Da wir die Technik des "Transfer Learnings" genutzt haben, konnte hierfür ein vortrainiertes Modell herausgesucht werden, bei dem nur die letzte Ebene - das sogenannte Outputlayer - ausgetauscht werden soll.

2 Klassische Bildverarbeitung

2.1 Übersicht

Das folgende Schaubild zeigt, welche Schritte vorgenommen werden und in welcher Reihenfolge. Dafür wurde ein Beispielbild gewählt, bei dem keine Komplikationen auftreten. Welche Probleme auftreten können wird im nachfolgenden Kapitel 2.3 erläutert. Auf die einzelnen Schritte wird im folgenden Kapitel 2.2 genauer eingegangen. Eine Übersicht über den Ablauf der Bilddatenverarbeitung ist dem Kapitel 2.4 zu entnehmen.

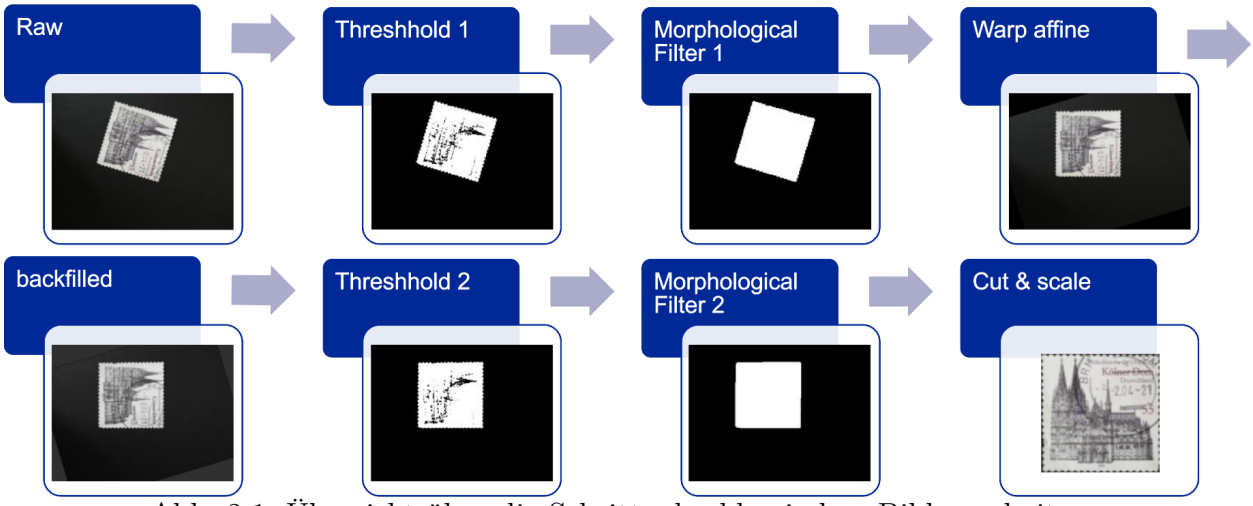

Abb. 2.1: Übersicht über die Schritte der klassischen Bildverarbeitung

2.2 Einzelbeispiel

Im Folgenden werden die Zwischenschritte und der Gesamtablauf der klassischen Bildverarbeitung am Beispiel einer einzelnen Briefmarke aufgezeigt.

Die Bilder wurden mit einer gewöhnlichen Handykamera aufgenommen. Des Weiteren wurde ein mattierter, dunkelgrauer Hintergrund für alle Bilder festgelegt, denn auf einem relativ hellen Hintergrund würde der "Schatten" der Briefmarke nicht ausreichen um die entsprechende Kontur zu finden.
Die beiden Bilder (Abbildung 2.2) sind Eingang und Ausgang des ersten Schrittes (Raw => Threshold1). Das Raw Bild ist dabei ein gewöhnliches RGB-Bild, auf dem die Briefmarke in zufälliger Orientierung abgelichtet ist. Dieses wird zunächst in ein Graustufenbild umgewandelt. Durch eine nachfolgende Otsu-Binarisierung wird der grobe Umriss der Briefmarke auf dem Hintergrund definitiv entdeckt. Je nach Motiv der Briefmarke sind im Binärbild jedoch noch viele "Löcher" beziehungsweise schwarze Stellen. Für die Briefmarkensammlung kann nicht vorhergesehen werden, ob oder wie viele dieser Löcher in diesem Schritt entstehen. Das Ziel dieses Schrittes soll allerdings lediglich das generelle Auffinden (Position, Orientierung) der Briefmarke sein.

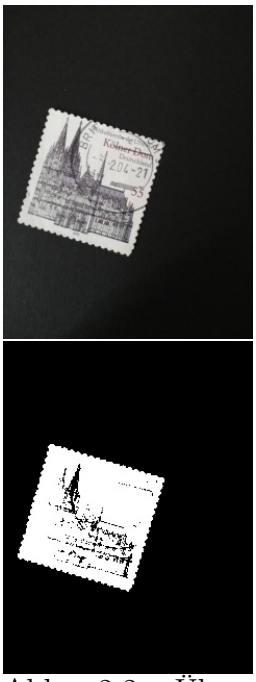

Abb. 2.2: Übergang zu Threshold1

Da zu diesem Zeitpunkt keine Details aus dem Originalbild benötigt werden, bietet es sich an, über das Binärbild einen morphologischen Rechteck-Filter laufen zu lassen um einerseits die Löcher im Inneren zu schließen und andererseits eventuelle Unförmigkeiten der Briefmarke auszugleichen. Dies wird durch eine Dilatation, gefolgt von einer Erosion erreicht. Die Größe der Kernel-Matrix für dieses Verfahren wurde bei einer Auflösung von 240x320 auf 11x11 festgelegt. Die Abbilung 2.3 zeigt das Ergebnis dieses zweiten Schrittes. Hierbei sind keine dunklen Stellen mehr auf der Fläche der Briefmarke erkennbar.

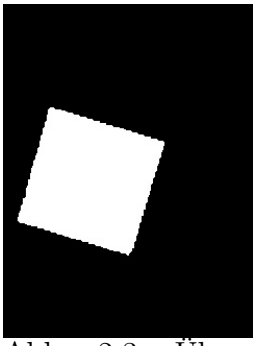

Abb. 2.3: Übergang zu morph. Filter1

Zu diesem Zeitpunkt ist es nun robust möglich die Kontur der Briefmarke zu finden. Für den Fall, dass trotz der morphologischen Filterung noch “Löcher” existieren, wird über Flächen- und Hierarchiebedingungen ausgeschlossen, dass deren Konturen weiter verfolgt werden.
Die Abbildung 2.4 zeigt, wie gut die gefundene Kontur aus dem Binärbild nun die Briefmarke repräsentiert. Der blaue Rahmen aus dem oberen Teil der rechten Abbildung wird mit der opencv Methode minAreaRect aus der Kontur des Binärbildes gewonnen. Über den Winkel dieses Rechteckts kann nun das ursprüngliche Bild am Bildkoordinatensystem ausgerichtet werden. Dies geschieht mithilfe der warpAffine Funktion. Um diese zu nutzen muss vorher noch eine 2D Rotationsmatrix erstellt werden. Dafür werden die Werte des Rechtecks verwendet. Das fertig gedrehte Bild ist im unteren Teil von Abbildung 2.4 zu erkennen. Allerdings bilden sich durch die Rotation leere (schwarze) Bildbereiche. Diese werden wie in Kapitel 2.3 beschrieben, mit dem Mittelwert des gesamten Bildes aufgefüllt.

Abb. 2.4: Rechteck, gerade gedrehtes Bild(warp Affine)

Nun werden die beiden bereits gezeigten Schritte (siehe Abbildungen 2.2 (binarisierung) und 2.3 (morphologische Filterung)) erneut auf das gedrehte Bild angewandt. Diese Redundanz erspart einerseits die benötigte Umrechnung des gefundenen “minAreaRects” auf die Drehachse und andererseits erhöht sich dadurch die allgemeine Fehlerrobustheit des Programms.
Aus dem gerade-gedrehten Bild der Abbildung 2.4 wird nun wieder ein Binärbild erstellt. Wenn man darauf wieder morphologische Filter anwendet, kann ein Binärbild wie in Abbildung 2.5 erreicht werden. Das Rechteck (boundingBox) dieser Kontur ist nun geeignet, um die Briefmarke aus dem gerade gedrehten Farbbild auszuschneiden. Das ausgeschnittene Bildstück hat natürlich von Marke zu Marke andere Größenverhältnisse. Die Briefmarken haben schließlich nicht nur unterschiedliche Formate, sondern sind auch nicht alle aus Derselben Richtung fotografiert worden. Da das NN eine bestimmte Inputgröße erfordert, muss das ausgeschnittene RGB-Bildstück Anschließend auf die Größe des “input-layers” skaliert werden (siehe nachfolgendes Kapitel 3.2).

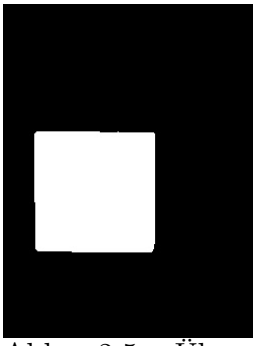

Abb. 2.5: Übergang zu morph. Filter2

2.3 Mögliche Probleme

In diesem Abschnitt sollen aufgetretene Probleme geschildert und die dafür gefundenen Lösungen gezeigt werden.

Wie in der Abbildung 2.6 zu sehen ist, können gerade gedrehte Bilder nach der Binarisierung durch ein ungünstiges Motiv mehrere zureichend große Konturen enthalten. Das Kriterium, welches entscheidet ob die iterierte Kontur eine potenzielle Briefmarke ist, war vorher nur die Größe der von der Kontur eingeschlossenen Fläche. Demnach würden hier drei Bilder für das NN ausgeschnitten werden. Um das zu verhindern, wird hierbei auf eine Hierarchiebedingung zurückgegriffen. Diese lautet, dass der Programmcode nur für Konturen ohne weitere umliegende Konturen (Elternkontur) weiter ausgeführt wird. Für das Beispiel (Abbildung 2.6) fallen deshalb die markierten Konturen (blau & rot) weg und nur die interessanteste (grün) bleibt übrig.

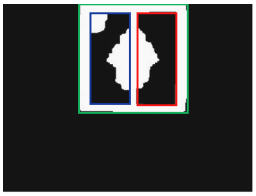

Abb. 2.6: Binarisierung mit drei großen Konturen

Ein weiteres Problem ist in Abbildung 2.7 zu erkennen. Hier sind durch die Rotation große schwarze Flächen beziehungsweise “leere” Bereiche in den Ecken entstanden. Zur Binarisierung wird die Otsu-Methode verwendet, der Schwellwert zur Binarisierung wird also automatisch und abhängig vom Bild und dessen Histogramm festgelegt. Normalerweise ist der Hintergrund dunkler als die Briefmarke. Jedoch wird durch die entstandenen schwarzen Ecken eine neue Histogramm-Klasse für die Zahl 0 gebildet. Dadurch verschiebt sich der optimierte Histogramm-Klassen-Schwellwert den die Otsu Methode liefert. Bei ausreichend großen schwarzen Ecken verschiebt sich der Schwellwert in eine geringere Graustufe, sodass und Teile des Briefmarken-Hintergrundes ebenfalls über die binarisierungsschwelle rutschen (oberer Teil der Abbildung 2.7). Um dieses Problem zu beheben wurden die schwarzen Pixel mit dem Mittelwert der Helligkeit des Originalbildes nach-gefärbt. Dadurch reduziert sich die Verschiebung des Otsu-Schwellwertes und die Binarisierung kann folglich die Briefmarke wieder korrekt herausfiltern.

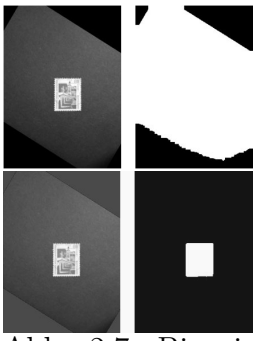

Abb. 2.7: Binarisierungsproblem durch schwarze Ecken

Das letzte Problem, das hier betrachtet werden soll, ist in Abbildung 2.8 zu sehen. Auch wenn es mit dem menschlichen Auge nur schwer zu erkennen ist, ist das Bild sehr ungleichmäßig beleuchtet. Dadurch ist der ursprünglich schwarze Hintergrund in großen Bereichen sogar heller als die eigentliche Briefmarke. Wodurch die Briefmarke gar nicht erst gerade gedreht werden kann. Im Vergleich zum vorher gegangenen Problem entstehen die Komplikationen also nicht erst während der Verarbeitung - wie beispielsweise die schwarzen Ecken - sondern die Ursache dafür ist eine schlechte Qualität rohen Originalbild. Dieser Fall ist jedoch im ganzen Datensatz nur einmal aufgetreten. Da wir entschieden haben, dass dieser Verlust verkraftbar ist, wurden hier keine weiteren Anstrengungen unternommen, um das Programm hierhin gehend weiter anzupassen.

Abb. 2.8: Binärisierungsproblem durch schlechte Lichtverhältnisse

2.4 Programmablaufplan

Der Programmablaufplan (PAP) zeigt, dass das Programm in zwei ineinander geschachtelten Schleifen abläuft. In der äußeren Schleife wird über die Liste der Bilder iteriert. Sollte eine Kontur mit ausreichender Größe durch die Binarisierung entstehen, wird daraufhin in die zweite Schleife übergegangen. In dieser wird über die gefundenen Konturen nach der zweiten Binarisierng und Dilatation iteriert. Zuerst wird dann die Fläche jeder Kontur abgefragt. Sollte die ausreichend sein, ist die nächste Bedingung die Hierarchiebedingung, die das Problem aus 2.6 löst. Wenn beide Bedingungen erfüllt wurden wird der Bildausschnitt dem Benutzer angezeigt. Sollte der bedienende Mensch den Ausschnitt für gut befinden kann er mit der "g"Taste der Tastatur die Briefmarke abspeichern. Mit der "b"Taste werden diverse Zwischenergebnisse in Form von Bilddateien in einen separaten Ordner abgelegt. Dies erleichtert die Fehlersuche, ist aber für die Projektaufgabe irrelevant. Sollte das Bild weder gut sein, noch einen interessanten Fehler hervorrufen, kann mit einer beliebigen Taste mit dem nächsten Bild fortgefahren werden. Zusätzlich zur Anzeige der Bilder gibt eine Konsolenausgabe auch noch Aufschluss über die Anzahl der Schleifendurchläufe (Anzahl bearbeiteter Bilder und Konturen). Durch die beiden Schleifen und vor allem durch das zweimalige Binarisieren ist der Code zwar nicht optimiert, dafür aber gut nachvollziehbar und performant genug (vor allem wegen der guten Effizienz der OpenCV-Bibliothek). Die Rechenzeit für den Durchlauf einer Marke durch das Programm verglichen mit der Reaktionszeit des Bedieners, ist geringer. Dadurch läuft das Programm auch auf kleinen Maschinen relativ flüssig.

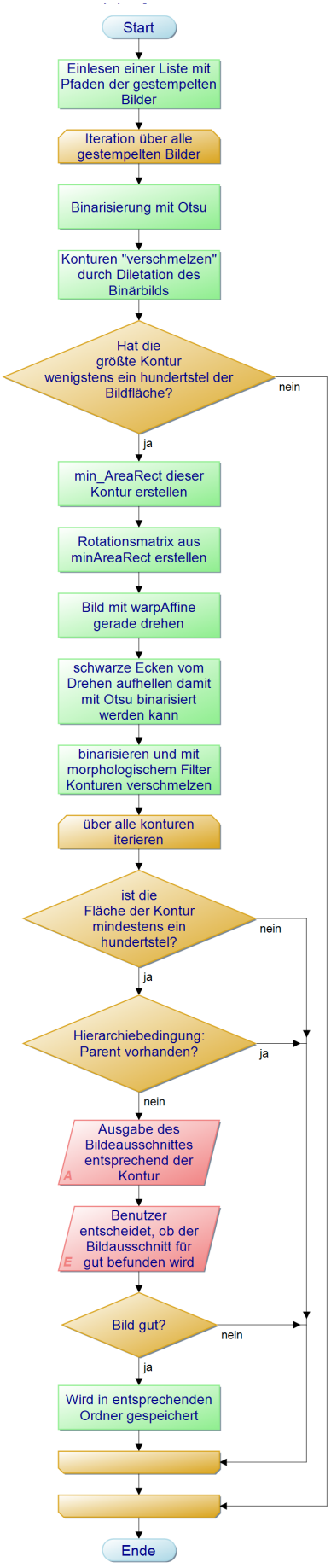

Abb. 2.9: Programmablaufplan Bildverarbeitung

3 Neuronales Netzwerk

Für unsere Arbeit haben wir die Technik des "Transfer Learnig" angewandt. Dabei wird ein Netzwerk, dass ursprünglich auf eine andere, aber ähnliche Aufgabe trainiert wurde, wiederverwendet. Dafür muss meistens nur die Ausgabe angepasst werden. Deshalb müssen auch nur neu hinzugefügten Teile trainiert werden. Dadurch kann sehr viel Rechenaufwand gespart werden. Beim verwendeten NN handelt es sich um das MobileNetV2. Es kann für "Supervised Learning" Probleme zum Klassifizieren von Bildern verwendet werden.

3.1 Architektur und Vorgehen

Beim MobileNetV2 handelt es sich um ein "Convolutional Neuronal Network" (CNN). Diese werden hauptsächlich auf Bilder angewandt, haben aber auch noch andere Anwendungsgebiete. Ein CNN zeichnet sich durch die Schichten, in denen mithilfe von Matrizen durch Faltung unterschiedliche Merkmale der Bilder extrahiert werden, aus. Das MobileNetV2 wurde, wie der Name schon verrät, für die Anwendung auf mobilen Geräten entwickelt. Das bedeutet für Geräte mit niedriger Rechenleistung und ohne performante Grafikkarten wie Smartphones und Tablets.Es ist also eine sehr effektive Architektur nötig. Wie wird diese Effizienz nun aber erreicht?
Die Basis dieses NN ist eine "bottleneck depth-seperable convolution with residuals"¨"depth-seperable" bedeutet, dass nicht alle drei Farbkanäle auf einmal mit einem Tensor der Tiefe drei gefaltet werden, sondern alle Farbkanäle einzeln mit einem flachen Tensor gefaltet. Anschließend werden die Ergebnisse der drei einzelnen Operationen mit einer "Pointwise Convolution" verbunden. Dabei wird mit einem Tensor der Dimension 1x1x3 die Tiefe auf eins reduziert. Mit diesem Vorgehen verliert das Netzwerk, im Gegensatz zum direkten Anwenden eines Tensors der Tiefe drei, zwar Komplexität, aber die benötigten Rechnungen werden auch stark verringert. [2]
"Residual Blocks" Verbinden den Eingang und den Ausgang eins Faltungsblocks mit einer Abkürzung. Durch die Addition dieser Zustände hat das NN Zugriff auf den Zustand vor der Faltung. Für gewöhnlich wird die Tiefe bei solchen Vorgängen immer erst reduziert und dann wieder erhöht. [3] Beim MobileNetV2 wird das jedoch gedreht, da bei der "Depthwise Convolution" schon Parameter gespart wurden. Das "bottleneck" wird mit einer Relu6 Aktivierungsfunktion erreicht. Damit wird die Ausgabe auf einen Bereich von 0-6 limitiert. Diese Nichtlinearität wird in den "bottleneck depth-seperable convolution with residuals" jeweils nach einer Faltung eingefügt, abgesehen von der letzten "Pointwise Convolution" .

Dieses Netzwerk wurde von Google auf einen Datensatz von 1.4 Millionen Bilder aus 1000 Kategorien trainiert. Da sind gestempelte und nicht gestempelte Briefmarken natürlich nicht

Die Abbildung 3.1 stellt eine "bottleneck depth-seperable convolution with residuals" Einheit dar. Sie besteht aus drei Faltungsoperationen(1x1, 3x3, 1x1), die die "Inverted Residuals" Struktur bilden. Insgesamt beinhaltet das Netzwerk nach einer ersten, reinen, Faltungsschicht 19 dieser Blöcke.

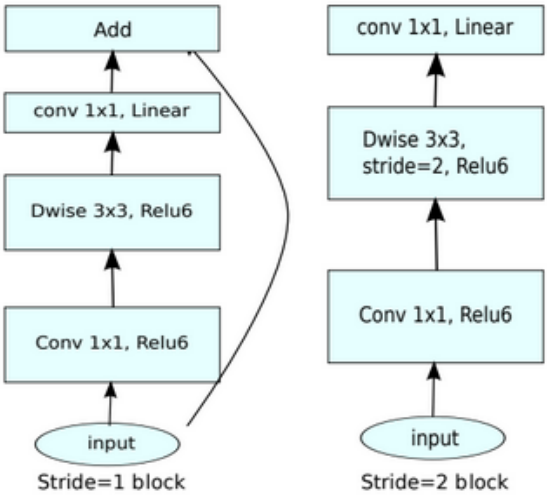

Abb. 3.1: Aufbau eines Blocks [4]

dabei. Jedoch werden auch um diese Kategorien zu differenzieren von NN zuerst Merkmalsextraktion betrieben. Durch das Nutzen eines anderen Outputlayers werden also aus den gleichen Merkmalen, die Google genutzt hat um 1000 Klassen zu unterscheiden, zwischen den zwei beiden Klassen unserer Projektarbeit unterschieden.

3.2 Implementierung und Auswertung

Die Abbildung 3.2 zeigt, welche Schritte zur Implementierung in Python nötig waren. Zuerst ist natürlich wieder das Einlesen der Daten durchzuführen. Mit Hilfe einer assoziativen Liste (Dictionary) werden die Listen, die die Pfade zu den einzelnen Bildern enthalten, den beiden Klassen 0 und 1 zugewiesen. 0 steht in diesem Fall für gestempelt und 1 für ungestempelt. Als Nächstes wird der Datensatz zufällig durchgemischt und dann in Verhältnis eins zu drei aufgeteilt. Der dreiviertelste Teil wird nachher zum Training verwendet. Mit dem kleineren Teil wird die Auswertung des Ergebnisses durchgeführt. Danach werden die Farbwerte aller Kanäle durch 255 geteilt, um den Wertebereich auf 0-1 zu normieren. Abschließend werden die Bilder des Datensatzes auf die Größe 224x224 geändert. Das NN kann schließlich nur mit Bildern gleicher Größe arbeiten. Damit ist die Arbeit am Datensatz abgeschlossen. Als Nächstes muss das vortrainierte Model heruntergeladen werden. An diesem fehlt jedoch noch das Ausgabelayer. Das wird dann noch hinzugefügt. Damit ist das Netzwerk vollständig, nur die Gewichte des Ausgabelayers müssen noch trainiert werden. Das Training erfolgt über fünf Epochen mit dem Adam-Optimizer. Bereits nach diesen fünf Epochen konvergieren die Ergebnisse des NN sehr gut. Damit ist das NN dann bereit, die Bilder des Testdatensatzes zu klassifizieren. Dazu wird zu jedem Bild des Testdatensatzes ein Forwardpass durchgeführt. Hat die richtige Klasse den höheren Wert am Ausgabelayer, war die Klassifizierung erfolgreich. Über die Konsole wird der Benutzer dann informiert, wie das NN abgeschnitten hat. Zusätzlich werden die falsch klassifizierten Bilder dann dem Benutzer angezeigt. Damit kann er dann versuchen nachzuvollziehen, warum es falsch klassifiziert wurde.

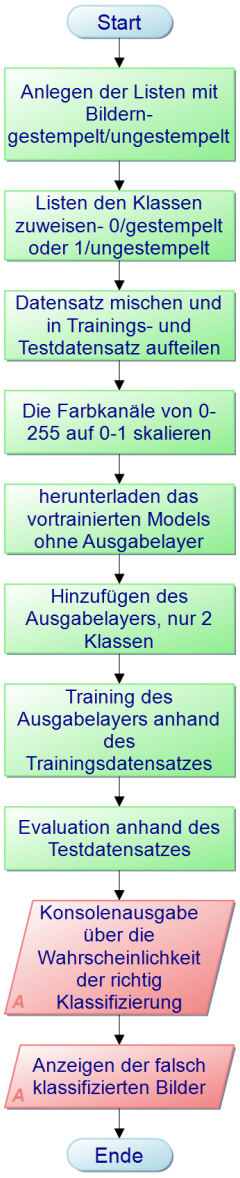

Abb. 3.2: Programmablaufplan NN

Im Diagramm aus Abbildung 3.2 ist zu sehen, wie das NN in zehn Durchläufen abgeschnitten hat. Bei jedem Durchlauf wurden die Daten des Testdatensatzes und des Trainingsdatensatzes neu gemischt. Das Netzwerk wurde damit neu trainiert und evaluiert. Die Daten aus den zehn Durchgängen ergeben einen Mittelwert für die Klassifizierungsrate von 90,5% und eine Standardabweichung von 3,5%.

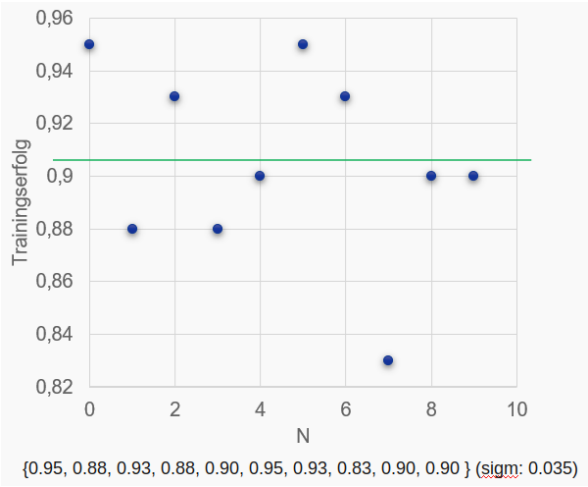

Abb. 3.3: Auswertung von zehn Testläufen

4 Fazit

Abschließend lässt sich sagen, dass das Projekt als Proof of Concept durchaus erfolgreich war. Es konnte gezeigt werden, dass mit dem NN MobilNetV2 mit einer zusätzlichen einfachen Vorverarbeitung Briefmarke in gestempelt und nicht gestempelt unterschieden werden können. Mit etwa 90% ist die Rate an richtigen Klassifizierungen zwar nicht besonders hoch, doch angesichts des sowohl kleinen als auch qualitativ schlechten Datensatzes ist es kein schlechtes Ergebnis.
Eine Möglichkeit zum Verbessern des Datensatzes wäre die Technik der "Data Augmentation" gewesen. Dabei werden aus den vorhandenen Daten weitere Daten durch Operationen wie Spiegelung, Drehung, Helligkeitsänderung (et cetera) generiert und der Datensatz so erweitert.
Das Verbesserungspotenzial liegt jedoch nicht nur beim Datensatz, wenn auch das wahrscheinlich das größte Defizit unserer Arbeit ausmacht. Ein anderer Bereich, der wohl verbessert werden kann, ist die Auswahl des Netzwerkes. Hier hätte kein Fokus auf Effizienz gelegt werden müssen, denn durch das "Transfer Learning" ist der Trainingsbedarf sehr niedrig.

Literaturverzeichnis

[1] B. M. Burger W., *Digitale Bildverarbeitung*. Springer-Verlag Berlin Heidelberg, 2015.

[2] towardsdatascience, "a-basic-introduction-to-separable-convolutions.", https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728.

[3] towardsdatascience, "mobilenetv2-inverted-residuals-and-linear-bottlenecks.", https://towardsdatascience.com/mobilenetv2-inverted-residuals-and-linear-bottlenecks-8a4362f4ffd5.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks", 2019.

[5] Prof. Dr. rer. nat. Dieter Maier, "Lecture in Computer & Robot-Vision", 2021.