Artificial Intelligence

Raphael Appenzeller, Sophia Erni, Patrick Knüppel, Andrino Moro

1. Einleitung

Artificial Intelligence (AI) ist der englische Fachbegriff für künstliche Intelligenz, ein aktuelles Forschungsgebiet der Informatik. AI bezeichnet die Intelligenz, die von Maschinen ausgeht. Eine solche Maschine oder ein solches Computerprogramm ist fähig, auf seine Umgebung zu reagieren und sich dieser anzupassen [1]. Es kann also lernen, eine ihr gestellte Aufgabe möglichst effizient zu erledigen. Dieses Projekt befasst sich mit der Erstellung einer Software, die künstliche Intelligenz besitzen soll. Genauer ist das Programm ein „Convolutional Neural Network", welches Objekte in Bildern erkennt. Die Art der Objekte ist variabel, wir haben uns auf Blätter von verschiedenen Baumarten und handgeschriebene Ziffern festgelegt. Während des Forschungslagers wurden Blätter von verschiedenen Bäumen gesammelt und fotografiert. In der Vorbereitungs- und Auswertungsphase widmete sich die Gruppe der Programmierung des Netzwerks. Inspiriert wurde das Team unter anderem von einem Video, in welchem ein AI-Programm vorgestellt wird, das selbstständig „Mario" spielen lernt [2]. Dies ist ein Pionierprojekt der **academia**. Es ist das erste Projekt, dessen Schwerpunkt auf der Erstellung einer Software liegt.
In diesem Bericht werden die Strategien, die dem Programm zugrunde liegen, erklärt und die Resultate präsentiert. Details zur Implementierung in Python und Dokumentation sind in der SOP [3] zu finden. Der Programmcode ist im Internet verfügbar [4].

2. Theorie

2.1. Convolutional Neural Network – die Grundidee

„Convolutional Neural Network" (CNN) bedeutet übersetzt „faltendes[1] neurales Netz". Es handelt sich um einen spezifischen Typ von künstlichen neuralen Netzwerken, welche zum Gebiet der Neuroinformatik gehören. Die Methoden der Neuroinformatik sind von der Biologie inspiriert, sind aber keine direkte Nachbildung dieser komplexen Systeme[2]. Die technischen Anwendungen stehen in der Neuroinformatik im Vordergrund. Das Hauptanwendungsgebiet von CNNs besteht darin, Objekte zu erkennen. Die Funktionsweise eines CNNs lehnt sich folgendermassen an die menschliche visuelle Wahrnehmung an: Im Auge reagieren Sinneszellen (Stäbchen und Zapfen) mit einem elektrischen Signal auf Licht bestimmter Wellenlängen. Diese Struktur der Netzhaut bildet den Input für die nachgeschalteten Schichten von Nervenzellen. Diese Weiterleitung basiert auf dem Alles-oder-nichts-Prinzip; nur wenn die Reizung der Sinneszellen stark genug ist, wird ein Signal weitergeleitet. Die nachgeschalteten Neuronen leiten die Information ans Gehirn weiter. Dort wird diese verarbeitet und aufgrund von Erfahrungswerten entschieden, was für ein Objekt betrachtet wird.
Ein CNN besteht ebenfalls aus vielen Schichten, nachfolgend Layer genannt. Diese Layer sind in Anlehnung an die obigen biologischen Prozesse miteinander verknüpft. Die Signalverarbeitung von Schicht zu Schicht ermöglicht es, aus den individuellen Inputsignalen einen interpretierbaren Output zu generieren.

2.2. Aufbau des CNN

2.2.1. Layer

Künstliche neurale Netzwerke bestehen aus aufeinanderfolgenden Layers, die jeweils unterschiedliche Aufgaben besitzen, jedoch dem gleichen Grundprinzip folgen. Ein Layer besteht aus Input-, Outputneuronen und deren Verbindungen. Hierbei sind die Outputneuronen des einen jeweils gleich wieder die Inputneuronen des darauffolgenden Layers. In Abb. 1 ist ein Beispiel mit drei Inputneuronen (in_1, in_2, in_3) und

[1] Eine Faltung (engl. convolution) ist eine spezielle lineare Integral-Abbildung aus der Mathematik, mit der ein Datensatz in einen anderen Datensatz übertragen wird. In diesem Projekt handelt es sich um diskrete Faltungen, wie sie oft in der Bildbearbeitung verwendet werden, beispielsweise als Weichzeichner oder Kantenfilter.
[2] Die Wissenschaft, die sich mit der exakten Modellierung des Nervensystems befasst, nennt sich Computational Neuroscience.

zwei Outputneuronen (out_1, out_2) dargestellt. Die Verbindung dazwischen ist linear, mit Gewichten (w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}). Das Gewicht w_{ij} verknüpft dabei das Neuron in_i mit dem Neuron out_j ($i = 1, 2, 3, j = 1, 2$) sprich

$$out_j = \sum_{i=1}^{3} in_i \cdot w_{ij} \qquad j = 1, 2.$$

Die Layer sind dafür zuständig, dass die Signale vom Anfang bis zum Ende des neuralen Netzwerkes gelangen, wo sie ausgelesen werden. Dieser Prozess heisst Forwardpropagation. Der Output des letzten Layers besteht aus Klassen, in die der Datensatz unterteilt werden soll. Unterscheidet zum Beispiel das Netzwerk zwischen sieben Baumarten, dann besitzt der letzte Layer sieben Outputneuronen. Die entsprechenden Zahlen out_j machen eine Aussage darüber, wie das Netzwerk das Eingabebild einschätzt. Wenn das Eingabebild von der ersten Baumart ist, dann sollte idealerweise $out_1 = 1$ und alle anderen $out_j = -1$ ($j \neq 1$) sein[3]. Die Layer liefern ausserdem die Lernfähigkeit des neuralen Netzwerkes, indem sie ihre Gewichte anpassen (Backpropagation). Am Anfang werden die Gewichte zufällig initialisiert. Dann ist die Einschätzung des Netzwerkes über ein Objekt noch nicht brauchbar. Durch die wiederholte Anwendung von Forward- und Backpropagation in der Trainingsphase werden die Gewichte automatisch so angepasst, dass die Aussagekraft besser wird – das neurale Netzwerk lernt. Dabei ist es zentral zu verstehen, dass es absolut unbekannt und irrelevant ist, wie genau das Netzwerk die Bilder erkennt.

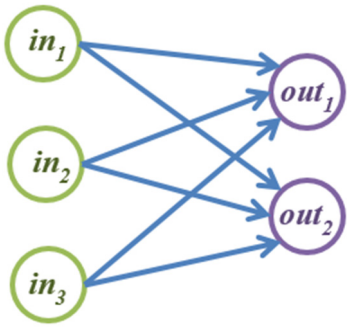

Abb. 1: Beispiel zur Veranschaulichung der Funktionsweise eines kleinen Layers.

In unserem Fall sollen Objekte auf Bildern erkannt werden. Im Computer sind Bilder mit Pixeln abgespeichert. Normalerweise stehen in jedem Pixel drei Werte, die seinen Rot-, Grün und Blauanteil bestimmen. Im Falle eines Schwarz-Weiss-Bildes steht in jedem Pixel nur ein Wert. Farbbilder können auch als Kombination drei verschiedener Bilder angesehen werden. Im Projekt wurde davon nur das grüne Bild verwendet. Alle Daten werden nun eingegeben. Dieser dreidimensionale Datensatz, bestehend aus der x/y-Position im Bild, sowie der Tiefe z wird als Anfangswert dem Netzwerk übergeben. Die Tiefe z nummeriert die Bilder so, dass jedes Bild mit seiner Nummer z adressiert werden kann. In Abb. 2 sind beispielsweise anfangs drei Bilder vorhanden, z geht also von 0 bis 2. Anschliessend werden bei jedem Layer neue Versionen des Anfangsbildes erstellt, weil jeder Filter im Layer je ein neues Bild generiert. Intern werden die Daten meist in dreidimensionalen Arrays als eine Art Stapel (z, x, y) gespeichert. Manchmal ist es jedoch nützlich, sie als lange eindimensionale Liste von Datenpunkten $n = (n_1, n_2, \ldots, n_N)$ zu interpretieren. Diese Konvention vereinfacht in den folgenden Kapiteln die Notation.

2.2.2. Forwardpropagation

Für das ganze neurale Netzwerk besteht eine Forwardpropagation aus dem Einlesen von Daten in die erste Schicht Inputneuronen, anschliessender stückweiser Berechnung des jeweils nächsten Layers, und am Schluss die Berechnung der Outputneuronen des letzten Layers. Das Outputneuron mit der maximalen Aktivierung entscheidet über die Klassifizierung der Inputdaten. Die Forwardpropagation auf Ebene der einzelnen Layers besteht aus der Weiterleitung der Inputneuronen *in* auf Outputneuronen $out = f(in)$, wobei die Funktion f die Verbindung des jeweiligen Layers beschreibt. In Kapitel 2.2.1 wurde eine lineare Abbildung W als Beispiel für eine mögliche Verbindung gegeben.

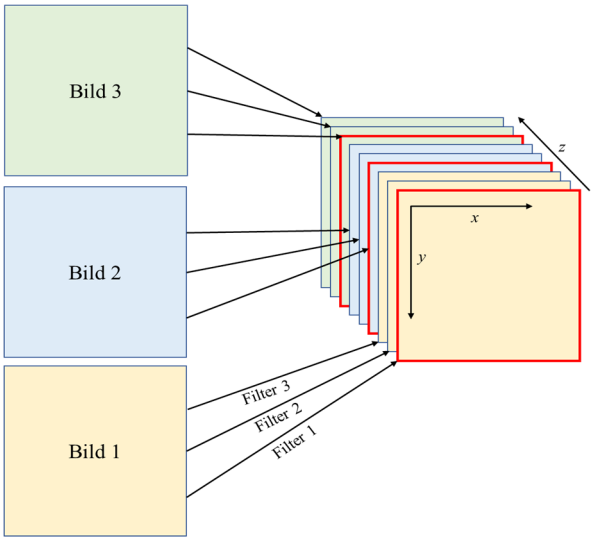

Abb. 2: Beispiel zur Erklärung der Tiefe z.

[3] Die Wahl dieser Referenzpunkte ist frei, aber (-1, 1) ist praktisch und numerisch günstig.

2.2.3. Backpropagation

Mittels Backpropagation kann das neurale Netzwerk lernen. Verwendet das neurale Netzwerk Trainingsdaten, so ist die Klassifikation der einzelnen Bilder bekannt. Der Output des Netzwerks wird mit der wahren Antwort verglichen. Die Gewichte des Netzwerks werden angepasst, damit der Output des Netzwerks näher am gewünschten Output liegt. Um ein Mass der Güte zu erhalten, wird eine Fehlerfunktion berechnet. Diese ist abhängig vom Output des letzten Layers *out* und von den wahren Werten *true*, wobei $true_j = 1$ ist, wenn das Eingabebild die *j*-te Baumart abbildet und andernfalls $true_j = -1$. Die verwendete Funktion für den Fehler *F*

$$F = \sum_{j=1}^{N} \left(out_j - true_j \right)^2$$

entspricht der aufsummierten Summe der Fehlerquadrate der einzelnen Outputneuronen. So werden wegen des Quadrates kleine Fehler weniger stark gewichtet als grosse Fehler. Das Ziel ist es, die Gewichte des Netzwerkes so zu bestimmen, dass der Fehler möglichst klein wird. Es handelt sich hierbei um ein Minimierungsproblem. Die Fehlerfunktion wird nach den Gewichten abgeleitet, um zu bestimmen, in welche Richtung jedes Gewicht verändert werden muss, um den Fehler zu minimieren. Sie werden um das Produkt der Ableitung mit einer frei wählbaren Lernrate verschoben, in Richtung von sinkendem Fehler *F*. Dieses Verfahren wird über eine grosse Anzahl Trainingsbilder iteriert.

2.2.4. Convolutional Layer

Das Convolutional Layer ist das Herzstück des CNNs. Der Layer besteht aus mehreren Filtern, die jeweils verschiedene Gewichte enthalten. Diese Filter bewegen sich über das Bild und multiplizieren ihre Gewichte mit den darunterliegenden Pixelwerten. Diese neu entstandenen Werte werden addiert und ersetzen so den alten Wert des Pixels, auf den der Filter zentriert ist. In Abb. 3 wird das orange Feld auf dem Bild durch das grüne Feld im Output ersetzt. Anschliessend durchwandert der Filter das Bild, und fährt nach und nach über alle Felder des Bildes. So entsteht der Output. Alle Filter müssen nun jeweils einmal über alle Inputbilder wandern. Somit ergibt sich eine neue Anzahl an Bildern, nämlich das Produkt der Anzahl Filter mit der alten Anzahl Bilder. Jeder dieser Filter wird zuerst mit zufälligen Werten als Gewichten versehen, die beim Lernprozess dann durch die Backpropagation so verändert werden, dass der Wert der Fehlerfunktion minimiert wird und das CNN Bilder möglichst gut erkennen kann. Bemerkenswert an diesem Layer ist die kleine Anzahl Gewichte (d. h. variable Parameter), da ein nützlicher Filter wesentlich kleiner sein kann als das Bild. Im Rechenbeispiel wurde das Bild der Einfachheit halber klein gewählt.

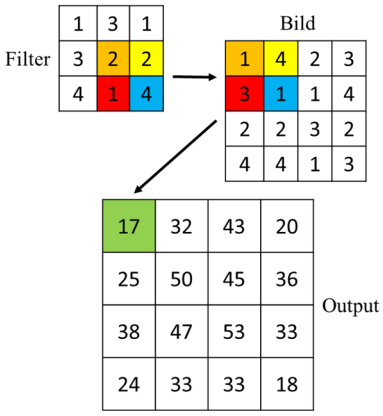

Abb. 3: Rechenbeispiel für Convolution (farbiger Bereich):
$2 \cdot 1 + 2 \cdot 4 + 1 \cdot 3 + 4 \cdot 1 = 17$
Danach wandert der Filter um einen Pixel nach rechts. Das Zentrum des Filters befindet sich dann bei der gelben Zelle mit der 4.

2.2.5. Hyperbolic Tangent Layer

Der Hyperbolic Tangent Layer besteht aus einer nichtlinearen Funktion, die elementweise auf das Inputbild angewendet wird. Jeder Pixel wird als *x*-Wert in eine geeignet skalierte hyperbolische Tangensfunktion (tanh) eingegeben

$$y = \tanh(x) \equiv \frac{e^x - e^{-x}}{e^x + e^{-x}},$$

und darauf durch den erhaltenen *y*-Wert ersetzt. Einerseits imitiert die Funktion das biologische Alles-oder-nichts-Prinzip. Andererseits wird ein übermässiges Auseinanderdriften und mögliches Divergieren der Pixelwerte eingedämmt, dank der Grenzwerte für $x \to \pm\infty$. Der hyperbolische Tangens wird nach [5] verwendet mit einem Streckfaktor von 1.72 und einem Vorfaktor des Funktionswertes von 2/3. Dadurch liegen die Punkte (–1, –1) und (1, 1) auf dem Graphen mit Steigung 1, wie in Abb. 4 gezeigt ist. Dies ist

abgestimmt auf die Wahl der Referenzpunkte[4] des letzten Layers $true_j$ (Kap. 2.2.3). Der Streckfaktor und Vorfaktor wurden fixiert, dieser Layer enthält also keinerlei freie Parameter; er wird vom Lernen nicht beeinflusst.

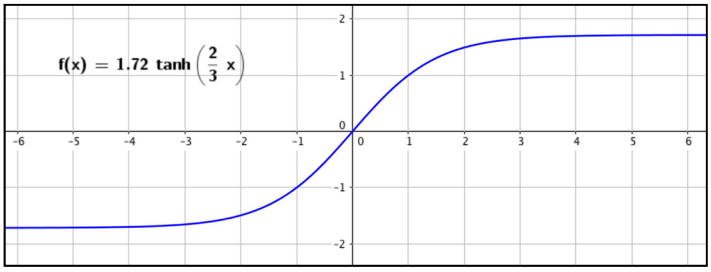

Abb. 4: Die hyperbolische Tangensfunktion mit dem gewählten Streckfaktor 1.72 und Vorfaktor 2/3.

2.2.6. Fully Connected Layer

Im Fully Connected Layer ist jedes Inputneuron gewichtet mit jedem Outputneuron verbunden. Dieser Layer entspricht dem Layer, der bereits als Beispiel in Kapitel 2.2.1 verwendet wurde. Speziell ist, dass die Anzahl der Inputneuronen und diejenige der Outputneuronen voneinander völlig unabhängig sind. Mathematisch gilt $out = W \cdot in$ mit einer Matrix W. So ergibt sich die ideale Anwendungsmöglichkeit des Fully Connected Layers als letzter Layer, da er die beliebige Menge an Inputs in eine von aussen vorgegebene Anzahl Outputs übersetzt.

3. Methodik

3.1. Datenerfassung

Im Lager wurden Blätter von mehreren Bäumen verschiedener Arten gesammelt. Die Teilblätter wurden von ihrem Zentralästchen entfernt und alle Blätter einzeln auf einem weissen Blatt fotografiert. Da die Sammelarbeit vorwiegend am Tag und die Ablichtung am späten Abend durchgeführt wurde, war die Beleuchtung nicht optimal. Mit zwei Lampen wurde versucht, diese so konstant wie möglich zu halten, doch trotzdem sieht der Hintergrund, wie in Abb. 5 erkennbar, gelblich aus. Der Schwerpunkt wurde auf Laubbäume gelegt, welche das Programm unterscheiden soll. Zusätzlich wurden Fichtennadeln gesammelt. Das Netzwerk und seine Resultate werden in Kapitel 4.2 genauer erläutert.
In Abb. 6 sind den gesammelten Arten die Anzahl der erfassten Bilder zugeordnet. Weiter ist die Codierung der Baumart im Code angegeben. Diese entspricht den „Lösungen“, die das Programm während der Trainingseinheiten erhält. Aus programmtechnischen Gründen ist die Klassifikation 0 einfacher handhabbar als „Zitterpappel“.

Abb. 5: Ein Bild aus den Rohdaten.

Codierung	Art	Anzahl Bilder
0	Zitterpappel	990
1	Birke	983
2	Vogelbeere	982
3	Felsenkirsche	902
4	Weide (Purpur klein)	1107
5	Schwarzwerdende Weide	998
6	Fichte	1213

Abb. 6: Zusammenfassung der aufgenommenen Bilder.

[4] Die Steigung der Funktion multipliziert sich mit der Lernrate aller Layer, die vor dem Hyperbolic Tangent Layer liegen, weshalb die Steigung zwischen $x = -1$ und $x = 1$ auf ungefähr 1 gewählt wurde, also insbesondere nicht nahe an 0. Zudem fördert ein Referenzpunkt im flachen Bereich des $\tanh(x)$ betragsmässig grosse Inputwerte x, wodurch wiederum grosse oder sogar divergierende Gewichte im Lernprozess entstehen können.

3.2. Datenaufbereitung

Zunächst sind die mit der Digitalkamera aufgenommenen Bilder der Blätter Bilddateien im JPG-Format. Der Input für das CNN verlangt allerdings eine Matrix aus Zahlen. Ausserdem ist die originale Auflösung der Bilder mit 4608x3456 Pixeln sehr hoch. Um die grosse Datenmenge in sinnvoller Zeit verarbeiten zu können, mussten die Bilder zuerst komprimiert werden. Einfachheitshalber wird nur der grüne Farbkanal der Bilder angeschaut. Für die Bearbeitung der Rohdaten, „preprocessing“ genannt, wurden kleine Programme geschrieben, um die Teilschritte zu erledigen. Im Folgenden wird nur deren Strategie erläutert.
Zuerst wird der interessante Bereich eines Bildes bestimmt. Alle Blatt-Bilder weisen einen dunklen (hier rot dargestellten) Rand im Hintergrund auf, der entfernt werden soll. Dafür werden alle Bilder eingelesen und deren grüne Farbkanäle übereinandergelegt. Konkret bedeutet Übereinanderlegen, dass für einen Pixel (x,y) das Maximum über alle Bilder ausgewählt wird. Wiederholt man dies für jeden Pixel erhält man ein Bild wie in Abb. 7 dargestellt. Anschliessend wird der Rand, sprich der uninteressante Teil bei allen Bildern abgeschnitten. Der schliesslich verwendete Bereich liegt zwischen dem oberen rechten Punkt (300, 500) und dem unteren linken (4000, 2900) und wird vom schwarzen Rahmen eingeschlossen. Anschliessend wird ein Rahmen um das Blatt zentriert. Dies bedeutet, vom eingelesenen Bild wird ein

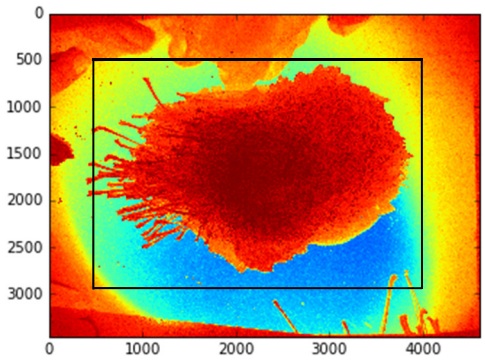

Abb. 7: Alle Blätter (gesamter Datensatz) übereinandergelegt.

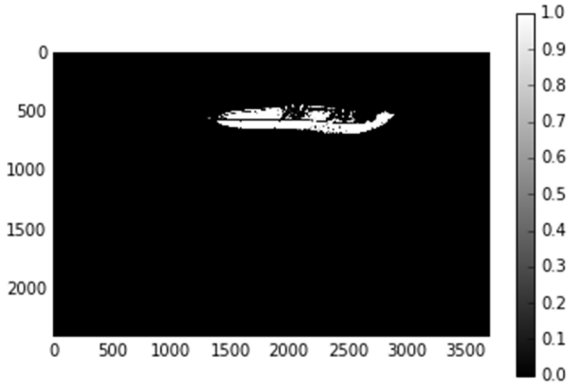

Abb. 8: Bearbeitung des Bildes, um das Zentrum zu finden.

quadratischer Bereich mit einer Seitenlänge von 2000 Pixeln bestimmt. Dieses Quadrat liegt zentriert auf dem Baumblatt, welches jedoch nicht dem Zentrum des Bildes entspricht.
Um das Zentrum des Blattes zu finden, werden alle Pixelwerte, die höher sind als ein Referenzwert, auch Threshold genannt, auf weiss gesetzt und alle unter dem Threshold auf schwarz. Da das Blatt einen hohen Grünanteil hat und somit auch die entsprechenden Pixelwerte tief sind, führt dies dazu, dass das Blatt weiss und der Rest schwarz wird. In Abb. 8 ist eine Veranschaulichung davon dargestellt. Dafür wurden das Orginalbild von Abb. 5 und ein Threshold von 180 verwendet. Von den weissen Pixeln wird im Anschluss der Mittelwert der x- wie auch der y-Koordinaten berechnet. Dadurch erhält man ziemlich präzise das Zentrum des Blattes. So kann der quadratische Bereich, in den das Baumblatt zentriert ist, weiterverwendet werden.
Nach diesem Bearbeitungsschritt handelt es sich noch um ein Bild von 2000x2000 Pixeln. Damit es sich für das CNN eignet, wird es mit einem Faktor komprimiert und in eine Matrix umgewandelt, deren Einträge zwischen null und 255 liegen. In Abb. 9 ist ein fertig bearbeitetes und mit einem Faktor 50 verkleinertes Bild dargestellt. Die Auflösung beträgt nur noch 40x40 Pixel. Im Rahmen des Projektes wurden aber auch andere Matrizengrössen ausprobiert wie 50x50, 80x80 oder 100x100 Pixel.
Die fertig bearbeiteten Bilder werden nun in zwei Listen eingeteilt. Die eine besteht aus sogenannten Trainings-Bildern, mit denen das CNN trainiert wird. Die restlichen Bilder fungieren als Test-Bilder, die verwendet werden, um das CNN nach der Trainingsphase mit unbekannten Bildern zu testen.

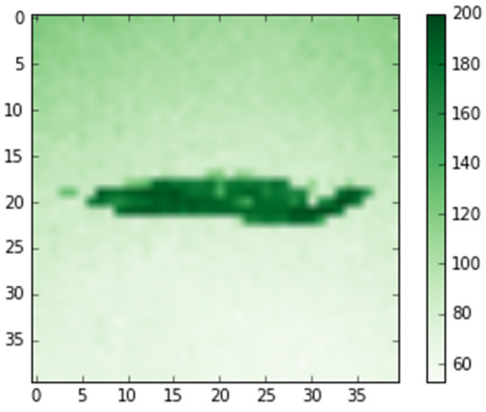

Abb. 9: Fertig bearbeitete Matrix als grün eingefärbtes Bild dargestellt.

Das Netz wird so mit neuen Bildern konfrontiert. Die im Folgenden an den handgeschriebenen Zahlen erklärte Normierung wurde optional auch bei den Blattbildern durchgeführt.

4. Resultate

4.1. Handgeschriebene Ziffern (Digits)

Zum Testen des Programmcodes wurden zuerst handgeschriebene Ziffern klassifiziert. Es handelt sich um den Standard-Datensatz MNIST [6]. Dieser Datensatz wird oft als Einstiegsbeispiel für Methoden zur Klassifizierung von Daten verwendet. Der verwendete Datensatz besteht aus 7000 Beispielen jeder Ziffer und wurde unterteilt in je 6000 Beispiele zum Trainieren des Netzwerks und 1000 Exemplare zum anschliessenden Testen des Netzwerks. Jedes Bild besteht aus 8bit Graustufen und misst 24x24 Pixel, davon 4 Pixel weisser Rand, welcher entfernt wurde. So ergibt sich eine Bildgrösse von 20x20 Pixeln. Ein solches Input Digit ist in Abb. 10 links gezeigt, normiert auf weiss = 0 und schwarz = 1. Als erstes wurden die Daten aufbereitet, indem zuerst der mittlere Wert jedes Pixels über die gesamte Trainingsmenge auf 0 verschoben wurde. Diese Operation schiebt die Inputdaten in den Arbeitsbereich des Netzwerks zwischen den Referenzpunkten (–1, 1). Danach wurde jeder Pixel durch seine Varianz über die gesamten Trainingsdaten dividiert (Normalisierung). Dies verstärkt die äusseren Pixel im Hintergrund und schwächt die inneren Pixel, wo häufig Teile der Ziffern liegen. Damit wird der Einfluss ausgeglichen, den die verschiedenen Pixel auf das Endresultat haben. Das Ergebnis ist in Abb. 10 rechts dargestellt, nun mit negativen wie positiven Pixelwerten.

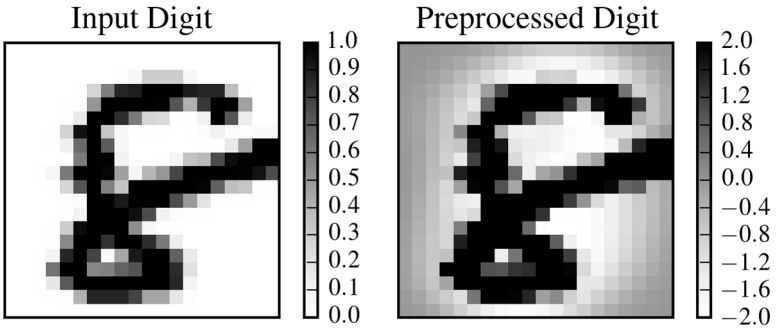

Abb. 10: Ursprüngliches MNIST Bild links und aufbereitetes Bild rechts. Dabei wurden die Pixelwerte um 0 zentriert und die Varianz der verschiedenen Pixel ausgeglichen.

Experimentell wurde eine spezielle Anordnung von drei Layern ermittelt, aus denen das erfolgreichste Netzwerk besteht:

Input Dimension (z, x, y) = (1, 20, 20)
1. Convolutional Layer mit neun Filtern von je 9x9 Gewichten
   Dimension (z, x, y) = (9, 20, 20)
2. Hyperbolic Tangent Layer
   Dimension (z, x, y) = (9, 20, 20)
3. Fully connected Layer
   Output Dimension (z, x, y) = (10, 1, 1)

Nachfolgend ist in Abbildungen dargestellt, wie die Ziffer nach jedem Layer verändert aussieht. Somit können die Operationen der einzelnen Layers bildlich nachvollzogen werden. Diese Darstellungen wurden mit einem trainierten CNN, das 97 % aller Test-Bilder richtig erkannte, durch eine Forwardpropagation erstellt.
Nach dem Convolutional Layer sind in Abb. 11 neun Bilder ersichtlich, weil der verwendete Layer neun Filter enthält (siehe Kapitel 2.2.4). Die Werte wurden durch die Filter gestreut, die Farbskala wurde um blau und rot jenseits von (–2, 2) ergänzt. Alle neun Bilder sind unterschiedlich, alle Filter sind durch das Training also auf andere Merkmale spezialisiert worden. Alle extremen Werte, die rot oder blau dargestellt würden, sind in der Abb. 12 verschwunden. Der Hyperbolic Tangent Layer hat die Werte zurück in den schwarz-weissen Bereich abgebildet.

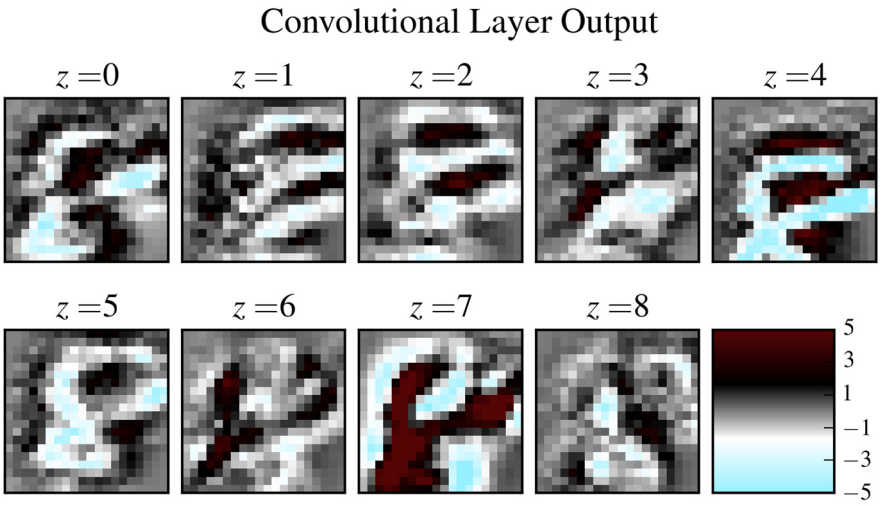

Abb. 11: Nach dem Convolutional Layer. Die Farbskala wurde auf +5 = dunkelrot und –5 = hellblau erweitert, um die Entwicklung der Pixelwerte zu verfolgen und den Unterschied zum nächsten Layer zu verdeutlichen.

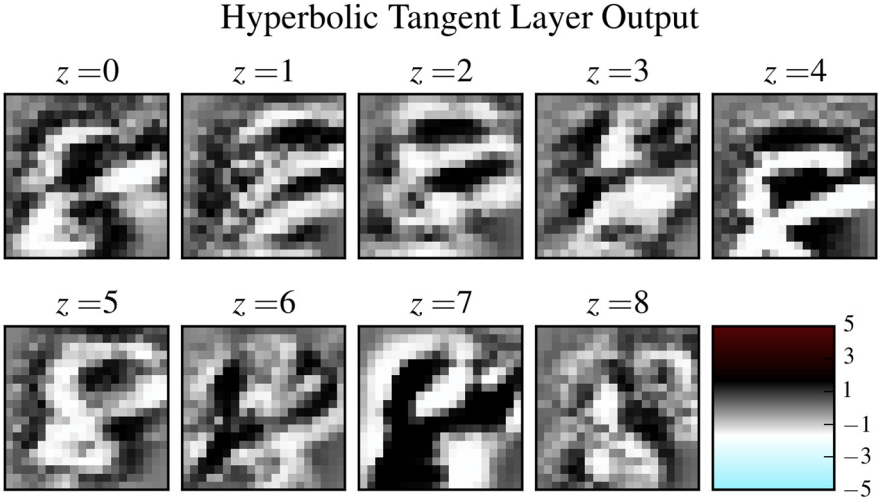

Abb. 12: Nach dem Hyperbolic Tangent Layer. Eine Funktion dieses Layers ist, die extremen Werte durch die Grenzwerte des hyperbolischen Tangens zu limitieren (siehe Kapitel 2.2.5).

Als letztes führt der Fully Connected Layer die Daten aus Abb. 12 in die zehn verschiedenen Output-Klassen über. Der endgültige Output des gesamten Netzes ist in Abb. 13 zu erkennen. Der höchste Wert befindet sich bei derjenigen Zahl, die vom Netz als die wahrscheinlichste Lösung ermittelt wurde. Das Netzwerk hat die Ziffer also korrekt als eine Acht identifiziert.

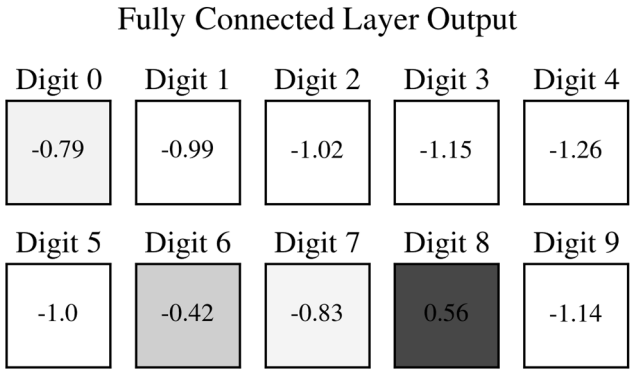

Abb. 13: Dies ist der endgültige Output. Die Zahl auf dem Ursprungsbild wurde korrekt als eine Acht erkannt.

Im Idealfall würde der Wert der richtigen Lösung plus eins und die Werte aller anderen möglichen Zahlen minus eins betragen. Es gibt auch einfachere Digits, bei denen dies beinahe zutrifft. Die hier verwendete Zahl ist jedoch ein mittelschwieriger Fall. In Abb. 14 wird eine Auswahl der 3% Bilder gezeigt, die vom Netz nicht richtig bestimmt wurden. Dies verdeutlicht, wie schwierig es sein kann, die Ziffern zu erkennen. Sie liegen teilweise nicht ganz im Bild oder sind besonders unleserlich geschrieben.

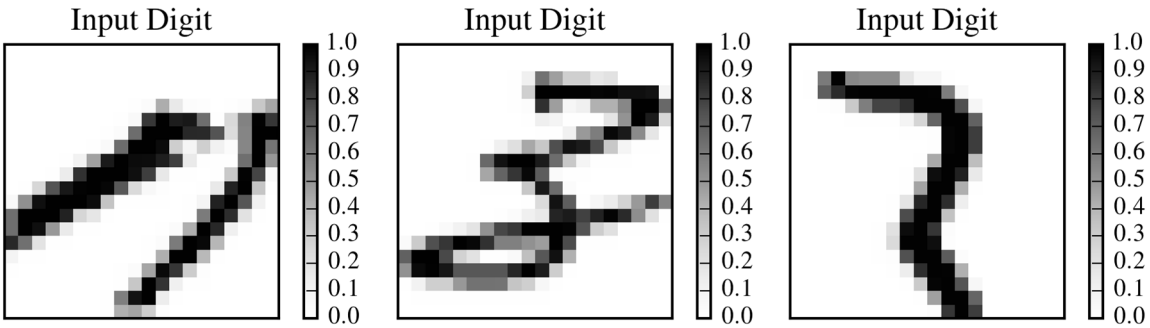

Abb. 14: Dies sind drei Digits, die vom Netz falsch bestimmt wurden. Bild 1: Netz: 0, wahre Antwort: 5; Bild 2: Netz: 7, wahre Antwort: 3; Bild 3: Netz: 7, wahre Antwort: 3.

In Abb. 15 ist die Entwicklung des Netzes durch Training ersichtlich. Mit steigender Anzahl Trainingsbilder *d*, gezeigt auf einer logarithmischen *x*-Skala, erreichte das Netz 97% Erfolg.

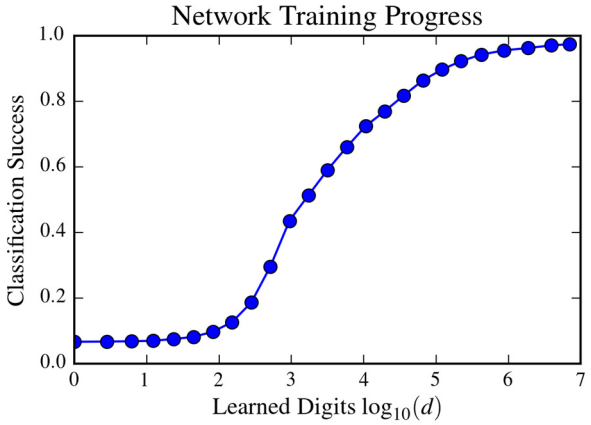

Abb. 15: Erfolg durch Trainingsprozess.

4.2. Fotografierte Blätter (Leafs)

Analog zur beschriebenen Methodik in Kapitel 4.1 wurde ein Netzwerk erstellt und getestet, um die gesammelten Baumblätter zu klassifizieren. Von den Bildern wurde mithilfe der im Kapitel 3.2. beschriebenen Methode ein Datensatz erstellt. Dieser besteht aus je 880 Bildern pro Baumart und ist wiederum unterteilt in je 800 Beispiele zum Trainieren des Netzwerks und 80 Exemplare zum anschliessenden Testen. Gesamthaft steht so also ein Datensatz von 5600 Trainingsbildern zur Verfügung. Die 50x50 Pixel grossen Bilder sind ebenfalls in 8-bit Graustufen eingeteilt. Genau genommen wurden mehrere Datensätze mit unterschiedlichen Auflösungen erstellt, diese werden aber nicht alle gezeigt. Das durch die Datenaufbereitung erhaltene Bild ist in Abb. 16 links dargestellt. Rechts davon befindet sich dasselbe Blatt, allerdings normalisiert.

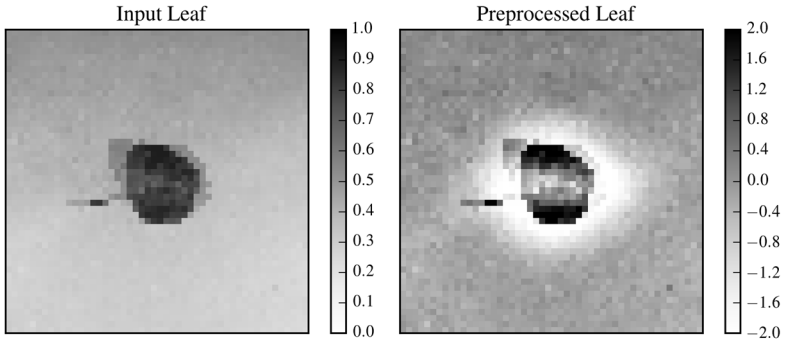

Abb. 16; Aufbereitetes Bild links und das noch zusätzlich normalisierte Bild rechts. Wie bereits bei den Ziffern wurden die Pixelwerte um 0 zentriert und die Varianz der verschiedenen Pixel wurde ausgeglichen (rechtes Bild).

Das Vorgehen bei den Blättern ist identisch zu dem der Ziffern, weshalb die Einzelheiten hier abgekürzt werden. Es wurden verschiedene Netzwerkstrukturen für die Blätter ausprobiert. Das erfolgreichste Netzwerk wurde wieder mit der gleichen Struktur wie bei den Ziffern erzielt. Maximal konnten so 80% der Bilder richtig erkannt werden. Bei den drei Layern wurden jedoch im Vergleich zu den Ziffern die Dimensionen und die Anzahl Filter geändert:

Input Dimension (z, x, y) = (1, 50, 50)

1. Convolutional Layer mit 15 Filtern von je 5x5 Gewichten
   Dimension (z, x, y) = (15, 50, 50)
2. Hyperbolic Tangent Layer
   Dimension (z, x, y) = (15, 50, 50)
3. Fully connected Layer
   Output Dimension (z, x, y) = (7, 1, 1)

In Abb. 17 ist die Funktionsweise des trainierten Netzwerkes dargestellt. Auffällig ist, dass das Bild nach dem Convolutional Layer (links) weniger von dem nach dem Hyperbolic Tangent Layer (rechts) unterscheidet, als es bei den Ziffern der Fall war.

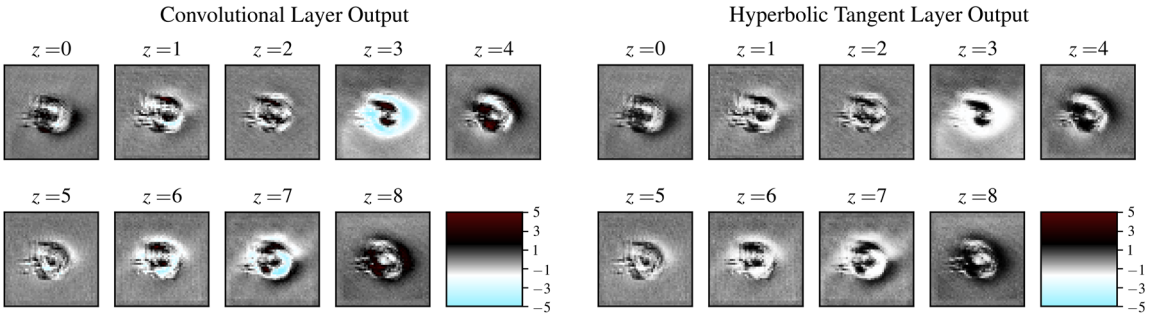

Abb. 17: Ausgabe des ersten und zweiten Layers nach dem Trainieren des Netzes.

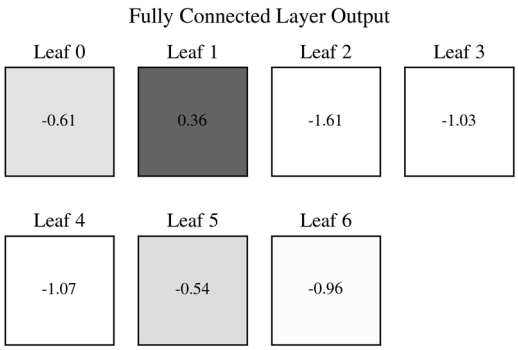

Abb. 18: Ausgabe des letzten Layers und damit des Netzwerks für Blätter. In diesem Fall wurde das Blatt korrekt als Typ 1 (Birke) erkannt.

Das Netzwerk erkannte die Blätter mit der gleichen Netzwerkstruktur wie bei den Ziffern, doch weniger gut als diese. Der Rekord bei der Trefferquote der Blätter war immerhin 80%.

5. Diskussion

Die Ziffern wurden erfolgreich mit einer 97% Trefferquote vom CNN erkannt. Dies wurde trotz den teilweise unklaren und unleserlichen Ziffern erreicht. Man könnte eventuell noch eine Testsession mit den **academia**-Mitgliedern durchführen, wie hoch die Trefferquote bei ihnen ist, um interessante Vergleiche anstellen zu können. Das Ziel, ein funktionsfähiges CNN komplett selbst zu programmieren, konnte also erreicht werden.
Als zusätzliche Aufgabe wurden in einem anderen Experiment dem CNN Blätter als Input gegeben. Um diese Blätter von einer normalen Bilddatei in eine für das CNN leserliche Matrix zu verwandeln, mussten diese Bilddateien stärker aufbereitet werden als die Ziffern. Diese Tatsache führte zu viel Arbeit und einer erhöhten Fehlerrate, da Fehler in der Aufbereitung nur sehr schwer erkennbar waren. Bei den Ziffern wurden die Rohdaten nur normalisiert, bei den Blättern war die zusätzliche Schwierigkeit, sie in ein Format zu bringen, mit welchem das Netz innert sinnvoller Zeit arbeiten kann. Diese Aufgabe konnte erfolgreich bewältigt werden. Das Netz erreichte eine Trefferquote von 80%. Dies könnte zum einen an den Trainingsdaten liegen. Die 5800 Baum-Blätter, die dem Netz zur Verfügung stehen, sind relativ wenige im Verhältnis zu den 60’000 Ziffern. Weiter ist die Auflösung von 50x50 Pixeln sehr klein. Dies vernichtet Information in der Datenmenge, und schlechtere Trefferquoten sind zu erwarten. Unser biologisches neurales Netz, das Gehirn, kann die meisten dieser Bilder nicht mehr korrekt der Baumart zuordnen.
Weiter wurden verschiedene Netzwerkstrukturen getestet, jedoch konnte immer gerade eine Trefferquote von 80% erreicht werden. Auch eine hohe Auflösung von zum Beispiel 100x100 Pixel führte zu keiner Verbesserung. Die Ursache dieses Phänomens ist nicht geklärt. Möglicherweise ist es darauf zurückzuführen, dass das Netz nur nach der Blattgrösse unterscheidet. Dies führt bei dem verwendeten Datensatz sehr wohl zum Erfolg, weil die gesammelten Blätter verschieden gross sind. Doch mit dem jetzigen Netz wäre es schwierig, ähnliche Blättersorten auseinanderzuhalten. Eine fehlerhafte Programmierung ist unwahrscheinlich, da dasselbe Netz bei den Ziffern erfolgreich arbeitete. Es könnte allerdings sein, dass gewisse Baumarten einander so ähnlich sind, dass sie vom Netz schlicht nicht auseinandergehalten werden können, respektive, dass dafür eine viel höhere und damit rechenaufwändige Auflösung nötig wäre. Im erfassten Datensatz sind alle Blattstängel gleich ausgerichtet und als Hintergrund wurde konsistent ein weisses Papier verwendet. Es wäre interessant, das Netz mit Blättern in natürlicher Umgebung und mit beliebiger Orientierung zu testen.
Im Projekt wurde ein eigenes CNN programmiert. Man könnte dieses mit anderen, bereits existierenden Programmen vergleichen. Obwohl das Programm funktionsfähig und der Code bereits weit ausgereift ist, muss wohl bezweifelt werden, ob es mit anderen professionellen Implementationen konkurrieren kann. Es konnte gezeigt werden, dass die **academia** im Prinzip ein CNN selbstständig von Grund auf programmieren kann, aber vor allem die nötige Zeit fehlte, um das Programm vollständig zu perfektionieren. Deshalb wäre für weitere Projekte ein Umstieg auf eine andere Lösung, zum Beispiel Open-Source Projekte, möglich.

6. Fazit

Das Hauptziel dieses Projektes konnte erreicht werden. Ein funktionsfähiges Netzwerk zum Erkennen von Objekten wurde selbstständig von Grund auf programmiert, was eine ziemliche Herausforderung darstellte. Die Funktionsweise des Programms kann durch die erfolgreiche Erkennung von handgeschriebenen Ziffern sichergestellt werden. Dabei überzeugt die Trefferquote von 97%. Das Ziel in dieser Hinsicht konnte also erreicht werden. Weiter wurde ein Datensatz von gut 7100 Bildern von Baumblättern erstellt. Die Verarbeitung der Rohbilder in weiter bearbeitbare Daten für das Netzwerk ist geglückt. Bei der Zuordnung der Bilder zu den Baumarten könnte die 80%ige Trefferquote in Folgeprojekten noch verbessert werden.

7. Quellen

[1] Anonym, https://en.wikipedia.org/wiki/Artificial_intelligence, (Zugriff 14.09.2016).
[2] Sethbling, https://www.youtube.com/watch?v=qv6UVOQ0F44, (Zugriff 11.01.2016).

[3] **academia**-SOP-A0037, „AI – Convolutional Neural Network“ (Version V01, 26.02.2017).
[4] **academia**, Github repository https://github.com/Knuppknou/academia_ai, (2017).
[5] LeCun Y., et al., „Efficient BackProp.“, in: Montavon G., Orr G., Müller K. R. (Hrsg.) „Neural Networks: Tricks of the Trade“, Springer Berlin, Heidelberg (1998).
[6] LeCun Y., Cortes C., Burges C. J. C., „MNIST Data“, http://yann.lecun.com/exdb/mnist/ (Zugriff 08.02.2017).