Proseminar
Künstliche neuronale Netze

Jens Ostertag

17. Mai 2022

Inhaltsverzeichnis

1 Computer lernen lassen

„Künstliche Intelligenz“, „Maschinelles Lernen“ und „Neuronale Netze“ - Alle diese Begriffe sind gewissermaßen miteinander verbunden, denn ihre Technik beruht darauf, dass Computer lernen können. Vor einem tieferen Einblick möchte ich diese Begriffe jedoch einordnen und differenzieren.

Bei einem künstlichen neuronalen Netz handelt es sich um die Simulation des menschlichen bzw. natürlichen neuronalen Netzes, einer Struktur, die im Gehirn vorhanden ist. Neuronale Netze sind die Grundlage für alles, das Lernen kann oder Entscheidungen treffen soll.

Maschinelles Lernen beschreibt das Lernen eines solchen neuronales Netzes. Wie auch der Mensch ist ein künstliches neuronales Netz in der Lage, sich unterschiedliche Fähigkeiten anzueignen, unabhängig von der Komplexität der jeweiligen Aufgabe. Dadurch wird beispielsweise ermöglicht, dass ein Programm Bilder erkennt und klassifiziert oder die Position bestimmter Objekte darin bestimmen kann.

Die künstliche Intelligenz ist die undefinierteste Form neuronaler Netze, was auch damit zusammenhängt, dass es keine sehr genaue Definition von Intelligenz gibt. Ist man intelligent, wenn man schnell rechnen kann? Somit ist bereits ein simpler Taschenrechner von vor 40 Jahren äußerst intelligent. Oder bedeutet Intelligenz, dass man selbstständig denken und fühlen können muss, in dessen Zusammenhang der Begriff "Künstliche Intelligenzäuch meistens verwendet wird? Aufgrund dieser Undefiniertheit wird im Folgenden, sofern möglich, auf diesen Begriff verzichtet.

Obwohl die Theorie schon etwas länger existiert, erfolgte ein großer Vorsprung in der Entwicklung des maschinellen Lernens erst in den letzten Jahren. Auch wenn es zuerst nicht so scheint, kommt heute jeder täglich damit in Berührung, beispielsweise

- im Verkehr, durch schlau gesteuerte Ampelphasen oder nahezu selbstfahrende Autos sowie Assistenzsysteme wie Spurhalteassistenten, Einparkhilfen oder Ähnliches,
- im Internet, wo jedem im Sekundentakt neue Inhalte basierend auf persönlichen Präferenzen vorgeschlagen werden,
- im medizinischen Bereich, wo Bilderkennung bei der Auswertung bildgebender Verfahren hilft,

- in der Industrie, wo Produktionsabläufe durch maschinelles Lernen optimiert wurden.

Im Folgenden soll es darum gehen, wie ein neuronales Netz funktioniert und wie man es mit maschinellem Lernen verwenden kann.

1.1 Aufbau neuronaler Netze [1]

Künstliche neuronale Netze sind sehr stark an natürliche neuronale Netze angelehnt, weshalb es sinnvoll ist, zuerst diese zu betrachten.

Sogenannte „Neuronen“ (Nervenzellen) sind durch Synapsen miteinander verbunden, welche für den Elektronenfluss zwischen jeweils zwei Neuronen mithilfe eines „Synapsengewichts“ verantwortlich sind. Mit diesem lässt sich die Anfälligkeit des nachgehenden Neurons auf das Signal des vorhergehenden Neurons regulieren. [4]

Es ergibt sich somit das folgende Bild:

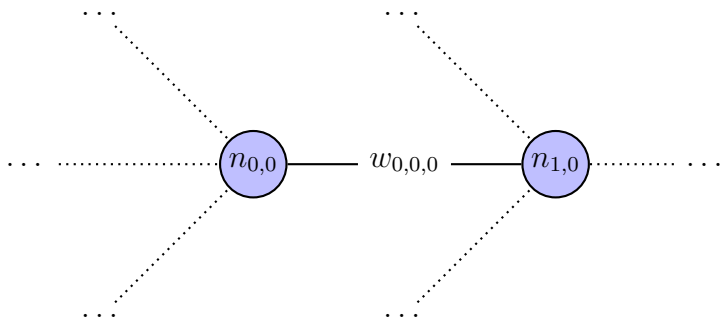

Abbildung 1: Synapse zwischen zwei Neuronen

Es sei jedoch angemerkt, dass ein Neuron beliebig viele vorausgehende oder nachgehende Neuronen haben kann, die in Abbildung 1 der Übersicht halber nicht enthalten sind. Als grobe Größenordnung sei die Anzahl der Neuronen eines ausgewachsenen, menschlichen Gehirns genannt, welche sich je nach Literatur auf 80 bis 100 Milliarden beläuft. In der Abbildung sind vernachlässigte Ein- und Ausgänge von Neuronen mit gepunkteten Linien symbolisiert.

Eine wichtige Grundlage, um aus dem natürlichen neuronalen Netz ein Künstliches abzuleiten, ist das Betrachten von Ausgaben von Neuronen und

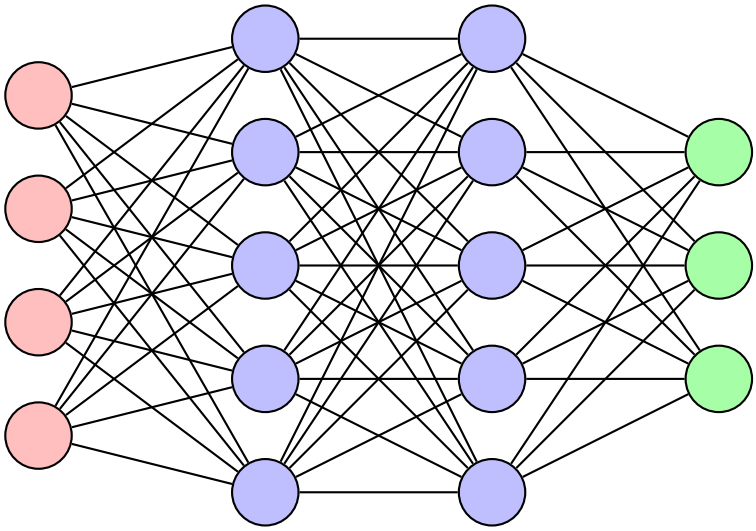

Abbildung 2: Aufbau eines einfachen neuronalen Netzes

von Synapsengewichten als Zahlen. Das ermöglicht die Anwendung von Formeln zur mathematischen Beschreibung.

Ebenso wichtig ist die Einteilung aller Neuronen des Netzes in unterschiedliche Schichten, die teilweise sogar unterschiedliche Aufgaben übernehmen müssen. Als Veranschaulichung dafür dient Abbildung 2, die den Aufbau eines einfachen neuronalen Netzes darstellt.

Die erste Schicht ist die sogenannte „Eingabeschicht“, dessen Neuronen (rot) vergleichbar sind mit Sinneszellen. Hier werden Daten gesammelt, mit denen eine Ausgabe generiert werden soll. Es folgen beliebig viele „versteckte Schichten“. Diese erfüllen den Zweck, dass das Netz auch kompliziertere Dinge lernen kann, da die Gesamtheit der Gewichte ohne versteckte Schicht nur eine linear separierbare Funktion darstellen könnte. Abschließend folgt eine „Ausgabeschicht“, welche beispielsweise Entscheidungen des Netzes ausgibt, welche dann in einem anderen Teil des Programms verarbeitet werden. In diesem Fall sind alle Neuronen einer Schicht mit allen Neuronen der nächsten Schicht verbunden. Wie jedoch in Kapitel 2 noch thematisiert wird, existieren auch weitere Strukturen für den Aufbau eines neuronalen Netzes.

Diese Grundlagen ermöglichen eine Berechnung von beispielsweise Aus-

gaben mithilfe der Formel

$$o_{i,j} = \varphi \left(\sum_{k=0}^{|n_{i-1}|} o_{i-1,k} * w_{i-1,k,j} \right) \tag{1}$$

mit

- $n_{i,j}$: Neuron in der Schicht i an der Stelle j
- $o_{i,j}$: Ausgabe des Neurons $n_{i,j}$
- φ: Differenzierbare Aktivierungsfunktion
- $|n_i|$: Anzahl der Neuronen in der Schicht i
- $w_{i,k,j}$: Synapsengewicht zwischen den Neuronen $n_{i,k}$ und $n_{i+1,j}$

1.2 Drei Arten des maschinellen Lernens [2]

Damit ein neuronales Netz lernt, müssen seine Gewichte angepasst werden. Dieser Vorgang wird auch Training genannt. Das kann auf drei unterschiedliche Arten erfolgen.

1.2.1 Überwachtes Lernen

Das überwachte Lernen ist die wohl am häufigsten verwendete Methode, ein neuronales Netz zu trainieren. Es zielt darauf ab, auch zu bislang unbekannten Eingaben eine passende Ausgabe zu generieren, meistens geht es darum, die Eingaben in Gruppen bestimmter Eigenschaften einzuteilen.

Während des Trainingsvorgangs wird mithilfe eines Trainingsdatensatzes (einige Eingaben mit zugehörigen erwarteten Ausgaben) angelernt, welche Muster in den Eingabedaten zu bestimmten Ausgaben gehören. Erhält das Netz nach abgeschlossenem Training eine Eingabe, ist es in der Lage, ein Muster darin zu erkennen und anhand dessen eine passende Ausgabe zu generieren.

In Abbildung 3 entsprechen alle Punkte einer bestimmten Eingabe, gleichfarbige Punkte haben dieselbe erwartete Ausgabe. Während des Trainings lernt das Netz beispielsweise, dass es Eingaben mit hohem x- und niedrigem y-Wert die Ausgabe grüner Punkte zuteilen soll.

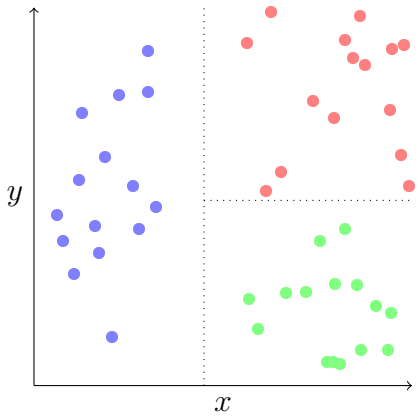

Abbildung 3: Klassifizierung von Daten

1.2.2 Unüberwachtes Lernen

Das Ziel des unüberwachten Lernens ist es, innerhalb einer Menge von Eingaben Gruppen anhand ähnlicher Strukturen zu finden. Dafür wird keine Bearbeitung oder Sortierung der Daten benötigt.

Daher ist diese Lernart besonders interessant für die Clusteranalyse, welche dasselbe Ziel anstrebt. Sie spielt besonders im Internet eine Rolle, wo Nutzer unterschiedlicher Zielgruppen unterschiedliche Inhalte, Produktvorschläge oder Werbungen angezeigt bekommen sollen.

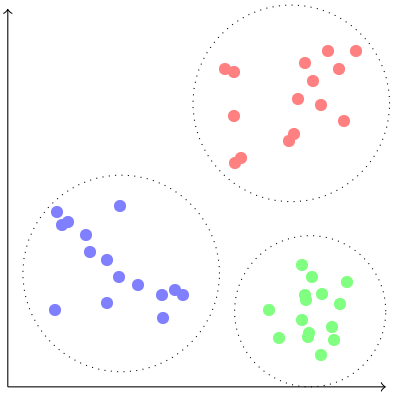

Abbildung 4: Gruppierung von Daten

In Abbildung 4 entsprechen alle Punkte unterschiedlichen Daten, gleichfarbige Punkte haben jedoch ähnliche Eigenschaften. Das neuronale Netz

soll diese Ähnlichkeiten erkennen und die Punkte gleicher Farbe jeweils in die gleiche Kategorie einteilen.

1.2.3 Bestärktes Lernen

Bestärktes Lernen dient dazu, den Entscheidungsprozess eines neuronales Netzes bezüglich einer bestimmten Tätigkeit zu trainieren.

Dabei wird mit Belohnungen gearbeitet: Je besser eine Ausgabe war, umso höher ist die Belohnung für das Netz, was darin resultiert, dass die jeweilige Entscheidung verstärkt wird. Entschied sich das Netz falsch, ist die Belohnung gering und die Entscheidung wird geschwächt. Während des Lernvorgangs wird stets versucht, die Belohnung zu maximieren, um immer die bestmögliche Ausgabe zu erreichen.

Die Belohnungen sind hierbei eine Art Feedback für das Netz, weshalb man auch von einer Art des überwachten Lernens sprechen kann.

1.3 Umgang mit neuronalen Netzen [2]

Vor der Implementation maschinellen Lernens in einem Programm muss ein neuronales Netz entwickelt werden, das die gewünschte Aufgabe zuverlässig erledigen kann. Für die einzelnen Lernarten existieren standardisierte Abläufe, die von Netz zu Netz nahezu gleich sind. Im Folgenden wird der des überwachten Lernens vorgestellt.

1.3.1 Vorbereiten eines Trainingsdatensatzes

Zuerst müssen Daten gesammelt und zu einem Trainingsdatensatz zusammengefügt werden. Bei allen Daten handelt es sich um mögliche Eingaben für das Netz. Sollen also beispielsweise Bilder von unterschiedlichen Objekten unterschieden werden, sind die Daten ebenfalls Bilder, auf denen jeweils eines der zu klassifizierenden Objekte enthalten ist.

Neben dem einfachen Zusammenstellen des Datensatzes kann es jedoch auch vorkommen, dass Daten zuerst bearbeitet werden müssen, sodass einzelne Details auffälliger sind. Das kann geschehen, indem beispielsweise ein Bild auf die relevanten Pixel reduziert wird. Dadurch lernt das Netz nicht nur besser, sondern auch schneller und mit einer geringeren Datenmenge als würden Daten nicht optimiert werden.

Sämtliche Anpassungen der Daten müssen einheitlich und auch bei der Anwendung des Netzes mit unbekannten Daten erfolgen.

Außerdem ist es für die bevorstehende Auswertung wichtig, dass ebenfalls ein Testdatensatz erstellt wird. Sein Aufbau ist gleich wie der des Trainingsdatensatzes, es sind lediglich andere Daten enthalten.

1.3.2 Trainieren

Mit dem fertigen Trainingsdatensatz kann das neuronale Netz trainiert werden. Dafür existieren unterschiedliche Trainingsalgorithmen, die sich auch zwischen besonderen Aufbauten neuronaler Netze und deren spezifischen Aufgaben unterscheiden können.

Je nach Aufgabe oder spezifischen Struktur des Netzes können die Algorithmen unterschiedlich gut abschließen. Daher ist es lohnenswert, das Training nicht nur auf einen der Algorithmen zu beschränken, sondern mehrere mit Bezug auf die durchzuführende Aufgabe zu testen.

1.3.3 Auswertung und Anwendung

Nachdem das Training abgeschlossen wurde, ist es interessant zu wissen, mit welcher Genauigkeit oder Zuverlässigkeit eine Aufgabe ausgeführt wird. Dafür kommt der zuvor erstellte Testdatensatz in Einsatz, mit dem die Anzahl der falschen bzw. unerwünschten Ausgaben ermittelt wird und der somit eine Einschätzung des Netzes möglich macht.

Wenn die Aufgabe zufriedenstellend abgeschlossen wird, können die Gewichte und die Konfiguration des Netzes in Dateien abgespeichert und in einem Anwendungsprogramm importiert. Das ermöglicht eine Rekonstruktion des neuronalen Netzes ohne großen Zeitaufwand und ohne erneut ein Training durchführen zu müssen.

2 Deep Convolutional Neural Networks [3]

Deep Convolutional Neural Networks sind besondere Formen der neuronalen Netze, die besonders in der Bilderkennung und -Klassifikation sehr häufig verwendet werden. Grund dafür ist, dass sie viel schneller und besser trainiert werden können als herkömmliche neuronale Netze. Während diese eine Genauigkeit von über 80% erst nach mehreren Trainingsepochen erreichen,

lernen Deep Convolutional Neural Networks innerhalb weniger Epochen ein Bild mit einer Wahrscheinlichkeit von über 95% korrekt zu klassifizieren.

Das ist auf deren Aufbau und Lernweise zurückzuführen, welche der des Menschen noch mehr ähnelt als die herkömmlicher neuronaler Netze.

2.1 Convolutional Layer

Verwendet man ein übliches neuronales Netz für die Bildklassifikation, ist die Relevanz zweier nebeneinanderliegender Pixel genauso hoch wie die von weit entfernten Pixeln. Das folgt daraus, dass die Ausgabe jedes Neurons an jedes Neuron der Folgeschicht propagiert wird.

In einem Convolutional Layer ist das anders: Mithilfe eines „Kernels“ werden aus nebeneinanderliegenden Pixeln Eigenschaften erkannt. Erkennbare Eigenschaften beschränken sich in den höheren Schichten auf einfache Formen wie zum Beispiel Kanten, je tiefer sich eine Schicht jedoch im Netz befindet, kann diese immer komplexere Objekte anhand von Konturen unterscheiden.

Um das zu erreichen, wird die Convolution, oder eine Vereinfachung, die häufiger in Machine-Learning-Frameworks implementiert ist, die Cross Correlation, angewandt.

2.1.1 Eindimensionale Cross Correlation

Als „Cross Correlation“ wird im Folgenden eine Operation mit einem Eingabevektor x und einem Kernel w betrachtet, aus dem ein Ausgabevektor y berechnet werden soll. Die Cross Correlation ist eine Vereinfachung der Convolution, wobei der Unterschied im Wesentlichen darin besteht, dass der Kernel bei der Convolution zuerst gespiegelt werden muss.

Mit der Bedingung

$$|x| \geq |w| \qquad |x|,\ |w| > 0$$

kann die valide Cross Correlation im eindimensionalen Raum (folgend $\odot$) mathematisch durch die Formel

$$y = x \odot w \quad \rightarrow \quad y_i = \sum_{k=0}^{|w|-1} x_{i+k} * w_k \tag{2}$$

beschrieben werden. Im Folgenden eintspricht das Symbol $\odot$ der Convolution-Operation. In Worten bedeutet das, dass der Kernel w an den Eingaben x

angelegt wird. Addiert man das Produkt der einzelnen Werte x_i und w_i, ermittelt man y_i. Um y_{i+1} zu berechnen, muss der Kernel um eine Einheit verschoben werden. Ein Sinnbild ist in Abbildung 5 dargestellt.

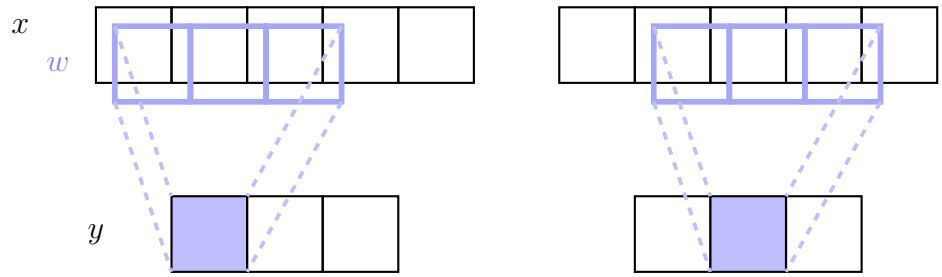

Abbildung 5: Cross Correlation im eindimensionalen Raum, ohne Padding

Es ist offensichtlich, dass die Länge des Ausgabevektors y unter diesen Bedingungen stets kleiner als die Länge der Eingabe sein wird. Ein solcher Informationsverlust von Pixeln kann jedoch zu Ungenauigkeiten führen. Daher wird ein „Padding“ eingeführt, welches den Eingabevektor x künstlich mit Nullen erweitern soll. Zur Ermittlung des Ausgabevektors y entsteht das in Abbildung 6 dargestellte Schema. Man spricht nun nicht mehr von der validen Cross Correlation, sondern von einer Cross Correlation mit „Same Padding“, bei dem der Ausgabevektor y dieselbe Länge hat wie der Eingabevektor x. Darüber hinaus existiert noch das „Full Padding“, bei dem der Eingabevektor x mit noch mehr Nullen erweitert wird. Dadurch wird die Ausgabe nach einem Convolutional Layer größer als die Eingabe, wodurch allerdings eine Vergrößerung von Daten stattfindet. Der damit verbundene Rechenaufwand ist der Grund, weshalb das Full Padding selten Anwendung findet.

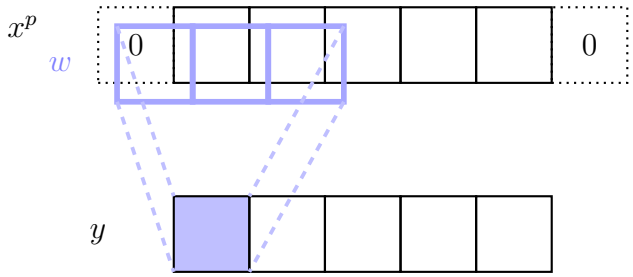

Abbildung 6: Cross Correlation im eindimensionalen Raum, mit Padding

2.1.2 Zweidimensionale Cross Correlation

Bilder sind allerdings nicht eindimensional, weshalb die zweidimensionale Cross Correlation benötigt wird.

Unter Definition der Variablen

- X^p: Eingabematrix ($\hat{=}$ Bild, einfarbig, mit Nullen erweitert)
- W: Kernel-Matrix
- Y: Ausgabematrix (Bild mit angewandtem Filter)

kann diese durch die Formel

$$Y = X \odot W \quad \rightarrow \quad Y_{i,j} = \sum_{k=0}^{|W|} \sum_{l=0}^{|W_l|} X^p_{i+k,j+l} * W_{k,j} \tag{3}$$

beschrieben werden, anhand eines Schemas und Abbildung 7 ist sie jedoch einfacher zu verstehen.

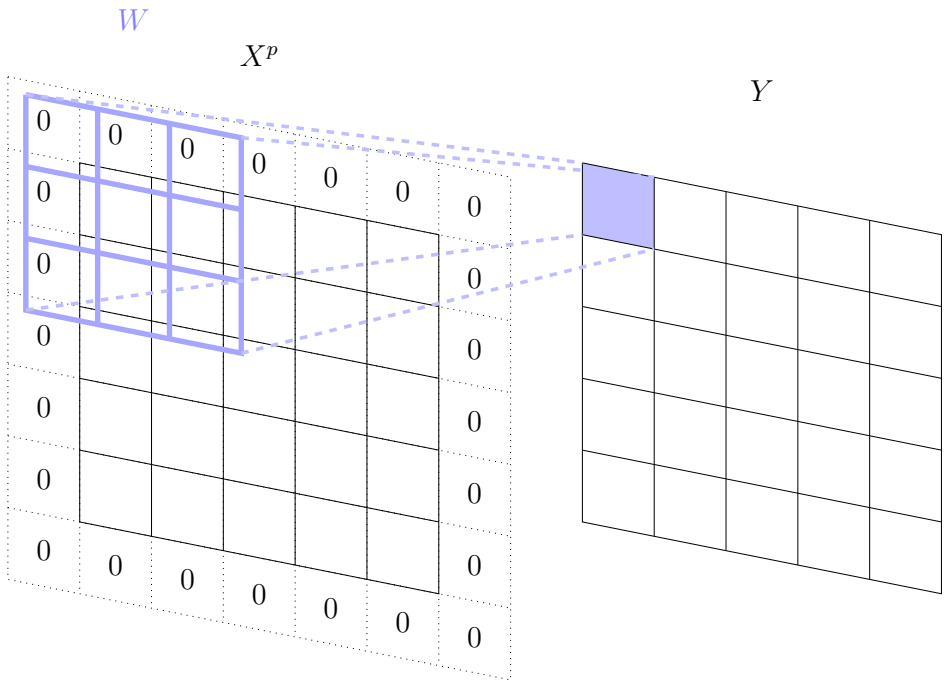

Abbildung 7: Cross Correlation im zweidimensionalen Raum

Auch hier wird die Kernel-Matrix W an die Eingabematrix X^p angelegt und verschoben, um die Werte der Ausgabe Y durch Addieren der Produkte zu berechnen. Es ist jedoch zu beachten, dass W in diesem Fall in beide Richtungen verschoben wird, sodass eine zweidimensionale Ausgabe entsteht.

2.1.3 Praktische Anwendung

In der Praxis muss ein Convolutional Layer mehrere zweidimensionale Eingabematrizen annehmen können, da beispielsweise Bilder nicht (alle) einfarbig sind. Daher werden mehrere Eingabechannel definiert, für jede Eingabematrix einer. Um beispielsweise ein Farbbild einzugeben, würden 3 Channel benötigt, da ein solches Bild in die Farben rot, grün und blau unterteilt ist.

Auch die Anzahl der Ausgabematrizen soll für jede Schicht variabel, aber konstant sein. Jede Eingabematrix hat dabei einen Einfluss auf jede Ausgabematrix.

Um diesen Einfluss zu steuern wird eine Anpassung der Dimension des Kernels benötigt. Diese sind nun die Anzahl der Ausgabematrizen, die Anzahl der Eingabematrizen, die Größe und die Breite der Gewichte, weshalb der Kernel nun als vierdimensionales Array implementiert wird.

Eine Berechnung der Ausgabematrix Y_i des Ausgabechannels i lässt sich nun formal durch

$$Y_i = \sum_{j=0}^{|X|} X_j \odot W_{i,j} \tag{4}$$

ausdrücken.

Im Anschluss an die Berechnung einer Ausgabematrix Y_i muss darauf noch die bereits aus Kapitel 1 bekannte differenzierbare Aktivierungsfunktion angewandt werden. Es entsteht mit

$$A_i = \varphi(Y_i)$$

eine sogenannte „Feature Map“. Diese enthalten in tieferen Schichten immer abstrakter werdende Eigenschaften der zu klassifizierenden Objekte, welche eine viel genauere Erkennung eines Objekts ermöglicht.

2.2 Subsampling Layer

Subsampling Layers kommen in neuronalen Netzen vor, um die Menge an Eigenschaften einer Eingabe mittels Pooling-Operationen zu verringern. Das

resultiert in einem deutlich schnelleren Trainingsvorgang, da nach einem Subsampling Layer deutlich weniger Gewichte angepasst werden müssen.

Eine Eingabe X wird in mehrere kleine Pooling-Matrizen P aufgeteilt. Ab diesem Punkt können zwei unterschiedliche Arten angewandt werden: Bei „Max-Pooling“ wird jeweils der größte Wert der Pooling-Matrix in die Ausgabe weitergeleitet, bei „Mean-Pooling“ wird der Durchschnitt aller darin enthaltenen Werte ermittelt.

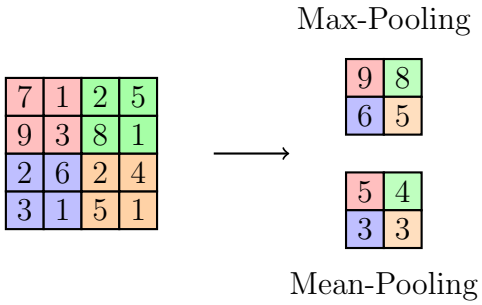

Abbildung 8: Min- und Mean-Pooling

Subsampling Layer selbst haben keine Gewichte sondern beruhen auf simplen mathematischen Berechnungen wie dem Durchschnitt oder einem Größenvergleich.

2.3 Aufbau Convolutional Neural Networks

Mit den nun bekannten neuen Schichten lässt sich ein Convolutional Neural Network zusammenstellen. Ihr Aufbau wird im Folgenden exemplarisch am Beispiel der Bilderkennung erklärt.

Zuerst wird noch außerhalb des Netzes ein farbiges Bild in seine drei Farbkomponenten rot, grün und blau aufgeteilt, sodass für jede eine einzelne Matrix entsteht mit Werten, die beschreiben, wie hoch der Anteil der entsprechenden Farbe an diesem Pixel ist.

Die dabei entstehenden Matrizen entsprechen der Eingabe des ersten Convolutional Layer. Diese Schicht führt Convolutions mit einem Kernel der Größe 3×3 oder 5×5 durch, wobei die Anzahl der Ausgabechannel meist zwischen 8 32 liegt. Die optimale Wahl dieser Anzahl ist abhängig von der auszuführenden Aufgabe und lässt sich am Besten in Tests herausfinden.

Anschließend erfolgt ein Subsampling der Ausgabematrizen mit Pooling-Matrizen der Größe 2×2, handelt es sich jedoch um größere Bilder mit größeren Details kann auch in Matrizen der Größe 4×4 aufgeteilt werden. In der Regel wird Max-Pooling verwendet, das Testen von Mean-Pooling kann sich allerdings auch lohnen.

Es folgt ein erneuter Convolutional Layer mit derselben Größe des Kernels und etwa doppelt so vielen Ausgabematrizen wie zuvor. Auch das anschließende Subsampling erfolgt mit Pooling-Matrizen derselben Größe wie in der vorherigen Subsampling-Schicht.

Nach Bedarf kann auch eine erneute Convolution mit Subsampling eingebaut werden, ist jedoch meistens nicht nötig.

Nach der letzten Subsampling-Schicht müssen alle Ausgabematrizen abgeflacht werden (engl. „Flattening“).

3 Resultate

Literatur

[1] Callan, Robert: *Neuronale Netze im Klartext.* Pearson Studium, 1. Auflage, 2003.

[2] Raschka, Sebastian und Vahid Mirjalili: *Python Machine Learning.* Packt, 3. Auflage, 2015. Kapitel 1: Giving Computers the Ability to Learn from Data.

[3] Raschka, Sebastian und Vahid Mirjalili: *Python Machine Learning.* Packt, 3. Auflage, 2015. Kapitel 15: Classifying Images with Deep Convolutional Networks.

[4] Spitzer, Manfred: *The Mind Within the Net: Models of Learning, Thinking and Acting*, Seite 21. MIT Press, 2. Auflage, 1999. `https://books.google.de/books?id=lniTl1DbRX4C&pg=0`, 23.04.2022.