Digitale Bildverarbeitung

Für die Bearbeitung der Aufgabe wurde ein neuronales Netzwerk für „Deep Learning“ implementiert. Das sog. „Deep Neural Network“ dient der digitalen Bildverarbeitung des MNIST-Datensatzes. Hierzu werden Bilder mit handgeschriebenen Ziffern des MNIST-Datensatzes von 0 bis 9 klassifiziert. Die Entscheidungen, die bei der Implementierung getroffen wurden und die Gedanken, welche für die Optimierung des neuronalen Netzwerks gemacht wurden, werden nachfolgend erläutert.

Im Gegensatz zum Standard-Netz wurde bei diesem neuronalen Netz eine Vorverarbeitung bzw. Preprocessing der Labels hinzugefügt. Dies hat folgenden Grund:

Die Idee ist es, die Daten vor dem Training für das Modell so vorzubereiten, dass das Modell effizienter mit den Daten arbeiten kann. Um dies zu erreichen wurde das Label-Format in einen binären Vektor mit der Länge 10 umgewandelt. Die Funktion One-Hot-Encoding wandelt dafür die Ziffern der Original-Labels in Vektoren um, die nur an der Stelle, die der Ziffer entspricht, eine Eins haben und sonst nur Nullen enthalten. Beispiel:

- Original-Label: 8
- One-Hot-Encoding: [0,0,0,0,0,0,0,0,1,0]

In dieser Form sind die Labels besser für die Verarbeitung durch das neuronale Netzwerk geeignet, was zur Optimierung des Netzwerks beiträgt, wie auch an den Ergebnissen deutlich beobachtet werden kann.

Des Weiteren wurde das neuronale Netz in der Architektur angepasst. Die Modelldefiniton zeigt den Aufbau des Netzwerks. Aus Gründen der Optimierung wurde das sequentielle Modell aus folgenden Schichten aufgebaut:

- Conv2D-Schichten (Convolutional Layer):
  Die Schichten führen eine Faltung auf dem Eingangsbild durch, um Merkmale aus den zweidimensionalen Eingangsbildern zu erhalten. Mit der Anzahl der Filter (gewählt wurden 32 bzw. 64) kann eingestellt werden, wie viele Merkmale die Schicht aus den Bildern erxtrahiert und im Training lernt. Im Beispiel des MNIST-Datensatzes können z.B. Kanten (z.B. Eck der Sieben) oder Muster (z.B. Kringel der Sechs) erkannt werden. Das steigert die Leistung des Netzwerks bei der Erkennung der Ziffern, da diese Muster u.a. unabhängig von der Position im Bild wiedererkannt werden können. Die Aktivierung erfolgt über Rectified Linear Unit (ReLU).
- MaxPooling2D-Schichten:
  Die Schichten sind in Kombination mit Conv2D-Schichten sehr effektiv, weshalb sie im Anschluss an solch eine Schicht angewandt werden. Die MaxPooling2D-Schicht reduziert die Größe der Merkmals-Daten der Conv2D-Schicht. Das kann als eine Art Filter gesehen werden und bewirkt einerseits die Fokussierung auf wesentliche Merkmale. Andererseits führt es dazu, dass das Modell nicht auf spezifische Merkmale oder Rauschen trainiert. Außerdem unterstützt es auch die positionsunabhängige Erkennung von Merkmalen. Weiterer positiver Effekt dieser Schichten ist, dass die Reduzierung der Datenmenge die Berechnungslast verringert und das neuronale Netzwerk dadurch effizienter wird.

- Flatten-Schicht:
  Die Flatten-Schicht wandelt die Daten (mehrdimensional), die in diese Schicht gegeben werden, in einen eindimensionalen Vektor um. Dies ist für die Verarbeitung der nachfolgenden Dense-Schichten notwendig.
- Dense-Schichten:
  Jedes Neuron der Dense-Schichten (gewählt wurden z.B. 256 Neuronen) ist mit allen Neuronen aus den vorherigen Schichten verbunden. In den Dense-Schichten geschieht der eigentliche Lern-/ Trainingsprozess. Die Dense-Schicht knüpft sozusagen die Synapsen zwischen den Neuronen, um eine Analogie zur Biologie herzustellen. Dies geschieht durch das Anpassen der Gewichtungen und Parameter der Verbindungen einzelner Neuronen. Dadurch kann das Modell komplexe Zusammenhänge in den Daten erfassen. Die Dense-Schichten werden mit der Rectified Linear Unit (ReLU) aktiviert.
- Dropout-Schicht:
  Die Dropout-Schicht deaktiviert zufällig Neuronen. Dies hat den Effekt, dass das neuronale Netzwerk nicht zu stark auf die Trainingsdaten abgestimmt wird und lernt nicht zu stark auf bestimmte Neuronen angewiesen zu sein. Das soll die Robustheit gegenüber Ausfällen stärken und die Generalisierungsfähigkeit des Modells verbessern, sodass das Netz auch auf neuen Daten gut funktioniert. Mit dem Parameter stellt man die Wahrscheinlichkeit für ein Neuron ein, mit der das Neuron deaktiviert wird.
- Dense-Schicht Ausgangsschicht):
  Die Ausgangs-Dense-Schicht hat nur 10 Neuronen die den 10 Klassen (Ziffern) entsprechen. Die Ausgangsschicht ist für die Klassifizierung zuständig und wird mit Softmax-Funktion aktiviert.

Für das Optimieren des Netzes während des Trainings wurden verschiedene Verfahren gewählt:

- Als Optimierer wird der Adam-Optimizer verwendet. Warum habe ich mich für den Adam-Optimizer entschieden?

  Der Adam-Optimierer verwendet das Verfahren des stochastischen Gradienten-Abstiegs. Das bietet den Vorteil, dass eine adaptive Lernrate für jedes Gewicht möglich. Das soll die Konvergenz des Modells optimieren, da Gewichte mit kleinen Gradienten schneller konvergieren als Gewichte mit großen Gradienten. Auch ist der Adam-Optimierer für große Datensätze geeignet und steigert damit die Effizienz des neuronalen Netzwerks. Die Lernrate des Optimierers wurde auf 0,001 eingestellt, da die Lernrate somit nicht zu groß, aber auch nicht zu klein ist. Eine kleine Lernrate lässt das Modell langsam, jedoch stabil und mit konstantem Tempo konvergieren. Eine große Lernrate lässt das Modell schnell konvergieren, allerdings kann es passieren, dass das Modell dadurch überschießt oder oszilliert. Aus diesem Grund wurde mit der Lernrate 0,001 ein Kompromiss gewählt, der gute Ergebnisse erzielt.

- Als Loss bzw. Verlustfunktion wurde CategoricalCrossentropy gewählt. Warum habe ich mich für diese Verlustfunktion entschieden?

Als Verlustfunktion wurde CategoricalCrossentropy gewählt, da die Verlustfunktion speziell für Klassifizierungen mit mehreren Klassen konzipiert ist. CategoricalCrossentropy ist effizient und bietet eine gute Softmax-Kompatibilität. Dadurch können Wahrscheinlichkeiten für Klassen interpretiert werden. Die zusätzliche Fokussierung auf die größte Wahrscheinlichkeit führt dazu, dass das Modell durch CategoricalCrossentropy in den Vorhersagen besser wird.

- Als Metriken wurden, zusätzlich zur Standard-Metrik Accuracy, noch Precision und Recall verwendet. Warum habe ich mehrere Metriken verwendet und warum genau diese?

  Mehrere Metriken sind sinnvoll, um die Leistung des Modells detaillierter zu bewerten. Auch können durch den Einsatz mehrerer Metriken verschiedene Aspekte, die die Leistung des Netzwerks bestimmen, betrachtet werden. Auch für die Anpassung der Hyperparameter und die Optimierung des Modells ist es hilfreich mehr Informationen über das Modell und seine Leistung zu haben.
  Die Accuracy-Metrik gibt die Genauigkeit des Modells an. Hierfür wird nur der Prozentsatz der richtig klassifizierten Samples betrachtet.
  Durch die Precision-Metrik kann die Präzision der Vorhersagen des Modells bewertet werden. Die Precision-Metrik misst den Prozentsatz der korrekt positiven Vorhersagen im Vergleich zu allen tatsächlich positiven Vorhersagen.
  Die Recall-Metrik bezieht sich im Gegensatz zur Precision-Metrik auf die Sensitivität der Vorhersagen. Sie misst den Prozentsatz der korrekt positiven Vorhersagen im Vergleich zu allen tatsächlich positiven Instanzen.

Für das Training des Modells werden verschiedene Einstellungen gewählt.

- Für das Training wurden 10 Epochen gewählt. Warum habe ich genau 10 Epochen gewählt?

  Aufgrund der Wahl der Hyperparameter und der Einstellung der Lernrate auf 0,001 ist es notwendig, dass das Modell mehrere Epochen trainiert. Wird die Zahl der Epochen verringert, kann das Modell nicht konvergieren und man erreicht schlechte Ergebnisse in Genauigkeit, Präzision und Sensitivität. Ebenso ist eine deutlich höhere Anzahl an Epochen nicht förderlich für die Leistung des Netzes, da das Training über viele Epochen ressourcenaufwändig ist und viel Zeit beansprucht. Außerdem sinken Trainingsfortschritt und Optimierung, die pro Epoche erzielt werden mit jeder weiteren Epoche, d.h. das Kosten-Nutzen-Verhältnis wird von Epoche zu Epoche schlechter. Um also die Effizienz des Modells nicht zu reduzieren, wurde die Zahl der Epochen für das Training auf 10 eingestellt. Mit einer Epochenzahl von 15 kann ebenfalls gut gearbeitet werden, wie weitere Untersuchungen gezeigt haben.

- Die Batch-Größe wurde für das Training auf 128 gesetzt. Warum habe ich diese Batch-Größe gewählt?

Die Batch-Größe ist mit 128 weder besonders groß noch besonders klein. Auch das liegt daran, dass hier eine Kompromiss-Lösung gewählt wurde, welche perfekt auf das Modell zugeschnitten ist. Da die Batch-Größe im mittleren Bereich liegt ist das Verhältnis aus Recheneffizienz und Speichereffizienz sehr gut. Ebenso ist die Batch-Größe auf das Zusammenspiel mit den Hyperparametern abgestimmt, sodass die Batchgröße bei gegebenen Hyperparametern eine schnellere Konvergenz und Anpassung der Gewichte pro Epoche ermöglicht. Aus diesem Grund wurde die Batch-Größe von 128 gewählt.

Ergebnisse

- Die durchschnittlichen Ergebnisse, die mit diesem Netzwerk erzielt werden können, sind:

  99,7% Genauigkeit bei 99,2% Validierungsgenauigkeit

- Das beste Ergebnis, das mit diesem Netzwerk erzielt wurde, ist:

  99,92% Genauigkeit bei 98,3% Validierungsgenauigkeit