Digitale Bildverarbeitung

Aufgabe dieser Abgabe ist es ein neuronalen Netzes zur Bilderkennungzu erstellen, trainieren und evaluieren. Das MNIST-Datenset dient als Grundlage, bestehend aus 28x28 Pixel großen Bildern der Ziffern 0-9. TensorFlow, eine Plattform von Google, wird für die Erkennung und Zuordnung der Ziffern genutzt.
Die Projektarbeit konzentriert sich auf das Design, Training und das Testen eines neuronalen Netzes zur Erkennung und Vorhersage handgeschriebener Ziffern (0 bis 9) aus dem MNIST-Datensatz. Die Keras-Bibliothek in Verbindung mit Tensorflow ermöglicht das Laden und Trainieren des Netzwerks auf 60.000 Daten, gefolgt von der Validierung auf 10.000 Daten. Die Bewertung erfolgt anhand von Genauigkeit (Accuracy) und Abweichung (Loss) im Vergleich zu den Soll-Daten.

Für die Aufgabe wir ein CNN (Convolutional Neural Network) benutzt. Dies ist eine Art von neuronalen Netzen die sich besonders für die Bildverarbeitung eignet. Hierbei werden verschiedene Schichten wie Convolutional Layer, Pooling Layer und Fully Connected Layer benutzt. Die Convolutional Layers setzen Filter ein, um Merkmale wie Kanten und Formen zu extrahieren, während die Pooling Layers die räumliche Dimension der Daten reduzieren.

Ziel ist eine möglichst hohe Genauigkeit (Accuracy) zu erreichen, aber das Modell nicht zu overfitten. Overfitting bedeutet, dass ein Modell zu stark auf Trainingsdaten spezialisiert ist und bei neuen Daten Schwierigkeiten hat, zu generalisieren. Da das Hauptziel des Trainings die Vorhersagbarkeit für neue Daten ist, sollte man Overfitting vermeiden.

Grundsätzlicher Aufbau:

Import von allen nötigen Bibliotheken:
Alle nötigen Bibliotheken werden importiert, um eine effiziente Entwicklung und Analyse zu ermöglichen.

Daten laden und Preprocessing
Der Datensatz MNIST wird geladen und in den Variablen x_train, y_train, x_test und y_test gespeichert. Dabei sind die x-Daten jeweils die Bilddaten mit 28x28 Pixeln und die y-Daten die jeweilige Ziffer, die zum Bild gehört. Zusätzlich wird eine Data Augmentation auf die Trainingsdaten angewendet

CNN Modell:
Dann wird das CNN (Convolutional Neural Network) erstellt. Dieses besteht aus vielen verschiedenen Layern, die im Folgenden näher betrachtet werden.

Conv2D:Convolutional Layer mit 32 Filtern 3x3 Kernel [1] und Relu Aktivierungsfunktion
MaxPooling2D

Flatten: Für die Umwandlung in einen eindimensionalen Vektor
Dense: Dense-Schicht mit 128 Neuronen und ReLU-Aktivierungsfunktion
Dropout: 50% Dropout Rate

Kompilieren des Netzes:
Nach der Erstellung des CNN Modells wird es nun mithilfe des Adam-Optimizers kompiliert. Die Verlustfunktion lautet “categorical_crossentropy” und wird verwendet, um zwischen Vorhersagen und tatsächlichen Labels zu messen. Als Metric wird Accuracy genommen.

Trainieren des Modells:
Das Modell wird nun mit den vorverarbeiteten und augmentierten Trainingsdaten trainiert.Training mit 10 Epochs und einer Batch Size von 64.

Evaluation:
Abschließend wird das Modell auf die Testdaten evaluiert und der Testverlust sowie die Testgenauigkeit ausgegeben.

Optimierungen:

Datenladen und Preprocessing:
Durch die Normalisierung der Bilddaten auf den Bereich zwischen 0 und 1 wird die Skalierung der Features verbessert.
One-Hot-Encoding: Wird verwendet, um die Klassenlabels als Vektoren zu repräsentieren. Dies ist notwendig, da neuronale Netzwerke in der Regel Klassen mit numerischen Werten besser verarbeiten können.

Data Augmentation:
Data Augmentation heißt, dass die Trainingsdaten künstlich variiert werden. Dies soll zu einer erhöhten Robustheit des Modells gegenüber Variationen in den Eingabedaten führen. Im Netz wurde der “ImageDataGenerator” auf Trainingsdaten mithilfe von verschiedenen Transformationen eingesetzt. [4]

Benutzen eines CNN Modells:
Convolutional-Schichten: Diese Schichten sind speziell für die Extraktion von Merkmalen aus Bildern geeignet. Sie ermöglichen es dem Netzwerk, lokale Muster zu lernen.
Max-Pooling-Schichten: Max-Pooling reduziert die räumliche Dimension der Aktivierungskarten und extrahiert die wichtigsten Informationen. Dies führt zu einer Verringerung der Modellkomplexität und einer erhöhten Rechenleistung.
Dense-Schichten: Die Dense-Schichten dienen dazu, die extrahierten Merkmale zu interpretieren und Klassen zuzuweisen.

Dropout Layer:
Um Overfitting [2] zu vermeiden, werden Dropout-Layer verwendet. Im Netz wurde ein Dropout Layer mit einer Wahrscheinlichkeit von 50% eingefügt. Das heißt dass dieser Layer zufällig 50% der Neuronen während des Trainings deaktiviert. Ziel davon ist die verbesserte Konnektivität des Netzwerkes und eine bessere Generalisierung.

Anpassung der Lernrate des Optimizers:
Im Netz wurde der Adam-Optimizer mit einer angepassten Lernrate verwendet. [3] Die Anpassung der Lernrate hat zur Folge, dass das Modell effektiver konvergiert und bessere Ergebnisse erzielt.

Categorical Crossentropy:
Dies ist die geeignete Verlustfunktion für Klassifikationsprobleme mit mehreren Klassen. Sie quantifiziert den Unterschied zwischen den vorhergesagten Wahrscheinlichkeitsverteilungen und den tatsächlichen Klassenlabels.

Quellen:

[1]
https://towardsdatascience.com/deciding-optimal-filter-size-for-cnns-d6f7b56f9363

[2]
https://databasecamp.de/ki/overfitting

[3]
https://onlytojay.medium.com/mnist-cnn-optimizer-comparison-with-tensorflow-keras-163735862ecd
[4]
https://medium.com/analytics-vidhya/image-augmentation-using-keras-99072b490c72