DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

Dokumentation digitale Bildverarbeitung und Mustererkennung

Studiengang Elektrotechnik

Studienrichtung Fahrzeugelektronik

Duale Hochschule Baden-Württemberg Ravensburg, Campus Friedrichshafen

von

Nikolas Gross

Abgabedatum:	4. Januar 2024
Bearbeitungszeitraum:	25.10.2023 - 05.01.2024
Matrikelnummer:	6982337
Kurs:	TFE21-2

Ziele des neuronalen Netzes

Ziel dieses neuronalen Netzes ist es, mit möglichst wenigen Parametern und Label eine möglichst hohe Genauigkeit zu erreichen. Dazu werden zunächst die Eingabedaten ausgewählt. Um bestmögliche Ergebnisse zu erzielen, werden zwei verschiedene Auswahlverfahren betrachtet. Um die Anzahl der Parameter des Netzes zu reduzieren, wird ein Algorithmus zur Dimensionsreduktion sowie ein recht simpel aufgebautes Feedforward Netz verwendet.

Vorverarbeitung der Eingabedaten

Berechnung des mse (mean squared error)

Um den mse zwischen zwei Label zu bestimmen, wird die Funktion calculate _mse() verwendet. Zunächst werden beide Bilder in Fließkommazahlen umgewandelt, um Genauigkeitsverluste zu vermeiden. Anschließend wird elementweise die quadratische Differenz der Pixelwerte der Bilder berechnet. Im zweiten Schritt wird die Summe der quadrierten Differenzen durch die Anzahl der Labelpixel geteilt, um den mse zu erhalten. Mit dieser Methode können die eingegebenen Label gut miteinander verglichen werden, um Aussagen über Ähnlichkeiten und Unterschiede zu treffen. Ein kleiner mse deutet darauf hin, dass die Bilder sehr ähnlich sind, während ein großer mse auf viele oder große Unterschiede zwischen den Labeln hinweist.

Berechnung des ähnlichsten Labels

Um die Auswahl der Eingabedaten zu erleichtern, bietet die Funktion search _for _most _similar _input() die Möglichkeit, die zwei ähnlichsten Bilder innerhalb eines Datensatzes zu finden. Die Funktion besitzt zwei Übergabeparameter: Die Liste der Label und einen Index. Ausgehend vom übergebenen Index wird über die restliche Länge der Label-Liste iteriert. Dabei wird jeweils der mse von zwei Labeln berechnet. Sobald ein kleinerer mse als der aktuell gespeicherte gefunden wird, wird dieser Index gespeichert. Am Ende der Iteration über die Liste wird das dem Label an der Position des Index ähnlichste Bild aus der Liste entfernt. Somit kann eine geschickte Auswahl der Eingabedaten zu einer erhöhten Leistung des Netzes mit weniger Eingabedaten führen.

Auswahl der besten 10 Label

Zur Auswahl der besten Eingabedaten für das Neuronale Netz wird die Funktion searchBestImage () verwendet. Diese iteriert zunächst über jede Zahl, die in den MNIST-Daten als Eingabelabel enthalten ist. Innerhalb dieser einen Iteration wird der mse zwischen einem Label und allen anderen Labeln derselben Zahl gebildet und der mittlere quadratische Fehler jedes Labels einer Zahl zu allen anderen Label in ein Array geschrieben. Die Indizes der zu haltenden Label werden in einem Array keep_index gespeichert. Auf diese Weise werden die 10 repräsentativsten Label des Datensatzes gefunden und gespeichert.

Uniform Manifold Approximation and Projection (UMAP) Model

UMAP ist ein Verfahren zur nichtlinearen Reduktion der Dimensionalität. Bei der Dimensionsreduktion erweist es sich als das überlegene Verfahren. Es ist sehr gut bei der effizienten Erfassung von räumlichen Mustern und zellulären Nachbarschaften [1]. Die Reduktion der Dimensionalität optimiert die Einbettungen unter Berücksichtigung der topologischen Struktur der Daten[2]. Die Dimensionsreduktion hat den Vorteil, dass die Anzahl der Parameter des neuronalen Netzes reduziert wird. Durch den Einsatz von UMAP als Dimensionsreduktionstechnik wird die Anzahl der Eingabemerkmale für das neuronale Netz stark reduziert, was zu einer Verringerung der Anzahl der Gewichtsparameter im neuronalen Netz führt.

Verwendetes neuronales Netz

In diesem Kontext wird ein Feedforward-Neuronales Netzwerk mit vergleichsweise simpler Struktur verwendet. Das hat sich als vorteilhaft erwiesen, da es trotz seiner geringen Parameteranzahl gute Leistung erzielt. Der initiale Flatten-Layer akzeptiert als Parameter die Dimension der Eingangsdaten und bewirkt eine Transformation der Eingabe in ein eindimensionales Array. Im Anschluss folgt ein Dense Layer mit 128 Neuronen. Die Aktivierungsfunk-

[1] https://arxiv.org/pdf/2312.15825.pdf
[2] https://arxiv.org/pdf/2312.13141.pdf

tion dieses Layers ist sigmoid und die Gewichte sind gemäß der glorot_uniform[3]-Initialisierung festgelegt. Ein nachfolgender Dropout Layer mit einer Dropout-Rate von 0.1 dient der Prävention von Overfitting. Abschließend wird ein Output Layer als Dense Layer mit 10 Neuronen verwendet. Die Aktivierungsfunktion dieses Layers ist softmax. Auch hier erfolgt die Initialisierung der Gewichte nach dem glorot_uniform-Verfahren. Diese Schicht prognostiziert die Wahrscheinlichkeiten für die Zugehörigkeit zu den einzelnen Klassen. Sie dient insbesondere dazu, die Ähnlichkeit zu einer bestimmten Zahl im Eingabelabel vorherzusagen.

Training des Modells

Um das Modell trainieren zu können, werden zunächst die Modellparameter konfiguriert. Über die Methode marvin.compile() werden der Optimizer, der Loss und die Metrics vorgegeben. In diesem Fall wird als Optimizer Adam [4] mit einer anfänglichen Lernrate von 0.01 verwendet. Als Loss ist die Funktion sparse_categorical_crossentropy die beste Option. Die verwendete Metrik ist die Genauigkeit des Netzes. Um das Netz möglichst effizient trainieren zu können, wird zusätzlich eine reduzierte Lernrate definiert.Hierbei wird die accuracy der Validierungsdaten überwacht und die learning rate halbiert, sobald sich nach 6 Epochen die accuracy der Validierungsdaten nicht weiter verbessert hat.Die minimale learningrate beträgt hier 0,0001.

Ergebnis

Das neuronale Netz wurde mit 10 ausgewählten Labeln trainiert, was zu einer Genauigkeit von 0.9520 auf den Validierungsdaten führt. Die Anzahl der Parameter des neuronalen Netzes beträgt 2698. Durch eine geeignete Auswahl der Eingabedaten und die Verwendung des UMAP-Algorithmus als Dimensionsreduktionsverfahren konnten sehr gute Ergebnisse mit einer geringen Anzahl an Parametern und Label erzielt werden.

[3] https://keras.io/api/layers/initializers/
[4] https://keras.io/api/optimizers/adam/