Neuronales Netz

Dokumentation

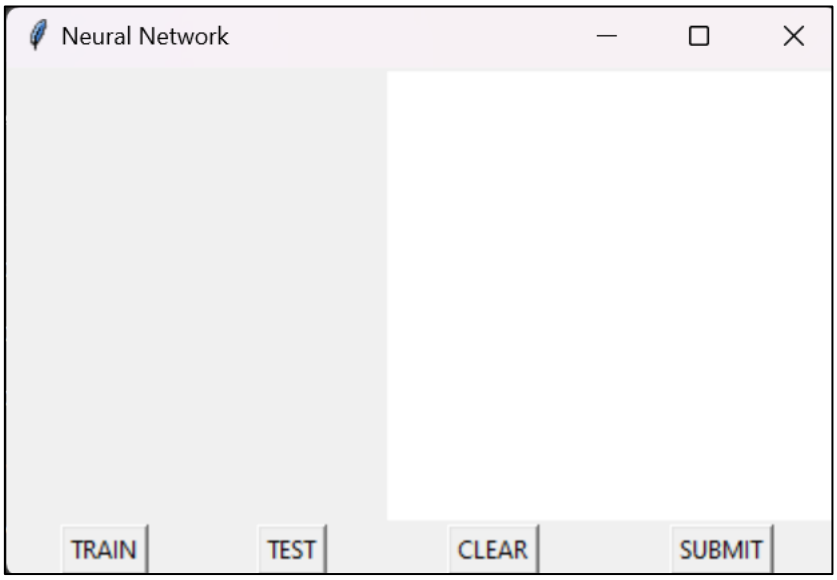
Jan Luca Emil Krüger
MTS-23

Contents

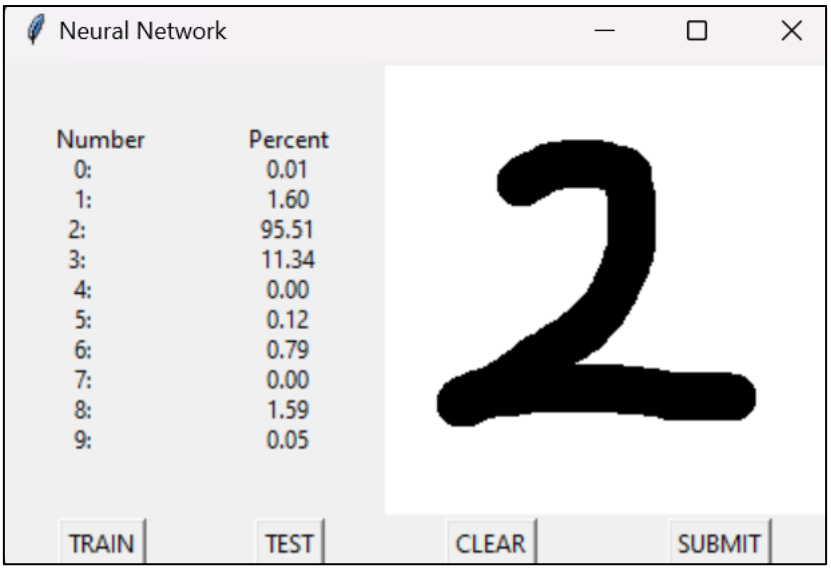
Verwenden des Tools

Das Tool kann über die Bat Datei *neuarl_net.bat* gestartet werden. Diese liegt auf der obersten Ordnerebene. Nach dem Starten dieser Datei, wird zunächst überprüft, ob eine virtuelle Entwicklungsumgebung vorhanden ist. Ist eine vorhanden, wird sie gestartet. Ist keine vorhanden, wird eine erstellt, sie gestartet und anschließend werden alle Side-Packages in diese mittels *pip* installiert. Da die Entwicklungsumgebung mit allen Side-Packages beim ersten Starten des Programms aufgesetzt werden muss, dauert das erste Starten des Tools etwas. Sollte es Probleme beim starten der Bat-Datei geben, könnte ich mir vorstellen, dass Sie eine andere Python-Version nutzen & deshalb die das „py“ in den Befehlen zu beispielsweise „python“ ändern.

Im Anschluss wird das Main-Skript des Tools, *__main__.py*, gestartet und folgende Benutzeroberfläche erscheint:



Diese Benutzeroberfläche verfügt über 6 Elemente. Oben Links befindet sich Feld, in welchem Statusmeldungen, zum Beispiel die Ausgaben des Neuronalen Netzes dargestellt werden.
Oben rechts befindet sich ein Canvas, auf welchem Sie selbst Zahlen zeichnen können, um diese durch das Neuronal Netzt zu schicken.
Unten befinden sich 4 Knöpfe. Die ersten beiden Knöpfe trainieren bzw. testen das neuronale Netz. Beachten Sie dabei, dass während des Testens bzw. Trainierens die Benutzeroberfläche nicht interagierbar ist, und sie somit warten müssen, bis das Testen (ca 1 Minute) bzw. Trainieren (ca 1 Stunde) beendet ist, um erneut mit der Oberfläche zu interagieren.
Die beiden Knöpfe unten rechts sind für das Canvas. Der linke Knopf (CLEAR) setzt den Canvas zurück, während der rechte Knopf (SUBMIT) das gezeichnete durch das neuronale Netz schickt. Nachdem Sie den rechten Knopf gedrückt haben, erscheint auf dem Statusfeld die Ausgabe des Neuronalen Netzes:



Zum Zeichnen der eigenen Ziffern ist noch anzumerken, dass es nicht 100%tig gut funktioniert, ich jedoch die Erfahrung beim Testen gemacht habe, dass Zahlen besser erkannt werden, wenn sie das Feld nicht komplett ausfüllen, das heißt, dass es nach oben, unten, rechts & links einen kleinen Rand gibt. (Wie im Beispiel oben)

Zum Trainieren ist noch anzumerken, dass die Lernrate aktuell auf 0,001 gesetzt ist. Sollten Sie einen Trainingszyklus mit einer anderen Lernrate *LEARNING_RATE* im Skript *./src/classes/user_interface.py* Zeile 23 entsprechend abändern.

Sollten Sie eigene Zahlen anhand von Pixelwerten testen wollen, können sie Ihre Datei auch in das neuronale Netz einbinden. Dafür müssen Sie nur die konstante zu der Datei *PATH_TO_TESTING_IMAGES* anpassen, sodass der Pfad zu Ihrer Datei führt. Wie ihre Daten dabei aussehen müssen, wird im Kapitel **Einlesen der Bilder** erklärt. Sollten Sie die Konstante abgeändert haben, müssen Sie einfach nur den TEST-Knopf drücken und Statusfeld wird Ihnen nach dem Testen die Genauigkeit angeben.

Einlesen der Bilder

Die Training- & Testdaten waren zunächst in Byte-Dateien gegeben. (Diese sind auch weiterhin im Programm enthalten, siehe Ordner *./src/images/idx*) Da die Labels & Pixel der Test- & Trainingsdaten in unterschiedlichen Dateien lagen, habe ich diese CSV-Dateien zusammengeführt, das heißt, es gibt nun eine Datei, die die Labels & Pixel der Trainingsdaten enthält (*training_data.csv*), sowie eine Datei, die die Labels & Pixel der Testdaten enthält (*testing_data.csv*). Beide Dateien befinden sich im Ordner *./src/images/csv*.

Beim Umwandeln der Byte-Dateien in diese CSV -Dateien war jedoch zu beachten, dass bei den Labels die erste 8 Bytes & bei den Pixel die ersten 16 Bytes übersprungen wurden, da diese nur Information über die Datei selbst enthielten und keine Informationen über die Bilder selbst. Außerdem ist zu beachten, dass die Pixelwerte im Intervall [0; 255] in den Byte-Dateien gespeichert sind, diese jedoch auf den Bereich [0; 1] konvertiert wurden. (Die Werte wurden mit 255 dividiert) Dies wurde gemacht, da die Gewichte sowie die Ausgabewerte des neuronalen Netzes ebenfalls in diesem Intervall sind.
Das Umwandeln der Bildinformationen hatte bei der Entwicklung große Vorteile, da ich so einfach auf einzelne Bilder zugreifen konnten, wenn ich bestimmte (Teil-)Funktionalitäten testen wollte.

In beiden CSV-Dateien sind die Bilder zeilenweise abgespeichert, das heißt, jede Zeile entspricht den Werten eines Objekts der selbst erstellten Klasse *Image*. Die erste Spalte jeder Zeile entspricht dabei dem Label, also der Zahl, die dieses Bild darstellen soll, während die zweite Spalte die Pixelwerte als eindimensionale Liste enthält. Die Pixelwerte sind dabei die, wie oben erwähnt, auf das Intervall [0; 1] konvertiert wurden.

Sollten Sie nun, wie oben bereits beschrieben, Bilder anhand von eigenen Pixelwerten testen wollen, müssen Sie Ihre Daten in dieser Format bringen.

Erklärung der Gewichte

Initialisierung der Gewichte

Die Gewichte wurden anhand der Xavier-Initialisierung initialisiert. Die verlinkte Quelle geht dabei auf die Herleitung und verschiedene Arten dieser Initialisierungsart ein. Zusammengefasst wurden die Gewichtsmatrizen im Intervall [-x; x] initialisiert mit $x = \frac{1}{n}$, wobei n gleich der Größe der nachfolgenden Schicht ist. Somit wurden die Werte der Gewichtsmatrix zwischen dem Input- & dem Hidden-Layer im Intervall $\left[-\frac{1}{81}; \frac{1}{81}\right]$ initialisiert, da der Hidden-Layer bei mir eine Größe von 81 Neuronen hat, während die Werte der Gewichtsmatrix zwischen dem Hidden- & dem Output-Layer im Intervall $\left[-\frac{1}{10}; \frac{1}{10}\right]$ wurden, da der Output-Layer eine Größe von 10 Neuronen hat.

Speicherung der Gewichte für die weitere Verwendung

Die Gewichtsmatrizen sind, ähnlich wie die Labels & Pixel der Bilder, in CSV-Dateien abgespeichert. Es gibt dabei 2 CSV-Dateien: *weight_matrices.csv* & *altered_weights.csv*. In der ersten Datei sind die initial erstellten Gewichte gespeichert, wie im Kapitel **Initialisierung der Gewichte** beschrieben wurde. In der zweiten Datei sind die Gewichte nach 10 Trainingszyklen gespeichert.

Beide Dateien bestehen nur aus 2 Zeilen plus einem Header. In jeder Zeile ist eine Gewichtsmatrix als eine zweidimensionale Liste gespeichert. Die erste Zeile spiegelt dabei die Gewichtsmatrix zwischen dem Input- & Hidden-Layer wieder, während die zweite Matrix die Gewichtsmatrix zwischen dem Hidden- & dem Output-Layer wiedergibt.

Genauigkeit nach Epochen

Das neuronale Netz wurde insgesamt 10 mal trainiert. Nach jedem Trainingszyklus wurde die Genauigkeit getestet. Zudem wurde die Genauigkeit vor dem ersten Trainingszyklus getestet. Somit ergeben sich folgende 11 Testresultate:

Anzahl Trainingsiterationen	Genauigkeit [%]	Trainingsrate
0	11,35	N/A
1	25,93	0,01
2	65,42	0,01
3	83,10	0,01
4	86,74	0,01
5	88,35	0,01
6	89,29	0,01
7	89,94	0,01
8	90,30	0,01
9	91,35	0,001
10	91,39	0,001

Graphisch ergibt sich folgendes: