Laborversuch 3

Versuch	**CNN, DNN, Keras, MNIST**
Fach	Int. Syst. u. Masch. Lernen (ISML)
Semester	WS 2023/24
Fachsemester	TIN5/ITS5/WIN5
Labortermine	18.01.2024
Abgabe bis spätestens	12.01.2024

Versuchsteilnehmer

Name: Vorname:

Semester: Matrikelnummer:

Bewertung des Versuches

Aufgabe:	1	2	3	
Punkte maximal:	40	30	40	
Punkte erreicht:				
Gesamtpunktezahl:	/100	Note:		Zeichen:

Anmerkungen:

Aufgabe 1: (5+10+5+5+15 (+10) = 40 (+10) Punkte)

Thema: MNIST Ziffernerkennung mit CNN/DNN und Keras

Betrachten Sie das Python-Skript `cnn_mnist.py` zum Aufbau und Training eines einfachen CNN zur Ziffernerkennung mit dem MNIST Datenset.

a) Abschnitt `(i) Load image data` lädt das MNIST-Datenset ein und trennt es in Trainings- und Testdaten bzw. Input-Bilder und Output-Klassenlabels auf. Aber was machen die Abschnitte `(ii) Normalize the images` und `(iii) Reshape the images` bzw. wozu braucht man diese Schritte? Könnte man die Schritte auch weglassen?

b) Abschnitt `(iv)` definiert die Hyperparameter des Netzwerkes und Abschnitt `(v)` baut das Netzwerk auf.

- Beschreiben Sie kurz die Bedeutung jedes Hyperparameters in Abschnitt `(iv)`.
- Skizzieren Sie das Netzwerk das aufgebaut wird (1 Box pro Layer).
- Geben Sie für jede Layer an wie groß sie ist (Anzahl Neurone) und wieviele Synapsen sie hat (ungefähre Abschätzung reicht).
- Entspricht die Anzahl "physischer" Synapsen in jeder Layer der Anzahl unabhängiger trainierbarer Parameter? Wo nicht? Wieviele "physische" Synapsen und wieviele trainierbare Parameter hat diese Layer?

c) Trainieren Sie jetzt das Netzwerk, indem Sie das Skript starten. Welche Accuracy erreichen Sie auf den Trainings- bzw. Validation-Daten für die vorgegebenen Hyper-Parameter?

d) Benutzen Sie Tensorboard um den Trainingsverlauf grafisch anzusehen. Siehe die Anweisungen in Abschnitt (xii). Machen Sie ein Bildschirmfoto.

e) Optimieren Sie jetzt die Hyper-Parameter durch Probieren (siehe Abschnitt `(iv)`).

- Welche Hyperparameter haben den stärksten Effekt auf die Accuracy?
- Was beobachten Sie für sehr kleine Minibatches?
- Gibt es bessere Optimizer als SGD?

Welche maximale Validation-Accuracy erreichen Sie nun? Geben Sie die besten Parameter an die Sie gefunden haben.

f) Zusatzaufgabe (freiwillig): Verändern Sie die Netzwerkstruktur um die Validation-Accuracy weiter zu erhöhen. (Sie können z.B. weitere CNN/MaxPool-Layers einfügen). Berichten Sie über Ihre Resultate.

Hinweise: Für zusätzliche Hinweise siehe z.B. `https://victorzhou.com/blog/keras-cnn-tutorial/` und die Keras-Dokumenation.

Aufgabe 1: (5+5+5+5+5+5 = 30 Punkte)

Thema: MNIST Ziffernerkennung mit IVISIT und Keras Integrieren Sie nun ihre CNN/DNN-Implementierung in IVISIT. Sie können dazu das (fast) fertige Python-Skript `ivisit_cnn_mnist.py` verwenden. Starten Sie wie üblich mit `python ivisit_cnn_mnist.py ivisit_cnn_mnist.db` und klicken Sie am besten gleich auf "RUN".

a) Mit den Slidern `filter_img_class` und `idx_image` können Sie sich die MNIST-Daten detailliert anschauen. Was fällit Ihnen auf? Stimmen die Ziffern immer mit deutscher Schreibweise überein?

b) Mit `Network-Parameters` und `Training-Parameters` können Sie die Hyper-Parameter einstellen. Wählen Sie gute Parameter, z.B. die aus Aufgabe 1. Lassen Sie dabei `epochs` auf einem relativ kleinen Wert (z.B. 1), damit ein Lernschritt nicht zu lange dauert. Trainieren Sie dann das Netzwerk:

   - Klicken Sie auf den Button `Build Network` um das Netzwerk zu definieren.
   - Klicken Sie dann auf `Compile Network` um das Netzwerk zu kompilieren (d.h. den Berechnungsgraph aufzubauen).
   - Klicken Sie dann auf `Train Network` um das Netzwerk zu trainieren. Sie können wiederholt klicken. Jedesmal wird das Netzwerk mit der gewählten Epochen-Zahl weitertrainiert.
   - Wieviele Trainingsschritte (bzw. Epochen insgesamt) brauchen Sie bis die Validation Accuracy über 95 Prozent ist? Siehe Textfeld `Results`.

c) Bei `Network Outputs` sehen sie die Aktivitätsverteilung der 10 Output-Neurone (=Klassenwahrscheinlichkeiten für die Ziffern 0,1,...,9). Bei korrekter Klassifikation ist die Anzeige in "grün" bei falscher Klassifikation "rot". Finden Sie für die Klassen "3" und "7" jeweils 4 Beispiele aus den TEST-Daten die falsch klassifiziert werden (Bildschirmfoto!). Sind die Fehler nachvollziehbar?

d) Zeichnen Sie jetzt in `new_input` 10 mal die Ziffer "3". Welche Accuracy erreicht das Netzwerk auf Ihren 10 Inputs?

e) Wählen Sie nun eine Input "5" der gut erkannt wird. Spielen Sie mit `Drawing-Parameters` um zu testen wie sehr man den Input verändern muss bis er falsch wird.

   - Um wieviel Grad müssen Sie rotieren bis der Input nicht mehr als "5" erkannt wird?
   - Um wieviele Pixel müssen Sie nach links oder rechts verschieben bis die "5" falsch klassifiziert wird?
   - Wieviel Prozent Noise-Pixel (`p_noise`) verträgt der Input bis ein Fehler entsteht?

f) Zu Adversary Sampling: Verändern Sie nun die "5" von Hand bis das Netzwerk eine "6" erkennt? Wählen Sie *linewidth=1* um einzelne Pixel gezielt zu ändern. Mit welcher minimaler Anzahl veränderter Pixel können Sie das Netzwerk "reinlegen"?

Aufgabe 3: (40 Punkte)

Thema: CIFAR10 Objekterkennung mit DNN/VGG und Keras

Betrachten Sie das Python-Skript `cnn_cifar10.py` zum Aufbau und Training eines VGG-DNN zur Objekterkennung auf Bildern mit dem CIFAR10 Datenset. Versuchen Sie ähnlich wie in Aufg.1 die Validierungs-Accuracy zu maximieren. Sie dürfen alle Hyper-Parameter (und wenn Sie wollen auch die Netzwerkstruktur) anpassen. Welche Accuracy erreichen Sie? Für welche Hyper-Parameter?

Hinweise: Für zusätzliche Hinweise siehe z.B. `https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-`
und die Keras-Dokumenation.