Labor 4

Keywords: MNIST, Keras, Dense-Layer, Optimizer

1) Kopieren Sie die Datei „MNIST.py“ in ihren Workspace.
   Verwenden Sie dazu am besten die Kommandozeile, der Befehl „cp“ dient zum kopieren.

   z.b: cp MNIST.py workspace

   kopiert die Datei MNIST.py (wenn sie im aktuellen Verzeichnis liegt) in Ihren workspace
2) Importieren Sie den MNIST-Datensatz aus der Keras Bibliothek (Tipp: Siehe letztes Labor)
   Gehen Sie dabei ähnlich vor wie im letzten Labor.

   Doku: https://keras.io/datasets/#mnist-database-of-handwritten-digits

   Der MNIST Datensatz ist einer der bekanntesten Datensätze im Deep-Learning Umfeld. Er wird häufig als das „Hello World“ im Deep-Learning gehandelt. Er beinhaltet ein-dimensionale (graustufen) Bilder von Zahlen, die handschriftlich erstellt wurden. Yann Lecun trainierte mit diesem Datensatz „das erste“ Deep-Learning Netzwerk. Das „LeNet“ wurde verwendet, um Postleitzahlen automatisch zu erkennen. In diesem Labor werden Sie das LeNet noch nicht nachbauen, stattdessen werden sie ein mehrschichtiges Neuronales Netz unter der Verwendung von Fully Connected Layer verwenden um den Datensatz zu trainieren.
3) Schreiben Sie eine Schleife in der die ersten 10 Bilder aus x_train mit der Hilfe von pyplot in 10 unterschiedlichen figures angezeigt werden. Sorgen Sie dafür, dass die angezeigten Bilder in der Graustufendarstellung angezeigt werden

   Tipp: cmaps,…

4) Visualiseren Sie die Verteilung der Targets in den Validations und Testdaten. Sieht diese Verteilung okay aus?
5) Erstellen Sie ein Mittelwertbild über die gesamten Trainings-Bilder. Dies können Sie mit Hilfe der Numpy-Bibliothek und dem darin enthaltenen „mean“ Befehl erreichen. Ihre Trainingsbilder bestehen momentan aus 3-Dimensionen: (N,X,Y) . Mitteln Sie über die N-Achse. Der Mean-Befehl kann eine Achse als Übergabeparameter entgegennehmen.

   Das Mittelwertbild eines Datensatzes ist eine weitere Möglichkeit den Datensatz genauer zu analysieren, um festzustellen ob es statistische Ausreißer innerhalb des Datensatzes gibt.
6) Mit der Hilfe von Boolean-Indexing können Sie explizit auf Elemente in einem Numpy-Array zugreifen, auf die eine bestimmte Bedinung zutrifft. So können Sie zum beispiel aus y_train alle Elemente herausbekommen, die die Klasse „0“ sind in dem Sie den folgenden Befehl ausführen:

Wie zu erwarten, besteht das Array nur aus „0“en.

Schreiben Sie eine Schleife, in der Sie über die 10 unterschiedlichen Klassen iterieren. Pro Schleifendurchgang / Klasse visualisieren Sie mit der Hilfe von Pyplot folgende Dinge:

```
In [55]: y_train[y_train ==0]
Out[55]: array([0, 0, 0, ..., 0, 0, 0], dtype=uint8)
```

a. Ein Bild aus der Klasse von dem das Mittelwertbild subtrahiert wurde (Bild – Mittelwertbild)
b. Die Verteilung der Pixel im Original Bild
c. Die Verteilung der Pixel aus dem Bild von dem das Mittelwertbild subtrahiert wurde

Was können Sie erkennen in den Verteilungen?

Tipp: Verwenden Sie die „hist“ Funktion von Pyplot zur Analyse der Pixel. Denken Sie daran den .flatten() Befehl auf dem Bild auszuführen um einen 1-D Vektor an die hist Funktion zu übergeben

7) Der folgende Aufruf erstellt einen Vektor der nun die Bilder in einem 1-D Vektor gespeichert hat über die gesamten Trainingsdaten:

```
flat_input_train = np.reshape(x_train,(len(x_train),-1))
```

a. Subtrahieren sie von flat_input_train das Mittelwertbild
b. Wiederholen Sie beides für die Tesdaten

8) In diesem Abschnitt werden wir Keras mit der sequenziellen API verwenden. D.h. es können keine Verzweigungen in der Netzwerkarchitektur eingebaut werden.

Doku : https://keras.io/getting-started/sequential-model-guide/

Bevor Sie weiter vorgehen vergewissern Sie sich, dass die Größe von flat_input_train (60000, 784) ist.

Machen Sie sich vertraut mit den aufrufen von Keras und erstellen Sie ein neuronales Netz nach ihren Wünschen.

9) Verwenden Sie den „to_categorical“ Befehl aus der Keras Bibilothek um eine „one-hot-encoding“ auf den Trainings- und Testdaten zu erhaltne. Dies dient dazu die Klassenwerte von 0-10 in eine binäre Darstellung zu überführen.

10) Der „compile“ (https://keras.io/models/model/#compile) Aufruf ist dazu da, die folgenden Parameter (und noch weitere) einzustellen:
- Optimizer
- Loss
- Metrics

Der „fit“ (https://keras.io/models/model/#fit) Aufruf trainiert letztendlich das Model und dient dazu folgende Parameter (und noch mehr) zu übergeben:

- Trainingsbilder

- Targets
- Validationsbilder
- Vaildationstargets
- Anzahl an Epochen

Machen Sie sich mit den Parametern und der Doku vertraut

11) Der „Fit“ Aufruf gibt ein „history“ objekt zurück, dies beinhaltet Informationen über die „Geschichte“ des Traninings. Dieses objekt beinhaltet ein Dictionary mit dem namen „history“. Dies beinhaltet die unterschiedlichen Losse, sowie die gewählten Metriken.
In dem unten dargestellten Code-Abschnitten können Sie erkennen, wie diese visualisiert werden. Außerdem können Sie am Ender der Schleife 4 Aufrufe sehen. Diese dienen dazu den GPU Speicher wieder zu leeren.

```
optimizers_to_test = ["rmsprop"]
for optimizer in optimizers_to_test:
    model = create_model()

    model.compile(optimizer='rmsprop',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    hist = model.fit(flat_input_train,y_train_cat,validation_data=\
              (flat_input_test,y_test_cat),epochs=10)

    plt.figure(999)
    plt.plot(hist.history["loss"])
    plt.title("Training Loss")

    plt.figure(998)
    plt.plot(hist.history["val_loss"])
    plt.title("Validation Loss")

    plt.figure(888)
    plt.plot(hist.history["acc"])
    plt.title("Trainings Accuracy")

    plt.figure(887)
    plt.plot(hist.history["val_acc"])
    plt.title("Validation Accuracy")

    del hist
    del model
    K.clear_session()
    gc.collect()
```

Erweitern Sie die Liste „optimizers_to_test“ um weitere optimizer (https://keras.io/optimizers/)

und führen Sie die Schleife aus. Was können Sie erkennen?

Speichern Sie die Bilder ab, in dem Sie entweder den savefig Befehl von Pyplot verwenden oder per rechtsklick auf die figure in der Konsole klicken und diese dann abspeichern.

12) Wiederholen Sie die Trainingsdurchläufe mit den Bildern von denen das Mittelwertbild abgezogen wurde. Was können Sie erkennen?

13) Die Accuracy gibt die Genauigkeit an, was bedeutet bei 10 Klassen eine Genauigkeit von 0.1, was bedeutet bei 2 Klassen eine Genauigkeit von 0.5?