HAW HAMBURG

Praktikum 4

Aufgabe 1 (Erkennung von Sprachkommandos)

Hier soll der Sprachkommando-Datensatz aus Praktikum 3 mit einem Faltungsnetz untersucht werden.

(a) Speichern Sie die Spektrogramm-Beträge des Trainings- und Testdatensatzes jeweils in einem 4-dimensionalen Array der Größe `[n1, n2, 1, N]`, wobei `(n1, n2)` die Matrixgröße des Spektrogramms und `N` die Größe des Datensatzes ist.

(b) Erstellen Sie ein eigenes Faltungsnetz zur Klassifikation der Sprachkommandos. Das Netz soll wenigstens 3 Faltungsblöcke, jeweils bestehend aus Conv-, Batchnorm-, Relu und evtl. Pooling-Schichten, enthalten. Schaffen Sie eine Testfehlerrate von unter 4%?

Hinweis: Bauen Sie erstmal ein eigenes Netz. Sie können sich dann aber auch gerne an dem Netz am Ende des Aufgabenblatts orientieren.

(c) Nehmen Sie ein eigenes Sprachkommando auf und lassen Sie es von Ihrem Netz klassifizieren. Geben Sie die Wahrscheinlichkeiten für alle vier trainierten Klassen aus, um zu sehen, wie sicher sich das Netz ist.

Hinweis: Siehe `audiorecorder` in der Matlab - Doku. Ein Signal fester Länge kann man mit der Funktion `recordblocking` aufnehmen.

Aufgabe 2 (Klassifizierung von Galaxien)

Der Ordner `Daten/Galaxien` enthält Bilder von 3 Typen von Galaxien: elliptisch, spiralförmig und irregulär. Die automatische Klassifizierung dieser Galaxien ist eine wichtige Aufgabe bei der Analyse von Daten, die bei großen Teleskopen anfallen.

(a) Laden Sie die Daten mit der `imageDatastore`-Funktion, sehen Sie sich einige Bilder inklusive Klassenlabel an und teilen Sie den Datensatz in 80% zum Trainieren und 20% zum Validieren auf (*Hinweis:* benutzen Sie die Matlab Funktion `splitEachLabel` für diese Aufspaltung).

(b) Bauen Sie das Faltungsnetz aus der Veröffentlichung *Deep Galaxy: Classification of Galaxies based on Deep Convolutional Neural Networks* (Khalifa et.al., https://arxiv.org/pdf/1709.02245) nach. Welche Testfehlerrate können Sie erreichen? Lässt sich die Fehlerrate durch die Technik des 'Image Augmentation' verbessern? (*Hinweis:* Sollte das Netz nichts lernen, fügen Sie `batchNormalizationLayer` nach den vollverbundenen und den Faltungs-Schichten ein.)

(c) Können Sie eine eigene Architektur finden, die die Testfehlerrate verbessert?

Aufgabe 3 (Erkennung von Blumenarten)

Im Ordner `Daten/Blumen` befinden sich ein Trainings- und ein Testdatensatz, der aus Bildern bestehen, die 10 verschiedene Blumenarten zeigen. Dieser Datensatz soll mit dem `googlenet` und Transfer-Learning analysiert werden.

(a) Laden Sie den Datensatz und spalten Sie einen kleinen Teil (z.B. 20%) vom Trainingsdatensatz zur Validierung des Trainingsprozesses ab. Sehen Sie sich einige Bilder inklusive Klassenlabel an.

(b) Laden Sie das `googlenet` und ersetzen Sie die 3 letzten Schichten durch für den gegebenen Datensatz passende Schichten.

(c) Trainieren Sie das Netz und benutzen Sie den Testdatensatz zur Bestimmung des Testfehlers, indem Sie bei gleichen Trainingsbedingungen:

(i) alles bis auf die neuen Schichten einfrieren,

(ii) die drei ersten Inception-Blöcke (d.h. die ersten 52 Schichten) einfrieren,

(iii) nichts einfrieren.

Notieren Sie die Testfehlerraten und Trainingszeiten für diese drei Fälle und vergleichen Sie.

Aufgabe 4 (Pulsbreite eines Femtosekundenlasers)

Hier geht es noch einmal um den Datensatz aus Aufgabe 3 vom 3. Praktikum.

(a) Entwickeln Sie ein Faltungsnetz zur Bestimmung der Pulsbreite. Versuchen Sie, auf dem Testsatz einen möglichst kleinen 'Root-Mean-Squared-Error' (RMSE) zu erreichen. Kommen Sie auf einen Wert von unter 3 Femtosekunden?

(b) Wenden Sie Ihr bestes Faltungsnetz auf die Spektrogramme im Ordner `Daten/FrogTraces2` an. Erstellen Sie eine csv-Datei mit zwei Spalten. Die erste Spalte soll die Dateinamen der Spektrogramm-Dateien enthalten und die zweite Spalte die mit Ihrem Faltungsnetz ermittelten Pulsbreiten. Schicken Sie diese csv-Datei per Email an klaus.juenemann@haw-hamburg.de. Der beste RMSE-Wert wird mit einem Preis prämiert.

Ein mögliches Netz zur Audioklassifikation:

1	imageinput 64×63×1 images with 'zerocenter' normalization
2	conv_1 12 3×3 convolutions with stride [1 1] and padding 'same'
3	batchnorm_1 Batch normalization
4	relu_1 ReLU
5	maxpool_1 3×3 max pooling with stride [2 2] and padding 'same'
6	conv_2 24 3×3 convolutions with stride [1 1] and padding 'same'
7	batchnorm_2 Batch normalization
8	relu_2 ReLU
9	maxpool_2 3×3 max pooling with stride [2 2] and padding 'same'
10	conv_3 48 3×3 convolutions with stride [1 1] and padding 'same'
11	batchnorm_3 Batch normalization
12	relu_3 ReLU
13	maxpool_3 3×3 max pooling with stride [2 2] and padding 'same'
14	conv_4 48 3×3 convolutions with stride [1 1] and padding 'same'
15	batchnorm_4 Batch normalization
16	relu_4 ReLU
17	conv_5 48 3×3 convolutions with stride [1 1] and padding 'same'
18	batchnorm_5 Batch normalization
19	relu_5 ReLU
20	maxpool_4 8×1 max pooling with stride [1 1] and padding [0 0 0 0]
21	dropout 20% dropout
22	fc 4 fully connected layer
23	softmax softmax
24	classoutput crossentropyex