Gesamtdokumentation

Inhaltsverzeichnis

Allgemein

Projektmitglieder

Name	Matrikelnummer
Sören Prieß	670266
Bjarne Richter	670273
Salar Haji Ibrahim	670366
Yazan Darwisheh	670693
Marik Malachewski	670298
Finn-Niklas Rathjen	670165

Aufgabenverteilung

Name	Aufgaben
Sören Pries	Vorbereitung Gesamtprogramm, Step1 Programmierung, Step1 Dokumentation, Step3 Programmierung, Anwenderdokumentation, Gesamtdokumentation
Bjarne Richter	Step1 Programmierung
Salar Haji Ibrahim	Bearbeitung Bilder
Yazan Darwisheh	Bearbeitung Bilder
Marik Malachewski	Step2 Programmierung
Finn-Niklas Rathjen	Vorbereitung Gesamtprogramm, Step2 Programmierung , Aufbereitung Gesamtdokumentation,

	Anwenderdokumentation, Step3 Programmierung

Accuracy der einzelnen Schritt(gerundeter)

Name	Beim Training	Beim Test
Step 1	94%	46%
Step 2	88%	55%
Step 3	90%	70%

Schritt 1:

Wir haben für Schritt 1 die Skripte Step1Train, Step1Test und Step1Main.

Vorbereitung

Wir haben mithilfe des Skriptes „script1_PlateGenerator.m“ als erstes 20 Plates für andere Kennzeichen erstellt. Anschließend haben wir den Code zum Erstellen folgendermaßen abgeändert: „`is = [6 12 randi([42 47],1) randi([48 63],1) randi([1 26],[1 2]) 64 randi([30 39],[1 3])];`“.
Dadurch ist es und möglich 20 zufällige Kennzeichen aus der Region Flensburg zu erstellen.

Anschließend haben wir erneut den Code folgendermaßen geändert: „`is =[19 12 randi([42 47],1) randi([48 63],1) randi([1 26],[1 2]) 64 randi([30 39],[1 3])];`“
Nun ist es uns möglich mit dem Code 20 zufällige Kennzeichen aus der Region Schleswig zu erstellen.

Nachdem wir insgesamt die 60 Kennzeichen erstellt haben, war es uns möglich mit dem Skript „resizeImages.m“ die Kennzeichen auf die richtige Größe zu formatieren (200x50x3). Die formatierten Bilder sind in dem Ordner „platesResize“ untergebracht, welcher für jede Kategorie (Andere, FL, SL) einen Unterordner enthält.

Anschließend, nachdem wir erfolgreich die Kennzeichen erstellt und formatiert hatten, haben wir ein neuronales Netz erstellt und dieses angefangen zu trainieren.

Step1Train:

In diesem Skript wird zuerst ein ImageDatastore (Zeile 29) erstellt, welches als Inhalt die zuvor erstellten Kennzeichen, welche in dem Ordner „platesResize“ liegen, erhält. Anschließend haben wir die Bilder aus dem ImageDatastore in Trainingsdaten und Validationsdaten zufällig aufgeteilt (Z.30). Dabei erhalten unsere Trainingsdaten (imageDStrain) 70% der Bilder und unsere Validationsdaten (imageDSvalidate) 30%.

In den folgenden Zeilen (Zeile 33 – 48) definieren wir das neuronale Netz. Dabei erhält das Netz unteranderem die Parameter, dass die Inputdaten von der Größe 200x50x3 sein müssen.

Nachdem wir unser Neuronales Netz definiert haben, definierten wir die Optionen für das Training (Zeile 52-57). Dabei haben wir als Trainingsoption sgdm (stochastic gradient descent with momentum) verwendet. Die maximale Anzahl an Durchläufen haben wir auf 20 gestzt (MaxEpochs = 20), für die Validation unserer Trainingsdaten haben die zuvor bestimmten Validationsdaten (imageDSvalidate) verwendet. Die Validierungsfrequenz liegt bei 5, sodass immer nach 5 Epochen

validiert wird. Durch „verbose, true“ haben wir dafür gesorgt, dass der Trainingsfortschritt in der Kommandozeile ausgegeben wird. Anschließend haben wir die Option „Plots“ auf „training-progress“ gesetzt, um anzugeben, dass der Trainingsprozess als Fenster ausgeben werden soll.

Wir haben festgestellt, dass wenn wir mehr als 20 Epochen durchlaufen lassen, bei einer Validierungsfrequenz von 5, ein Risiko besteht. Es kann dazu führen, dass das Neuronale Netz übertrainiert wird. Dies wirkt sich negativ auf die Validation Genauigkeit aus.

Nachdem wir die Optionen für das Training festgelegt haben, können wir mit dem Training starten.

Wir trainieren unser Netz mit den Trainingsdaten (imageDStrain) und übergeben die vorher festgelegten Layer und Optionen (Zeile 59).

Anschließend geben wir die Genauigkeit aus. Dies bereiten wir in Zeile 60 vor und lassen anschließend in Zeile 61 die Genauigkeit unseres Netzes ausgeben.

Step1Test:

In diesem Skript Testen wir unser zuvor trainiertes Netz. Dazu laden wir zuerst unsere Testdaten, welche in dem Ordner PlatesCuttedFromPicAndLabels liegen, in einen Imagedatastore (Zeile 26).

Anschließend lesen wir unseren Datastore aus (Zeile 30), lassen uns die richtige Kategorie der Bilder ausgeben (Zeile 30) und zeigen diese an (Zeile 31). Nun lassen wir uns die Antwort des Netzes ausgeben (Zeile 32).

Zuletzt testen wir, ob die Antwort des Netztes mit der richtigen Kategorie übereinstimmt und lassen uns die Genauigkeit ausgeben (Zeile 35 – 36).

Step1Main:

Wir haben unser Training und den Test in eine Funktion geschrieben, um diese gebündelt in einem Skript aufrufen zu können. Dies passiert in diesem Skript. In Zeile 7 wird zuerst das Training durchgeführt und in Zeile 8 anschließen der Test des trainierten Netztes.

Schritt 2

Für Schritt 2 wurden die geforderten Skripte(Funktionen) Step2Train und Step2Test angelegt. Zusätzlich existiert ein Step2Main, zum Ausführen der beiden Funktionen nacheinander.

Schritt2Train

Das Programm wurde aus Step1Train übernommen und eine Augumentierung eingefügt. Hierbei wurden die geforderten Werte einbezogen:

- Shear
- Scale
- Translation
- Rotation

Die Festlegung der Werte findet über die imageAugmenter Variable statt. In dieser ist der „imageDataArgumenter“ mit den jeweiligen Werten festgelegt. Die Werte für Shear und Translation sind doppelt vorhanden, da diese Werte für die X und Y-Achse angegeben werden müssen.

Die Werte sind durch systematisches Testen entstanden. Beachtet haben wir, dass die Bilder in der Breite mehr Pixel haben als in der Höhe, und dadurch die Werte in der X Achse höher sein können als in Y Achse.

Hierdurch sind neue Image DataStores entstanden. Diese haben dann im weiteren Programmverlauf die DataStores aus Schritt 1 ersetzt.

Schritt2Test

Die Schritt2Test Datei wurde aus Schritt1Test übernommen. Es wurde lediglich der Dateikopf sowie der Funktionsname angepasst.

Schritt 3

Für Schritt 3 wurden die geforderten Skripte(Funktionen) Step3Train und Step3Test angelegt. Zusätzlich existiert ein Step3Main, zum Ausführen der beiden Funktionen nacheinander.

Schritt3Train

Hier haben wir erneut zuerst einen ImageDatastore generiert, welcher die Bilder aus dem Ordner „platesResize“ enthält. Anschließend haben wir wie bei den vorherigen Schritten die Bilder aus dem ImageDatastore in Trainingsdaten(70%) und Validierungsdaten(30%) aufgeteilt.

Danach führen wir erneut eine Augmentierung durch. Diese verläuft wie in den vorherigen Schritten, außer dass hier die outputSize auf [227 227 3] abgeändert wurde. Diese outputSize ist notwendig, da das Alexnet nur Bilder mit dieser Größe als Input verarbeiten kann. Danach werden die Trainingsdaten, wie die Validierungsdaten, beide augmentiet.

Wenn die Daten erfolgreich Augmentiert sind, widmen wir uns der Definierung des Netzes. Hierfür verwenden wir das vordefinierte Alexnet. Wir übernehmen von diesem Netz sämtliche Layer, bis auf die letzten drei. Anschließend definieren wir die Anzahl der Klassen und die Layers. Bei den Layers laden wir zuerst den übernommenen Layer (layerTransfer) und definieren anschließend die „fullyConnectedLayer“, „softmaxLayer“ und „classificationLayer“.

Nachdem die Layers definiert sind, widmen wir uns den Optionen für das Training. Diese sind gleich aufgebaut, wie bei den vorherigen Trainings und erhalten zudem drei zusätzliche Optionen. Die erste zusätzliche Option ist die „MiniBatchSize“, dies ist dafür verantwortlich die Verluste zu bewerten und die Gewichte zu aktualisieren. Die zweite ist die „ValidationPatience“, welche die Vorgabe für den Abbruch des Trainings enthält. Die letzte ist die „InitialLearnRate“, welche die anfängliche Lernrate bestimmt.

Anschließend trainieren wir das Netz wie in den vorherigen Schritten und geben die Genauigkeit aus.

Schritt3Test

Hier laden wir auch als allererstes die Testdaten in einen ImageDatastore. Anschließend müssen wir die Größe der Testbilder auf dieselbe Größe der Trainingsbilder bringen. Dafür verwenden wir eine Augmentierung, welche als Parameter nur die Outputsize und die Bilder der Testdaten erhält.

Sämtliche Testdaten haben nun die passende Größe, damit das Alexnet diese verarbeiten kann. Anschließend lassen wir den Datastore auslesen, lassen die richtigen Kategorien ausgeben und die Kategorien, welche das Alexnet als Antwort liefert.

Zuletzt testen wir das trainierte Netz und lassen uns die Genauigkeit ausgeben.

Schritt3Main

Hier führen wir erneut zuerst das Training durch und anschließend den Test.