Aufgabenblatt 6

Praktikum Computer Vision
WiSe 17/18

Christian Wilms

28. November 2017

Aufgabe 1 — Bilder mit Fully Connected Network klassifizieren

1. Ladet das Python-Skript `kerasMNIST.py` herunter und probiert unterschiedliche Anzahlen an Layern sowie unterschiedliche Anzahlen an Neuronen pro Layer für das Neuronale Netz aus. Was für Auswirkungen auf Ergebnis, Laufzeit und Anzahl an Gewichten (Parametern) hat dies?
2. Was passiert, wenn ihr die Aktivierungsfunktion (ReLU) entfernt?

Aufgabe 2 — Ein erstes Convolutional Neural Network

1. Ändert nun den Datensatz von MNIST auf CIFAR-10 (`from keras.datasets import cifar10`). Beachtet, dass die Bilder im CIFAR-10 Datensatz 32×32 RGB-Bilder sind. Was für Änderungen müsst ihr nun am Netz und der Vorverarbeitung der Daten vornehmen?
2. Anstatt nun noch mehr Dense-Layer in das Netz zu integrieren, verwendet 2D-Convolution Layers am Anfang des Netzes. Welche Kombination bringt das beste Validierungs-Ergebnis? Was bringt das Ergebnis auf den Testdaten?
   Tipp: Nutzt nicht mehr als 5 2D-Convolution Layers und nicht mehr als 256 Kanäle pro Layer.
3. Fügt an verschiedenen Stellen zwischen und nach den 2D-Convolution Layern einen oder mehrere Max-Pooling Layer ein. Was bewirken diese?
4. Was bewirkt eine veränderte Learning Rate? Probiert die Werte von $1, 0.1, 0.01, 0.001, 0.0001$ aus.
5. Trainiert das Netz noch weiter als nur bis zum besten Validierungs-Ergebnis. Warum verbessert sich das Trainingsergebnis weiter?
6. Speichert die Gewichte des Modells in einer `h5`-Datei.

Aufgabe 3 — Feature Extraktion mit vordefinierten CNNs

1. Ladet nun die aus Woche 4 bekannten Haribo-Daten. Die Validierungsdaten sollen diesmal als Testdaten dienen. Achtet darauf, dass ihr die Bilder zu einem großen Array zusammenschließt und den Datentyp sowie den

Wertebereich anpasst (Datentyp `np.float`, Wertebereich $0, \ldots, 1$). Skaliert die Bilder außerdem auf 32×32. Ein Ausschneiden der Objekte ist weiterhin sinnvoll, die Extraktion der Labels zwingend notwendig.

2. Erstellt nun das gleiche Modell, das sich in Aufgabe 2 als bestes herausgestellt hat und ladet die gespeicherten Gewichte. Entfernt allerdings alle Dense-Layers, da wir das Netz nun nur zur Extraktion der Merkmale nutzen wollen. Als Klassifikator soll weiter der nächste Nachbar in der Trainingsmenge genutzt werden. Wie verändert sich das Ergebnis auf den beiden Datensätzen (Haribo und Haribo2)?
3. Zusatzaufgabe: Ladet das VGG19-Netz (`keras.applications.vgg19.VGG19`) mit den Imagenet-Gewichten. Nutzt dieses nun zur Extraktion der Features auf den beiden Haribo-Datensätzen. Entfernt auch das Zuschneiden der Bilder. Was beobachtet ihr?
   Tipp: Ein Beispiel zur Benutzung des VGG19 und zur Extraktion von Features mit einem Standardnetz findet ihr hier.
   Tipp2: Führt unbedingt die dort beschriebene Vorverarbeitung (`preprocess_input`) durch, da das Netz mit Bildern trainiert wurde, von denen der mittlere Farbwert des Datensatzes je Farbkanal abgezogen wurde.

Aufgabe 4 — Zusatzaufgabe: Fine-tuning mit VGG19

1. Ladet erneut die CIFAR-10 Daten sowie das VGG19-Netz mit Imagenet-Gewichten (vergesst die Vorverarbeitung nicht, siehe Aufgabe 3.3), diesmal jedoch zum fine-tuning. D.h. der Klassifikator ist Teil des Netzes. Allerdings müsst ihr dazu neue Dense-Layer auf das Netz aufsetzen. Ein Beispiel wie dies bei einem Standard-Netz funktioniert, ist hier dargestellt. Nutzt zwei Dense-Layer mit 4096 Neuronen und anschließend einen Dense-Layer mit 10 Neuronen und der `'softmax'`-Aktivierungsfunktion als Output. Trainiert das gesamte Netz. Werden die Ergebnisse besser als in Aufgabe 3?
2. Verändert das Trainieren nun so, dass zunächst nur die neuen Dense-Layer trainiert werden (bei allen anderen Layern `trainable=false`) und anschließend das gesamte Netz noch einmal trainiert wird. Für den ersten Schritt solltet ihr eine Learning Rate von 0.01 und für den zweiten Schritt eine Learning Rate von 0.001 nutzen.