Machine Learning 3
Refugeeks / intoCODE
2022

Torben Laue, M.Sc.
Florian Kluger, M.Sc.

Convolutional Neural Network (CNN)

Für die folgenden Aufgaben verwenden Sie den CIFAR10-Datensatz. Dieser besteht aus 60000 Bildern mit 32x32 Pixeln in Farbe. Jedes Bild gehört zu einer von 10 verschiedenen Klassen. Ein paar Beispiele aus dem Datensatz sind in Abbildung 1 dargestellt.

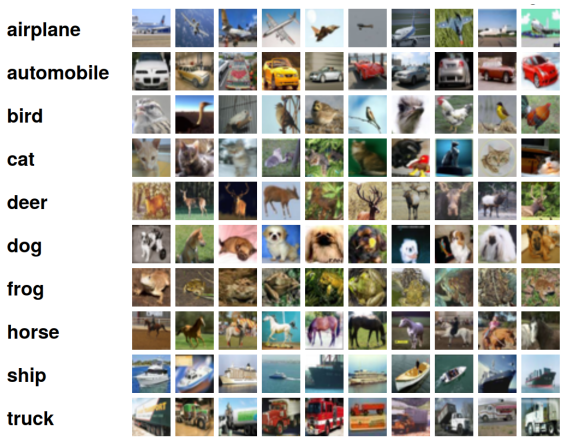

Abbildung 1: Beispiele aus dem CIFAR10-Datensatz.

Der Datensatz ist aufgeteilt in ein **Trainings-Set** mit 50000 Bildern, und ein **Test-Set** mit 10000 Bildern. Sie werden das Test-Set als **Validation-Set** verwenden.

1 Neuronales Netz Definieren

Verwenden Sie für diese Aufgabe die Datei `network.py`.

Vervollständigen Sie die Klasse `SimpleCNN` und implementieren Sie ein neuronales Netz bestehend aus:

1. Vier Convolutional-Layer mit je 32 Kerneln der Größe (3,3), Padding=1 und Stride=1
2. Nach jedem Conv-Layer folgt ein Max-Pooling mit Fenster-Größe (2,2)
3. Nach jedem Max-Pooling folgt eine ReLU-Aktivierungsfunktion

Fügen Sie am Ende noch ein **Average-Pooling** hinzu, welches die Auflösung der Features auf (1,1) reduziert, sowie einen **Fully-Connected-Layer** für die Klassifikation.

2 Augmentierung

Verwenden Sie für diese Aufgabe die Datei `train.py`.

Sie sollen mithilfe von `torchvision.transforms.Compose` zwei Transformationen definieren:

1. Training: mit Augmentierung und Normalisierung
2. Validation: nur Normalisierung

Für die Trainingsdaten soll die Transformation folgende Operationen **in dieser Reiehenfolge** enthalten:

1. Random Crop der Größe (32, 32) mit Padding=4 und Padding Mode 'reflect' (torchvision.transforms.RandomCrop)
2. Random Horizontal Flip (torchvision.transforms.RandomHorizontalFlip)
3. Konvertierung von Bild zu Tensor (torchvision.transforms.ToTensor)
4. Normalisierung (torchvision.transforms.Normalize)

Für die Validationdaten soll die Transformation nur folgende Operationen enthalten:

1. Konvertierung von Bild zu Tensor (torchvision.transforms.ToTensor)
2. Normalisierung (torchvision.transforms.Normalize)

Hinweise:

- Für die Normalisierung haben Sie Mittelwert und Standardabweichung der Daten in der Variable `data_mean_and_std` gegeben.
- Die beiden Transformationen werden vom Dataset-Objekt angewendet. Sie müssen sie *nur* definieren.

3 Training

Führen Sie nun die `train.py` aus, um ihr Netz zu trainieren.

Welche Genauigkeit auf dem Validation-Set erhalten Sie?

4 Batch-Normalisation

Fügen Sie Ihrem CNN nun noch Batch-Normalisation-Layer hinzu. Diese Layer sollten Sie zwischen Max-Pooling und Aktivierungsfunktion anwenden. Trainieren Sie ihr Netz anschließend nochmal. Welche Genauigkeit erreichen Sie nun?

5 ResNet

Verwenden Sie nun anstelle Ihres CNNs das `ResNet6` CNN, welches in der `network.py` definiert ist. Welche Genauigkeit erreicht dieses Netzwerk?

Schauen Sie sich den Code von `ResNet6` an, während das Netz trainiert. Welche Unterschiede fallen Ihnen zu dem von Ihnen erstellten CNN auf?