

Aufgabe 29 (Modulare Implementierung künstlicher neuronaler Netze, Teil 2)

In `modular_net2.zip` findet sich eine leicht verbesserte modulare Implementierung neuronaler Netze¹, die im Vergleich zu Aufgabe 28 eine effizientere Umsetzung der Convolutional und Max-Pool Layers enthält.²

- (a) Trainieren Sie das `BasicConvNet` mittels `run_training_mnist.py`. Das Training sollte im Vergleich zu A28 (c) nun viel weniger Zeit benötigen. Welche maximale Klassifizierungsgenauigkeit beim Testdatensatz können Sie erreichen, wenn Sie 100 (anstatt 20) Epochen lang trainieren? Wiederholen Sie nun das Training mit allen verfügbaren 50000 Ziffern (wobei eine Epoche genügt). Auf welche Klassifizierungsgenauigkeit kommen Sie jetzt? [2 Punkte]

Hinweis: Beachten Sie, dass der Trainingsdatensatz in dieser Teilaufgabe zur direkten Vergleichbarkeit mit A28 (c) zunächst auf insgesamt nur 40 Ziffern eingeschränkt ist!

Sie können das trainierte Netzwerk natürlich auch nochmals an Ihren eigenen handschriftlichen Ziffern ausprobieren.

- (b) Der CIFAR-10 Datensatz³ ist eine populäre Referenz für Machine Learning-Forschungsprojekte (konzeptionell ähnlich zu MNIST). Er besteht aus 60000 32×32 Farbbildern, eingeteilt in 10 Klassen. Laden Sie zunächst den Datensatz herunter (siehe Beschreibung in `get_cifar10.md`), und lassen Sie anschließend `run_training_cifar.py` laufen. Visualisieren Sie nun die Filter, d.h. die Gewichte $w_{f,c,k}$ der Convolutional Layer des trainierten Netzwerks, mit Hilfe der Funktion `visualize_conv_weights` in `visualization.py`. Worin spiegeln sich die Dimensionen von w (Anzahl Features, Anzahl Channels, Höhe und Breite) im Ausgabebild wider? [2 Punkte]

Hinweis: Die Klassifizierungsgenauigkeit beim Testdatensatz sollte knapp über 40% liegen.

- (c) Versuchen Sie als Herausforderung, einen möglichst hohen Klassifizierungserfolg beim CIFAR-10 Datensatz zu erreichen, indem Sie die Netzwerk-Architektur (Filterdimensionen, Anzahl Zellen in der versteckten Schicht, ...) bzw. "Hyperparameter" (Regularisierungsstärke, ...) optimieren. [2 Punkte]

Hinweis: Sie können den Trainingsdatensatz durch die Modifikation `get_CIFAR10_data(num_training=20000)` verkleinern, um die Trainingszeiten zu verkürzen. Für die Teilaufgabe genügt ein zusätzlicher Trainingslauf mit modifizierten Parametern.

Aufgabe 30 (Grundlagen von TensorFlow und Artistic Style Transfer)

- (a) Vervollständigen Sie folgendes Python-Programm, um die Ableitung der Funktion $x \mapsto 2x \sin(\pi x)$ an der Stelle $x = 3/2$ mittels TensorFlow auszurechnen, und vergleichen Sie das Ergebnis mit dem analytischen Referenzwert -2 . [2 Punkte]

```
import numpy as np
import tensorflow as tf

x = tf.placeholder(tf.float64)
y = ...
dydx = tf.gradients(y, x)
sess = tf.Session()
y_eval, dydx_eval = sess.run((y, dydx), feed_dict=...)
print('y_eval:', y_eval)
print('dydx_eval:', dydx_eval)
```

Hinweis: Das Jupyter-Notebook `tensorflow_intro.ipynb` auf der Vorlesungswebsite eignet sich als mögliche Orientierung.

¹basierend auf <http://cs231n.github.io/assignments2018/assignment2>

²Die `im2col`-Operation verwendet einen Trick basierend auf den "strides" eines NumPy-Arrays.

³<https://www.cs.toronto.edu/~kriz/cifar.html>

Der Artistic Style Transfer-Algorithmus⁴ verwendet sogenannte Gram-Matrizen zur Repräsentation des Zeichenstils. Die Gram-Matrix der ℓ -ten Schicht ist (basierend auf der Netzwerk-Aktivierung) definiert als

$$G_{ff'}^\ell = \sum_j a_{fj}^\ell a_{f'j}^\ell, \quad (1)$$

wobei f, f' Feature-Indizes bezeichnen und $j \in \mathbb{Z}^2$ den räumliche (Pixel-)Index. In Matrixschreibweise lässt sich G^ℓ also ausdrücken als $G^\ell = (a^\ell) \cdot (a^\ell)^T$.

Die "Kostenfunktion" $\mathcal{L}_{\text{style}}^\ell$ quantifiziert die Abweichung vom vorgegebenen Zeichenstil auf der ℓ -ten Schicht durch

$$\mathcal{L}_{\text{style}}^\ell = \frac{1}{(2N^\ell)^2} \|G^\ell - G_{\text{style}}^\ell\|_F^2,$$

wobei die symmetrische Matrix G_{style}^ℓ fest vorgegeben ist, G^ℓ in Gl. (1) definiert ist, N^ℓ die Anzahl der Einträge von a^ℓ bezeichnet und $\|X\|_F = \sqrt{\text{Tr}[XX^T]}$ die Frobenius-Norm einer Matrix X ist.

(b) Verifizieren Sie, dass

$$\frac{\partial \mathcal{L}_{\text{style}}^\ell}{\partial a_{fj}^\ell} = \frac{1}{(N^\ell)^2} ((G^\ell - G_{\text{style}}^\ell) \cdot a^\ell)_{fj}.$$

[2 Punkte]

(c) In `style_transfer.zip` findet sich eine Python/TensorFlow Implementierung des Artistic Style Transfer-Algorithmus.⁵ Vervollständigen Sie die fehlende Zeile innerhalb der Python-Funktion `gram_matrix` in `style_transfer.py` zur Berechnung der Gram-Matrix mittels TensorFlow. Lassen Sie nun als Demonstration `run_style_transfer.py` laufen. [2 Punkte]

Hinweis: In TensorFlow wird die Aktivierung a_{fj}^ℓ als Tensor der Dimension $1 \times H \times W \times F$ gespeichert, wobei die Höhe H und Breite W die räumlichen Dimensionen der Ausgabe der aktuellen Schicht angeben und F die Anzahl der Features, d.h. die Feature-Dimension kommt zum Schluss. Sie können Gl. (1) mittels `tf.tensordot` und geeignete Wahl des `axes`-Parameters auswerten; insbesondere sind keine `for`-Schleifen nötig.

Bemerkung: Oft findet sich in der Literatur auch die Bezeichnung "channels" anstatt "features", da ja die Ausgabe-Features der ℓ -ten Schicht als Eingabe-Channels der $(\ell + 1)$ -ten Schicht aufgefasst werden.

Tutoraufgabe (Visualisierung von Klassifikations-Netzwerken)

Wir wollen ein bereits trainiertes Netzwerk zur Bildklassifizierung untersuchen; das Netzwerk ordnet ein Eingabebild einer Klasse (z.B. Katze, Haus, Auto, ...) zu, wobei die möglichen Klassen fest vorgegeben sind. Beim "ImageNet Large Scale Visual Recognition Challenge" gibt es beispielsweise 1000 Klassen. Die Idee besteht darin, zu einer vorgegebenen Klasse ein synthetisches Eingabebild zu finden, das das Netzwerk maximal "anspricht".⁶ Konkret besitze das Netzwerk für jede Klasse y eine separate Ausgabezelle, die jeweils einen "Score" S_y liefere. (Die "Scores" werden typischerweise noch mittels einer Softmax-Schicht in eine Wahrscheinlichkeitsverteilung umgewandelt, was hier allerdings nicht benötigt wird.) Es sei nun eine Klasse y fest ausgewählt; generieren Sie ein synthetisches Bild $x \in \mathbb{R}^{H \times W \times C}$, das die Netzwerk-Ausgabe für diese Klasse maximiert:

$$x = \arg \max_{x'} S_y(x') - \lambda \|x'\|^2.$$

Der zweite Term dient zur Regularisierung. Implementieren Sie diese Idee für das "SqueezeNet"⁷.

⁴siehe L. A. Gatys, A. S. Ecker, M. Bethge. Image style transfer using convolutional neural networks. CVPR (2016)

⁵basierend auf <http://cs231n.github.io/assignments2018/assignment3>

⁶siehe K. Simonyan, A. Vedaldi, A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. ICLR (2014)

⁷siehe F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. arXiv:1602.07360