BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Prof. Dr.-Ing. Kristian Hildebrand

Visual and Scientific Computing
Neuronales Netz zur Bildklassifikation

WiSe 2019/20

Das Lernziel dieser Übungsaufgabe ist, ein Neuronales Netz - wie in Abbildung 2 dargestellt - größtenteils selbstständig zu implementieren und dabei die wichtigsten Schritte zu verinnerlichen. Sie werden dabei lernen, wie man ein Neuronales Netzwerk trainiert, die Ergebnisse auswertet und darstellt. Ziel ist es, Bilder des CIFAR-10 Bilddatensatzes zu klassifizieren. Dieser besteht aus 60000 Bilder, die jeweils 32x32 Pixel groß sind und es gibt insgesamt 10 verschiedene Klassen: airplane, automobile, bird, cat, deer, dog, frog, horse, ship und truck (wie in Abbildung 1 zu sehen). Laden Sie sich vorab bitte den Datensatz herunter unter: https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz.

Abbildung 1: Beispielbilder aus dem CIFAR-10 Datensatz und die dazugehörigen Klassen.

Aufgabe 1: Vorbereitung der Daten (1 Punkte)

Laden Sie den CIFAR-10 Datensatz unter Zuhilfenahme der vorgegebenen Funktion *prepare_CIFAR10_images()*. Dabei werden die Bilder in einem Vorverarbeitungsschritt angepasst werden indem man den *np.mean()* der Trainingsbilder von allen Bildern (in Trainings- und Testdatensatz) abzieht.

Aufgabe 2: Implementierung des Vorwärtsschrittes (2 Punkte)

Implementieren Sie die Berechnung des Forward-Passes in der Funktion *forward()*. Diese gibt den Loss und die Aktivierungen der Layer 1 (Wert nach der Aktivierungsfunktion ReLU) und 2 (Wert nach dem Softmax / Wahrscheinlichkeiten) zurück.

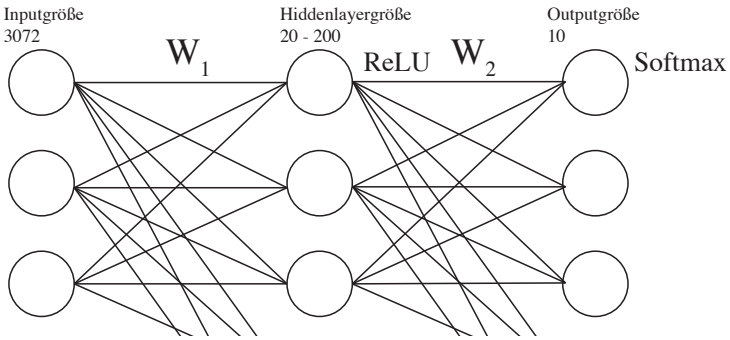


Abbildung 2: Darstellung des zu implementierenden 2-Layer Netzes. Die Eingabedimension ist ein $32x32x3 = 3072$ großes Eingabebild (oder mehrere abhängig von der sogenannten Batch-Size) als Zeilenvektor. Der erste Layer ist 'Fully-connected' - jeder Eingabewert ist mit allen Knoten im Hidden-Layer verknüpft. Der Hidden-Layer aktiviert das Neuron über die *ReLU* Funktion. Der zweite Layer ist wieder 'Fully-connected'. Das Netzwerk wird mit der *Softmax Fehlerfunktion* trainiert und führt eine L2-Regularisierung beim Anpassen der Gewichte W_1 und W_2 durch. Die Architektur sieht also folgendermaßen aus: **Eingabe - Fully-connected Layer - ReLU - Fully-connected Layer - Softmax (Ausgabe)**. Die Ausgabe des zweiten Layers sind die Wahrscheinlichkeiten für jeden der 10 Klassen.

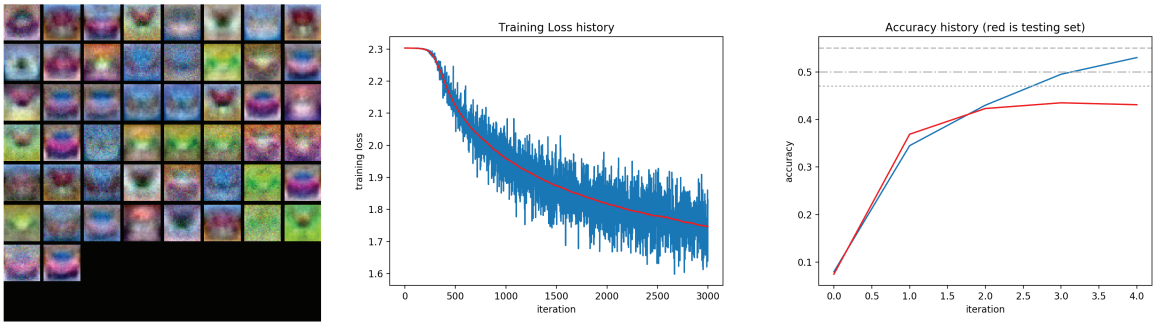
Aufgabe 3: Training des Netzes und die Vorhersage (5 Punkte)

Implementieren Sie den Trainingsprozess. Dazu gehört der Rückwärtsprozess, das Updaten der Parameter und der Optimierungsdurchlauf.

a) Implementieren Sie zunächst die Gradienten in der Funktion *backward*. (Siehe Kommentare im Sourcecode.)

b) Implementieren Sie den fehlenden Sourcecode in der Funktion *train*. Dabei wird die Anzahl der angegebenen Optimierungsschritte durchgeführt. In jeder Iteration werden dabei entsprechend des Gradienten und der Lernrate die Gewichte und Biases angepasst. Mit den angepassten Gewichten wird weitertrainiert. (Siehe Kommentare im Sourcecode.)

c) Implementieren Sie den fehlenden Sourcecode in der Funktion *predict*. Diese liefert ihnen eine Vorhersage der Klasse. (Siehe Kommentare im Sourcecode.)



(a) (b) (c)

Abbildung 3: (a) Beispielhafte Darstellung von 50 Gewichten W_1 nach der Trainingsphase. der Hidden-Layer hat hier 50 Neuronen. (b) Beispielhafte Darstellung des abnehmenden Trainingsfehlers während einer Trainingsphase über 3000 Iterationen. (c) Beispielhaft Darstellung der Klassifikationsgenauigkeit. Wenn die rote Linie über den grauen Linien ist, gibt es 1,2 bzw. 3 Zusatzpunkte.

Aufgabe 4: Finden der besten Hyperparameter (2 Punkte)

Eine Schwierigkeit bei Neuronalen Netzen, ist die Wahl der geeigneten Hyperparameter. Abbildung 3(b) und (c) zeichnen die Fehlerfunktion und die Genauigkeit. Die Fehlerfunktion wird in jeder Iteration

ausgewertet, die Genauigkeit nur in jeder Epoche. Eine Epoche ist die Anzahl der Trainingsdaten durch die Batchgröße (also: wenn einmal 49k Bilder durch das Netz gegangen sind). Die Hyperparameter können die Klassifikationsgenauigkeit enorm verbessern. Verbessern Sie durch eine geeignete Wahl z.B. der folgenden Hyperparameter das Klassifikationsergebnis des Validierungs-/Testsets (rote Linie im Graphen):

a) Anzahl der Epochen

b) Batchgröße (Anzahl der Bilder für die der Fehler berechnet wird und dann in der Summe (normalisiert) zurück durch das Netz gegeben wird.

c) Lernrate

d) Dämpfung der Lernrate

e) Regularisierungsfaktor

f) Anzahl der Hidden-Nodes

g) Anzahl der Hidden-Layer (3 Zusatzpunkte) - egal ob sich dadurch die Klassifikationsgenauigkeit erhöht. (Dafür müssen Sie eine ganze Menge Sourcecode anpassen. Forward und Backward-Pass)

Mit den im Sourcecode vorgegebenen Hyperparametern sollten Sie in etwa eine Genauigkeit von etwa 43% erreichen. Finden Sie bessere Parameter um eine Klassifikationsgenauigkeit von über 47% (1 Zusatzpunkt) oder über 51% (2 Zusatzpunkte) zu erzielen.

Hinweis 1: Die Gewichte werden zufällig initialisiert. D.h. die Ergebnisse können leicht schwanken. Mit dem random.seed sind die Ergebnisse für die gleichen Parameter aber immer gleich. Zum besseren Debuggen zu Beginn nutzen Sie eine *batch_size* und *num_iter* von 1. Ihr Startfehler sollte zu Beginn bei 2.3 liegen (10 Klassen $= \ln(10)$).

Hinweis 2: 1 Zusatzpunkt gibt es zusätzlich, wenn eine systematische Durchsuchung des Parameterraumes durchgeführt. D.h. nicht einfach durch probieren, sondern durch bspw. doppelte For-Loops, die über eine Reihe von Hyperparametern laufen und diese systematische untersuchen. Dabei müssen Sie sich dann in jedem Durchlauf die Ergebnisplots unter einem geeigneten Namen abspeichern.

Hinweis 3: Es gibt also maximal 3 ($+3$ wenn ein Layer mehr implementiert wird) Zusatzpunkte zu erreichen.

Abgabe Die Bearbeitungszeit der Teilaufgabe ist für ca. zwei Wochen ausgelegt, kann aber aufgrund der Anfangshürden etwas mehr Zeit benötigen. Die Abgabe soll via Moodle bis zu dem dort angegebenen Termin erfolgen. Verspätete Abgaben werden mit einem Abschlag von 3 Punkten je angefangener Woche Verspätung belegt. Geben Sie bitte jeweils nur eine einzige .zip-Datei mit den Quellen Ihrer Lösung ab.

Ich werde mich stichprobenartig Ihre Lösungen in der Übung demonstrieren und erläutern lassen. Die Qualität Ihrer Demonstration ist, neben dem abgegebenen Code, ausschlaggebend für die Bewertung! Ich behalte mir vor, einzelne hochgeladene Aufgaben zusätzlich anzuschauen.