Homework 3

Collaborators:
Name: Youchao Zhang
Student ID: 3170100125

Problem 3-1. Neural Networks

In this problem, we will implement the entire process of the neural networks training, such as feedforward, backpropagation and optimizer.

(a) Affine layer

Answer: forward: $y = wx + b$
difference: 9.769849468192957e-10
backward: The backpropagation is $\frac{dy}{dx} = \omega$, $\frac{dy}{dw} = x$,$\frac{dy}{db} = 1$
difference:
dx error: 2.0764490137839975e-08
dw error: 1.3748146602899365e-09
db error: 5.040934046417031e-12

(b) Relu layer

Answer: forward:
difference: 4.999999798022158e-08

backward: The backpropagation is
difference:

$$\frac{dy}{dx} = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Inline Question 1:
We've only asked you to implement ReLU, but there are a number of different activation functions that one could use in neural networks, each with its pros and cons. In particular, an issue commonly seen with activation functions is getting zero (or close to zero) gradient flow during backpropagation. Which of the following activation functions have this problem? If you consider these functions in the one dimensional case, what types of input would lead to this behaviour? 1. Sigmoid 2. ReLU 3. Leaky ReLU

Answer: :
ALL the three activation functions may get zero (or close to zero) gradient flow during backpropagation,but they are different from each other.
1.About the Sigmoid function,if the input is too large or to small then the gradient will be zero.
2.About the Relu ,if the input is smaller than zero ,the gradient is zero.
3.About LeakyRelu ,if the input is smaller than zero,the gradient is close to zero.

(c) Solver

Answer:

1. TwoLayerNet

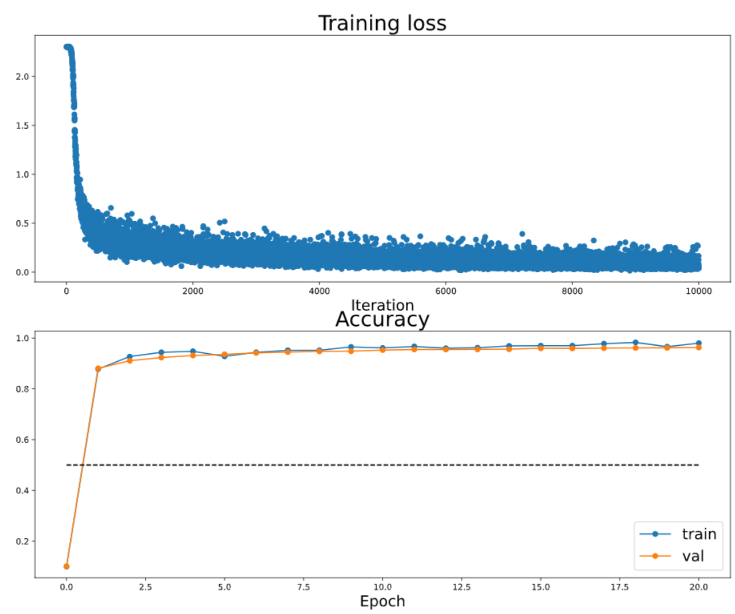

Figure 1: TwoLayerNet

2. Three-layer Net to overfit

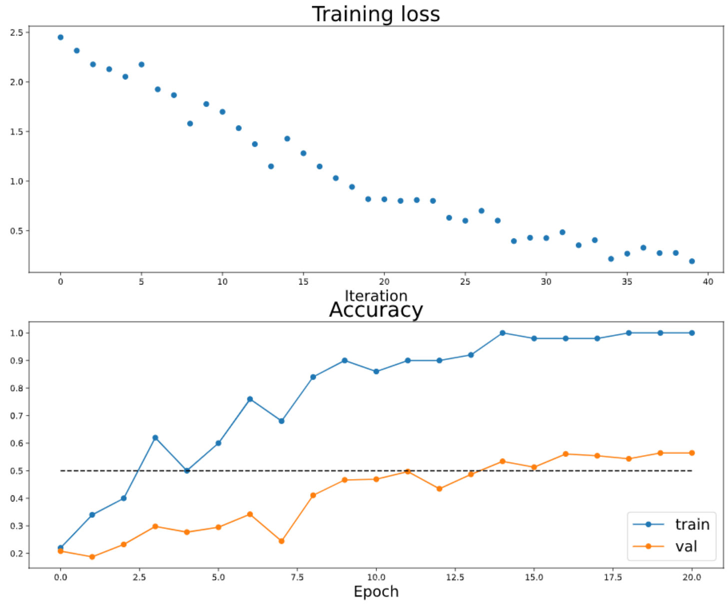

Figure 2: Three-layer Net to overfit

3. Five-layer Net to overfit

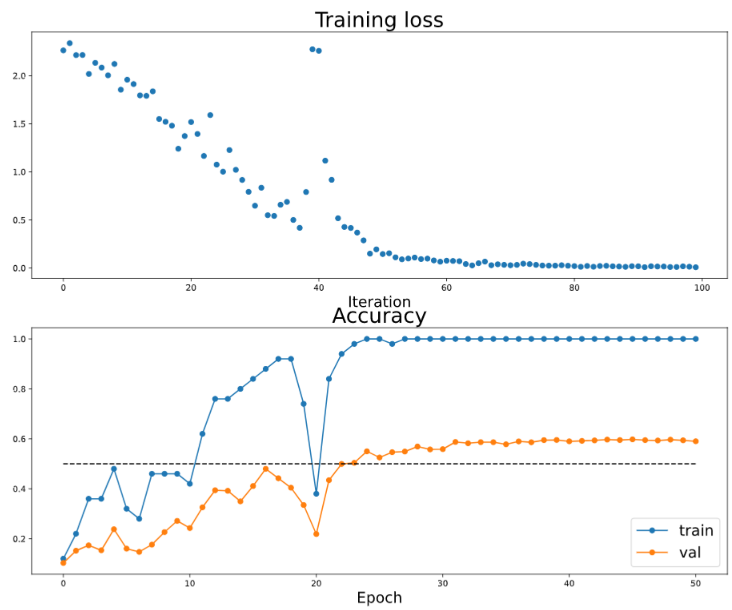

Figure 3: Five-layer Net to overfit

4. Inline Question 2
   Did you notice anything about the comparative difficulty of training the three-layer net vs training the five layer net?

Answer:
Because the five-layer network is more complex than the three-layer network, the parameters of the network increase, and the depth increases, so it is easier to overfit. In order to deal with over-fitting, the weight attenuation should be increased. In order to ensure rapid convergence of the model, the learning rate should also be increased. The three-layer network has a larger bias, and the five-layer network has a larger variance.

(d) Update relus

Answer:
next w error: 8.882347033505819e-09
velocity error: 4.269287743278663e-09

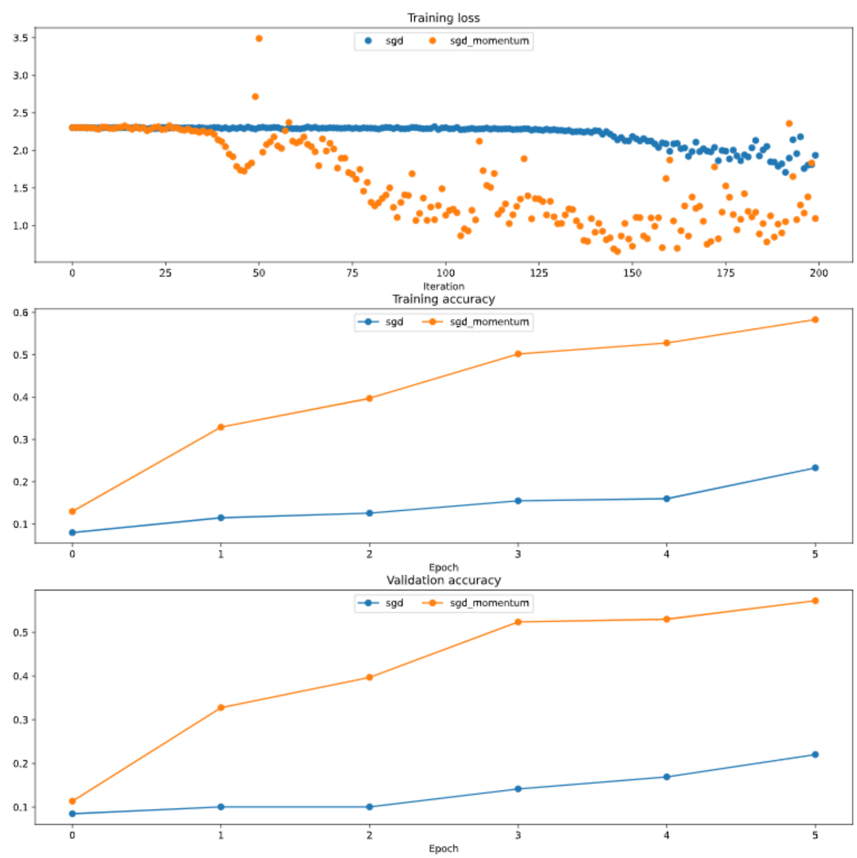

Figure 4: SGD+momentum update rule converge faster

(e) Conv layer

Answer:

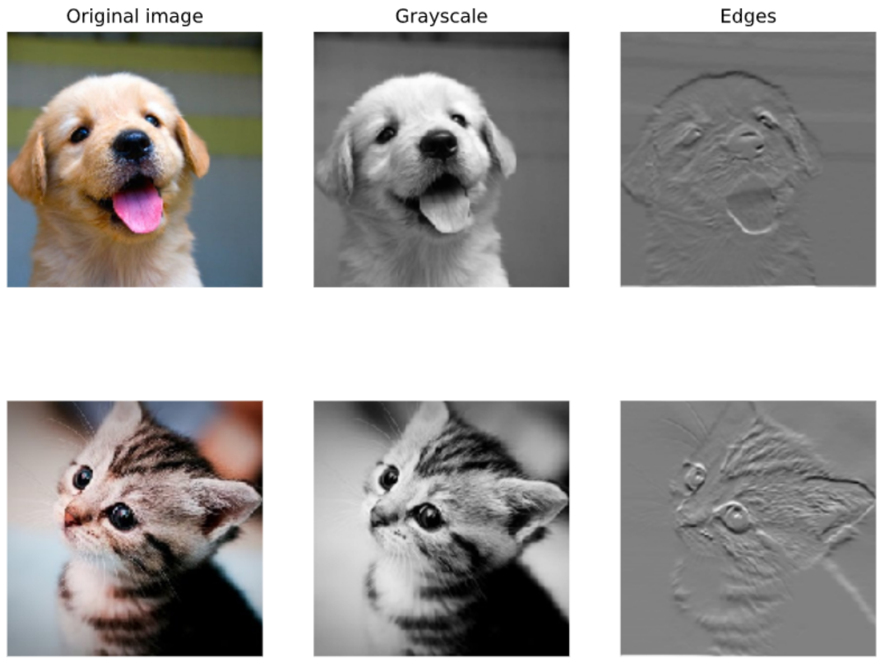

Figure 5: grayscale conversion and edge detection

Testing conv forward naive
difference: 2.2121476417505994e-08
Testing conv backward naive function
dx error: 1.159803161159293e-08
dw error: 2.2471264748452487e-10
db error: 3.37264006649648e-11

(f) Pooling layer

Answer:
Testing max pool forward naive function:
difference: 4.1666665157267834e-08
Testing max pool backward naive function:
dx error: 3.27562514223145e-12

(g) Experiment

Answer:
Filter size: Above we used 7x7; this makes pretty pictures but smaller filters may be more efficient
Number of filters: Above we used 32 filters. Do more or fewer do better?
Network architecture: The network above has two layers of trainable parameters. Can you do better with a deeper network?

You can implement alternative architectures in the file cnn.py. Some good architectures to try include:
conv-relu-poolxN - conv - relu - affinexM - softmax or SVM
conv-relu-poolXN - affineXM - softmax or SVM
conv-relu-conv-relu-poolxN - affinexM - softmax or SVM

Figure 6: My result

The training accuracy is 99.4%
The validation accuracy is 98.6%

Problem 3-2. Batch Normalization

The backpropagation of batch normalization.

(a) Answer:

$$\frac{\partial \ell}{\partial \widehat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$
$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$
$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m}$$
$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$
$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x}_i$$
$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$