HW2-批改标准

本次作业基础分15分，扣分制。有加分项，加分最多2分。

代码

- `run_cnn.py`（或自己编写的主程序）能正常运行，否则扣10分。
- 相关代码填补正确，一处错误扣1分。参考代码如下：

`functions.py`

```
import numpy as np
from scipy.signal import convolve

def conv2d_forward(input, W, b, kernel_size, pad):
    n, c_in, h_in, w_in = input.shape
    h_pad, w_pad = h_in + 2 * pad, w_in+2*pad
    padded_input = np.zeros((n, c_in, h_pad, w_pad))
    padded_input[:, :, pad:h_in + pad, pad:w_in + pad] = input
    h_out, w_out = h_pad - kernel_size + 1, w_pad - kernel_size + 1
    c_out = W.shape[0]
    output = np.zeros((n, c_out, h_out, w_out))
    for i in range(c_out):
        for j in range(c_in):
            ker = np.rot90(W[i, j], 2)[np.newaxis, :, :]
            output[:, i] += convolve(padded_input[:, j], ker, 'valid')

    output += b[np.newaxis, :, np.newaxis, np.newaxis]
    return output

def conv2d_backward(input,grad_output, W, b, kernel_size, pad):
    n, c_in, h_in, w_in = input.shape
    _, _, h_out, w_out = grad_output.shape
    assert h_out == h_in + 2 * pad - kernel_size + 1 and w_out == w_in + 2
* pad - kernel_size + 1, \
    "grad_output shape not consistent with output"

    h_pad, w_pad = h_in + 2 * pad, w_in + 2 * pad
    c_out = W.shape[0]
    # grad_input
    padded_grad_input = np.zeros((n, c_in, h_pad, w_pad))
    for i in range(c_in):
        for j in range(c_out):
            padded_grad_input[:, i] += convolve(grad_output[:, j], W[j, i]
[np.newaxis, :, :], 'full')
    grad_input = padded_grad_input[:, :, pad:h_in + pad, pad:w_in + pad]
    # grad_W
```

```
    padded_input = np.zeros((n, c_in, h_pad, w_pad))
    padded_input[:, :, pad:h_in + pad, pad:w_in + pad] = input
    grad_W = np.zeros((c_out, c_in, kernel_size, kernel_size))
    for i in range(c_in):
        for j in range(c_out):
            delta = np.flip(np.rot90(grad_output[:, j], 2, (1, 2)), 0)
            grad_W[j, i] += convolve(padded_input[:, i], delta,
'valid').squeeze()
    # grad_b
    grad_b = np.sum(grad_output, (0, 2, 3))
    return grad_input, grad_W, grad_b

def avgpool2d_forward(input, kernel_size, pad):
    n, c_in, h_in, w_in = input.shape
    h_pad, w_pad = h_in + 2 * pad, w_in + 2 * pad
    padded_input = np.zeros((n, c_in, h_pad, w_pad))
    padded_input[:, :, pad:h_in + pad, pad:w_in + pad] = input
    h_out, w_out = int(h_pad / kernel_size), int(w_pad / kernel_size)
    output_1 = np.zeros((n, c_in, h_out, w_pad))

    for i in range(kernel_size):
        output_1 += padded_input[:, :, i:h_pad:kernel_size, :]
    output_2 = np.zeros((n, c_in, h_out, w_out))
    for i in range(kernel_size):
        output_2 += output_1[:, :, :, i:w_pad:kernel_size]
    output = output_2 / (kernel_size * kernel_size)
    return output

def avgpool2d_backward(input, grad_output, kernel_size, pad):
    n, c_in, h_in, w_in = input.shape
    _, _, h_out, w_out = grad_output.shape
    assert h_out == (h_in + 2 * pad) / kernel_size and w_out == (w_in + 2 *
pad) / kernel_size, \
    "grad_output shape not consistent with output"
    h_pad, w_pad = h_in + 2 * pad, w_in + 2 * pad

    padded_grad_input_1 = np.zeros((n, c_in, h_pad, w_out))
    for i in range(kernel_size):
        padded_grad_input_1[:, :, i:h_pad:kernel_size, :] = grad_output
    padded_grad_input_2 = np.zeros((n, c_in, h_pad, w_pad))
    for i in range(kernel_size):
        padded_grad_input_2[:, :, :, i:w_pad:kernel_size] =
padded_grad_input_1
    grad_input = padded_grad_input_2[:, :, pad:h_in + pad, pad:w_in + pad]
/ (kernel_size * kernel_size)
    return grad_input
```

loss.py

```
class SoftmaxCrossEntropyLoss(object):
    def __init__(self, name):
        self.name = name

    def forward(self, input, target):
        input -= np.max(input)
        exp_input = np.exp(input)
        prob = exp_input / (np.sum(exp_input, axis=1, keepdims=True) + 1e-
20)  # for stablity
        return np.mean(np.sum(- target * np.log(prob + 1e-20), axis=1))  #
for stablity

    def backward(self, input, target):
        input -= np.max(input)
        exp_input = np.exp(input)
        prob = exp_input / (np.sum(exp_input, axis=1, keepdims=True) + 1e-
20)  # for stablity
        return (prob - target) / len(input)
```

注意：这一次实现并不唯一，有可能有的同学利用im2col方法。附件里有`test_functions.py`和`test_functions_nopad.py`文件。首先用`test_functions.py`进行测试，若通过则没有问题。如果没有通过，请再使用`test_functions_nopad.py`进行测试，因为有同学可能没有考虑pad，但运算是正确的，扣1分。使用方法是直接将`test_functions.py`放到和`functions.py`同一个文件夹下面，运行python test_functions.py即可。

易错点：

- 没有考虑pad。如果输出shape不对或者计算值不对，扣1分;
- `conv2d_backward`和`avgpool2d_backward`计算gradient除以了batch_size；
- `SoftmaxCrossEntropyLoss.forward`没有除以batch_size；
- `SoftmaxCrossEntropyLoss.backward`没有除以batch_size；
- `SoftmaxCrossEntropyLoss.backward`写成 target - prob；
- 如果程序有bug，无法运行通过，希望再细致检查一下，因为有的同学可能接口不同或者是 其他原因。
    - 如果属于非逻辑性错误，而是兼容性错误(比如一些新的语法写法)，看代码逻辑是否正确决定扣 分;
    - 如果属于未考虑的情况或其他原因，按错误处数量扣分，一处扣1分；
- 总扣分最多不超过10分。

  加分点：

- 如果按照`SoftmaxCrossEntropyLoss`给出的形式，考虑了input减去最大值，prob计算时分母 + eps保证数值稳定，加1分;
- 整体代码可以正常运行,无bug出现。

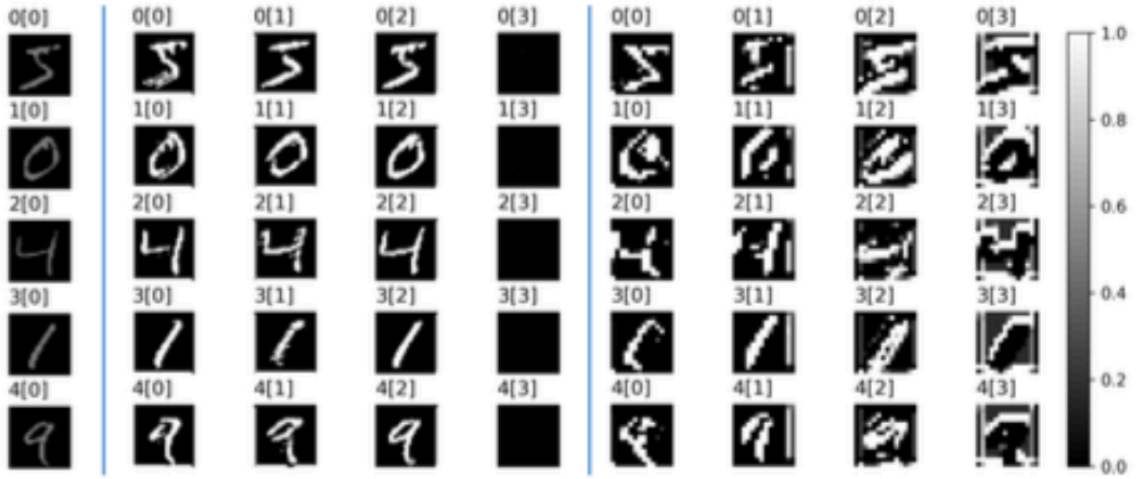
报告

- 一共三个内容：
  - 完整考察网络里有2个Conv+Relu+Pool模块 + 最后Linear + softmaxloss时的情况；
  - 对比2层MLP实验结果；
  - 可视化结果。

    报告结果包括CNN训练曲线，和MLP进行精度，参数量，训练时间对比，必须同时配备图或数值的说明。缺少一个实验扣1分，缺少图扣1分，缺少具体数值扣1分。
- CNN结构为：

```
model = Network()
model.add(Conv2D('conv1', 1, 4, 3, 1, 0.01))
model.add(Relu('relu1'))
model.add(AvgPool2D('pool1', 2, 0))
model.add(Conv2D('conv2', 4, 4, 3, 1, 0.01))
model.add(Relu('relu2'))
model.add(AvgPool2D('pool2', 2, 0))
model.add(Reshape('flatten', (-1, 196)))
model.add(Linear('fc3', 196, 10, 0.1))
```

- 结论：CNN网络优于MLP，accuracy更高，收敛更快，但是运行时间更长。
- 可视化结果大致如下图：



  注意: 是要可视化第一次ReLU输出的结果,不是weight可视化结果。至少显示4个数字的可视化结果。
- 扣分项
  - 展示结果时直接将程序log粘贴过来，没有画图，列表说明；
- 加分项
  - 对于超参数进行探索，并研究对于最终结果影响。比如对于learning rate，weight decay，momentum，hidden size的调整；
  - 总体上加分不超过2分。如探索非常详实、报告完整可以加2分。其他情况加1分。
- 抄袭： 由于此次代码实现比较困难，会可能存在普遍抄袭现象，需要关注：
  - 有一些同学选择了使用im2col方法，采用了网上的一个公开源码。如果注明出处, 不认为抄

袭。
    - 但如果在functions.py内部实现逻辑雷同，认为抄袭。

补交

每迟一天扣一分，对于补充说明原来作业的以及有特殊情况的，可视情况酌情扣分。