Homework 8: Convolutional Neural Networks

Student id:	B06502028	Name:	莊立楷

1. **Create a model only with fully connected layer, then train and test on fashion mnist dataset.**

```
net.add(nn.Dense(120, activation = 'relu'),nn.BatchNorm(),
        nn.Dense(84, activation = 'relu'),nn.BatchNorm(),
        nn.Dense(10,activation = 'relu'),nn.BatchNorm(),
        )
```

```
training on gpu(0)
epoch 1, loss 0.5180, train acc 0.823, test acc 0.858, time 9.1 sec
epoch 2, loss 0.3936, train acc 0.861, test acc 0.868, time 9.2 sec
epoch 3, loss 0.3531, train acc 0.873, test acc 0.872, time 12.0 sec
epoch 4, loss 0.3308, train acc 0.881, test acc 0.881, time 16.6 sec
epoch 5, loss 0.3125, train acc 0.887, test acc 0.881, time 15.2 sec
epoch 6, loss 0.2980, train acc 0.893, test acc 0.877, time 15.2 sec
epoch 7, loss 0.2843, train acc 0.898, test acc 0.886, time 16.0 sec
epoch 8, loss 0.2739, train acc 0.900, test acc 0.883, time 14.7 sec
epoch 9, loss 0.2618, train acc 0.904, test acc 0.888, time 14.4 sec
epoch 10, loss 0.2534, train acc 0.907, test acc 0.882, time 14.6 sec
epoch 11, loss 0.2433, train acc 0.911, test acc 0.889, time 14.6 sec
epoch 12, loss 0.2422, train acc 0.912, test acc 0.888, time 14.6 sec
epoch 13, loss 0.2314, train acc 0.916, test acc 0.893, time 14.2 sec
epoch 14, loss 0.2231, train acc 0.918, test acc 0.893, time 14.2 sec
epoch 15, loss 0.2170, train acc 0.920, test acc 0.895, time 14.0 sec
epoch 16, loss 0.2126, train acc 0.922, test acc 0.898, time 14.0 sec
epoch 17, loss 0.2096, train acc 0.923, test acc 0.889, time 14.0 sec
epoch 18, loss 0.2027, train acc 0.925, test acc 0.892, time 14.4 sec
epoch 19, loss 0.2000, train acc 0.927, test acc 0.895, time 14.0 sec
epoch 20, loss 0.1955, train acc 0.928, test acc 0.893, time 14.0 sec
epoch 21, loss 0.1904, train acc 0.930, test acc 0.894, time 14.5 sec
epoch 22, loss 0.1874, train acc 0.932, test acc 0.896, time 14.4 sec
epoch 23, loss 0.1861, train acc 0.932, test acc 0.893, time 14.0 sec
epoch 24, loss 0.1789, train acc 0.935, test acc 0.897, time 14.3 sec
epoch 25, loss 0.1769, train acc 0.936, test acc 0.890, time 14.2 sec
epoch 26, loss 0.1717, train acc 0.937, test acc 0.894, time 14.5 sec
epoch 27, loss 0.1693, train acc 0.939, test acc 0.899, time 14.1 sec
epoch 28, loss 0.1658, train acc 0.939, test acc 0.897, time 14.2 sec
epoch 29, loss 0.1629, train acc 0.941, test acc 0.897, time 14.1 sec
epoch 30, loss 0.1607, train acc 0.941, test acc 0.899, time 14.1 sec
```

Train accuracy:0.941
Test accuracy:0.899

2. **Create a model only with convolutional layer without pooling, then train and test on fashion mnist dataset.(the parameter number need to same as first question)**

```
net.add(nn.Conv2D(channels=6, kernel_size=5, activation='relu'),
        nn.Conv2D(channels=16, kernel_size=3, activation='relu'),
        nn.Flatten(),
        nn.Dense(120, activation = 'relu'),nn.BatchNorm(),
        nn.Dense(84, activation = 'relu'),nn.BatchNorm(),
        nn.Dense(10,activation = 'relu'),nn.BatchNorm(),
```

```
training on gpu(0)
[11:44:04] c:\jenkins\workspace\mxnet-tag\mxnet\src\operator\nn\cudnn\.
et the environment variable MXNET_CUDNN_AUTOTUNE_DEFAULT to 0 to disabl
epoch 1, loss 0.4769, train acc 0.838, test acc 0.871, time 14.3 sec
epoch 2, loss 0.3161, train acc 0.887, test acc 0.896, time 16.2 sec
epoch 3, loss 0.2626, train acc 0.907, test acc 0.898, time 18.4 sec
epoch 4, loss 0.2268, train acc 0.918, test acc 0.905, time 18.3 sec
epoch 5, loss 0.1968, train acc 0.930, test acc 0.904, time 18.7 sec
epoch 6, loss 0.1699, train acc 0.939, test acc 0.915, time 18.7 sec
epoch 7, loss 0.1479, train acc 0.948, test acc 0.906, time 18.5 sec
epoch 8, loss 0.1279, train acc 0.955, test acc 0.913, time 18.6 sec
epoch 9, loss 0.1130, train acc 0.960, test acc 0.914, time 18.9 sec
epoch 10, loss 0.1017, train acc 0.965, test acc 0.909, time 19.1 sec
epoch 11, loss 0.0874, train acc 0.971, test acc 0.914, time 18.9 sec
epoch 12, loss 0.0796, train acc 0.973, test acc 0.914, time 19.2 sec
epoch 13, loss 0.0711, train acc 0.976, test acc 0.910, time 19.1 sec
epoch 14, loss 0.0661, train acc 0.978, test acc 0.914, time 19.3 sec
epoch 15, loss 0.0578, train acc 0.980, test acc 0.915, time 19.1 sec
epoch 16, loss 0.0561, train acc 0.981, test acc 0.905, time 18.9 sec
epoch 17, loss 0.0504, train acc 0.983, test acc 0.911, time 19.0 sec
epoch 18, loss 0.0468, train acc 0.985, test acc 0.912, time 19.1 sec
epoch 19, loss 0.0415, train acc 0.986, test acc 0.915, time 19.1 sec
epoch 20, loss 0.0419, train acc 0.987, test acc 0.914, time 19.2 sec
epoch 21, loss 0.0373, train acc 0.988, test acc 0.907, time 18.7 sec
epoch 22, loss 0.0367, train acc 0.988, test acc 0.912, time 18.9 sec
epoch 23, loss 0.0319, train acc 0.989, test acc 0.913, time 18.9 sec
epoch 24, loss 0.0322, train acc 0.989, test acc 0.912, time 19.1 sec
epoch 25, loss 0.0334, train acc 0.990, test acc 0.914, time 18.9 sec
epoch 26, loss 0.0288, train acc 0.991, test acc 0.909, time 19.0 sec
epoch 27, loss 0.0277, train acc 0.992, test acc 0.913, time 19.2 sec
epoch 28, loss 0.0273, train acc 0.991, test acc 0.911, time 18.9 sec
epoch 29, loss 0.0275, train acc 0.991, test acc 0.912, time 18.8 sec
epoch 30, loss 0.0263, train acc 0.992, test acc 0.908, time 18.7 sec
```

Train accuracy:0.992
Test accuracy:0.908

3. **Create a model only with convolutional layer, then train and test on fashion mnist dataset.(the parameter number need to same as first question)**

```
net.add(nn.Conv2D(channels=6, kernel_size=5, activation='relu'),
        nn.MaxPool2D(pool_size=2, strides=2),
        nn.Conv2D(channels=16, kernel_size=3, activation='relu'),
        nn.MaxPool2D(pool_size=2, strides=2),
        nn.Flatten(),
        nn.Dense(120, activation = 'relu'),nn.BatchNorm(),
        nn.Dense(84, activation = 'relu'),nn.BatchNorm(),
        nn.Dense(10,activation = 'relu'),nn.BatchNorm()
```

```
training on gpu(0)
[11:04:40] c:\jenkins\workspace\mxnet-tag\mxnet\src\operator\nn\cudnn\
et the environment variable MXNET_CUDNN_AUTOTUNE_DEFAULT to 0 to disab
epoch 1, loss 0.4855, train acc 0.835, test acc 0.864, time 12.0 sec
epoch 2, loss 0.3520, train acc 0.875, test acc 0.868, time 13.9 sec
epoch 3, loss 0.3150, train acc 0.888, test acc 0.896, time 20.1 sec
epoch 4, loss 0.2916, train acc 0.894, test acc 0.898, time 17.2 sec
epoch 5, loss 0.2766, train acc 0.899, test acc 0.897, time 17.6 sec
epoch 6, loss 0.2619, train acc 0.905, test acc 0.898, time 17.1 sec
epoch 7, loss 0.2497, train acc 0.909, test acc 0.902, time 17.0 sec
epoch 8, loss 0.2406, train acc 0.912, test acc 0.900, time 17.2 sec
epoch 9, loss 0.2316, train acc 0.915, test acc 0.904, time 17.0 sec
epoch 10, loss 0.2255, train acc 0.919, test acc 0.911, time 17.0 sec
epoch 11, loss 0.2163, train acc 0.920, test acc 0.904, time 17.1 sec
epoch 12, loss 0.2106, train acc 0.922, test acc 0.909, time 17.1 sec
epoch 13, loss 0.2056, train acc 0.925, test acc 0.901, time 17.1 sec
epoch 14, loss 0.1990, train acc 0.926, test acc 0.910, time 17.3 sec
epoch 15, loss 0.1956, train acc 0.928, test acc 0.905, time 17.1 sec
epoch 16, loss 0.1890, train acc 0.930, test acc 0.913, time 17.2 sec
epoch 17, loss 0.1833, train acc 0.933, test acc 0.907, time 17.3 sec
epoch 18, loss 0.1813, train acc 0.934, test acc 0.911, time 17.1 sec
epoch 19, loss 0.1765, train acc 0.935, test acc 0.908, time 17.1 sec
epoch 20, loss 0.1729, train acc 0.936, test acc 0.909, time 17.3 sec
epoch 21, loss 0.1679, train acc 0.938, test acc 0.906, time 17.1 sec
epoch 22, loss 0.1639, train acc 0.938, test acc 0.908, time 17.0 sec
epoch 23, loss 0.1618, train acc 0.941, test acc 0.911, time 17.6 sec
epoch 24, loss 0.1562, train acc 0.942, test acc 0.906, time 17.8 sec
epoch 25, loss 0.1524, train acc 0.944, test acc 0.913, time 18.0 sec
epoch 26, loss 0.1503, train acc 0.944, test acc 0.908, time 17.3 sec
epoch 27, loss 0.1481, train acc 0.945, test acc 0.913, time 17.2 sec
epoch 28, loss 0.1459, train acc 0.946, test acc 0.910, time 17.5 sec
epoch 29, loss 0.1420, train acc 0.948, test acc 0.913, time 17.3 sec
epoch 30, loss 0.1404, train acc 0.948, test acc 0.917, time 17.1 sec
```

Train accuracy:0.948
Test accuracy:0.917

4. **What differences do you observe from the above three questions? like the accuracy or cost time.**

   A:

使用 CPU 訓練時，convolution layer 會大幅增加訓練時間，但若改用 GPU，則在 convolution 層數較少的情況下，訓練時間主要由 fully-connected 層的 neron 數量決定(越多越久)，故在上例子 1~3 中時間皆差不多在 14~18 秒左右。其中例子 3 較例子 2 多了 pooling 層，其運算量直接砍半，故例子 2 的運算時間平均

多了 1 秒左右，符合預期。另外，例子 3 與 2 較例子 1 多了兩層的 convolution layer，運算時間每個 epoch 增加 3 秒左右。

對於準確度而言，例子 2(無 pooling)可以在 training 做到 99%準確率，但在 testing 的結果卻稍微不足有加 pooling 的例子 3，可見 pooling 有些微 regularization 的效果。另外，例子 3 較例子 2 多了 convolution layer，在 training 的結果準確率差不多，但在 testing 的準確率平均提升了 3%左右，相當於 300 筆 data，可見增加了 convolution 有 regularization 的效果及幫助。