HW3-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

- `mlp/main.py`和`cnn/main.py`能正常运行，一个不能运行扣5分。
  - 注意train和valid\test acc都要能上涨，要是不能，仔细看看哪里有问题
- `mlp/model.py`填补
  - forward参数 （1分，其他合理也可以）

```
self.loss, self.pred, self.acc = self.forward(True)
self.loss_val, self.pred_val, self.acc_val = self.forward(False, reuse=True)
    #reuse可以都写tf.AUTO_REUSE
```

  - forward函数内容（1分。网络结构必须和要求一致。若不一致但合理可酌情扣0.5）

```
incoming = tf.layers.dense(self.x_, 256,
kernel_initializer=tf.truncated_normal_initializer(stddev=0.1))
incoming = batch_normalization_layer(incoming, is_train)
incoming = tf.nn.relu(incoming)
incoming = dropout_layer(incoming, FLAGS.drop_rate, is_train)
logits = tf.layers.dense(incoming, 10,
kernel_initializer=tf.truncated_normal_initializer(stddev=0.1))
```

  - batch_normalization_layer实现（1分。有测试函数，但测试函数**不能覆盖测试参数是否被训练，需要人工检查**。两者缺一，全扣。）

```
def batch_normalization_layer(incoming, is_train=True):
    return tf.layers.batch_normalization(incoming, 1, training=is_train)
```

    以下代码能够保证BN参数被训练

```
update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)
        with tf.control_dependencies(update_ops):
            self.train_op =
tf.train.AdamOptimizer(self.learning_rate).minimize(self.loss,
global_step=self.global_step,var_list=self.params)  # Use Adam Optimizer
```

  - dropout_layer实现（1分。有测试函数）

```
def dropout_layer(incoming, drop_rate, is_train=True):
    return tf.layers.dropout(incoming, drop_rate, training=is_train)
```

`cnn/model.py` 填补

- forward参数 （1分，其他合理也可以。**若mlp已错相同位置，不重复扣分**）

```
self.loss, self.pred, self.acc = self.forward(True)
self.loss_val, self.pred_val, self.acc_val = self.forward(False, reuse=True)
```

- forward函数内容（1分。网络结构必须和要求一致（但卷积层大小没有规定）。若不一致但合理可酌情扣0.5）**常见错误：增加了softmax层**

```
incoming = tf.layers.conv2d(self.x_, 5, 4,
kernel_initializer=tf.truncated_normal_initializer(stddev=0.1))
incoming = batch_normalization_layer(incoming, is_train)
incoming = tf.nn.relu(incoming)
incoming = dropout_layer(incoming, FLAGS.drop_rate, is_train)
incoming = tf.layers.max_pooling2d(incoming, 4, 4)
incoming = tf.layers.conv2d(self.x_, 32, 4,
kernel_initializer=tf.truncated_normal_initializer(stddev=0.1))
incoming = batch_normalization_layer(incoming, is_train)
incoming = tf.nn.relu(incoming)
incoming = dropout_layer(incoming, FLAGS.drop_rate, is_train)
incoming = tf.layers.max_pooling2d(incoming, 4, 4)
logits = tf.layers.dense(tf.reshape(incoming, [-1, 6*6*32]), 10,
kernel_initializer=tf.truncated_normal_initializer(stddev=0.1))
```

- batch_normalization_layer实现（1分。有测试函数，但测试函数**不能覆盖测试参数是否被训练，需要人工检查**。两者缺一，全扣。**若mlp已错相同位置，不重复扣分**）

```
def batch_normalization_layer(incoming, is_train=True):
    return tf.layers.batch_normalization(incoming, 3, training=is_train)
```

  以下代码能够保证BN参数被训练

```
update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)
        with tf.control_dependencies(update_ops):
            self.train_op =
tf.train.AdamOptimizer(self.learning_rate).minimize(self.loss,
global_step=self.global_step,var_list=self.params)  # Use Adam Optimizer
```

- dropout_layer实现（1分。有测试函数。**若mlp已错相同位置，不重复扣分**）

```
def dropout_layer(incoming, drop_rate, is_train=True):
    return tf.layers.dropout(incoming, drop_rate, training=is_train)
```

助教提示：

- 总扣分不超过10分
- 附件里有 `test_mlp.py` 和 `test_cnn.py`，但注意只能测试BN和dropout。因为涉及到初始化参数可能不同，所以测试错误不一定是BN实现错误。并且BN不能保证测试的参数被训练了，这一部分需要人工检查

- `test_cnnmodel.py`、`test_mlpmodel`用来测试其BN实现的正确性，如果val和test集降的厉害，说明BN实现存在问题。

报告

- 解释is_train和reuse的设置。需答出BN和dropout训练和测试不同，reuse可以共享参数。1分
- MLP和CNN with BN&dropout的实验。其中每个实验分别计分：
  - 需画图，包括train loss和validation loss。（可以不画acc曲线）若只画一个，全扣。MLP 1分 CNN 1分
  - 汇报最终acc数值（可以不汇报loss和时间）MLP 0.5分 CNN 0.5分
  - 性能太差，MLP扣0.5分，CNN扣0.5分
  - 分析MLP和CNN的不同 1分
- MLP和CNN without BN（至于dropout可以选择有或没有，但必须保持一致）。
  - 可以不画图
  - 汇报每个实验acc数值（MLP0.5分，CNN0.5分，若只画图了没数扣0.5分）
  - 分析BN的作用，加速收敛或者使网络更robust 答到一点即对。 1分
- Tune dropout （可以只选MLP和CNN中一种，至于BN可以选择有或没有，但必须保持一致）
  - 可以不画图
  - 汇报每个实验acc，需要至少2个实验的对比。 （所有实验一共1分，只画图了没数扣0.5分）
  - 分析dropout的作用 1分
- 解释train loss和val loss的区别，怎么调参。答出检查过拟合，调整dropout即可。1分
- 报告一共扣分不超过5分
- 这次点较多，以上全部答到，报告组织结构好，或者研究了其他内容报告可以加1~2分。