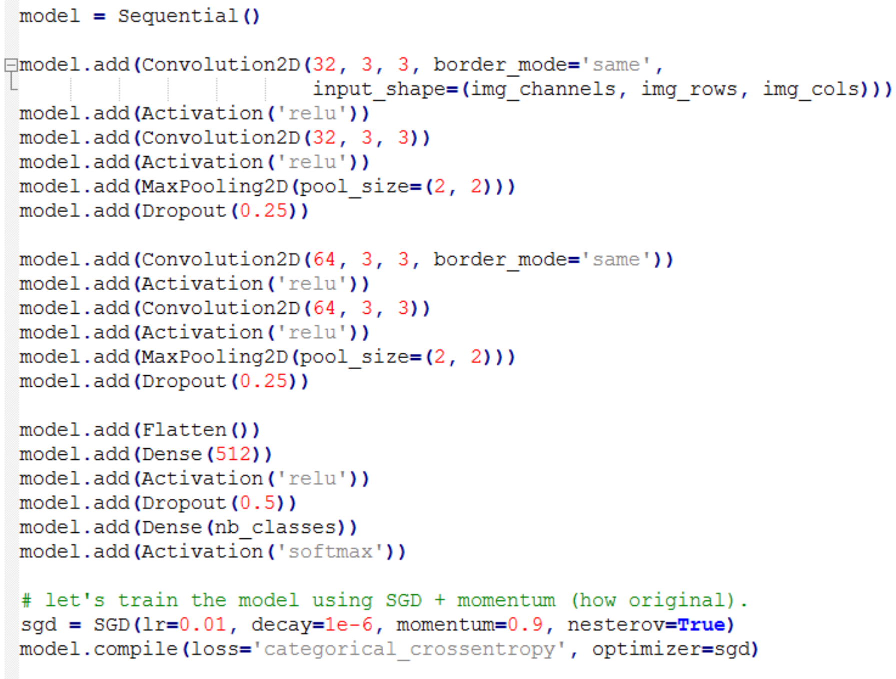
Cifar-100 实验总结

训练展示：

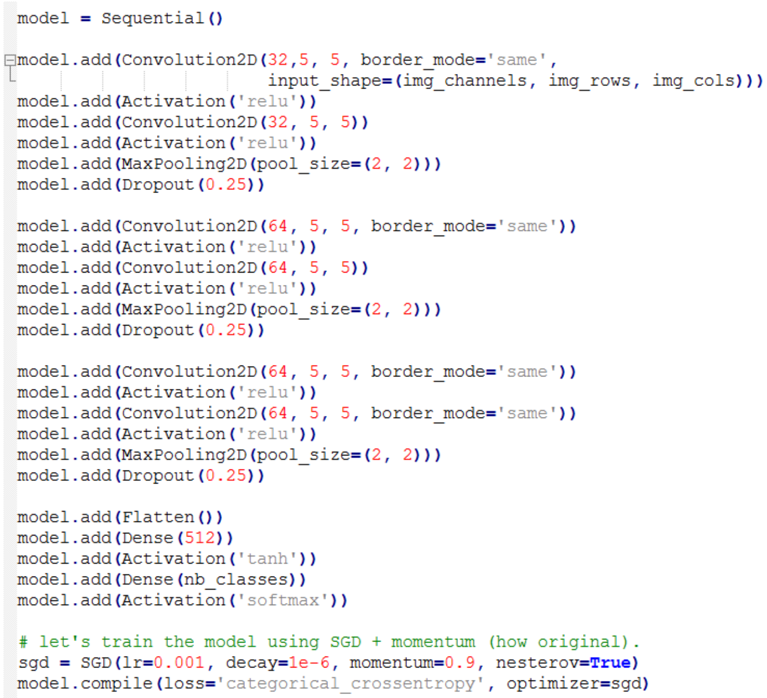
测试代码	网络结构	nb_epoch	loss	acc	val_loss	val_acc	训练时间/s
cifar100_cnn_dl	参考结构	50	2.4899	0.3609	2.3722	0.3856	1641
cifar100_cnn_dl1	卷积核：3*3->5*5 最后一个激活函数：’relu’->’ tanh’ SGD: lr = 0.01 -> lr = 0.001 增加卷积层：第二部分	50	1.8501	0.4938	1.7869	0.5177	2646

cifar100_cnn_dl 网络结构：



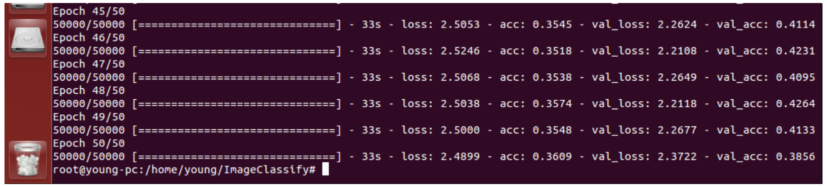
```
model = Sequential()

model.add(Convolution2D(32, 3, 3, border_mode='same',
                        input_shape=(img_channels, img_rows, img_cols)))
model.add(Activation('relu'))
model.add(Convolution2D(32, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(64, 3, 3, border_mode='same'))
model.add(Activation('relu'))
model.add(Convolution2D(64, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))

# let's train the model using SGD + momentum (how original).
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd)
```

cifar100_cnn_dl1 网络结构：



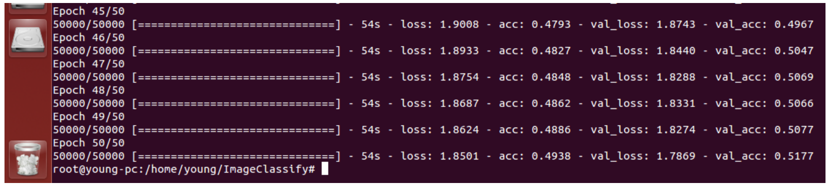
```
model = Sequential()

model.add(Convolution2D(32,5, 5, border_mode='same',
                        input_shape=(img_channels, img_rows, img_cols)))
model.add(Activation('relu'))
model.add(Convolution2D(32, 5, 5))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(64, 5, 5, border_mode='same'))
model.add(Activation('relu'))
model.add(Convolution2D(64, 5, 5))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(64, 5, 5, border_mode='same'))
model.add(Activation('relu'))
model.add(Convolution2D(64, 5, 5, border_mode='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('tanh'))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))

# let's train the model using SGD + momentum (how original).
sgd = SGD(lr=0.001, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd)
```

cifar100_cnn_dl 结果：



```
Epoch 45/50
50000/50000 [==============================] - 33s - loss: 2.5053 - acc: 0.3545 - val_loss: 2.2624 - val_acc: 0.4114
Epoch 46/50
50000/50000 [==============================] - 33s - loss: 2.5246 - acc: 0.3518 - val_loss: 2.2108 - val_acc: 0.4231
Epoch 47/50
50000/50000 [==============================] - 33s - loss: 2.5068 - acc: 0.3538 - val_loss: 2.2649 - val_acc: 0.4095
Epoch 48/50
50000/50000 [==============================] - 33s - loss: 2.5038 - acc: 0.3574 - val_loss: 2.2118 - val_acc: 0.4264
Epoch 49/50
50000/50000 [==============================] - 33s - loss: 2.5000 - acc: 0.3548 - val_loss: 2.2677 - val_acc: 0.4133
Epoch 50/50
50000/50000 [==============================] - 33s - loss: 2.4899 - acc: 0.3609 - val_loss: 2.3722 - val_acc: 0.3856
root@young-pc:/home/young/ImageClassify#
```

ifar100_cnn_dl1 结果：



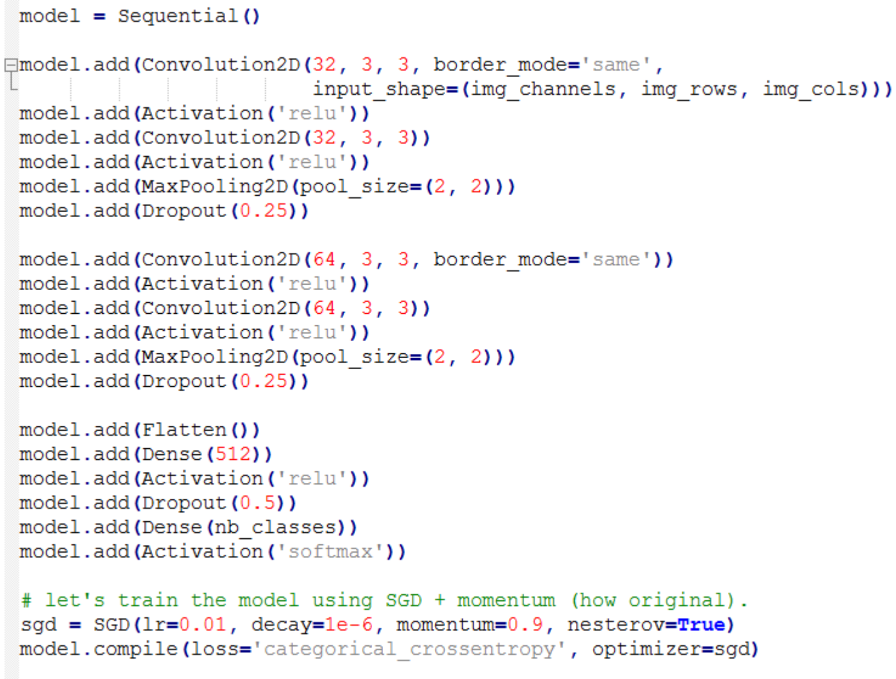
```
Epoch 45/50
50000/50000 [==============================] - 54s - loss: 1.9008 - acc: 0.4793 - val_loss: 1.8743 - val_acc: 0.4967
Epoch 46/50
50000/50000 [==============================] - 54s - loss: 1.8933 - acc: 0.4827 - val_loss: 1.8440 - val_acc: 0.5047
Epoch 47/50
50000/50000 [==============================] - 54s - loss: 1.8754 - acc: 0.4848 - val_loss: 1.8288 - val_acc: 0.5069
Epoch 48/50
50000/50000 [==============================] - 54s - loss: 1.8687 - acc: 0.4862 - val_loss: 1.8331 - val_acc: 0.5066
Epoch 49/50
50000/50000 [==============================] - 54s - loss: 1.8624 - acc: 0.4886 - val_loss: 1.8274 - val_acc: 0.5077
Epoch 50/50
50000/50000 [==============================] - 54s - loss: 1.8501 - acc: 0.4938 - val_loss: 1.7869 - val_acc: 0.5177
root@young-pc:/home/young/ImageClassify#
```

根据 cifar10 得到的最好测试模型训练 cifar100 数据集得到的效果，比根据 cifar10 参考模型训练 cifar100 的效果要好很多，提高了 13 个百分点，但是，两者的不足是测试成功率和交叉验证成功率都较低。如果需要对 cifar100 数据集训练和测试，还需要探索新的深度学习网络结构。

测试展示：

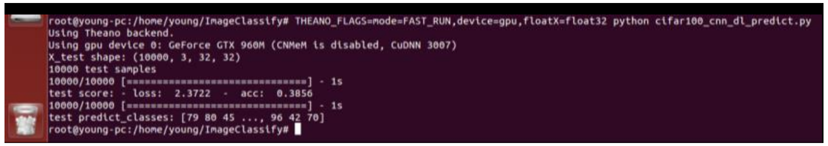
测试代码	网络结构	测试数据	loss	acc	测试时间/s
cifar100_cnn_dl	参考结构	Cifar100 TestSet: 10000	2.3722	0.3856	1
cifar100_cnn_dl1	卷积核：3*3->5*5 最后一个激活函数：’relu’->’ tanh’ SGD: lr = 0.01 -> lr = 0.001 增加卷积层：第二部分	Cifar100 TestSet: 10000	1.7869	0.5177	2

cifar100_cnn_dl 网络结构：



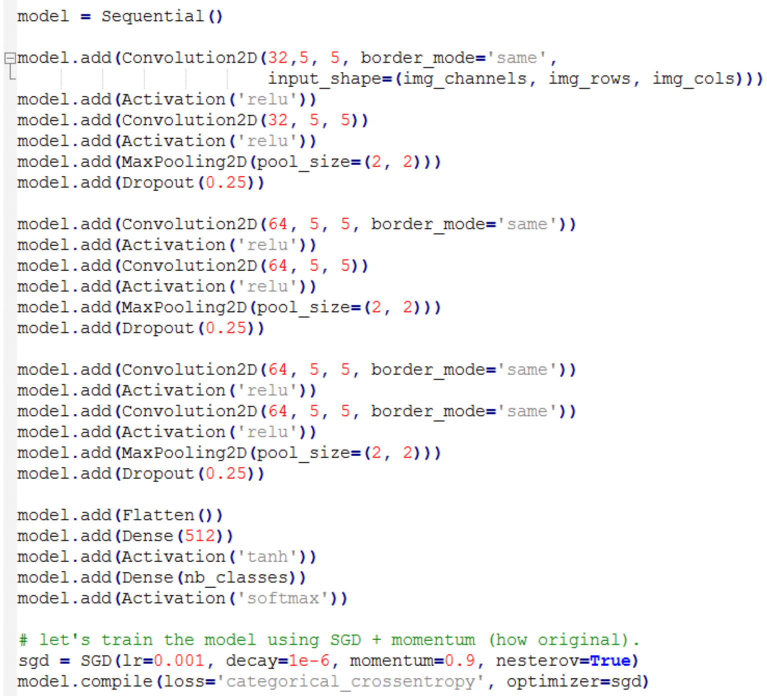
```
model = Sequential()

model.add(Convolution2D(32, 3, 3, border_mode='same',
                        input_shape=(img_channels, img_rows, img_cols)))
model.add(Activation('relu'))
model.add(Convolution2D(32, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(64, 3, 3, border_mode='same'))
model.add(Activation('relu'))
model.add(Convolution2D(64, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))

# let's train the model using SGD + momentum (how original).
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd)
```

cifar100_cnn_dl 测试结果：



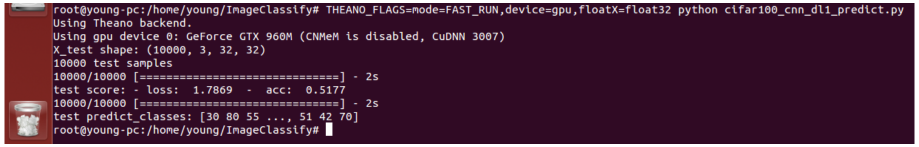
```
root@young-pc:/home/young/ImageClassify# THEANO_FLAGS=mode=FAST_RUN,device=gpu,floatX=float32 python cifar100_cnn_dl_predict.py
Using Theano backend.
Using gpu device 0: GeForce GTX 960M (CNMeM is disabled, CuDNN 3007)
X_test shape: (10000, 3, 32, 32)
10000 test samples
10000/10000 [==============================] - 1s
test score: - loss:  2.3722  -  acc:  0.3856
10000/10000 [==============================] - 1s
test predict_classes: [79 80 45 ..., 96 42 70]
root@young-pc:/home/young/ImageClassify# █
```

cifar100_cnn_dl1 网络结构：



```
model = Sequential()

model.add(Convolution2D(32,5, 5, border_mode='same',
                        input_shape=(img_channels, img_rows, img_cols)))
model.add(Activation('relu'))
model.add(Convolution2D(32, 5, 5))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(64, 5, 5, border_mode='same'))
model.add(Activation('relu'))
model.add(Convolution2D(64, 5, 5))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(64, 5, 5, border_mode='same'))
model.add(Activation('relu'))
model.add(Convolution2D(64, 5, 5, border_mode='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('tanh'))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))

# let's train the model using SGD + momentum (how original).
sgd = SGD(lr=0.001, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd)
```

cifar100_cnn_dl1 测试结果：



```
root@young-pc:/home/young/ImageClassify# THEANO_FLAGS=mode=FAST_RUN,device=gpu,floatX=float32 python cifar100_cnn_dl1_predict.py
Using Theano backend.
Using gpu device 0: GeForce GTX 960M (CNMeM is disabled, CuDNN 3007)
X_test shape: (10000, 3, 32, 32)
10000 test samples
10000/10000 [==============================] - 2s
test score: - loss:  1.7869  -  acc:  0.5177
10000/10000 [==============================] - 2s
test predict_classes: [30 80 55 ..., 51 42 70]
root@young-pc:/home/young/ImageClassify# █
```

小结：实验表明，改进过的结构可以将测试成功率提高 13 个百分点，效果还是不错的。

附录：

1. 文档中所有源文件都在文件夹 ImageClassifySummary_Cifar100 里，实验结果截图在 resultimage 文件夹下。
2. 保存的模型和测试结果均以其对应源代码文件名区别。例如，源代码为 cifar100_cnn_dl.py，训练保存的模型文件为 my_model_architecture_cifar100_dl.json，模型权重为 my_model_weights_cifar100_dl.h5，测试代码为 cifar100_cnn_dl_predict.py，测试预测结果为 test_predict_cifar100_dl.txt，测试分类结果为 test_predict_classes_cifar100_dl.txt。其他源文件的测试结果文件名与之类似。
3. 测试平台参数：

   CPU：Inter(R) Core(TM) i7-6700HQ CPU @ 2.60

   GPU：GeForce GTX 960M

   Cuda：7.5

   CuDNN：7.0

   Python：2.7.6

   Theano：0.8.1

   Keras：0.3.3