注意事项：

1. 进入AI-1系统，密码100875（不要进入其他系统）。
2. 由于本课程和AI-1课程共享一个系统，所以还有别的同学在别的课程的作业，所以**不要删除系统内任何已经存在的文件!**
3. 由于下一组同学和大家使用相同的系统，所以请大家离开教室前**务必做好备份（U盘or邮件）并删除**208机房上你的作业。
4. 每一次上机后，助教都会检查每一台电脑中作业是否删除，**如果还存在，助教会删除。**

程序保存目录

- home\目录下，新建“image_processing”文件夹。
- 这里面你们可以随意**鼓捣**！

让我们先安装一些依赖库

“**Ctrl+Alt+t**” 进入终端界面。

1. 输入 **sudo pip install matplotlib** 回车
   输入密码100875
2. 输入**sudo apt-get install python-tk**回车

2. 输入**sudo pip install h5py**回车

Deep Learning for Computer Vision: Experiment1:CNN

Libao Zhang, Jie Ma

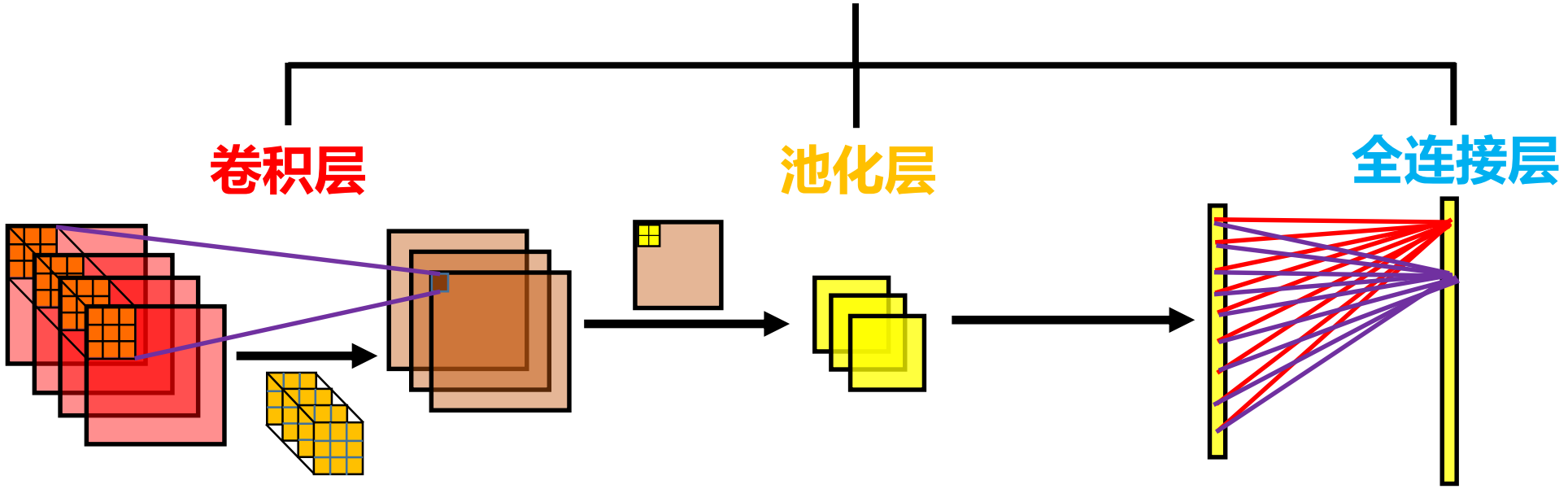
2017.12.1

内容

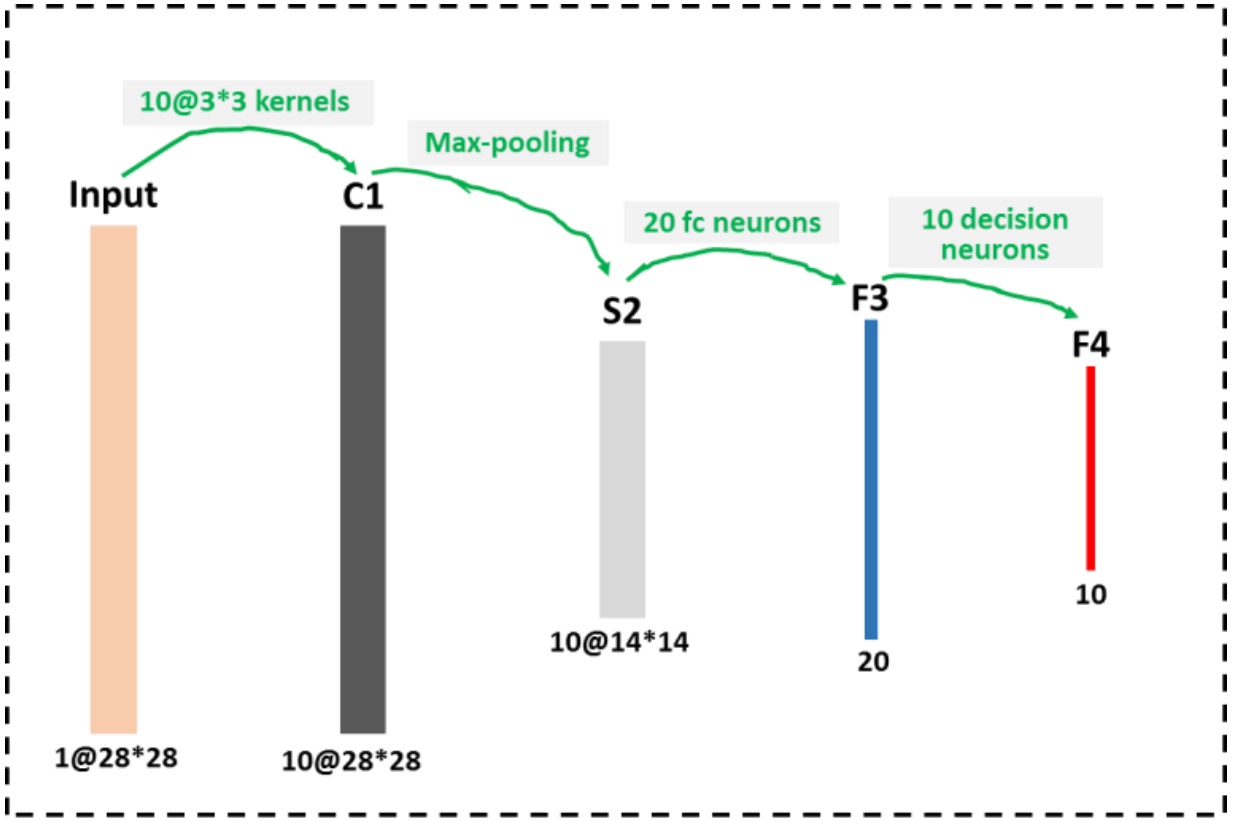
- 实验原理
- 网络结构
- 实验步骤

实验原理

- 将**卷积层**和**池化层**堆叠，完成图像的*特征提取*，通过**全连接层**完成特征的*非线性分类*。



网络结构



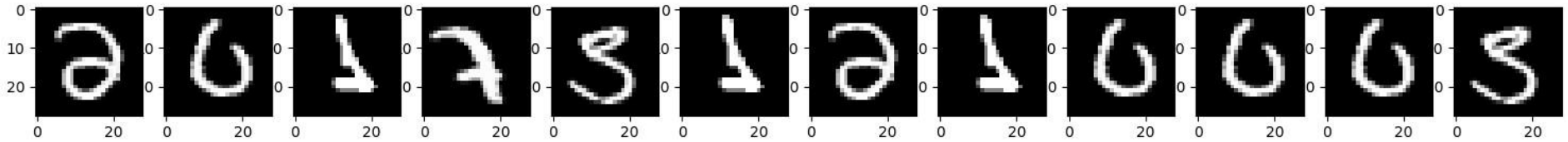
The architecture of CNN

Input：1@28*28

layer1：10@3*3	featuremap1：10@28*28
layer2：max-pooling	featuremap2：???
layer3：???	featuremap3：???
layer4：???	featuremap4：???

实验步骤

1. 完成 demo1 中”1.1网络结构“部分，根据 demo1 中的网络结构写出每一层的网络结构和参数个数。
2. 运行 demo1.py，训练网络，完成“1.2网络训练”。
3. 自己完成代码 demo2.py，找出训练好的模型中判错的且能组成自己学号的数字，完成“1.3 找出 10 副判错图像”。 例如： 助教的学号是 201731210003 输出的图像为：



这些数字的 groundtruth 为：201731210003 但是都被模型判错了。

4.改写demo1中的核心代码，自己设计一个层数大于 10 且准确率大于 0.99 的网络，并完成 2.1-2.3 。

5.结合课件, demo1.py, demo2.py, 完成“问题”部分。

Demo1.py

类别	One of C									
7	0	0	0	0	0	0	0	1	0	0
1	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0

```python
#batch size, num classes and epochs
batch_size = 128
num_classes = 10
epochs = 10

# input image dimensions
img_rows, img_cols = 28, 28

# load minst and reshape the input data
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
input_shape = (img_rows, img_cols, 1)


x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.np_utils.to_categorical(y_train, num_classes)
y_test = keras.utils.np_utils.to_categorical(y_test, num_classes)
```

确定batch_size大小
根据分类任务，确定类别个数。
确定迭代次数。

输入待识别图像的尺寸。

载入mnist数据集，重新排列每张图像的像素点位置，使其和网络的输入维度相同。

将标签变为one-of-c 标签

建立一个顺序模型

```python
##################################################################
model = Sequential()
model.add(Conv2D(10,(3,3),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(20, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
##################################################################

#model compile:loss function, optimiazer and metrics
model.compile(loss=keras.metrics.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

#visualize the architecture of CNN
model.summary()

#train the model
model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
          verbose=1, validation_data=(x_test, y_test))

#save weights
model.save('mnist_demo1.h5')

#evaluate the model
score_test = model.evaluate(x_test, y_test, verbose=0)
score_train = model.evaluate(x_train,y_train,verbose=0)
print(score_test)
print(score_train)
```

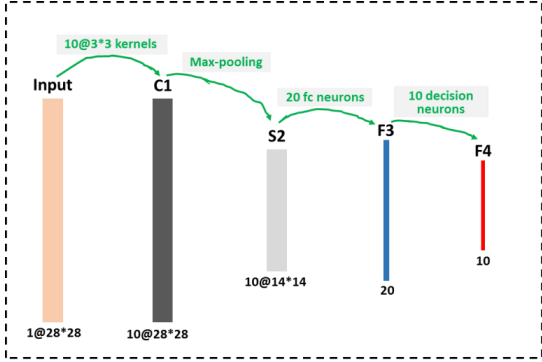
损失函数，优化器，度量

训练

保存参数

评估模型

构建一个CNN



The architecture of CNN

增加第一层，用10个3*3的卷积核和Relu激活函数构建第一层，需要告诉网络输入的维度input_shape。

```
##################################################################

model = Sequential()
model.add(Conv2D(10,(3,3),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(20, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

##################################################################
```

构建一个顺序模型。

为了之后连接全连接层，把现在的特征图拉成列向量。

加入20个节点的全连接层。

由于分10类，所以分类层10个节点。

增加max-pooling层，每2*2的点使用max函数映射到一个点。

```
input_shape = (img_rows, img_cols, 1)
```

Demo2.py

```python
#batch_size, num_classes and epochs
batch_size = 128
num_classes = 10
epochs = 10

# input image dimensions
img_rows, img_cols = 28, 28

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.np_utils.to_categorical(y_train, num_classes)
y_test = keras.utils.np_utils.to_categorical(y_test, num_classes)
```

```python
#construct a convolutional neuron network
#if you change the architecture, you should change the following code
###################################################################

model = Sequential()
model.add(Conv2D(10,(3,3),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(20, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

###################################################################


#model compile:loss function, optimiazer and metrics
model.compile(loss=keras.metrics.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

#visualize the architecture of CNN
model.summary()
```

```python
#load the well-trained parameters, including weights and bias
model.load_weights('mnist_demo1.h5')
```

载入权重

```python
#obtain the model prediction in testing set
model_prediction = model.predict(x_test)
```

查看输出

Demo2.py——tips

```
#########################################################################

plt.figure(figsize=(20, 4))
b = [2,0,1,7,3,1,2,1,0,0,0,3]
for j in range(len(b)):
    for i in range(10000):
        pos_model = np.argmax(model_prediction[i,:])
                                                                    [j]))

plt.show()
```

学号

对于学号的各个位，还有每一个样本进行双重循环。