《深度学习构架》实验二 卷积神经网络的深度学习架构实现

一、实验目的

1. 掌握卷积神经网络的基本用法；
2. 设计并实现一个简单卷积神经网络实例；
3. 掌握常见CNN架构的原理和结构；
4. 能够用TensorFlow或PyTorch实现简单卷积神经网络的训练、验证和检测过程；
5. 用卷积神经网络完成对数据集MNIST的分类任务。

二、实验环境

1. 笔记本电脑，Linux或MacOs系统；
2. Python3.x.

三、实验内容

1. 了解卷积神经网络的基本结构
2. 掌握卷积神经网络的基本用法，设计并实现一个简单卷积神经网络实例；
3. 熟悉卷积神经网络的卷积核的用法；
4. 熟悉卷积神经网络池化层的用法；
5. 了解典型的CNN架构；
6. 用CNN实现MNIST数字识别；
7. 用TensorFlow或PyTorch完成卷积神经网络的各个阶段。

四、主要实验步骤

1. 卷积神经网络的基本结构

- 每个输入图像通常表示为[高度，宽度，通道]形状的3D张量。
- 小批量表示为[小批量，高度，宽度，通道]形状的4D张量。
- 卷积层的权重表示为[f_h，f_w，f_n，f_n']形状的4D张量。
- 卷积层的偏置项简单地表示为[f_n]的形状1D张量。

2. 卷积神经网络的Filters(卷积核/过滤器)

神经元的权重可以用大小为感受区的图像来表示，我们把它称为过滤器（或卷积内核）。使用下列代码实现对两种过滤器的对比分析。

install tensorflow 2.0 and other modules

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple ten
sorflow
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple skl
earn
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple mat
plotlib
pip isntall -i https://pypi.tuna.tsinghua.edu.cn/simple pil
low
```

In [2]:
```
import tensorflow as tf
tf.__version__
```

Out[2]:
```
'2.0.0'
```

In [1]:
```
from sklearn.datasets import load_sample_image
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

#tf.enable_eager_execution()

def plot_image(image):
    plt.imshow(image, cmap="gray", interpolation="nearest")

def plot_color_image(image):
    plt.imshow(tf.cast(image,np.uint8),interpolation="nearest")

china = load_sample_image("china.jpg")
image = china[150:220, 130:250]
height, width, channels = image.shape
image_grayscale = image.mean(axis=2).astype(np.float32)
images = image_grayscale.reshape(1, height, width, 1)

fmap = np.zeros(shape=(7, 7, 1, 2), dtype=np.float32)
fmap[:, 3, 0, 0] = 1
fmap[3, :, 0, 1] = 1
plot_image(fmap[:, :, 0, 0])
plt.show()
plot_image(fmap[:, :, 0, 1])
plt.show()

feature_maps = tf.constant(fmap)
convolution = tf.nn.conv2d(images, feature_maps,
                           strides=[1,1,1,1],
                           padding="SAME")
```

...

In [2]:
```
plot_image(images[0, :, :, 0])
plt.show()

plot_image(convolution[0, :, :, 0])
plt.show()

plot_image(convolution[0, :, :, 1])
plt.show()
```

...

3. 卷积神经网络池化层

池化层的目标是对输入图像进行二次采样（即缩小），以减少计算量、内存占用和参数数量（减少过拟合风险）。

- 和卷积层一样，池化层中的每个神经元都和前一层神经元的输出相连，位于一个小的矩形感受区内。
- 然而，池化神经元没有权重，它所做的就是使用聚合函数（如max_pool()或avg_pool()）聚合输入。

下列代码实现了一个2x2内核的池化层，请分析池化层的作用。

In [3]:
```
dataset = np.array([china], dtype=np.float32)
max_pool = tf.nn.max_pool(dataset, ksize=[1,2,2,1],
                          strides=[1,2,2,1],
                          padding="VALID")

plt.figure(figsize=(16, 12))

plt.subplot(121)
plot_color_image(dataset[0])

plt.subplot(122)
plot_color_image(max_pool[0])

plt.show()
```

...

4. 典型的CNN架构

- 一些卷积层（每个通常后跟一个ReLU层），
- 然后是一个池化层，
- 然后是另外几个卷积层（+ ReLU），
- 然后是另一个池层.

下面是最近几年出现的一些优秀的CNN架构。

- 经典LeNet-5架构（1998）
- AlexNet（2012）
- GoogLeNet（2014）
- ResNet（2015）

5. MNIST CNN实现

分析并运行下列代码，完成如下任务：

1. 阅读代码，分析CNN和全链接网络的相同点和不同点；
2. 运行代码，从运行时间、准确性等方面与全连接网络进行比较；
3. 尝试提出改进方案，提高模型的准确率。

In [4]:
```
import os
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D
import numpy as np

(X_tarin, y_train), (X_test, y_test) = mnist.load_data()
X_train4D = X_tarin.reshape(X_tarin.shape[0], 28, 28, 1).astype('fl
X_test4D = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float

X_train4D_Normalize = X_train4D / 255 # 归一化
X_test4D_Normalize = X_test4D / 255
y_trainOnehot = to_categorical(y_train)
y_testOnehot = to_categorical(y_test)
```

In [20]:
```
# 建立模型
model = Sequential()

# 一层卷积
model.add(Conv2D(filters=16,
                 kernel_size=(5, 5),
                 padding='SAME',
                 input_shape=(28, 28, 1),
                 activation='relu'))

# 池化层1
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# 二层卷积
model.add(Conv2D(filters=32,
                 kernel_size=(5, 5),
                 padding='SAME',
                 activation='relu'))
```

In [21]:
```
# 池化层2
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(filters=64,
                 kernel_size=(5, 5),
                 padding='same',
                 activation='relu'))
model.add(Conv2D(filters=128,
                 kernel_size=(5, 5),
                 padding='same',
                 activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(10, activation='softmax'))

model.summary()
```

...

In [22]:
```
# 编译模型
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [23]:
```
#训练模型
train_history = model.fit(x=X_train4D_Normalize,
                          y=y_trainOnehot,
                          validation_split=0.2,
                          batch_size=300,
                          epochs=10,
                          verbose=2)
```

...

In [24]:
```
#评估模型
model.evaluate(X_test4D_Normalize, y_testOnehot)[1]

#预测
prediction = model.predict_classes(X_test4D_Normalize)
```

...

五、实验要求

1. 按顺序完成上述操作要求，掌握TensorFlow的基本用法；
2. 记录实验过程及结果；
3. 完成实验报告，双面打印。
4. 每个人独立完成实验报告，内容如有雷同则所有雷同报告将都会被判为不及格。