使用卷积神经网络实现手写体数字识别

数据集：MNIST

深度学习框架：PaddlePaddle（百度飞桨）

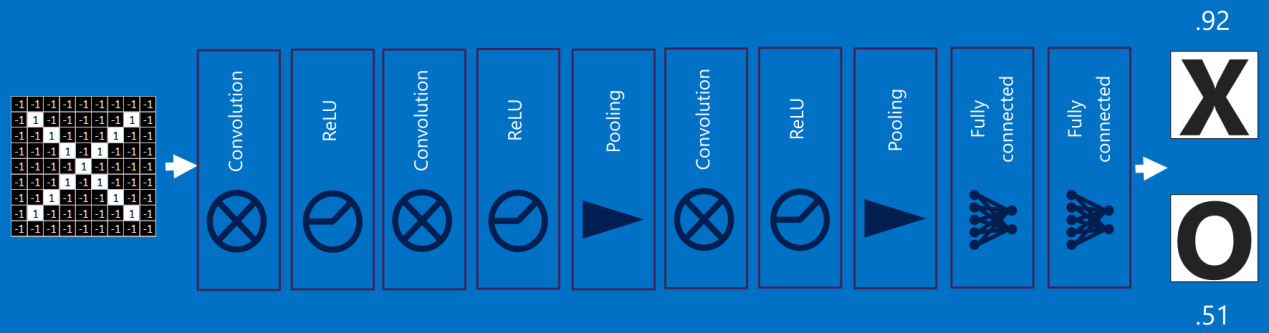
MNIST一览：



PaddlePaddle介绍：



卷积神经网络（CNN）构建过程：



依次为：

原始图像输入（Tensor）

↓

卷积运算（Conv）

↓

稀疏化（ReLU）

↓

池化

↓

全连接

↓

投票（概率输出）

依赖包导入：

import numpy as np

import matplotlib.pyplot as plt

import paddle

from paddle.nn import Linear

import paddle.nn.functional as F

from paddle.vision.datasets import MNIST

模型定义：

# CNN网络定义

class MNISTNet(paddle.nn.Layer):

    def \_\_init\_\_(self):

        super(MNISTNet, self).\_\_init\_\_()

        # 卷积层定义

        self.conv1 = paddle.nn.Conv2D(in\_channels=1, out\_channels=20, kernel\_size=5, stride=1, padding=2)

        # 池化层定义

        self.pool1 = paddle.nn.MaxPool2D(kernel\_size=2, stride=2)

        self.conv2 = paddle.nn.Conv2D(in\_channels=20, out\_channels=20, kernel\_size=5, stride=1, padding=2)

        self.pool2 = paddle.nn.MaxPool2D(kernel\_size=2, stride=2)

        self.fc = Linear(in\_features=980, out\_features=10)

    def forward(self, inputs, label=None, check\_shape=False, check\_content=False):

        outputs\_conv1 = self.conv1(inputs)

        outputs\_conv1\_relu = F.relu(outputs\_conv1)

        outputs\_pool1 = self.pool1(outputs\_conv1\_relu)

        outputs\_conv2 = self.conv2(outputs\_pool1)

        outputs\_conv2\_relu = F.relu(outputs\_conv2)

        outputs\_pool2 = self.pool2(outputs\_conv2\_relu)

        outputs\_pool2 = paddle.reshape(outputs\_pool2, [outputs\_pool2.shape[0], -1])

        outputs\_fc = self.fc(outputs\_pool2)

        if label is not None:

            acc = paddle.metric.accuracy(input=F.softmax(outputs\_fc), label=label)

            return outputs\_fc, acc

        else:

            return outputs\_fc

加载数据：

def split\_set(mnist):

    imgs = []

    labels = []

    for img in mnist.images:

        imgs.append(img)

    for label in mnist.labels:

        labels.append(label[0])

    return imgs, labels

# 训练数据加载

def load\_data\_opt(mode='train'):

    # 下载数据集

    train\_set = MNIST(mode='train')

    test\_set = MNIST(mode='test')

    # 图像宽高

    IMG\_ROWS = 28

    IMG\_COLS = 28

    if mode == 'train':

        imgs, labels = split\_set(train\_set)

    elif mode == 'test':

        imgs, labels = split\_set(test\_set)

    imgs\_length = len(imgs)

    index\_list = list(range(imgs\_length))

    BATCHSIZE = 100

    # 数据投喂

    def data\_generator():

        if mode == 'train':

            random.shuffle(index\_list)

        imgs\_list = []

        labels\_list = []

        for i in index\_list:

            img = np.reshape(imgs[i], [1, IMG\_ROWS, IMG\_COLS]).astype('float32')

            label = np.reshape(labels[i], [1]).astype('int64')

            imgs\_list.append(img)

            labels\_list.append(label)

            if len(imgs\_list) == BATCHSIZE:

                yield np.array(imgs\_list), np.array(labels\_list)

                imgs\_list = []

                labels\_list = []

        if len(imgs\_list) > 0:

            yield np.array(imgs\_list), np.array(labels\_list)

    return data\_generator

train\_loader = load\_data\_opt('train')

模型训练：

# 模型训练

def train(model):

    model.train()

    # 学习率优化策略：动量 + 自适应

    opt = paddle.optimizer.Adam(learning\_rate=0.01, parameters=model.parameters())

    # 训练轮次

    EPOCH\_NUM = 10

    for epoch\_id in range(EPOCH\_NUM):

        for batch\_id, data in enumerate(train\_loader()):

            images, labels = data

            images = paddle.to\_tensor(images)

            labels = paddle.to\_tensor(labels)

            predicts, acc = model(images, labels)

            # 损失函数定义：交叉熵

            loss = F.cross\_entropy(predicts, labels)

            avg\_loss = paddle.mean(loss)

            if batch\_id % 200 == 0:

                print(f'轮次：{epoch\_id}, 批次：{batch\_id}, 损失值：{avg\_loss.numpy()}, 精度：{acc.numpy()}')

            # 反向传播梯度

            avg\_loss.backward()

            opt.step()

            opt.clear\_grad()

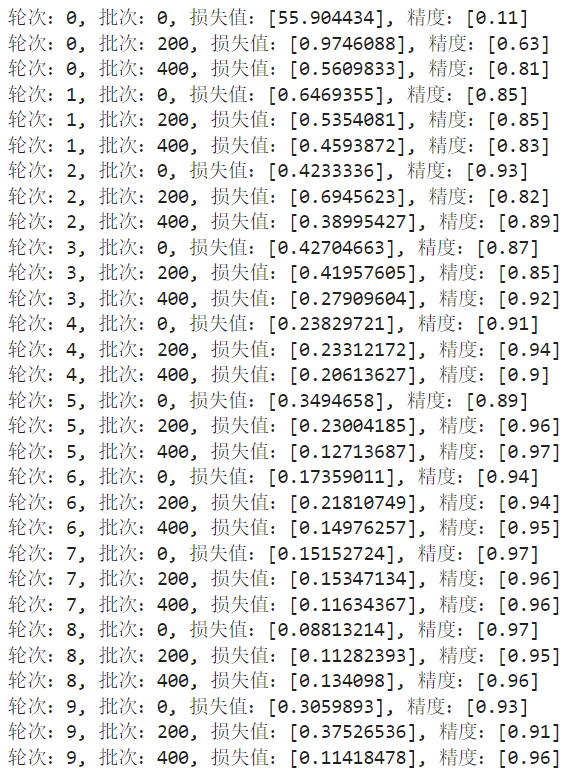
    # 模型存储

    paddle.save(model.state\_dict(), 'mnist.pdparams')

model = MNISTNet()

train(model)

训练输出：



验证模型：

# 测试模型

def test\_model(model):

    param\_dict = paddle.load('mnist.pdparams')

    model.load\_dict(param\_dict)

    model.eval()

    eval\_loader = load\_data\_opt(mode='test')

    acc\_set = []

    avg\_loss\_set = []

    for batch\_id, data in enumerate(eval\_loader()):

        images, labels = data

        images = paddle.to\_tensor(images)

        labels = paddle.to\_tensor(labels)

        predicts, acc = model(images, labels)

        loss = F.cross\_entropy(input=predicts, label=labels)

        avg\_loss = paddle.mean(loss)

        acc\_set.append(float(acc.numpy()))

        avg\_loss\_set.append(float(avg\_loss.numpy()))

    acc\_val\_mean = np.array(acc\_set).mean()

    avg\_loss\_val\_mean = np.array(avg\_loss\_set).mean()

    print(f'损失值：{avg\_loss\_val\_mean}, 精度：{acc\_val\_mean}')

model = MNISTNet()

test\_model(model)

验证结果：



单次测试：

ds = MNIST(mode='test')

img = ds.images[84]

img\_for\_show = img.copy()

img = np.reshape(img, [1, 1, 28, 28]).astype('float32')

img = paddle.to\_tensor(img)

model\_dict = paddle.load('mnist.pdparams')

model.load\_dict(model\_dict)

model.eval()

result = model(img)

print('本次预测的数字是：', np.argsort(result.numpy())[0][-1])

plt.imshow(img\_for\_show.reshape(28, 28))

预测结果：

