给定mnist手写数据集（图像大小为28\*28），请编写一个卷积神经网络模型完成手写数字的分类。

任务一：搭建一个模型，由一层卷积层，一层最大池化层，一层全连接层组成。卷积层包含10个大小为3\*3的卷积核，步长为1，卷积后特征图尺寸不变；池化层大小为2\*2，步长为2；全连接层输出图像属于0到9每一类的概率。部分代码已给出。附：PaddlePaddle常用函数表。

import numpy as np

import paddle

import paddle.fluid as fluid

import paddle.fluid.dygraph as digraph

#请自行补充需要的依赖

# 设置数据读取器，读取MNIST数据训练集

trainset = paddle.dataset.mnist.train()

testset = paddle.dataset.mnist.test()

# 包装数据读取器，每次读取的数据数量设置为batch\_size=8

train\_loader = paddle.batch(trainset, batch\_size=8)

valid\_loader = paddle.batch(testset, batch\_size=8)

#图像数据形状为（8，784）

# 定义mnist数据识别网络结构

class model\_mnist(fluid.dygraph.Layer):

def \_\_init\_\_(self, name\_scope, num\_classes=1):

super(model\_mnist, self).\_\_init\_\_(name\_scope)

#请完成代码

def forward(self, x):

#请完成代码

任务二：训练模型。自行选择合适的优化方法与参数，利用数据集训练模型。要求训练5轮，每一轮中每隔1000个batch输出训练损失；每一轮结束后输出模型在验证集上的准确率与损失。

def train(model):

model.train()

epoch\_num = 5

#请完成代码,代码中需包含模型验证部分

if \_\_name\_\_ == '\_\_main\_\_':

# 创建模型

with fluid.dygraph.guard():

model = model\_mnist("mnist", num\_classes=10)

#启动训练过程

train(model)

import numpy as np

import paddle

import paddle.fluid as fluid

import paddle.fluid.dygraph as digraph

from paddle.fluid.dygraph.nn import Conv2D, Pool2D

# 设置数据读取器，读取MNIST数据训练集

trainset = paddle.dataset.mnist.train()

testset = paddle.dataset.mnist.test()

# 包装数据读取器，每次读取的数据数量设置为batch\_size=8

train\_loader = paddle.batch(trainset, batch\_size=8)

valid\_loader = paddle.batch(testset, batch\_size=8)

#图像数据形状为（8，784）

# 定义网络结构

class model\_mnist(fluid.dygraph.Layer):

def \_\_init\_\_(self, name\_scope, num\_classes=1):

super(model\_mnist, self).\_\_init\_\_(name\_scope)

self.conv = Conv2D(num\_channels=1, num\_filters=10, padding=1, filter\_size=3, act='relu')

self.pool = Pool2D(pool\_size=2, pool\_stride=2, pool\_type='max')

self.fc = Linear(input\_dim=1960, output\_dim=10, act='softmax')

def forward(self, x):

x = self.conv(x)

x = self.pool(x)

x = fluid.layers.flatten(x)

x = self.fc(x)

return x

def train(model):

model.train()

epoch\_num = 5

opt = fluid.optimizer.Adam(learning\_rate=0.001, parameter\_list=model.parameters())

for epoch in range(epoch\_num):

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

# 调整输入数据形状和类型

x\_data = np.array([item[0] for item in data], dtype='float32').reshape(-1, 1, 28, 28)

y\_data = np.array([item[1] for item in data], dtype='int64').reshape(-1, 1)

# 将numpy.ndarray转化成Tensor

img = fluid.dygraph.to\_variable(x\_data)

label = fluid.dygraph.to\_variable(y\_data)

# 计算模型输出

logits = model(img)

# 计算损失函数

loss = fluid.layers.softmax\_with\_cross\_entropy(logits, label)

avg\_loss = fluid.layers.mean(loss)

if batch\_id % 1000 == 0:

print("epoch: {}, batch\_id: {}, loss is: {}".format(epoch, batch\_id, avg\_loss.numpy()))

avg\_loss.backward()

opt.minimize(avg\_loss)

model.clear\_gradients()

model.eval()

accuracies = []

losses = []

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

# 调整输入数据形状和类型

x\_data = np.array([item[0] for item in data], dtype='float32').reshape(-1, 1, 28, 28)

y\_data = np.array([item[1] for item in data], dtype='int64').reshape(-1, 1)

# 将numpy.ndarray转化成Tensor

img = fluid.dygraph.to\_variable(x\_data)

label = fluid.dygraph.to\_variable(y\_data)

# 计算模型输出

logits = model(img)

pred = fluid.layers.softmax(logits)

# 计算损失函数

loss = fluid.layers.softmax\_with\_cross\_entropy(logits, label)

acc = fluid.layers.accuracy(pred, label)

accuracies.append(acc.numpy())

losses.append(loss.numpy())

print("[validation] accuracy/loss: {}/{}".format(np.mean(accuracies), np.mean(losses)))

model.train()

if \_\_name\_\_ == '\_\_main\_\_':

# 创建模型

with fluid.dygraph.guard():

model = model\_mnist("mnist", num\_classes=10)

#启动训练过程

train(model)