

华为“智能基座”系列课程

《深度学习》 卷积网络

版本：2.2



华为技术有限公司

版权所有 © 华为技术有限公司 2021。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编： 518129

网址： <http://e.huawei.com>

目录

1 实验介绍	4
1.1 实验目的	4
1.2 实验清单	4
1.3 实验开发环境	4
1.4 开发平台介绍	5
1.5 背景知识	5
2 正则化实验	错误！未定义书签。
2.1 实验介绍	错误！未定义书签。
2.2 实验环境要求	错误！未定义书签。
2.3 实验总体设计	错误！未定义书签。
2.4 实验过程	错误！未定义书签。
2.4.1 下载调用包	错误！未定义书签。
2.4.2 超参数设定	错误！未定义书签。
2.4.3 建立数据集	错误！未定义书签。
2.4.4 建立网络	错误！未定义书签。
2.4.5 启动训练	错误！未定义书签。
2.5 实验总结	错误！未定义书签。
2.6 创新设计	错误！未定义书签。
3 FashionMnist 分类任务正则化对比实验	9
3.1 实验介绍	9
3.2 实验环境要求	9
3.3 实验总体设计	9
3.4 实验过程	9
3.4.1 数据准备	10
3.4.2 导入实验环境	10
3.4.3 数据读取和预处理	11
3.4.4 训练模型	14
3.4.5 观察总结	20
3.5 实验总结	21
3.6 创新设计	错误！未定义书签。

4 花卉图像分类实验	错误！未定义书签。
4.1 实验介绍	错误！未定义书签。
4.2 实验环境要求	错误！未定义书签。
4.3 实验总体设计	错误！未定义书签。
4.3.1 功能结构	错误！未定义书签。
4.3.2 体系结构	错误！未定义书签。
4.4 实验过程	错误！未定义书签。
4.4.1 导入实验环境	错误！未定义书签。
4.4.2 数据集获取与预处理	错误！未定义书签。
4.4.3 构建 CNN 图像识别模型	错误！未定义书签。
4.4.4 图像分类模型验证	错误！未定义书签。
4.5 实验总结	错误！未定义书签。
4.6 创新设计	错误！未定义书签。
5 LeNet 的手写数字识别实验	22
5.1 实验介绍	22
5.2 实验环境要求	22
5.3 实验准备	22
5.4 实验总体设计	23
5.5 实验过程	23
5.5.1 导入实验环境	23
5.5.2 数据处理	24
5.5.3 定义模型	25
5.5.4 训练	26
5.6 实验总结	27
5.7 创新设计	错误！未定义书签。
6 图像识别全流程代码实战	错误！未定义书签。
6.1 实验介绍	错误！未定义书签。
6.2 实验环境要求	错误！未定义书签。
6.3 实验总体设计	错误！未定义书签。
6.4 实验过程	错误！未定义书签。
6.4.1 数据集获取	错误！未定义书签。
6.4.2 导入实验环境	错误！未定义书签。
6.4.3 读取数据集	错误！未定义书签。
6.4.4 模型构建训练	错误！未定义书签。
6.4.5 模型预测	错误！未定义书签。

6.4.6 模型保存和转换	错误! 未定义书签。
6.4.7 模型部署上线	错误! 未定义书签。
6.5 实验总结	错误! 未定义书签。
7 前沿网络案例- DeepLabv3.....	错误! 未定义书签。
7.1 实验介绍	错误! 未定义书签。
7.2 实验环境要求	错误! 未定义书签。
7.3 背景知识	错误! 未定义书签。
7.3.1 网络介绍	错误! 未定义书签。
7.4 实验步骤	错误! 未定义书签。
7.5 实验总结	错误! 未定义书签。
8 前沿网络案例-YOLOV3.....	错误! 未定义书签。
8.1 实验介绍	错误! 未定义书签。
8.2 实验环境要求	错误! 未定义书签。
8.3 实验准备	错误! 未定义书签。
8.4 背景知识	错误! 未定义书签。
8.4.1 网络介绍	错误! 未定义书签。
8.5 实验步骤	错误! 未定义书签。
8.6 实验总结	错误! 未定义书签。
9 附录：ModelArts 开发环境搭建.....	28

1

实验介绍

卷积网络，也叫做卷积神经网络，是专门用来处理具有类似网格结构的数据的神经网络。例如时间序列数据（可以认为是在时间轴上有规律的采样形成的一维网格）和图像数据（可以看作二维的像素网格）。卷积在诸多应用领域都表现优异。本章主要围绕深度学习的卷积网络而开设的实验。

1.1 实验目的

本章实验的主要目的是掌握卷积网络相关基础知识点。掌握不同神经网络架构的设计原理，熟悉使用 MindSpore 框架实验的一般流程，以及最后将模型部署上线。

1.2 实验清单

表格：实验、简述、难度、软件环境、硬件环境。

表 1-1

实验	简述	难度	软件环境	开发环境
Fashion Mnist正则化前后对比	FashionMnist卷积网络分类识别实验外加正则化效果对比。	中级	MindSpore-1.5-python3.7	ModelArts
Lenet手写数字识别实验	Mnist手写图像Lenet卷积网络分类识别实验。	高级	MindSpore-1.5-python3.7	ModelArts

1.3 实验开发环境

- MindSpore

若选择在华为云 ModelArts 上快速搭建开发环境，可参考文末附录：ModelArts 开发环境搭建。

1.4 开发平台介绍

- MindSpore 最佳匹配昇腾芯片的开源 AI 计算框架，支持 Ascend、GPU、CPU 平台。
MindSpore 官网：<https://www.mindspore.cn>
- ModelArts 是面向开发者的一站式 AI 开发平台，为机器学习与深度学习提供海量数据预处理及半自动化标注、大规模分布式 Training、自动化模型生成，及端-边-云模型按需部署能力，帮助用户快速创建和部署模型，管理全周期 AI 工作流。

具体内容请参考平台介绍 ppt。

1.5 背景知识

- 卷积神经网络结构

卷积神经网络是深度学习与神经网络算法中主流算法之一，主要用于图像识别。其结构图如下：

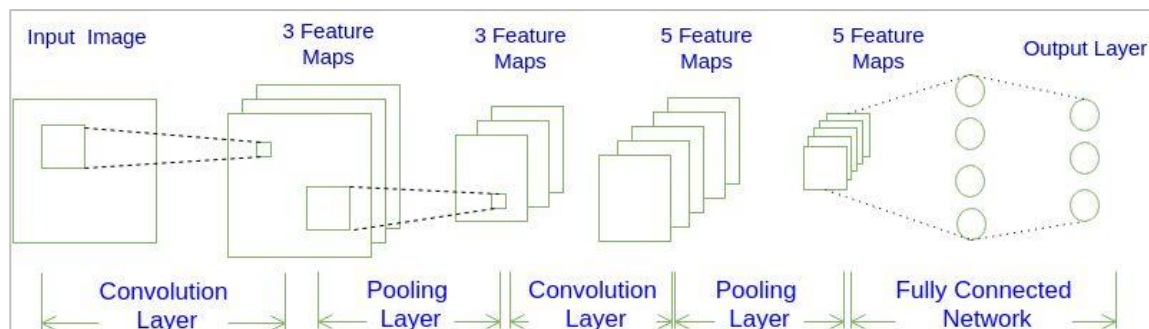
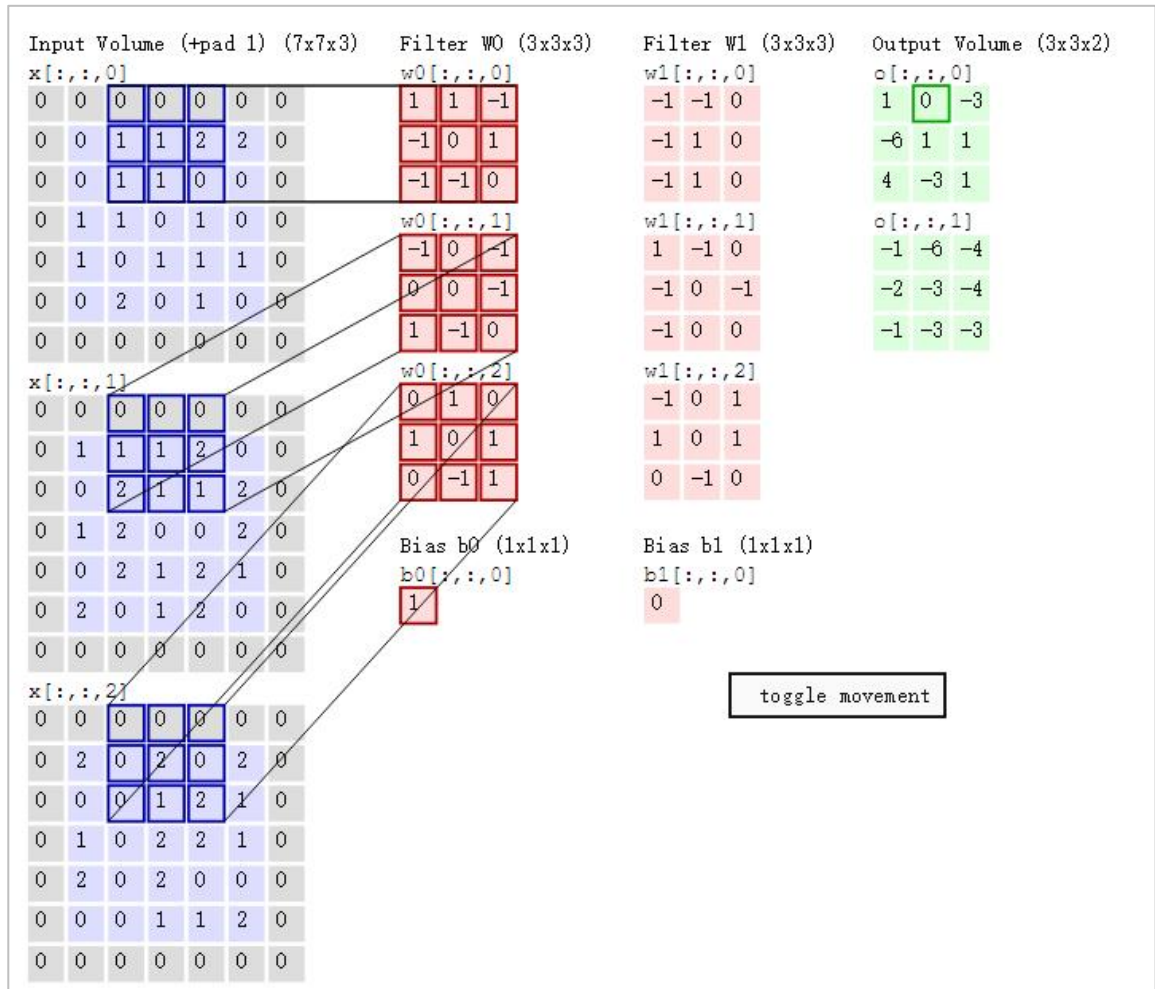


图 1-2 卷积神经网络结构

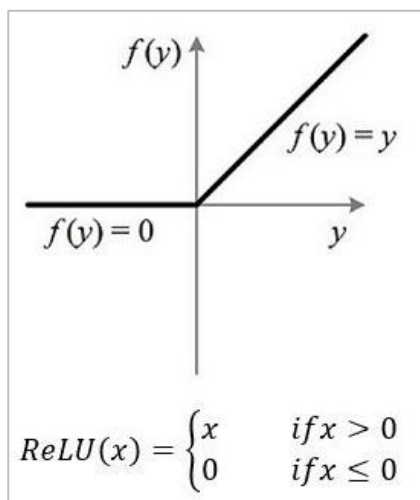
- 卷积层

在卷积层计算过程中，输入是一定区域大小(width*height)的数据，和滤波器 filter（带着一组固定权重的神经元）做内积后得到新的二维数据。

具体来说，滤波器 filter（带着一组固定权重的神经元）通过滑动窗口的方式，对输入图像进行扫描，扫描过程中，对输入图像中的像素值进行点乘，不同的滤波器 filter 会得到不同的输出数据，比如颜色深浅、轮廓。相当于如果想提取图像的不同特征，则用不同的滤波器 filter，提取想要的关于图像的特定信息：颜色深浅或轮廓。



• 激活函数：

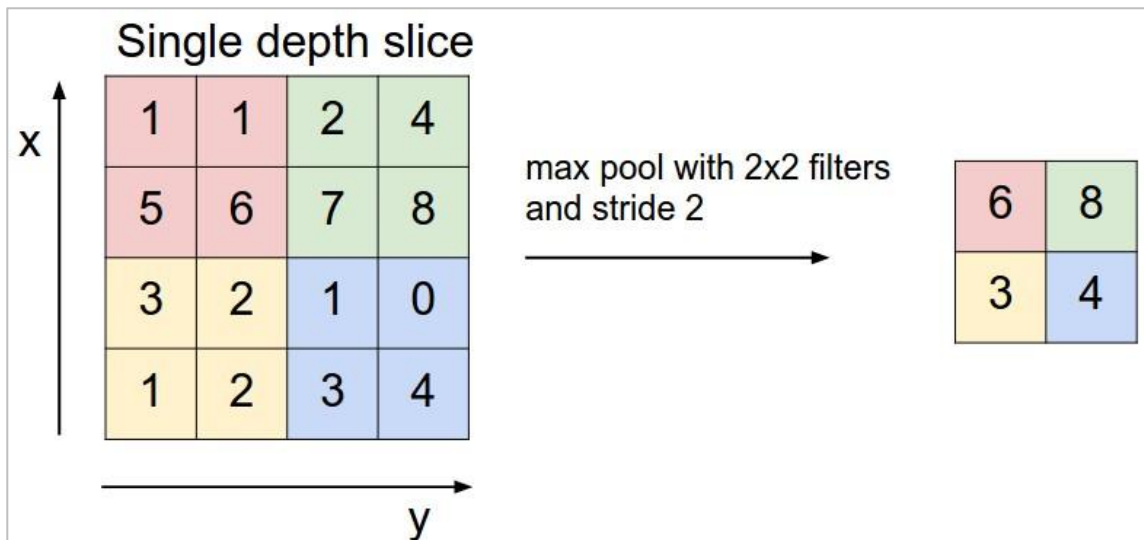


ReLU 函数其实是分段线性函数，把所有的负值都变为 0，而正值不变，这种操作被成为单侧抑制。可别小看这个简单的操作，正因为有了这单侧抑制，才使得神经网络中的神经元也具有了稀疏激活性。

此外，相比于其它激活函数来说，ReLU 有以下优势：对于线性函数而言，ReLU 的表达能力更强，尤其体现在深度网络中；而对于 sigmoid 等激活函数而言，ReLU 由于非负区间的导数为 1，

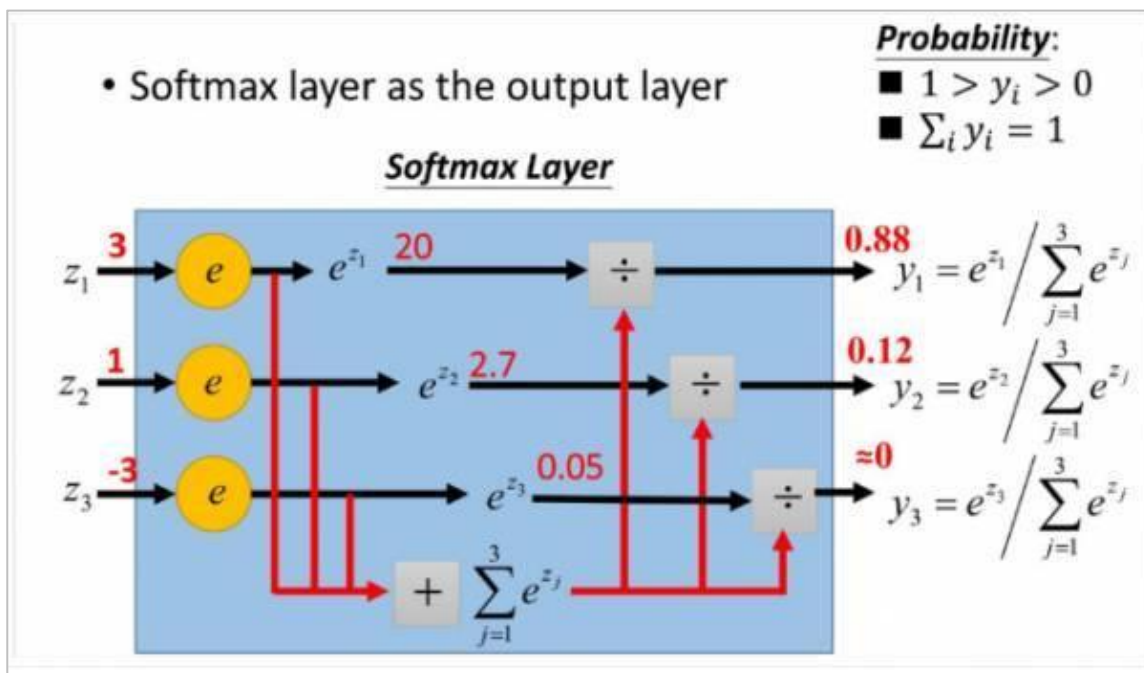
因此可以缓解梯度消失问题(Vanishing Gradient Problem)，使得模型的收敛速度维持在一个稳定状态。这里稍微描述一下什么是梯度消失问题：当梯度小于 1 时，预测值与真实值之间的误差每传播一层会衰减一次，如果在深层模型中使用 sigmoid 作为激活函数，这种现象尤为明显，将导致模型收敛停滞不前。

池化层：



上图所展示的是区域最大，即上图左边部分中 左上角 2x2 的矩阵中 6 最大，右上角 2x2 的矩阵中 8 最大，左下角 2x2 的矩阵中 3 最大，右下角 2x2 的矩阵中 4 最大，所以得到上图右边部分的结果：6 8 3 4。

- 全连接层：



softmax 直白来说就是将原来输出是 3,1,-3 通过 softmax 函数一作用,就映射成为(0,1)的值,而这些值的累和为 1 (满足概率的性质),那么我们就可以将它理解成概率。在训练网络时可以通过对预测值和标签值计算交叉熵损失。

2 FashionMnist 分类任务正则化对比实验

2.1 实验介绍

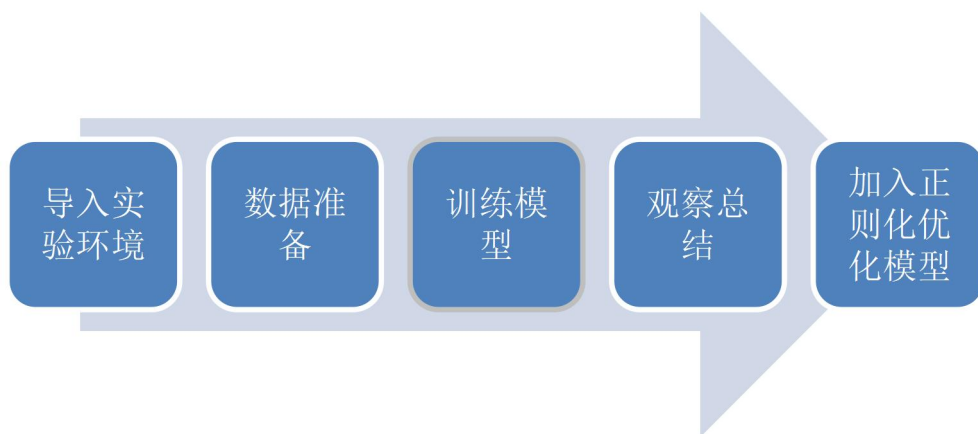
本实验使用 Fashion-MNIST 数据集，它是一个替代 MNIST 手写数字集的图像数据集，由 Zalando（一家德国的时尚科技公司）旗下的研究部门提供。其涵盖了来自 10 种类别的共 7 万个不同商品的正面图片。Fashion-MNIST 的大小、格式和训练集/测试集划分与原始的 MNIST 完全一致。60000/10000 的训练测试数据划分，28x28 的灰度图片。通过上述实验我们对比不同正则化技术效果，此实验我们应用正则化技术做案例分析，并对比有无正则化的训练结果。

2.2 实验环境要求

- ModelArts 平台：Mindspore

若选择在 ModelArts 平台快速搭建，可参考文末附录：ModelArts 开发环境搭建。

2.3 实验总体设计



2.4 实验过程

本节将详细介绍实验的设计与实现。

- 数据准备；
- 导入实验环境；

- 数据读取和预处理；
- 训练模型；
- 观察总结。

2.4.1 数据准备

步骤 1 下载和解压数据集

本实验需要用到的是 Fashion-MNIST 数据集，由于华为云下载该数据集会花费较多时间，因此可以选择从 OBS 上下载并解压到 notebook 的环境中。只下载一次就可。

```
#从 OBS 桶下载数据集
!wget
https://ascend-professional-construction-dataset.obs.myhuaweicloud.com/deep-learning/fashion-mnist.zip
#解压文件
!unzip fashion-mnist.zip
```

Fashion Mnist 有 10 个标签，如下表所示：

标签	描述
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

表 2-1

2.4.2 导入实验环境

步骤 1 导入库

```
import os
import struct
import sys
```

```
from easydict import EasyDict as edict

import matplotlib.pyplot as plt
import numpy as np

import mindspore
import mindspore.dataset as ds
import mindspore.nn as nn
from mindspore import context
from mindspore.nn.metrics import Accuracy
from mindspore.train import Model
from mindspore.train.callback import ModelCheckpoint, CheckpointConfig, LossMonitor, TimeMonitor
from mindspore import Tensor

context.set_context(mode=context.GRAPH_MODE, device_target='Ascend')
```

步骤 2 定义常量

```
cfg = edict({
    'train_size': 60000, # 训练集大小
    'test_size': 10000, # 测试集大小
    'channel': 1, # 图片通道数
    'image_height': 28, # 图片高度
    'image_width': 28, # 图片宽度
    'batch_size': 64,
    'num_classes': 10, # 分类类别
    'lr': 0.001, # 学习率
    'epoch_size': 20, # 训练次数
    'data_dir_train': os.path.join('fashion-mnist', 'train'),
    'data_dir_test': os.path.join('fashion-mnist', 'test'),
})
```

2.4.3 数据读取和预处理

步骤 1 定义函数用于读取数据

```
def read_image(file_name):
    """
    :param file_name: 文件路径
    :return: 训练或者测试数据
    如下是训练的图片的二进制格式
    [offset] [type]          [value]          [description]
    0000     32 bit integer  0x00000803(2051) magic number
    0004     32 bit integer  60000          number of images
    0008     32 bit integer  28             number of rows
    0012     32 bit integer  28             number of columns
    0016     unsigned byte   ??             pixel
```

```

0017      unsigned byte  ??          pixel
.....
xxxx      unsigned byte  ??          pixel
"""

file_handle = open(file_name, "rb") # 以二进制打开文档
file_content = file_handle.read() # 读取到缓冲区中
head = struct.unpack_from('>IIII', file_content, 0) # 取前 4 个整数，返回一个元组
offset = struct.calcsize('>IIII')
imgNum = head[1] # 图片数
width = head[2] # 宽度
height = head[3] # 高度
bits = imgNum * width * height # data 一共有 60000*28*28 个像素值
bitsString = '>' + str(bits) + 'B' # fmt 格式: '>47040000B'
imgs = struct.unpack_from(bitsString, file_content, offset) # 取 data 数据，返回一个元组
imgs_array = np.array(imgs, np.float32).reshape((imgNum, width * height)) # 最后将读取的数据
reshape 成 【图片数，图片像素】二维数组
return imgs_array

def read_label(file_name):
    """
    :param file_name:
    :return:
    标签的格式如下:
    [offset] [type]          [value]          [description]
    0000      32 bit integer  0x00000801 (2049) magic number (MSB first)
    0004      32 bit integer  60000          number of items
    0008      unsigned byte   ??          label
    0009      unsigned byte   ??          label
    .....
    xxxx      unsigned byte   ??          label
    The labels values are 0 to 9.
    """
    file_handle = open(file_name, "rb") # 以二进制打开文档
    file_content = file_handle.read() # 读取到缓冲区中
    head = struct.unpack_from('>II', file_content, 0) # 取前 2 个整数，返回一个元组
    offset = struct.calcsize('>II')
    labelNum = head[1] # label 数
    bitsString = '>' + str(labelNum) + 'B' # fmt 格式: '>47040000B'
    label = struct.unpack_from(bitsString, file_content, offset) # 取 data 数据，返回一个元组
    return np.array(label, np.int32)

def get_data():
    # 文件获取
    train_image = os.path.join(cfg.data_dir_train, 'train-images-idx3-ubyte')

```

```
test_image = os.path.join(cfg.data_dir_test, "t10k-images-idx3-ubyte")
train_label = os.path.join(cfg.data_dir_train, "train-labels-idx1-ubyte")
test_label = os.path.join(cfg.data_dir_test, "t10k-labels-idx1-ubyte")
# 读取数据
train_x = read_image(train_image)
test_x = read_image(test_image)
train_y = read_label(train_label)
test_y = read_label(test_label)
return train_x, train_y, test_x, test_y
```

步骤 2 数据预处理

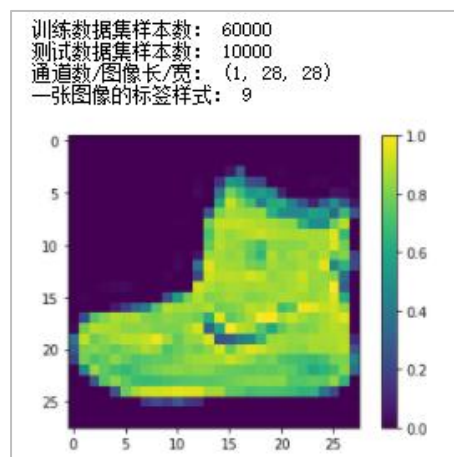
图像最后一个维度，即通道（颜色），使用 `reshape()` 函数将其添加到 `train_images` 和 `test_images` 的维度中。在这种情况下，它是单一颜色，因此通道为 1，即“灰度”。为了减少计算量，还需要把图片的像素值进行归一化，八位图像的像素值范围在 0-255 之间，将所有像素值除以 255，使得像素值范围控制在 0-1 之间。并打印数据集形状和一张图片作为例子。

```
train_x, train_y, test_x, test_y = get_data()
train_x = train_x.reshape(-1, 1, cfg.image_height, cfg.image_width)
test_x = test_x.reshape(-1, 1, cfg.image_height, cfg.image_width)
train_x = train_x / 255.0
test_x = test_x / 255.0

print('训练数据集样本数: ', train_x.shape[0])
print('测试数据集样本数: ', test_y.shape[0])
print('通道数/图像长/宽: ', train_x.shape[1:])
print('一张图像的标签样式: ', train_y[0]) # 一共 10 类，用 0-9 的数字表达类别。

plt.figure()
plt.imshow(train_x[0,0,...])
plt.colorbar()
plt.grid(False)
plt.show()
```

输出结果：



步骤 3 数据集预处理

在训练之前，需要先对数据集中的数据进行“洗牌”，打乱数据集的顺序。

```
# 转换数据类型为 Dataset
def create_dataset():
    XY_train = list(zip(train_x, train_y))
    ds_train = ds.GeneratorDataset(XY_train, ['x', 'y'])
    ds_train = ds_train.shuffle(buffer_size=1000).batch(cfg.batch_size, drop_remainder=True)
    XY_test = list(zip(test_x, test_y))
    ds_test = ds.GeneratorDataset(XY_test, ['x', 'y'])
    ds_test = ds_test.shuffle(buffer_size=1000).batch(cfg.batch_size, drop_remainder=True)
    return ds_train, ds_test
```

2.4.4 训练模型

步骤 1 创建卷积神经网络

该实验中，可以选择使用不含正则化的卷积神经网络或者选择加入正则化的卷积神经网络，用于对比结果。

下面这部分用于创建不加入正则化的卷积神经网络，网络结构为：卷积层 1→卷积层 2→卷积层 3→最大池化层→全连接层 1→全连接层 2。

```
# 定义卷积神经网络，无正则化
class ForwardFashion(nn.Cell):
    def __init__(self, num_class=10): # 一共分十类，图片通道数是 1
        super(ForwardFashion, self).__init__()
        self.num_class = num_class
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, stride=1, padding=0, has_bias=False,
                                pad_mode="valid")
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=0, has_bias=False,
                                pad_mode="valid")
        self.conv3 = nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=0, has_bias=False,
                                pad_mode="valid")
        self.maxpool2d = nn.MaxPool2d(kernel_size=2, stride=2)
        self.relu = nn.ReLU()
        self.flatten = nn.Flatten()
        self.fc1 = nn.Dense(128 * 11 * 11, 128)
        self.fc2 = nn.Dense(128, self.num_class)

    def construct(self, x):
        x = self.conv1(x)
        x = self.relu(x)
        x = self.conv2(x)
        x = self.relu(x)
        x = self.conv3(x)
        x = self.relu(x)
```



```
x = self.maxpool2d(x)
x = self.flatten(x)
x = self.fc1(x)
x = self.relu(x)
x = self.fc2(x)
return x
```

下面这部分用于创建加入正则化的卷积神经网络,网络结构为卷积层 1→dropout 层 1→卷积层 2→dropout 层 1→卷积层 3→dropout 层 1→最大池化层→ dropout 层 2→全连接层 1→ dropout 层 2→全连接层 2:

```
# 定义卷积神经网络,有正则化
class ForwardFashionRegularization(nn.Cell):
    def __init__(self, num_class=10): # 一共分十类,图片通道数是 1
        super(ForwardFashionRegularization, self).__init__()
        self.num_class = num_class
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, stride=1, padding=0, has_bias=False,
                                pad_mode="valid")
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=0, has_bias=False,
                                pad_mode="valid")
        self.conv3 = nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=0, has_bias=False,
                                pad_mode="valid")
        self.maxpool2d = nn.MaxPool2d(kernel_size=2, stride=2)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout()
        self.flatten = nn.Flatten()
        self.fc1 = nn.Dense(3200, 128)
        self.bn = nn.BatchNorm1d(128)
        self.fc2 = nn.Dense(128, self.num_class)

    def construct(self, x):
        x = self.conv1(x)
        x = self.relu(x)
        x = self.conv2(x)
        x = self.relu(x)
        x = self.maxpool2d(x)
        x = self.dropout(x)
        x = self.conv3(x)
        x = self.relu(x)
        x = self.maxpool2d(x)
        x = self.dropout(x)
        x = self.flatten(x)
        x = self.fc1(x)
        x = self.relu(x)
        x = self.bn(x)
        x = self.dropout(x)
        x = self.fc2(x)
        return x
```

步骤 2 启动训练

为这个模型指定优化器（adam）、损失函数（crossentropy）和度量(accuracy)，然后启动训练，最后进行验证。

```
def train(Net):
    ds_train, ds_test = create_dataset()
    # 构建网络
    network = Net(cfg.num_classes)
    # 定义模型的损失函数，优化器
    net_loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction="mean")
    net_opt = nn.Adam(network.trainable_params(), cfg.lr)
    # 训练模型
    model = Model(network, loss_fn=net_loss, optimizer=net_opt, metrics={'acc': Accuracy()})
    loss_cb = LossMonitor()
    print("===== Starting Training =====")
    model.train(30, ds_train, callbacks=[loss_cb], dataset_sink_mode=True)
    # 验证
    metric = model.eval(ds_test)
    print(metric)

    return model
```

训练并验证无正则化的网络。

```
# 训练无正则化的网络
model = train(ForwardFashion)
```

输出结果：

```
===== Starting Training =====
epoch: 1 step: 937, loss is 0.36209568
epoch: 2 step: 937, loss is 0.11113132
epoch: 3 step: 937, loss is 0.107788906
epoch: 4 step: 937, loss is 0.12908919
epoch: 5 step: 937, loss is 0.078461185
epoch: 6 step: 937, loss is 0.18977618
epoch: 7 step: 937, loss is 0.15168177
epoch: 8 step: 937, loss is 0.06739945
epoch: 9 step: 937, loss is 0.14379226
epoch: 10 step: 937, loss is 0.076876596
epoch: 11 step: 937, loss is 0.055951692
epoch: 12 step: 937, loss is 0.022819532
epoch: 13 step: 937, loss is 0.10054826
epoch: 14 step: 937, loss is 0.012528818
epoch: 15 step: 937, loss is 0.0076259132
epoch: 16 step: 937, loss is 0.07877082
epoch: 17 step: 937, loss is 0.031406786
epoch: 18 step: 937, loss is 0.009203883
```

```
epoch: 19 step: 937, loss is 0.005287296
epoch: 20 step: 937, loss is 0.0929834
epoch: 21 step: 937, loss is 0.0015465739
epoch: 22 step: 937, loss is 0.03491202
epoch: 23 step: 937, loss is 0.0005662525
epoch: 24 step: 937, loss is 0.010102608
epoch: 25 step: 937, loss is 0.003999765
epoch: 26 step: 937, loss is 0.011775437
epoch: 27 step: 937, loss is 0.080310896
epoch: 28 step: 937, loss is 0.0018242185
epoch: 29 step: 937, loss is 0.007360851
epoch: 30 step: 937, loss is 0.003999797
{'acc': 0.9147636217948718}
```

训练并验证有正则化的网络。

```
# 训练有正则化的网络
model = train(ForwardFashionRegularization)
```

输出结果：

```
===== Starting Training =====
epoch: 1 step: 937, loss is 0.29856867
epoch: 2 step: 937, loss is 0.28910726
epoch: 3 step: 937, loss is 0.18035105
epoch: 4 step: 937, loss is 0.2785972
epoch: 5 step: 937, loss is 0.21400028
epoch: 6 step: 937, loss is 0.27920294
epoch: 7 step: 937, loss is 0.17452516
epoch: 8 step: 937, loss is 0.309029
epoch: 9 step: 937, loss is 0.30411178
epoch: 10 step: 937, loss is 0.2842149
epoch: 11 step: 937, loss is 0.22666603
epoch: 12 step: 937, loss is 0.16507925
epoch: 13 step: 937, loss is 0.17004505
epoch: 14 step: 937, loss is 0.23396353
epoch: 15 step: 937, loss is 0.20207018
epoch: 16 step: 937, loss is 0.43118417
epoch: 17 step: 937, loss is 0.23762615
epoch: 18 step: 937, loss is 0.24660718
epoch: 19 step: 937, loss is 0.12197974
epoch: 20 step: 937, loss is 0.22719634
epoch: 21 step: 937, loss is 0.2809552
epoch: 22 step: 937, loss is 0.21124852
epoch: 23 step: 937, loss is 0.2100177
epoch: 24 step: 937, loss is 0.29766798
epoch: 25 step: 937, loss is 0.115716025
epoch: 26 step: 937, loss is 0.41360933
epoch: 27 step: 937, loss is 0.11700327
```

```
epoch: 28 step: 937, loss is 0.2552187
epoch: 29 step: 937, loss is 0.14747506
epoch: 30 step: 937, loss is 0.19088028
{'acc': 0.9234775641025641}
```

步骤 3 预测模型

使用上述训练好的模型对测试数据集进行预测。打印预测结果

```
# 预测
ds_test, _ = create_dataset()
test_ = ds_test.create_dict_iterator(output_numpy=True).__next__()
predictions = model.predict(Tensor(test_['x']))
predictions = predictions.asnumpy()
for i in range(15):
    p_np = predictions[i, :]
    p_list = p_np.tolist()
    print('第' + str(i) + '个 sample 预测结果: ', p_list.index(max(p_list)), ' 真实结果: ', test_['y'][i])
```

输出结果：

```
第 0 个 sample 预测结果: 3    真实结果: 3
第 1 个 sample 预测结果: 5    真实结果: 5
第 2 个 sample 预测结果: 6    真实结果: 2
第 3 个 sample 预测结果: 4    真实结果: 4
第 4 个 sample 预测结果: 3    真实结果: 3
第 5 个 sample 预测结果: 3    真实结果: 3
第 6 个 sample 预测结果: 7    真实结果: 7
第 7 个 sample 预测结果: 8    真实结果: 8
第 8 个 sample 预测结果: 3    真实结果: 3
第 9 个 sample 预测结果: 5    真实结果: 5
第 10 个 sample 预测结果: 9    真实结果: 9
第 11 个 sample 预测结果: 2    真实结果: 4
第 12 个 sample 预测结果: 6    真实结果: 6
第 13 个 sample 预测结果: 5    真实结果: 5
第 14 个 sample 预测结果: 0    真实结果: 0
```

步骤 4 可视化结果

定义可视化函数。

```
# -----定义可视化函数-----
# 输入预测结果序列，真实标签序列，以及图片序列
# 目标是根据预测值对错，让其标签显示为红色或者蓝色。对：标签为红色；错：标签为蓝色
def plot_image(predictions_array, true_label, img):
    plt.grid(False)
```

```
plt.xticks([])
plt.yticks([])
# 显示对应图片
plt.imshow(img, cmap=plt.cm.binary)
# 显示预测结果的颜色，如果对上了是蓝色，否则为红色
predicted_label = np.argmax(predictions_array)
if predicted_label == true_label:
    color = 'blue'
else:
    color = 'red'
# 显示对应标签的格式，样式
plt.xlabel('{},{:2.0f}% ({})'.format(class_names[predicted_label],
                                     100 * np.max(predictions_array),
                                     class_names[true_label]), color=color)
# 将预测的结果以柱状图形状显示蓝对红错
def plot_value_array(predictions_array, true_label):
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    this_plot = plt.bar(range(10), predictions_array, color='#777777')
    plt.ylim([0, 1])
    predicted_label = np.argmax(predictions_array)
    this_plot[predicted_label].set_color('red')
    this_plot[true_label].set_color('blue')

import numpy as np
def softmax_np(x):
    x = x - np.max(x)
    exp_x = np.exp(x)
    softmax_x = exp_x/np.sum(exp_x)
    return softmax_x
```

预测结果可视化，输入预测结果序列，真实标签序列，以及图片序列。目标是根据预测值对错，让其标签显示为红色或者蓝色。对：标签为蓝色；错：标签为红色。最后预测 15 个图像与标签，将预测的结果以柱状图形状显示蓝对红错。

```
# 预测 15 个图像与标签，并展现出来
num_rows = 5
num_cols = 3
num_images = num_rows * num_cols
plt.figure(figsize=(2 * 2 * num_cols, 2 * num_rows))
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
for i in range(num_images):
    plt.subplot(num_rows, 2 * num_cols, 2 * i + 1)
    pred_np_ = predictions[i, :]
    pred_np_ = softmax_np(pred_np_)
    plot_image(pred_np_, test_['y'][i], test_['x'][i, 0, ...])
```

```
plt.subplot(num_rows, 2 * num_cols, 2 * i + 2)
plot_value_array(pred_np_, test_['y'][i])
plt.show()
```

输出结果：



2.4.5 观察总结

如果神经网络具有较少的层数和神经元数量，则它可能在训练数据集上表现不佳，即可能导致欠拟合问题。但是，如果神经网络具有很多的层数和神经元数量，它可能在训练数据集上表现得很好（高精度），但是可能在测试数据集上表现不好，即可能导致过拟合问题。

不幸的是，没有确定公式来确定模型的正确结构（层数和每层中的神经元数量），搭建网络时将需要不断尝试，来找到表现不错的架构。

虽然对于神经网络，我们还有其他正则化技术，如 L1, L2 正则化，Dropout 是所有正则化技术中最有效的，也是最常用的。Dropout 会随机地丢弃层的一些输出特征（神经元），即设置为零。假设在应用 Dropout 之前层的输出为[0.5, 0.8, 2.2, 0.9, 0.1]，在应用 Dropout 之后，输出将为[0, 0.8, 2.2, 0, 0.1]。“Dropout 率”是层中被置为零的特征（神经元）的分数。一般来说，Dropout 率保持在 0.2 至 0.5 之间。

请注意，Dropout 只在训练阶段使用。在对测试数据集进行预测时，我们不需要从模型中手动移除 Dropout 层。在测试阶段，不会应用 Dropout。

因此，为了减少本案例中的过拟合问题，让我们应用正则化技术-Dropout 来提升测试表现。

2.5 实验总结

本章提供了一个 FashionMnist 正则化前后对比实验。该实验选取 FashionMnist 灰度数据集将模型进行训练预测，当初始模型表现过拟合时，参数量多的模型的性能反而不如参数量少的模型，我们加入正则化技术重新建立新模型并预测。

3

LeNet 的手写数字识别实验

3.1 实验介绍

LeNet5 + MNIST 被誉为深度学习领域的“Hello world”。本实验主要介绍使用 MindSpore 在 MNIST 数据集上开发和训练一个 LeNet5 模型，并验证模型精度。

- 了解如何使用 MindSpore 进行简单卷积神经网络的开发。
- 了解如何使用 MindSpore 进行简单图片分类任务的训练。
- 了解如何使用 MindSpore 进行简单图片分类任务的验证。

3.2 实验环境要求

- ModelArts 平台：Mindspore

若选择在 ModelArts 平台快速搭建，可参考文末附录：ModelArts 开发环境搭建。

3.3 实验准备

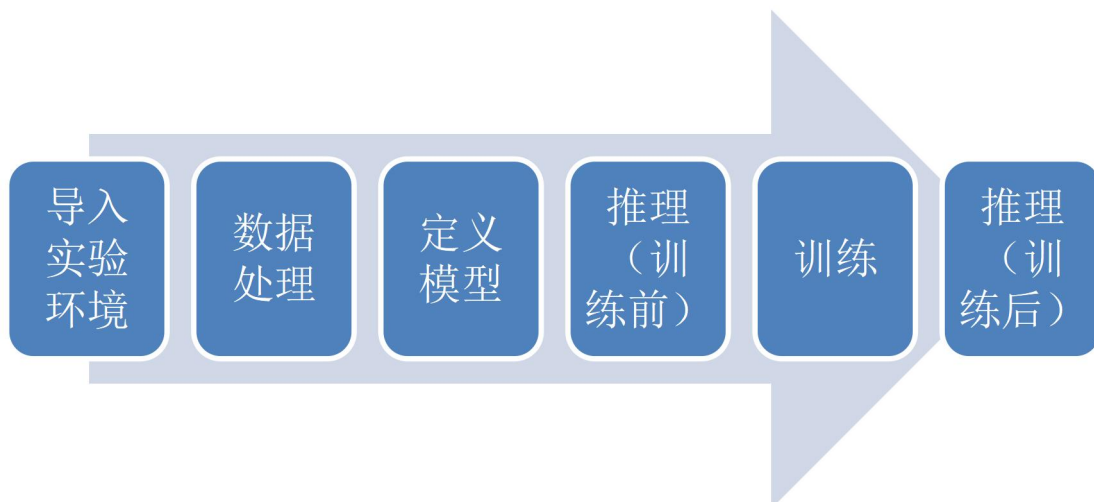
步骤 1 数据集准备

MNIST 是一个手写数字数据集，训练集包含 60000 张手写数字，测试集包含 10000 张手写数字，共 10 类。

- 方法 1：从 MNIST 官网下载数据集到本地，MNIST 数据集的官网 <http://yann.lecun.com/exdb/mnist/>
- 方法 2：从华为云 OBS 中下载 MNIST 数据集并解压。
<https://ascend-professional-construction-dataset.obs.myhuaweicloud.com/deep-learning/MNIST.zip>

本实验采用方法 2 下载数据集。

3.4 实验总体设计



3.5 实验过程

本节将详细介绍实验的设计与实现。

- 导入实验环境；
- 数据处理；
- 定义模型；
- 训练和推理。

3.5.1 导入实验环境

```

import os
# os.environ['DEVICE_ID'] = '0'
import numpy as np
import mindspore as ms
#导入 mindspore 中 context 模块，用于配置当前执行环境，包括执行模式等特性。
import mindspore.context as context
#c_transforms 模块提供常用操作，包括 OneHotOp 和 TypeCast
import mindspore.dataset.transforms.c_transforms as C
#vision.c_transforms 模块是处理图像增强的高性能模块，用于数据增强图像数据改进训练模型。
import mindspore.dataset.vision.c_transforms as CV

from mindspore import nn
from mindspore.train import Model
from mindspore.train.callback import LossMonitor
# 设置 MindSpore 的执行模式和设备
context.set_context(mode=context.GRAPH_MODE, device_target='Ascend') # Ascend, CPU, GPU
  
```

3.5.2 数据处理

步骤 1 获取数据集

```
!wget
https://ascend-professional-construction-dataset.obs.myhuaweicloud.com/deep-learning/MNIST.zip
!unzip MNIST.zip
```

步骤 2 对数据进行预处理。

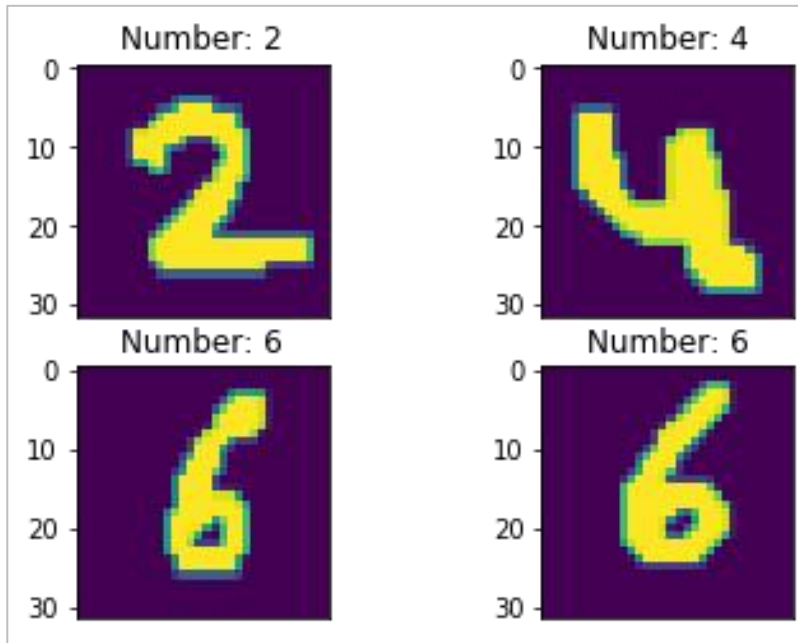
```
def create_dataset(data_dir, training=True, batch_size=32, resize=(32, 32),
                  rescale=1/(255*0.3081), shift=-0.1307/0.3081, buffer_size=64):
    data_train = os.path.join(data_dir, 'train') # 训练集信息
    data_test = os.path.join(data_dir, 'test') # 测试集信息
    ds = ms.dataset.MnistDataset(data_train if training else data_test)
    #将操作中的每个操作应用到此数据集。
    ds = ds.map(input_columns=["image"], operations=[CV.Resize(resize), CV.Rescale(rescale, shift),
    CV.HWC2CHW()]])
    ds = ds.map(input_columns=["label"], operations=C.TypeCast(ms.int32))
    # When `dataset_sink_mode=True` on Ascend, append `ds = ds.repeat(num_epochs)` to the end
    ds = ds.shuffle(buffer_size=buffer_size).batch(batch_size, drop_remainder=True)

    return ds
```

步骤 3 对其中几张图片进行可视化，可以看到图片中的手写数字，图片的大小为 32x32。

```
import matplotlib.pyplot as plt
ds = create_dataset('MNIST', training=False)
data = ds.create_dict_iterator().__next__()
images = data['image'].asnumpy()
labels = data['label'].asnumpy()
#显示前 4 张图片以及对应标签
for i in range(1, 5):
    plt.subplot(2, 2, i)
    plt.imshow(images[i][0])
    plt.title('Number: %s' % labels[i])
    plt.xticks([])
plt.show()
```

输出：



3.5.3 定义模型

MindSpore model_zoo 中提供了多种常见的模型，可以直接使用。LeNet5 模型结构如下图所示：

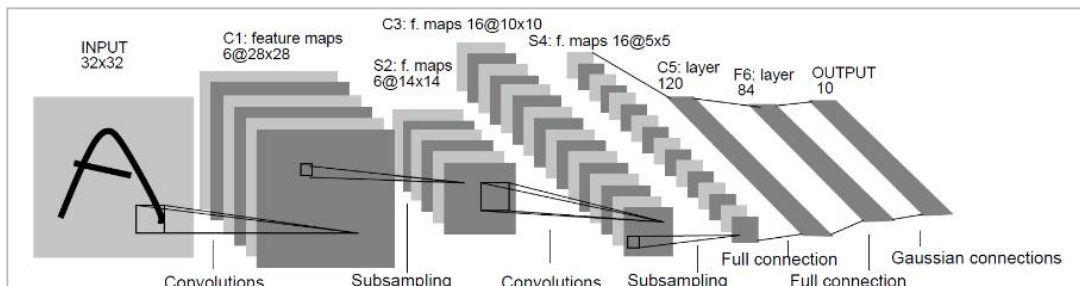


图 3-1 图片来源于 <http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf>

#定义 LeNet5 模型

```
class LeNet5(nn.Cell):
```

```
    def __init__(self):
```

```
        super(LeNet5, self).__init__()
```

```
        #设置卷积网络（输入输出通道数，卷积核尺寸，步长，填充方式）
```

```
        self.conv1 = nn.Conv2d(1, 6, 5, stride=1, pad_mode='valid')
```

```
        self.conv2 = nn.Conv2d(6, 16, 5, stride=1, pad_mode='valid')
```

```
        self.relu = nn.ReLU()
```

```
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
```

```
        self.flatten = nn.Flatten()
```

```
        self.fc1 = nn.Dense(400, 120)
```

```
        self.fc2 = nn.Dense(120, 84)
```

```
        self.fc3 = nn.Dense(84, 10)
```

```
    #构建网络
```

```
    def construct(self, x):
```

```
x = self.relu(self.conv1(x))
x = self.pool(x)
x = self.relu(self.conv2(x))
x = self.pool(x)
x = self.flatten(x)
x = self.fc1(x)
x = self.fc2(x)
x = self.fc3(x)

return x
```

3.5.4 训练

使用 MNIST 数据集对上述定义的 LeNet5 模型进行训练。训练策略如下表所示，可以调整训练策略并查看训练效果，要求验证精度大于 95%。

batch size	number of epochs	learning rate	optimizer
32	3	0.01	Momentum 0.9

表 3-1

```
def train(data_dir, lr=0.01, momentum=0.9, num_epochs=3):
    ds_train = create_dataset(data_dir)
    ds_eval = create_dataset(data_dir, training=False)

    net = LeNet5()
    #计算 softmax 交叉熵。
    loss = nn.loss.SoftmaxCrossEntropyWithLogits(sparse=True, reduction='mean')
    #设置 Momentum 优化器
    opt = nn.Momentum(net.trainable_params(), lr, momentum)
    loss_cb = LossMonitor(per_print_times=ds_train.get_dataset_size())

    model = Model(net, loss, opt, metrics={'acc', 'loss'})
    # dataset_sink_mode can be True when using Ascend
    model.train(num_epochs, ds_train, callbacks=[loss_cb], dataset_sink_mode=True)
    metrics = model.eval(ds_eval, dataset_sink_mode=True)
    print('Metrics:', metrics)

train('MNIST')
```

输出结果：

```
epoch: 1 step 1875, loss is 0.23394052684307098
epoch: 2 step 1875, loss is 0.4737345278263092
epoch: 3 step 1875, loss is 0.07734094560146332
Metrics: {'loss': 0.10531254443608654, 'acc': 0.9701522435897436}
```

3.6 实验总结

本实验展示了如何使用 MindSpore 进行手写数字识别，以及开发和训练 LeNet5 模型。通过对 LeNet5 模型做几代的训练，然后使用训练后的 LeNet5 模型对手写数字进行识别，识别准确率大于 95%。即 LeNet5 学习到了如何进行手写数字识别。

4

附录：ModelArts 开发环境搭建

在华为云 ModelArts 平台上创建 AI 框架为 Mindspore-1.5，硬件环境为 Ascend 910+ARM 的开发环境。

步骤 1 进入华为云 ModelArts 控制台

在[华为云 ModelArts 主页](#)，点击“管理控制台”进入 ModelArts 的管理页面。

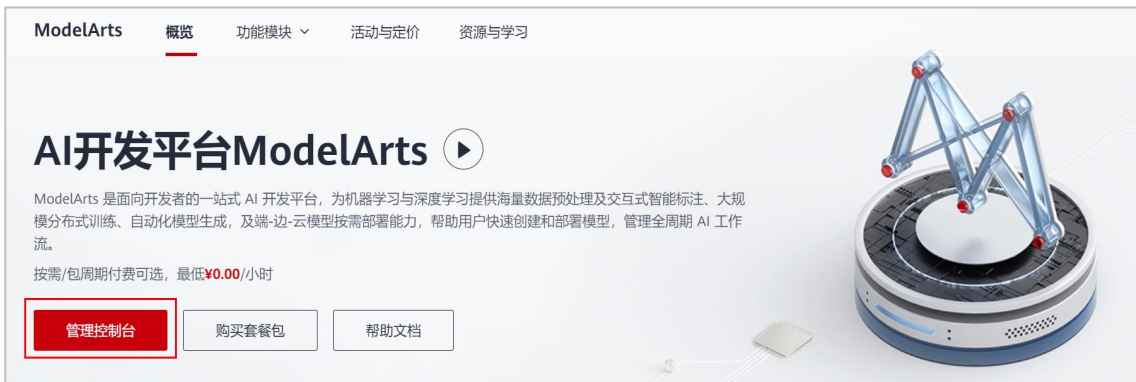


图 4-1 华为云 ModelArts 主页

步骤 2 创建 Notebook 训练作业

控制台区域选择“华北-北京四”，在左侧菜单栏中选择“开发环境”的“Notebook”，点击进入 Notebook 创建页面。

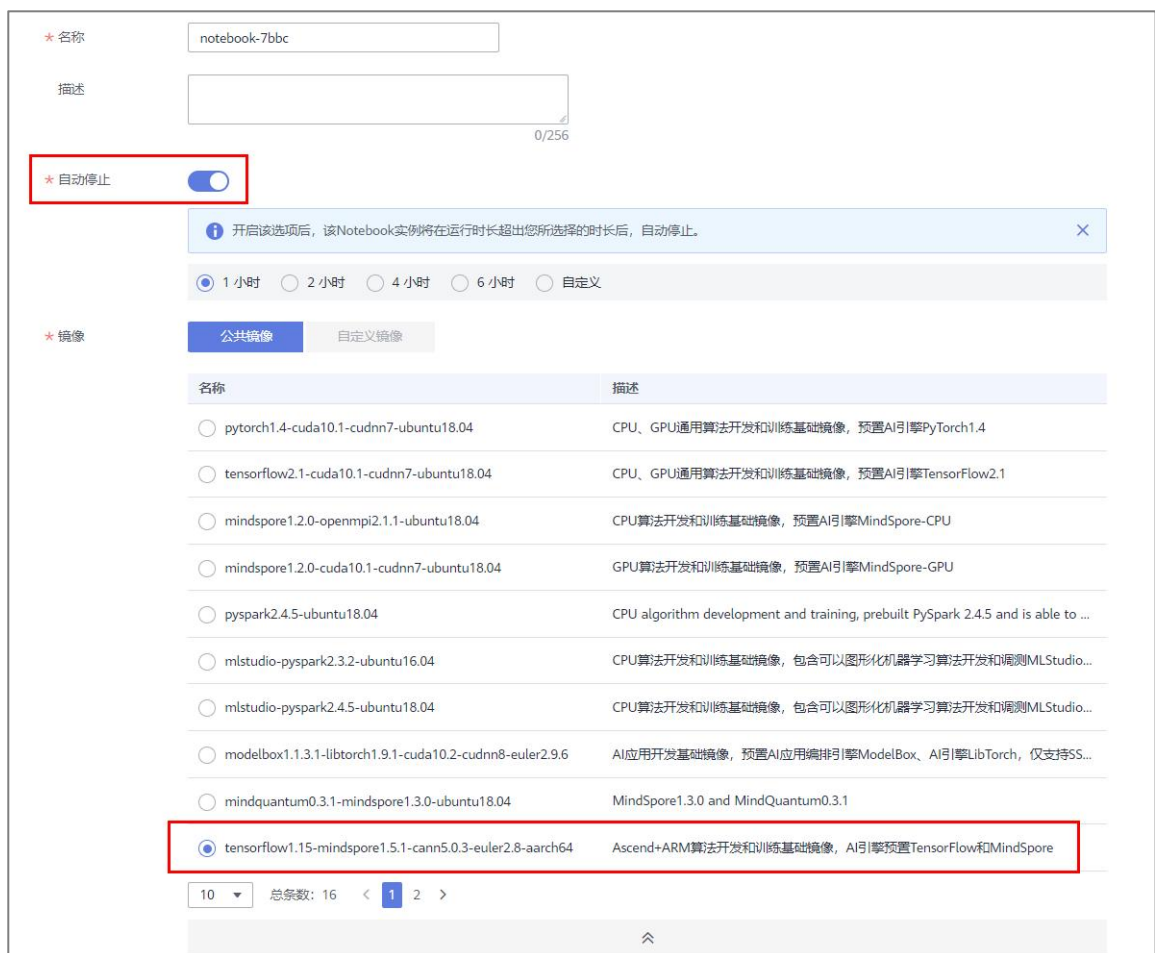


图 4-2 ModelArts 控制台

点击“创建”按钮，创建一个新的 Notebook，其配置如下：

- 名称：自定义。
- 自动停止：建议 1 小时。
- 镜像：Ascend+ARM 算法开发和训练基础镜像。
- 资源池：公共资源池。
- 类型：ASCEND。
- 规格：Ascend: 1*Ascend910|CPU: 24 核 96GB。
- 存储配置：默认存储（50GB），亦可选择 EVS，支持自定义存储规格且为专属资源。

如图所示：



* 名称

描述 0/256

* 自动停止 ☒

开启该选项后，该Notebook实例将在运行时长超出您所选择的时长后，自动停止。

☒ 1 小时 ☐ 2 小时 ☐ 4 小时 ☐ 6 小时 ☐ 自定义

* 镜像 ☒ 公共镜像 ☐ 自定义镜像

名称	描述
<input type="radio"/> pytorch1.4-cuda10.1-cudnn7-ubuntu18.04	CPU、GPU通用算法开发和训练基础镜像，预置AI引擎PyTorch1.4
<input type="radio"/> tensorflow2.1-cuda10.1-cudnn7-ubuntu18.04	CPU、GPU通用算法开发和训练基础镜像，预置AI引擎TensorFlow2.1
<input type="radio"/> mindspore1.2.0-openmpi2.1.1-ubuntu18.04	CPU算法开发和训练基础镜像，预置AI引擎MindSpore-CPU
<input type="radio"/> mindspore1.2.0-cuda10.1-cudnn7-ubuntu18.04	GPU算法开发和训练基础镜像，预置AI引擎MindSpore-GPU
<input type="radio"/> pyspark2.4.5-ubuntu18.04	CPU algorithm development and training, prebuilt PySpark 2.4.5 and is able to ...
<input type="radio"/> mlstudio-pyspark2.3.2-ubuntu16.04	CPU算法开发和训练基础镜像，包含可以图形化机器学习算法开发和预测MLStudio...
<input type="radio"/> mlstudio-pyspark2.4.5-ubuntu18.04	CPU算法开发和训练基础镜像，包含可以图形化机器学习算法开发和预测MLStudio...
<input type="radio"/> modelbox1.1.3.1-libtorch1.9.1-cuda10.2-cudnn8-euler2.9.6	AI应用开发基础镜像，预置AI应用编排引擎ModelBox、AI引擎LibTorch，仅支持SS...
<input type="radio"/> mindquantum0.3.1-mindspore1.3.0-ubuntu18.04	MindSpore1.3.0 and MindQuantum0.3.1
<input checked="" type="radio"/> tensorflow1.15-mindspore1.5.1-cann5.0.3-euler2.8-aarch64	Ascend+ARM算法开发和训练基础镜像，AI引擎预置TensorFlow和MindSpore

10 总条数: 16 < 1 2 >



图 4-3 Notebook 创建配置

配置完成之后“立即创建”，规格确认无误之后“提交”。

步骤 3 启动 Notebook 进入开发环境

当上一步创建好 Notebook 状态显示为“运行中”时，在右侧“操作”栏中“打开”，即可进入在线编程页面。

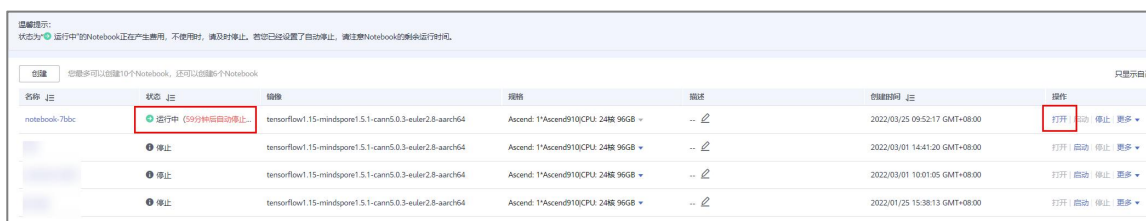


图 4-4 打开 Notebook 环境

可以在此页面创建或编辑 MindSpore 的项目，如图所示：

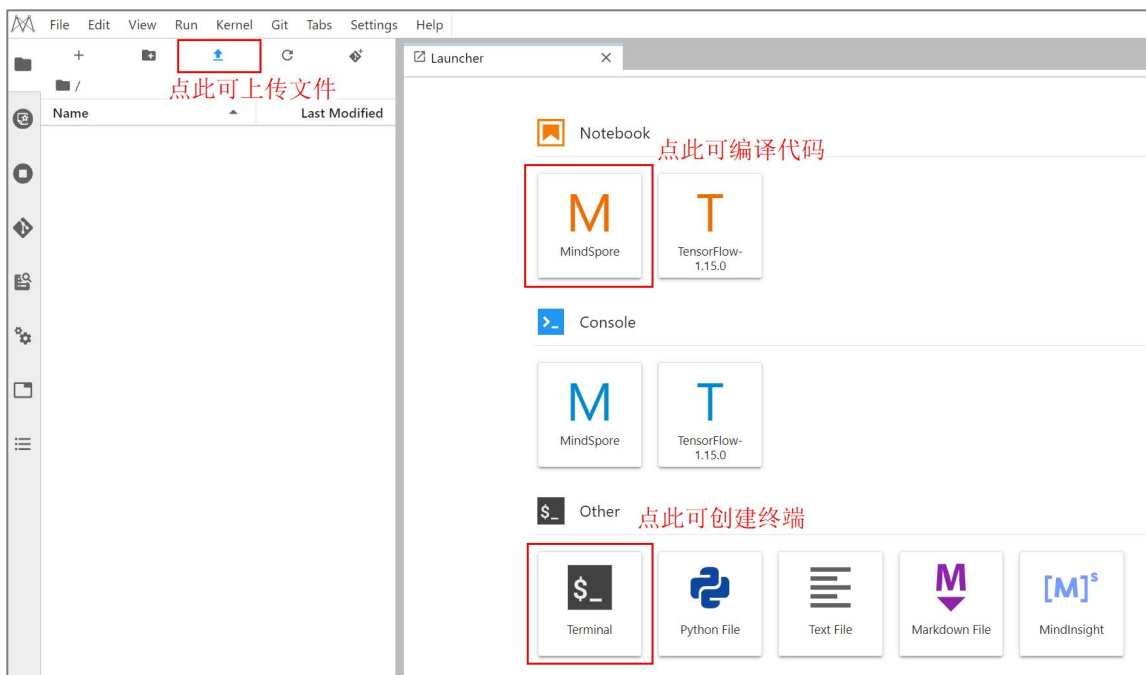


图 4-5 Notebook 开发页面

*注意：Notebook 环境内上传、创建和编辑的文件均在/home/ma-user/work 目录下。

步骤 4 停止 Notebook 训练作业

实验完成之后，请及时关闭 Notebook 训练作业，避免产生不必要的资源浪费。

登录[华为云 ModelArts 控制台](#)，在“操作”栏选择“停止”操作。

如下图所示：



图 4-6 及时停止 Notebook

至此训练用的线上 Notebook 环境搭建完成。