实验四：图像分类与恢复实验

刘昱杉 2024214103

2024 年 12 月 23 日

注：本实验报告为单人独立完成

关于本实验报告对应的源码及实验环境，详见 `code` 目录下 `readme`

FashionMnist 图像分类实验

实验原理

本实验旨在利用 MindSpore 框架构建前馈神经网络，对 Fashion-MNIST 数据集进行分类任务。

Fashion-MNIST 数据集是一个包含 10 个类别的图像数据集，每个类别包含 6000 张训练图像和 1000 张测试图像，图像大小为 28×28 像素。数据集中的 10 个类别分别为：T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot。

实验步骤

首先对数据集下载并预处理，并标定每张图片的数据格式：通道数，图像长，宽，标签

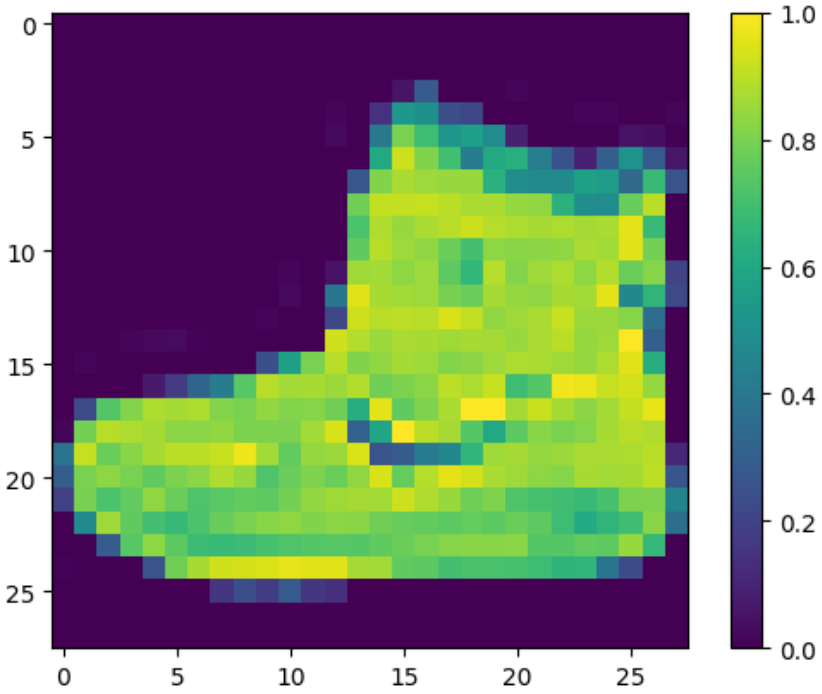

图 1: FashionMnist 数据可视化

然后构建前馈神经网络，包含两个隐藏层，每个隐藏层包含 128 个神经元，激活函数为 ReLU，输出层包含 10 个神经元，激活函数为 Softmax。

Listing 1: 前馈神经网络模型

```
class Forward_fashion(nn.Cell):
    def __init__(self, num_classes=10, num_channel=1):
        super(Forward_fashion, self).__init__()
        self.conv1 = nn.Conv2d(num_channel, 6, 5, pad_mode='same')
        self.conv2 = nn.Conv2d(6, 16, 5, pad_mode='valid')
        self.relu = nn.ReLU()
        self.max_pool2d = nn.MaxPool2d(kernel_size=2, stride=2)
        self.flatten = nn.Flatten()
        self.fc1 = nn.Dense(16 * 5 * 5, 120, weight_init=Normal
            (0.02))
        self.fc2 = nn.Dense(120, 84, weight_init=Normal(0.02))
        self.fc3 = nn.Dense(84, num_classes, weight_init=Normal
            (0.02))
```

模型训练步骤如下：

- 损失函数：Softmax 交叉熵计算多类别分类损失
- 优化器：Adam 优化器，自适应学习率调整
- 训练配置：使用回调函数记录损失值和保存检查点

Listing 2: 分类网络训练

```
net_loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction="
    mean")
net_opt = nn.Adam(network.trainable_params(), learning_rate=cfg.lr)
model = Model(network, loss_fn=net_loss, optimizer=net_opt, metrics
    ={"acc"})
loss_cb = LossMonitor(per_print_times=int(cfg.train_size / cfg.
    batch_size))
config_ck = CheckpointConfig(save_checkpoint_steps=cfg.
    save_checkpoint_steps,
                                 keep_checkpoint_max=cfg.
                                     keep_checkpoint_max)
ckpoint_cb = ModelCheckpoint(prefix=cfg.output_prefix, directory=cfg
    .output_directory, config=config_ck)
print("============== Starting Training ==============")
model.train(cfg.epoch_size, ds_train, callbacks=[ckpoint_cb, loss_cb
    ], dataset_sink_mode=False)
```

实验结果

通过 30 轮训练，准确率不断提升，训练集准确率约 95.7%，随机选 15 张图片进行 label 预测，结果均正确。

图 2: FashionMnist 分类结果

模型性能分析如下：

- 优点：简单的前馈网络能够快速收敛，适用于中等规模数据集；网络参数适中，计算量较低。
- 缺点：未使用正则化或数据增强，可能存在一定程度的过拟合；网络结构较浅，对于更复杂的图像数据可能表现有限。
- 优化方向：增加正则化项，如 Dropout、L2 正则化；尝试更深的网络结构，如 ResNet、VGG 等。

图像恢复实验

实验原理

图像作为重要的信息载体，在获取、传输、存储过程中常常受到噪声的影响。去除噪声的影响并恢复图像原本的信息，是计算机视觉中的一个重要研究问题。本实验选择基于区域二元线性回归模型的图像去噪方法：

1. **高斯噪声**：为图像添加噪声，模拟真实场景中的图像噪声。
2. **区域回归建模**：对每个噪声像素，基于其邻域内未受污染的像素点进行回归建模，预测噪声点的像素值。
3. **线性回归的图像恢复**：根据采用岭回归模型，对图像中的每个噪声像素进行像素值恢复。

实验步骤

首先生成高斯噪声图像，通过随机生成服从正态分布的噪声矩阵，并将噪声矩阵与原图像叠加，模拟真实场景中的图像噪声。

Listing 3: 生成高斯噪声

```
def add_noise(img, noise_ratio):
    #numpy.random.binomial(n,p,size=None)
    noise_img = np.random.binomial(1, 1 - noise_ratio, size=img.
       shape)
    noise_img = np.multiply(noise_img,img)
    return noise_img
```

为原始图片添加了 90% 的高斯噪声，得到如下图像对比：

图 3: 初始图象

图 4: 高斯噪声图象

然后对噪声图像进行区域回归建模，对每个噪声像素点，利用其邻域内的未受污染的像素点进行线性回归建模，预测噪声点的像素值。通过计算恢复程度来结束迭代，并计算最终恢复图像与初始图像的 loss 值。

Listing 4: 复原图像函数

```
def restore(img, domain_length):
    resImg = np.copy(img)
    noiseMask = np.array(img != 0, dtype='double')
    rows, cols, channel = img.shape
    count = 0
    for row in range(rows):
        for col in range(cols):
            for chan in range(channel):
                if noiseMask[row, col, chan] != 0.:
                    continue
                x_train = []
                y_train = []
                for i in range(row-domain_length, row+domain_length)
                    :
                    if i < 0 or i >= row:
                        continue
                    for j in range(col-domain_length, col+
                        domain_length):
                        if j < 0 or j >= col:
                            continue
                        elif noiseMask[i, j, chan] == 0.:
                            continue
                        elif i == row and j == col:
                            continue
                        x_train.append([i, j])
                        y_train.append([img[i, j, chan]])
                if x_train == []:
                    continue
                Regression = LinearRegression()
                Regression.fit(x_train, y_train)
                resImg[row, col, chan] = Regression.predict([[row,
                    col]])
            count += 1
            if count % 5000 == 0:
                print("picture restored:" +
                        str(float(count)/rows/cols))
    print("picture restore finish!")
    return resImg
```

实验结果

通过区域回归建模，对高斯噪声图像进行像素值恢复，得到如下结果，同时 loss 从初始的 1136.72 降低为 716.79。由于人为加入噪声比例很大，恢复图像可以大致看出初始图像的全貌，可以认为恢复算法效果较好。

图 5: 恢复图像

可能的改进

- 批量处理：对多个噪声点同时进行建模与预测，提升计算效率。
- 采用深度学习方法：使用卷积神经网络等深度学习方法，对图像进行去噪处理。
- 区域自适应：根据噪声点分布调整邻域大小，提高恢复效果。

参考文献

[1] T.A. Schonhoff & A.A. Giordano, *Detection and Estimation: Theory and its Applications*. Pearson Education, Inc., 2007.（信号检测与估计——理论与应用，关欣等译，电子工业出版社，2012 年）.

[2] M.D. Srinath, P.K. Rajasekaran & R. Viswanathan, *Introduction to Statistical Signal Processing with Applications*. Prentice Hall, 1996.

[3] Steven M. Kay, *Fundamentals of Statistical Signal Processing, Volume I: Estimation Theory* (©1993) & *Volume II: Detection Theory* (©1998). Pearson Education.（《统计信号处理基础：估计与检测理论（卷 I、卷 II 合集）》，罗鹏飞等译，电子工业出版社，2023 年）.

[4] James V. Candy, *Bayesian Signal Processing: Classical, Modern, and Particle Filtering Methods* (2nd ed.). John Wiley & Sons, Inc., 2016.（宗华等译，哈尔滨工业大学出版社，2023 年）.

[5] Harry L. Van Trees, Kristine L. Bell, with Zhi Tian, *Detection Estimation and Modulation Theory, Part I: Detection, Estimation, and Filtering Theory* (2nd ed.). John Wiley & Sons, Inc., 2013.

[6] ChatGPT by OpenAI (2024). Personal communication and consultation for generating LaTeX formatting, experimental methodology, and model evaluation strategies. OpenAI, `https://www.openai.com`.