**计算机视觉实践-练习2-实验报告**

1. **实验目标**

实现LeNet5在MNIST数据集上的训练和测试，并进行分析，完成实验报告，提交代码。

1. **实验原理**
2. 卷积神经网络基本原理

卷积神经网络主要由卷积层、池化层和全连接层三个部分构成。其中，卷积层是卷积神经网络的核心部分，它通过对输入图像进行卷积操作来提取图像的特征。卷积层的输入通常是一个多通道的（例如多通道图像），每个通道代表一个特征，卷积层的输出也是多通道的，每个通道表示一个不同的特征。

（1）卷积层

卷积操作的实现可以使用滑动窗口的方式，即在输入图像上滑动一个卷积核，将卷积核和输入图像对应位置的像素值相乘并求和，得到输出图像中对应位置的像素值。对于多通道卷积，卷积核的通道数需要与输入数据的通道数相同，每个卷积核的每个通道都会与输入数据的相应通道进行卷积操作。

（2）池化层

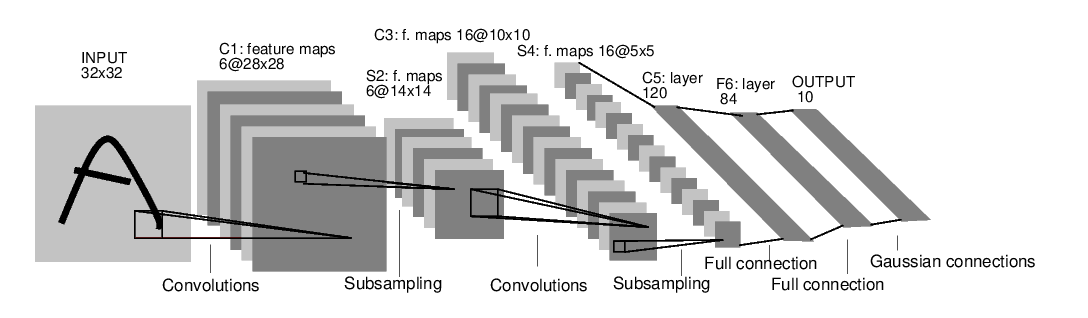
池化层用于降低特征图的空间分辨率，并增强模型对输入图像的平移不变性和鲁棒性。常用的池化方式包括最大池化和平均池化。最大池化的操作是在一个滑动窗口中取最大值作为输出，平均池化的操作是在一个滑动窗口中取平均值作为输出。

（3）全连接层

通常用于将卷积层和池化层提取的特征进行分类或回归。它的输入是一维向量，其输出的维度与任务的分类数或回归值的维度相同。

1. LeNet5的基本结构

LeNet5的基本结构包括7层网络结构（不含输入层），其中包括2个卷积层、2个降采样层（池化层）、2个全连接层和输出层。



**图2.1 LeNet5的结构**

输入层：输入层接收大小为 32×32 的手写数字图像，其中包括灰度值（0-255）。

卷积层C1：卷积层C1包括6个卷积核，每个卷积核的大小为 5×5 ，步长为1，填充为0。因此，每个卷积核会产生一个大小为 28×28 的特征图（输出通道数为6）。

采样层S2：采样层S2采用最大池化（max-pooling）操作，每个窗口的大小为 2×2 ，步长为2。因此，每个池化操作会从4个相邻的特征图中选择最大值，产生一个大小为 14×14 的特征图（输出通道数为6）。这样可以减少特征图的大小，提高计算效率，并且对于轻微的位置变化可以保持一定的不变性。

卷积层C3：卷积层C3包括16个卷积核，每个卷积核的大小为 5×5 ，步长为1，填充为0。因此，每个卷积核会产生一个大小为 10×10 的特征图（输出通道数为16）。

采样层S4：采样层S4采用最大池化操作，每个窗口的大小为 2×2 ，步长为2。因此，每个池化操作会从4个相邻的特征图中选择最大值，产生一个大小为 5×5 的特征图（输出通道数为16）。

全连接层C5：C5将每个大小为 5×5 的特征图拉成一个长度为400的向量，并通过一个带有120个神经元的全连接层进行连接。120是由LeNet-5的设计者根据实验得到的最佳值。

全连接层F6：全连接层F6将120个神经元连接到84个神经元。

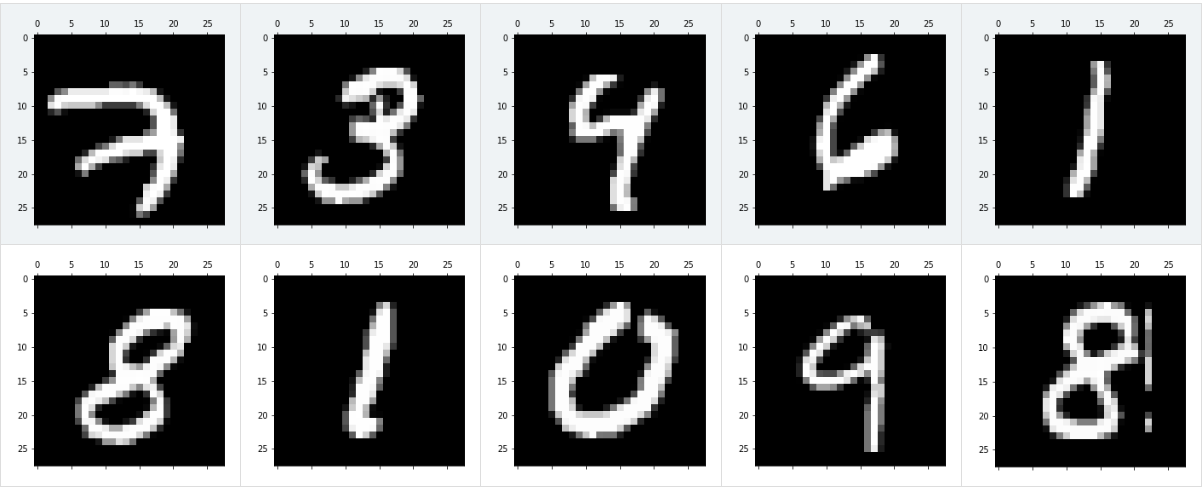
输出层：输出层由10个神经元组成，每个神经元对应0-9中的一个数字，并输出最终的分类结果。在训练过程中，使用交叉熵损失函数计算输出层的误差，并通过反向传播算法更新卷积核和全连接层的权重参数。

1. **实验内容**
2. 实验说明

在本次实验中，采用了PyTorch和MindSpore两种深度学习框架分别对LeNet5模型进行了实现，然后在MNIST数据集上训练。在训练集上进行30轮的训练，PyTorch框架中采取动态学习率调度，MindSpore中采取常数学习率，batchsize的大小256，最后在测试集上评估模型的泛化能力。

1. MNIST数据集

  MNIST数据集一共有7万张图片，其中6万张是训练集，1万张是测试集。每张图片是28 × 28的0 − 9的手写数字图片组成。每个图片是黑底白字的形式，黑底用0表示，白字用0-1之间的浮点数表示，越接近1，颜色越白。



**图3.1 MNIST部分图片**

1. 数据集下载和预处理

在PyTorch，使用torchvision.datasets.MNIST接口下载数据集；在MindSpore中，使用mindvision.dataset.Mnist接口下载数据集。对数据集的图像采取Resize操作，使用默认的插值方法，得到32x32的图像，然后转化为相对应框架的张量。

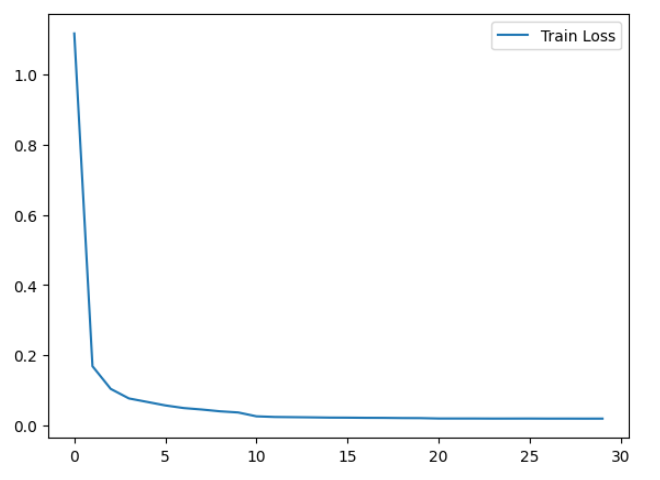
1. 损失函数和优化器

在PyTorch框架中，采用CrossEntropyLoss损失函数和SGD优化器，设置momentum大小为0.9；在MindSpore中，采用SoftmaxCrossEntropyWithLogits损失函数和Momentum优化器，设置momentum大小为0.9。两种损失函数都是基于交叉熵的概念，刻画了两个概率分布之间的距离。当模型输出与真实标签之间的差异越大时，交叉熵的值也会越大。

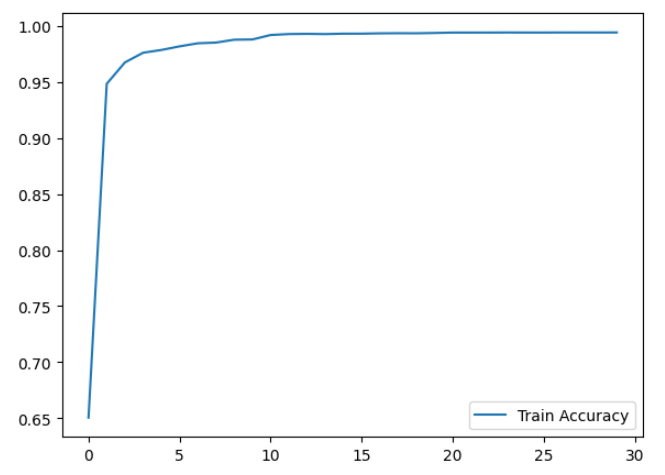
1. 动态学习率

定义学习率调度器，每10个epoch降低学习率到原来的0.1倍。

1. **结果分析**
2. 训练过程的损失和正确率



**图4.1 训练集的损失曲线**



**图4.2 训练集的正确率曲线**

训练采用了动态学习率，由于前期的学习率较大，模型训练的损失快速下降，准确率快速提升，。在这一阶段，模型展现了出色的学习效率和适应性。中期采用了较小的学习率，训练损失缓慢下降并趋于0，正确率缓慢上升并趋于100%，显示出模型在优化过程中的稳定性和准确性；训练的后期，两种曲线都变得平缓，此时说明模型已经收敛，模型参数已经调整到了一个相对最优的状态。

1. 测试集的正确率

在测试集上的进行评估，正确率达到了99.2%。证明了模型对于新数据的泛化能力非常出色，99.2%的正确率意味着模型在绝大多数情况下都能做出正确的预测，这充分展现了模型的高精度和可靠性。

1. **实验总结**

通过本次实验，成功地实现了LeNet-5模型在MNIST数据集上的训练和测试，并取得了优异的结果。实验结果表明，LeNet-5模型在手写数字识别任务中具有较高的精度和可靠性，并且具有良好的泛化能力。

经过此次实验，感受到了PyTorch框架和MindSpore框架的部分差异。对于一个简单的深度神经网络，通过MindSpore提供的model.train接口可以方便地进行网络的训练，LossMonitor可以监控训练过程中loss值的变化，降低了学习门槛。如果使用PyTorch框架，网络训练的主要代码以及训练过程的指标监控都需要我亲手编写。在学习更新方面有所不同，MindSpore的学习率是包含在优化器里面的，每调用一次优化器，学习率更新的step会自动更新一次。PyTorch提供了torch.optim.lr\_scheduler包用于动态修改学习率，使用的时候需要显式地调用optimizer.step()和scheduler.step()来更新学习率。