中南财经政法大学

R 语言期末大作业

姓	名	朱莹
学	号	201721130037
专	业	信息管理与信息系统
班	级	信管 1701

目录

一、卷积神经网络（CNN）模型介绍

（一）基本原理

1. 三个基本层

（1）卷积层（convolutional layer）

对于一张输入图片，将其转化为矩阵，矩阵的元素为对应的像素值，假设有一张 4×4 的图像，设计两个 2×2 的卷积核，运用卷积核后可得到两个 3×3 的特征图。卷积核也称为滤波器（filter）。过程如下图 1 所示：

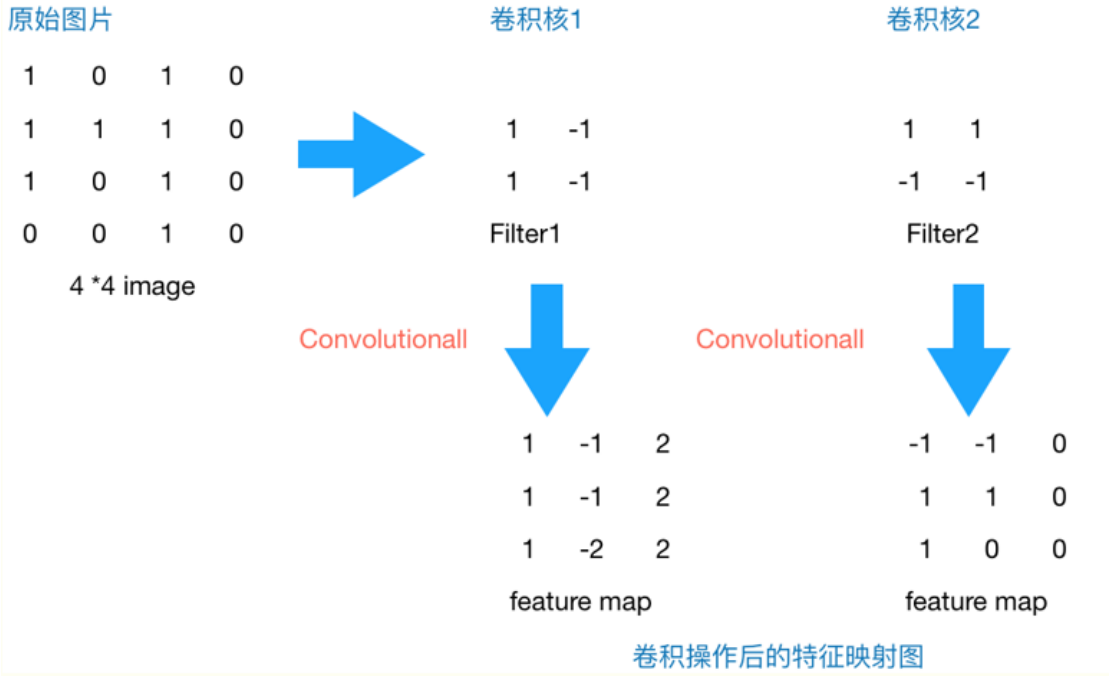

图 1 卷积操作

对于这个 4×4 的图像，采用 2×2 的卷积核来计算。设定步长为 1，即每次以 2×2 的固定窗口往右滑动一个单位。以第一个卷积核 filter1 为例，采用最简单的内积公式，计算过程如下：

$$\text{feature\_map1}(1,1)=1\times1+0\times(-1)+1\times1+1\times(-1)=1$$

$$\text{feature\_map1}(1,2)=0\times1+1\times(-1)+1\times1+1\times(-1)=-1$$

…

$$feature\_map1(3,3)=1\times1+0\times(-1)+1\times1+0\times(-1)=2$$

feature_map1(1,1)表示在通过第一个卷积核计算完后得到的feature_map1 的第一行第一列的值，随着卷积核的窗口不断的滑动，可以计算出一个 3×3 的 feature_map1，这一层卷积操作完成了。

注意到，卷积核移动的步长如果大于 1，则卷积核有可能无法恰好滑到边缘，针对这种情况，可在矩阵最外层补零，可根据需要设定补零的层数。补零层称为 Zero Padding，是一个可以设置的超参数，但要根据卷积核的大小，步幅，输入矩阵的大小进行调整，以使得卷积核恰好滑动到边缘。补一层零后的矩阵如下图 2 所示：

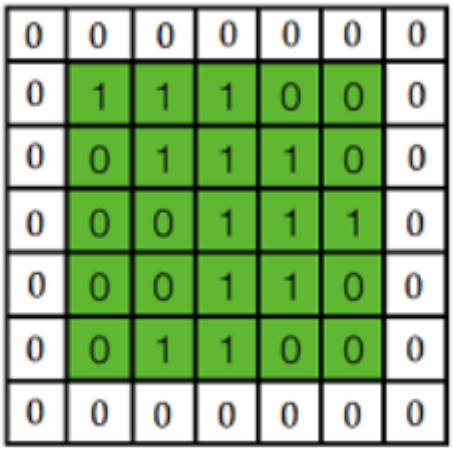

图 2 补零后的矩阵

一般情况下，输入的图片矩阵以及后面的卷积核，特征图矩阵都是方阵，这里设输入矩阵大小为 w，卷积核大小为 k，步幅为 s，补零层为 p，则卷积后产生的特征图大小计算方式为：

$$\omega' = \frac{(w + 2p - k)}{s} - 1$$

通过计算可以发现，同一层的神经元可以共享卷积核，对于高位数据的处理将会变得非常简单。并且使用卷积核后图片的尺寸变小，方便后续计算，且不需要手动去选取特征，只用设计好卷积核的尺寸，数量和滑动的步长就可以让它自己去训练。

使用卷积核计算后分类效果要优于普通的神经网络的原因是通过第一个卷积核计算后的 feature_map 是一个三维数据，在第三列的绝对值最大，说明原始图片上对应的地方有一条垂直方向的特征，即像素数值变化较大;而通过第二个卷积核计算后,第三列的数值为0，第二行的数值绝对值最大，说明原始图片上对应的地方有一条水平方向的特征。即设计的两个卷积核分别能够提取出原始图片的特定的特征。所以可以把卷积核就理解为特征提取器，这样只需要把图片数据输入，设计好卷积核的尺寸、数量和滑动的步长就可以让其自动提取出图片的某些特征，从而达到分类的效果。

（2）池化层（pooling layer）

通过上一层 2×2 的卷积核操作后，将 4×4 的原始图像变成了一张 3×3 的新图像。池化层的主要目的是通过降采样的方式，在不影响图像质量的情况下,压缩图片,减少参数。池化的操作和卷积一样，池化也有一个滑动的核，可以称之为滑动窗口。假设现在设定池化层采用 MaxPooling，大小为 2×2，步长为 1，取每个窗口最大的数值作为输出值，如下图 3 所示：

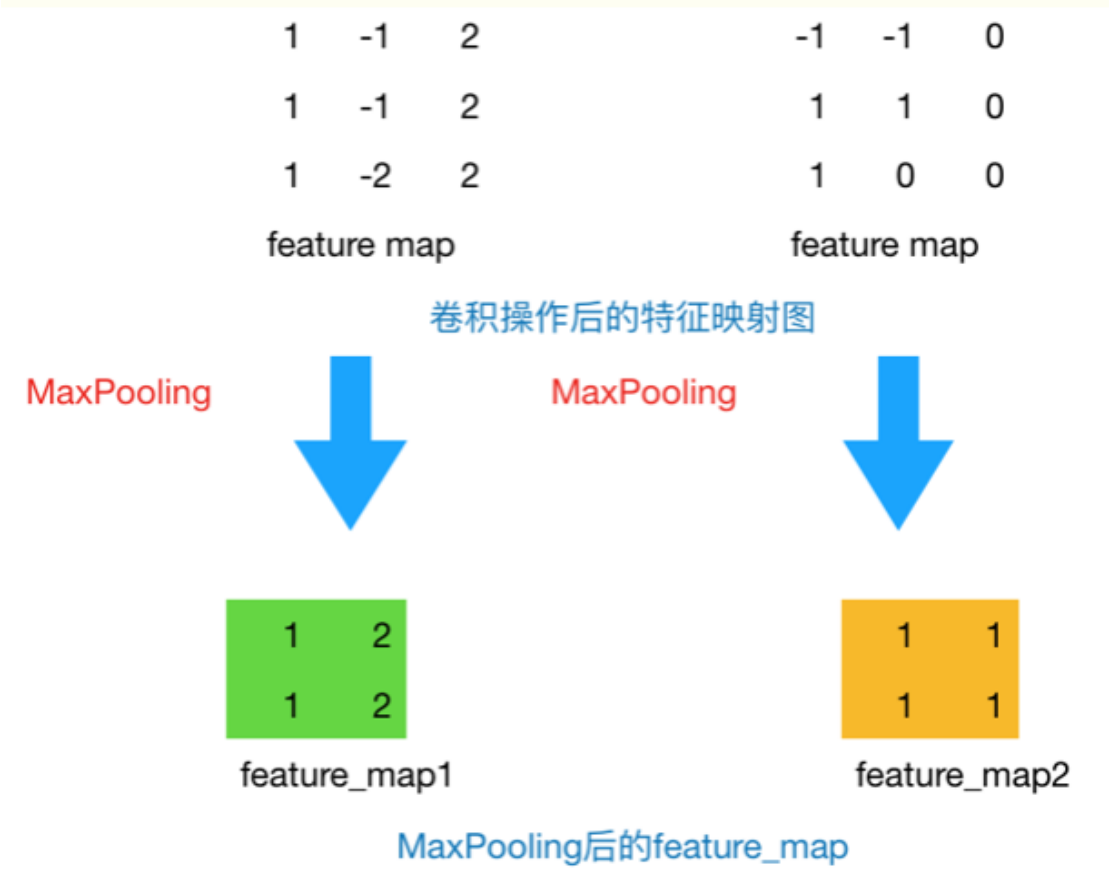

图 3 池化操作

feature_map1 的第一行第一列的数值为 feature map 第一个 2×2 滑动窗口中数值最大的数，这样图片的尺寸就会由 3×3 变成 2×2。

通常来说，池化方法一般有两种：第一种为 maxpooling，取滑动窗口里最大的值；第二种为 averagepooling，取滑动窗口内所有值的平均值。

（3）全连接层（full connected layer）

经过若干层的卷积、池化操作后，将得到的特征图依次按行展开，连接成向量，输入全连接网络中，采用 softmax 对其进行分类。过程如下图 4 所示：

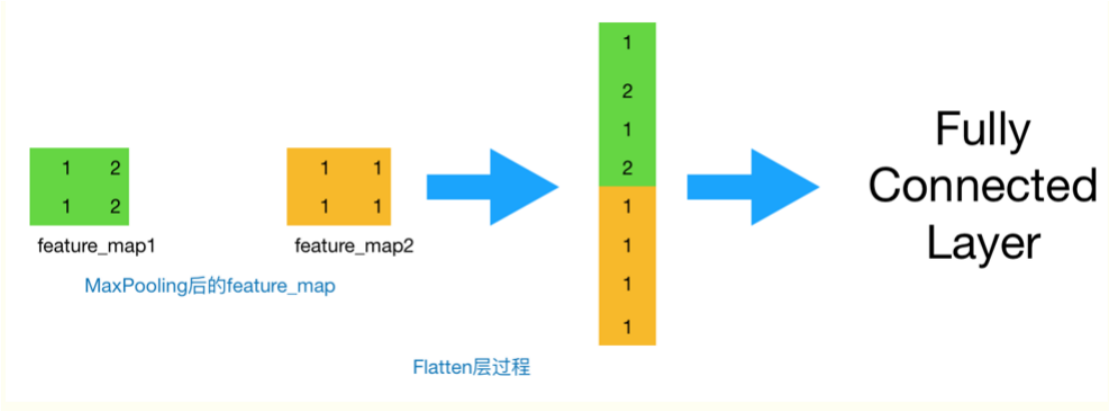

图 4 全连接层

2. 卷积神经网络的前向传播

下图 5 为一个最简单的卷积神经网络，为输入层-卷积层-池化层-全连接层-输出层。

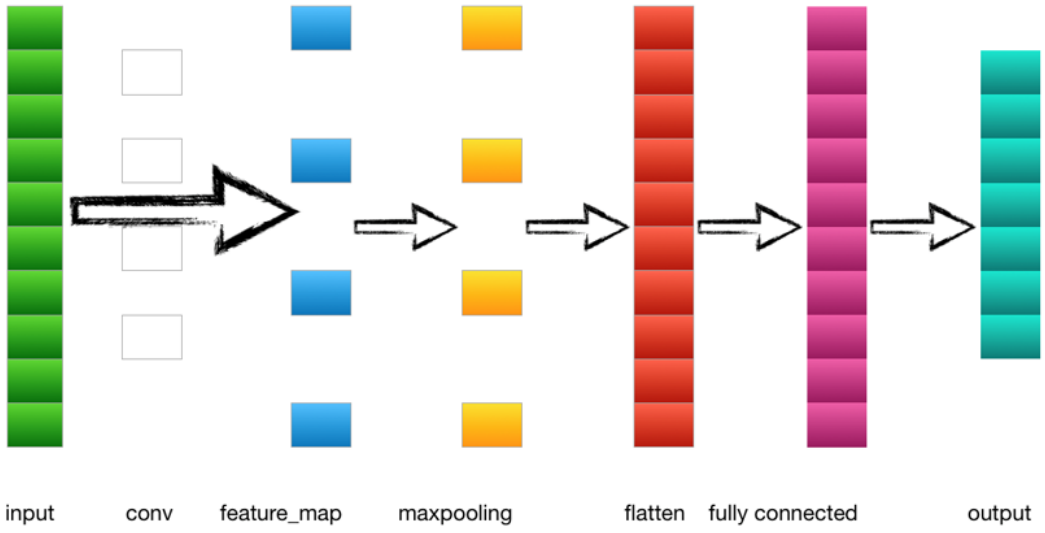

图 5 卷积神经网络结构

（1）输入层-卷积层

上述例子中输入层为 4×4 的图像，经过两个 2×2 的卷积核进行卷积运算后，变成了两个 3×3 的 feature_map。过程如下图 6 所示：

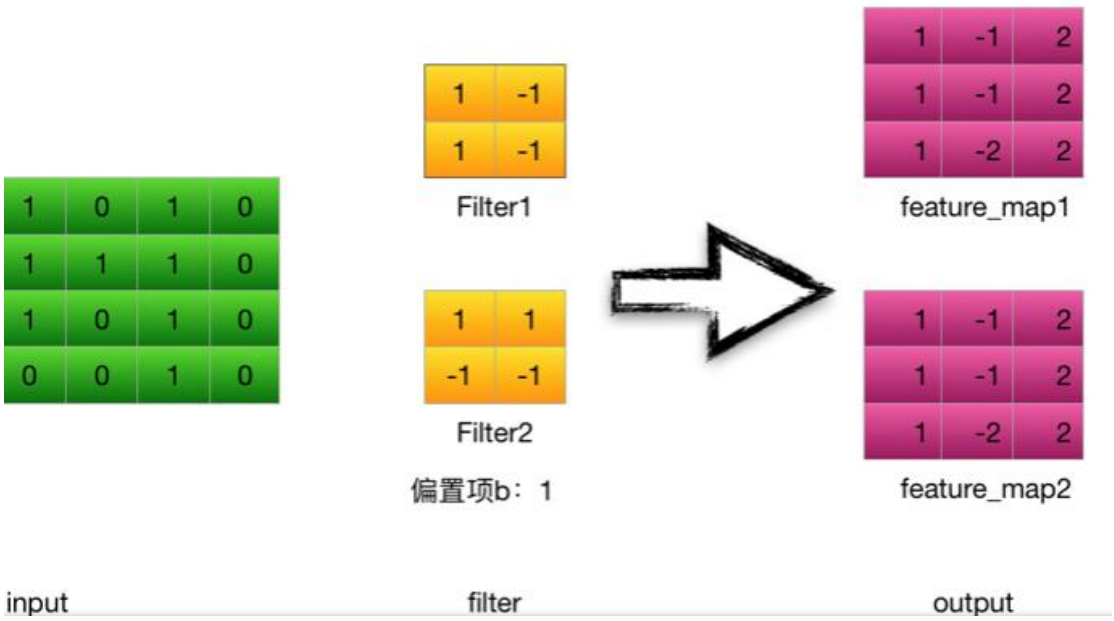

图 6 输入层到卷积层的过程

以卷积核 filter1 为例（stride=1）参数如下图 7 所示：

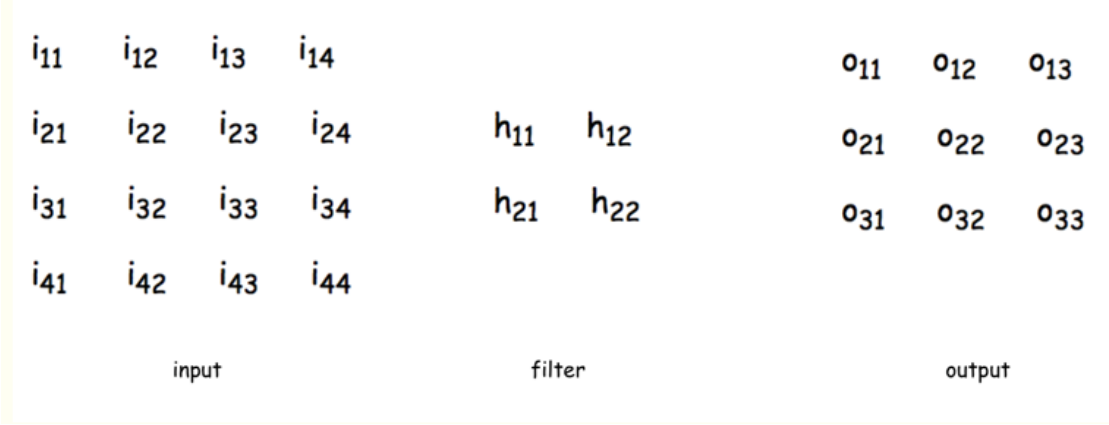

图 7 参数

计算第一个卷积层神经元o_{11}的输入：

$$net_{o_{11}} = conv(input, filter)$$

$$= i_{11} \times h_{11} + i_{12} \times h_{12} + i_{21} \times h_{21} + i_{22} \times h_{22}$$

$$= 1 \times 1 + 0 \times (-1) + 1 \times 1 + 1 \times (-1) = 1$$

神经元o_{11}的输出（使用 relu 激活函数）：

$$out_{o_{11}} = activators(net_{o_{11}}) = max(0, net_{o_{11}}) = 1$$

（2）卷积层-池化层

参数如下图 8 所示：

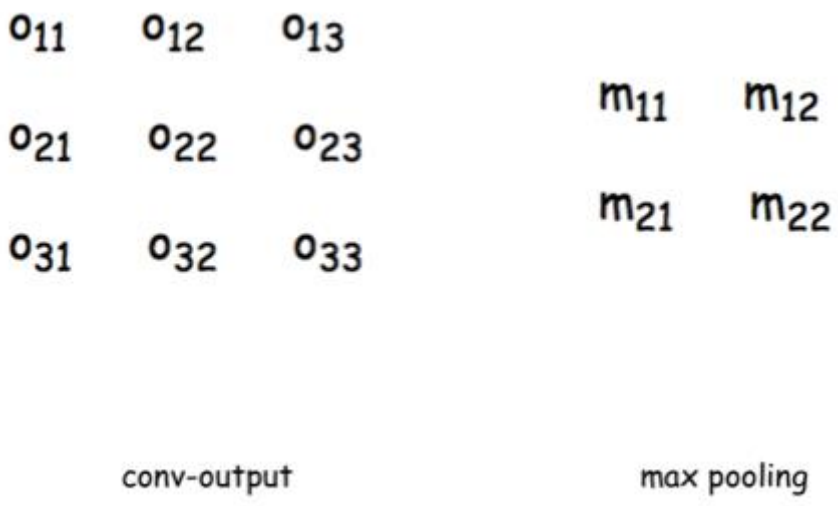

图 8 参数

计算池化层m_{11}的输入（取窗口为 2×2），池化层没有激活函数，计算公式如下：

$$net_{m_{11}} = \max(o_{11}, o_{12}, o_{21}, o_{22}) = 1$$

$$out_{m_{11}} = net_{m_{11}} = 1$$

（3）池化层-全连接层

池化层的输出到 flatten 层把所有元素依次按行展开，连接成向量，然后到全连接层。

（4）全连接层-输出层

全连接层到输出层为正常的神经元与神经元之间的邻接相连，通过 softmax 函数计算后输出到 output，得到不同类别的概率值，输出概率值最大的即为该图片的类别。

3. 卷积神经网络的反向传播

（1）卷积层的反向传播

每一个神经元的值都是上一个神经元的输出作为这个神经元的输入，经过激活函数激活之后输出，作为下一个神经元的输入，在这里用i_{11}表示前一层,o_{11}表示i_{11}的下一层。那$net_{i_{11}}$就是i_{11}这个神经元的输入，$out_{i_{11}}$就是i_{11}这个神经元的输出，同理，$net_{o_{11}}$就是o_{11}这个神经元的输入，$out_{o_{11}}$就是o_{11}这个神经元的输出，因为上一层神经元的输出=下一层神经元的输入，所以$out_{i_{11}} = net_{o_{11}}$。

$$\mathrm{out}_{i_{11}} = \mathrm{activators}(\mathrm{net}_{i_{11}})$$

$$\mathrm{net}_{o_{11}} = \mathrm{conv}(\mathrm{input}, \mathrm{filter})$$

$$= i_{11} \times h_{11} + i_{12} \times h_{12} + i_{21} \times h_{21} + i_{22} \times h_{22}$$

$$\mathrm{out}_{o_{11}} = \mathrm{activators}(\mathrm{net}_{o_{11}}) = \max(0, \mathrm{net}_{o_{11}})$$

$\mathrm{net}_{i_{11}}$表示上一层的输入，$\mathrm{out}_{i_{11}}$表示上一层的输出。

首先计算卷积的上一层的第一个元素i_{11}的误差项δ_{11}：

$$\delta_{11} = \frac{\partial E}{\partial \mathrm{net}_{i_{11}}} = \frac{\partial E}{\partial \mathrm{out}_{i_{11}}} \times \frac{\partial \mathrm{out}_{i_{11}}}{\partial \mathrm{net}_{i_{11}}} = \frac{\partial E}{\partial i_{11}} \times \frac{\partial i_{11}}{\partial \mathrm{net}_{i_{11}}}$$

先把 input 层通过卷积核做完卷积运算后的输出 feature_map 写出来，如下所示：

$$\mathrm{net}_{o_{11}} = i_{11} \times h_{11} + i_{12} \times h_{12} + i_{21} \times h_{21} + i_{22} \times h_{22}$$

$$\mathrm{net}_{o_{12}} = i_{12} \times h_{11} + i_{13} \times h_{12} + i_{22} \times h_{21} + i_{23} \times h_{22}$$

$$\mathrm{net}_{o_{13}} = i_{13} \times h_{11} + i_{14} \times h_{12} + i_{23} \times h_{21} + i_{24} \times h_{22}$$

$$\mathrm{net}_{o_{21}} = i_{21} \times h_{11} + i_{22} \times h_{12} + i_{31} \times h_{21} + i_{32} \times h_{22}$$

$$\mathrm{net}_{o_{22}} = i_{22} \times h_{11} + i_{23} \times h_{12} + i_{32} \times h_{21} + i_{33} \times h_{22}$$

$$\mathrm{net}_{o_{23}} = i_{23} \times h_{11} + i_{24} \times h_{12} + i_{33} \times h_{21} + i_{34} \times h_{22}$$

$$\mathrm{net}_{o_{31}} = i_{31} \times h_{11} + i_{32} \times h_{12} + i_{41} \times h_{21} + i_{42} \times h_{22}$$

$$\mathrm{net}_{o_{32}} = i_{32} \times h_{11} + i_{33} \times h_{12} + i_{42} \times h_{21} + i_{43} \times h_{22}$$

$$\mathrm{net}_{o_{33}} = i_{33} \times h_{11} + i_{34} \times h_{12} + i_{43} \times h_{21} + i_{44} \times h_{22}$$

然后依次对输入元素$i_{i,j}$求偏导

i_{11}的偏导：

$$\frac{\partial E}{\partial i_{11}} = \frac{\partial E}{\partial \mathrm{net}_{o_{11}}} \times \frac{\partial \mathrm{net}_{o_{11}}}{\partial i_{11}} = \delta_{11} \times h_{11}$$

i_{12}的偏导：

$$\frac{\partial E}{\partial i_{12}} = \frac{\partial E}{\partial \mathrm{net}_{o_{11}}} \times \frac{\partial \mathrm{net}_{o_{11}}}{\partial i_{12}} + \frac{\partial E}{\partial \mathrm{net}_{o_{12}}} \times \frac{\partial \mathrm{net}_{o_{12}}}{\partial i_{12}} = \delta_{11} \times h_{12} + \delta_{12} \times h_{11}$$

i_{13}的偏导：

$$\frac{\partial E}{\partial i_{13}} = \frac{\partial E}{\partial \mathrm{net}_{o_{12}}} \times \frac{\partial \mathrm{net}_{o_{12}}}{\partial i_{13}} + \frac{\partial E}{\partial \mathrm{net}_{o_{13}}} \times \frac{\partial \mathrm{net}_{o_{13}}}{\partial i_{13}} = \delta_{12} \times h_{12} + \delta_{13} \times h_{11}$$

i_{21}的偏导：

$$\frac{\partial E}{\partial i_{21}}=\frac{\partial E}{\partial net_{o_{11}}}\times\frac{\partial net_{o_{11}}}{\partial i_{21}}+\frac{\partial E}{\partial net_{o_{21}}}\times\frac{\partial net_{o_{21}}}{\partial i_{21}}=\delta_{11}\times h_{21}+\delta_{21}\times h_{11}$$

i_{22}的偏导：

$$\frac{\partial E}{\partial i_{21}}=\frac{\partial E}{\partial net_{o_{11}}}\times\frac{\partial net_{o_{11}}}{\partial i_{21}}+\frac{\partial E}{\partial net_{o_{21}}}\times\frac{\partial net_{o_{21}}}{\partial i_{21}}=\delta_{11}\times h_{21}+\delta_{21}\times h_{11}$$

观察一下上面几个式子的规律，归纳一下，可以得到如下表达式：

$$\begin{bmatrix}0&0&0&0&0\\0&\delta_{11}&\delta_{12}&\delta_{13}&0\\0&\delta_{21}&\delta_{22}&\delta_{23}&0\\0&\delta_{31}&\delta_{32}&\delta_{33}&0\\0&0&0&0&0\end{bmatrix}\times\begin{bmatrix}h_{22}&h_{21}\\h_{12}&h_{11}\end{bmatrix}=\begin{bmatrix}\frac{\partial E}{\partial i_{11}}&\frac{\partial E}{\partial i_{12}}&\frac{\partial E}{\partial i_{13}}&\frac{\partial E}{\partial i_{14}}\\\frac{\partial E}{\partial i_{21}}&\frac{\partial E}{\partial i_{22}}&\frac{\partial E}{\partial i_{23}}&\frac{\partial E}{\partial i_{24}}\\\frac{\partial E}{\partial i_{31}}&\frac{\partial E}{\partial i_{32}}&\frac{\partial E}{\partial i_{33}}&\frac{\partial E}{\partial i_{34}}\\\frac{\partial E}{\partial i_{41}}&\frac{\partial E}{\partial i_{42}}&\frac{\partial E}{\partial i_{43}}&\frac{\partial E}{\partial i_{44}}\end{bmatrix}$$

卷积核进行了 180°翻转，与这一层的误差敏感项矩阵$delta_{i,j}$周围补零后的矩阵做卷积运算后，就可以得到$\frac{\partial E}{\partial i_{11}}$，即

$$\frac{\partial E}{\partial i_{i,j}}=\sum_m\sum_n h_{m,n}\delta_{i+m,j+n}$$

第一项求完后，我们来求第二项$\frac{\partial i_{11}}{\partial net_{i_{11}}}$

因为$out_{i_{11}}=activators(net_{i_{11}})$

所以$\frac{\partial i_{11}}{\partial net_{i_{11}}}=f'(net_{i_{11}})$

$$\delta_{11}=\frac{\partial E}{\partial net_{i_{11}}}=\frac{\partial E}{\partial i_{11}}\times\frac{\partial i_{11}}{\partial net_{i_{11}}}=\sum_m\sum_n h_{m,n}\delta_{i+m,j+n}\times f'(net_{i_{11}})$$

此时误差敏感矩阵就求完了，得到误差敏感矩阵后，即可求权重的梯度。

上面已经写出了卷积层的输入$net_{o_{11}}$与权重$h_{i,j}$之间的表达式，所以可以直接求出：

$$\frac{\partial \mathrm{E}}{\partial \mathrm{h}_{11}}=\frac{\partial \mathrm{E}}{\partial \mathrm{net}_{\mathrm{o}_{11}}}\times\frac{\partial \mathrm{net}_{\mathrm{o}_{11}}}{\partial \mathrm{h}_{11}}+\cdots+\frac{\partial \mathrm{E}}{\partial \mathrm{net}_{\mathrm{o}_{33}}}\times\frac{\partial \mathrm{net}_{\mathrm{o}_{33}}}{\partial \mathrm{h}_{11}}$$

$$=\delta_{11}\times \mathrm{h}_{11}+\cdots+\delta_{33}\times \mathrm{h}_{11}$$

推论出权重的梯度：

$$\frac{\partial E}{\partial h_{i,\mathrm{j}}}=\sum_{m}\ \sum_{n}\delta_{m,\mathrm{n}}\,\mathrm{out}_{\mathrm{o}_{i+m,\mathrm{j+n}}}$$

偏置项的梯度：

$$\frac{\partial \mathrm{E}}{\partial \mathrm{b}}=\frac{\partial \mathrm{E}}{\partial \mathrm{net}_{\mathrm{o}_{11}}}\times\frac{\partial \mathrm{net}_{\mathrm{o}_{11}}}{\partial \mathrm{w}_{\mathrm{b}}}+\frac{\partial \mathrm{E}}{\partial \mathrm{net}_{\mathrm{o}_{12}}}\times\frac{\partial \mathrm{net}_{\mathrm{o}_{12}}}{\partial \mathrm{w}_{\mathrm{b}}}+\frac{\partial \mathrm{E}}{\partial \mathrm{net}_{\mathrm{o}_{21}}}\times\frac{\partial \mathrm{net}_{\mathrm{o}_{21}}}{\partial \mathrm{w}_{\mathrm{b}}}$$

$$+\frac{\partial \mathrm{E}}{\partial \mathrm{net}_{\mathrm{o}_{22}}}\times\frac{\partial \mathrm{net}_{\mathrm{o}_{22}}}{\partial \mathrm{w}_{\mathrm{b}}}=\delta_{11}+\delta_{12}+\delta_{21}+\delta_{22}$$

$$=\sum_{i}\ \sum_{j}\delta_{i,\mathrm{j}}$$

可以看出，偏置项的偏导等于这一层所有误差敏感项之和。得到了权重和偏置项的梯度后，就可以根据梯度下降法更新权重和梯度了。

（2）池化层的反向传播

如下图 9 所示，左边为上一层的输出，即卷积层的输出 feature_map，右边是池化层的输入。

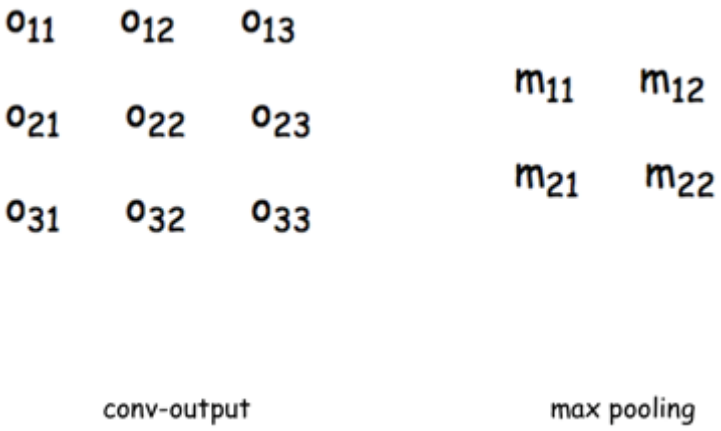

图 9 参数

假设上一层这个滑动窗口的最大值是$\mathrm{out}_{\mathrm{o}_{11}}$。

因为$\mathrm{net}_{m_{11}} = \max(\mathrm{out}_{o_{11}}, \mathrm{out}_{o_{12}}, \mathrm{out}_{o_{21}}, \mathrm{out}_{o_{22}})$

所以$\frac{\partial \mathrm{net}_{m_{11}}}{\partial \mathrm{out}_{o_{11}}} = 1$

$$\frac{\partial \mathrm{net}_{m_{11}}}{\partial \mathrm{out}_{o_{12}}} = \frac{\partial \mathrm{net}_{m_{11}}}{\partial \mathrm{out}_{o_{21}}} = \frac{\partial \mathrm{net}_{m_{11}}}{\partial \mathrm{out}_{o_{22}}} = 0$$

$$\delta_{11}^{l-1} = \frac{\partial E}{\partial \mathrm{out}_{o_{11}}} = \frac{\partial E}{\partial \mathrm{net}_{m_{11}}} \times \frac{\partial \mathrm{net}_{m_{11}}}{\partial \mathrm{out}_{o_{11}}} = \delta_{11}^{l}$$

$$\delta_{11}^{l-1} = \delta_{21}^{l-1} = \delta_{22}^{l-1} = 0$$

这样就求出了池化层的误差敏感项矩阵。同理可以求出每个神经元的梯度并更新权重。

（二）模型在 mnist 数据集上的适用性

卷积神经网络是受生物自然视觉认知机制启发而来的。且在模式分类领域，卷积神经网络避免了对图像的复杂前期预处理，可以直接输入原始图像，在图像分类中得到了广泛的应用。

CNN 是为了识别二维形状而特殊设计的多层感知器，对二维形状的缩放、倾斜或其它形式的变形具有高度不变性。每一个神经元从上一层的局部区域得到输入,这可使神经元提取局部特征。一旦一个特征被提取出来,它相对于其它特征的位置被近似保留下来，而忽略掉精确的位置。每个卷积层后面跟着一个池化，使得特征图的分辨率降低，而有利于降低对二维图形的平移或其他形式的敏感度。

权值共享是 CNN 的一大特点，每一个卷积核滑遍整个输入矩阵，而参数的数目等于卷积核矩阵元素的数目加上一个偏置，相对于 BP 网络大大减少了训练的参数。如果输入的是一张 28×28 的图片，hidden 层有 15 个神经元，输出层有 10 个神经元。使用 BP 网络,参数的数目可达 117625 个(28×28×15×10+15+10)。

二、模型实现

（一）数据预处理的方法

本次训练使用的是mnist数据库的手写数字，该数据集包含60000个示例的训练集以及10000个示例的测试集，图片是28×28的像素矩阵，标签则对应着0-9的10个数字。每张图片都经过了大小归一化和居中处理，该数据集的图片是一张黑白的单通道图片。

该模型是通过PaddlePaddle实现的，在训练时不需要单独下载数据集，调用paddle.dataset.mninst就可以直接下载。

利用paddle.batch获取mnist数据集的reader，batch_size为128，即返回的reader将输入reader的数据（mnist.train和mnist.test打包成指定128大小的批处理数据，即一次训练128张图片。具体如下图10、11所示：

```
train_reader=paddle.batch(mnist.train(),batch_size=128)
test_reader=paddle.batch(mnist.test(),batch_size=128)
```

图 10 读取 mnist 数据集

```
[==================================================]t/train-images-idx3-ubyte.gz not found, downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
[==================================================]t/train-labels-idx1-ubyte.gz not found, downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
[==================================================]t/t10k-images-idx3-ubyte.gz not found, downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
[==================================================]t/t10k-labels-idx1-ubyte.gz not found, downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
```

图 11 下载成功

（二）模型实现的过程

1. 模型运行环境

paddlepaddle1.3.2和python3.7，利用百度AI Studio进行项目环境的搭建。

2. 模型说明

（1）定义卷积神经网络LeNet-5

网络结构为输入层-卷积层-最大池化层-卷积层-最大池化层-输出层，通过调用PaddlePaddle的接口fluid.layers.conv2d()做一次卷积操作，通过num_filters参数设置卷积核的数量，通过filer_size设置卷积核的大小，通过stride来设置卷积操作时移动的步长，act为设置激活函数的类型，这里激活函数为relu。使用fluid.layers.pool2d()接口

做一次池化操作，通过参数 pool_size 可以设置池化的大小，通过参数 pool_stride 设置池化滑动的步长，通过参数 pool_type 设置池化的类型，最大池化为 max，平均池化则可以选择 avg。最后一层为全连接输出层，设定 10 个 label 作为输出，采用 softmax 函数作为分类器，输出每个 label 的概率。Softmax 一般用于多分类问题最后一层输出层的激活函数，作用是对输出归一化。结构如下图 12 所示：

```
#卷积神经网络
def convolutional_neural_network(input):
    #卷积层，卷积核大小为3*3，步长是1，一共有32个卷积核
    conv_1=fluid.layers.conv2d(input=input,num_filters=32,filter_size=3,stride=1,act='relu')
    #池化层，池化核大小为2*2，步长是1，最大池化
    pool_1=fluid.layers.pool2d(input=conv_1,pool_size=2,pool_stride=1,pool_type='max')
    #第二个卷积层，卷积核大小为3*3，步长1，一共有64个卷积核
    conv_2=fluid.layers.conv2d(input=pool_1,num_filters=64,filter_size=3,stride=1,act='relu')
    #第二个池化层，池化核大小是2*2，步长1，最大池化
    pool_2=fluid.layers.pool2d(input=conv_2,pool_size=2,pool_stride=1,pool_type='max')
    #以softmax为激活函数的全连接输出层，大小为label的大小
    #softmax一般用于多分类问题最后一层输出层的激活函数，作用是对输出归一化，这种情况下一般损失函数使用交叉熵
    fc=fluid.layers.fc(input=pool_2,size=10,act='softmax')
    return fc
```

图 12 卷积神经网络定义代码

该实验定义五组不同参数的卷积神经网络：

a.第一层卷积层为卷积核大小为 3×3，步长为 1，有 64 个卷积核，池化层为池化核大小为 2×2，步长为 1，采用最大池化。第二层卷积层为卷积核大小为 3×3，步长为 1，一共有 32 个卷积核，池化层为池化核大小是 2×2，步长为 1，最大池化。

b.第一层卷积层为卷积核大小为 3×3，步长为 1，有 32 个卷积核，池化层为池化核大小为 2×2，步长为 1，采用平均池化。第二层卷积层为卷积核大小为 3×3，步长为 1，一共有 64 个卷积核，池化层为池化核大小是 2×2，步长为 1，平均池化。

c.第一层卷积层为卷积核大小为 3×3，步长为 1，有 32 个卷积核，池化层为池化核大小为 2×2，步长为 1，采用最大池化。第二层卷积层为卷积核大小为 3×3，步长为 1，一共有 64 个卷积核，池化层为池化核大小是 2×2，步长为 1，最大池化。

d.第一层卷积层为卷积核大小为 5×5，步长为 1，有 32 个卷积核，池化层为池化核大小为 2×2，步长为 1，采用最大池化。第二层

卷积层为卷积核大小为 5×5，步长为 1，一共有 64 个卷积核，池化层为池化核大小是 2×2，步长为 1，最大池化。

e.第一层卷积层为卷积核大小为 3×3，步长为 1，有 32 个卷积核，池化层为池化核大小为 2×2，步长为 1，采用最大池化。第二层卷积层为卷积核大小为 3×3，步长为 1，一共有 128 个卷积核，池化层为池化核大小是 2×2，步长为 1，最大池化。

（2）定义输入层

输入的是图像数据。图像为 28×28 的灰度图片，所以输入的形状为[1，28，28]。

```
image=fluid.layers.data(name='image',shape=[1,28,28],dtype='float32')
label=fluid.layers.data(name='label',shape=[1],dtype='int64')
```

图 13 定义输入层

（3）调用定义好的网络来获取分类器

```
result=convolutional_neural_network(image) #使用卷积神经网络
```

图 14 获取分类器

（4）定义损失函数

损失函数使用交叉熵损失函数，因为定义的是一个 batch 上的损失值，所以需要对其求平均值。同时还定义一个准确率函数，可以在训练的时候输出分类的准确率。

```
cost=fluid.layers.cross_entropy(input=result,label=label)#交叉熵
avg_cost=fluid.layers.mean(cost)#整个Batch的平均值
accuracy=fluid.layers.accuracy(input=result,label=label)
```

图 15 定义损失函数

（5）从主程序中克隆一个程序作为预测程序

可以使用该预测程序预测测试的准确率。

```
test_program=fluid.default_main_program().clone(for_test=True)
```

图 16 克隆程序

（6）定义优化方法

使用 adam 优化方法，同时指定学习率为 0.001。

```
optimizer=fluid.optimizer.AdamOptimizer(learning_rate=0.001)
opts=optimizer.minimize(avg_cost)
```

图 17 定义优化方法

（7）定义一个执行器和初始化参数

```
#定义一个使用CPU的执行器
place=fluid.CPUPlace()
exe=fluid.Executor(place)
#初始化参数
exe.run(fluid.default_startup_program())
```

图 18 定义执行器和初始化参数

（8）定义输入数据的维度

输入的数据维度是图像数据和图像对应的标签，每个类别的图像都要对应一个标签这个标签是 0-9 的整型数值。

```
feeder=fluid.DataFeeder(place=place,feed_list=[image,label])
```

图 19 定义输入数据的维度

（9）开始训练和测试

训练 5 个 pass，在训练的时候每 100 个 batch 输出一次当前的准确率。每一个 pass 训练结束之后，进行一次测试，使用测试集进行测试，求出当前的 cost 和准确率的平均值。

```
#开始训练，5个pass
for pass_id in range(5):
    #开始训练
    for batch_id,data in enumerate(train_reader()):
        train_cost,train_acc=exe.run(program=fluid.default_main_program(),feed=feeder.feed(data),fetch_list=[avg_cost,accuracy])
        #每100个batch打印一次信息
        if batch_id%100==0:
            print('Pass:%d Batch:%d Cost:%0.5f Accuracy:%0.5f'%(pass_id,batch_id,train_cost[0],train_acc[0]))

    #每一个pass进行一次测试
    test_accs=[]
    test_costs=[]
    for batch_id,data in enumerate(test_reader()):
        test_cost,test_acc=exe.run(program=test_program,feed=feeder.feed(data),fetch_list=[avg_cost,accuracy])
        test_accs.append(test_acc[0])
        test_costs.append(test_cost[0])
    #求测试结果的平均值
    test_cost=(sum(test_costs)/len(test_costs))
    test_acc=(sum(test_accs)/len(test_accs))
    print('Test:%d Cost:%0.5f Accuracy:%0.5f'%(pass_id,test_cost,test_acc))
```

图 20 训练和测试

（三）模型的结果

在上面实验过程中，定义了五组卷积神经网络的参数。分别进行数据集的训练和测试，将五组实验的参数值和测试准确率汇总为如下表格 1 所示：（实验中卷积层和池化层的步长相同，准确率为最后一次 pass 的测试准确率）

表 21 实验结果

	层数	卷积核尺寸	卷积核个数	池化尺寸	步长	池化类型	准确率
1	一层	3×3	64	2×2	1	Max	97.607%
	二层	3×3	32	2×2	1	Max	
2	一层	3×3	32	2×2	1	Avg	98.438%
	二层	3×3	64	2×2	1	Avg	
3	一层	3×3	32	2×2	1	Max	98.161%
	二层	3×3	64	2×2	1	Max	
4	一层	5×5	32	2×2	1	Max	98.309%
	二层	5×5	64	2×2	1	Max	
5	一层	3×3	32	2×2	1	Max	97.992%
	二层	3×3	128	2×2	1	Max	

实验运行结果如下图 21、22（以第三组参数为例）所示：

```
Pass:0 Batch:0 Cost:3.00562 Accuracy:0.04688
Pass:0 Batch:100 Cost:0.07981 Accuracy:0.97656
Pass:0 Batch:200 Cost:0.09154 Accuracy:0.97656
Pass:0 Batch:300 Cost:0.10297 Accuracy:0.98438
Pass:0 Batch:400 Cost:0.13219 Accuracy:0.95312
Test:0 Cost:0.06258 Accuracy:0.97943
Pass:1 Batch:0 Cost:0.07379 Accuracy:0.98438
Pass:1 Batch:100 Cost:0.06285 Accuracy:0.96875
Pass:1 Batch:200 Cost:0.03919 Accuracy:0.98438
Pass:1 Batch:300 Cost:0.07574 Accuracy:0.98438
Pass:1 Batch:400 Cost:0.04889 Accuracy:0.98438
Test:1 Cost:0.05106 Accuracy:0.98428
Pass:2 Batch:0 Cost:0.05275 Accuracy:0.98438
Pass:2 Batch:100 Cost:0.05466 Accuracy:0.97656
Pass:2 Batch:200 Cost:0.03006 Accuracy:0.98438
Pass:2 Batch:300 Cost:0.05636 Accuracy:0.99219
Pass:2 Batch:400 Cost:0.02572 Accuracy:0.99219
Test:2 Cost:0.05271 Accuracy:0.98200
Pass:3 Batch:0 Cost:0.03674 Accuracy:0.99219
Pass:3 Batch:100 Cost:0.06902 Accuracy:0.97656
Pass:3 Batch:200 Cost:0.00852 Accuracy:1.00000
Pass:3 Batch:300 Cost:0.05509 Accuracy:0.99219
Pass:3 Batch:400 Cost:0.01318 Accuracy:0.99219
Test:3 Cost:0.06703 Accuracy:0.98190
Pass:4 Batch:0 Cost:0.03779 Accuracy:0.98438
Pass:4 Batch:100 Cost:0.00807 Accuracy:1.00000
```

图 21 第三组参数实验结果 1

```
Pass:4 Batch:200 Cost:0.00837 Accuracy:1.00000
Pass:4 Batch:300 Cost:0.01469 Accuracy:0.99219
Pass:4 Batch:400 Cost:0.03630 Accuracy:0.99219
Test:4 Cost:0.06934 Accuracy:0.98161
```

图 22 第三组参数实验结果 2

三、模型分析

（一）正确率分析

设置五组不同参数的卷积神经网络，以第三组作为对照组，其他四组作为实验组。

通过第一组和第三组的比较，发现靠近输入层的卷积层的卷积核个数比后面的卷积层的卷积核个数少时，准确率越高。通常情况下，靠近输入的卷积层，譬如第一层卷积层，会找出一些共性的特征，如手写数字识别中第一层一般是找出诸如"横线"、“竖线”、“斜线”等共性特征，称之为 basic feature，经过 max pooling 后，在第二层卷积层，

可以找出一些相对复杂的特征，如“横折”、“左半圆”、“右半圆”等特征，越往后，卷积核设定的数目越多，越能体现 label 的特征就越细致，就越容易分类出来。

通过第二组和第三组的比较，发现使用平均池化时，准确率却高。可以发现进行最大池化操作后，提取出真正能够识别特征的数值，其余被舍弃的数值，对于提取特定的特征并没有特别大的帮助。在进行后续计算时，减小了 feature map 的尺寸，从而减少参数，达到减小计算量，却不损失效果的情况。但是并不是所有情况最大池化的效果都很好，有时候有些周边信息也会对某个特定特征的识别产生一定效果，那么这个时候舍弃这部分的信息就会出现准确率降低的情况。

通过第四组、第五组和第三组的比较，发现改变卷积核的大小与卷积核的数目会对结果产生一定的影响，其中扩大卷积核的尺寸会提高准确性，增加卷积核数目会降低准确性。

（二）避免过拟合现象

Dropout 是一种正则化手段，通过在训练过程中阻止神经元节点间的相关性来减少过拟合。根据给定的丢弃概率，dropout 操作符按丢弃概率随机将一些神经元输出设置为 0，这些神经元将不对外传递信号。应用 Dropout 之后，会将标了 X 的神经元从网络中删除，让它们不向后面的层传递信号，其他的仍保持不变。在学习过程中，丢弃哪些神经元是随机决定，因此模型不会过度依赖某些神经元，能一定程度上抑制过拟合，是深度学习中一种常用的抑制过拟合的方法。

在第三组实验全连接层前面加上一层 dropout 层，定义输入层-卷积层-池化层-卷积层-池化层-dropout 层-全连接层-输出层的网络结构，具体参数为：第一层卷积层为卷积核大小为 3×3，步长为 1，有 32 个卷积核，池化层为池化核大小为 2×2，步长为 1，采用最大池化。第二层卷积层为卷积核大小为 3×3，步长为 1，一共有 64 个卷积核，池化层为池化核大小是 2×2，步长为 1，最大池化，dropout 层，丢弃概率为 0.5。结构如下图 23 所示：

```
def convolutional_neural_network(input):
    #卷积层，卷积核大小为3*3，步长是1，一共有32个卷积核
    conv_1=fluid.layers.conv2d(input=input,num_filters=32,filter_size=3,stride=1,act='relu')
    #池化层，池化核大小为2*2，步长是1，最大池化
    pool_1=fluid.layers.pool2d(input=conv_1,pool_size=2,pool_stride=1,pool_type='max')
    #第二个卷积层，卷积核大小为3*3，步长1，一共有64个卷积核
    conv_2=fluid.layers.conv2d(input=pool_1,num_filters=64,filter_size=3,stride=1,act='relu')
    #第二个池化层，池化核大小是2*2，步长1，最大池化
    pool_2=fluid.layers.pool2d(input=conv_2,pool_size=2,pool_stride=1,pool_type='max')
    drop = fluid.layers.dropout(pool_2, dropout_prob=0.5)
    #以softmax为激活函数的全连接输出层，大小为label的大小
    #softmax一般用于多分类问题最后一层输出层的激活函数，作用是对输出归一化，这种情况下一般损失函数使用交叉熵
    fc = fluid.layers.fc(input=drop,size=10,act='softmax')


    return fc
```

图 23 加入 dropout 层

实验结果如下图 24、25 所示：

```
Pass:0 Batch:0 Cost:2.89328 Accuracy:0.07812
Pass:0 Batch:100 Cost:0.08544 Accuracy:0.96875
Pass:0 Batch:200 Cost:0.09691 Accuracy:0.98438
Pass:0 Batch:300 Cost:0.09581 Accuracy:0.98438
Pass:0 Batch:400 Cost:0.11403 Accuracy:0.95312
Test:0 Cost:0.06232 Accuracy:0.97854
Pass:1 Batch:0 Cost:0.07459 Accuracy:0.97656
Pass:1 Batch:100 Cost:0.06000 Accuracy:0.98438
Pass:1 Batch:200 Cost:0.05904 Accuracy:0.99219
Pass:1 Batch:300 Cost:0.07876 Accuracy:0.98438
Pass:1 Batch:400 Cost:0.04708 Accuracy:0.98438
Test:1 Cost:0.04332 Accuracy:0.98556
Pass:2 Batch:0 Cost:0.06040 Accuracy:0.98438
Pass:2 Batch:100 Cost:0.06596 Accuracy:0.97656
Pass:2 Batch:200 Cost:0.05313 Accuracy:0.98438
Pass:2 Batch:300 Cost:0.03333 Accuracy:0.98438
Pass:2 Batch:400 Cost:0.03240 Accuracy:0.98438
Test:2 Cost:0.04311 Accuracy:0.98596
Pass:3 Batch:0 Cost:0.02711 Accuracy:0.98438
Pass:3 Batch:100 Cost:0.06092 Accuracy:0.97656
Pass:3 Batch:200 Cost:0.05252 Accuracy:0.97656
Pass:3 Batch:300 Cost:0.04665 Accuracy:0.98438
Pass:3 Batch:400 Cost:0.03715 Accuracy:0.97656
Test:3 Cost:0.04064 Accuracy:0.98705
Pass:4 Batch:0 Cost:0.03149 Accuracy:0.99219
Pass:4 Batch:100 Cost:0.04839 Accuracy:0.99219
Pass:4 Batch:200 Cost:0.02922 Accuracy:0.99219
```

图 24 带 dropout 层的实验结果 1

```
Pass:4 Batch:300 Cost:0.00582 Accuracy:1.00000
Pass:4 Batch:400 Cost:0.08831 Accuracy:0.96875
Test:4 Cost:0.04273 Accuracy:0.98774
```

图 25 带 dropout 层的实验结果 2

可以发现加入了 dropout 层的准确率高达 98.774%，比不加 dropout 层的同参数的模型的准确率上升了 0.6%。

四、程序源代码

```
import numpy as np
import paddle as paddle
import paddle.dataset.mnist as mnist
import paddle.fluid as fluid
from PIL import Image
import matplotlib.pyplot as plt

def convolutional_neural_network(input):
    #卷积层，卷积核大小为 3×3，步长是 1，一共有 32 个卷积核
    conv_1=fluid.layers.conv2d(input=input,num_filters=32,filter_size=3,stride=1,act='relu')
    #池化层，池化核大小为 2×2，步长是 1，最大池化
    pool_1=fluid.layers.pool2d(input=conv_1,pool_size=2,pool_stride=1,pool_type='max')
    #第二个卷积层，卷积核大小为 3×3，步长 1，一共有 64 个卷积核
    conv_2=fluid.layers.conv2d(input=pool_1,num_filters=64,filter_size=3,stride=1,act='relu')
    #第二个池化层，池化核大小是 2×2，步长 1，最大池化
    pool_2=fluid.layers.pool2d(input=conv_2,pool_size=2,pool_stride=1,pool_type='max')
    #加入 dropout 层
    drop = fluid.layers.dropout(pool_2, dropout_prob=0.5)
    #以 softmax 为激活函数的全连接输出层，大小为 label 的大小
    #softmax 一般用于多分类问题最后一层输出层的激活函数，作用是对输出归一化，这种情况下一般损失函数使用交叉熵
    fc = fluid.layers.fc(input=drop,size=10,act='softmax')
    return fc
```

```
train_reader=paddle.batch(mnist.train(),batch_size=128)
test_reader=paddle.batch(mnist.test(),batch_size=128)

#定义占位输入层和标签层
image=fluid.layers.data(name='image',shape=[1,28,28],dtype='float32')
label=fluid.layers.data(name='label',shape=[1],dtype='int64')

#获取前向传播网络结果
result=convolutional_neural_network(image) #使用卷积神经网络

#定义损失函数和准确率函数
cost=fluid.layers.cross_entropy(input=result,label=label)#交叉熵
avg_cost=fluid.layers.mean(cost)#整个 Batch 的平均值
accuracy=fluid.layers.accuracy(input=result,label=label)

#克隆主程序，获得一个预测程序
test_program=fluid.default_main_program().clone(for_test=True)

#定义优化器，使用 Adam 优化器
optimizer=fluid.optimizer.AdamOptimizer(learning_rate=0.001)
opts=optimizer.minimize(avg_cost)

#定义一个使用 CPU 的执行器
place=fluid.CPUPlace()
exe=fluid.Executor(place)
#初始化参数
exe.run(fluid.default_startup_program())

#定义输入数据维度
feeder=fluid.DataFeeder(place=place,feed_list=[image,label])
```

```
#开始训练，5 个 pass
for pass_id in range(5):
    #开始训练
    for batch_id,data in enumerate(train_reader()):
        train_cost,train_acc=exe.run(program=fluid.default_main_prog
ram(),feed=feeder.feed(data),fetch_list=[avg_cost,accuracy])
        #每 100 个 batch 打印一次信息
        if batch_id%100==0:
            print('Pass:%d              Batch:%d              Cost:%0.5f
Accuracy:%0.5f'%(pass_id,batch_id,train_cost[0],train_acc[0]))

    #每一个 pass 进行一次测试
    test_accs=[]
    test_costs=[]
    for batch_id,data in enumerate(test_reader()):
        test_cost,test_acc=exe.run(program=test_program,feed=feeder.
feed(data),fetch_list=[avg_cost,accuracy])
        test_accs.append(test_acc[0])
        test_costs.append(test_cost[0])
    #求测试结果的平均值
    test_cost=(sum(test_costs)/len(test_costs))
    test_acc=(sum(test_accs)/len(test_accs))
    print('Test:%d                                    Cost:%0.5f
Accuracy:%0.5f'%(pass_id,test_cost,test_acc))
```