基于BP神经网络车牌识别

汇报内容

一 BP神经网络

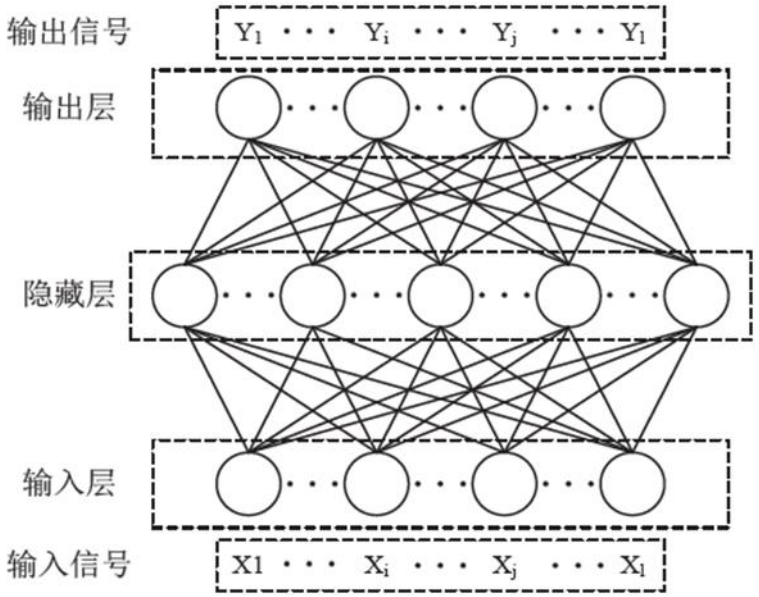
二 车牌识别系统的方案设计

三 实验结果

汇报人：焦昕远

BP神经网络基本结构

BP神经网络的全称是前馈型神经网络，其英文全称是Back Propagation，因为在广泛采用以及在文献中也有被多次提及为BP神经网络，所以本系统设计中也将前馈型神经网络简称为BP神经网络。



BP神经网络基本结构

BP网络模型采用了一种误差后向传递算法。当前，大部分的神经网络模型都是建立在 BP 神经网络及其变形基础上，而 BP神经网络是前馈网络中最重要的一环。可以看出BP神经网络在人工智能领域，特别实在人工神经网络中的重要地位。与感知、线性神经网络等传统的神经网络相比， BP网络中的神经元多以 Sigmoid函数为传输函数，可以在多个输入、输出间建立非线性关系。BP神经网路由输入层、隐含层及输出层三个层次组成，其中每一层的神经元都是完整的。相应地，所述隐藏层中的每个神经元的输出数值为

$$h_j = f\left[\sum_{i=0}^{N-1} v_{ij}x_i - \psi_j\right]$$

$$y_k = f\left[\sum_{j=0}^{L-1} w_{jk}h_j - \theta_k\right]$$

BP神经网络工作原理

BP神经网络主要的原理是对权重和偏差进行调节，以减小网络的实际输出和预期输出的偏差[5]。BP神经网络遵循如下的程序步骤：

1.正向传递：利用神经网路将输入样本传递至各结点，以求出各结点之输入与活化数值。

2.算法错误：通过对系统中各结点的真实值与预期值的偏差，求出各结点的偏差值的平方和。

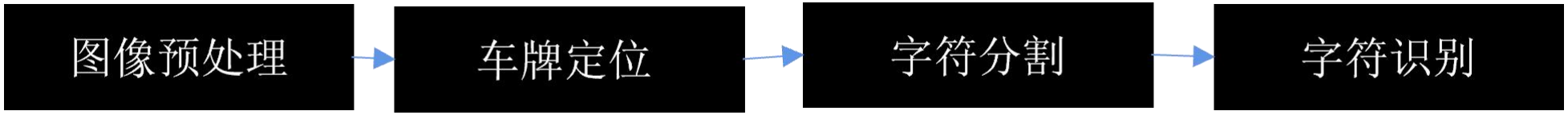
3.后传递错误：针对错误源，采用连锁规则，将错误传递到后端，并对各结点进行权重与偏差的修正。

4.权重与偏差修正：基于错误逆向传递的成果，采用梯度下降等方法对神经网络进行权重与偏差修正。

5.反复进行：反复进行第1~4步，直至网路的输出错误满足预先设定的准确度需求或设定的学习数目限制。该方法通过对神经网络的权重进行动态调节，使 BP网络能够有效地学习和记住输入-输出之间的对应关系，使其能够更好地完成不同的任务。此外， BP神经网路的自组织与学习特性，使其对复杂的输入与产出之间的对应关系有较强的适应性，可以较好地解决常规算法无法处理的问题。

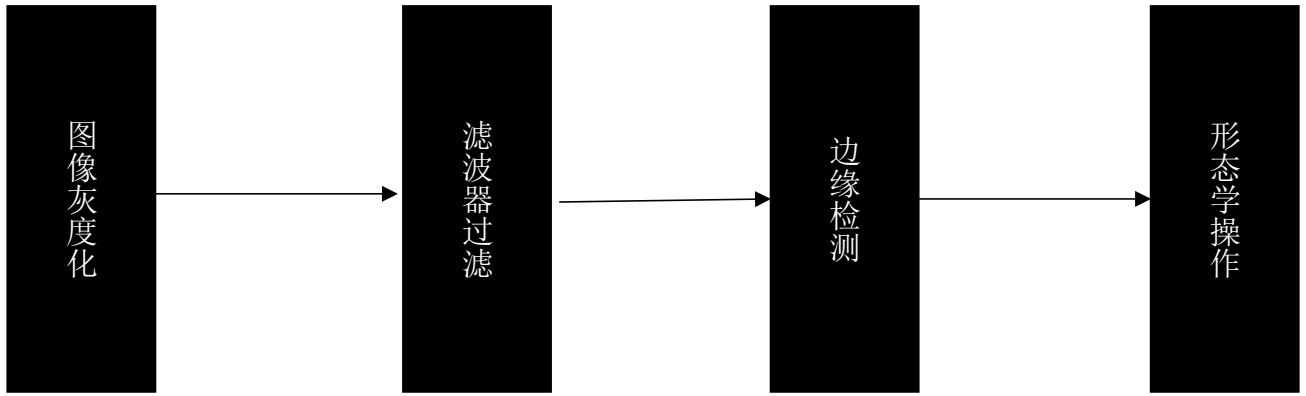
车牌识别系统的方案设计

1）对车牌进行预处理。
2）对车牌进行定位。
3）对车牌字符信息进行划分，获得车牌的每一个特征。
4）再将这些特征输入识别模块。通过对这些部件的特征进行辨识，由此可以获得车牌数量的数据。具体框架流程如图



图像预处理

在汽车牌照识别过程中，对汽车牌照进行预处理，其效果好坏直接影响到汽车牌照的正确与否。对汽车图像进行预处理，其目的在于凸显汽车牌照的主体特性，消除图像中的噪点，消除图像传输过程中的模糊问题等。



图像预处理

车牌灰度化

在对图像灰度化的过程进行分析时，灰度过程主要处理的对象是RGB三种成分（也就是红、绿、蓝三种英文的简写），运用RGB成分对颜色进行表征；本文使用了一种基于平均值的图象灰度化方法，以 R, G, B三个变量的平均值来表示图象的灰度，从而**降低图象的信息量**。对原图进行灰度化处理后，就可以得到得到灰度化图像。

图像预处理

滤波器过滤

不管是车牌图像，还是别的图像，只要是图像，其在**图像处理的过程中都会被噪音所影响**，也就是会生成难以去除的噪点，导致图片模糊，在输入输出的过程中，噪声主要包括有三个，高斯噪声，椒盐噪声，量化噪声，这三个主要的噪声都会对图像的处理产生非常大的影响，比如，会导致后续边缘提取过程中出现差错，所以去噪在图像处理的过程中是非常重要也是非常必要的。针对这一问题，本课题拟对均值滤波，中值滤波，维纳滤波等三种降噪算法进行研究。采用最优滤波器，使图像具有较好的灰度级。

在本次基于BP神经网络的系统设计中，采取Wiener维纳滤波对图像进行去噪处理，滤波后的图像如图所示，也就会使得图像更加平滑更加清晰化。

图像预处理

边缘检测

基于灰度变异原理，提出了一种基于灰度变异的图像分割方法——边缘检测方法。常用的边界提取算子有 Prewitt, Sobel, Sanny算子等。在图中给出sobel的边界提取的结果。

图像预处理

形态学操作

腐蚀是一个去除界面上的结点并使其向内收缩的图像处理过程。这个方式经常被用来移除微小和不重要的图像参照物。膨胀运算是指把被摄对象和被摄对象之间的点与被摄对象之间的关系进行合并，以扩大被摄对象的边缘，一般被用来填充被摄对象的空白部分。而开运算操作属于“先腐蚀运算后膨胀运算”的算法。开运算操作主要是对小对象进行剔除，细化边界，在不显著变化的情况下对大对象进行平滑处理，特别是在消除噪音以及分离连接对象方面。其基本作用与腐蚀相似，但与腐蚀的区别在于，其可以在一定程度上维持被腐蚀图像的原有尺寸。然后对图象进行先蚀后扩的封闭操作。该物质能够填充对象中细小的间隙，将邻近对象联系起来，并且保持它们的边界光滑，而不会显著地改变它们的区域。该算法能在消除牌照边缘的情况下，**有效地去除了图象中的细小噪声，并使目标边缘光滑，而不影响目标区域。**如图所示

车牌定位

粗定位

粗定位是指从图像中提取出汽车牌照的边缘点。首先通过直方图均衡来强化影像中的信息，其次通过灰度变换来提高影像的效果，接着通过滤波器来降低噪音，利用 Sobel算子进行边界检测来获取结构特性，最终通过封闭操作等形式来获得粗定位影像。如图所示。

细定位

精细定位的目的是要得到完整且准确的车辆牌照的字符信息。本文使用HSV 颜色空间的汽车车牌识别方法。在获得HSV颜色空间中所抽取的汽车牌照区域以后，该方法把它转化成 RGB影像，然后获得一张蓝色的牌照，在判断出车牌的具体位置后，通过对该对象区域的长宽比进行过滤，就可以获得一张被识别出来的车牌，然后用 Radon算法对所获得的车牌进行倾斜修正，最终获得了一张精确的定位结果。如图所示

字符分割

2.3字符分割

目前，我国的车牌字符规定由7个字符组成，所以要从已确定的牌照图像中分离出七个确定的字符位置。这个过程中，要求首先将汽车牌照图像进行灰度处理，然后对牌照图像做二值运算，消除噪音点。在图中，基于每一个汉字的长、宽、字间隔等几何规则，利用竖直射分割方法将汉字分割成七个较小的区域，然后将各区域按32x40的尺寸进行调节，调整后的字符如图所示。

2.4字符识别

在对牌照图象进行特征提取后，就可以进行车牌字符的识别了。汽车牌照识别的终极目的就是对汽车牌照进行精确的辨识，而文字辨识可以说是汽车牌照识别的关键。文字辨识的理论基础包括图案比对与网路学习。其主要流程为：从已划分出来的汉字中提取表征待识别汉字图案实质的表现方式与特征，并将该表示与事先储存在电脑中的规范汉字格式的表示方式及特性一一比对，最后依据某些判定标准作出判定。从计算机储存的规范文字样式的表现方式及特征组合中，找出与要判断文字样式最相近的表现方式及特性，并根据该决定作出判断，得到辨识的结果。在对各种算法进行归纳后，将 BP神经网络应用于字符识别中。

构建BP神经网络

针对一般汽车牌照的特征，将其分为两个子网络进行分类。其中，第一子网络用于对牌照编号中的首汉字进行标识，而第二子网络用于对牌照编号中的2~7位进行标识，其中除了英文以外，其他6位都是由英文与数码组成。第一个子网络输入结点212个，隐含结点100个，输出结点31个。第二个子网路则是建立一种 BP网路，它有44个输入，35个隐藏级，34个输出级。这两个子网络的使用 Sigmode。

BP神经网络字符识别

训练 BP 神经网络

每个车牌的字符特征值分别用 BP网络进行训练。

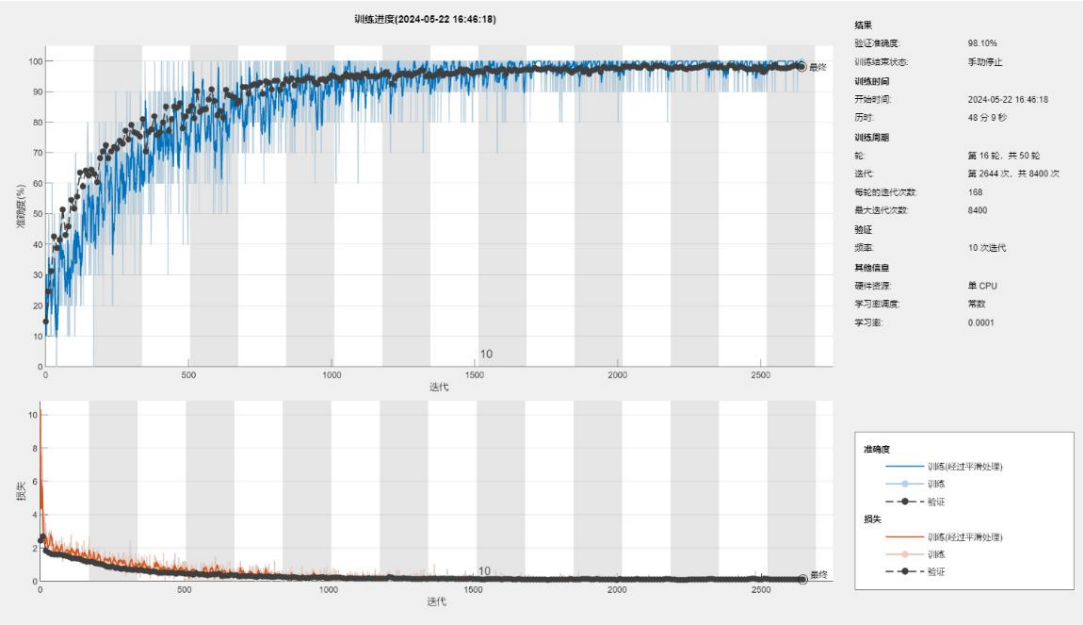
第一分网络搜集了中国31个省的缩写汉字，总共3000多幅文字图像。在此基础上，对以上图像分别进行了特征值的抽取，并将其送入 BP网络中用于神经网络的学习，其中一些汉字的训练样例见图所示。

第2个 BP网络中，收集英文汉字24个，“0”~“9”10位数字字符，共计4000多幅图像。在此基础上，对图像进行特征抽取，并将其送入 BP网络中用于神经网络的学习训练，其中一些数值及英文字母作为训练样例。

实验结果

实验结果

完成BP神经网络训练后，就可以对所拍到的车牌进行图像增强，灰度化处理，滤波器过滤，边缘检测，闭运算，膨胀操作，粗定位，精细定位，字符分割及识别等一系列操作。实验结果如下图所示，就可以得到目标车牌的信息。



实验结果

字符类型	字符总数	识别成功数	识别正确率
汉字	20	19	95%
英文字母	38	36	94.7%
数字	82	79	96.3%
总计	140	134	95.7%

参考文献

1. 李洪林.神经网络方法与高信噪比方法联合拾取初至[D].成都理工大学,2008.
2. 胡来丰.基于粗糙集BP神经网络个人信用评估模型[D].电子科技大学,2015.
3. 吕凯煜.基于图像处理技术的车牌检测系统[J].数码世界,2016,(05):22.
4. 刘小燕.基于图像处理方法的股票数据分析研究[D].重庆大学,2012.
5. 桂方燚,武文星,任维康.基于PSO-CNN神经网络的车牌识别系统[J].华北科技学院学 报,2021,18(05):100-106.
6. 张玉静,王大全,马艳辉.一种改进的自适应中值滤波算法研究[J].工业控制计算机,2016,29(11):109-110+113.
7. 高勇.基于BP神经网络的车牌识别建模及实现[J].电子测试,2021,(01):44-45+78.
8. 周科伟.Matlab环境下基于神经网络的车牌识别[D].西安电子科技大学,2009.
9. 吴秀丽.车牌识别系统的设计与实现[D].东北大学,2008.
10. 马永力.图像处理在车牌识别系统中的应用[D].武汉理工大学,2006.
11. 高勇.基于BP神经网络的车牌识别建模及实现[J].电子测试,2021,(01):44-45+78.
12. 赖媛媛,原虹.基于神经网络的印刷体数字识别算法的研究[J].科技传播,2012,4(18):215+211.

我的汇报结束
谢谢大家！

清華大學
Tsinghua University

CAE大作业报告

Group | 刘程鸣 焦昕远

目录 / Contents

清華大學
Tsinghua University

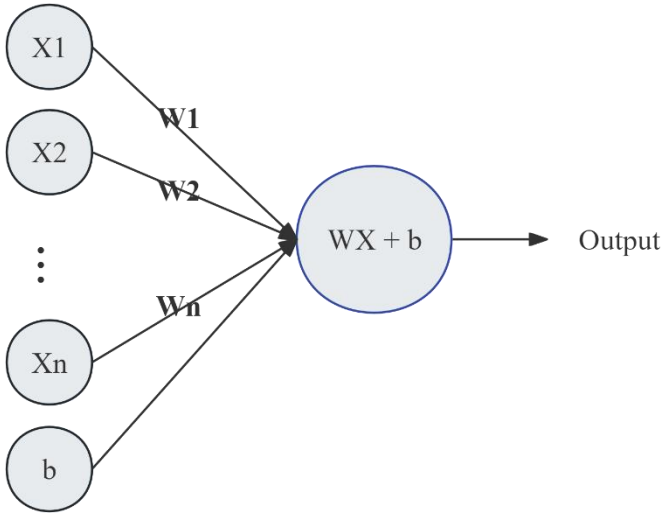
01.知识脉络梳理

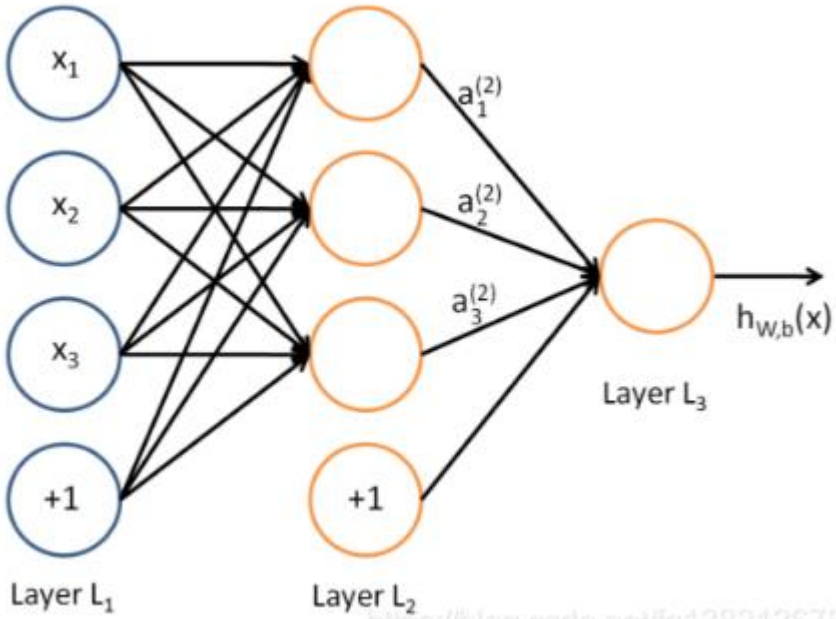
知识脉络梳理

一.多层感知机模型

多层感知机（MLP Multilayer Perceptron）也叫人工神经网络（ANN Artificial Neural Network），除了输入输出层，它中间可以有多个隐藏层。

目标：完成分类任务；回归任务等





在层间添加了激活函数后，就可以产生非线性，使得深度出现。（手动提取特征→自动提取特征）

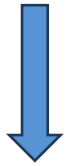
知识脉络梳理

二.卷积神经网络

1.Conv的由来

假设我们有一个足够充分的照片数据集，数据集中是拥有标注的照片，每张照片具有百万级像素，这意味着网络的每次输入都有一百万个维度。

多层感知机的输入是二维图像X，其隐藏表示H在数学上是一个矩阵，在代码中表示为一维张量。其中X和H具有相同的形状。这意味着我们的中间层参数为四阶的张量。



数据量巨大！！！

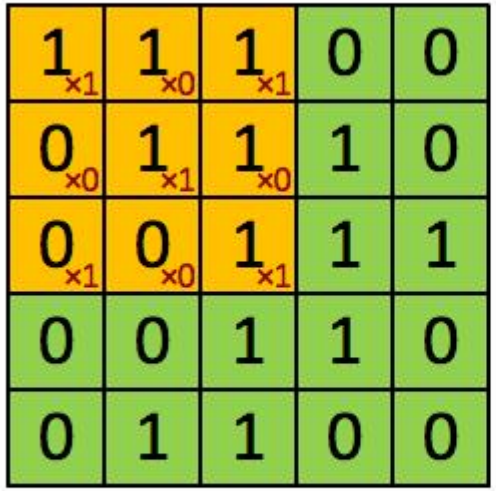
图片的特性：
平移不变性，局部性

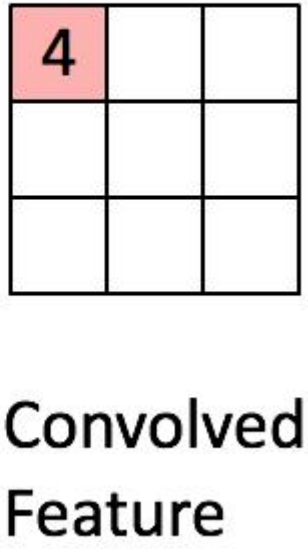
知识脉络梳理

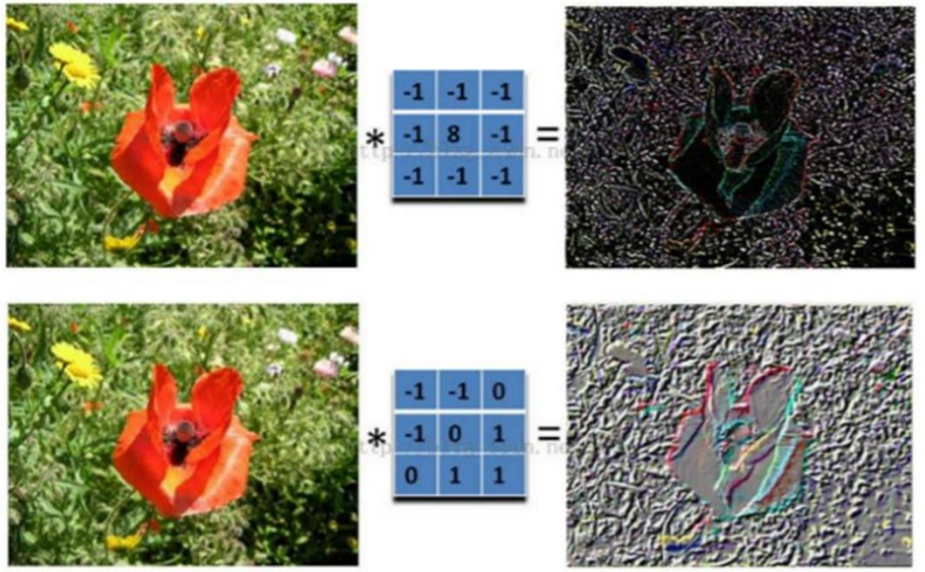
二.卷积神经网络

何为卷积?

简单的理解：与人眼观看事物原理相似，卷积神经网络可以看到事物的轮廓，分析不同通道的局部空间的元素大小，以此来抽象提取特征。







不同的卷积核　　不同的效果

清華大學
Tsinghua University

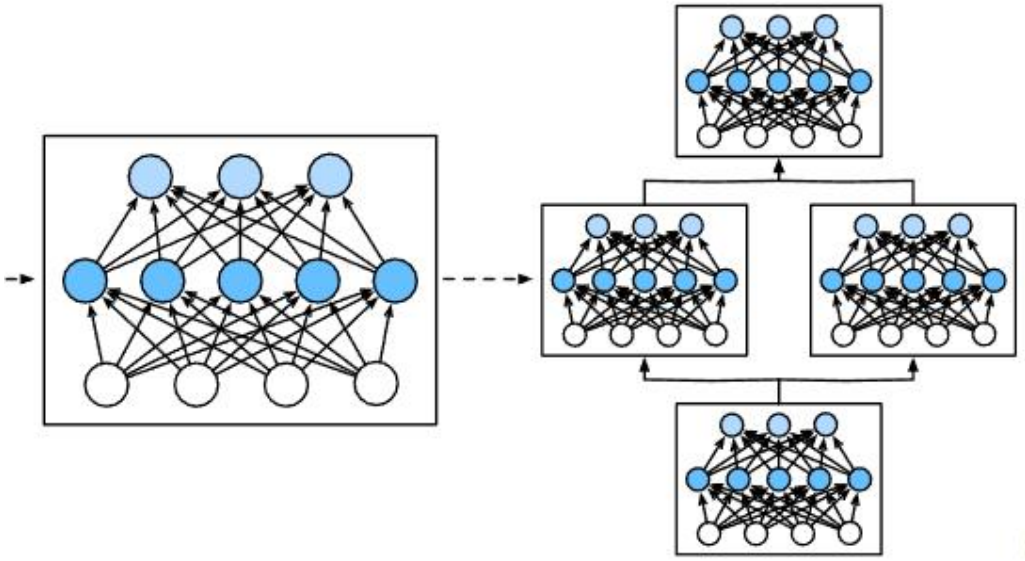
02.底层逻辑从0实现ResNet

底层逻辑从0实现ResNet

一.预备知识

1.何为块?

块（*block*）可以描述单个层、由多个层组成的组件或整个模型本身。 使用块进行抽象的一个好处是可以将一些块组合成更大的组件， 这一过程通常是递归的



2.批量规范化（Batch Normalization）

在每次训练迭代中，我们首先规范化输入，即通过减去其均值并除以其标准差，其中两者均基于当前小批量处理。 接下来，我们应用比例系数和比例偏移。

1.全连接层：我们将批量规范化层置于全连接层中的仿射变换和激活函数之间。

2.卷积层：在卷积层之后和非线性激活函数之前应用批量规范化

底层逻辑从0实现ResNet

二.实现Batch_Norm

$$\mathrm{BN}(\mathbf{x}) = \boldsymbol{\gamma} \odot \frac{\mathbf{x} - \hat{\boldsymbol{\mu}}_{\mathcal{B}}}{\hat{\boldsymbol{\sigma}}_{\mathcal{B}}} + \boldsymbol{\beta}.$$

$$\hat{\boldsymbol{\mu}}_{\mathcal{B}} = \frac{1}{|\mathcal{B}|} \sum_{\mathbf{x} \in \mathcal{B}} \mathbf{x},$$

$$\hat{\boldsymbol{\sigma}}_{\mathcal{B}}^2 = \frac{1}{|\mathcal{B}|} \sum_{\mathbf{x} \in \mathcal{B}} (\mathbf{x} - \hat{\boldsymbol{\mu}}_{\mathcal{B}})^2 + \epsilon.$$

μ ^B 是小批量B 的样本均值， σ ^B 是小批量B 的样本标准差。 应用标准化后， 生成的小批量的平均值为0和单位方差为1。通常包含 拉伸参数（scale） γ 和偏移参数（shift） β ，它们的形状与x 相同。 γ 和β 是需要与其他模型参数一起学习的参数。

在方差估计值中添加一个小的常量ϵ >0，以确保永远不会除以零。

一.batch_norm实现

```python
import torch
from torch import nn
from d2l import torch as d2l

def batch_norm(X, gamma, beta, moving_mean, moving_var, eps, momentum): #moving_mean/var:近似于遍历的数据集的均值与方差。最终为整个数据集
    # 通过is_grad_enabled来判断当前模式是训练模式还是预测模式
    if not torch.is_grad_enabled():
        # 如果是在预测模式(推理)下，直接使用传入的移动平均所得的均值和方差（因为甚至可能是一张图片）
        X_hat = (X - moving_mean) / torch.sqrt(moving_var + eps)
    else:
        assert len(X.shape) in (2, 4)
        if len(X.shape) == 2:  #batch_size,features_num
            # 使用全连接层的情况，计算特征维上的均值和方差
            mean = X.mean(dim=0)   #沿着batch通道做平均
            var = ((X - mean) ** 2).mean(dim=0)
        else:
            # 使用二维卷积层的情况，计算不同通道维上（axis=1）的均值和方差。
            # 这里我们需要保持X的形状以便后面可以做广播运算
            mean = X.mean(dim=(0, 2, 3), keepdim=True)  #沿着batch通道对同一个channel编号的同一个h，w编号的元素做平均；保持相同维。
            var = ((X - mean) ** 2).mean(dim=(0, 2, 3), keepdim=True)
        # 训练模式下，用当前的均值和方差做标准化
        X_hat = (X - mean) / torch.sqrt(var + eps)
        # 更新移动平均的均值和方差
        moving_mean = momentum * moving_mean + (1.0 - momentum) * mean
        moving_var = momentum * moving_var + (1.0 - momentum) * var
    Y = gamma * X_hat + beta  # 缩放和移位
    return Y, moving_mean.data, moving_var.data
```

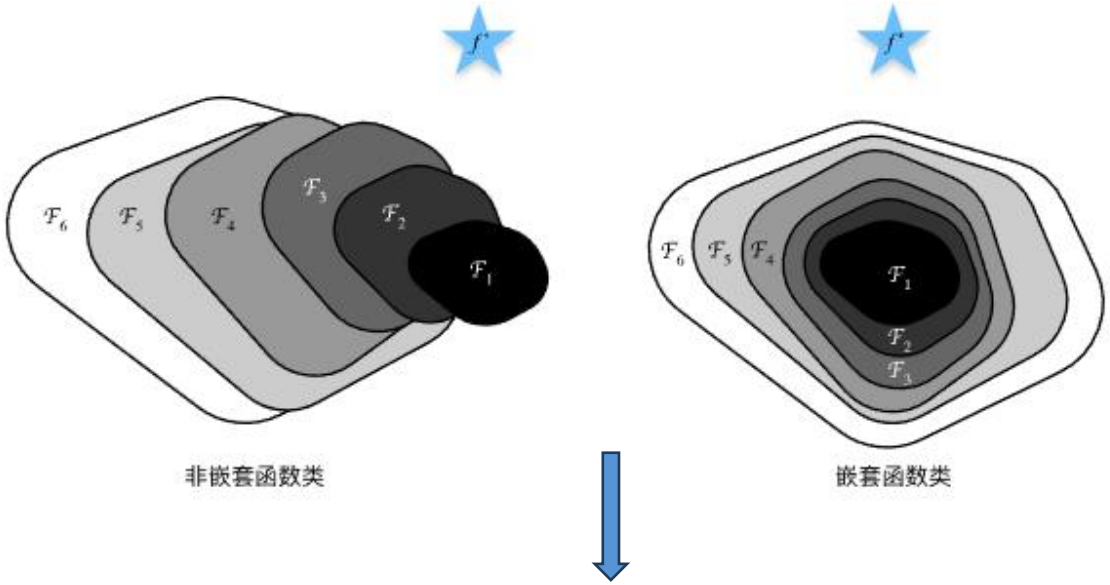
```python
class BatchNorm(nn.Module):
    # num_features: 完全连接层的输出数量或卷积层的输出通道数。
    # num_dims: 2表示完全连接层，4表示卷积层
    def __init__(self, num_features, num_dims):
        super().__init__()
        if num_dims == 2:
            shape = (1, num_features)
        else:
            shape = (1, num_features, 1, 1)
        # 参与求梯度和迭代的拉伸和偏移参数，分别初始化成1和0
        self.gamma = nn.Parameter(torch.ones(shape))  #需要迭代，放入parameter
```

底层逻辑从0实现ResNet

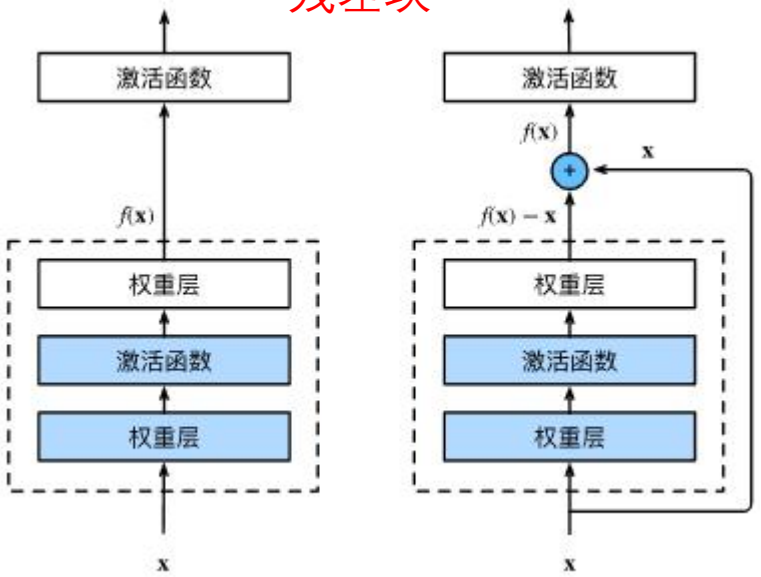
三.ResNet（残差网络）

目标：得到更近似真正f^*的函数。

需要：我们需要设计一个更强大的架构F'。
然而现实中，我们的更新可能会更糟。只有当较复杂的函数类包含较小的函数类时，我们才能确保提高它们的性能。 对于深度神经网络，如果我们能将新添加的层训练成**恒等映射**（identity function）$f(x)=x$，新模型和原模型将同样有效。



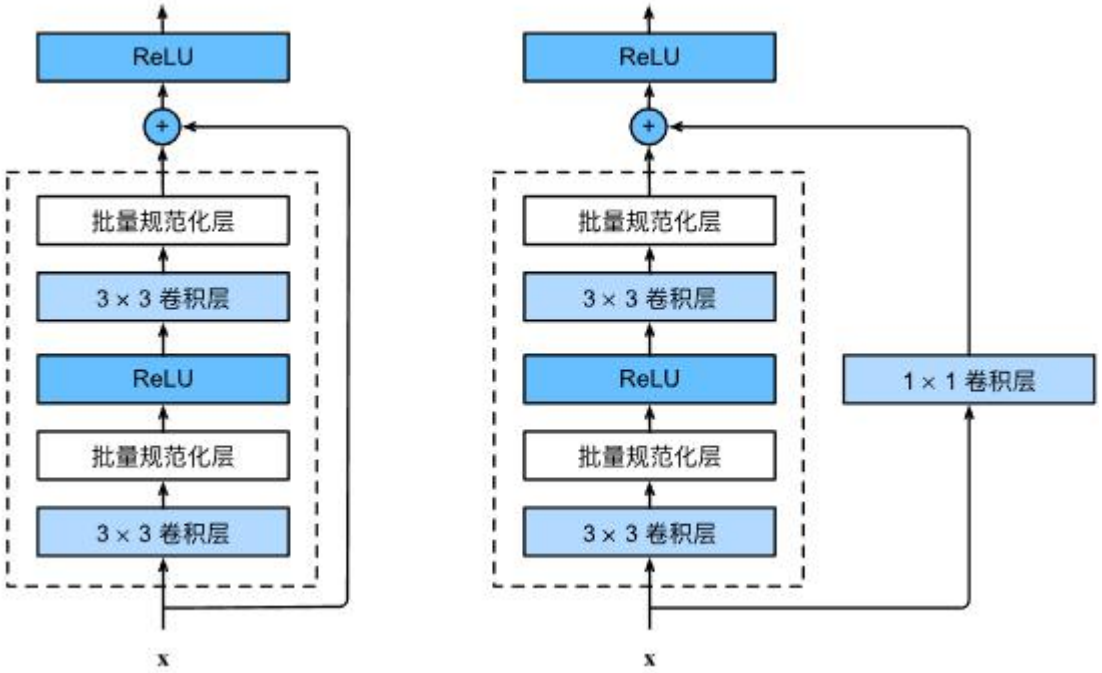
残差块



底层逻辑从0实现ResNet

三.构造残差块

目标：得到更近似真正f * 的函数。



构造残差块

```
class Residual(nn.Module):
    def __init__(self, input_channels, num_channels,
                 use_1x1conv=False, strides=1):  #strides默认为1
        super().__init__()
        self.conv1 = nn.Conv2d(input_channels, num_channels,
                               kernel_size=3, padding=1, stride=strides)
        self.conv2 = nn.Conv2d(num_channels, num_channels,
                               kernel_size=3, padding=1)
        if use_1x1conv:
            self.conv3 = nn.Conv2d(input_channels, num_channels,
                                   kernel_size=1, stride=strides)
        else:
            self.conv3 = None
        self.bn1 = nn.BatchNorm2d(num_channels)  #创建对象
        self.bn2 = nn.BatchNorm2d(num_channels)

    def forward(self, X):
        Y = F.relu(self.bn1(self.conv1(X)))
        Y = self.bn2(self.conv2(Y))
        if self.conv3:          #非None的对象会被判断为True
            X = self.conv3(X)
        Y += X
        return F.relu(Y)
```

```
def resnet_block(input_channels, num_channels, num_residuals,
                 first_block=False):
    blk = []
    for i in range(num_residuals):
        if i == 0 and not first_block:
            blk.append(Residual(input_channels, num_channels,
                               use_1x1conv=True, strides=2))
        else:
            blk.append(Residual(num_channels, num_channels))
    return blk
```

底层逻辑从0实现ResNet

四.训练‘土模型’

1. 加载数据集

2.构建模型

3.构造训练函数

训练模

构建模型

+ Code

[5]:

```
# 头部
b1 = nn.

# ResNet
b2 = nn.
b3 = nn.
b4 = nn.
b5 = nn.
# 集合
net = nn

if torch
    net
```

```
loss_func = nn.CrossEntropyLoss()
if torch.cuda.is_available():
    loss_func = loss_func.cuda()
#优化器
learning_rate = 1e-2  #方便修改学习速率 1*(10)^-2
optimizer = torch.optim.SGD(net.parameters(),lr=learning_rate)

total_train_step = 0
#记录测试的次数
total_test_step = 0
#训练的轮数
epoch = 10

#添加tensorboard反映训练成果可视化
writer = SummaryWriter('./logs_train_module')
start_time = time.time()  #开始计时
x_time = 0

for i in range(epoch):
    print(f'---------第{i+1}轮训练开始--------')
    # 训练步骤开始
    net.train()  # 进入训练状态
    for data in train_dataloader:
        imgs, targets = data
        if torch.cuda.is_available():
            imgs = imgs.cuda()
            targets = targets.cuda()
        outputs = net(imgs)
        loss = loss_func(outputs,targets)
        #优化器调优
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        #记录训练次数
        total_train_step += 1
```

底层逻辑从0实现ResNet

清華大學 Tsinghua University

结果展示

可以看见第7轮已经基本收敛。（没有添加drop_out,权重衰减，数据增强）

```
---------第10轮训练开始--------
第85百次训练打点计时为：202.35275077819824
训练次数为8500时,损失值的大小为0.0953054130077362
第86百次训练打点计时为：204.40627431869507
训练次数为8600时,损失值的大小为0.08025156706571579
第87百次训练打点计时为：206.49851942062378
训练次数为8700时,损失值的大小为0.1531320959329605
第88百次训练打点计时为：208.55358910560608
训练次数为8800时,损失值的大小为0.051539644598960876
第89百次训练打点计时为：210.58685541152954
训练次数为8900时,损失值的大小为0.05554988607764244
第90百次训练打点计时为：212.70303463935852
训练次数为9000时,损失值的大小为0.10396118462085724
第91百次训练打点计时为：214.72246980667114
训练次数为9100时,损失值的大小为0.034840334206819534
第92百次训练打点计时为：216.76597833633423
训练次数为9200时,损失值的大小为0.11041515320539474
第93百次训练打点计时为：218.79982137680054
训练次数为9300时,损失值的大小为0.10826140642166138
整体测试集上的loss为55.38446523249149
整体测试集上的正确分类的概率为0.8937000036239624
模型已保存
```

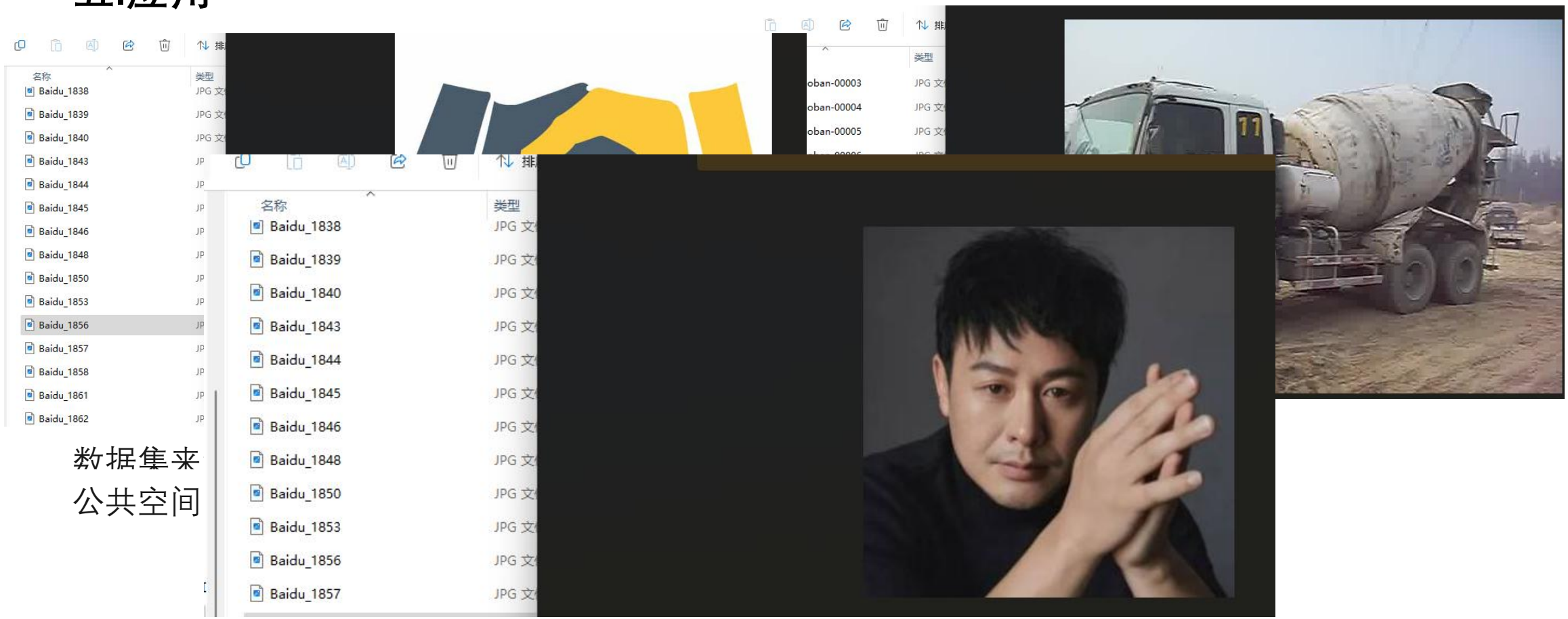
+ Code + Markdown

```
---------第7轮训练开始--------
第57百次训练打点计时为：136.24569058418274
训练次数为5700时,损失值的大小为0.37525245547294617
第58百次训练打点计时为：138.2963502407074
训练次数为5800时,损失值的大小为0.0802246704697609
第59百次训练打点计时为：140.34791707992554
训练次数为5900时,损失值的大小为0.07480219006538391
第60百次训练打点计时为：142.43676948547363
训练次数为6000时,损失值的大小为0.1142892986536026
第61百次训练打点计时为：144.48410296440125
训练次数为6100时,损失值的大小为0.14732776582241058
第62百次训练打点计时为：146.55884170532227
训练次数为6200时,损失值的大小为0.22007973492145538
第63百次训练打点计时为：148.61256170272827
训练次数为6300时,损失值的大小为0.13594159483909607
第64百次训练打点计时为：150.6430733203888
训练次数为6400时,损失值的大小为0.0727200135588646
第65百次训练打点计时为：152.70774006843567
训练次数为6500时,损失值的大小为0.1740550547838211
整体测试集上的loss为48.880593713372946
整体测试集上的正确分类的概率为0.8960999846458435
```

成功！！

底层逻辑从0实现ResNet

五.应用



数据集来公共空间

最终测试集精度只有47%

项目分工

一.共同任务

提高代码能力;
共同学习重要机器学习库;
学习并掌握理论知识;
动手构建底层逻辑代码，搭建模型。

二.分工任务

刘程鸣：查找学习，算力资源，在云计算平台上完成代码实践。
焦昕远：查找数据集，尝试应用于特种车辆识别。

清華大學
Tsinghua University

感谢观看！