A2: 卷积网络 (30 points)

- 任务：构建一个卷积神经网络，完成图像分类 (30 Points)
- 数据集：CIFAR-10

本题目考察如何设计并实现一个基于卷积神经网络的图像分类器。设置本题目的目的如下：

– 理解卷积神经网络的基本结构、代码实现及训练过程

– 应用dropout和多种normalization方法，理解它们对模型泛化能力的影响

– 理解如何通过交叉验证，为神经网络找到最好的hyperparameters

- 附加题

– 在训练网络的过程中，可根据需要自由尝试其它提升性能的方法，例如通过增加模型层数、使用不同的正则化方法、使用模型集成等 (+10 points)

基础知识

卷积神经网络

在CNN中，一般都是使用一个卷积核来提取图像的局部特征，其一般计算的数学公式如下

$$y(i,j) = (X * K)(i,j) = \sum_m \sum_n X(i+m, i+n) \cdot K(m,n)$$

下面我们来举例说明如何计算

假设我们有一个$3*3$的图片，卷积核的尺寸为$2*2$

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 19 & 25 \\ 37 & 43 \end{bmatrix}$$

其中$19 = 0*0 + 1*1 + 2*3 + 3*4$，$25 = 0*1 + 1*2 + 2*4 + 3*5$，其余两个不作赘述

值得注意的是，就在刚才的例子中，3*3的输入经过卷积运算后变为2*2的输出，这显然丢失了一部分信息（这部分信息在输入矩阵的边框）

为了解决这个问题，需要在原始输入的矩阵周围padding一圈0，例如将刚才的例子padding一圈0将原市输入矩阵包围起来，那padding后的尺寸为$5*5$，再经过卷积计算后的结果为$4*4$

一般来说，padding的宽与高有如下公式(其中p代表padding，k代表kernel)

$$
\begin{aligned}
p_h &= k_h - 1 \\
p_w &= k_w - 1
\end{aligned}
$$

填充后输出形状表达式如下（n代表输入图像）

$$
(n_h - k_h + p_h + 1) \cdot (n_w - k_w + p_w + 1)
$$

池化操作可以理解为一种特殊的卷积运算，每次取与池化层相同大小的区域经过特殊运算得出池化操作的结果（这里选取最大池化，也有平均池化等操作）

$$
\begin{bmatrix}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8
\end{bmatrix}
*
\begin{bmatrix}
\text{最} & \text{大} \\
\text{池} & \text{化}
\end{bmatrix}
=
\begin{bmatrix}
4 & 5 \\
7 & 8
\end{bmatrix}
$$

BatchNorm2d

对图像作归一化操作也是很重要的一件事，可以平滑梯度，防止梯度消失梯度爆炸的问题，假设我们输入的Tensor是$[N, C, H, W]$，是在对每个channel这个维度进行归一化操作，归一化操作肯定是要计算平均值μ和方差σ

$$
\begin{aligned}
\mu_C &= \frac{1}{N*H*W} \sum_{i=1}^{N} \sum_{j=1}^{H} \sum_{k=1}^{W} x_{i,C,j,k} \\
\sigma_c^2 &= \frac{1}{N*H*W} \sum_{i=1}^{N} \sum_{j=1}^{H} \sum_{k=1}^{W} (x_{i,C,j,k} - \mu_C)^2
\end{aligned}
$$

计算获得了均值和方差，实际中归一化的计算公式如下（其中ϵ是为了防止除0问题，一般是一个很小的数值）

$$
\hat{x}_{i,C,j,k} = \frac{x_{i,C,j,k} - \mu_c}{\sqrt{\sigma_C^2 + \epsilon}}
$$

到此为止仅仅是数学层面的表达，实际模型中是要有一个线性关系的，多了两个参数γ_c β_c，这两个参数是可以学习的，实际模型的计算公式如下

$$y_{i,C,j,k} = \gamma_c \cdot \frac{x_{i,C,j,k} - \mu_c}{\sqrt{\sigma_C^2 + \epsilon}} + \beta_c$$

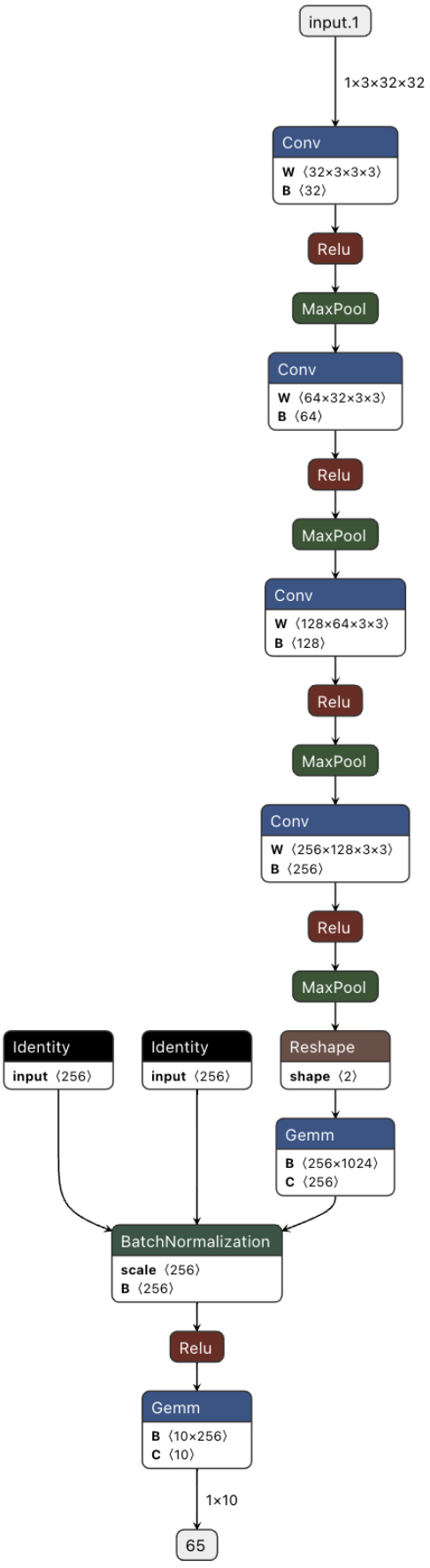
数据准备

该部分省略，因数据与上次全连接神经网络的数据完全一致，故不在此赘述。

模型设计

模型结构如下图所示（选择的是经过交叉验证后的最好超参数构建的模型）

激活函数选择的为ReLU


input.1
1×3×32×32
Conv
W ⟨32×3×3×3⟩
B ⟨32⟩
Relu
MaxPool
Conv
W ⟨64×32×3×3⟩
B ⟨64⟩
Relu
MaxPool
Conv
W ⟨128×64×3×3⟩
B ⟨128⟩
Relu
MaxPool
Conv
W ⟨256×128×3×3⟩
B ⟨256⟩
Relu
MaxPool
Identity
input ⟨256⟩
Identity
input ⟨256⟩
Reshape
shape ⟨2⟩
Gemm
B ⟨256×1024⟩
C ⟨256⟩
BatchNormalization
scale ⟨256⟩
B ⟨256⟩
Relu
Gemm
B ⟨10×256⟩
C ⟨10⟩
1×10
65

实验结果

损失函数选择为Cross Entropy，优化器为Adam

利用K折交叉验证搜索的参数如下

```
lr_grid = [1e-2, 1e-3, 1e-4]
epoch_grid = [9,12,15]
conv_layer_grid = [3,4,5]
```

如下是我们通过K折交叉验证得到的结果，以json的格式保存了结果，key的位置是由`{learning_rate}-{epoch}-{convultion_layer_number}`，K折交叉验证选择的K=5，每一折都作epoch次训练，最后在验证集上和测试集上计算出F1-score，最终对5次（K=5）在本轮尝试的超参数组合上训练的结果分别求取F1-score的平均值，作为最后的衡量指标


```
{"0.01+9+3 validation performance": 1.794,
"0.01+9+3 test performance": 1.8199999999999998,
"0.01+9+4 validation performance": 1.764,
"0.01+9+4 test performance": 1.8199999999999998,
"0.01+9+5 validation performance": 1.802,
"0.01+9+5 test performance": 1.8199999999999998,
"0.01+12+3 validation performance": 1.8060000000000003,
"0.01+12+3 test performance": 1.8199999999999998,
"0.01+12+4 validation performance": 2.07,
"0.01+12+4 test performance": 3.7380000000000004,
"0.01+12+5 validation performance": 1.778,
"0.01+12+5 test performance": 1.8199999999999998,
"0.01+15+3 validation performance": 3.56,
"0.01+15+3 test performance": 2.2380000000000004,
"0.01+15+4 validation performance": 1.834,
"0.01+15+4 test performance": 1.8199999999999998,
"0.01+15+5 validation performance": 1.812,
"0.01+15+5 test performance": 1.8199999999999998,
"0.001+9+3 validation performance": 74.43199999999999,
"0.001+9+3 test performance": 74.24,
"0.001+9+4 validation performance": 74.4,
"0.001+9+4 test performance": 74.40200000000002,
"0.001+9+5 validation performance": 73.96,
```

```
"0.001+9+5 test performance": 73.99000000000001,
"0.001+12+3 validation performance": 75.202,
"0.001+12+3 test performance": 75.09400000000001,
"0.001+12+4 validation performance": 75.376,
"0.001+12+4 test performance": 75.14000000000001,
"0.001+12+5 validation performance": 74.696,
"0.001+12+5 test performance": 74.636,
"0.001+15+3 validation performance": 75.214,
"0.001+15+3 test performance": 75.05,
"0.001+15+4 validation performance": 75.744,
"0.001+15+4 test performance": 75.578,
"0.001+15+5 validation performance": 74.764,
"0.001+15+5 test performance": 74.57000000000001,
"0.0001+9+3 validation performance": 71.53,
"0.0001+9+3 test performance": 71.45599999999999,
"0.0001+9+4 validation performance": 70.874,
"0.0001+9+4 test performance": 70.832,
"0.0001+9+5 validation performance": 71.646,
"0.0001+9+5 test performance": 71.63,
"0.0001+12+3 validation performance": 72.68199999999999,
"0.0001+12+3 test performance": 72.52799999999999,
"0.0001+12+4 validation performance": 72.50999999999999,
"0.0001+12+4 test performance": 72.47,
"0.0001+12+5 validation performance": 72.554,
"0.0001+12+5 test performance": 72.138,
"0.0001+15+3 validation performance": 73.516,
"0.0001+15+3 test performance": 73.28200000000001,
"0.0001+15+4 validation performance": 72.872,
"0.0001+15+4 test performance": 73.088,
"0.0001+15+5 validation performance": 73.75800000000001,
"0.0001+15+5 test performance": 73.644}
```


综合考量验证集和测试集的F1-score平均值，学习率为`1e-3`，训练轮次为`15`，卷积的层数为`4`的时候，模型的综合性能表现最好

其中可以看到学习率为0.01的时候，训练的结果十分差，用0.01的学习率随机选取一个卷积层数和训练轮数，观察到在第二轮训练的时候loss固定在一个十分高的数值左右来回震荡，显然是学习率过大导致的无法收敛。