作业4：神经网络

1. 神经网络的学习能力

1.AND 与 OR 分类任务

由于激活函数形式

$$f(a)\begin{cases}1 & a>0\\0 & a<0\end{cases}$$

AND型分类任务不妨取$\omega=[0.5,0.5]^T$，$b=-0.75$

$x=[1,1]$时，$\omega^Tx+b=0.25>0,f(\omega^Tx+b)=1$

$x=[0,0],[0,1]$或$[1,0]$时，均有$\omega^Tx+b<0,f(\omega^Tx+b)=0$

可以完成AND型分类任务

OR型分类任务不妨取$\omega=[0.5,0.5]^T$，$b=-0.25$

$x=[1,1]$时，$\omega^Tx+b=0.75>0,f(\omega^Tx+b)=1$

$x=[0,1]$或$[1,0]$时，$\omega^Tx+b=0.25>0,f(\omega^Tx+b)=1$

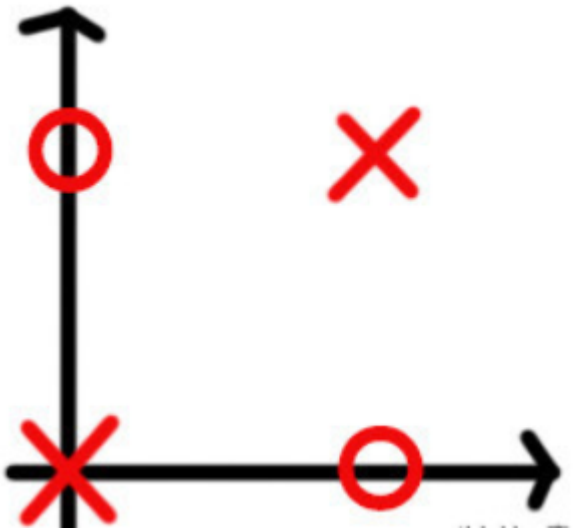
$x=[0,0]$时，$\omega^Tx+b=-0.25<0,f(\omega^Tx+b)=0$

可以完成OR型分类任务

2.两层神经网络缺陷

两层网络只能完成线性分类任务。上述网络结构中激活函数为感知器，可看作二维空间中的超平面决策面，对于超平面一侧的样本，感知器输出 1，对于另一侧的样本则输出 0。

与、或、非问题都是线性可分的问题，因而两层神经网络可以对其进行分类，而当样本分布为XOR型（异或型）时，上述网络结构无法完成分类任务。



如上图所示，在”异或“问题上找不到一条直线能把X和O分开，”异或“问题是一个不能用直线分类的问题，因而两层神经网络不能完成分类任务。

3.三层神经网络设计

不妨取$[\omega_{11},\omega_{21}]^T=[0.5,0.5]^T$，$b_1=-0.75$，即$z_1=f(\omega_{11}x_1+\omega_{21}x_2+b_1)$

$[\omega_{12},\omega_{22}]^T=[-0.5,-0.5]^T$，$b_2=0.25$，即$z_2=f(\omega_{12}x_1+\omega_{22}x_2+b_2)$

$[\omega_1,\omega_2]^T=[-1,-1]^T$，$b_1=0.5$，即$y=f(\omega_1z_1+\omega_2z_2+b)$

各层输出如下：

x_1	x_2	z_1	z_2	y
0	0	0	1	0
0	1	0	0	1
1	0	0	0	1
1	1	1	0	0

2. 单隐层全连接神经网络梯度计算

1.计算输出层与隐层节点间权值梯度

$$E(n)=\frac{1}{2}\sum_{k=1}^{c}(d_k(n)-y_k(n))^2, z_k(n)=\sum_{j=1}^{p}\omega_{ij}y_j(n)$$

$$\begin{aligned}\frac{\partial E}{\partial \omega_{kj}}&=\frac{\partial E}{\partial z_k}\cdot\frac{\partial z_k}{\partial \omega_{kj}}\\&=\frac{\partial E}{\partial y_k}\cdot\frac{\partial y_k}{\partial z_k}\cdot\frac{\partial z_k}{\partial \omega_{kj}}\\&=\frac{\partial E}{\partial y_k}\cdot\varphi'(z_k(n))\cdot\frac{\partial z_k}{\partial \omega_{kj}}\\&=(y_k(n)-d_k(n))\cdot\varphi'(z_k(n))\cdot y_j(n)\end{aligned}$$

2.计算隐层节点与输入层间权值梯度

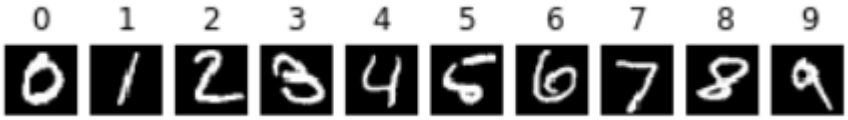
$$E(n)=\frac{1}{2}\sum_{k=1}^{c}(d_k(n)-y_k(n))^2, z_k(n)=\sum_{j=1}^{p}\omega_{ij}y_j(n)$$

$$\begin{aligned}\frac{\partial E}{\partial \omega_{ji}}&=\frac{\partial E}{\partial z_j}\cdot\frac{\partial z_j}{\partial \omega_{ji}}\\&=\frac{\partial E}{\partial y_j}\cdot\frac{\partial y_j}{\partial z_j}\cdot\frac{\partial z_j}{\partial \omega_{ji}}\\&=\frac{\partial E}{\partial y_j}\cdot\varphi'(z_j(n))\cdot y_i(n)\\&=\sum_{k=1}^{c}(\delta_k\cdot\omega_{kj})\cdot\varphi'(z_j(n))\cdot y_i(n)\\&=\sum_{k=1}^{c}[(y_k(n)-d_k(n))\cdot\varphi'(z_k(n))\cdot\omega_{kj}]\cdot\varphi'(z_j(n))\cdot y_i(n)\end{aligned}$$

4. 多层神经网络在MNIST数据集上的应用

a.显示手写数字图片

```
for i in range(10):
    pos=0
    for j in y_test:
        if j==i:
            plt.subplot(1,10,i+1)
            plt.imshow(X_test[pos].squeeze(),cmap='gray')
            plt.title('%i'% y_test[pos])
            plt.axis('off')
            break
        else:
            pos+=1
plt.savefig('./show_all.png')
plt.show()
```



b.数据预处理

```
dims = X_train.shape[0]
x_train=X_train.reshape(dims,1,784)
x_train=x_train.astype(np.float32)
x_train/=255.0
#维数转换与归一化
dims_test = X_test.shape[0]
x_test=X_test.reshape(dims_test,1,784)
x_test=x_test.astype(np.float32)
x_test/=255.0
```

c.搭建全连接网络

```
y_train=to_categorical(y_train)
y_test=to_categorical(y_test)
#设置隐层节点为100
model = Sequential()
model.add(Dense(100, input_shape=(784,)))
model.add(Activation('relu'))
model.add(Dense(10))
model.add(Activation('softmax'))
sgd = optimizers.SGD(lr=0.5, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(optimizer='sgd', loss='categorical_crossentropy')
model.fit(x_train, y_train,epochs = 20)
model.summary()
```

d.分割数据集及防止过拟合

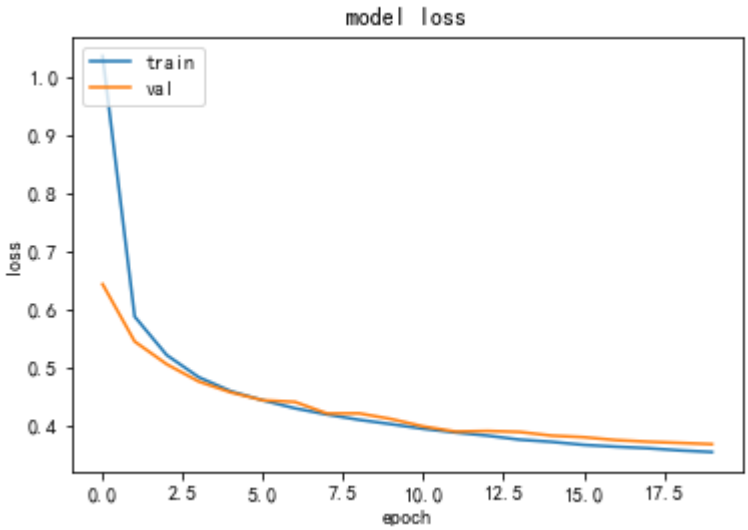
```
from sklearn.model_selection import train_test_split
x_train_new, x_test_new, y_train_new, y_test_new = train_test_split(x_train, y_train,
test_size=0.3, random_state=42)
x_test_new, x_val, y_test_new, y_val = train_test_split(x_test_new, y_test_new,
test_size=0.5, random_state=42)

from keras.callbacks import EarlyStopping, ModelCheckpoint
fBestModel = 'best_model.h5'
#5次内验证集loss不再下降，停止训练
early_stop = EarlyStopping(monitor='val_loss', patience=5, verbose=1)
best_model = ModelCheckpoint(fBestModel, verbose=0, save_best_only=True)
sgd = optimizers.SGD(lr=0.02, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(optimizer= sgd, loss='categorical_crossentropy')
model.fit(x_train_new, y_train_new, validation_data = (x_val, y_val), epochs=20,
          batch_size=128, callbacks=[best_model, early_stop])
model.summary()
```

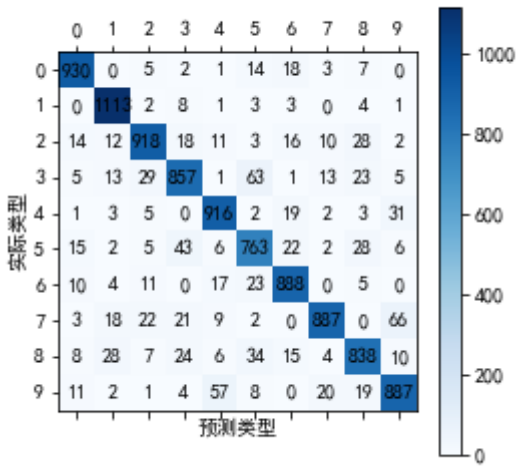
e.模型效果评估

(1).隐层节点个数5

训练集误差曲线与验证集误差曲线如下



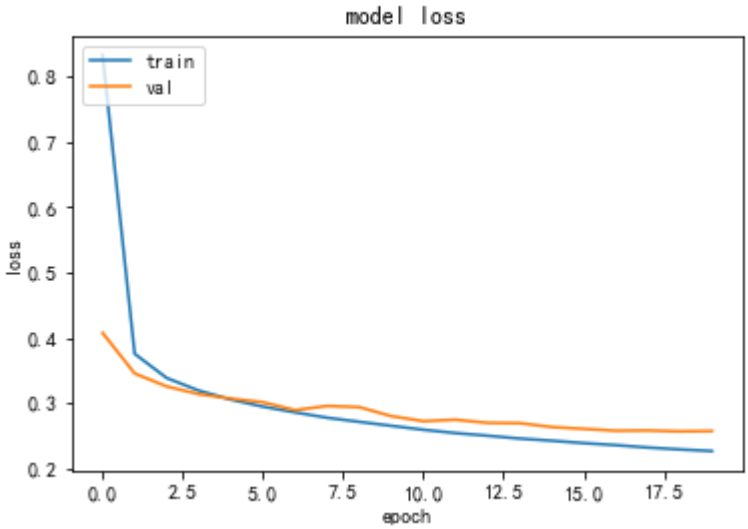
在测试集合上进行测试，使用sklearn绘制混淆矩阵如下（其中每列代表预测值，每行代表的是实际值，与作业示例略有不同，下同）



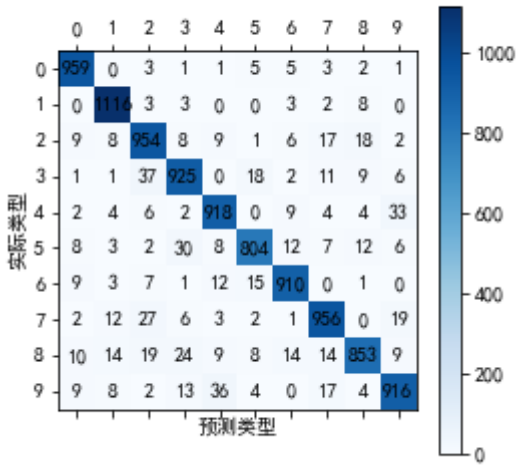
分类正确率为89.97%

(2).隐层节点个数10

训练集误差曲线与验证集误差曲线如下



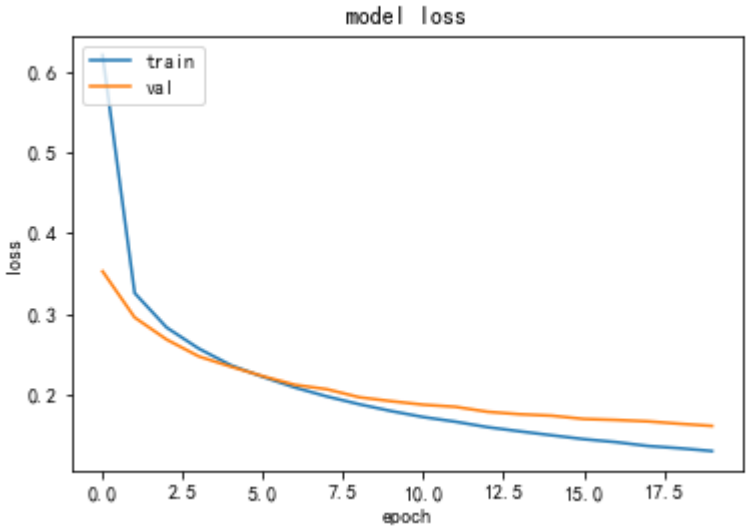
测试集合上进行测试，保存混淆矩阵如下:



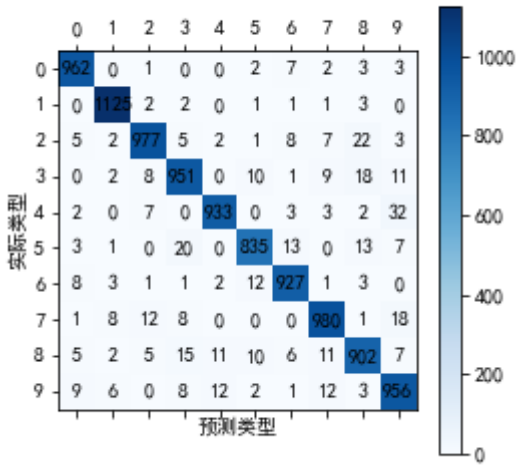
分类正确率为93.11%

(3).隐层节点个数20

训练集误差曲线与验证集误差曲线如下



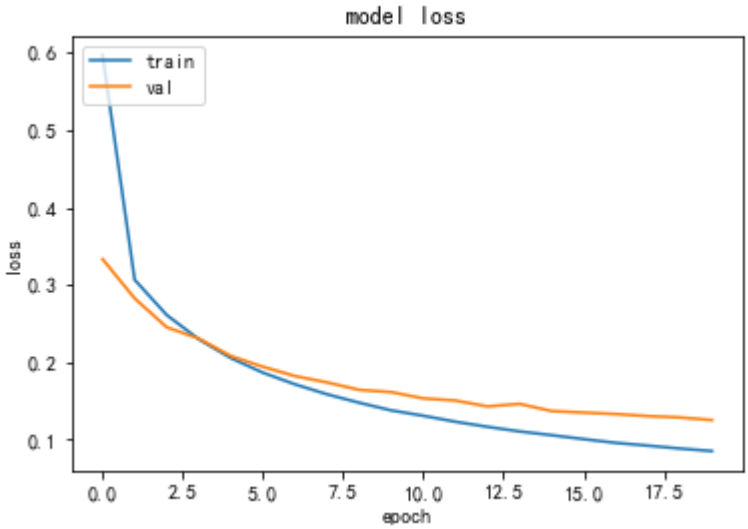
测试集合上进行测试，保存混淆矩阵如下:



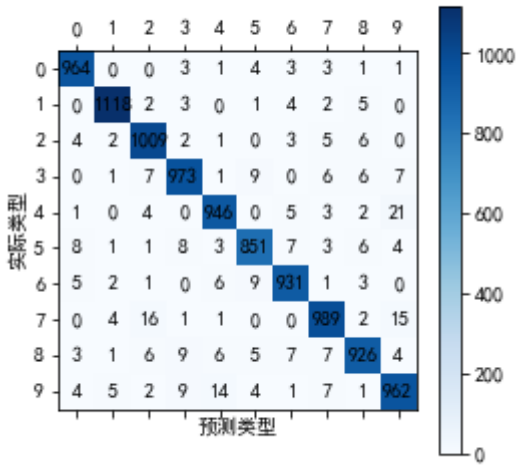
分类正确率为95.48%

(4).隐层节点个数50

训练集误差曲线与验证集误差曲线如下



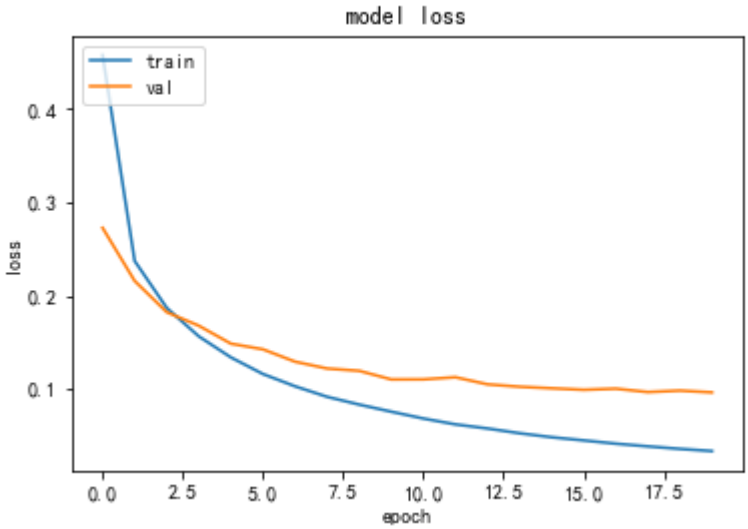
测试集合上进行测试，保存混淆矩阵如下:



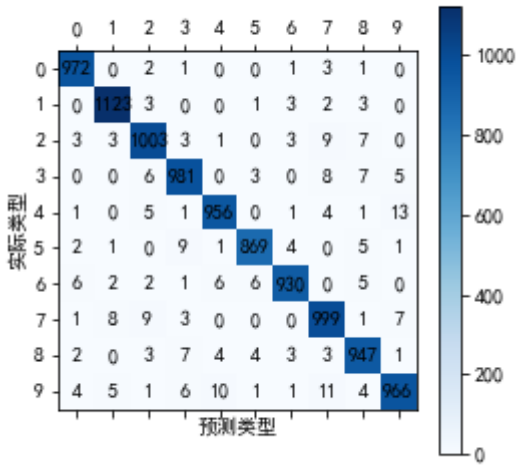
分类正确率为96.69%

(5).隐层节点个数100

训练集误差曲线与验证集误差曲线如下



测试集合上进行测试，保存混淆矩阵如下:



分类正确率为97.46%

(6).实验结果分析

隐层节点数	5	10	20	50	100
accuracy	89.97%	93.11%	95.48%	96.69%	97.46%

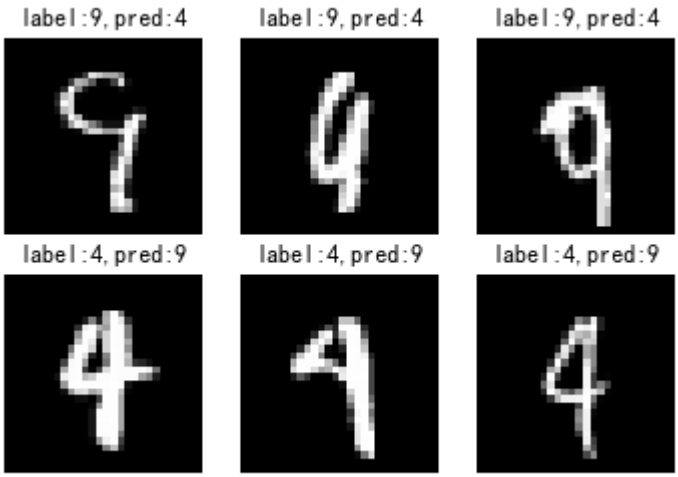
从图中可以看出，随着隐层节点的增多，混淆矩阵的总体准确性提高，分类错误率降低；由训练集和验证集误差曲线可以看出，随着隐层节点的增多，训练误差曲线的下降斜率越大，收敛速度越快；由测试集合准确性可以看出隐层节点越多，测试集合准确率越高。

模型中间隐层节点数越多，模型参数就越多，模型表达训练集特征的能力就越强。因而随着中间隐层节点数的增多，模型收敛速度会加快；不同类型的数据更容易被模型加以区分，分类的错误率会下降。但当隐层节点过多时，由于模型参数过多，可能出现过拟合现象，因而综合考虑训练时间，通常选择满足准确率要求的最小节点数作为隐层节点数。

值得注意的是，由于搭建模型时epochs仅设置为20，训练轮次较少，因而并没有在实验中观察到early stop现象。若进一步增大epoch, 最终将观察到过拟合现象：训练集错误率不断下降，验证集误差却开始上升，模型训练停止。模型参数较多时在训练集上过度学习会使得模型缺乏泛化能力，因而实际使用机器学习算法时需要设置验证集以防止过拟合。

f.分错数字举例

由混淆矩阵可以观察出，数字“4”和“9”最容易被分错，隐层节点为100时，共有10个数字“9”被判别为数字“4”，13个数字“4”被判别为数字“9”，错误示例如下：



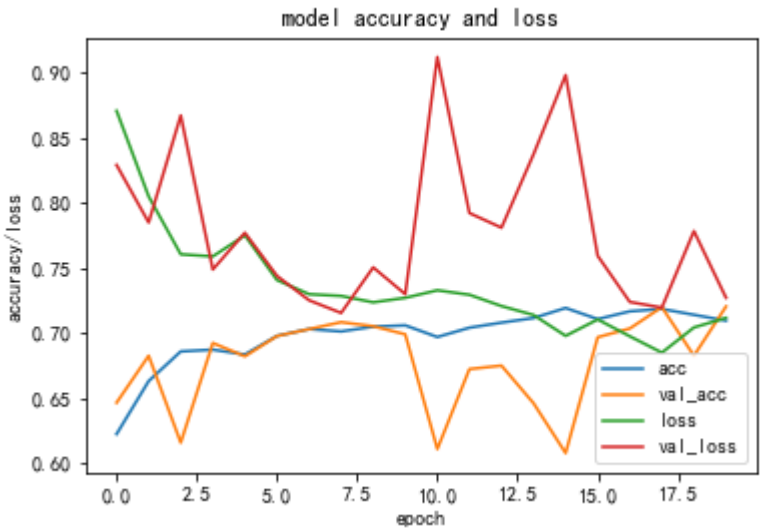
第一排是数字“9”被模型误判为数字“4”的情况，第二排是数字“4”被模型误判为数字“9”的情况。

5.卷积神经网络进行cifar10图片分类

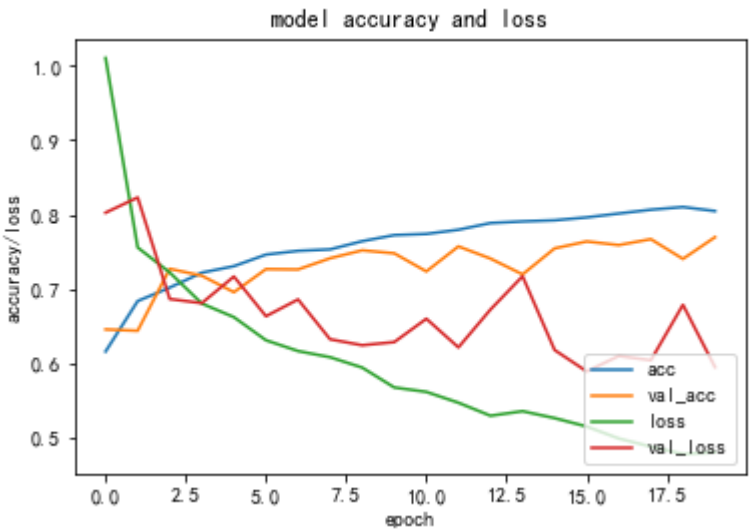
1.全连接神经网络与卷积神经网络比较

全连接神经网络的搭建在‘cifar_dense.py’中，隐层节点为512，具有0,1,2,3个隐层的全连接神经网络的验证集正确率以及验证集loss随训练代数epochs的变化曲线如下：

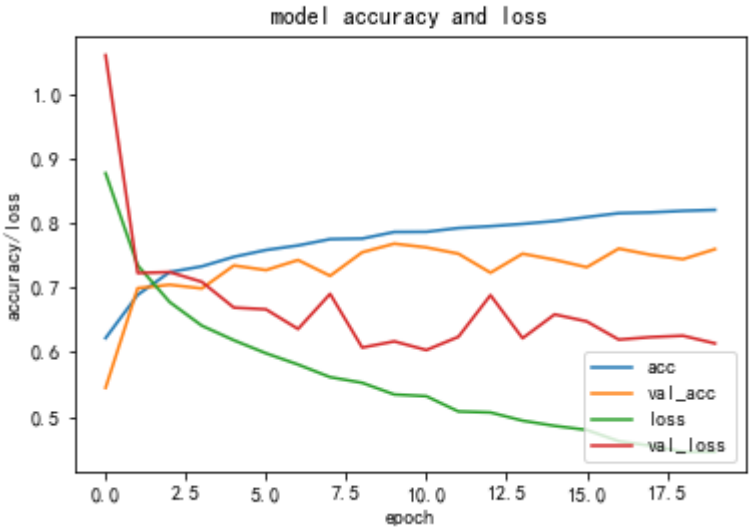
隐层数为0时学习曲线如下，模型参数为9219，测试集合正确率为71.7%：



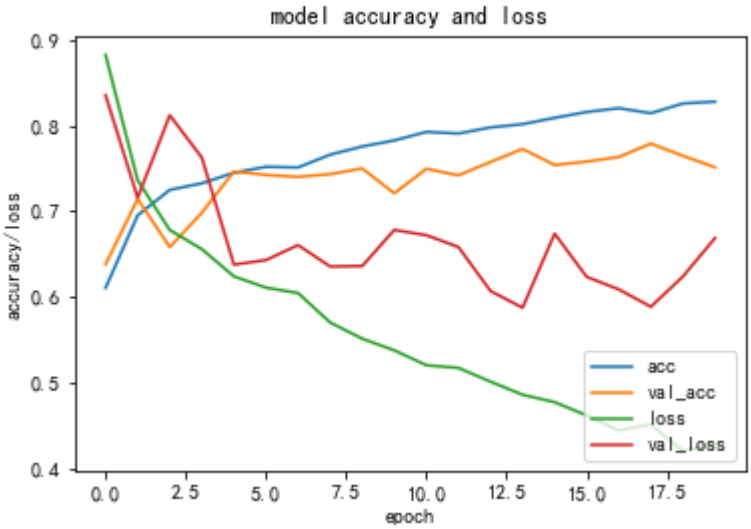
隐层数为1时学习曲线，模型参数为1,574,915，测试集合正确率为77.73%：



隐层数为2时学习曲线，模型参数为1,837,571，测试集合正确率为77.17%：



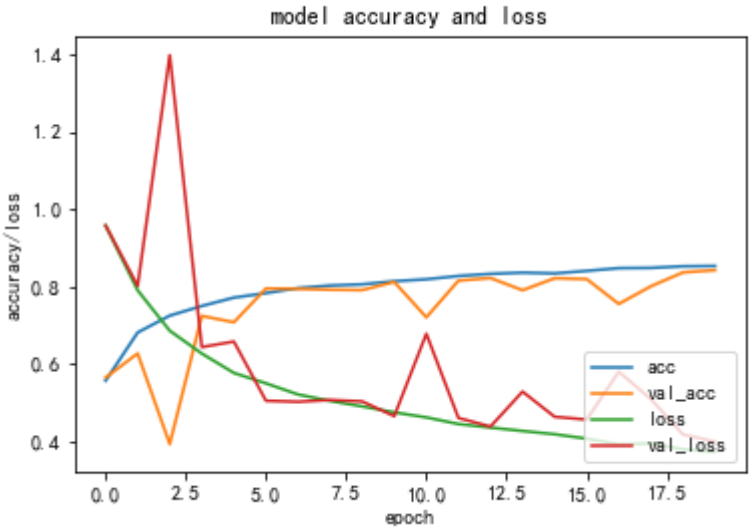
隐层数为3时学习曲线，模型参数为2,100,227， 测试集合正确率为76.6%：



从全连接神经网络可以看出，当不含有隐层时模型的训练效果较差，验证集loss与正确率存在很大波动，模型收敛速度很慢；且由于模型参数过少，模型不能对训练集特征进行充分表达，测试集合正确率较低。

当添加1个隐层时，模型收敛速度明显加快，训练过程中验证集loss与正确率波动明显减小，测试集合的正确率有所提高；然而随着隐层数的进一步增加，当隐层数为2层、3层时，模型在训练集上的收敛速度进一步加快，训练集准确率提高，然而在验证集上的表现却甚至有所下降；训练集与验证集上模型表现的差距进一步增大，说明此时由于参数过多，模型已出现过拟合现象。模型缺乏泛化能力，因而测试集合上正确率反而下降。

‘cifar10.py’对卷积神经网络进行实现，验证集loss和正确率随训练代数epochs的变化曲线如下：



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 30, 30, 64)        1792
_________________________________________________________________
activation_166 (Activation)  (None, 30, 30, 64)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 15, 15, 64)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 15, 15, 64)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 14400)             0
_________________________________________________________________
dense_171 (Dense)            (None, 64)                921664
_________________________________________________________________
activation_167 (Activation)  (None, 64)                0
_________________________________________________________________
dropout_5 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_172 (Dense)            (None, 3)                 195
_________________________________________________________________
activation_168 (Activation)  (None, 3)                 0
=================================================================
Total params: 923,651
Trainable params: 923,651
Non-trainable params: 0
_________________________________________________________________
test accuarcy: 0.858666666667
```

此时模型参数为923,651，明显小于含隐层的全连接神经网络，模型训练速度较快，收敛速度较快；且此时测试集合正确率为85.87%，与全连接神经网络相比有明显提高。

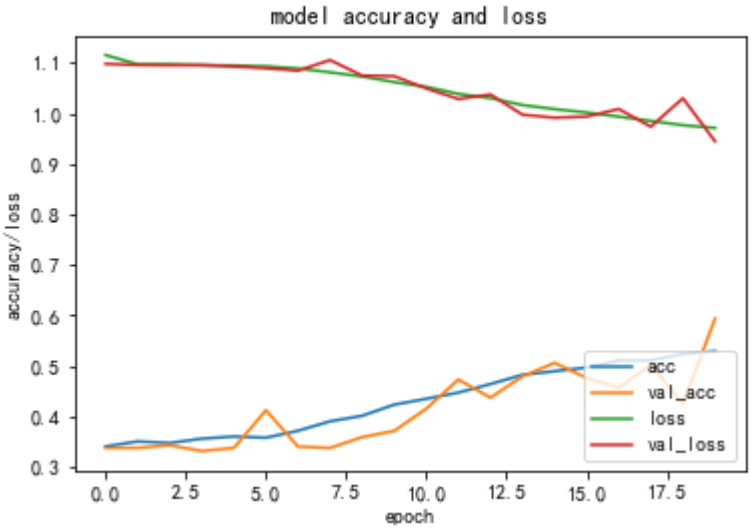
使用卷积神经网络进行训练时，验证集loss与正确率的变化趋势与训练集上基本一致，没有出现过拟合现象。可以预测，随着训练代数的增加，测试集合的正确率进一步上升，卷积神经网络完成图像识别任务时性能更佳。

全连接神经网络完成图像识别任务时存在明显缺点：全连接神经网络参数数量太多，处理较大图像时扩展性很差；网络中每个神经元都和上一层所有神经元相连，没有充分利用像素之间的相对位置信息，学习大量并不重要的权重将导致模型学习非常低效；通过梯度下降方法训练深度全连接神经网络时，全连接神经网络的梯度很难传递超过3层，全连接层过多时反而易出现过拟合现象，导致网络的表达能力将受到限制。

相比之下，卷积神经网络使用卷积核捕捉图像特征，卷积层可以更好地利用像素的相对位置关系，池化层在一定程度上保持原始图像的同时减少了描述对象所需要的参数。卷积神经网络通过尽可能保留重要的参数，去掉大量不重要的参数，达到了更好的学习效果。

2.Sigmoid激活函数比较

将卷积神经网络中间层激活函数换为sigmoid时，验证集loss和正确率随训练代数epochs的变化曲线如下：



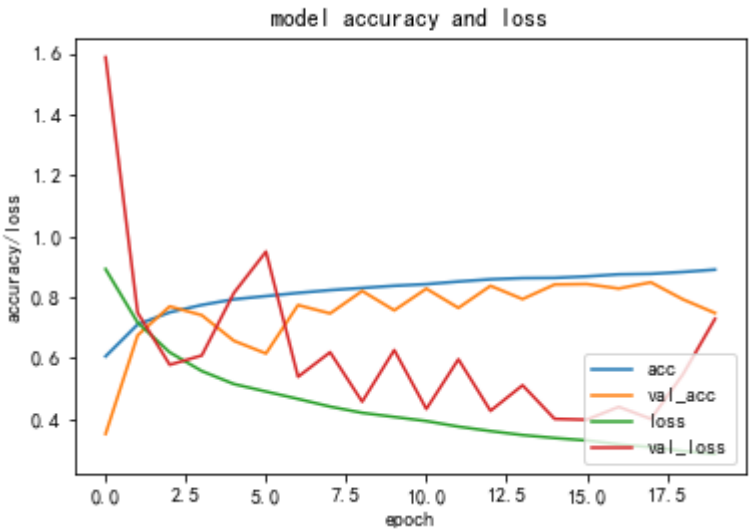
20 epochs后测试集合准确率仅为61.03%，比不含隐层的全连接神经网络更差。

和ReLU激活函数相比，由于sigmoid激活函数计算量大、反向传播求误差梯度时求导涉及除法，因而模型训练速度很慢，每个epoch的训练时间约为使用ReLU时的9倍；且和使用ReLU时相比，使用sigmoid激活函数时模型收敛速度明显较慢；由于模型在训练20 epochs后仍处于明显的欠拟合状态，因而测试集合准确率很低。

此外，由于sigmoid函数本身性质，反向传播时容易出现梯度消失的情况，使得无法完成深层网络的训练，因而实际使用中ReLU或Leaky-ReLU激活函数使用较多。

3.去除Dropout层比较

将卷积神经网络Dropout层除去后，验证集loss和正确率随训练代数epochs的变化曲线如下：



训练 20 epochs后测试集合上准确率为77.2%。

和原卷积神经网络相比，去除Dropout层后模型出现了一定的过拟合现象：训练集loss以较快速率不断下降，而验证集loss却变化缓慢甚至有一定上升（原卷积神经网络未出现明显过拟合现象）；去除Dropout层后，模型在测试集合上的准确率也有所下降。

当模型参数过多时，在训练集上过度学习会使得模型缺乏泛化能力。Dropout层在模型的训练过程中，按照一定的概率将一部分神经网络单元暂时从网络中丢弃。不工作的节点暂时认为不是网络的一部分，但其权重得以保留。由于每次用输入网络样本进行权值更新时，隐含节点都以一定的随机概率出现，因而权值的更新不再依赖于有固定关系隐含节点的共同作用。Dropout 强迫一个神经单元和随机挑选出来的其他神经单元共同工作，从而消除减弱了神经元节点间的联合适应性，增强了模型的泛化能力。因而和去除Dropout层后相比，原神经网络更不易出现过拟合现象，即原神经网络的泛化能力更好。