**《计算机视觉与机器学习》课外实践练习二**

——实现简单的全连接层网络

* 作业截止时间

请于该时间之前提交作业：2023年10月15日，5:00 pm

* 团队合作

允许自由组建团队合作完成本次实践练习（团队人数上限3人）。每个团队共同提交一份作业文件，团队成员共享本次实习成绩。请在Jupyter Notebook(.ipynb)文件对应位置处注明团队各个成员的姓名和学号。

* 考察方式

作业收齐后，将择期开展线上会议（具体时间另行通知，一般在周末），对各团队作业完成情况进行考评，考评人员（老师或助教）将随机指定程序功能或代码，要求进行演示和讲解。团队的**每位成员**都将被要求独立进行演示和回答问题。

本次实习占总成绩比例为20%；团队实习成绩 = 提交作业评分(10%) + 线上考评表现10%)

线上会议室将设等候室，考评过程中各团队将轮流进入会场，团队每位组员将被要求开启摄像头确认身份，故请各位同学开会前调试好设备，并选择适宜的场地。

* 诚信问题

严禁不同团队间代码的拷贝或抄袭，经代码查重检测确定为抄袭的作业，实习成绩将为0分。

# 实习要求

1. **实现基本功能**

1. 全连接的前向传播与反向传播；

2. ReLU的前向传播与反向传播；

3. （选做）Leaky\_ReLU的前向传播与反向传播；

4. （选做）Sigmoid的前向传播与反向传播；

5. svm loss, softmax loss的计算（可参考实习1）；

1. **实现简单多层全连接神经网络,该网络结构为：input - affine - relu - affine – softmax**

1. 以L2正则化的softmax损失结束训练；

2. 实现该网络的损失计算、梯度计算与参数更新；

3. 以python字典的形式储存可模型中可学习的参数W和B；

4. （选做）将relu替换为Leaky\_ReLU或Sigmoid，尝试比较不同激活函数的差异；

5. （选做）选择更深的（或更浅的）全连接隐藏层，尝试比较不同深度的网络的差异；

6. 网络训练的超参数调参；

请仔细阅读以下内容，认真理解其中的提示信息，分步骤依次解决课外实践中的问题，完成代码编写和测试，最后按要求提交作业。

# 全连接层神经网络结构介绍

在本实习中，使用模块化方法实现全连接网络。每一层都要分别实现一个forward和一个backward函数。forward函数将接收输入、权重和其他参数，并返回一个输出和一个cache对象来存储向后传递所需的数据，如下所示:

**def** layer\_forward(x, w):

""" Receive inputs x and weights w """

*# Do some computations ...*

z **=** *# ... some intermediate value*

*# Do some more computations ...*

out **=** *# the output*

cache **=** (x, w, z, out) *# Values we need to compute gradients*

**return** out, cache

反向传播将接收上游的导数和cache对象，并返回相对于输入和权重的梯度，如下所示:

**def** layer\_backward(dout, cache):

"""

Receive dout (derivative of loss with respect to outputs) and cache,

and compute derivative with respect to inputs.

"""

*# Unpack cache values*

x, w, z, out **=** cache

*# Use values in cache to compute derivatives*

dx **=** *# Derivative of loss with respect to x*

dw **=** *# Derivative of loss with respect to w*

**return** dx, dw

在以这种方式实现一堆层之后，我们将能够轻松地将它们组合起来，以构建具有不同架构的分类器。

# **任务一：基本功能的实现**

需要补充完善的文件：layers.py

## 全连接层的前向传播

实现 affine\_forward 功能。完成后需要通过运行以下代码来测试你的实现: 完成目标：difference小于1e-9



结果示例：

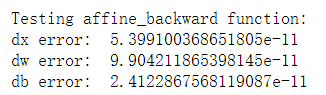


## 全连接层的反向传播

实现 affine\_forward 功能。完成后需要通过运行以下代码来测试你的实现: 完成目标：difference小于1e-9。

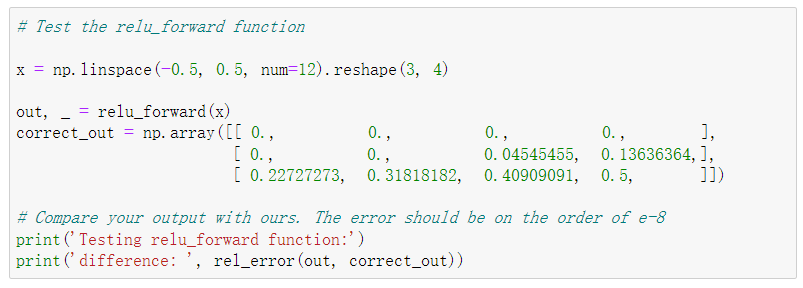


结果示例：



## ReLU激活函数的前向传播

在 relu\_forward 函数中实现ReLU激活函数的前向传递并测试实现。完成目标：相对误差在1e-8量级

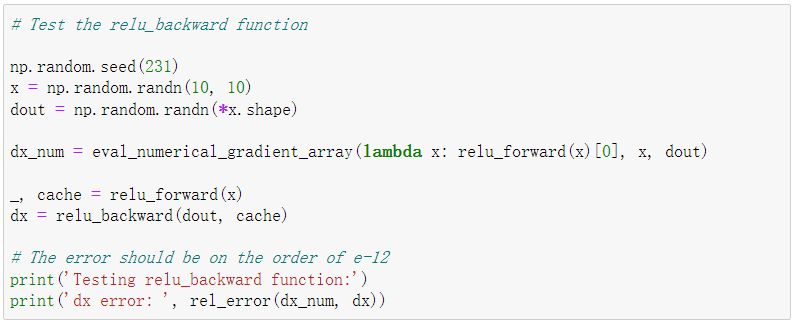


结果示例：



## ReLU激活函数的后向传播

在 relu\_backward 函数中为ReLU激活函数实现后向传播，并测试实现。完成目标：相对误差在1e-12量级



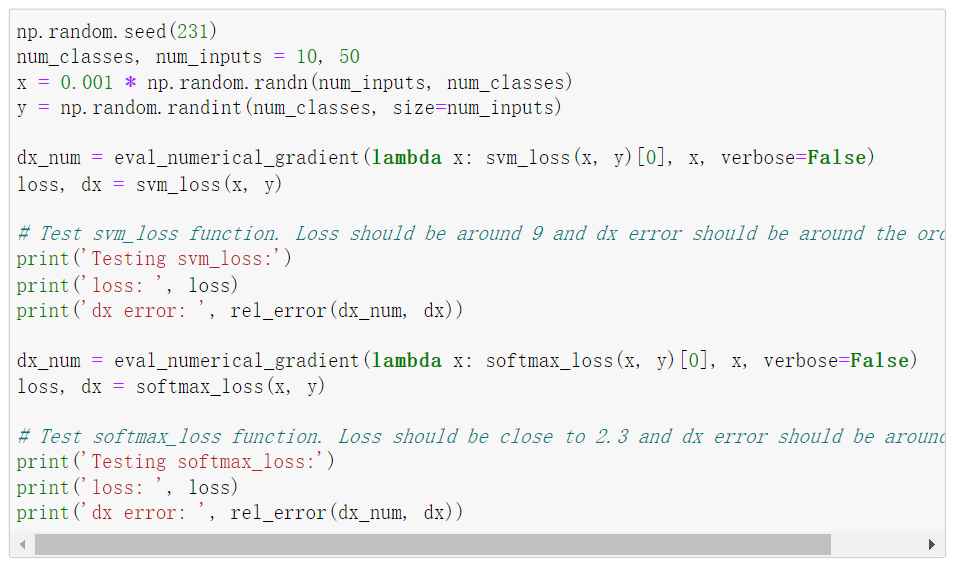
结果示例：



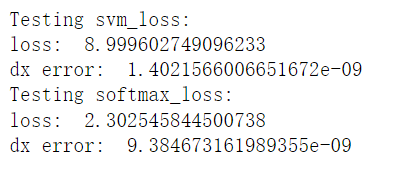
## Softmax和SVM损失层

在 softmax\_loss 和 svm\_loss 函数中实现softmax和SVM的损失和梯度。与实习1中的实现相似。

通过运行以下命令来确保实现正确



结果示例：



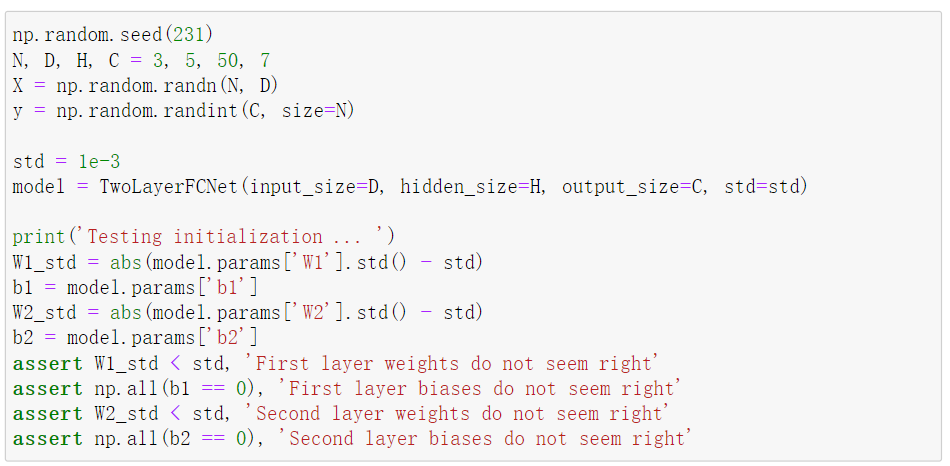
# 任务二：实现简单的两层全连接神经网络

打开文件 two\_layer\_net.py 并补充 TwoLayerFCNet 类的相关代码。然后运行下面的代码块来测试你的实现。

* 此任务需要补充完善的文件：two\_layer\_net.py

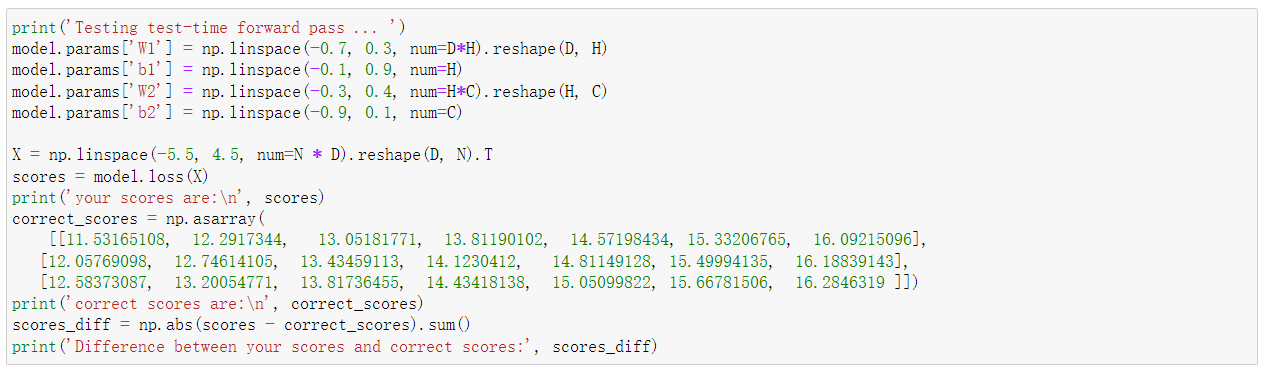
## 初始化网络

测试网络初始化代码的实现，完成目标：网络参数的初始化符合要求。

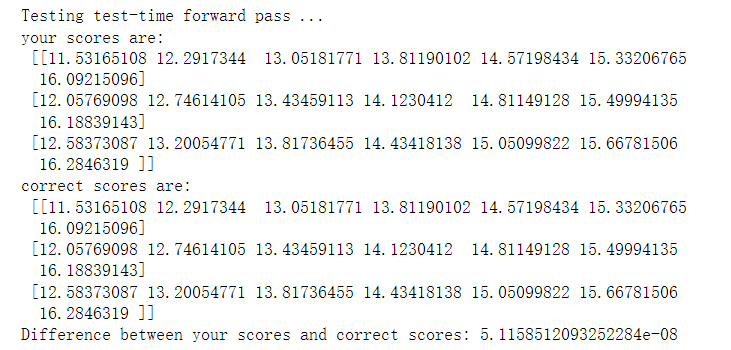


## 前向传播：计算得分

打开文件 fc\_net.py ，查看 TwoLayerFCNet.loss. 这个函数非常类似于SVM和Softmax练习编写的损失函数：获取数据和权重，并计算类分数、损失和参数上的梯度。  
实现正向传递的第一部分，该部分使用权重和偏差来计算所有输入的分数。 完成目标：计算得分与实际得分差异小于1e-6

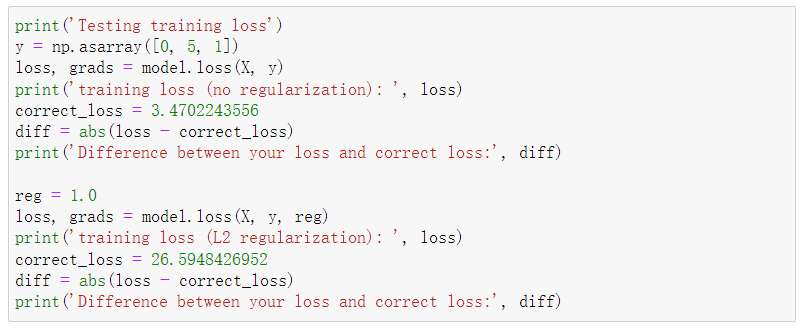


结果示例：

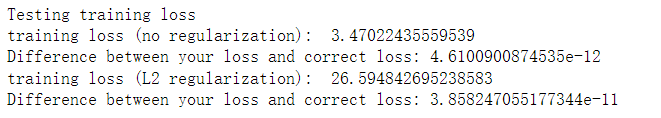


## 前向传播：计算损失

在同一函数中，实现计算数据和正则化损失的第二部分。 完成目标：计算得分与实际得分差异小于1e-6

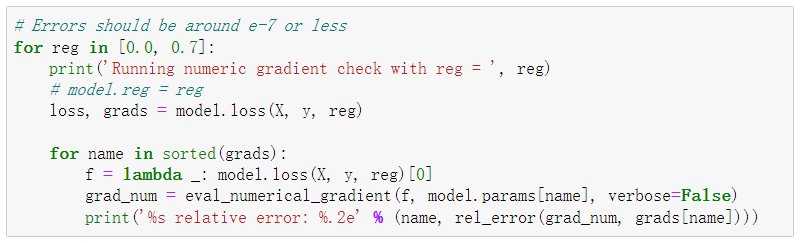


结果示例：

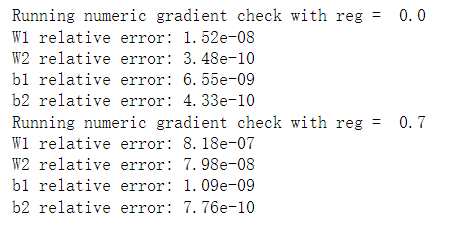


## 反向传播

实现函数的其余部分。计算变量 W1, b1, W2, b2的损失梯度。实现了前向传播后，可以使用数值梯度检查来调试向后传递。 完成目标：误差小于1e-7

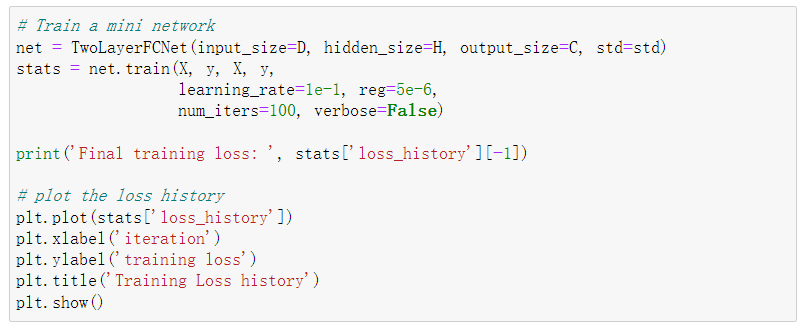


结果示例：



## 训练两层全连接网络

使用随机梯度下降(SGD)训练网络，类似于SVM和Softmax分类器。 完成 TwoLayerNet.train 函数缺失的部分。该过程非常类似于用于SVM和Softmax分类器的训练过程。 实现 TwoLayerNet.predict 函数，当网络进行训练时，训练过程周期性地进行预测，以跟踪精度随时间的变化。 完成目标：Final training loss在1.4左右



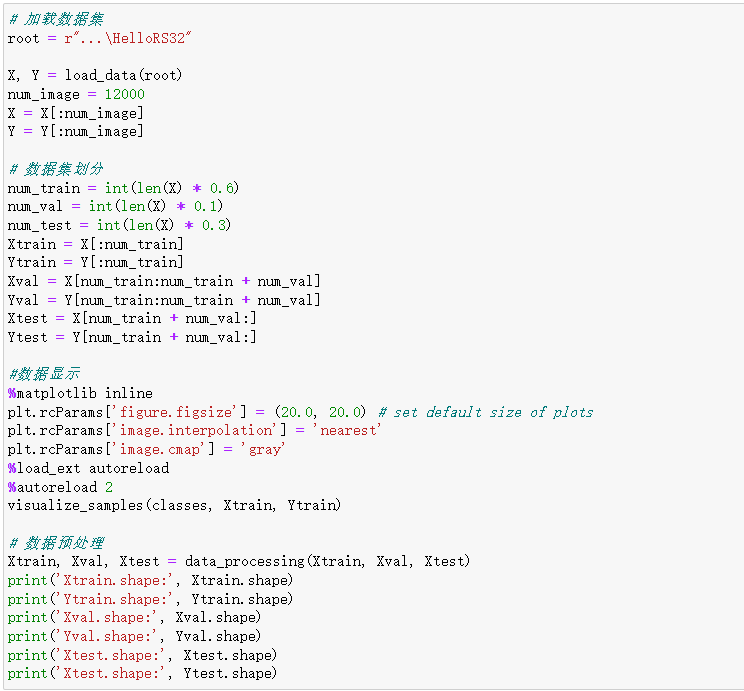
结果示例：



## 在HelloRS32上训练两层全连接网络

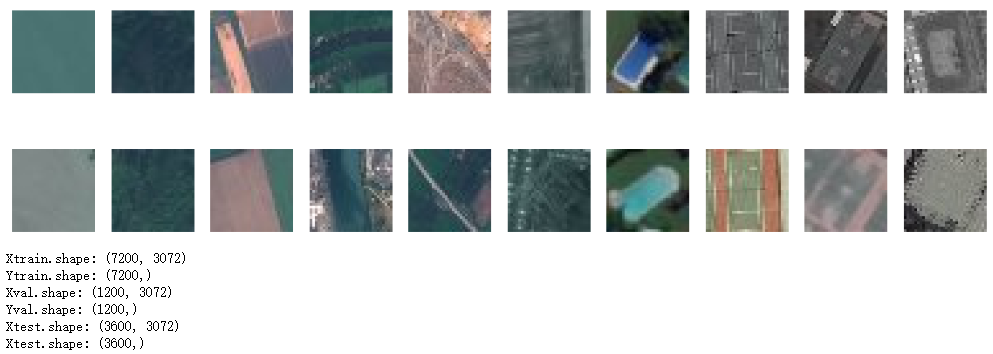
现在，你已经实现了一个通过梯度检查并处理简单数据的两层全连接网络，加载 HelloRS32 数据，使用它来训练一个分类器来处理真实的数据集。

### 加载HelloRS32数据



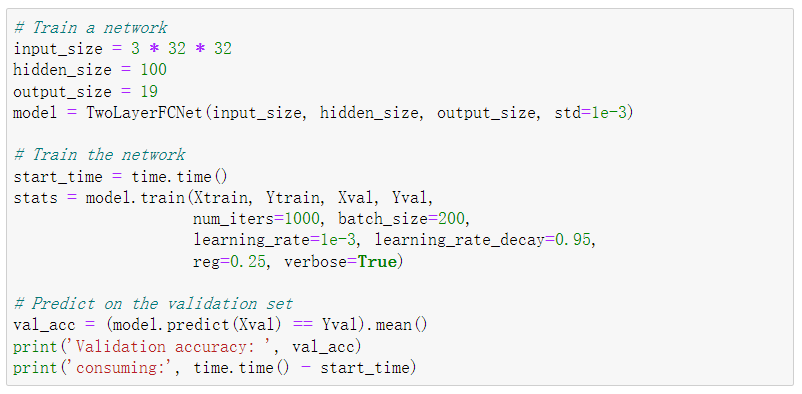
结果示例（下图为HelloRS32的结果）：



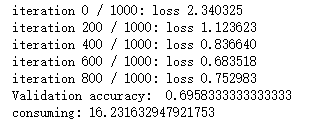


### 训练全连接网络

使用SGD训练网络。 优化时使用指数学习率时间表来调整学习速率; 在每个epoch之后，通过将学习速率乘以衰减速率来降低学习速率。按照如下的参数设置运行，实现目标：val\_acc接近0.69



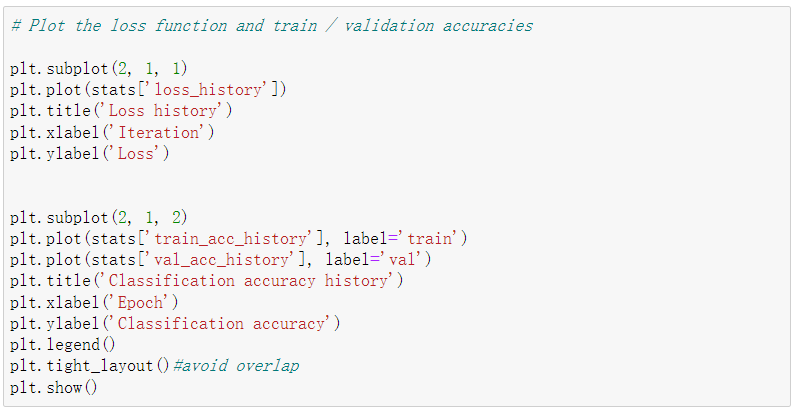
结果示例：



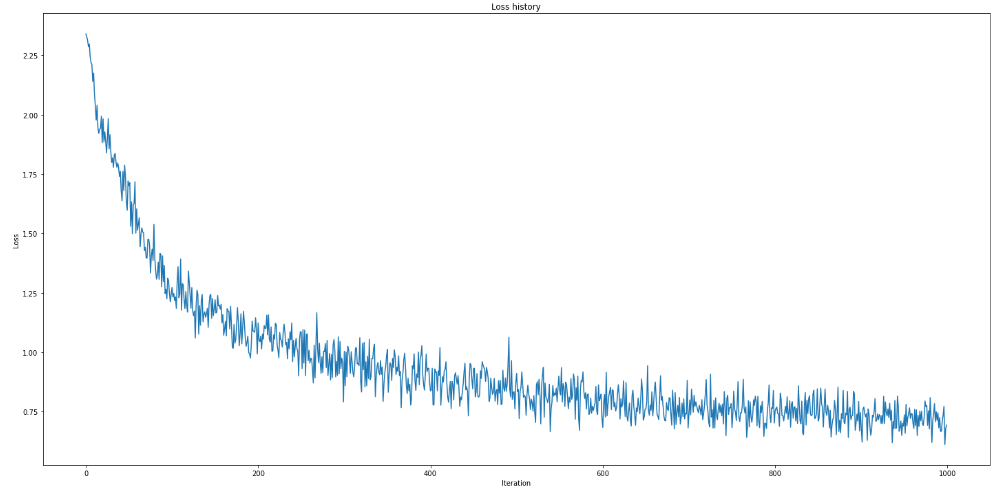
### 调试训练

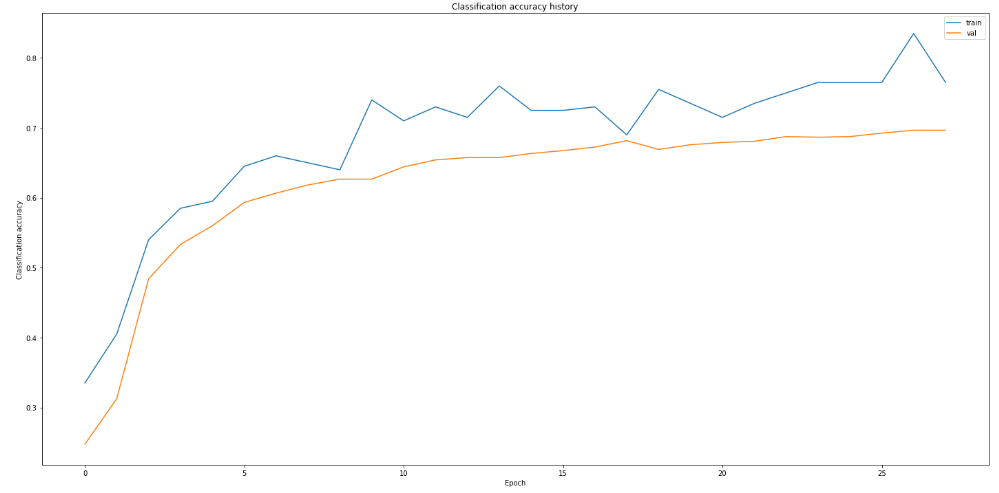
使用上面提供的默认参数，在验证集中获得大约0.29的验证精度。

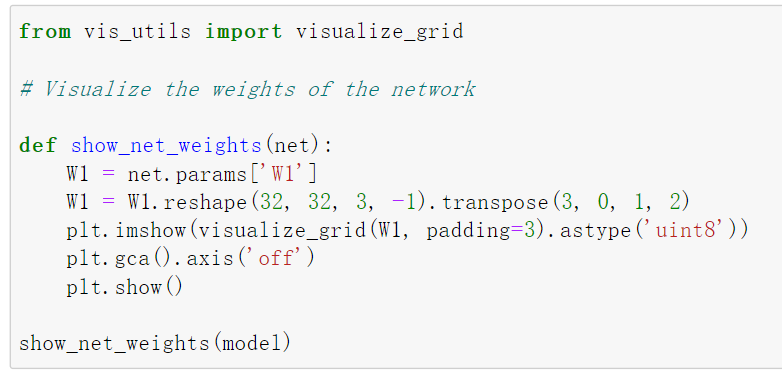
如果想要提高精度，一个方法是在优化过程中绘制损失函数以及训练和验证集的准确性；另一个方法是可视化在网络的第一层训练出的权值。 在大多数以视觉数据为训练对象的神经网络中，第一层权值在可视化时通常呈现出一些可视化的结构。



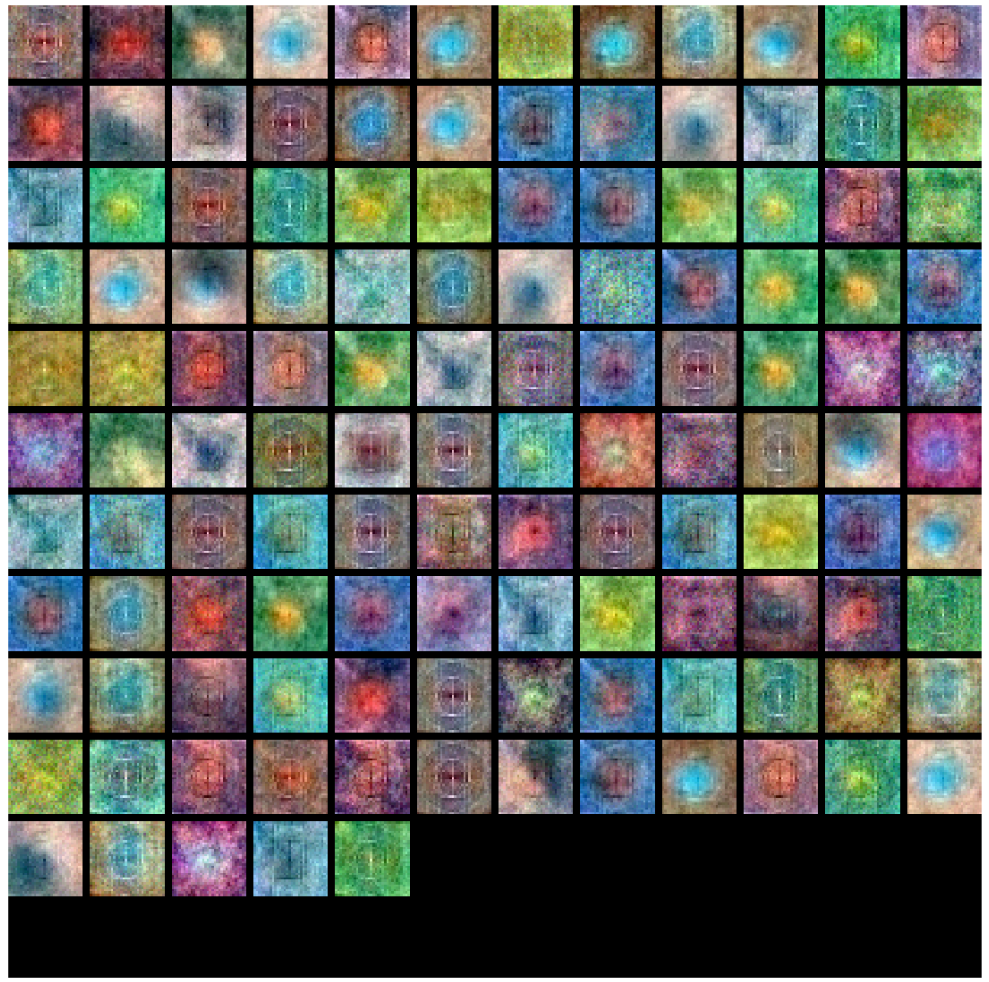
结果示例（下图为HelloRS32示例）：







结果示例：



### 优化超参数

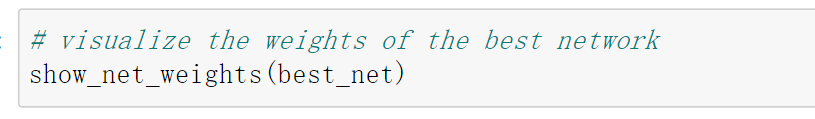
通过观察如上可视化图，我们可以看到损失呈或多或少的线性下降，这似乎表明学习速率可能过低；此外，在训练和验证精度之间没有差距，说明使用的模型容量较低，应该增加它的规模；另一方面，一个非常大的模型会产生更多的过拟合，说明训练和验证准确性之间有较大差距。

调优：调整超参数是使用神经网络的一个重要部分，应该大量练习超参数的调整。接下来，使用各种超参数的不同值进行实验，包括隐藏层大小(hidden\_size)、学习速率(learning\_rate)、训练周期数(num\_iters)和正则化强度(reg)。也可以考虑调整学习速率衰减(learning\_rate\_decay)，但是应该能够使用默认值获得良好的性能。

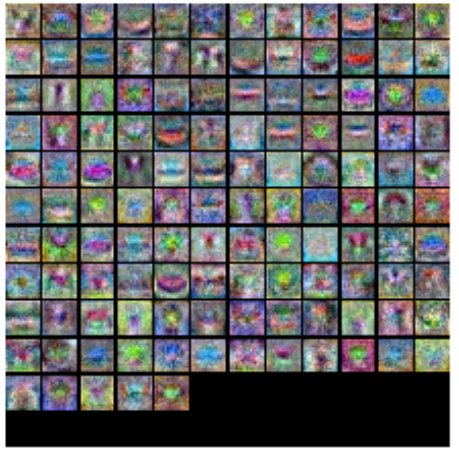
实验：用全连接神经网络在HelloRS32上取得尽可能好的成绩。

结果示例：



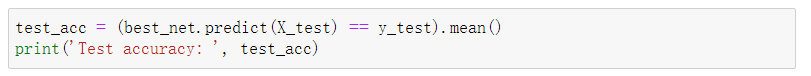


结果示例：



### 在测试集上运行模型

在测试集中运行表现最好的模型。



结果示例：



# 作业提交要求

1. 在Jupyter Notebook下运行fc\_net.ipynb笔记本，首先需在笔记本开头处填写团队各成员的学号和姓名（如下图所示）；完成全部功能和输出相应结果后，将该笔记本内容打印为PDF文件提交，PDF需统一重命名为**队长学号\_队长姓名.pdf**。
2. 实习过程中需要不断更新layers.py和two\_layer\_net.py中的代码，实习完成后，将两个源码文件**合并保存**到一个py文件下。该源码文件需作为作业附件提交用于检查，文件需统一命名为**队长学号\_队员学号\_队员学号.py**。

Text

Description automatically generated with low confidence

A picture containing text

Description automatically generated