Machine Learning

Lab 2: Neural Network

说明

- 无需完成所有实验，但在课上尽可能多地完成
- 实验结束后不需要提交材料
- 如有疑问可向助教提出，或与同学讨论

实验要求

- 利用Python神经网络库Keras完成以下任务：
  - 搭建简单的神经网络，输出网络的架构图
  - 利用该网络在mnist数据集上完成图像分类，获得分类结果
  - 调整网络以及训练参数，尝试让分类结果提升
  - 可视化训练曲线
  - 搭建VGG 16网络在Cifar10数据集上进行图像分类*

实验数据

- 实验采用 Keras 库自带的mnist手写数字识别数据集，训练集共 60000 张图像和标签，测试集共 10000 张图像和标签。图片为28*28像素点的0 ~ 9的灰质手写数字图片。
- 数据集已划分为训练数据（X_train, y_train）与验证数据（X_test, y_test），并已设置为全局变量，无需自行划分。可关注此处的数据处理实践，如将标签转化为one-hot向量的步骤。

实验环境

- 实验在python环境下运行，主要的库为keras, numpy, matplotlib等
- 实验1~5在CPU上运行，条件允许时，实验6可在GPU上运行
- 在Lab2.py的基础上进行代码的填充与改动，注意代码中的注释
- Keras文档：https://keras.io/
- Numpy文档：https://numpy.org/doc/stable/
- Matplotlib文档：https://matplotlib.org/stable/tutorials/index.html

实验1

- **任务1：构建网络**
- 在代码的如下部分构建你的网络

```
'''第二步：构建网络层'''
# 在此处构建你的网络
################################################################################################
################################################################################################
```

- 阅读文档或查阅资料，查看keras中的网络层、激活函数等应该如何使用。

实验1

- 一个简单的网络示例，只包含了全连接层

```
'''第二步：构建网络层'''
# 在此处构建你的网络
#################################################################################################
model.add(Input((28,28,1)))

model.add(Flatten())

# 1000个神经元的全连接层
model.add(Dense(500))
model.add(Activation('tanh'))

model.add(Dense(500))
model.add(Activation('tanh'))

model.add(Dense(10)) # 输出结果是10个类别，所以维度是10
model.add(Activation('softmax')) # 使用softmax转换为概率
#################################################################################################
```

实验1

- **注意**
- 为了完成分类，网络最后两行代码是固定的（即一个10个神经元的全连接层与一个softmax函数）
- X_train和X_test可以直接输入卷积层，但需要flatten以后才能输入全连接层
- 不要遗漏Input层（对训练可能无影响，但会影响下一步的网络结构图输出）

实验1

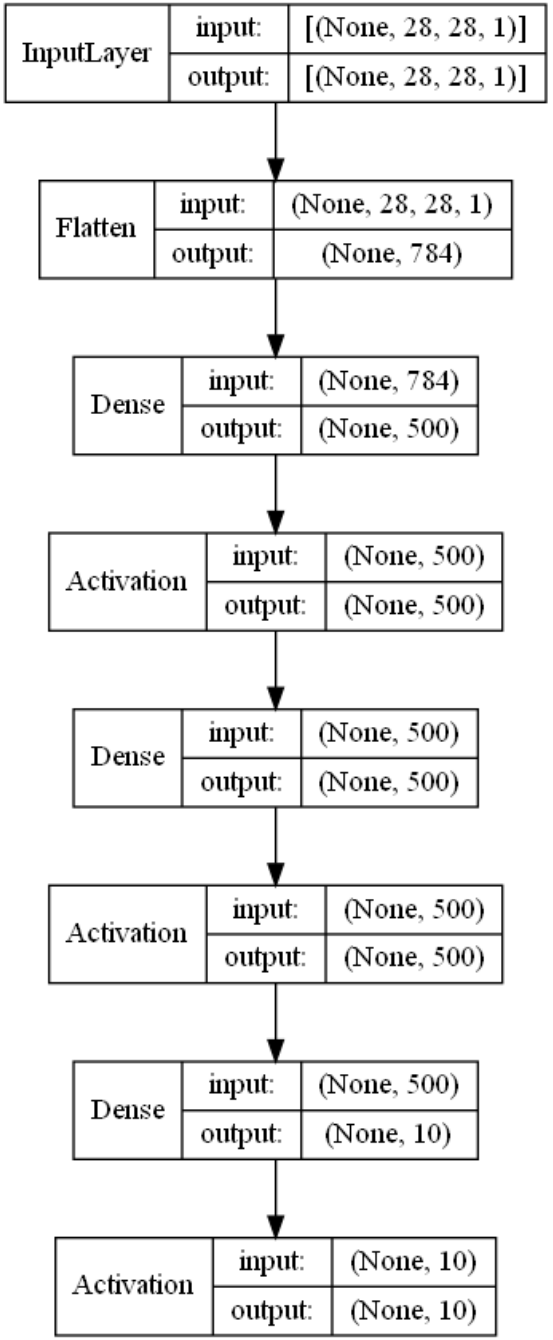
- **任务2：输出网络架构图**
- 以下代码将输出一张png格式的网络架构图

```
# 在此处输出网络的架构。此处参数可以不用调整。
# model表示自定义的模型名 to_file表示存储的文件名 show_shapes是否显示形状  rankdir表示方向T(top)B(Bottow)
from keras.utils.vis_utils import plot_model
plot_model(model,to_file='model.png',show_shapes=True,show_layer_names=False,rankdir='TB')
```

- 为实现该功能，需要安装pydot-ng(或pydotplus)和graphviz。前者通过pip安装，后者需要在 Download | Graphviz 进行下载。安装graphviz时注意选择将路径添加到环境变量的选项。

实验1

- 示例：前述网络的网络架构图



实验2

- **任务：训练网络，获得测试集上的分类准确率**
- 网络模型搭建好后，如无错误，直接运行代码即可开始训练。不更改Lab2.py中设置的情况下应该有如下输出格式：

```
Epoch 1/50
329/329 - 3s - loss: 0.4510 - accuracy: 0.8700 - val_loss: 0.3164 - val_accuracy: 0.9096
Epoch 2/50
329/329 - 2s - loss: 0.2832 - accuracy: 0.9181 - val_loss: 0.2692 - val_accuracy: 0.9234
Epoch 3/50
329/329 - 2s - loss: 0.2443 - accuracy: 0.9296 - val_loss: 0.2482 - val_accuracy: 0.9277
Epoch 4/50
```

- 训练完成后将进行测试，并输出准确率，格式为：

```
test set
79/79 [==============================] - 0s 4ms/step - loss: 0.0767 - accuracy: 0.9772
The test loss is 0.076728
The accuracy of the model is 0.977200
```

实验3

- **任务：调整网络与训练设置，争取获得更高的分类准确率**
- 可按照以下思路进行调整
- 网络
  - 增加网络深度：加一些层
  - 使用卷积层，BatchNorm等
  - 使用Dropout缓解过拟合
  - 更换激活函数：ReLu等
  - 卷积、全连接层的参数调整：不同卷积核个数/大小，不同神经元数量
- 训练设置
  - 优化器：SGD、Adam等
  - 损失函数：交叉熵、MSE等
  - 学习率：初始值，decay值（衰减速率）
  - 训练的Epochs数
  - 训练的Batchsize

实验3

- **小提示**
- 不要修改随机数种子，否则结果可能产生较大幅度波动
- Epochs需要根据训练情况进行设置，至少要让loss收敛，但也不要大到出现严重的过拟合
- 能否使用多种方式让测试集准确率超过99%？

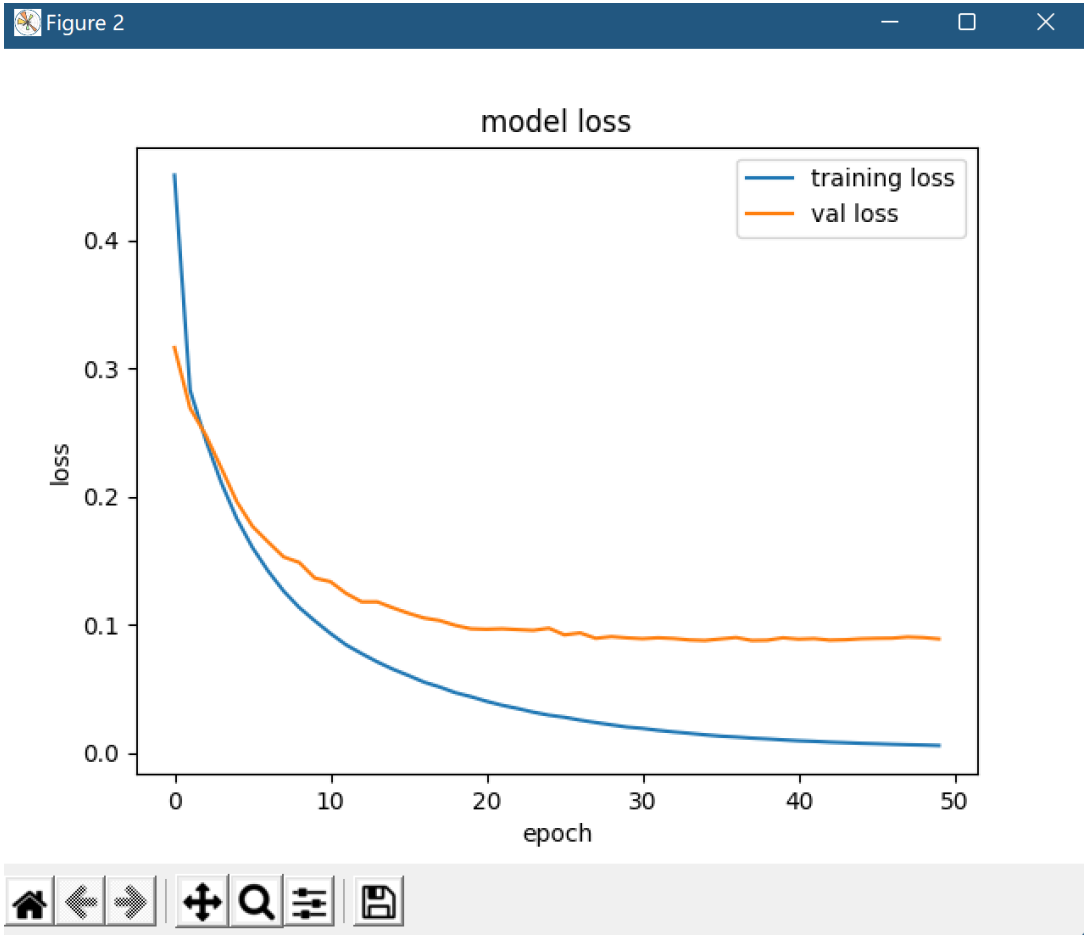
实验4

- **任务：可视化训练曲线**
- 可视化训练时的一些中间结果能帮助做出判断，如是否过拟合
- 本实验需要大家可视化loss曲线和accuracy曲线
- 曲线含义
  - 横坐标：epoch
  - 纵坐标：train/val集合上的loss/accuracy值
- 在指定位置补充代码

```
# 在此处实现你的可视化功能
################################################################################
################################################################################
```

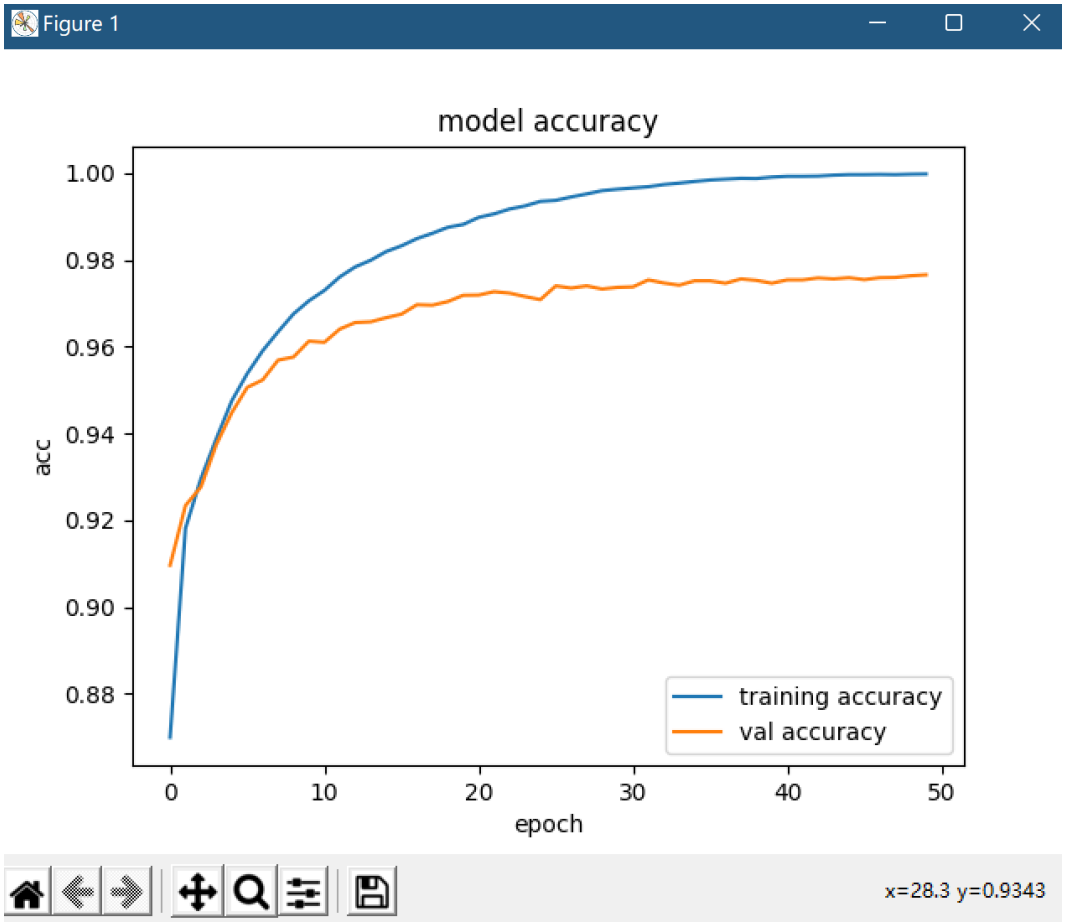
实验4

- loss曲线示例



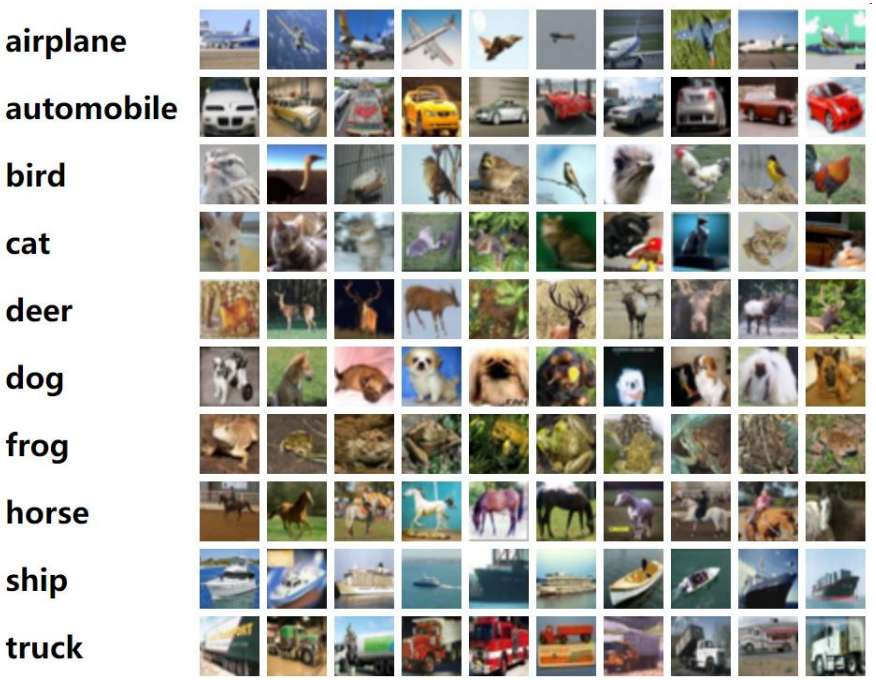
实验4

- accuracy曲线示例



实验5（拓展实验）

- **Cifar10数据集**
- CIFAR-10 是一个用于识别普适物体的小型数据集。一共包含 10 个类别的 RGB **彩色图 片（3通道）**。图片的尺寸为 32×32，数据集中一共有 50000 张训练图片和 10000 张测试图片。其比mnist数据集更加复杂。



实验5（拓展实验）

- **任务：在Cifar10数据集上进行网络训练**
- 自行进行数据处理（Keras自带Cifar10库）
- 首先将你之前调好的在mnist数据上表现最好的网络及相关设置应用在Cifar10上进行分类。这一设置能取得和mnist上一样好的结果吗？
- 按照mnist上调参的思路，对你的网络进行调整，尽可能使得在Cifar10测试集上获得更高的准确率。

实验6（拓展实验）

- **VGG**
- VGGNet 是由牛津大学视觉几何小组（Visual Geometry Group, VGG）提出的一种深层卷积网络结构，以 7.32% 的错误率赢得了 2014 年 ILSVRC 分类任务的亚军，以 25.32% 的错误率夺得定位任务的第一名。
- 论文地址：https://arxiv.org/pdf/1409.1556.pdf

实验6（拓展实验）

- **VGG架构**

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

实验6（拓展实验）

- **任务：按照VGG的思路构建网络，完成Cifar10上的分类**
- 网络应该具有至少8个卷积层（例如，使用VGG 19的第3和第4个stage）
- 由于Cifar10图像尺寸的限制，你应该使用不超过两次max pooling
- 你之前调好的网络在Cifar10上的分类结果好于新构建的网络吗？

实验6（拓展实验）

- **小提示**
- 本实验建议在有Nvidia的GPU的电脑上进行
  - 显存允许，则可以尝试以较大的Batchsize运行
  - 显存不允许，则不断减小Batchsize（一般需要是2的倍数）
- 如果使用的设备只能使用CPU训练，可尝试切分出一小部分数据集进行训练，以免耗费太多时间
- 使用GPU运行时，请注释掉以下两行代码：

```
import os
os.environ["CUDA_VISIBLE_DEVICES"] = ""
```

如果你已完成了全部……

- 尝试新的数据集：Cifar100（Keras自带）
- 尝试不断加深你构建的类VGG网络的深度，再观察准确率。你有什么发现？