LAB01：Classification

小组编号：

小组成员 1：孙士晨

小组成员 2：李兵磊

小组成员 3：董祥虎

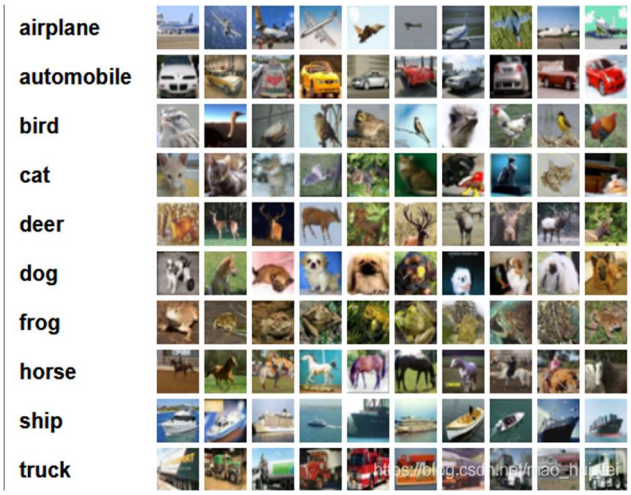
小组成员 4：聂　尧

姓名	学号	分工	占比
孙士晨			25%
李兵磊			25%
董祥虎			25%
聂　尧			25%
总　计			100%

1. 数据准备

1.1. 数据集的获取

本次实验的数据集为多伦多大学的机器学习数据库，该数据集是机器学习领域常用的标准数据库之一。课题使用的数据集为图片数据集，包含 60000 张彩色图片。具体样例请参看右图所示：



获取的方式有通过官网（http://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz）下载到本地和通过 Python 的 torchvision.datasets.CIFAR10 下载两种方式。本次实验采取的方式是将数据集下载到本地使用，能够保证实验快速准确，不依赖网络而运行。

1.2. 数据预处理

为方便后续的代码编写和阅读，我们首先简化数据集的标签类别，将“airplane”、“bird”、“dog”等标签用 0-9 数字代表。

```
nb_classes = 10
class_name = {
    0: 'airplane',  1: 'automobile',    2: 'bird',    3: 'cat',
    4: 'deer',      5: 'dog',           6: 'frog',    7: 'horse',
    8: 'ship',      9: 'truck',
}
```

1.3. 通过 PCA 和白化转换图片数据

利用 PCA 对原有的数据进行降维与白化预处理。选择保持主成分 97%，及保留了数据的主要特征，同时将数据的特征维度从 3072 降到了 400。大大提高了算法的训练时间，减少了内存消耗。

同时再对降维后的数据进行白化处理，去掉数据之间的关联度，减少冗余信息量，有助于提升训练及预测准确率。数据的白化必须满足两个条件：

- 是不同特征间相关性最小，接近 0；
- 二是所有特征的方差相等（不一定为 1）。常见的白化操作有 PCA whitening 和 ZCA whitening。在本实验中采用的是 PCA whitening。

```
from sklearn.decomposition import PCA
# whiten 参数控制是否进行白化操作
pca = PCA(n_components=250,whiten=True)
pca.fit(X,y)
```

1.4. 深度学习构造批次

CIFAR10 数据集有 50000 张训练图片，10000 张测试图片。设置迭代次数为 200 epochs，现在分别选择 Batch Size = [64,128,256] 对模型进行训练。在后续章节进行结果对比分析。

2. 数据清洗

2.1. 异常点清洗

高斯滤波是一种线性平滑滤波，适用于消除高斯噪声，高斯滤波就是对整幅图像进行加权平均的过程，每一个像素点的值，都由其本身和邻域内的其他像素值经过加权平均后得到。高斯滤波的具体操作是：用一个模板（或称卷积、掩模）扫描图像中的每一个像素，用模板确定的邻域内像素的加权平均灰度值去替代模板中心像素点的值。

```
#### 高斯滤波
import cv2
Gaussian_filter_img = cv2.GaussianBlur(img, (3,3), 0)
```

2.1. 划分训练集和测试集

然后，我们将数据集分为 2 个集合——训练集和测试集。训练集和测试集又分别按照自变量和因变量分开。

```
    X_train, X_test, y_train, y_test = train_test_split
                                (X, y, test_size=0.2, random_state=2021)
```

2.2. 标准归一化

之后，我们对训练集和测试集的因变量均进行了标准归一化，可以加快梯度下降求解的速度。在本实例中，每一个数据是 32×32×3 的范围在 0-255 的整数值，先将 RGB 三通道的像素值各减去各通道像素的平均值，再进行归一化。

```
# 平均化，将像素值各减去三通道的平均值
def mean_subtraction(data):
    [mean_r, mean_g, mean_b] = np.mean(data, axis=(0,1,2))
    data[:, :, :, 0] -= mean_r
    data[:, :, :, 1] -= mean_g
    data[:, :, :, 2] -= mean_b
```

```
# 归一化
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
mean_subtraction(X_train)
mean_subtraction(X_test)
X_train /= 255.0
X_test /= 255.0
```

2.3. 特征工程

2.3.1. 灰度处理

在传统图像处理部分，进行先对图像进行灰度处理，根据重要性及其它指标，将三个分量以不同的权值进行加权平均。由于人眼对绿色的敏感最高，对蓝色敏感最低，因此，按下式对RGB三分量进行加权平均能得到较合理的灰度图像。

$$\mathrm{Gray}(i,j) = 0.2989 * \mathrm{R}(i,j) + 0.5780 * \mathrm{G}(i,j) + 0.1140 * \mathrm{B}(i,j)$$

2.3.2. 图像增强

图像增强技术通过对训练图像进行一系列随机更改以生成相似但不同的训练示例来扩展训练数据集的规模。并且随机更改一些示例可以有效的减小模型对某些属性的依赖，提高模型的泛化能力：通过不同的方式裁剪图像，使感兴趣对象出现在不同位置，减小模型对位置的依赖。通过对图片颜色的调节，降低模型对颜色的敏感度。

在本实例中采取的图像增强的方法有三种

- 翻转：左右翻转图像通常不会更改对象的类别。这是最早且使用最广泛的图像增强方法之一。

- 剪切：通过随机剪切图像可以使对象以不同的比例出现在图像的不同位置。可以降低模型对目标对象位置的依赖，从而提高泛化能力。

- 改变图像颜色：通过颜色增强，修改图像的亮度、对比度、饱和度、色相等来更改图像的颜色。

```
from keras.preprocessing.image import ImageDataGenerator
aug = ImageDataGenerator(   # 设置图像增强的超参数
            width_shift_range = 0.125,   height_shift_range = 0.125,
            horizontal_flip = True,      brightness_range = 0.125,
            rotation_range = 15
             )
aug.fit(X_train)
```

```
gen = aug.flow(X_train, y_train, batch_size=batch)
```

3. 模型搭建

在这部分，着重介绍传统方法的模型搭建和深度学习模型搭建。传统回归问题采用逻辑分类，决策树，随机森林，KNN 四种；深度学习模型从最基本的 CNN 模型，到深度 CNN 模型，再到基于 VGG-16 的网络模型。下面简要介绍这些模型的数学形式、原理、优缺点及搭建过程。

3.1. 逻辑回归分类

3.1.1. 原理与数学形式

逻辑回归（Logistic Regression）是用于处理因变量为分类变量的回归问题，常见的是二分类或二项分布问题，也可以处理多分类问题，它实际上是属于一种分类方法。

分类器的函数形式为 Sigmoid 函数的向量模式：

$$\text{sigmoid} = \frac{1}{1 + e^{-W^T x}}$$

其中 W 是最优分类系数的列向量，$W = [w_1, w_2, w_3, \dots, w_n]^T$。

3.1.2. 特点

- 优点：计算代码不多，易于理解和实现，计算代价不高，速度快，存储资源低。
- 缺点：容易欠拟合，分类精度可能不高。
- 适用数据类型：数值型和标称型数据。

3.1.3. 搭建过程

调用 sklearn 库函数即可建立新型分类模型。

```
from sklearn.linear_model import LogisticRegression
linreg = LogisticRegression(verbose=True, n_jobs=-1)
linreg.fit(X, y)
```

3.2. 决策树分类

3.2.1. 原理与数学形式

决策树的输入训练数据是一组带有类别标记的样本，构造结果是一棵多叉树。树的分支节点一般表示为一个逻辑判断。某一节点上选择特征的标准是在该节点上选取能对该节点处的训

练数据进行最优划分的属性。划分的标准是信息增益（Information Gain），即划分前后数据集的熵的差异。取能带来最大信息增益的那个特征作为当前划分标准。

3.2.2. 特点

- 优点：效率高，在相对短的时间内能够对大型数据源做出可行且效果良好的结果。
- 缺点：容易过拟合、对连续性字段比较难预测。

3.2.3. 搭建过程

调用 sklearn 库函数即可建立新型分类模型。

```
from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor()
```

3.3. 随机森林分类

3.3.1. 原理与数学形式

随机森林属于 Bagging 类算法，而 Bagging 方法又属于集成学习一种方法，集成学习的大致思路是训练多个弱模型打包起来组成一个强模型，强模型的性能要比单个弱模型好很多，其中的弱模型可以是决策树、SVM 等模型，在随机森林中，弱模型选用决策树。

在训练阶段，随机森林采样从输入训练数据集中采集多个不同的子训练数据集来依次训练多个不同决策树；在预测阶段，随机森林将内部多个决策树的预测结果取平均得到最终的结果。

3.3.2. 特点

- 在当前所有算法中，具有极好的准确率；
- 能够有效地运行在大数据集上；
- 能够处理具有高维特征的输入样本，而且不需要降维；
- 能够评估各个特征在分类问题上的重要性。

3.3.3 搭建过程

调用 sklearn 库函数即可建立新型分类模型。

```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(n_estimators=100)
```

3.4. KNN 分类

3.4.1. 原理与数学形式

在本例中，就是比较 32x32x3 的像素块。最简单的方法就是逐个像素比较，最后将差异值全部加起来。换句话说，就是将两张图片先转化为两个向量 I_1 和 I_2，然后计算他们的 L1 距离或者 L2 距离：

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p| \quad 或 \quad d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

3.4.2. 特点

- 思路清晰，易于理解，实现简单；
- 算法的训练不需要花时间，因为其训练过程只是将训练集数据存储起来；
- 但测试需花费大量时间计算，因为每个测试图像都需和所有存储的训练图像进行比较。

3.4.3 搭建过程

调用 sklearn 库函数即可建立新型回归模型。

```
from sklearn.neighbors import KNeighborsClassifier
KNN =KNeighborsClassifier(n_neighbors=10)
KNN.fit(X, y)
```

3.5. CNN 分类

3.5.1. CNN 网络的步骤

数据输入层

该层要做的处理主要是对原始图像数据进行预处理，其中包括：

- 去均值：把输入数据各个维度都中心化为 0。
- 归一化：幅度归一化到同样的范围。
- PCA/白化：用 PCA 降维；白化是对数据各个特征轴上的幅度归一化。

卷积计算层

在这个卷积层，有两个关键操作：

- 局部关联。每个神经元看做一个滤波器(filter)

- 窗口(receptive field)滑动， filter 对局部数据计算

激励层

把卷积层输出结果做非线性映射。CNN 采用的激励函数一般为 ReLU(The Rectified Linear Unit/修正线性单元)，它的特点是收敛快，求梯度简单，但较脆弱。

池化层

池化层夹在连续的卷积层中间，用于压缩数据和参数的量，减小过拟合。下面是池化层的具体作用:

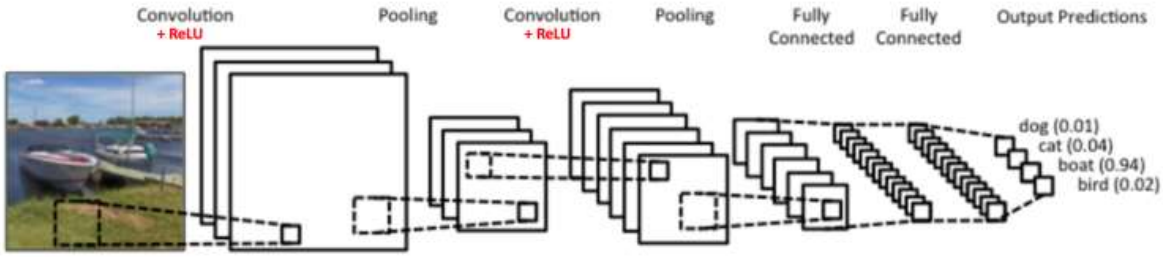
- 维持特征不变性，仅留下的最能表达图像特征的信息。
- 特征降维，去除冗余信息，提取最重要的特征。
- 在一定程度上防止过拟合，更方便优化。

全连接层

两层之间所有神经元都有权重连接，通常全连接层在卷积神经网络尾部。

3.5.2. 基础 CNN 网络模型

下面是我们使用的最简单的 CNN 模型结构，可以看出最左边的船的图像就是我们的输入层，计算机理解为输入若干个矩阵。接着是卷积层（Convolution Layer）。卷积层的激活函数使用的是 ReLU。在卷积层后面是池化层(Pooling layer)，池化层没有激活函数。卷积层+池化层的组合可以在隐藏层出现很多次，下图中出现两次。在若干卷积层+池化层后面是全连接层（Fully Connected Layer, 简称 FC），输出层使用了 Softmax 激活函数来做图像识别的分类。

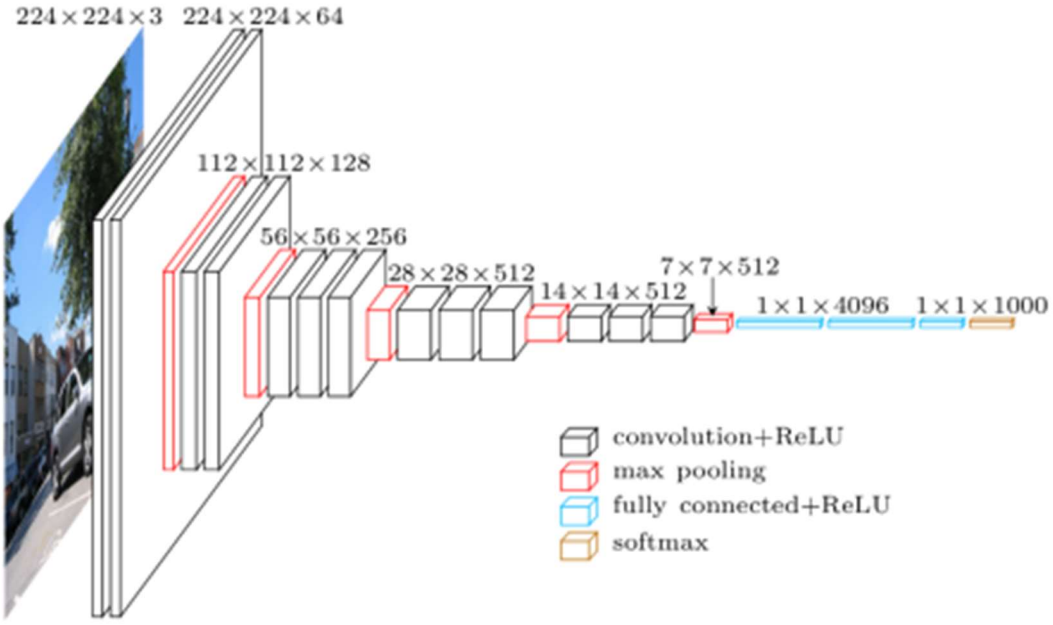


3.5.3. 加深的 CNN 网络模型

与上面的基础 CNN 不同，在这个网络模型中，加深了卷积和池化的部分，具体的网络结构请参看文件代码 deep_cnn.ipynb。

3.5.4. 基于 VGG 改进的 CNN 网络模型

VGG-16 总共有 16 层，其中包括 13 个卷积层和 3 个全连接层。官方给出的 VGG16 结构如下图所示：



VGG16 网络特点

VGG-16 网络结构很规整，没有那么多的超参数，专注于构建简单的网络，都是几个卷积层后面跟一个可以压缩图像大小的池化层。即：全部使用 3*3 的小型卷积核和 2*2 的最大池化层。

- 卷积层：CONV=3*3 filters, s = 1, padding = same convolution。
- 池化层：MAX_POOL = 2*2 , s = 2。

优点：简化了卷积神经网络的结构；

缺点：训练的特征数量非常大。

4. 模型训练测试

4.1. PCA 降维结果

在传统方式分类器中，统一采用 PCA 方式降维，并且进行白化处理，去掉数据之间的关联度，减少冗余信息量，有助于提升训练及预测准确率。

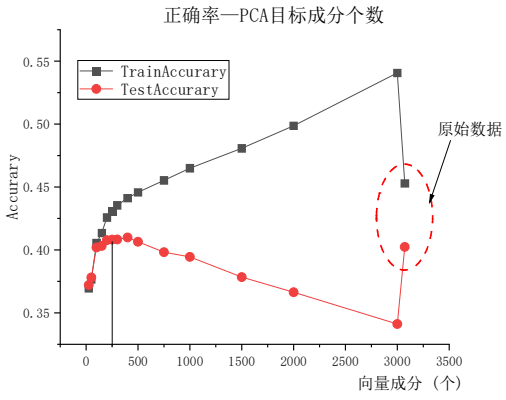
在决定 PCA 降维的向量的个数时，原有向量是 32*32*3 = 3072 维，因此利用分别设置降维后的 PCA 的个数分别为 3000,2000,1500,1000,750,500,400,300,250,100,50,25。依据逻辑回归分类进行实验。

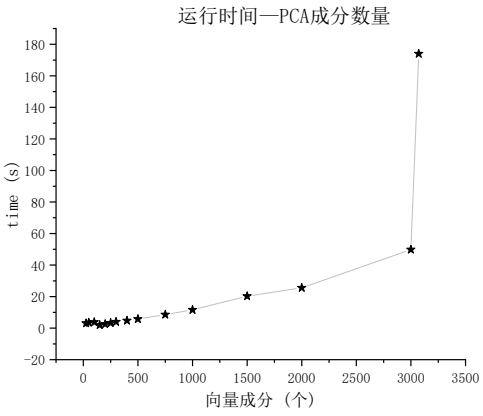
得到的结果如下表所示：

PCAnum	time	TrainAccurary	TestAccuracy	PCAnum	time	TrainAccurary	TestAccuracy
3072	174	0.4528	0.4024				
25	3.1	0.3695	0.3721	**400**	4.8	0.4412	0.4079
50	3.5	0.3767	0.3781	**500**	5.8	0.4458	0.4065
100	3.8	0.4054	0.402	**750**	8.6	0.4553	0.3982
150	2	0.4135	0.4032	**1000**	11.6	0.4649	0.3945
200	2.6	0.4258	0.4078	**1500**	20.3	0.4808	0.3784
250	3.3	0.4306	0.4089	**2000**	25.6	0.4987	0.3664
300	4	0.4355	0.4080	**3000**	49.8	0.5407	0.3411

将该表数据可视化得到下面两图，左图是准确率与 PCA 降维之后的向量维度，右图是运行时间的折线图，可以得出以下结论：

- 使用 PCA 和白化之后分类程序的运行时间会大大减少
- 随着 PCA 剩余的向量维度不断增大，测试集上正确率不断提高，最终过拟合。
- 最优的 PCA 降维维度数量应为 250。





4.2. 调整训练 BATCH_SIZE 的结果

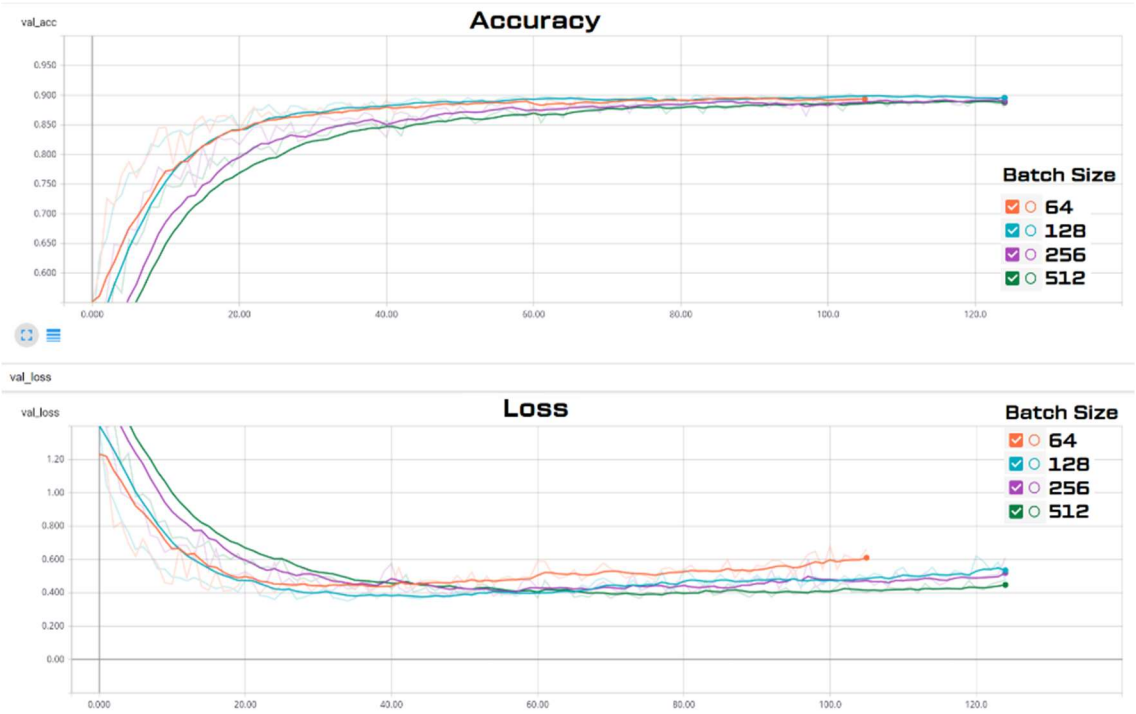
在深度学习过程中，批次大小影响学习效果：

- Batch_size 太小，算法在 200 epoches 内不收敛。
- 随着 Batch_size 增大，处理相同数据量的速度越快。
- 随着 Batch_size 增大，达到相同精度所需要的 epoches 数量越来越多。

- 由上述两个因素，Batch_size 增大到某时刻，达到时间上的最优。
- 由于最终收敛精度会陷入不同的局部极值，因此 Batch_size 增大到某些时候，达到最终收敛精度上的最优。

分别设置 Batch_size 大小为 64,128,256,512，得到的 Accurary 和 Loss 图如下图所示，根据下面的 `accuracy` 曲线，我们可以看到模型在测试集上所能达到的最高准确度为 `90%`，这一准确度在 `batch_size = 64` 的时候大概在 `85` 代达到，而在 `batch_size = 128` 的时候大概在 `104` 代达到，其余的批次都需要更多迭代代数。而反观 `Loss` 函数，`batch_size=64` 时候较大，其余批次依次减少。

因此，选择 `batch_size = 128` 是最佳选择。



4.3. 服务器硬件参数

服务器硬件配置如下表所示：

GPU	NVIDIA Tesla K80	**通道数**	**16**
每秒浮点运算次数	2.01 TFLOPS	**PCIE 带宽**	15.75 GB/s
显卡内存	12 GB	**主板型号**	X10DRG-O+-CPU
GPU 带宽	240.48 GB/s	**CPU 型号**	Intel (R) Xeon (R) CPU E5-2678 v3 @ 2.50GHz
可使用内核数	3cores	**可使用内存**	8GB

5. 结果分析与可视化

5.1. 模型结果综述

我们主要用在训练图片集中的准确率和待检测图片（训练未使用）的预测准确率来衡量不同及其学习方法的优劣。两大类模型的实验结果如下表：

	方法种类	验证集准确率	测试集准确率
传统方法	逻辑回归分类	0.4258	0.4078
	决策树分类	0.8562	0.2678
	随机森林分类	1.0	0.4706
	KNN 分类		
深度学习	基础 CNN	0.7645	0.7187
	深度 CNN	0.8943	0.8022
	改进 VGG	0.9940	0.9142

根据上述结果可以得到以下结论：

- 深度学习的效果比传统方法效果要好很多。
- 目前验证集上最高准确率是 91.42%，训练模型的参数足足有 8966986 个，在上述硬件配置机器上面花费时间 2 h 35 min 29 s。
- 决策树的和随机森林等树型模型在不同程度上都过拟合，导致训练集上的正确率远高测试集上的正确率。

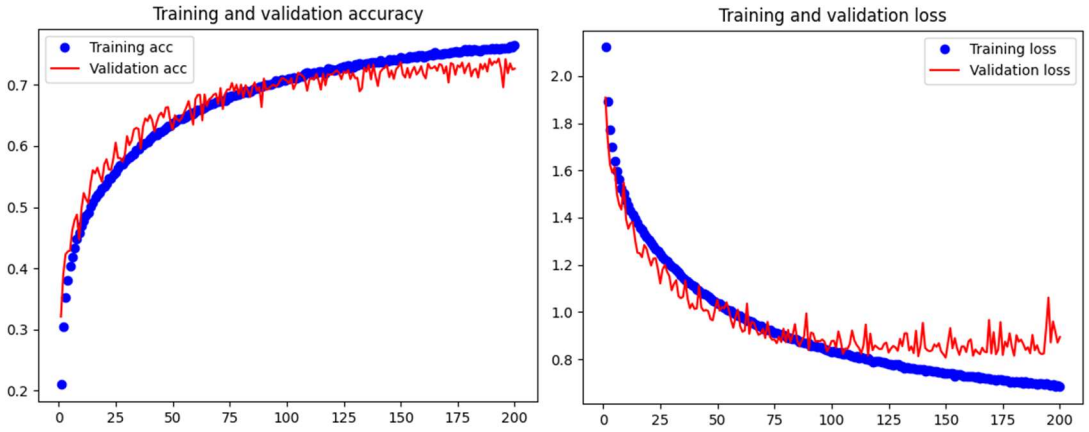
5.2. 模型复杂度对训练效果的影响

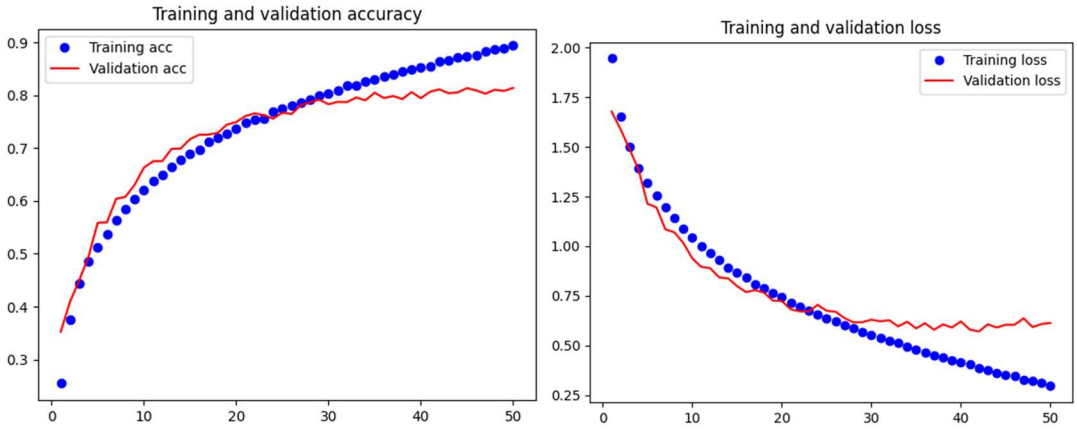
在深度学习中，适当增加模型复杂度虽然会增加训练费时，但会提高模型的最终准确率，下面对最基础的 CNN 模型和深度 CNN 模型进行对比，进一步说明此问题。

评价指标	**Basic_CNN**	**Deep_CNN**
卷积层数	3	10
池化层	3	4
全连接层	1	2
训练参数个数	66218	651766
训练代数 epoches	200	50
训练用时/s	7128.72	24087.79
正确率	71.87%	80.22%

最终收敛代数	150	40

深度 CNN 层数较深，训练参数个数几乎是基础 CNN 的 10 倍，而训练时间也是其 13.5 倍（按照训练 50epoches 用时计算），但是收敛代数深度 CNN 表现优秀。下面四张图片分别是基础 CNN 和深度 CNN 的准确率和 Loss 值的图，能够从中体现出深度 CNN 的优异性。





EXTRA CREDIT：手写实现决策树模型

EC.1. 算法原理

决策树的定义和原理课上已经学习过，不再在此赘述。决策树的三种主流算法是 ID3、C4.5、CART 算法，其中 ID3 和 C4.5 基于信息熵，而 CART 基于基尼系数。我们主要实现了 ID3 算法和 C4.5 算法，并使用 C4.5 算法作为预测的主要手段。

信息熵定义为

$$\text{Entropy} = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)$$

信息增益的定义为

$$\text{infoGain}(D \mid A) = \text{Entropy}(D) - \text{Entropy}(D \mid A)$$

ID3 算法根据信息增益值生成决策树。C4.5 算法是 ID3 算法的改进和拓展。信息增益准则对可取值数目较多的属性有所偏好，为减少这种偏好可能带来的不利影响，C4.5 算法采用信息增益率来选择最优划分属性。信息增益率的定义为

$$\text{GainRatio}(D \mid A) = \frac{\text{infoGain}(D \mid A)}{\text{IV}(A)}$$

$$\text{IV}(A) = -\sum_{k=1}^{K} \frac{|D_k|}{|D|} \log_2 \frac{|D_k|}{|D|}$$

同时，ID3 算法仅可以处理离散型特征值，而 C4.5 算法还可以处理连续型特征值。

EC.3. 剪枝策略

决策树是一个非常容易发生过拟合的模型，因为如果没有任何限制，在生成阶段，它将会为所有特征生成分支。这样的后果是一方面叶子节点过多，容易缺少泛化能力，另一方面训练速度会非常缓慢。

可以通过剪枝的方法来降低树模型复杂度。剪枝可分为预剪枝、后剪枝两类。W 偶们主要采用预剪枝的方法。

EC.3.1. 预剪枝

预剪枝指在完全正确分类之前，决策树会较早地停止树的生长。停止生长的方法可以被总结为通用的停止和严格的停止两种。

EC.3.1.1 通用的停止

- 如果所有样本均属同一类。
- 如果样本的所有的特征值都相同。

EC.3.1.2. 严格的停止

- 如果树到达一定高度。
- 如果节点下包含的样本点小于指定的阈值。
- 如果扩展当前节点不会改善信息增益，即信息增益小于指定的阈值。

我们选择的剪枝方法是通用的停止中“如果所有样本均属同一类”、严格的停止中“如果树到达一定高度”、“如果节点下包含的样本点小于指定的阈值”和“如果扩展当前节点不会改善信息增益，即信息增益小于指定的阈值”。具体实现如下：

“如果所有样本均属同一类”：

```
if len(values) == 1:
    continue  # entropy = 0
```

“如果树到达一定高度”：

```
if node.depth < self.max_depth:
    #...
else:
    #...
```

“如果节点下包含的样本点小于指定的阈值”：

```
if min(map(len, splits)) < self.min_samples_split:
    continue
```

“如果扩展当前节点不会改善信息增益，即信息增益小于指定的阈值”：

```
if gain < self.min_gain:
    continue  # stop if small gain
```

EC.4. 结果

由于数据集为图片，每个数据样本的大小都是$32 \times 32 \times 3 = 3072$，所以如果将像素点直接作为特征，一方面特征数量太多，训练速度比较慢，另一方面特征作为特征不明显，所以我选择了 PCA 的方法，首先将数据降维。

降维之后，分别用手写 ID3 算法、手写 CART 算法和 sklearn 的决策树对模型进行预测，准确率和运行时间如下。

	验证集准确率	测试集准确率	运行时间/s
手写 ID3	87.30%	16.32%	500.32
手写 C4.5	88.45%	26.53%	560.58
sklearn 决策树	86.52%	27.24%	15.24

可以看到，相较于官方库的决策树，手写的决策树运行时间都较长，可能是剪枝不到位的原因。而 ID3 算法的测试机准确率较低，C4.5 算法测试集准确率与官方库相差无几，说明构造出来的决策树是正确的。

参考文献

[1] http://zh.wikipedia.org/wiki/%E6%AD%A3%E6%80%81%E5%88%86%E5%B8%83

[2] http://www.opencv.org.cn/index.php/Cv%E5%9B%BE%E5%83%8F%E5%A4%84%E7%90%86

[3] http://www.cnblogs.com/donj/articles/1656122.html

[4] https://www.cnblogs.com/skyfsm/p/6790245.html

[5] 学习 OpenCV 中文版

[6] 计算机视觉（马颂德版）