**卷积神经网络优化方法的分析**

方可乐

（东南大学网络空间安全学院 无锡 202202）

**摘要** 卷积神经网络是深度学习关于图像处理的一个应用，卷积神经网络的模型选取和优化方法选择，决定了卷积神经网络的准确率。本文选用两个代表性数据集，以LeNet模型为基础，分别用到了图片增强、过拟合处理、欠拟合处理、批标准化处理、自动调节学习率、提前结束等多个优化方法，使两个图片分类器达到98%和99%的准确率，并针对卷积神经网络的不同情况，对优化方法的比较和选择做了总结。

**关键词** 卷积神经网络；优化方法；图像分类

**Analysis of Optimization Methods of Convolutional Neural Networks**

FANG Ke-Le

(*School of Cyberspace Security, Southeast University, Wuxi, 202202*)

**Abstract** Convolutional neural network is an application of deep learning on image processing. The choice of model and optimization method of convolutional neural network determines the accuracy of convolutional neural network. In this paper, two representative datasets are selected and based on the LeNet model, multiple optimization methods such as image enhancement, overfitting, underfitting, batch normalization, automatic learning rate adjustment, and early stopping are used to make the two Each image classifier achieves 98% and 99% accuracy, and for different situations of convolutional neural networks, the comparison and selection of optimization methods are summarized.

**Keywords** convolutional neural network; optimization method; image classification

# 1 前言

当使用卷积神经网络[1][2]优化系统和提高预测率的时候，往往会选用不同的优化方法，在卷积神经网络模型、激活函数、优化器、训练轮数、学习率、过拟合欠拟合的处理方法上有多种选择，本文主要讨论不同的优化方法以及常用的优化流程。

为了体现工作量，本文选用了两个经典的数据集，分别实现手写数字识别分类器和交通标志识别分类器。

# 2 相关知识介绍

**2.1卷积神经网络模型介绍**

常用的经典卷积神经网络（Convolutional Neural Network, CNN）有LeNet、AlexNet、VGGNet、InceptionNet等。

本文讨论主要选用LeNet来讨论，它包括了两个卷积层，两个下采样层（也叫池化层），两个全连接层和一个激活层，最后有一个softmax分类层。可以说，LeNet麻雀虽小，五脏俱全。包括了基本的卷积神经网络的所有单元。

LeNet是最早的卷积神经网络之一。LeNet通过巧妙的设计，利用卷积、参数共享、池化等操作提取特征，避免了大量的计算成本，最后再使用全连接神经网络进行分类识别，这个网络也是最近大量神经网络架构的起点。除了输入层之外，还有七层结构，分别是卷积层、池化层、卷积层、池化层、卷积层、全连接层、输出层。

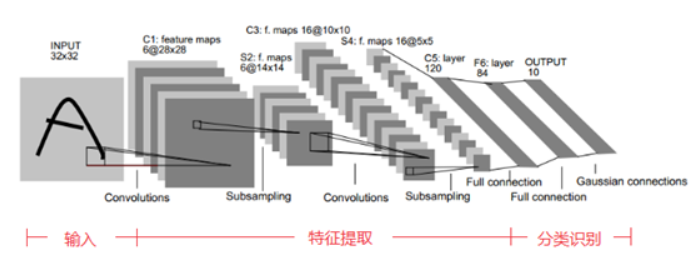


图2.1 LeNet结构

卷积计算可认为是一种有效提取图像特征的方法。

一般会用一个正方形的卷积核，按指定步长，在输入特征图上滑动，遍历输入特征图中的每个像素点。每一个步长，卷积核会与输入特征图出现重合区域，重合区域对应元素相乘、求和再加上偏置项得到输出特征的一个像素点。图片分为单通道图（黑白）和三通道图（彩色）。输入特征图的深度(channel数），决定了当前层卷积核的深度;当前层卷积核的个数,决定了当前层输出特征图的深度。对于单通道图，一般用2\*2\*1的卷积核，对于三通道图，使用3\*3\*3或者5\*5\*3的卷积核。

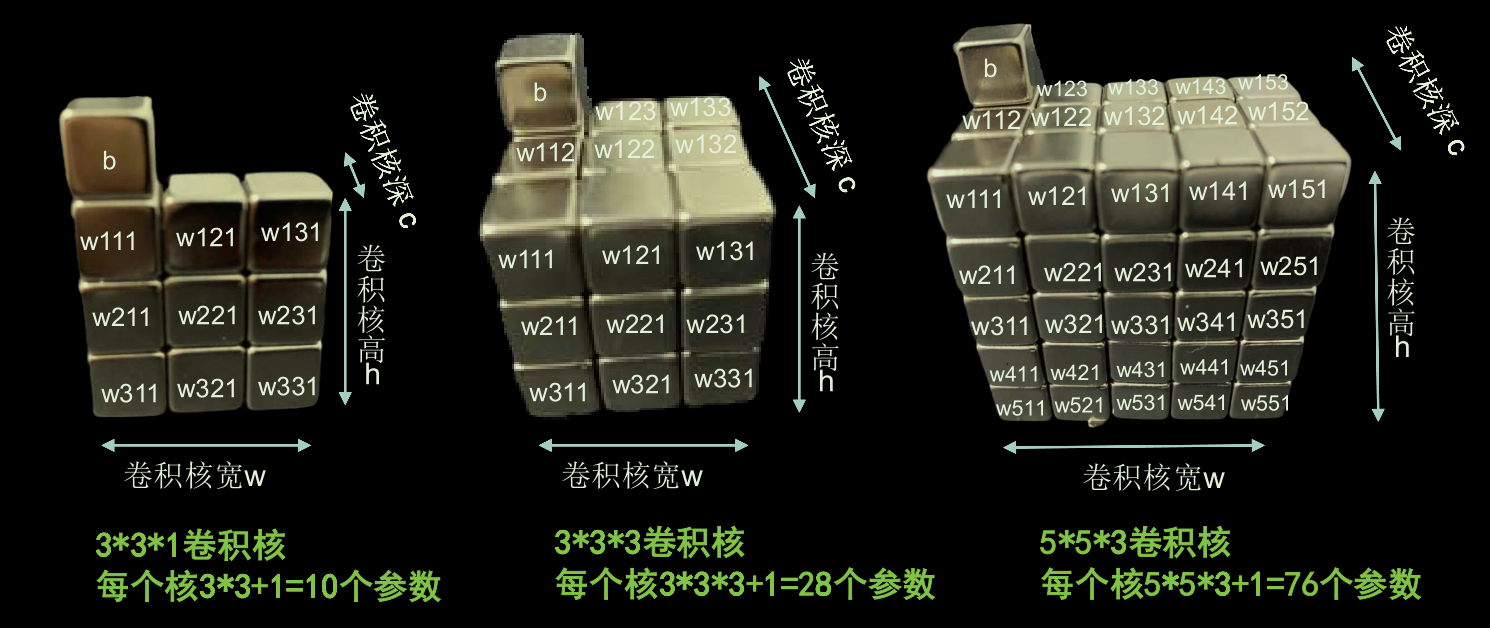


图2.2 卷积核

卷积核的计算是指卷积核在输入层每滑动一个步长就跟重叠区域进行相乘运算，相乘求和再加偏置项，得到输出特征的一个像素点。需要说明的是，卷积核的每个单元都预先设置一个参数，这些待训练参数在反向传播时会被梯度下降算法更新。卷积计算如下图：

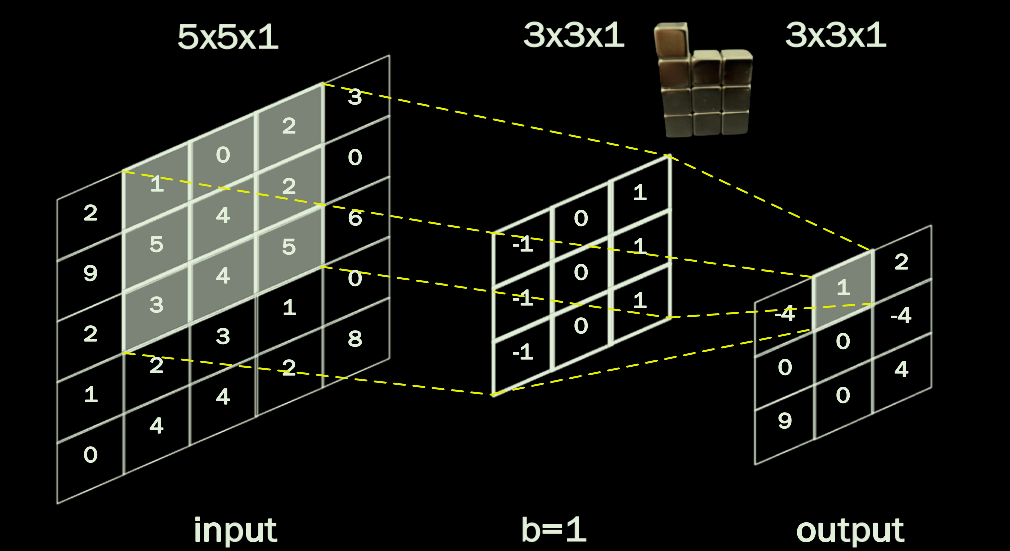


图2.3 卷积计算

如果希望卷积计算保持输入特征图的尺寸不变，就可以使用全零填充，在输入特征图周围填充0，在卷积核的计算后得到的输出特征大小保持和输入特征一致。

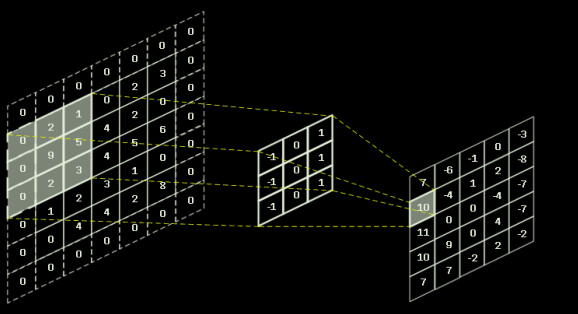


图2.4 全零填充

池化用于减少特征数据量。

最大值池化可提取图片纹理，均值池化可保留背景特征。如下图所示：

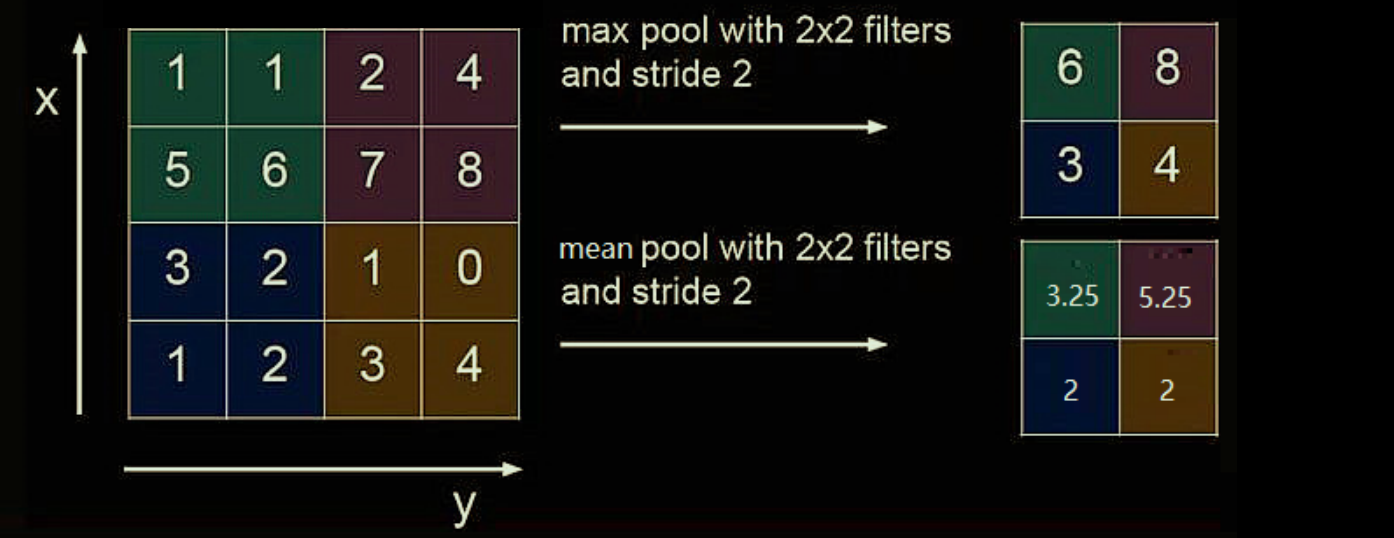


图2.5池化

在神经网络训练时，将一部分神经元按照一定概率从神经网络中暂时舍弃。神经网络使用时,被舍弃的神经元恢复链接。

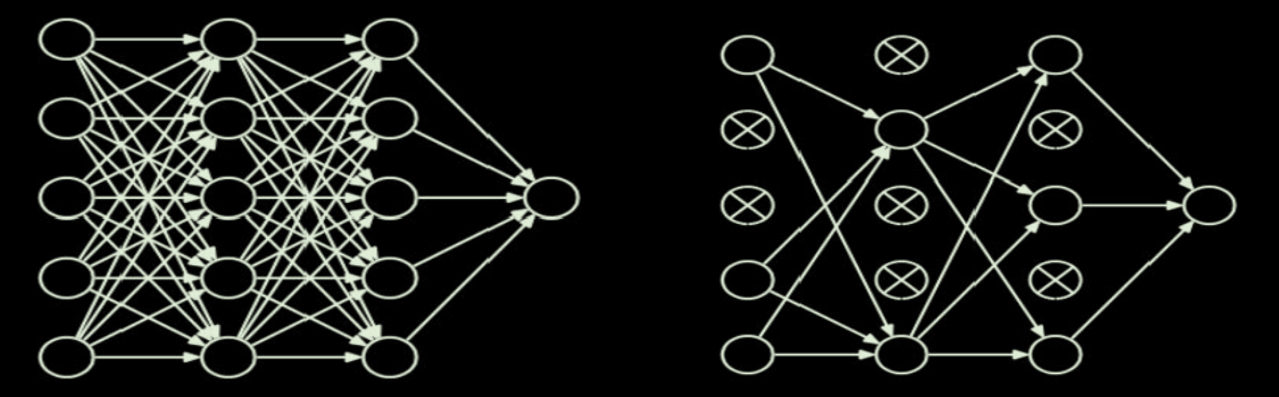


图2.6 Dropout

全连接层，是每一个结点都与上一层的所有结点相连，用来把前边提取到的特征综合起来。由于其全相连的特性，一般全连接层的参数也是最多的。为了提升 CNN 网络性能，全连接层每个神经元的激励函数一般采用ReLu函数。最后一层全连接层的输出值被传递给一个输出，可以采用softmax逻辑回归（softmax regression）进行分类，该层也可称为softmax层。

其余模型只做简单介绍，不做展开。

AlexNet首次在CNN中成功应用了ReLu激活函数、Dropout和数据增强。也使用CUDA加速深度卷积网络的训练，利用GPU强大的并行计算能力，处理神经网络训练时大量的矩阵运算。

VGGNet沿用了AlexNet的技术特点，新特征是用小尺寸卷积核减少参数,网络结构规整，需要的收敛的迭代次数较少许多。

InceptionNet的特点在于一层内使用不同尺寸卷积核,提升感知力使用批标准化,缓解梯度消失。

**２.２激活函数**

一开始用的较多的激活函数[3][4][5]是Sigmoid函数，即f（x）=1/(1+e-x) 。但现在用的较少。由于神经网络需要计算反向传播，需要从后往前计算倒数，进行链式求导，而Sigmoid函数的导数值是介于0和0.25,链式求导需要多个倒数连续相乘，许多个介于0和0.25的值相乘，结果是趋于0的，这显然会导致梯度消失。另一个不用Sigmoid函数的原因是该函数的求导幂运算复杂，训练时间长。

比Sigmoid函数更好的选择是tanh函数，f(x)=(1-e-2x)/(1+e-2x)，但该函数也会造成梯度消失的现象，也有幂运算复杂，训练时间长的问题。

现阶段用的较多的是Relu函数。f(x)=max(x,0)

Relu函数的优点如下:

(1)解决了梯度消失问题(在正区间)。

(2)只需判断输入是否大于0，计算速度快。

(3)收敛速度远快于sigmoid和tanh。

缺点如下：

(1)输出非0均值，收敛慢。

(2)Dead Relu问题：某些神经元可能永远不会被激活，导致相应的参数永远不能被更新。

改进的Relu函数有Leaky Relu和ERelu函数，它们具备Relu的全部优点而且也不存在Dead Relu问题，在x>0的区间，表达式和Relu完全一直，在x<0的区间，Leaky Relu为f(x)=ax,ERelu为f(x)=a(ex-1)。

**2.3欠拟合和过拟合**

欠拟合主要产生的原因:模型复杂度过低，无法很好的去拟合所有的训练数据，导致训练误差大。

解决方法：

* 增加模型复杂度，加入更多的卷积层和全连接层；
* 增加模型参数；
* 使用非线性模型等。

过拟合主要产生的原因:模型复杂度过高，训练数据少，训练误差小，但是测试误差大。

解决方法：

* 在层之间添加 dropout层，丢弃一些数据。
* 提前结束训练；
* 在输入中添加噪声
* 选用合适的优化器
* 批标准化
* 数据增强

上述优化方法在本文中均有体现。

# 3 实验方法

实验环境

系统：Windows 10

CPU: AMD Ryzen 7 5800H with Radeon Graphics

GPU:NVIDIA GeForce GTX 1650

开发工具版本：Pycharm2021.2.3

jupyter core 4.7.1

jupyter-notebook 6.3.0

为了体现工作量，本文以两个经典数据集来实现卷积神经网络的优化过程[6][7][8]。分别是手写数字识别分类器和交通标识识别分类器

# 4手写数字识别分类器

**4.1数据集分析**

本文使用的数据是kaggle比赛Digit Recognizer项目中的数据集，如下图所示：

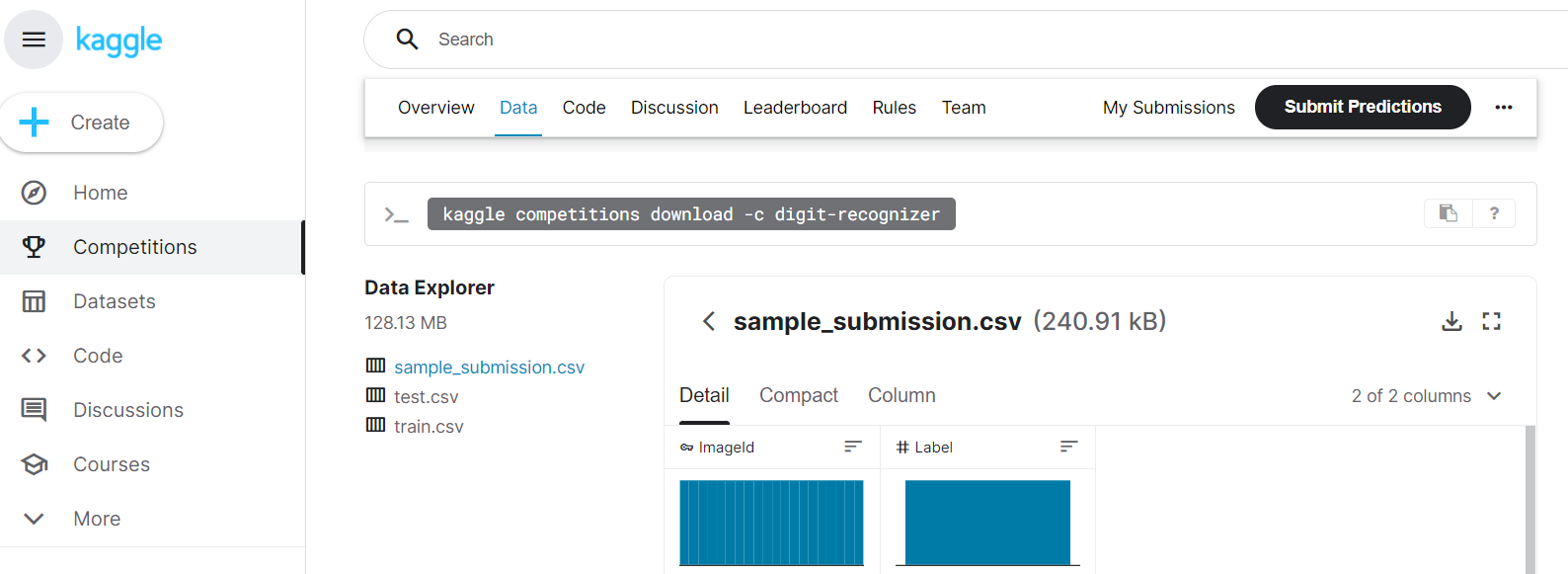


图4.1 数据集

首先导入相关的tensorflow和keras相关的包，并读入数据集，如下图所示：

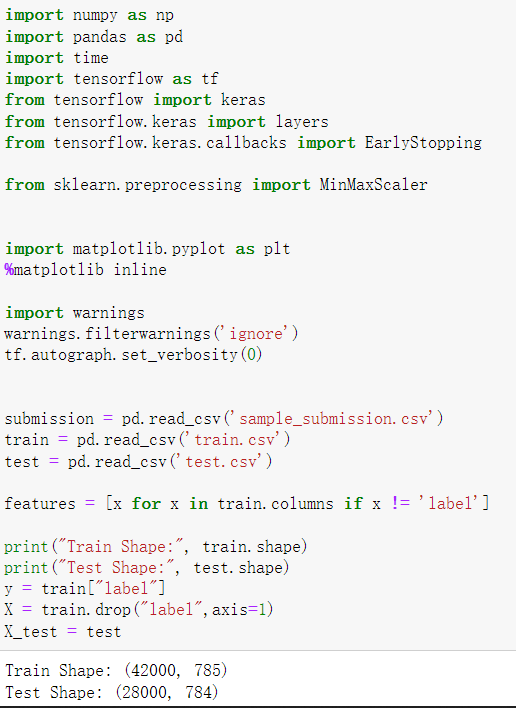


图4.2 读入数据集

可以得知训练集有42000张图片，测试集有28000张图片。下一步对数据集划分，把对应的数据分别赋值给训练集、测试集、验证集以及相应的标签，并用MinMaxScaler()函数对数据进行归一化处理，使每个像素值都成为介于0到1之间的数值，方便后面神经网络计算，相关操作的代码如下图：

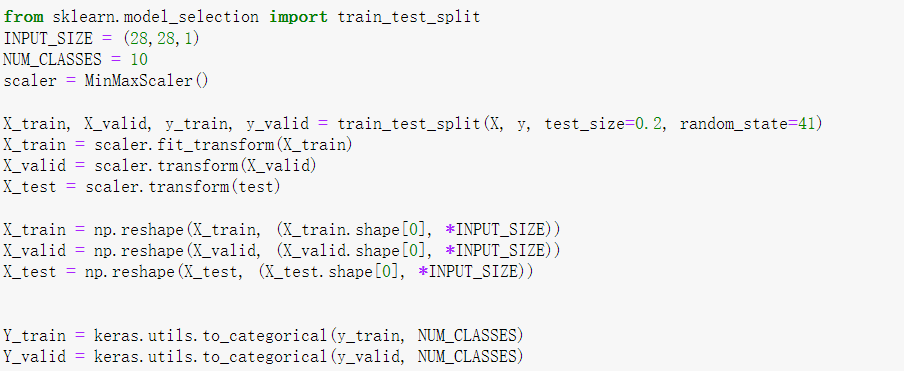


图4.3 相关处理

下一步构建第一个神经网络，如下图：



图4.4构建第一个神经网络

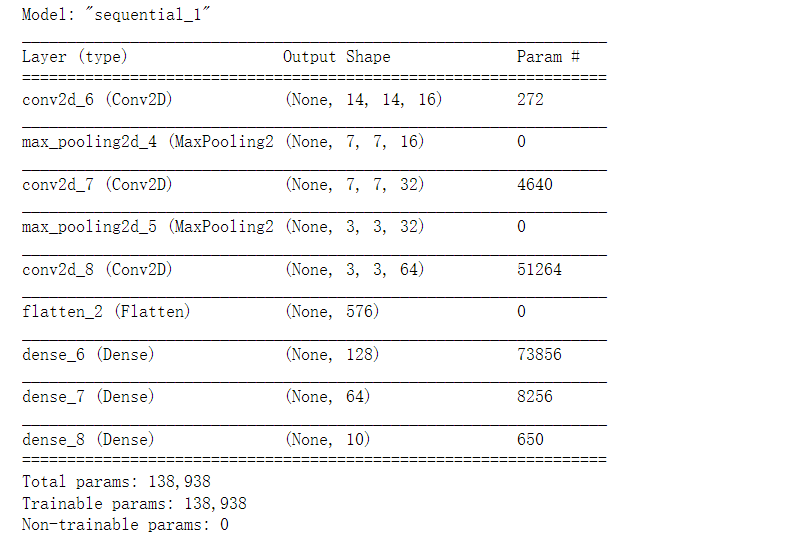


图4.5 神经网络概述

下一步用LearningRateScheduler包自动调节学习率，并将学习率与epoch的关系画出来：

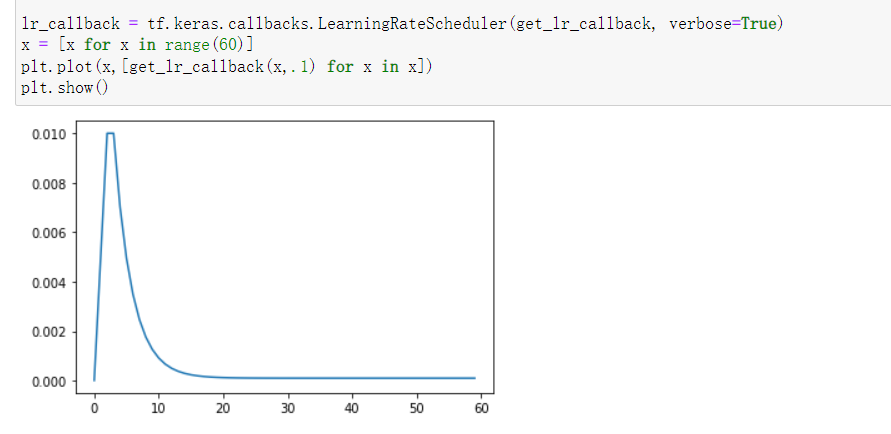
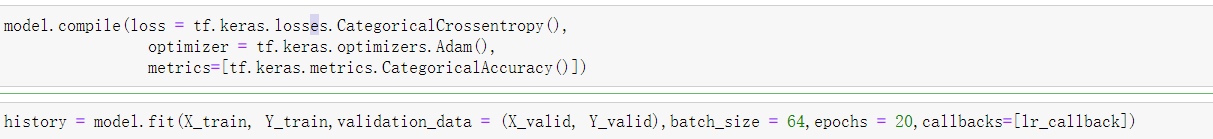


图4.6学习率与epoch的关系图

从图中可以看出一开始快速学习，然后收敛到理想状态。

神经网络后开始训练：



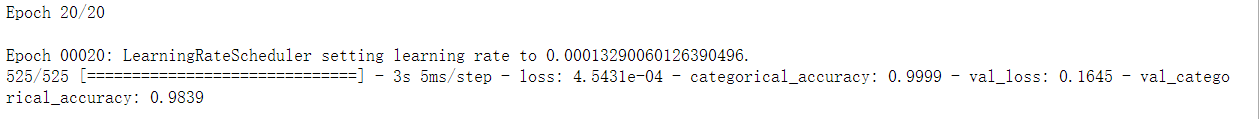


图4.7 训练神经网络

从结果可以看出此时的准确率为98.39%，距离99%还有一定差距。

画出loss曲线观察结果：

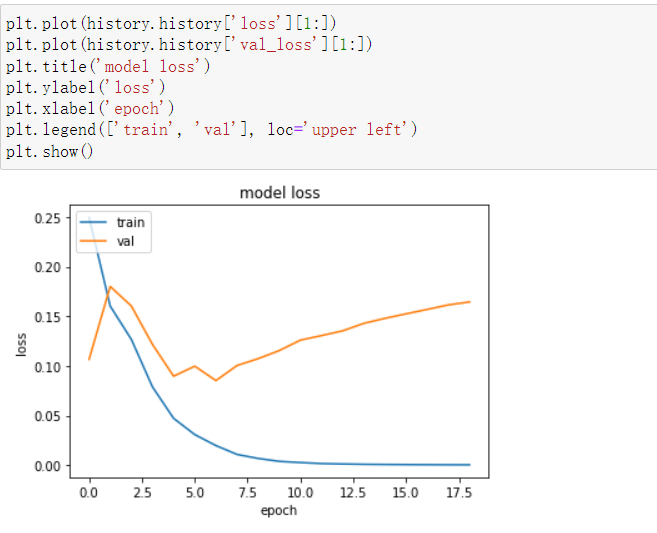


图4.8 loss曲线

可以看出，该模型出现了过拟合（overfitting）现象可以看到，训练损失继续减少，而价值损失增加。此时需要修改模型使得模型的泛化能力更强，通常情况下可以使用以下方法：

* 在层之间添加 dropout层；
* 使用Earlystopping提前结束训练；
* 在输入中添加噪声；
* 用SDG优化器代替Adam（Adam 更快，但 SDG 泛化更多，一般使用 Adam 来加快进程然后用SDG）；
* 添加 BatchNormalization 以加快学习速度；
* 使用数据增强。

首先用图片增强的方法。为了保证图片预测的准确率更高，为了增加训练样本的多样性，使整个系统更加稳定，可以对训练的图片做图片增强。这个操作可以借助keras的ImageDataGenerator来实现。

本文将应用以下几种变换：

width\_shift: 水平方向移动，幅度10%；

height\_shift: 垂直方向移动，幅度10%；

zoom: 放大与缩小，幅度20%；

rotation: 顺时针或逆时针旋转，幅度10度。

随后构建新的神经网络模型，如下图：



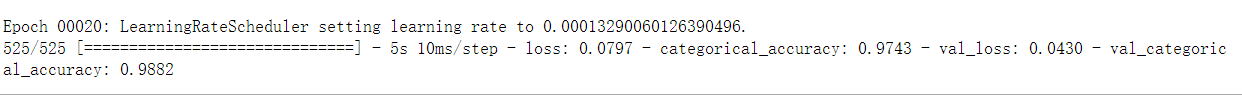


图4.9 构建新的神经网络模型

从结果可以看出结果已经提高到了98.82%。

下一步用ReduceLROnPlateau进一步优化。ReduceLROnPlateau的作用是当标准评估停止提升时，降低学习速率。当学习停止时，模型总是会受益于降低 2-10 倍的学习速率。 这个回调函数监测一个数据并且当这个数据在一定“有耐心”的训练轮之后还没有进步， 那么学习速率就会被降低。

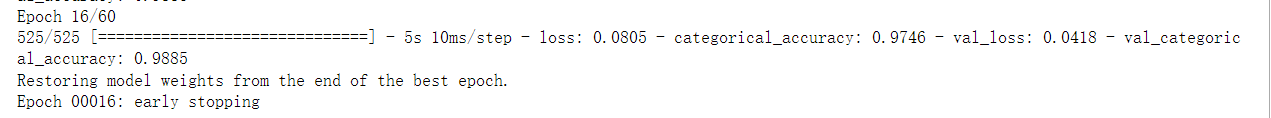


图4.10 结果

从上图可知，模型在第16轮早停止，准确率为98.85%。

绘制出loss曲线，如下图：

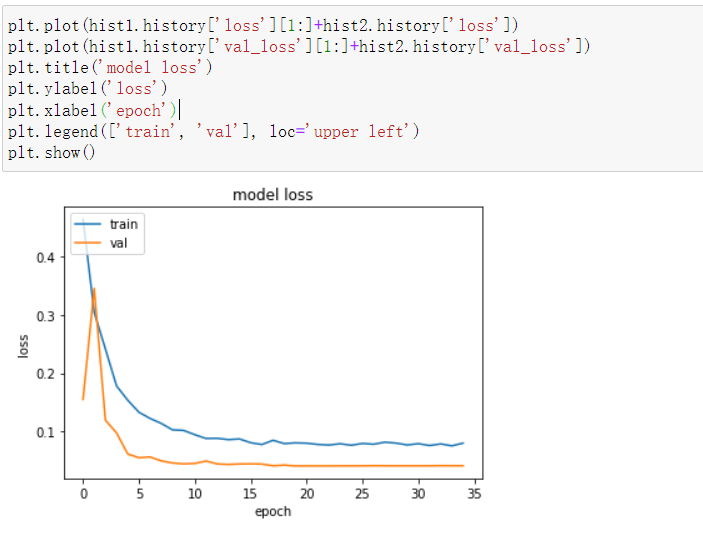


图4.11 绘制loss曲线

可以看到 val\_loss 仍然随着训练而下降：模型泛化。

但是，也可以注意到模型不再改进。可以减少dropout以增加模型的表示能力或增加模型的大小。

构建新的神经网络模型，如下图：



图4.12 构建新的神经网络模型

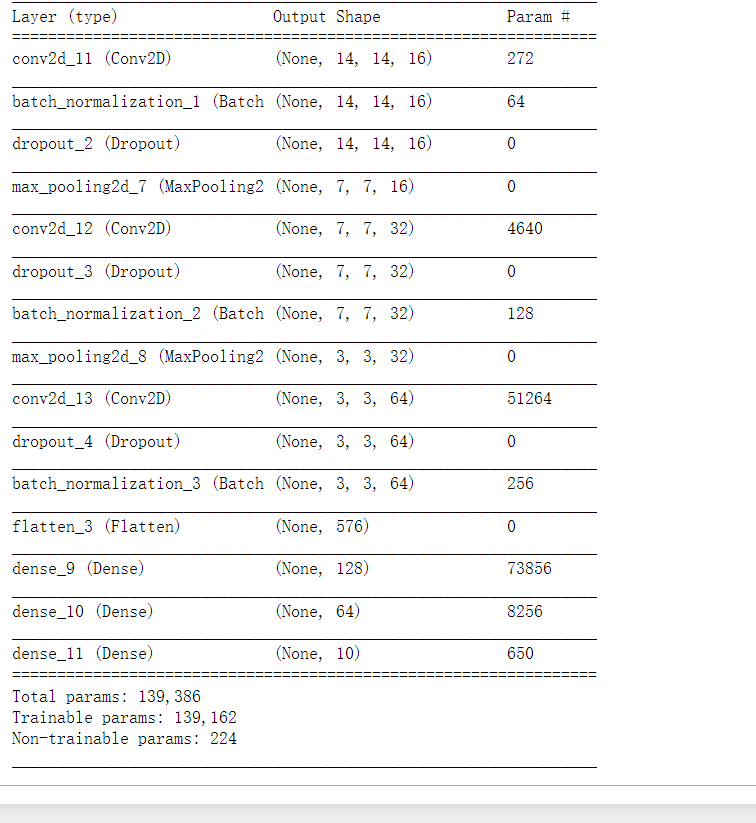


图4.13 神经网络概述

下一步用Earlystopping进一步优化模型，一般来说机器学习的训练次数会设置到很大，如果模型的表现没有进一步提升，那么训练可以停止了，继续训练很可能会导致过拟合keras.callbacks.EarlyStopping就是用来提前结束训练的。代码如下：

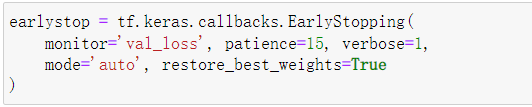


图4.14 相关代码

对代码作简要解释：

monitor：指的是监控的数据接口，这里需要监控的是损失曲线，自然选val\_loss。

patience :没有进步的训练轮数，在这之后训练就会被停止。

verbose：详细信息输出。

mode : {auto, min, max}其中之一。在min模式中，当被监测的数据停止下降，训练就会停止;在max模式中，当被监测的数据停止上升，训练就会停止;在auto模式中，方向会自动从被监测的数据的名字中判断出来。

restore\_best\_weights :是否从具有监测数量的最佳值的时期恢复模型权重。如果为False，则使用在训练的最后一步获得的模型权重。

训练模型，在训练的参数中加入callback回调函数来优化结果，优化器部分选用最基础的Adam优化器，代码如下：

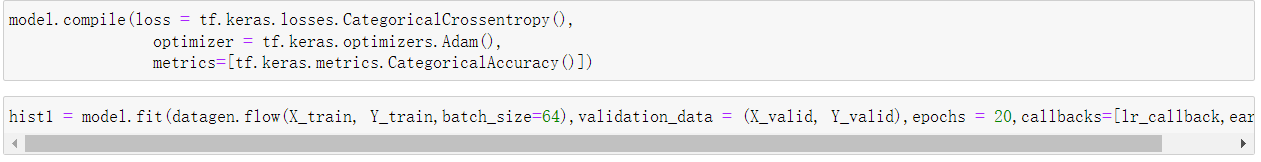


图4.15 相关代码

训练结果如下：

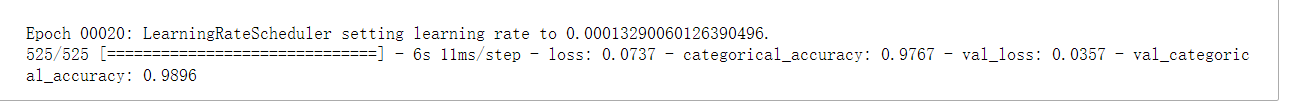


图4.16 训练结果

可以看到，经过优化，准确率达到了98.96%。

在优化器部分，选用SGD替换掉Adam。这是因为Adam 更快，但 SDG 泛化更多，一般使用 Adam 来加快进程然后用SDG。

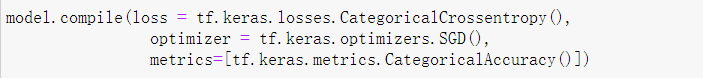


图4.17 优化器

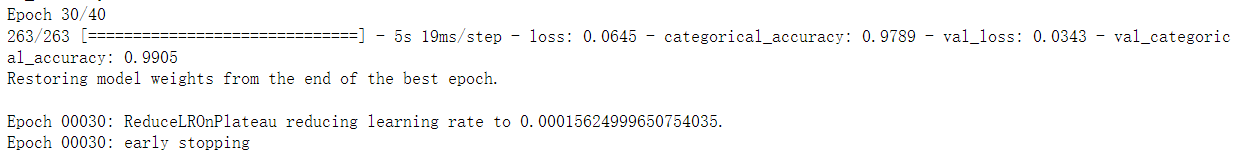


图4.18 结果

可以看到，模型在第30轮达到了早停止，准确率达到了99.05%。

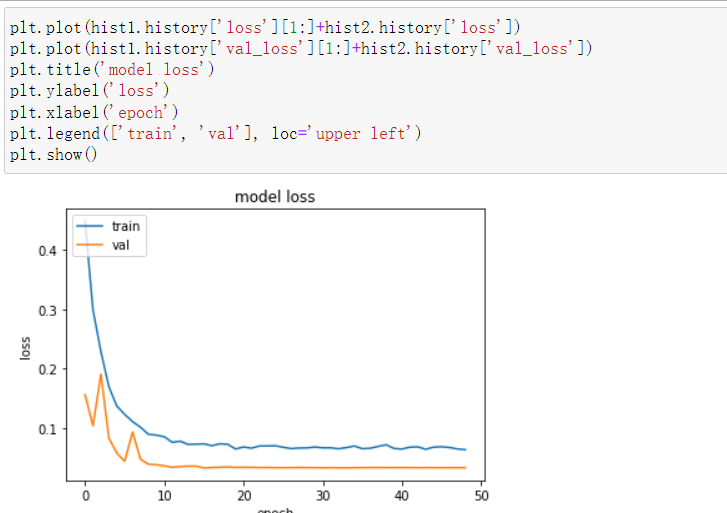


图4.19 绘制loss曲线

最新的loss曲线可以看出，在验证集的结果已经相当好。

在kaggle提交预测结果的csv文件，可以看出得分为0.99003，说明预测率达到了99%。

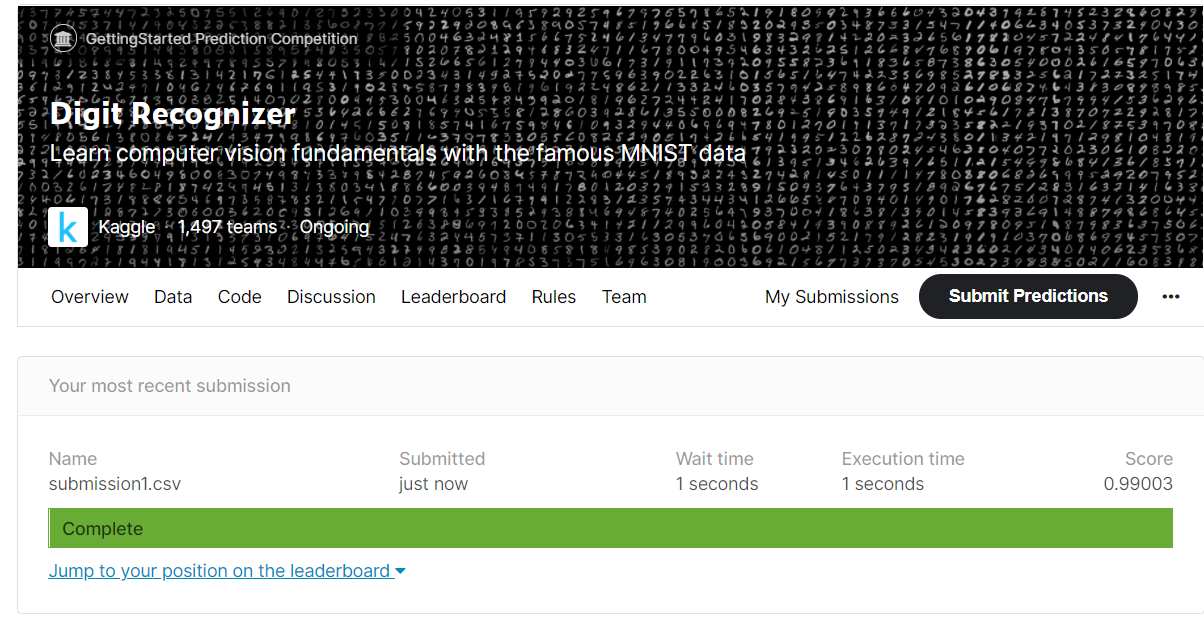


图4.20 结果

# 5 交通标志识别分类器

**5.1数据集分析**

使用的数据是一份德国的交通标志图片：德国交通标志数据集——GTSRB交通信号数据集（包含43种交通信号）数据地址为：<https://bitbucket.org/jadslim/german-traffic-signs>

数据集文件如图所示：

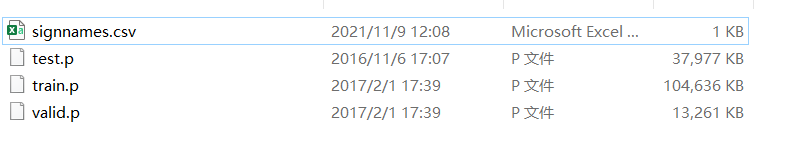


图5.1 数据集文件

其中，signnames.csv文件包括各种交通标志的名称和对应的编号，如图所示：

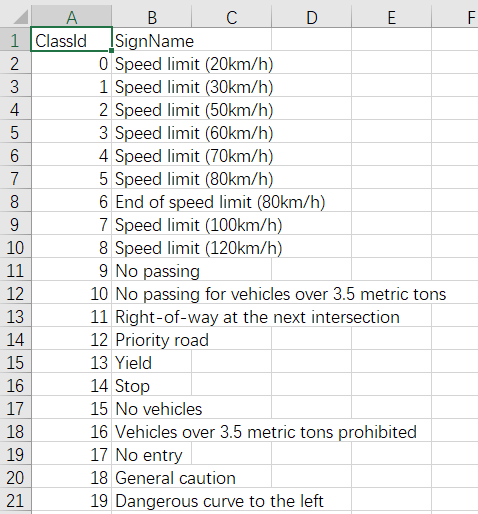


图5.2 signnames.csv

数据集分为训练集、测试集和验证集，具有以下特点:

1.图像为32(宽)×32(高)×3 (RGB彩色通道)

2.训练集由34799张图片组成

3.验证集由4410个图像组成

4.测试集由12630个图像组成

5.共有43个种类

**5.2数据处理**

由于数据集.p结尾的文件，是pickle序列化的文件，需要导入pickle模块进行数据的处理。

为了每一步执行的结果都可以可视化，故选用Jupyter Notebook来编辑代码。

首先读取训练集、测试集和验证集的内容，并读取csv文件中的信息，如下图3所示：



图5.3 数据处理

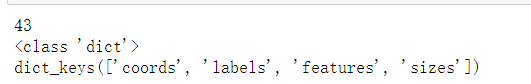


图5.4 数据集输出结果

从输出结果可以看出，signnames.csv文件长度为43，说明有43种不同的交通标志，训练集有四个属性，分别是'coords', 'labels', 'features', 'sizes'，本文只需要用到labels和features，分别为图片的分类id和图片本身。

以下代码实现的功能是查看数据集的大小，如图所示：

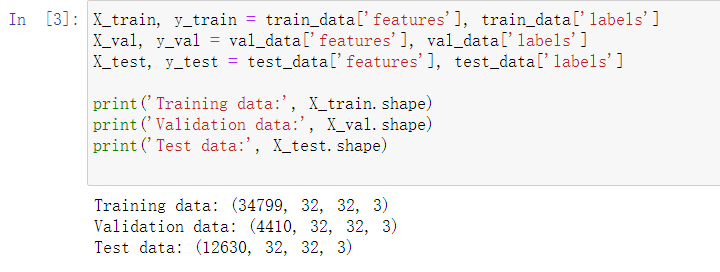


图5.5 查看数据集大小

从结果可以看出，训练集数据有34799个，验证集数据有4410个，测试集数据有12630个；图片的大小为32 x 32像素。

接下来输出每个交通标志类型的五张图片和对应的标签，如图所示，由于图片类型共有43种，限于篇幅，输出结果仅展示前4种标志，如图所示：

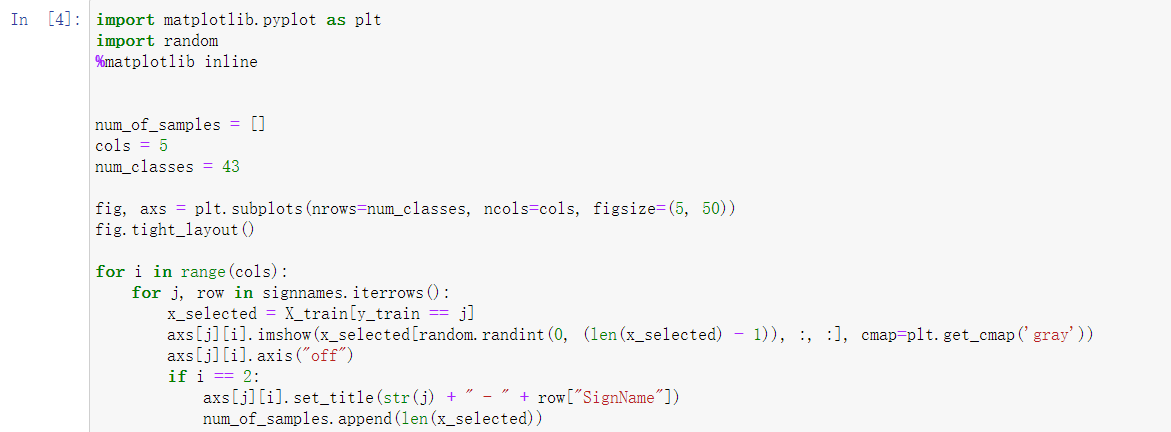


图5.6 查看图片

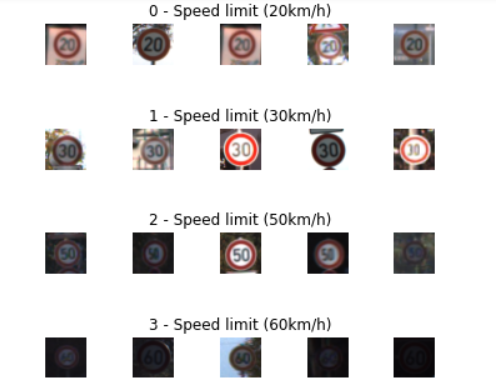


图5.7 图片输出结果

从输出结果可以看出，每类图片的差别较大，亮度不一致或者背景不一致，以下代码实现输出每类图片的数量。如图所示：

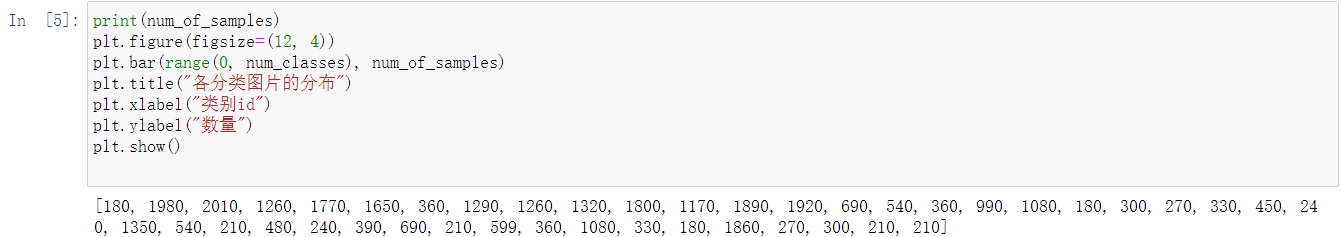


图5.8 查看图片数量

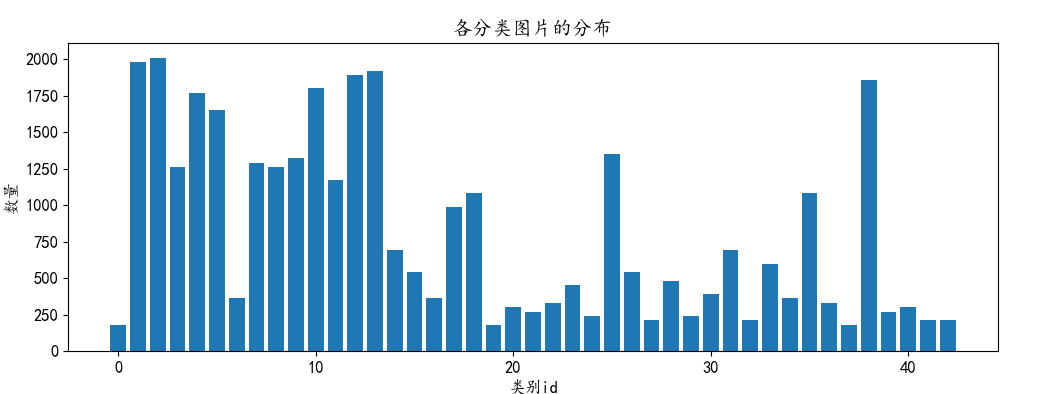


图5.9 图片数量分布

从上图可以看出，每一类图片的数量不一致，有的仅有不到200张图片，有的超过了1800张图片。

**5.3分类**

在不对图片进行处理的情况下，先使用最简单的卷积神经网络进行分类，在这种情况下观察准确率可以达到多少。本文用LeNet对图片分类，LeNet是早期卷积神经网络中最有代表性的实验系统之一。在LeNet发明初期，计算机并没有GPU来支持模型训练，只有CPU运算，LeNet通过巧妙的设计，利用卷积、参数共享、池化等操作提取特征，避免了大量的计算成本，最后再使用全连接神经网络进行分类识别，这也是选择LeNet的原因之一。

以下是对模型的定义，如图所示：

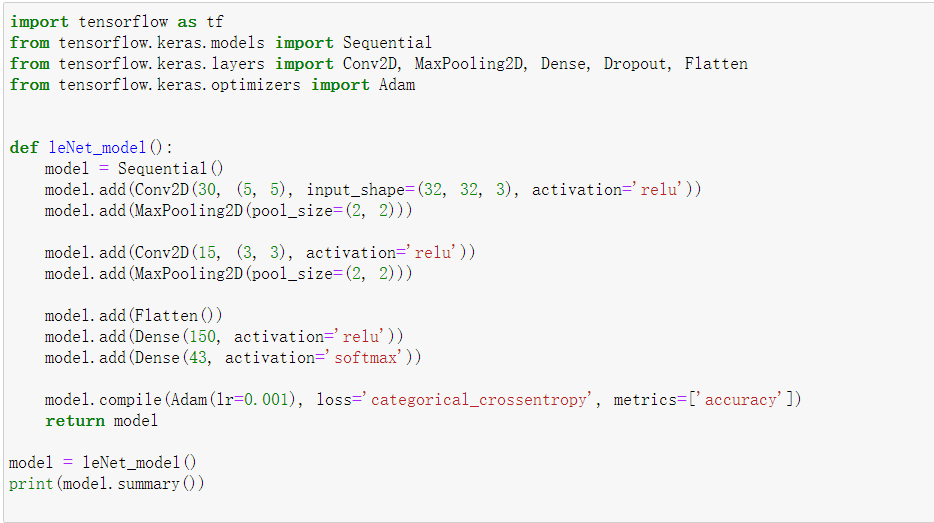


图5.10 模型定义

从代码可以看出，该模型有两层CNN加MaxPooling，将输入数据进行Flatten操作，Flatten的作用是“压平”操作，由于图片是32\*32的像素，用RGB表示法表示整个图片的信息，每个像素都是一个介于0到255的数值，整个图片是一个矩阵，为了方便后续运算，需要把图片的多维信息变为一维矩阵，这一操作常用于卷积层到全连接层的过渡。在Flatten操作之后，经过一层全连接Dense，随后使用softmax归一化指数函数将分类结果以概率的形式表示，激活函数选用的是ReLu函数。

模型定义后，模型相关信息如下图所示：

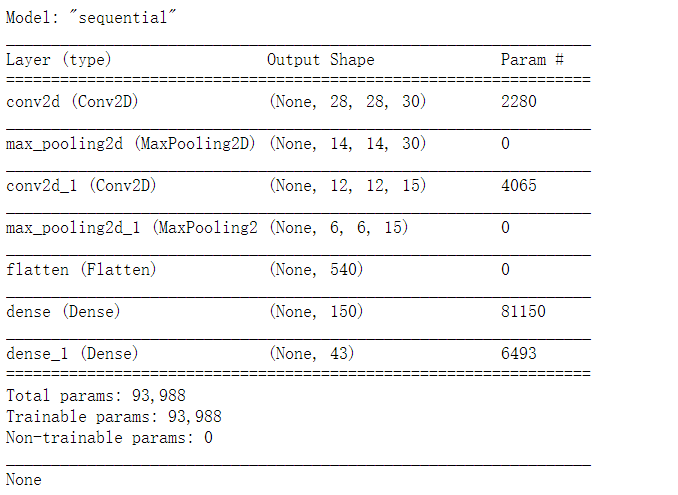


图5.12 模型相关信息1

接下来将43个分类做one hot独热编码处理，方便后续分类，用to\_categorical函数实现，接着进行训练，训练20轮，batch设置为400。

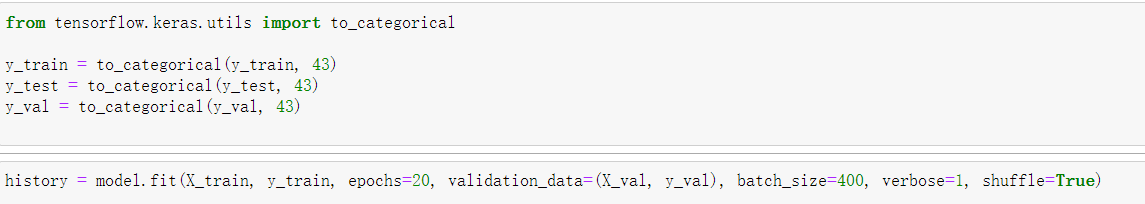


图5.13 独热编码处理与训练

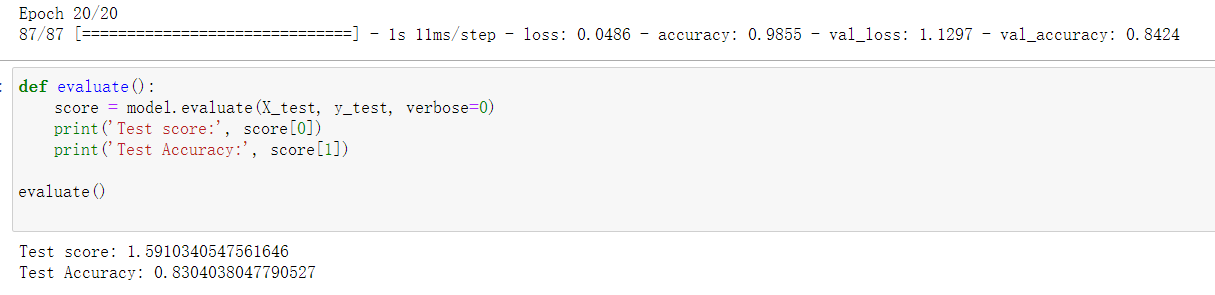


图5.14 模型训练结果1

从代码执行结果可以看出，训练集的准确率达到了98.55%，验证集的准确率是84.24%，测试集的准确率仅有83.04%，这说明存在着严重的过拟合现象。

为了更直观的观察，用matplotlib画出loss曲线和accuracy曲线。

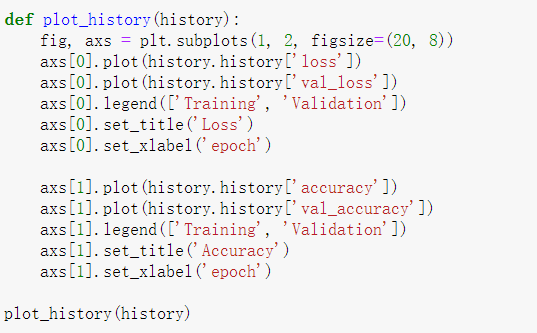


图5.15 绘制曲线代码

绘制结果如下图所示：

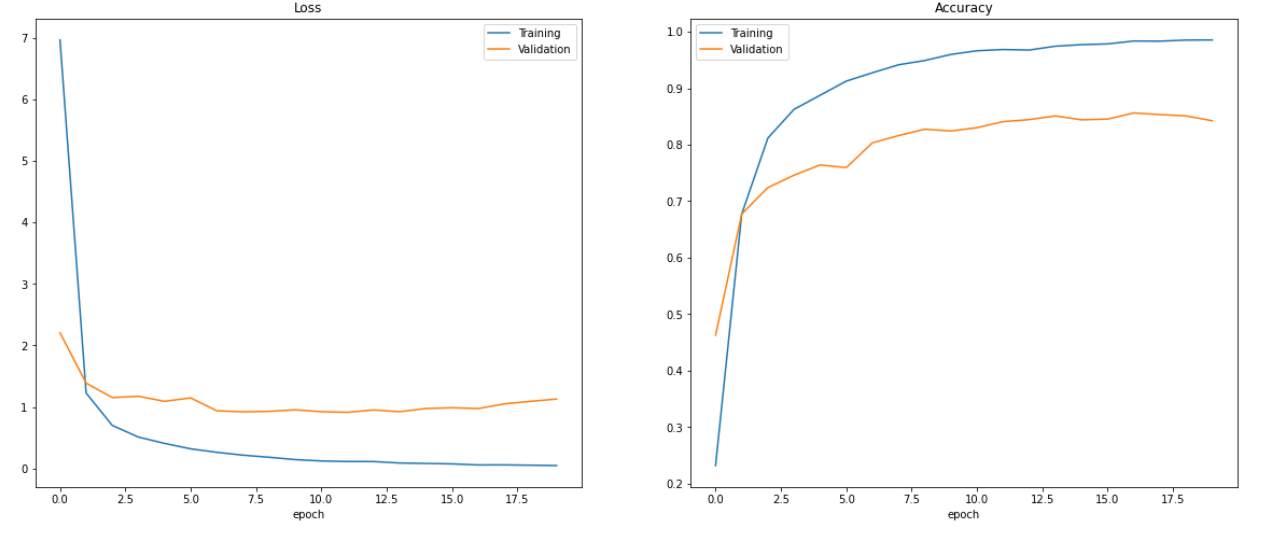


图5.16预测绘制结果1

从图中可以看出验证曲线和训练曲线的差距较大。

**5.4优化**

针对过拟合，最简单的模型优化方式就是使用Dropout，在模型中添加Dropout层。只需要在modified\_model()函数中稍作变化即可，模型相关信息如下图所示：

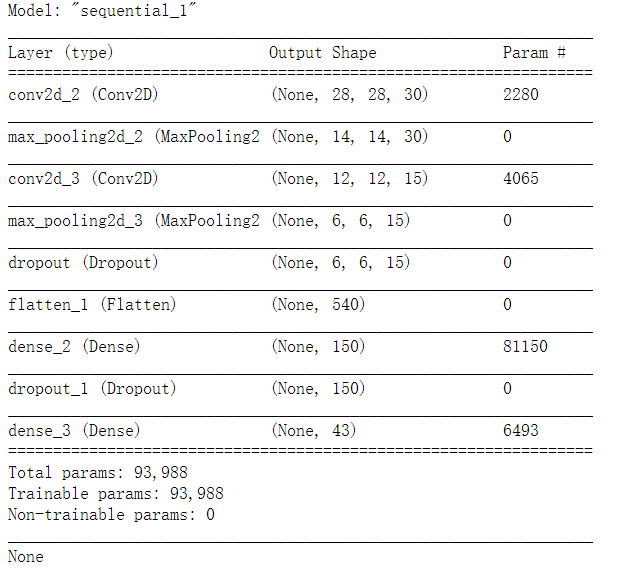


图5.17模型相关信息2

为了提高准确率，一般情况下激活函数不选用ReLu函数，这是因为它有如下缺点：

(1）输出非0均值,收敛慢

(2) Dead Relu问题:某些神经元可能永远不会被激活，导致相应的参数永远不能被更新。

而leak ReLu函数和eReLu函数，这两种函数可以保证当x是负值的时候，函数的梯度不为0，本文选用的是后者。f(x)=max(x, α（ex-1）)

优化后的代码实现如下图所示：



图5.18 代码优化

模型训练结果如下图所示：

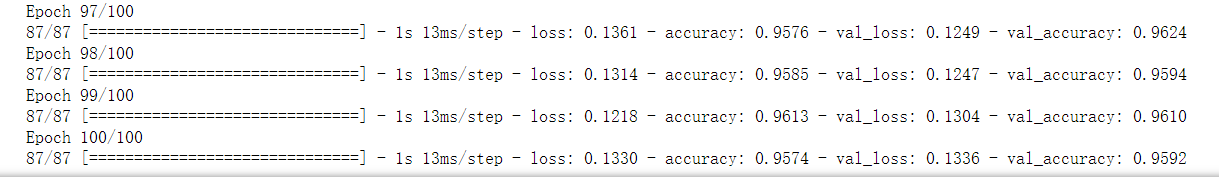


图5.19 模型训练结果2

可以看出训练的准确率已经有了较大提升，接近96%。

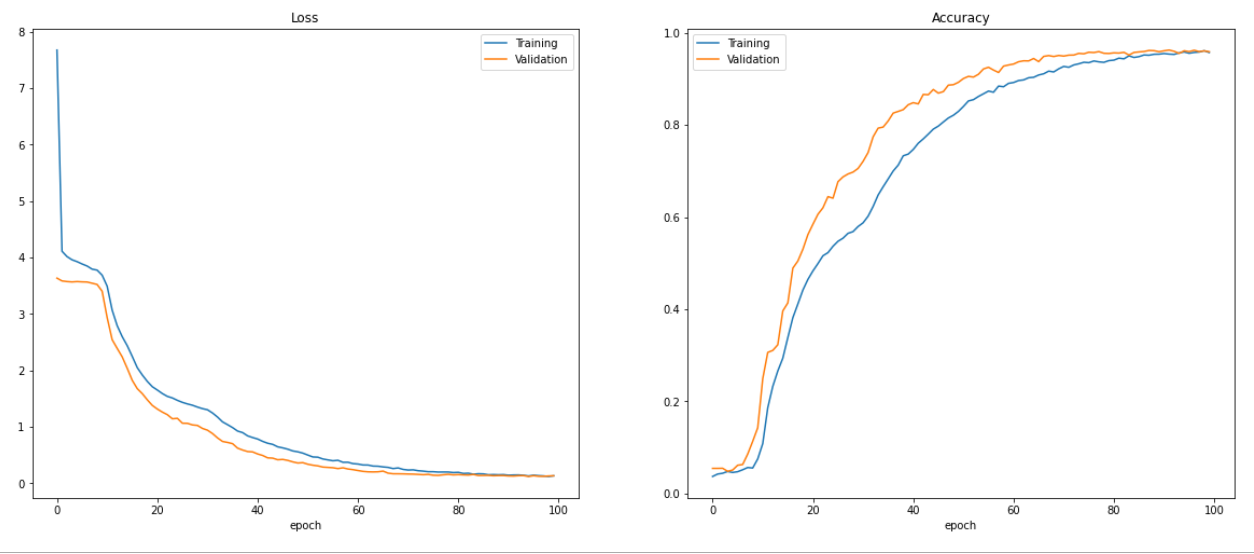


图5.20 预测绘制结果2

从曲线图可以明显看出验证曲线和训练曲线的差距越来越小。

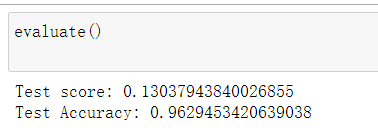


图5.21 验证集准确率

验证集的准确率也达到了96.29%。

为了保证图片预测的准确率更高，为了增加训练样本的多样性，使整个系统更加稳定，可以对训练的图片做图片增强。这个操作可以借助keras的ImageDataGenerator来实现。

本文将应用5种变换：

width\_shift: 水平方向移动，幅度10%；

height\_shift: 垂直方向移动，幅度10%；

zoom: 放大与缩小，幅度20%；

shear: 平行四边形变换，可沿水平或者垂直方向变换，幅度为与坐标轴夹角0.1度；

rotation: 顺时针或逆时针旋转，幅度10度。

代码实现如下图所示：

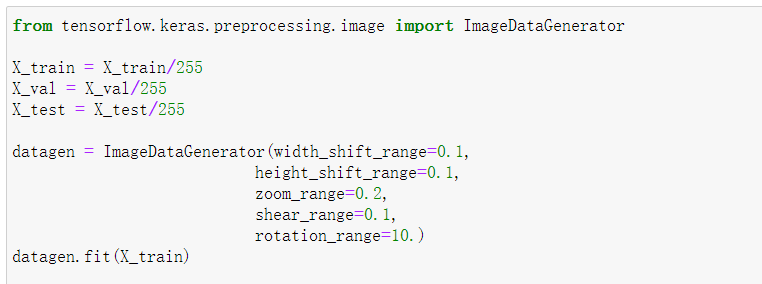


图5.22 图片增强

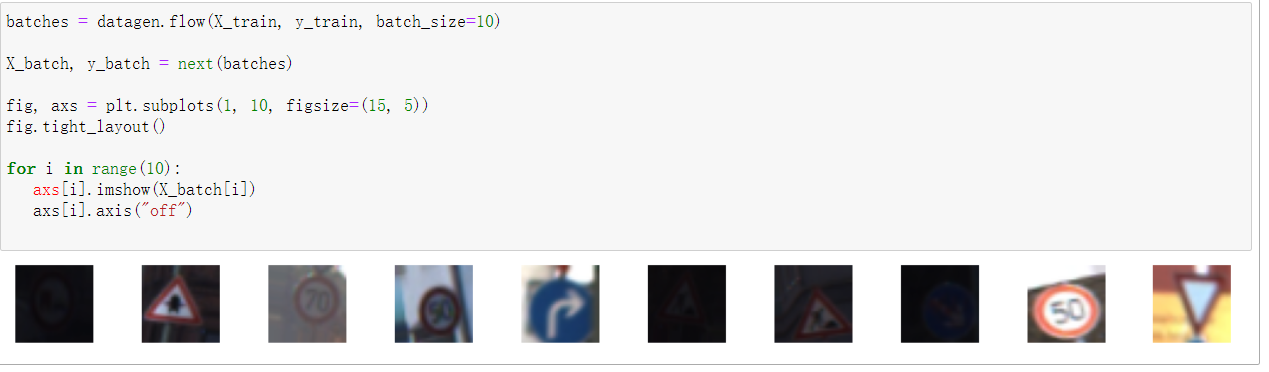


图5.23 处理结果

从上图可以看出，随机选中的图片的确发生了变化，方便满足样本的多样性，便于后续的训练。

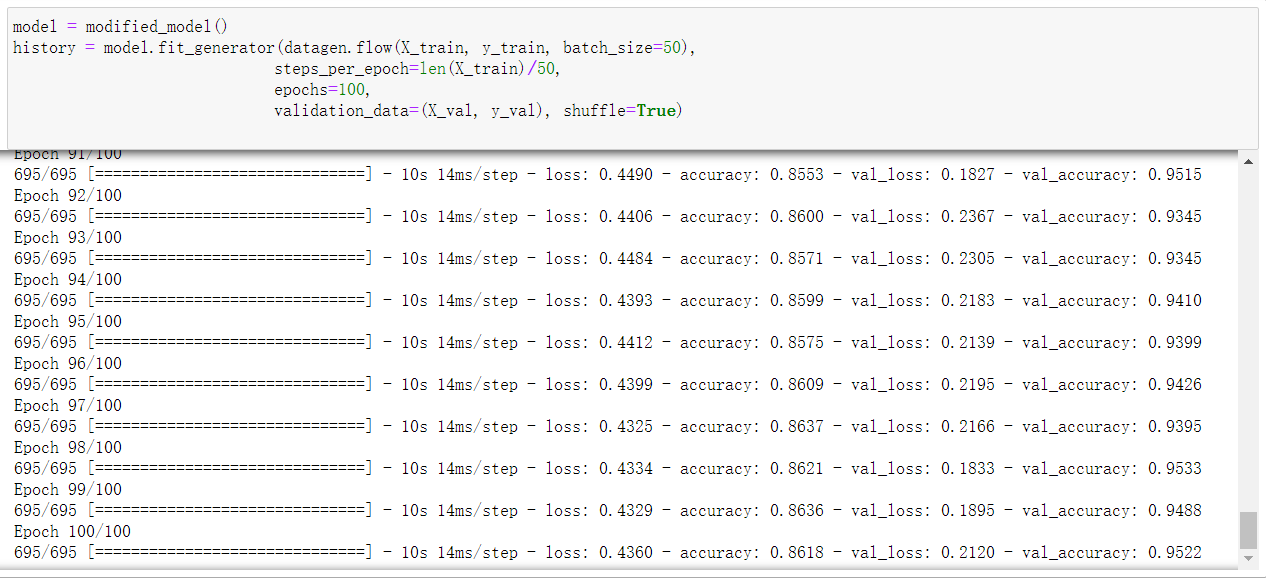


图5.24 模型训练结果3

再次训练，虽然运用了图片增强，可以看出训练的准确率没有上升反而下降，说明训练模型存在欠拟合(underfitting)现象，这是因为模型是最简单的LeNet结构，需要增加一些层次来抽取更复杂的特征。可以给模型加入更多的CNN层，增加全连接层的output，可以看出训练模型的参数增加到16万多个。

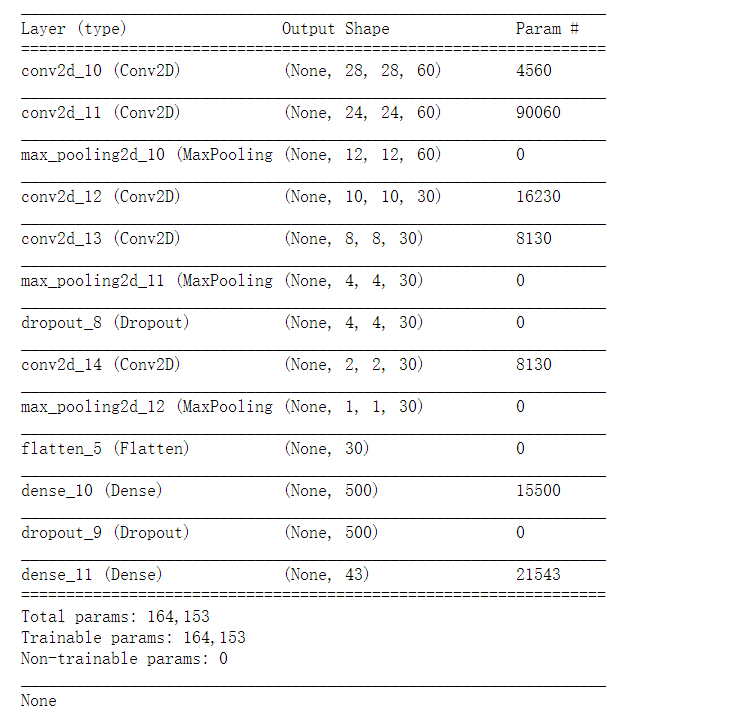


图5.25 模型相关信息3

模型训练结果如下图所示：

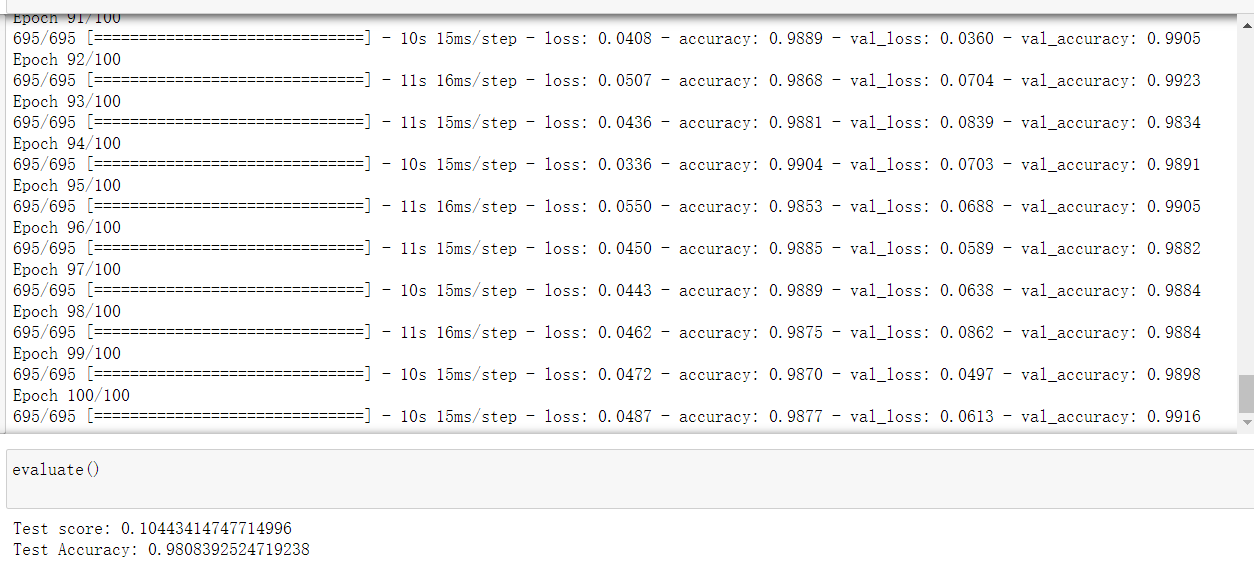


图5.26 模型训练结果4

从图中可以看出，训练集准确率达到了98.77%，验证集上的准确率有99.16%，测试集的准确率有98.08%，可见这次优化有一定效果。

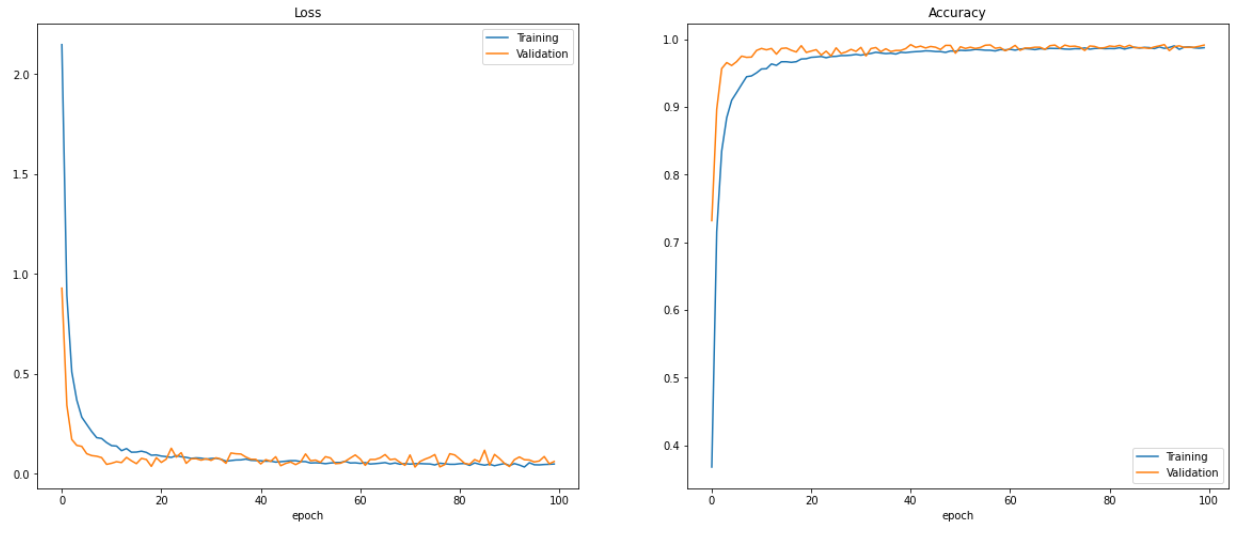


图5.27 预测绘制结果3

上图中可以看出，随着训练轮次增多，验证曲线和训练曲线的差距更小，预测效果更好。

# 6 结论

表6-1展示了手写数字识别分类器各优化方法前后的准确率。

表6-1手写数字识别分类器

|  |  |
| --- | --- |
| 优化方法 | 准确率 |
| 初始模型 | 97.01% |
| 自适应学习率调整 | 98.39% |
| 图片增强 | 98.82% |
| 提前结束训练 | 98.96% |
| SGD优化器 | 99.05% |

优化卷积神经网络模型时候，一般先选用最简单的LeNet模型进行训练，如果预测率不理想，可以考虑加入更多的卷积层和全连接层，或者换用更复杂的模型，例如AlexNet、VggNet、InceptionNet等，但对应的训练时间会更长。

当手动设置的学习率learnrate和训练轮数，训练出的效果不佳时，可以用tensorflow的LearningRateScheduler包自动调节学习率，选取合适的参数传递给训练模型。

为了增加训练样本的多样性，可以对训练的图片做图片增强，对原始数据集中的图片做旋转、平移、放大和缩小，这个操作可以借助keras的ImageDataGenerator来实现。需要注意的是，并不是所有的数据集在用了图片增强方法后，训练的准确率都会提高，也存在准确率降低的情况，这时可能是因为模型出现了欠拟合现象，需要用其他方法做优化。

训练过程中如果出现过拟合现象，可以考虑在层之间添加 dropout层，丢弃一定比例的数据。也可以用批标准化的方式以加快学习速度。有时候最佳的训练结果在训练过程中出现而不是在训练最后一轮才出现，因此可以使用Earlystopping提前结束训练，在训练结果不再提高时候提前结束。

训练过程中如果出现欠拟合现象，可以选用更复杂的训练模型，或者在原有训练模型的基础上加入几层卷积层和全连接层。

激活函数一般选用Relu函数，当训练效果不佳，可以考虑使用ERelu和Leaky Relu。但需要注意的是，后两种激活函数不一定比Relu的效果更好。

# 7 展望

除了使用LearningRateScheduler包自动调节学习率，还可以使用网格参数搜索法来确定合适的学习率和训练轮数，这个方法会在后续学习中进一步强化。

断点续训是训练神经网络的一个重要方法，断点续训的意思是因为某些原因模型还没有训练完成就被中断，下一次训练可以在上一次训练的基础上继续训练而不用从头开始；这种方式对于那些训练时间很长的模型来说非常友好。但由于本文的数据集选择的训练模型比较简单，另外在使用断点续训的时候发现对于模型准确率的提高并没有帮助，因此放弃了该方法。在今后的学习中可以考虑更多的使用到该方法。

**参 考 文 献**

1. 周飞燕,金林鹏,董军.卷积神经网络研究综述[J].计算机学报,2017,40(06):1229-1251.
2. 卢宏涛,张秦川.深度卷积神经网络在计算机视觉中的应用研究综述[J].数据采集与处理,2016,31(01):1-17.DOI:10.16337/j.1004-9037.2016.01.001.
3. 李彦冬,郝宗波,雷航.卷积神经网络研究综述[J].计算机应用,2016,36(09):2508-2515+2565.
4. 许可. 卷积神经网络在图像识别上的应用的研究[D].浙江大学,2012.
5. 蒋帅. 基于卷积神经网络的图像识别[D].吉林大学,2017.
6. 汪济民. 基于卷积神经网络的人脸检测和性别识别研究[D].南京理工大学,2015.
7. 段萌. 基于卷积神经网络的图像识别方法研究[D].郑州大学,2017.
8. 谢宝剑. 基于卷积神经网络的图像分类方法研究[D].合肥工业大学,2015.