自主实现卷积神经网络图像分类器

小组成员：李豹 唐瑞 裴浩渊

中国科学技术大学

课程：机器学习

# 摘要

实验的任务是自主实现卷积神经网络图像分类器。本次实验的主要目的是深入理解CNN网络中的实现细节，掌握CNN网络中的不同层的功能以及对实验结果的影响。作为对课程上CNN网络学习成果的一种验证。

# 目录

[摘要 1](#_Toc534639559)

[目录 1](#_Toc534639560)

[1 简介 2](#_Toc534639561)

[2 背景 2](#_Toc534639562)

[3 实验内容 2](#_Toc534639563)

[4 代码结构分析 4](#_Toc534639564)

[5 实验结果 6](#_Toc534639565)

[6 结论 6](#_Toc534639566)

[7 引用 6](#_Toc534639567)

# 1 简介

目前实现CNN有许多简便的方法，例如tensorflow，以及更加上层，封装的更好的keras，简单的学习就可以掌握基本的网络搭建。但是自主实现CNN有助于增强对CNN的理解，也更有助于增强对各种超参数的理解。本次实验就是基于上述目的开展的，充分利用numpy矩阵运算的优势实现神经网络的卷积层，池化层，Flatten层，全连接层及相应的激活函数，softmax函数和交叉熵损失函数。

# 2 背景

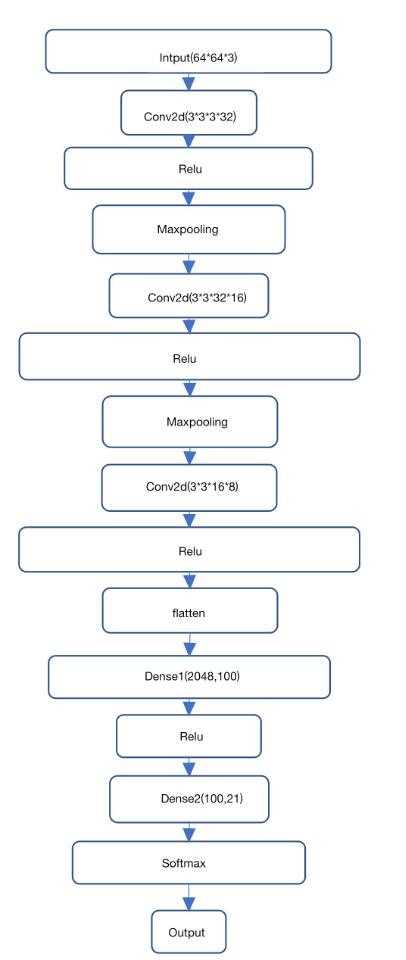
卷积神经网络是一类包含卷积计算且具有深度结构的前馈神经网络，是深度学习的代表算法之一 。由于卷积神经网络能够进行平移不变分类，因此也被称为“平移不变人工神经网络”。

对卷积神经网络的研究始于二十世纪80至90年代，时间延迟网络和LeNet-5是最早出现的卷积神经网络 ；在二十一世纪后，随着深度学习理论的提出和数值计算设备的改进，卷积神经网络得到了快速发展，并被大量应用于计算机视觉、自然语言处理等领域。[1]

# 3 实验内容

通过构建卷积神经网络模型来实现对图像的分类。

模型结构如图



代码文件：

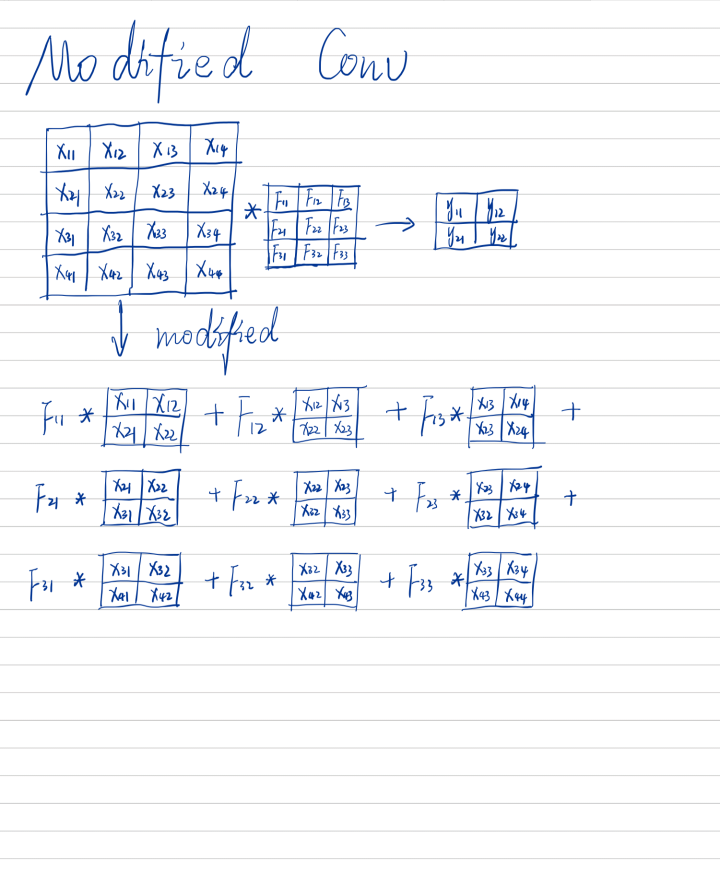
* net.py（实现对网络各层的定义）
* train.py (网络训练代码)
* val.py（测试验证代码）

# 4 代码结构分析

1.net.py:

网络各层的函数定义说明：

* 每个网络层定义为一个class，每个class主要包含forward和backward两个函数。
* 损失函数定义成一个类，使用的是交叉熵损失函数。
* 每一层的反向链接函数所求的值是交叉熵loss输出对参数，数据的导数。
* 卷积层：所有卷积层padding均为1，kernal\_size均为3\*3，故卷积层不改变特征图大小，在正向和反向传播时均要先进行padding。
* forward：先对输入的图片的通道做检查，然后进行卷积操作。考虑到算法时间复杂度的要求，在卷积操作时没有使用两层64的for循环，而是对各个卷积核的每个通道同时进行逐像素遍历，再将结果叠加，只需要循环9次即可。
* backward：输入为Loss输出对下一层的数据导数，输出为Loss输出对当前层数据，权重，偏移量，即（x,w,bias）的导数，求导方法使用的是deconv方法。
* 权重初始化使用glorot\_uniform初始化，Glorot均匀分布初始化方法，又称Xavier均匀初始化，参数从[-limit, limit]的均匀分布产生，其中limit为sqrt(6 / (fan\_in + fan\_out))。fan\_in为权值张量的输入单元数，fan\_out是权重张量的输出单元数。
* Bias初始化为0初始值。
* 卷积操作的优化：



变换循环的坐标轴从而使循环的次数与卷积核大小有关而不是输入数据大小有关，从而减少循环次数提升前相链接的速度。

* 池化层：
* forward：最大池化，同时记录下输出的index以便反向传播时求导。输出的尺寸为输入大小的一半
* backward:对输入的梯度进行双线性插值然后与index矩阵点乘，将最大点的梯度传至上一层。
* Relu层：
* Forward:像素值为负的置为0，非负数不变
* Backward:梯度值为负的置为0，非负数置1
* Flatten层：将输入的特征图拉成一行，反向求导时将输入的梯度向量reshape为特征图的维度，传至上一层
* Dense层：将权值与输入数据进行矩阵相乘，反向求导时分别对输入数据、权值、偏移求导
* Softmax层：根据定义计算概率值和梯度值
* 交叉熵损失函数类：只有前向函数，根据定义计算交叉熵损失
* Net类：输入为网络各层，包含forward，backward，SGD，train，validate，predict等函数
  + forward：将输入数据逐层传递，返回最终的输出值
  + backward：将forward的输出逐层反向求导
  + SGD：计算一个batch训练完后的平均梯度，然后更新一次，更新时为防止梯度爆炸采用了梯度剪切的方法
  + train:调用net.train（）对网络进行训练，每个epoch训练完成后调用self.validate()对验证集进行验证，并且当准确率高于最优模型时保存该模型。
  + validate:用验证集对模型进行验证，返回top1和top3的准确率。
  + predict:测试函数，对数据的预测结果进行格式化输出，并返回top1和top3的准确率

2.train.py

* + - 数据读取函数

readDATASET：完成对数据的读取

* + - 数据预处理代码

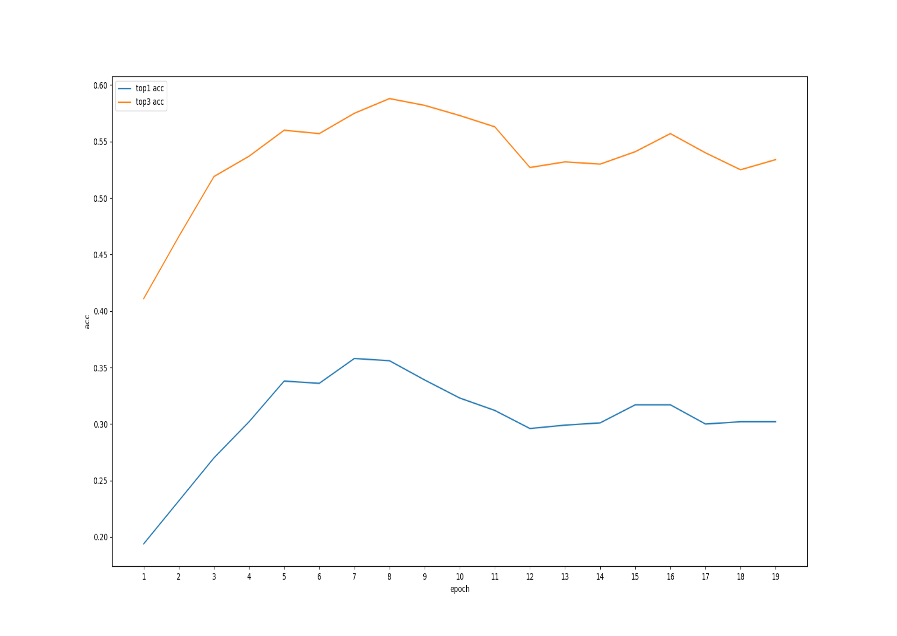
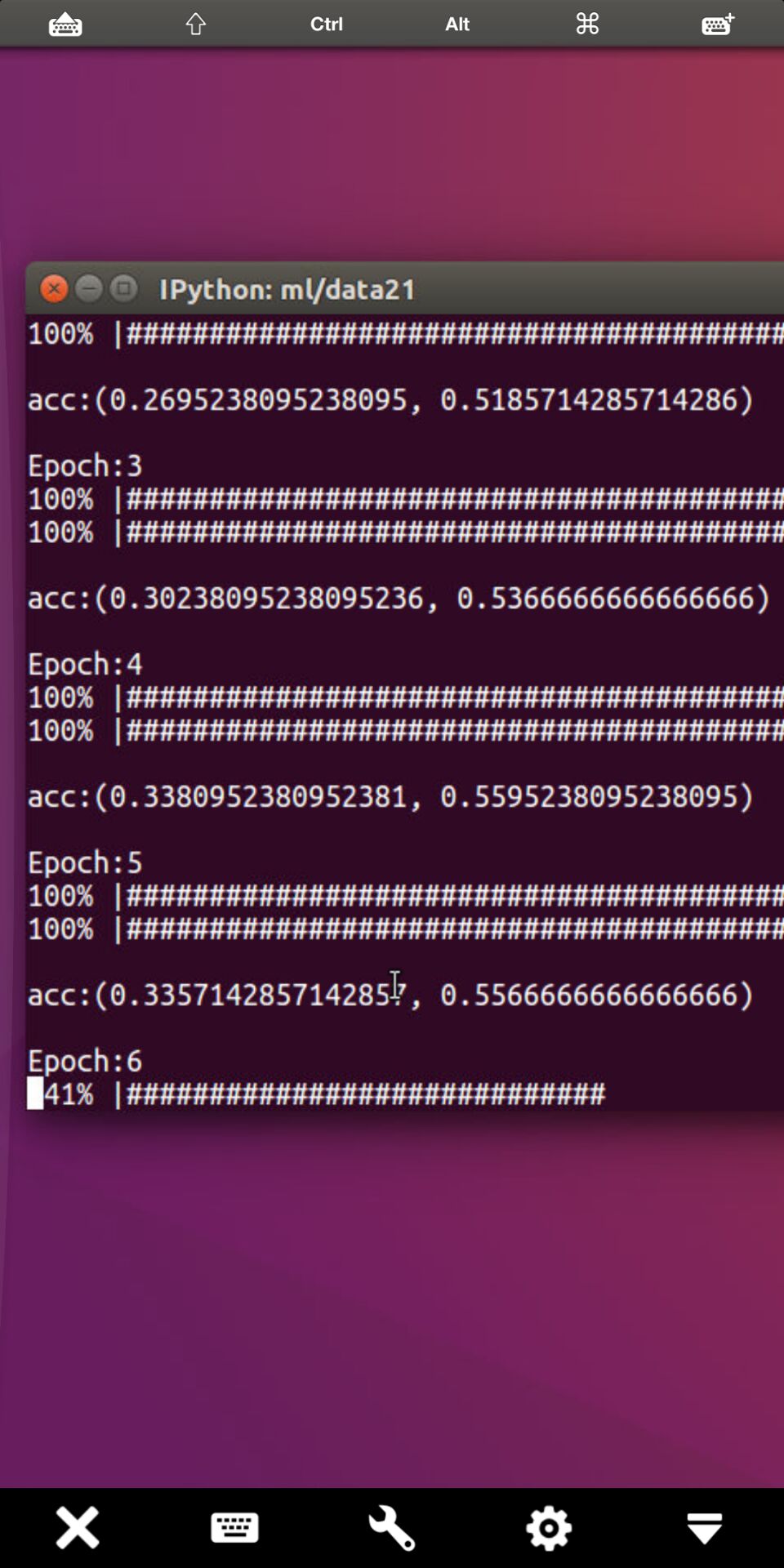
调用数据与处理函数进行读取，然后对数据进行归一化处理，归一化采取Z-score方法，并且划分出训练集和验证集。

其中Z-score方法如下：表示数据，μ表示均值，δ表示标准差。

* + - 网络结构与执行代码

定义网络结构，执行训练代码，学习率lr=0.001，每个epoch训练完成后验证一次，保存的model为其在验证集top1 acc最高。

下图为训练过程中的验证集准确率变化，第一项为top1 acc， 第二项为top3 acc：



由图可见，在epoch 7之后，验证集准确率开始下降，网络逐渐过拟合，故训练代码会自动保存epoch 7对应的模型。

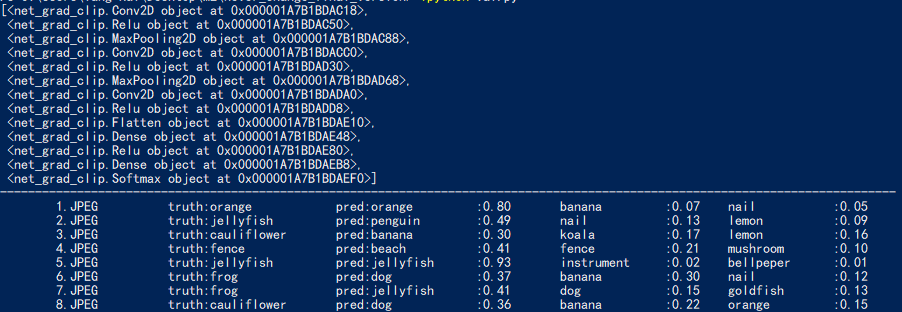
3.val.py:

* 数据读取函数：

getImageData，getDATADIR：完成对数据的读取和格式转换以及存储，然后对数据进行归一化处理，归一化采取Z-score方法。

* 网络模型代码：

定义的网络结构如上图所示，执行测试代码，其会格式化输出网络结构，预测种类及其置信度，效果如下图所示

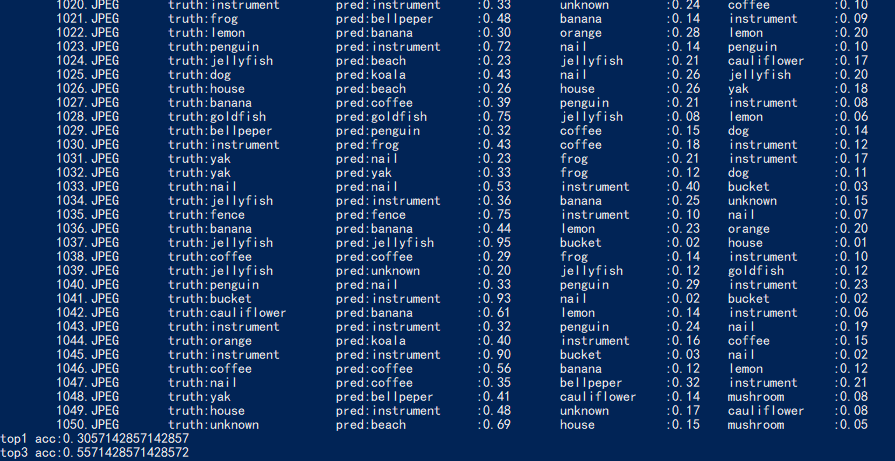


4.网络模型定义代码 ：

* conv1 = mycnn.Conv2D(3, 32, (3, 3))
* relu1 = mycnn.Relu()
* pool1 = mycnn.MaxPooling2D()
* conv2 = mycnn.Conv2D(32, 16, (3, 3))
* relu2 = mycnn.Relu()
* pool2 = mycnn.MaxPooling2D()
* conv3 = mycnn.Conv2D(16, 8, (3, 3))
* relu3 = mycnn.Relu()
* flat = mycnn.Flatten()
* dense1 = mycnn.Dense(2048, 100)
* relu4 = mycnn.Relu()
* dense2 = mycnn.Dense(100, 21)
* softmax = mycnn.Softmax()
* criterion = mycnn.CrossEntropy()
* layers = [conv1, relu1, pool1, conv2, relu2, pool2,conv3,relu3, flat, dense1,relu4,dense2, softmax]
* Convnet = mycnn.Net(layers)

# 5 实验结果

在test集中的结果为top3准确率55.71%。



# 6 结论

虽然实验中对卷积等操作进行了优化，但是实验并没有实现batch\_size内图片的并行操作，且直接使用CPU运算，所以训练数据时仍然非常耗时，一个epoch有8400张图片，训练需要大概25分钟。网络层级比较浅，有213,701个需要训练的参数，其次没有加正则项以及Dropout层可能也是导致实验结果不怎么理想的原因之一。但是在本次实验中，我们深入地理解了CNN的工作原理，也纠正了之前的一些理解偏差，并对传统卷积操作进行优化，这些都是在实验过程中了解和掌握到的，对我们以后关于深度学习的研究有重大的影响。

# 7 引用

[1] 百度百科