Python大作业实验报告——基于卷积神经网络(CNN)的性别识别系统

无03 王与进 2020010708

2021 年 9 月 11 日

目录

1 综述

本作业基于pytorch机器学习框架，通过CNN（卷积神经网络）方法对样例数据集（人脸图片）进行学习，得到性别识别神经网络，在测试集上得到了94%-95%左右的正确率。随后，笔者又在在线机器学习平台Kaggle上搭建了vgg16神经网络，借助GPU训练，得到了96%-97%左右的正确率。

此外，本系统还实现了GUI界面，使得使用者能更好地观察到神经网络对具体图片的辨识能力，并在此基础上更好地分析出现误差的原因，对模型进行调整。

本项目整体结构如下：

- lfw_funneled（数据集）
- malenames.txt（存放男性名字，便于读取）
- femalenames.txt（存放女性名字，便于读取）
- Dataloader.py （定义了MyDataLoader类，对数据的读取、分割等做了统一管理）
- Dataset.py（定义了MyDataSet类，对图像数据和标签的处理做了统一管理）
- CNN.py（主程序，定义了神经网络的结构以及训练、验证、测试的过程）
- CNN.pkl（最后一次训练所得到的参数）
- gui.py（运行图形用户界面的程序）
- run.cmd（批处理脚本，可以批量运行不同参数设置的python脚本）

此外，项目中还包含一些不同超参数下的训练结果图片，以及与本项目相关的.git、.gitignore文件等

本项目参考了一些纸质和网络教程[1][2]，以及pytorch的官方文档[3]。

2 第三方库

- Pytorch（机器学习框架）
- opencv（图像处理库）
- PIL（图像处理库）
- numpy（数据处理库）
- matplotlib（图表绘制）
- tqdm（进度条）
- termcolor（实现训练结果的可视化输出）
- tkinter（GUI实现库）

3 详细介绍

3.1 数据加载

在数据加载部分，本项目参考了助教文档给出的Dataloader实现方式，依次读入男女性的名字，遍历文件夹中的图片载入内存，并做好标签记录。随后，将这些图像数据按照4：5：1的比例分为训练集、验证集、测试集以备使用。在每次载入数据时，数据的顺序都会被打乱，防止训练结果的同质化。

本项目训练集的图片大约有5000张。为了实现Data Argumentation（数据增强），克服训练集图片数目不够的情况，同时也为了防止出现过拟合，本项目对图片进行了预处理，基本预处理方法依次如下（以下列举名称均为pytorch中提供的原生方法）：

Resize 由于性别识别仅是一个二分类问题，对图像的精度、大小等要求不是特别高，为减小运算量，将250x250的图片统一调整尺寸为50x50。

RandomRotation 随机使图片顺时针或逆时针旋转一个角度（20°以内）。

RandomHorizontalFlip 图片有50%几率水平旋转（值得注意的是，垂直翻转对神经网络的训练没有帮助，因为数据集中没有上下翻转的人像）。

ToTensor 将图像信息转换为可计算的Tensor。

Normalize 以$mean = 0.5, std = 0.5$对数据进行归一化，便于梯度下降的计算。

3.2 神经网络训练

3.2.1 超参数

本项目中主要有以下几个需要重点关注的超参数：lr（学习率）、$epoch$（学习轮数）、$batchSize$（批处理大小）和lr_loss（学习率衰减指数）。

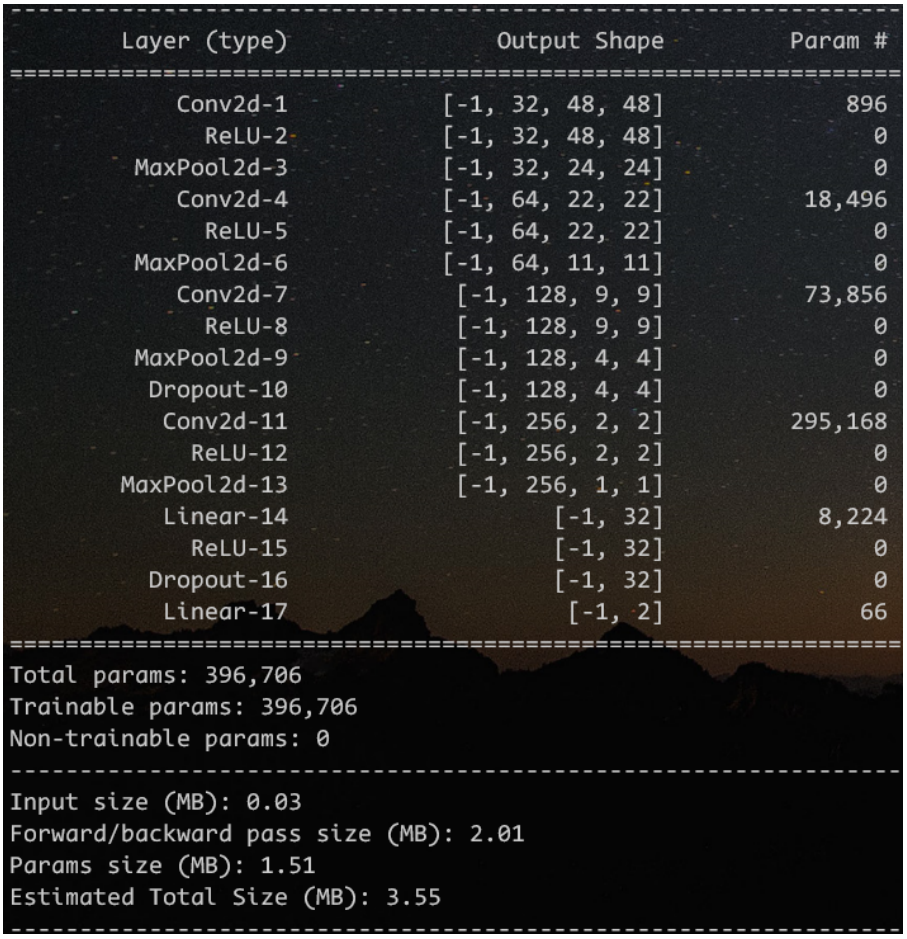
初始lr按照Adam算法的常规标准取0.001，之后每隔5个$epoch$，lr将会衰减为原来的80%。

$batchSize$的选取[4]，兼顾了$batchSize$较小时的高精确度、高泛化能力和较大时的高速度。

除了在程序中手动修改超参数外，程序还提供了命令行设定超参数的方法，方便使用者批量运行程序，对比不同超参数设置对训练结果的影响。输入`python CNN.py [lr] [batchSize] [epoch] [lr_loss]`，即可按照对应超参数运行程序。

3.2.2 网络结构

考虑到算力的问题，本项目神经网络整体上模仿LeNet[5]进行设计，但考虑到LeNet识别的是28x28的MNIST手写数字，图片信息量与大小均小于本项目，本项目结合AlexNet[6]，vggNet[7]等经典网络，以及一些现有项目[8]，对网络进行了进一步的调整。网络结构如下：



Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 48, 48]	896
ReLU-2	[-1, 32, 48, 48]	0
MaxPool2d-3	[-1, 32, 24, 24]	0
Conv2d-4	[-1, 64, 22, 22]	18,496
ReLU-5	[-1, 64, 22, 22]	0
MaxPool2d-6	[-1, 64, 11, 11]	0
Conv2d-7	[-1, 128, 9, 9]	73,856
ReLU-8	[-1, 128, 9, 9]	0
MaxPool2d-9	[-1, 128, 4, 4]	0
Dropout-10	[-1, 128, 4, 4]	0
Conv2d-11	[-1, 256, 2, 2]	295,168
ReLU-12	[-1, 256, 2, 2]	0
MaxPool2d-13	[-1, 256, 1, 1]	0
Linear-14	[-1, 32]	8,224
ReLU-15	[-1, 32]	0
Dropout-16	[-1, 32]	0
Linear-17	[-1, 2]	66

Total params: 396,706
Trainable params: 396,706
Non-trainable params: 0

Input size (MB): 0.03
Forward/backward pass size (MB): 2.01
Params size (MB): 1.51
Estimated Total Size (MB): 3.55

图 1: 网络结构

网络由4层卷积层和2层全连接层构成，输入通道数为3，输出结果有2种。卷积核的尺寸均为3x3，卷积步长为1。除此之外，本网络的主要特点如下：

激活函数 未采用LeNet中的Sigmond函数，而采用AlexNet中ReLu函数，防止Sigmond函数可能会带来的梯度损失。

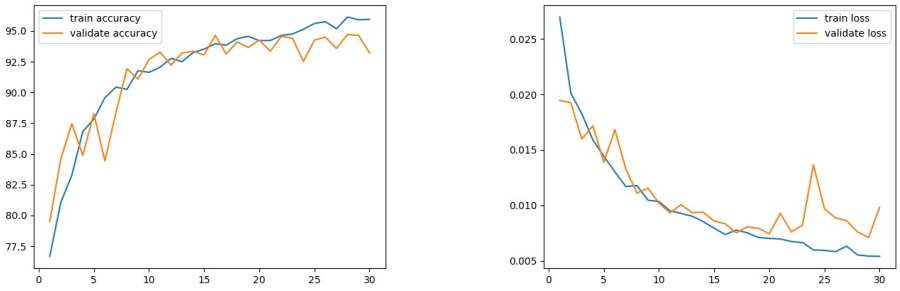
梯度下降算法 采用目前较为主流的Adam算法[9]。

参数初始化 由于激活函数为Relu函数，所以采用He方法[10]初始化参数。He方法是指当前一层的节点数为n时，参数使用标准差为$\sqrt{\frac{2}{n}}$的高斯分布。

损失函数 采用分类问题中常用的交叉熵函数（CrossEntropyLoss）。

dropout 在第3层卷积层和第1层全连接层之间，分别设置了0.25和0.5的DropOut[11]层，以便减小模型过拟合的程度。

batch Normalization 在每个卷积层对数据进行正则化，可以加快学习速度，并在一定程度上规避初始值设置不合理可能带来的弊端。



(a) accuracy (b) loss

图 2: 某次测试结果

3.2.3 训练过程

3.3 神经网络验证

为了实时监控网络在训练过程中识别精度的提升速度、观察网络是否发生过拟合，以确定最佳的epoch轮数，本项目在每次epoch后都会对验证集的数据进行验证。

每次测试精度（$train\_acc$）、测试损失（$train\_loss$）、验证精度（$validate\_acc$）和验证损失（$validate\_loss$）都会被记录下来，并通过matplotlib绘制为折线图并保存，以评估不同训练方法的优劣。

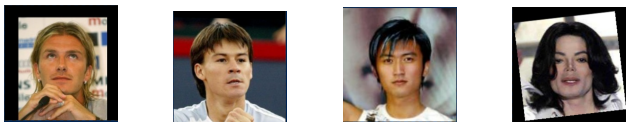
另外，每次测试和验证的过程都借助tqdm（进度条库）进行装饰，方便使用者了解训练进度。

3.4 神经网络测试

在测试时，测试图片的图像信息与图像名字会被一并读入。经过神经网络运算后，控制台将会一并输出图片的绝对路径和预测结果、正确答案（正确将显示为绿色，错误将显示为红色），使用者可以直接在终端点击图片路径来查看图片，以便及时了解预测出错的图片状况。

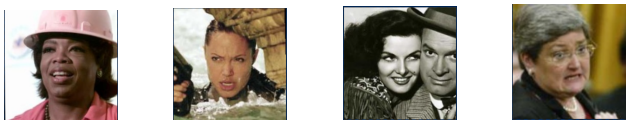
```
actual:male     predicted:male
E:\vscode-python\MachineLearning\lfw_funneled\Javier_Vazquez\Javier_Vazquez_0001.jpg
actual:male     predicted:male
E:\vscode-python\MachineLearning\lfw_funneled\Ricardo_Maduro\Ricardo_Maduro_0002.jpg
actual:male     predicted:male
E:\vscode-python\MachineLearning\lfw_funneled\Hernan_Crespo\Hernan_Crespo_0001.jpg
actual:male     predicted:male
                                ing\lfw_funneled\Yogi_Berra\Yogi_Berra_0001.jpg
在编辑器中打开文件 (Ctrl + 单击)
E:\vscode-python\MachineLearning\lfw_funneled\Gunter_Pleuger\Gunter_Pleuger_0005.jpg
actual:male     predicted:male
E:\vscode-python\MachineLearning\lfw_funneled\Sarah_Michelle_Gellar\Sarah_Michelle_Gellar_0002.jpg
actual:female   predicted:female
E:\vscode-python\MachineLearning\lfw_funneled\Ruben_Wolkowyski\Ruben_Wolkowyski_0001.jpg
actual:male     predicted:male
test accuracy 94.8%
 86181@LAPTOP-FNHRJFBO  E:\  MachineLearning  master ≡/z ~4 ?7
~#@>
```

图 3: 测试效果图例



(a) (b) (c) (d)

图 4: 实为男性，却被判断为女性的样例



(a) (b) (c) (d)

图 5: 实为女性，却被判断为男性的样例

具体分析测试结果，造成神经网络误判的原因主要有两点：

1. 网络主要基于头发长度对男女进行判断，对面部要素分析不够，容易将长发男和短发女误判。
2. 图像中有其他人像干扰。

3.5 图形用户界面

本项目的GUI框架基于tkinter。tkinter是Python的标准Tk GUI工具包的接口，基本界面如下：

图 6: 界面展示

方法介绍：

next 切换为下一人的图片。名字也会随之改变。

last 切换为上一人的图片。名字也会随之改变。

test one 预测界面上所显示的人的性别。

test all 测试网络在所有图片上的准确率。

3.6 GPU部署和训练

为取得更好的训练效果和更快的训练速度，笔者尝试将神经网络部署到Kaggle在线机器学习平台上进行训练。经测试，原模型在Kaggle上仅需大约200s就可将网络模型训练完毕，仅为本地CPU训练时间的$\frac{1}{9}$。

为了让系统得识别精度更大，笔者尝试搭建了vgg16神经网络。相比于较为简单的LeNet和AlexNet，vgg16有以下特点：

层数高 有13个卷积层和3个全连接层。但其实现并不复杂。vgg16的卷积层的构造有一定的规律——被称为“vggBlock”，利用这一规律可以制造5个结构较为相似的vggBlock部署在网络中。

以深度替代广度 vgg16没有采用较大尺度的卷积核（LeNet中5x5的卷积核、AlexNet中11x11的卷积核），而是采用多层3x3的卷积核。多层3x3卷积核与单层大尺寸卷积核的Receptive Field相同，但对图像的表现力更强（代价是占用较高的内存）。

vgg16较大的可训练参数量，使得它在本地CPU环境下的训练较为困难，但是在GPU平台上，训练耗时仅为15min左右。使用$lr = 0.00005, lr_loss = 0.7, epoch = 30, batchSize = 20$的参数进行训练，正确率可达96% 97%以上。

```
.....................................................................
epoch 29:
train: loss 0.00161, train accuracy 98.9%
validate: loss 0.00782, validate accuracy 96.7%
.....................................................................
epoch 30:
train: loss 0.00169, train accuracy 98.8%
validate: loss 0.00908, validate accuracy 96.1%
.....................................................................
```

图 7: vgg16的学习效果

Kaggle项目地址（原版）： https://www.kaggle.com/yujinwang/sexrecognition
Kaggle项目地址（vgg16）：https://www.kaggle.com/yujinwang/sexrecognition-vgg16

4 实验结果

为确定效果最佳的超参数，本项目进行了多项对比试验，并将一些实验结果展示如下。

4.1 学习率

学习率过低会导致模型拟合速度过慢，学习率过高会模型难以收敛。此外学习率的衰减对模型训练精度也有影响。

下列测试中，分别取$lr = 0.005, 0.002, 0.001, 0.0005$，对模型进行30轮训练：

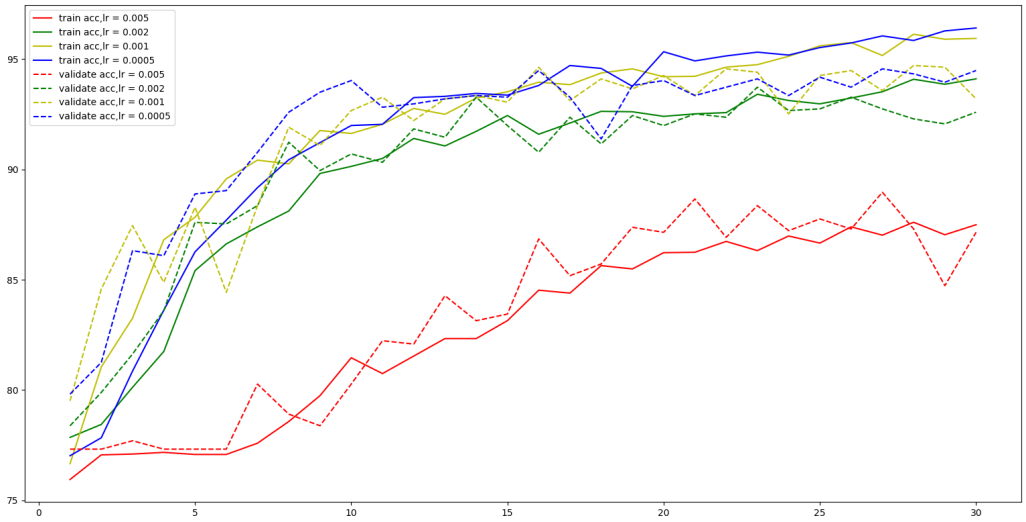


图 8: 不同学习率的学习效果（$lr\_loss = 0.9, batchSize = 20, epoch = 30$）

可以看到，$lr = 0.005$时训练效果明显不如后三者，此时学习率偏大，学习率为0.001时较为合适。

下列测试中，分别取$lr_loss = 1, 0.9, 0.8, 0.7$，对模型进行30轮训练：

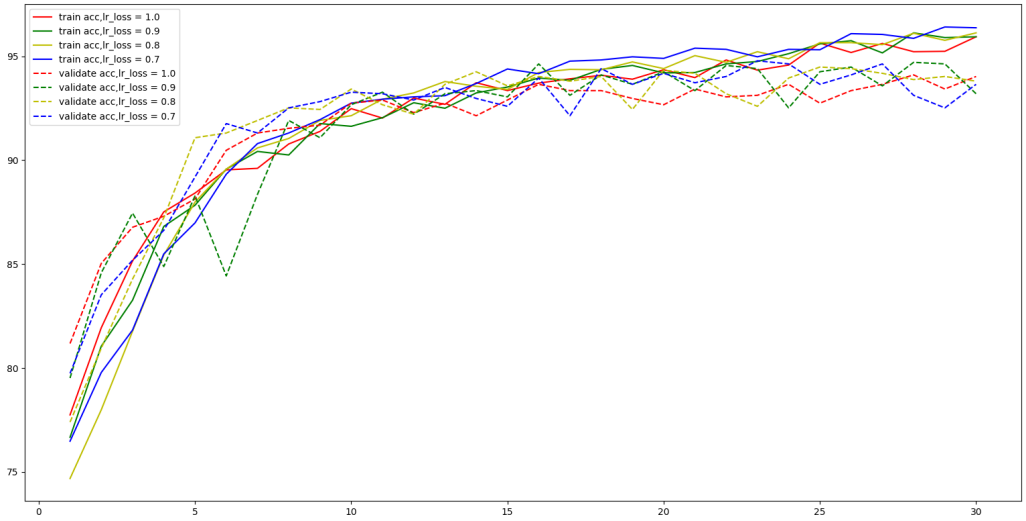


图 9: 不同学习率衰减值的学习效果（$lr = 0.001, batchSize = 20, epoch = 30$）

总体而言，四种衰减幅度相差不大。

4.2 epoch

受制于算力限制，网络精度等原因，当训练的epoch进行到一定数目后，模型的测试精确度就难以提升了。此时继续训练，不但浪费时间，而且可能会加重过拟合。以下为对模型训练60轮训练的数据：

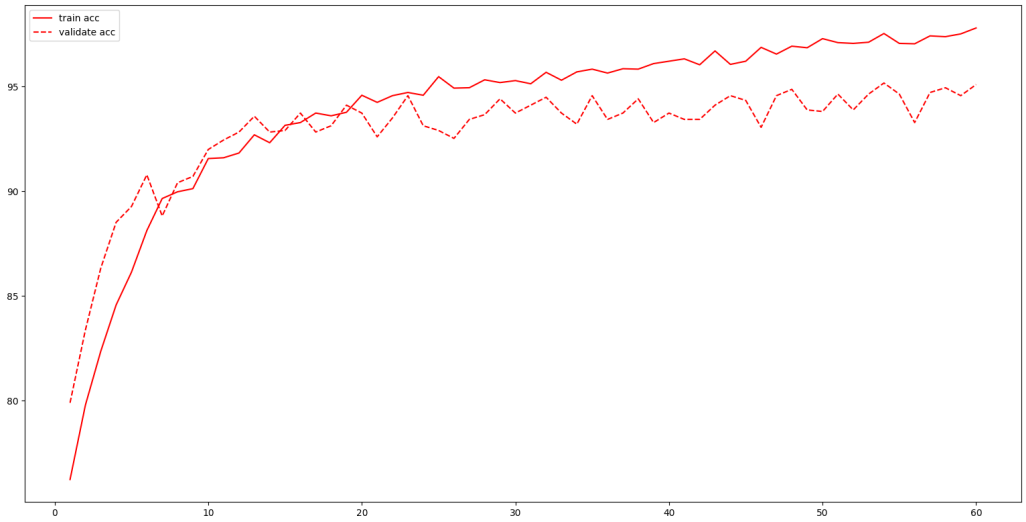


图 10: 60轮训练（$lr = 0.001, lr\_loss = 0.9, batchSize = 20$）

在测试集上，准确率有小幅度提升，但是验证集准确率提升幅度并不大，说明过大的epoch在本项目中并无太大必要。

4.3 batch size

下列测试中，分别取$batchSize = 10, 20, 50, 100, 200$，比较训练效果：

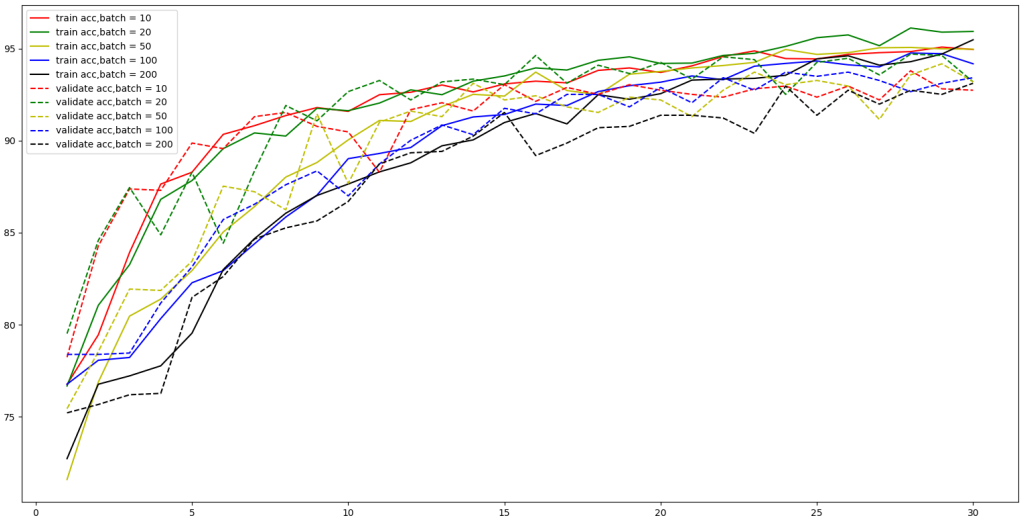


图 11: 不同batch size的训练结果（$lr = 0.001, lr\_loss = 0.9, epoch = 30$）

总体而言，*batchsize*越小，训练效果越好，但此差距随着epoch轮数的增大而减小。

4.4 梯度下降法

前文提到，Adam算法可以适应大部分的学习场景，但是一些论文指出[12]，在某些特殊情况，带有momentum的SGD算法可能会更胜一筹。以下对Adam算法和momentium算法的训练效果进行了比较：

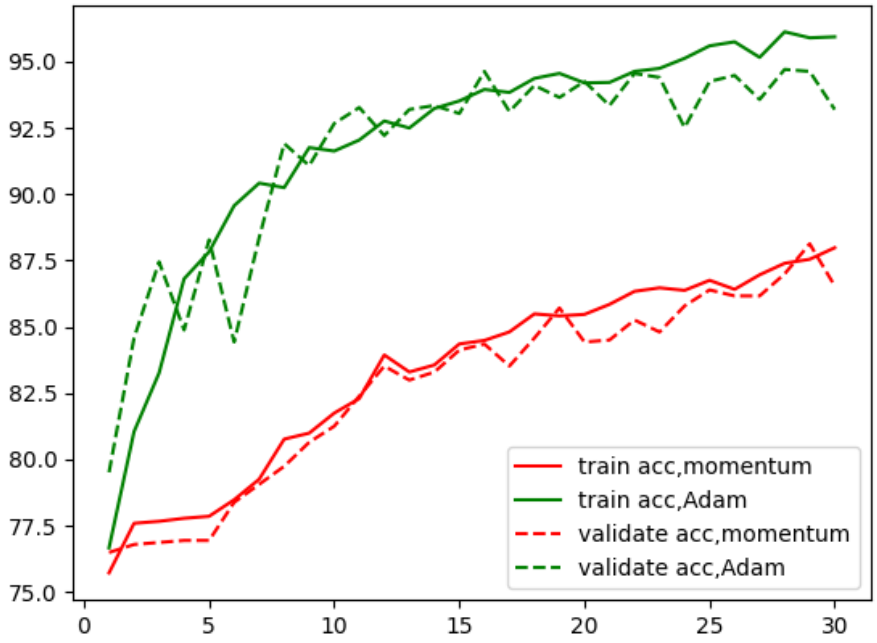


图 12: 梯度下降方法比较

在此场景中，显然Adam算法更胜一筹。

4.5 结论

综合上述实验可知，最佳超参数设置应为$lr = 0.001, lr_loss = 0.9, epoch = 30, batchSize = 20$，此外梯度下降算法采用Adam。经过20-30轮训练后，在测试集上的准确率可以达到94%-95%。

参考文献

[1] 斋藤康毅. 深度学习入门：基于*python*的理论和实现. 人民邮电出版社, 2018.

[2] Mu Li et al. 动手学深度学习. 人民邮电出版社, 2020.

[3] pytorch official documents. `https://pytorch.org/docs/stable/index.html`.

[4] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836, 2016.

[5] Y. LeCun, L. Bottou, Yoshua Bengio, and P. Haffner. Gradient-based learning applied to document recognition. 1998.

[6] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[7] K. Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[8] Gil Levi and Tal Hassner. Age and gender classification using convolutional neural networks. *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 34–42, 2015.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[12] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning, 2018.