深入MNIST项目卷积神经网络

项目概括：本项目采用卷积神经网络对MNIST手写数字进行分类识别。项目开发使用工具是pycharm,项目开发语言采用Tensorflow,项目采用的数据是Tensorflow 中mnist数据集。

项目的目的是通过卷积神经网络对mnist数据集进行训练，建立模型，用于识别新的手写数字，即从图片中识别出手写的数字。

**一、项目设计**

**1.卷积神经网络简介**：卷积神经网络在传统的神经网络中提出的，通过滤波器提取局部特征，参数共享方式，提高神经网络的性能。一般卷积神经网络包括卷积层和池化层。

简单卷积神经架构设计：

输入层--卷积层--激活层--池化层--全连接层--输出层。

输入数据

卷积层Conv

激活层active

池化层pool

全连接层FC

预测得分logist

计算模型

卷积层计算公式

输入数据体的尺寸为W1xH1xD1

4个超参数

滤波器的数量K 滤波器的空间尺寸F

步长S 零填充数量P

输出数据体的尺寸为：W2xH2XD2,其中

W2=(W1-F+2P)/S+1

H2=(H2-F+2P)/S+1

D2=K

池化层计算公式

输入数据体的尺寸为W1xH1xD1

2个超参数

滤波器的空间尺寸F

步长S

输出数据体的尺寸为：W2xH2XD2,其中

W2=(W1-F)/S+1

H2=(H2-F)/S+1

D2=D1

“data”

W\*H\*D

24\*24\*K

“data”

W\*H\*D

28\*28\*1

“conv2d”

K F=5\*5

S=1 p=0

“activate”

relu

“data”

W\*H\*D

24\*24\*K

“pool2d”

Maxpool

F=2\*2 S=2

“data”

W\*H\*D

1\*1\*10

FC\_linear

Unist=10

“data”

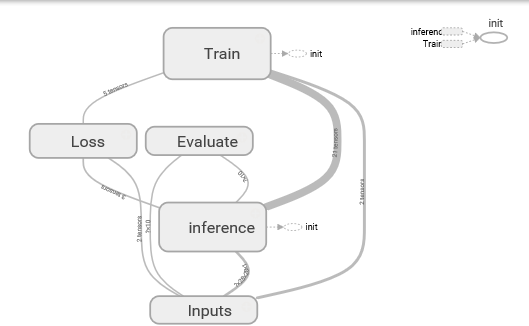
W\*H\*D

12\*12\*K

Softmax-

Cross\_Entrcopy

1. **计算图设计**：input参数设置--->inferenc前向推断-->Loss计算损失-->Train模型训练及反向传播更新参数-->Evaluate模型预测



1. **项目编程**

**1.设置算法超参数**

Learning\_rate=0.01 #学习率

Train\_epochs=1 #训练轮数

Batch\_size=100 #训练批次的大小

Disp;ay\_step=10 #每多少步打印训练和验证信息

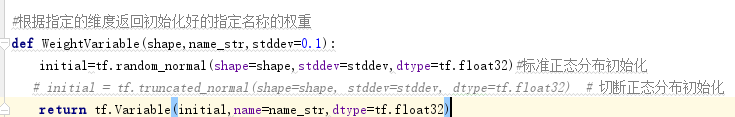
#NetWork Param

N\_input =784 # 输入数据的特征数

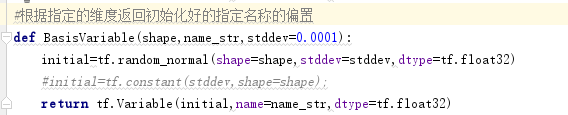
N\_classes=10 # 输出数据的分类数

**2.定义函数**

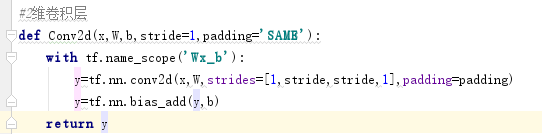
#定义权重初始函数



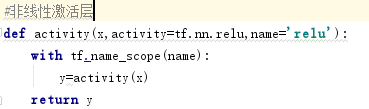
#定义偏置初始函数



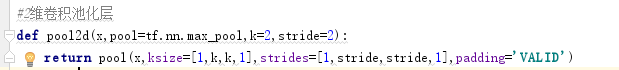
#定义卷积层



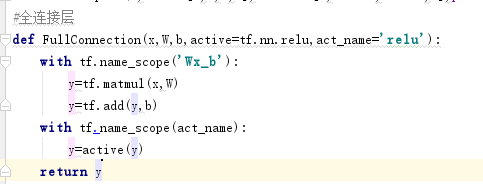
#定义非线性层



#定义池化层

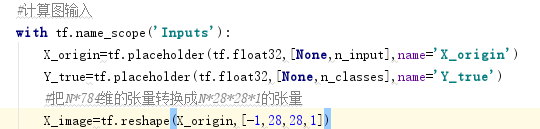


#定义全连接层



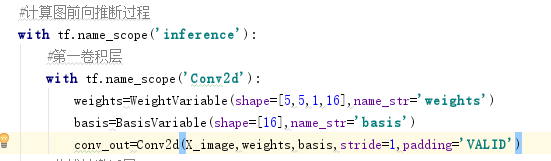
1. **根据计算图设计构建计算图**

#输入数据初始化(占位符占位，对输入数据一维到二维转换)

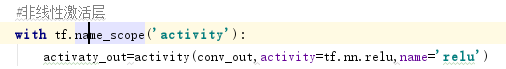


#前向推断过程

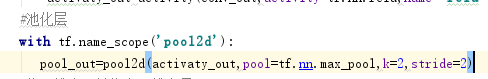
#搭建卷积层(根据滤波器大小，步长和数量k，初始化权重，根据k初始化偏置，把数据，权重，偏置传给卷积函数)



#搭建非线激活性层(这一层实际可以和卷积层合在一起，把卷积层数据，激活函数传给非线性激活函数)



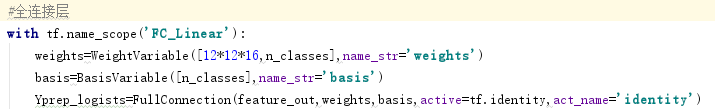
#搭建池化层(池化层需要把池化函数，滤波器大小，步长，传给池化函数，完成降 维操作)



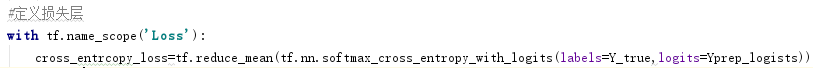
#将二维卷积转换一维向量（下一个层是全连接层，这里进行转换一下）



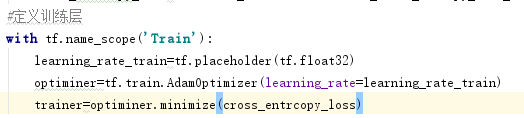
#搭建全连接层(目前设计只有一个全连接层，这里的激活函数定义为线性激活函 数，同时需要初始化权重，偏置的参数)



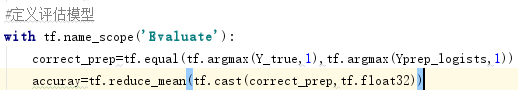
#定义完前向推断，现在定义损失函数(损失函数采用交叉熵损失，用于单标签多分类)



#定义训练层(根据tensorflow 给定的优化器函数，对数据进行交叉熵损失进行方向传播，然后根据梯度下降法计算修改参数)



#定义评估层(根据预测最大概率和实际最大概率所在的index，是否一致判断是否预测准确，并计算准确率)



**4.运行计算图**

(1)加载数据集

(2)启动会还

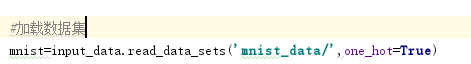
(3)一轮一轮的训练模型

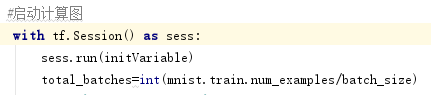
在每一轮中分多批次喂给数据

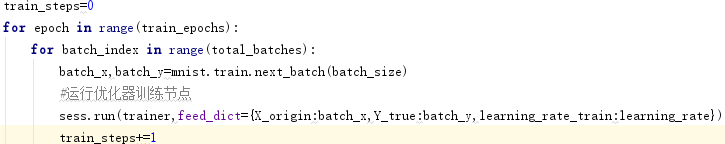
在每个批次上运行训练节点，训练模型

经过若干批次后，评估当前模型，计算训练集和验证集上的损失值，准确率

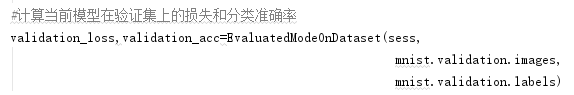
(4)在测试集上评估模型：损失值，准确率



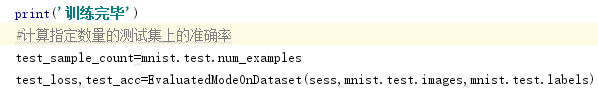


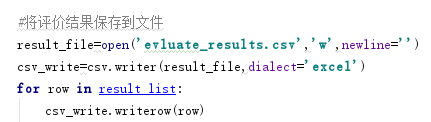








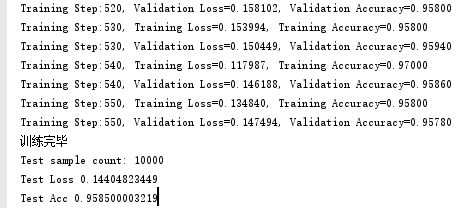




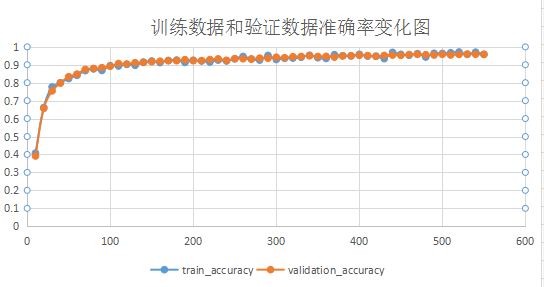
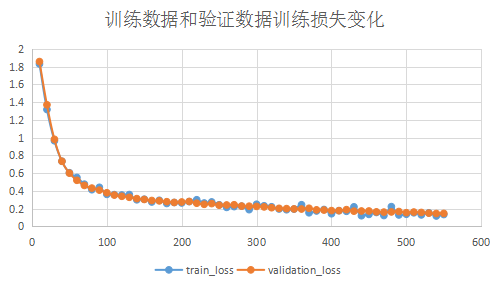
1. **项目评估调参**

**1.项目评估**

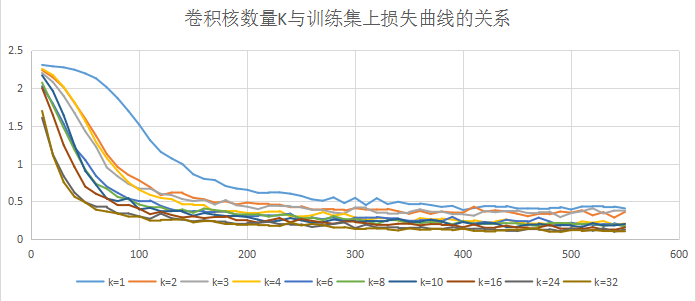
查看打印的损失和准确率，准确率高达96%



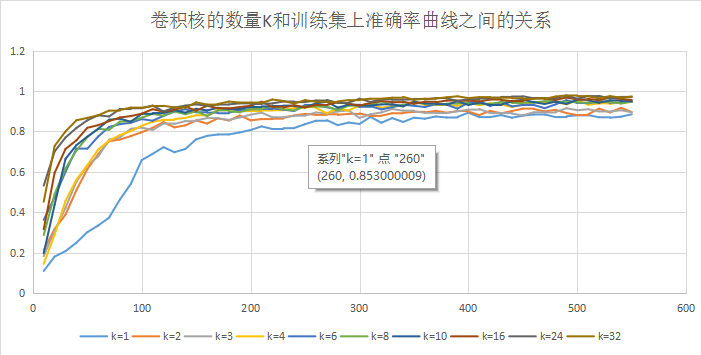
把数据写人excle文件，并根据数据，绘制损失值和准确率变化图



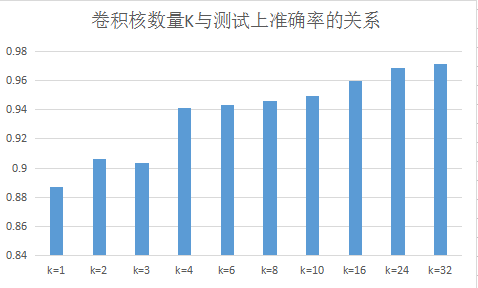
1. **卷积滤波器数量k，即增加卷积层的宽度，看网络性能变化**



卷积核的数量k的增加，损失变化越快



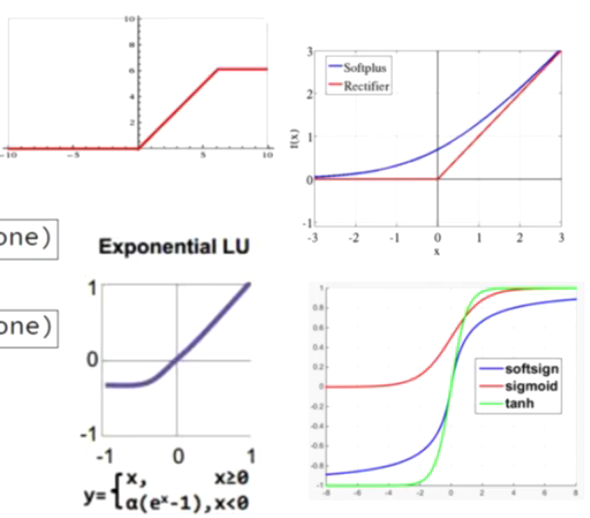
卷积核的数量k的增加，准确率提高的也越快

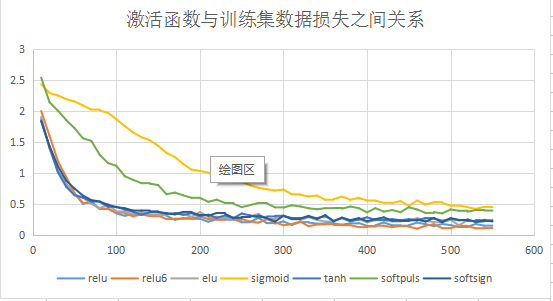


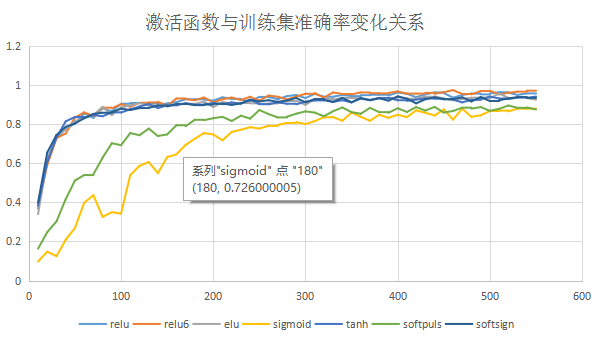
卷积核数量与准确率呈现正相关

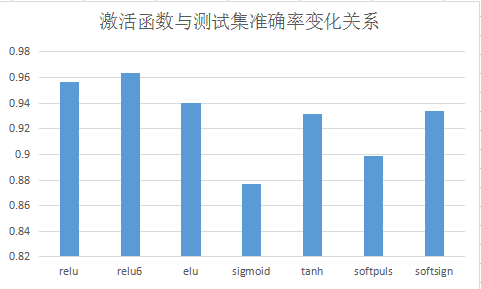
1. **改变激活函数对网络影响**

目前conv2d 5x5x1 k =16 七个激活函数，注:全连接层为线性激活函数



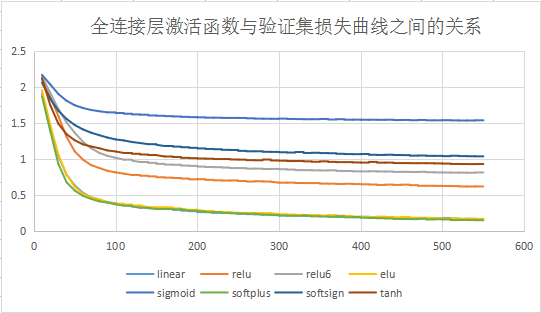


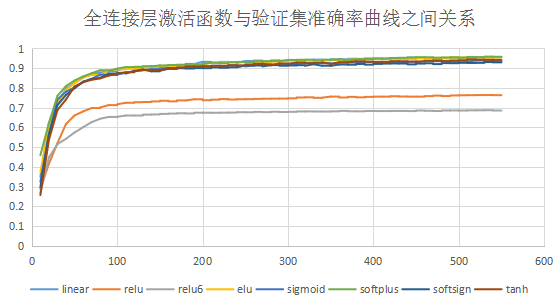


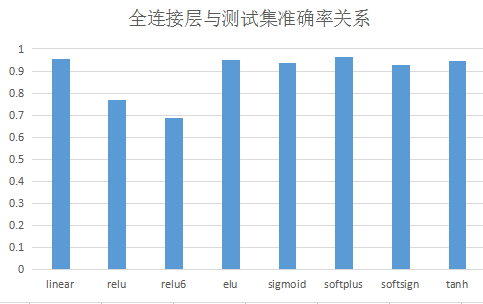


在众多的激活函数中，在目前的设计模型中relu,relu6,elu 表现比较优异

下面是conv2d 层的激活函数固定，全连接层激活函数进行改变





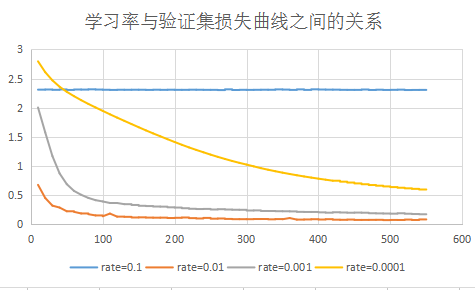


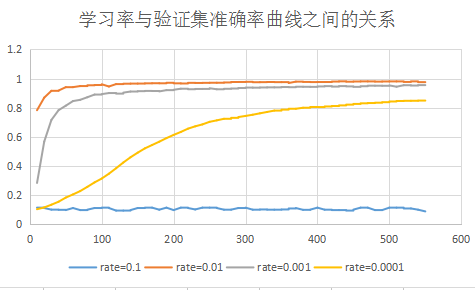
全连接层激活函数变化中relu和relu6表现并不好

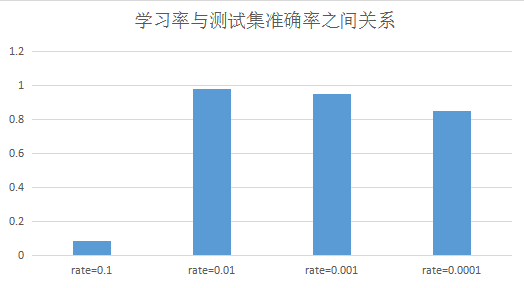
1. **学习率对网络性能的影响**

网络模型：conv2d+relu+maxpool+linear\_fc

修改学习率：0.1，0.01，0.001，0.0001



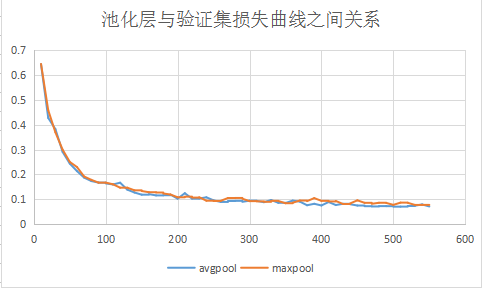


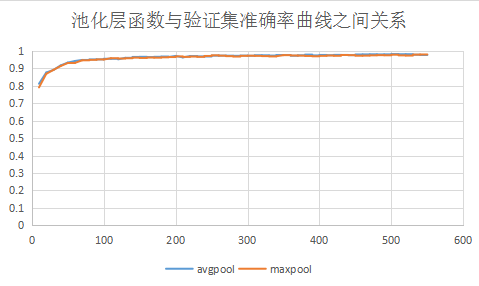


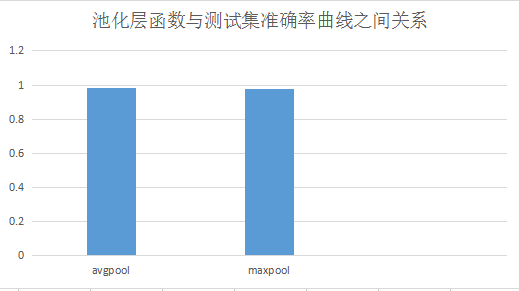
学习率对于模型的影响是非常大的，因此选择合适的学习率非常重要

1. **池化层的maxpool 和avgpool的差异对比**

训练模型为：conv2d+relu+pool(maxpool/avgpool)+fc\_linear learning\_rate =0.01





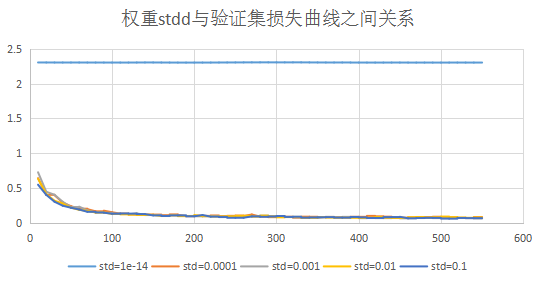


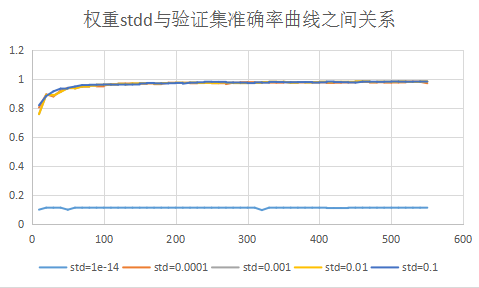
在当前的模型下，maxpool和avgpool 差异不大，在learning\_rate=0.01情况下，测试集上的准确率都能达到98%左右

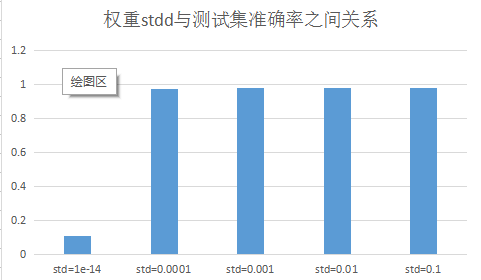
1. **权重初始化，当前初始化，采用标准正态分布，stddev的值对初始化有一定影响。**

网络模型：conv2d+relu+maxpool+fc\_linear learning\_rate=0.01

Stddev：1e-14,0.0001,0.001,0.01,0.1







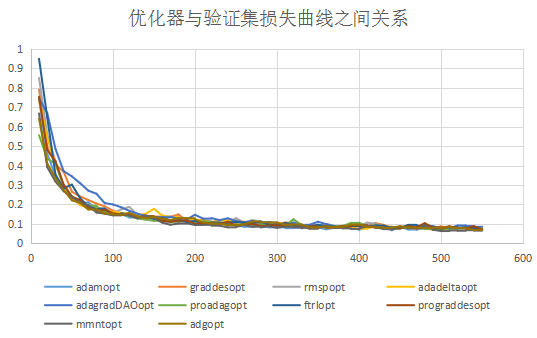
从图上可以看出，当std=1e-14时，损失不能降低，准确率维持在10%左右。0.0001-0.1之间损失和准确率变化没有特殊。

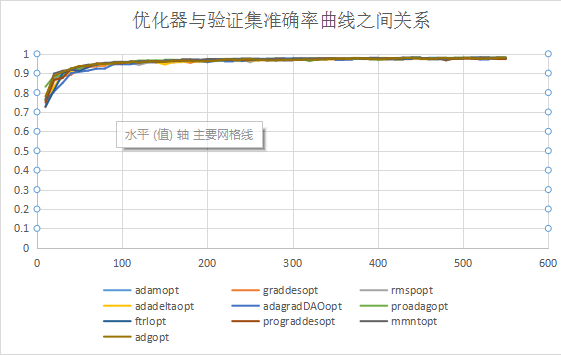
1. **tf.train时的优化器对网络影响。**

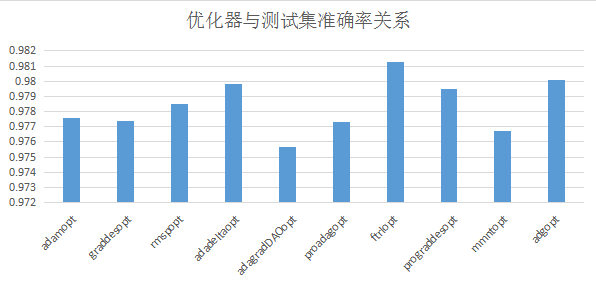
网络模型：conv2d+relu+maxpool+fc\_linear learning\_rate=0.01 stdd=0.0001

一共有十个优化器，下面对比不同优化器对网络影响。

根据下面图可以发现，在目前网络模型下，十个优化器，在验证集上都能有效的降低损失，提高准确率，相差并不多。







1. 项目总结

项目架构采用简单卷积神经网络，一个卷积层，一个池化层，一个全连接层。项目架构虽然简单包含了卷积神经网络的基本构件。项目进行了大量的调参工作，对比不同参数对损失值和准确率的影响。其中卷积核数，激活函数，学习率对项目的影响有显著的规律。

根据调参数据优化网络模型为：

Con2d+relu6+maxpool+fc\_linear learning\_rate=0.01 stdd=0.001 opti=FtrlOptimizer k=32

在当前模型下运行三次求其平均值绘制下图

