针对网络图片的优化方法效果比较分析报告

视频质量组实习生 张佳瑄

目录

一、实验目的

在不同领域数据分布各不相同，所以，同一优化方法在不同数据集上，优化效果会有差异。本篇报告针对文字识别领域的网络图片数据，测试了各种优化器的表现，分析各种优化器在网络图片识别领域的特点，为后续实验做基础。

二、实验设置

1. 数据集

训练集：模拟网络图片生成数据集。使用 75 种字体，生成约 75 万张合成数据。为拟合网络图片分布，其中图片背景 3/5 为网络图片，1/5 为纯色图片，1/5 为白色背景。其中文本内容取随机语料，9/10 的文本为 1-10 个字符，1/10 为 11-20 个字符的长文本。对生成图片加后处理，其中抽取 1/10 图片加入高斯噪声，1/20 加入椒盐噪声，字体阴影特效为 1/10，字体描边特效为 1/10。

示例：

测试集：

a. 泡泡真实数据（水平方向），共 128087 张。

示例：

b. 阿里天池竞赛，淘宝图片，经筛查清洗后，共 133787 张。

示例：

2. 网络结构

使用 CRNN 网络，将输入图片 resize 到 32*529，不足背景用黑色填充。输入识别网络，网络结构如下：

Type	Configurations
Transcription	-
Bidirectional-LSTM	#hidden units:256
Bidirectional-LSTM	#hidden units:256
Map-to-Sequence	-
Convolution	#maps:512, k:2 × 2, s:1, p:0
MaxPooling	Window:1 × 2, s:2
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
MaxPooling	Window:1 × 2, s:2
Convolution	#maps:256, k:3 × 3, s:1, p:1
Convolution	#maps:256, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:128, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:64, k:3 × 3, s:1, p:1
Input	W × 32 gray-scale image

3. 训练设置

训练环境：python2.7，cuda8.0，pytorch1.1.0，warpctc

具体代码与设置：http://gitlab.qiyi.domain/zhangjiaxuan_sx/paopao-try/tree/master

三、实验结果

1. SGD 与 Momentum

由于数据集较为复杂，参数初始值和学习率选取困难，不能进行正常训练。

实验参数设置:

a. SGD: optimizer=torch.optim.SGD(params,lr=0.1);

optimizer=torch.optim.SGD(params,lr=0.01)

optimizer=torch.optim.SGD(params,lr=0.001)

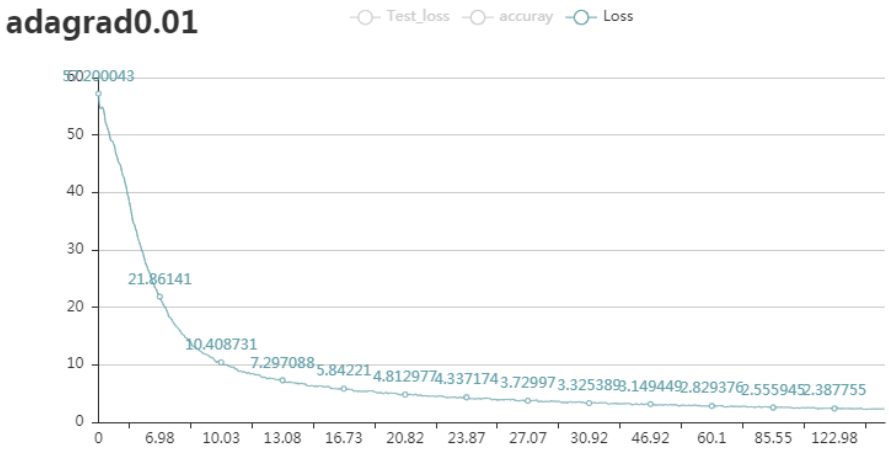
b. Momentum: optimizer=torch.optim.SGD(params,lr=0.1,momentum=0.9);

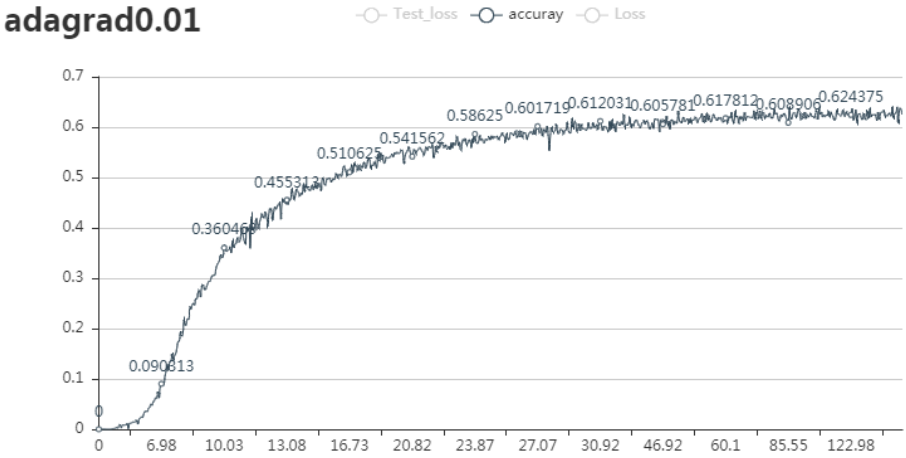
optimizer=torch.optim.SGD(params,lr=0.01,momentum=0.9);

optimizer=torch.optim.SGD(params,lr=0.001,momentum=0.9)

2. Adagrad

对学习率加了约束,使用之前全部的累加梯度作为分母,随着训练次数增加，累计梯度上升，学习率下降。初期训练 loss 下降速度很快，且收敛速度较快，但 loss 收敛值较大。下图为训练 loss 下降图和在验证集上的准确率的变化图。

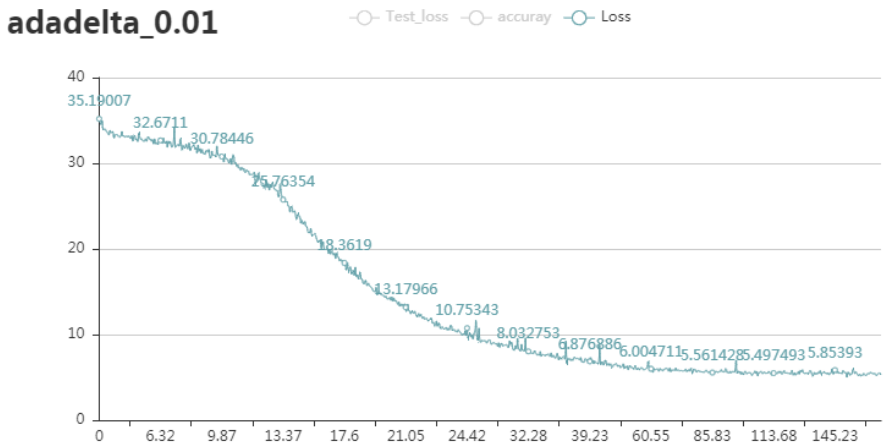


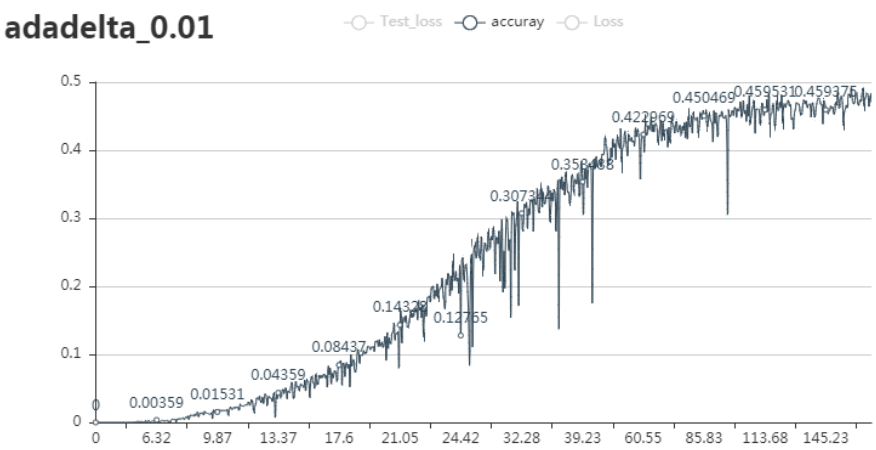


实验参数设置：optimizer=torch.optim.Adagrad(params,lr=0.01)

3. Adadelta

Adadelta 对 Adagrad 做了改进，将累计梯度从全部变为一个窗口期的累计梯度，不再依赖全局学习率。使用 adadelta 优化器，尝试设置不同学习率，最后选择将 lr 设置为 0.01。由于 adadelta 优化器中，lr 并不表示学习率，而代表在 adadelta 被应用到参数更新之前对它缩放的系数。由于 lr 设置较小，所以训练收敛速度过慢，且训练后期 loss 下降缓慢，很难判断收敛时间。下图为训练 loss 下降图和在验证集上的准确率的变化图。

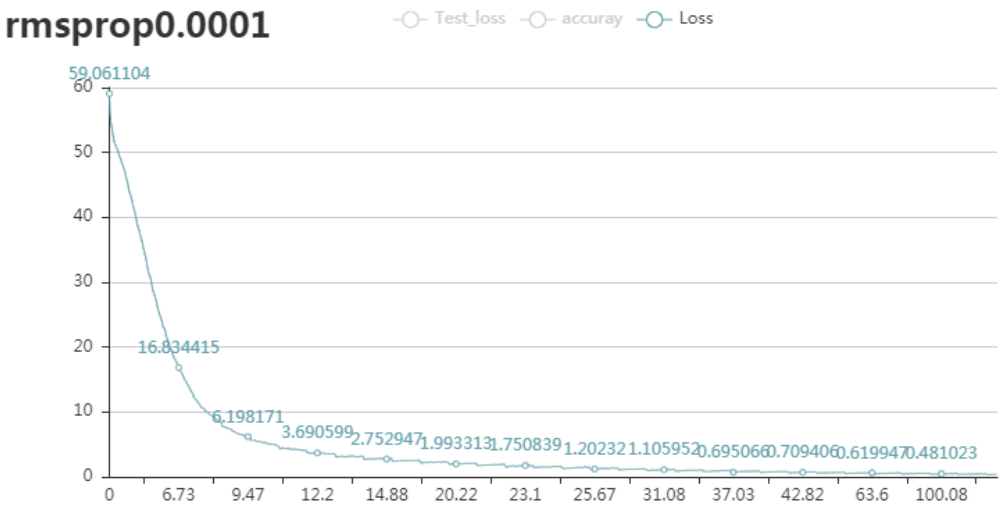


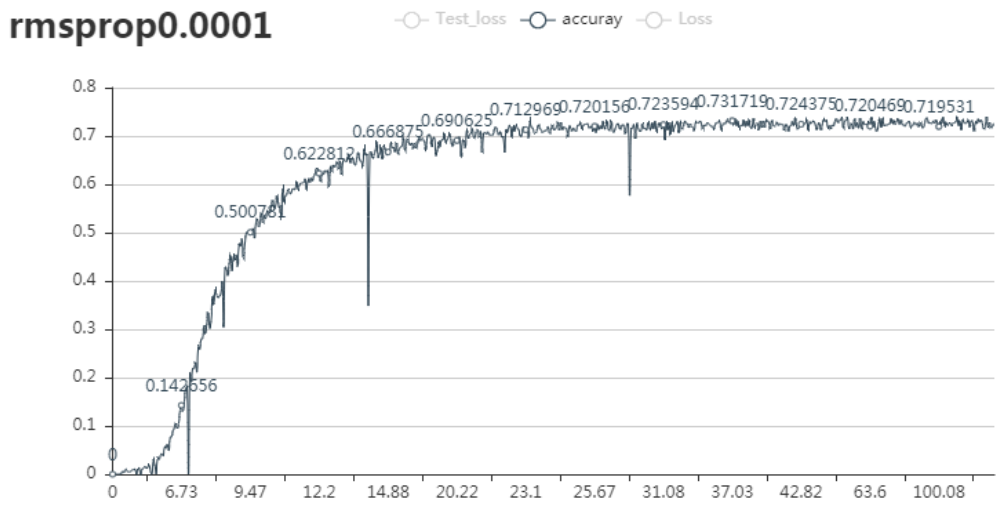


实验参数设置: optimizer=torch.optim.Adadelta(params,lr=0.01,rho=0.9,eps=1e-06)

4. RMSprop

RMSprop 在 RMSprop 基础上加了衰减系数，但仍依赖全局学习率。使用 RMSprop 优化器，首先选用 lr=0.001，训练中期出现 loss 为 inf 情况。减小学习率后，重新训练。可以看到 RMSprop 收敛速度较快，且收敛到的 loss 也很低。下图为训练 loss 下降图和在验证集上的准确率的变化图。

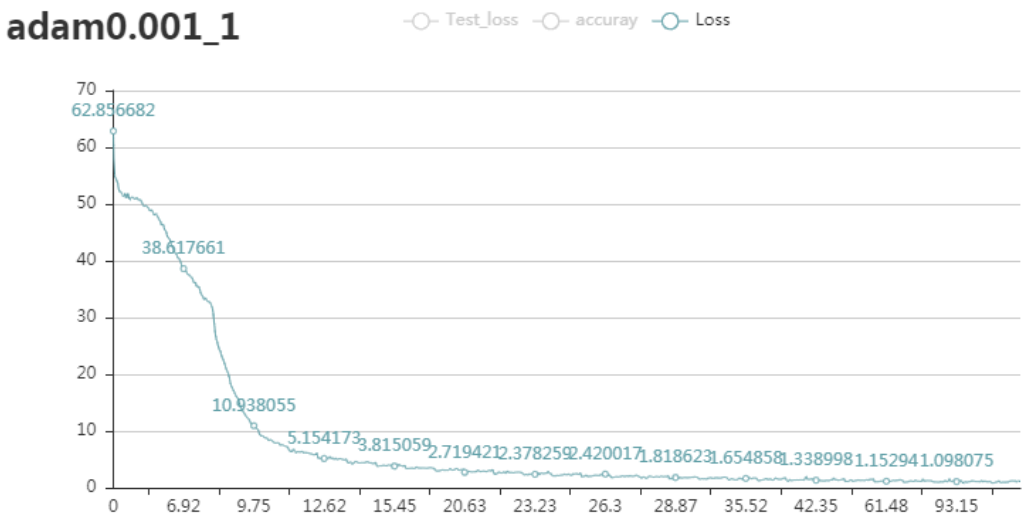


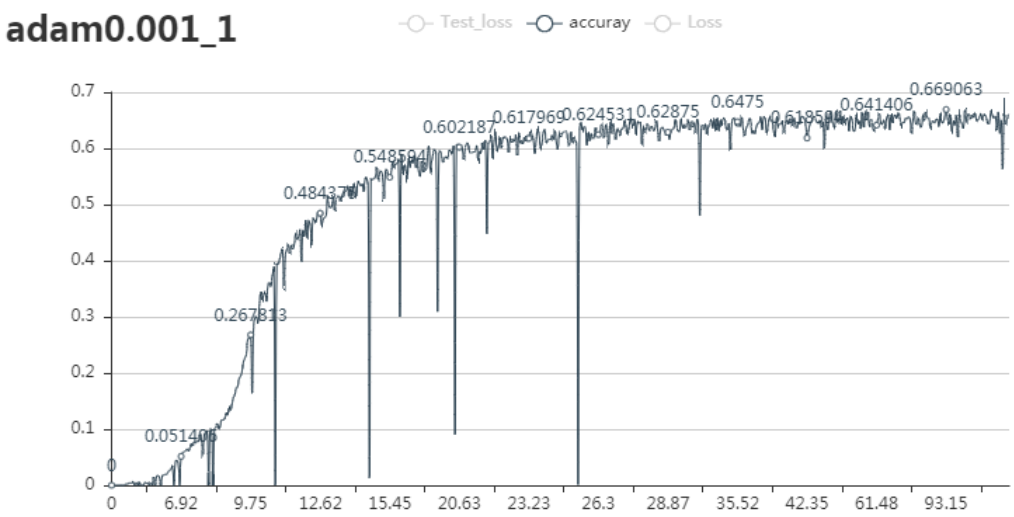


实验参数设置: optimizer=torch.optim.RMSprop(params,lr=0.0001, alpha=0.99,eps=1e-06)

5. Adam

Adam 是带有动量的 RMSprop 优化器，利用梯度的一阶矩估计和二阶估计动态调整每个参数的学习率，并加入偏基矫正做无偏估计。使用 Adam 优化器，选取 lr=0.001。观察 loss 下降图，发现由于初期学习率过大，loss 在开始的短时间内下降过快，之后速度放缓，loss 逐渐下降，收敛速度较快，且收敛 loss 值较低，但是收敛过程不平稳，学习率变换不均。下图为训练 loss 下降图和在验证集上的准确率的变化图。

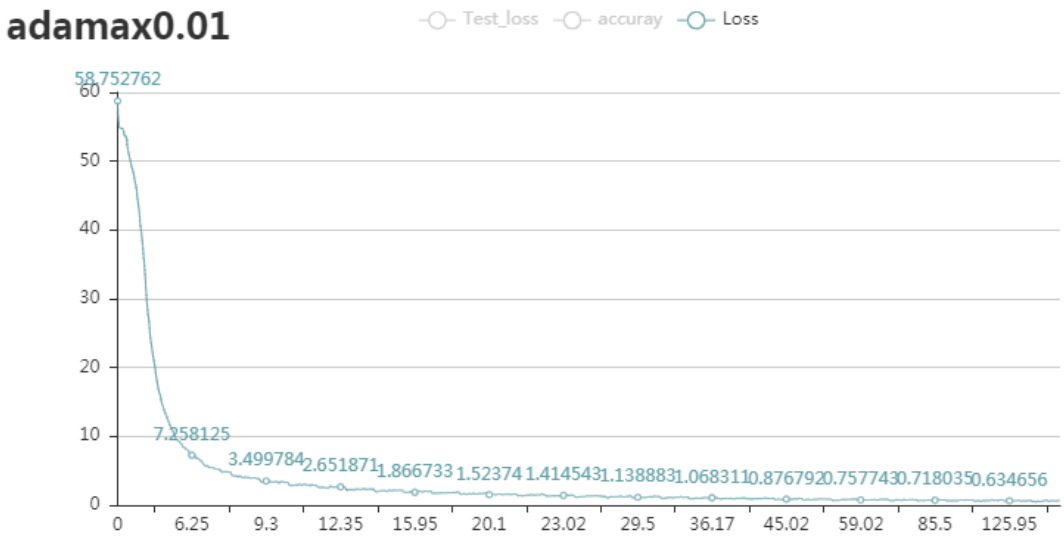


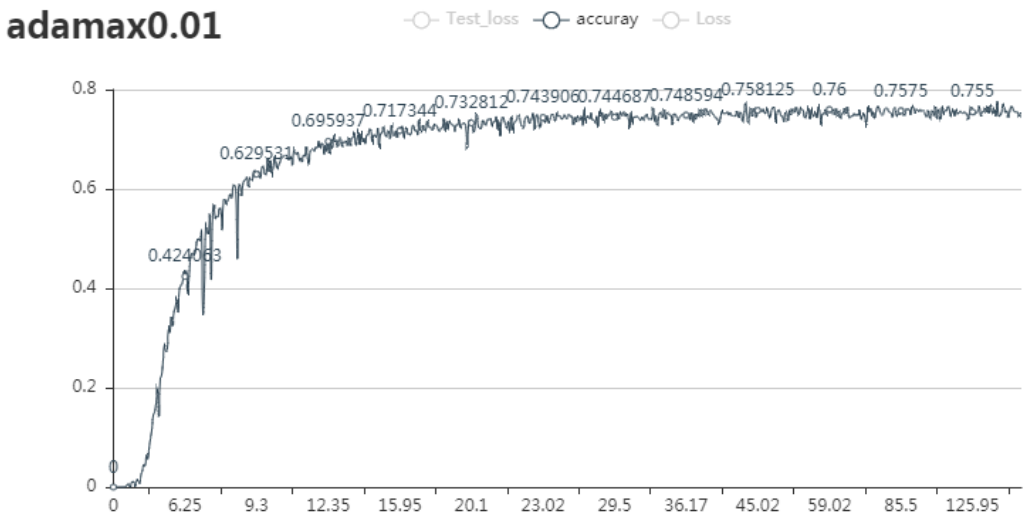


实验参数设置：optimizer=torch.optim.Adam(params,lr=0.001,betas=(0.9,0.999),eps=1e-06)

6. adamax

Adamax 是 Adam 优化算法的变体，对学习率的上限提供了更简单的范围。使用 Adam 优化器，选取 lr=0.01。可以看到使用 Adamax，loss 下降快，短时间就收敛到一个较小的值，下降过程平稳。下图为训练 loss 下降图和在验证集上的准确率的变化图。





实验参数设置：optimizer=torch.optim.Adamax(params,lr=0.01,betas=(0.9,0.999),eps=1e-06)

四、比较分析

将各优化器效果做横向比较,分为训练过程表现比较和在测试集上识别率比较两个方面。

1.训练过程

Optimizer	学习率	收敛时间(h)	验证集准确率	训练loss	paopao识别率	ali识别率
adagrad	0.01	30.92	64.0%	2.330447	**22.9%**	**19.4%**
adadelta	0.01	145.23	47.4%	2.130619	**16.7%**	**15.1%**
rmsprop	0.0001	22.13	73.9%	0.344701	**26.0%**	**25.5%**
adam	0.001	23.23	66.6%	0.997142	**17.6%**	**19.0%**
adamax	0.01	15.95	77.5%	0.490075	**27.5%**	**26.8%**

分析比较各优化器在训练过程中表现。

a. 由于 SGD 和 Momentum 使用整个 batch 的数据计算梯度，对参数更新，所以选择合适的学习率和参数初始值较为困难，而复杂的网络图片（非稀疏数据）数据集更加剧了这个困难。

b. Adagrad 可以根据全部累加梯度自适应调整学习率，但是在训练中后期，

梯度趋于 0，会使得训练提前结束，从表中也可以看出 Adagrad 收敛到的 loss 值约为 2.3 左右，比较看来这个值偏大，也验证了训练可能已经提前结束。

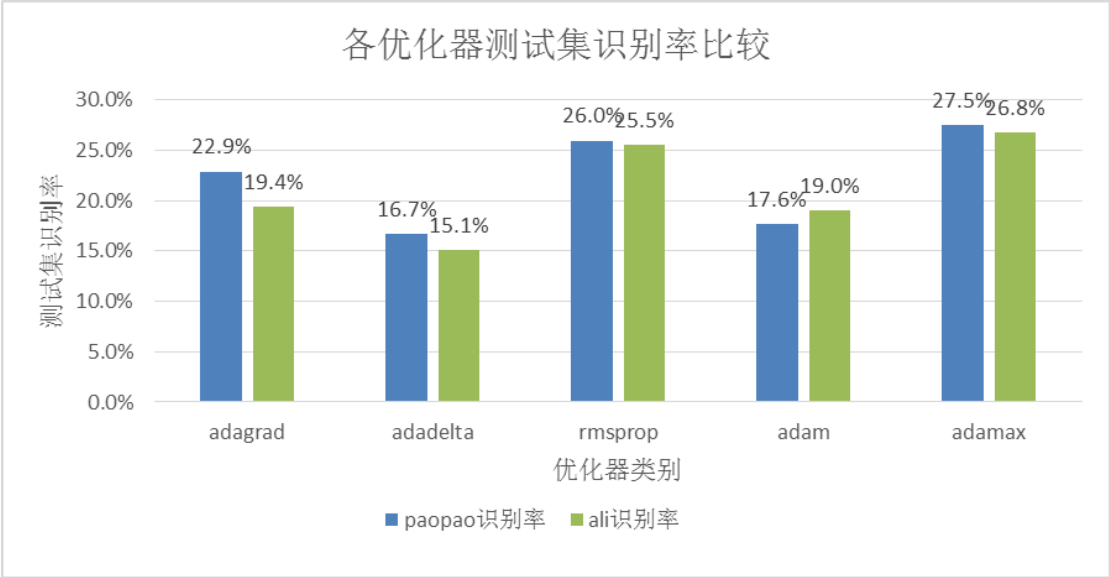
c. 对于 Adagrad 的这个缺点，Adadelta 和 RMSprop 做了不同的改进。其中 Adadelta 将计算全部的累加梯度，改为计算一个窗口期的梯度和（实际使用中利用类似 moving average 的方法实现），有效避免了 Adagrad 中让梯度被惩罚至 0。然而，训练后期，进入局部最小值区之后，AdaDelta 就会反复在局部最小值附近抖动，无法挣脱出局部最小值，这个在实验中也得到了验证：验证集上的准确率维持在较低水平。

d. 对于 RMSprop 优化器，解决 Adagrad 提前停止的问题，采用了和 Adadelta 类似的方法，加入衰减系数，让梯度累积量每一回合衰减一定比例，有效解决了 Adagrad 提早结束的问题，且收敛速度较快。

e. 对于 Adam 和 Adamax 优化器，均是在调整学习率时加入了一阶矩估计和偏基矫正，使得训练时参数变化更为平稳。并且，Adamax 对学习率上限做了限制，相较于 Adam 的学习率变化更为稳定。

比较而言，Adadelta 收敛速度过慢且容易困在局部极值点，Adagrad 由于算法原因会提前结束训练，故这类自适应优化器中，RMSprop 较优。在这类非稀疏的数据集下，Adam 没有 RMSprop 收敛效果好，但是 Adamax 对于 Adam 的提升使其效果与 RMSprop 不相上下。

2.测试集表现



在两种测试集上，各优化器效果差别不大。在各优化器的横向比较中，RMSprop 和 Adamax 表现更优，对应模型在测试集上取得了较高的识别率。

3.验证集跳变原因探究

看数据的话，训练的 loss 是平滑的不存才跳变，所以我觉得跟下降算法无关。那就是生成的合成数据样式太多差距较大，bn 过小的话，抽样样本个体间差异较大，验证集 loss 和 accuray 会出现跳变。

五、结论

本篇报告通过一系列实验，比较了各优化器在网络图片数据集上的表现效果。根据实验数据和结果分析，我们得出如下结论。

首先，SGD 和 Momentum 在网络图片这类复杂数据集上，很难找到合适的初始值和全局学习率，建议使用自适应优化器。而在自适应学习率的优化器中，RMSprop 和 Adamax 优化器训练过程平稳，收敛速度较快，且收敛 loss 值很低，

在两个测试集上识别率较高，更适合本类数据集。

最后，优化器的效果与数据集有很大的关系，数据分布的稀疏性、凹凸性、维度高低等都会影响优化器的效果。本篇报告针对网络图片识别的实验结果只是提供借鉴，实际应用还是需要具体问题具体分析。