Lookahead Optimizer 复现报告

沈冠霖 武笑石 邱中昱 陈俣策

摘要

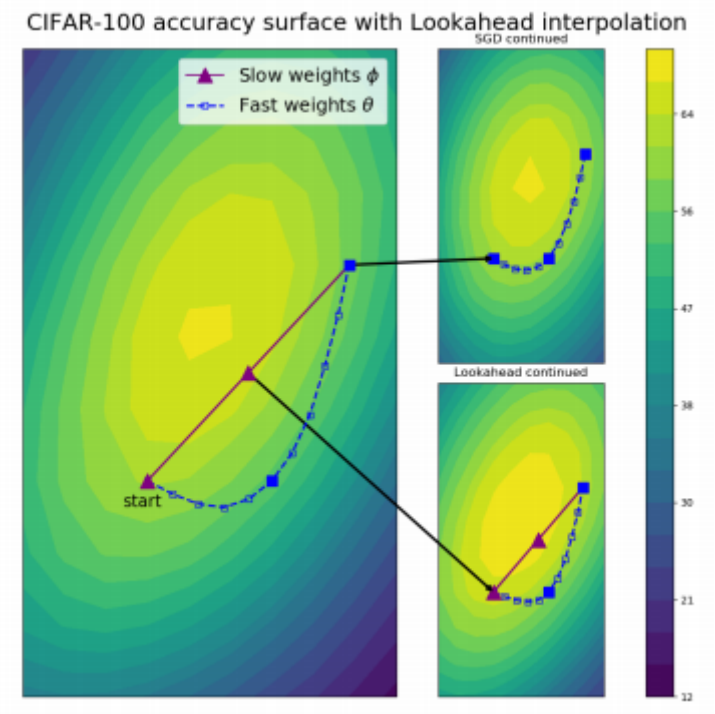
我们小组复现了发表在NeurIPS 2019上的论文Lookahead Optimizer: k steps forward, 1 step back。这篇论文介绍了一种新型的优化器设计方法。我们在论文中用到的数据集上验证了该方法的性能、鲁棒性、进行一些个性化的探究性实验，此外，我们在多种不同任务上对该方法的可拓展性进行了实验。

1. 简介

1.2 算法思想

Lookahead Optimizer的核心思想来源于上世纪凸优化领域的研究。在凸优化领域中，人们发现当把训练不同阶段的权重加权平均后，往往可以取得更低的训练误差。Lookahead Optimizer: k steps forward, 1 step back中将这一研究结果拓展到了深度学习的非凸优化问题中，取得了相对理想的效果。本报告将着重于展示我们小组的复现结果，如果需要更多理论上的阐释，请参考论文原文。

从算法结构而言，Lookahead Optimizer本质上是一个优化器的优化器。Lookahead Optimizer的运行需要一个其他的优化器，例如Adam等，在后文中我们将称之为内层优化器。与之相对应的外层优化器则是Lookahead Optimizer着重实现的部分。算法的实现方法可以大致描述为：内层优化器每运行k个iter，得到参数更新总量为dW，外层优化器更新α * dW。下图是该方法的一种直观表达。



1.3 实验环境

实验使用的操作系统和python库如下：

- 操作系统：Ubuntu 18.04
- python 3.6.9
- pytorch 1.4.0

- numpy 1.18.4
- torchvision 0.5.0
- tensorboard 1.14.0
- keras 2.3.1

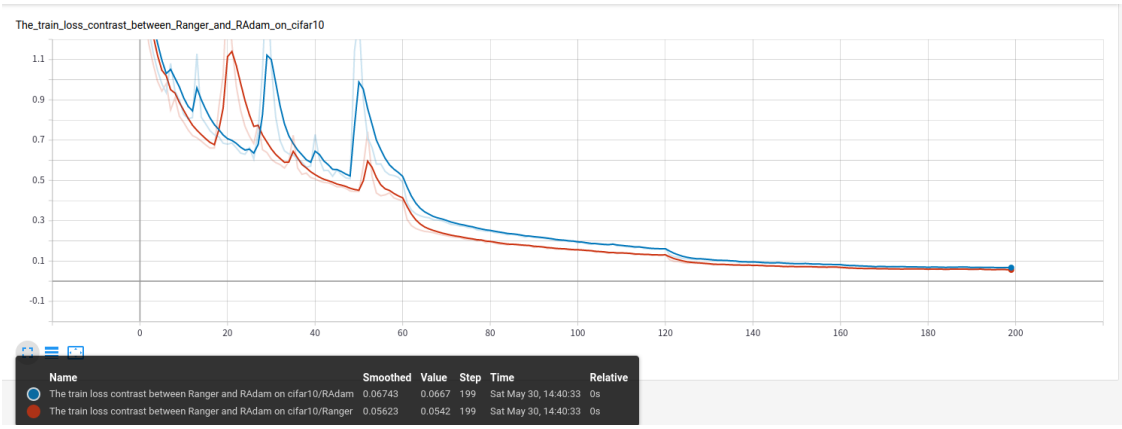
2. 实验结果

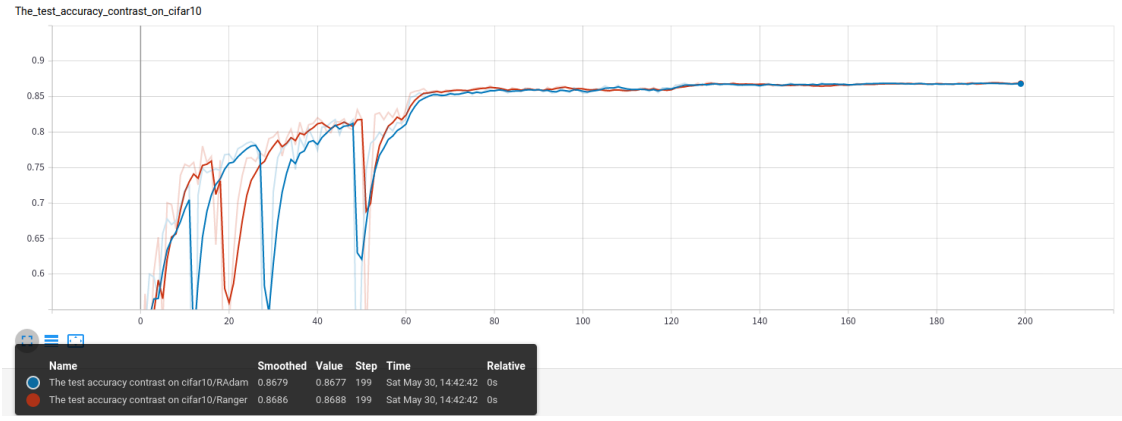
2.1 基本性能比较

在本实验中，我们以RAdam为baseline，验证了Lookahead算法的有效性。测试方法和参数选取如下：

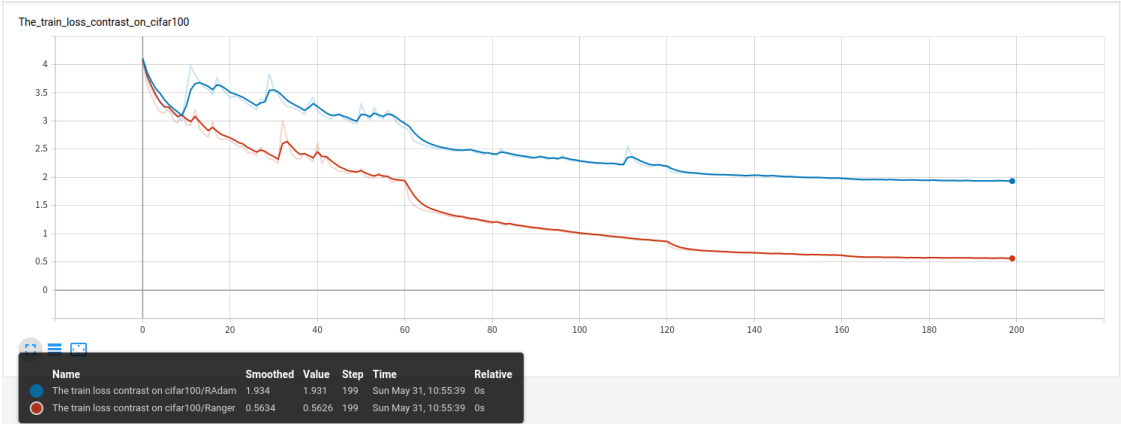
- 数据集： cifar10， cifar100
- 网络模型： Resnet18
- 学习率及衰减策略：初始 learning_rate = 0.1，无预训练，在第60,120,160 epoch时降为原值0.2倍
- weight decay：5e-4
- Lookahead参数：步长k=5，外层参数更新率α=0.8

在cifar10数据集上训练和测试时，train_loss 和 test_accuracy 曲线如下：





在cifar100数据集上训练和验证时，train_loss 和 test_accuracy 曲线如下：



cifar100_bass_acc

由以上图表可以得出以下结果：

- **Lookahead+RAdam的收敛速度比RAdam快，在cifar100上的效果比cifar10更明显**
- **在cifar10上，两者的准确率相当；但在cifar100上，Lookahead+RAdam的准确率明显高于RAdam**

  综上所述，Lookahead方法在cifar-10/100上有效。

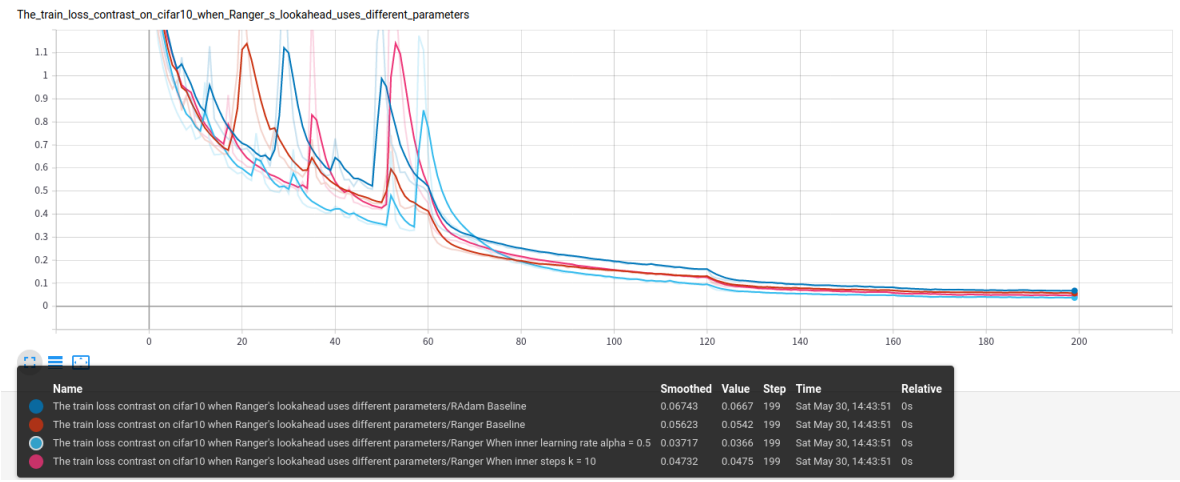
2.2 对超参数鲁棒

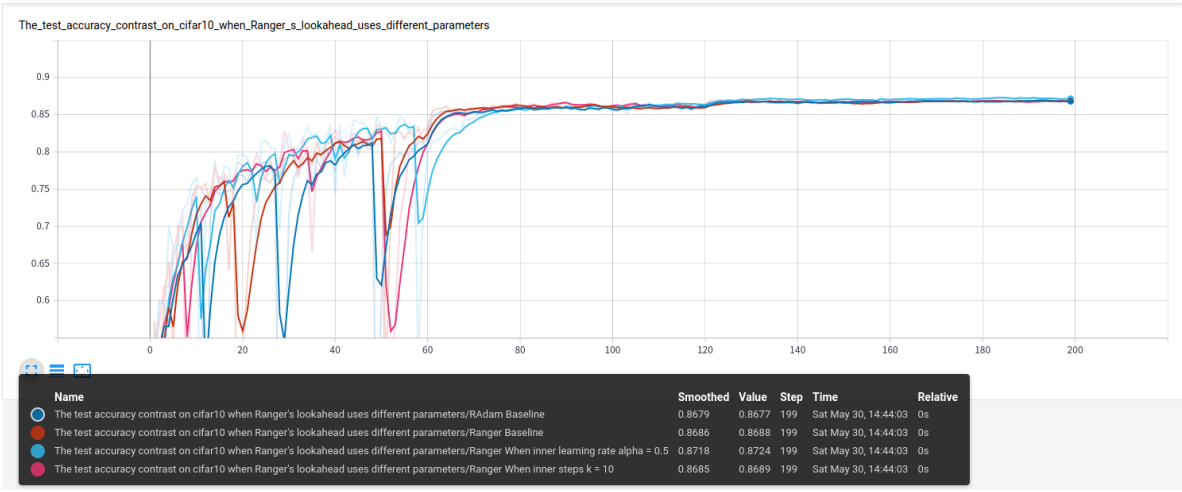
本实验中，我们对Lookahead中的两个超参数 （1）步长k（2）外部参数更新率α 进行鲁棒性测试，探究参数取值的不同是否会影响Ranger优化器的性能，实验中采用了三种取值：

- 与2.1 baseline相同： k=5，α=0.8
- 增大k值： k=10，α=0.8
- 减小α值： k=5，α=0.5

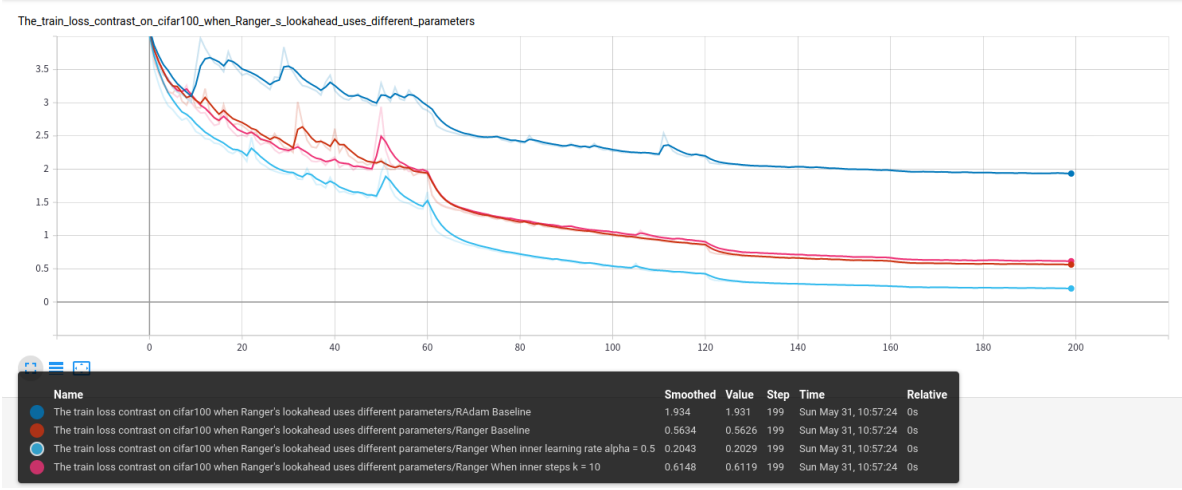
将这三种情况的实验结果与2.1中RAdam的baseline进行比较，作出4条曲线

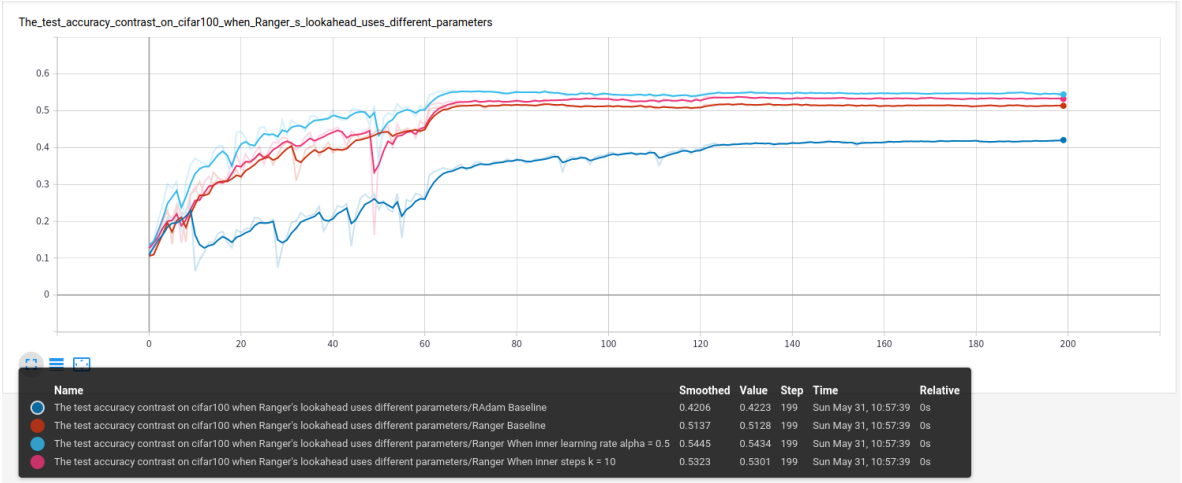
在cifar10数据集上训练和测试时，train_loss 和 test_accuracy 曲线如下：





在cifar100数据集上训练和验证时，train_loss 和 test_accuracy 曲线如下：





由以上图表可以得出以下结果：

- **无论参数的取值如何，Lookahead的优化速率均快于RAdam，在cifar100上的效果更加明显**
- **在cifar10上，优化器的准确率相当；但在cifar100上，无论Lookahead的参数如何，准确率均明显高于RAdam**

实验取得了与2.1相同的结果，说明Lookahead优化器的鲁棒性良好。

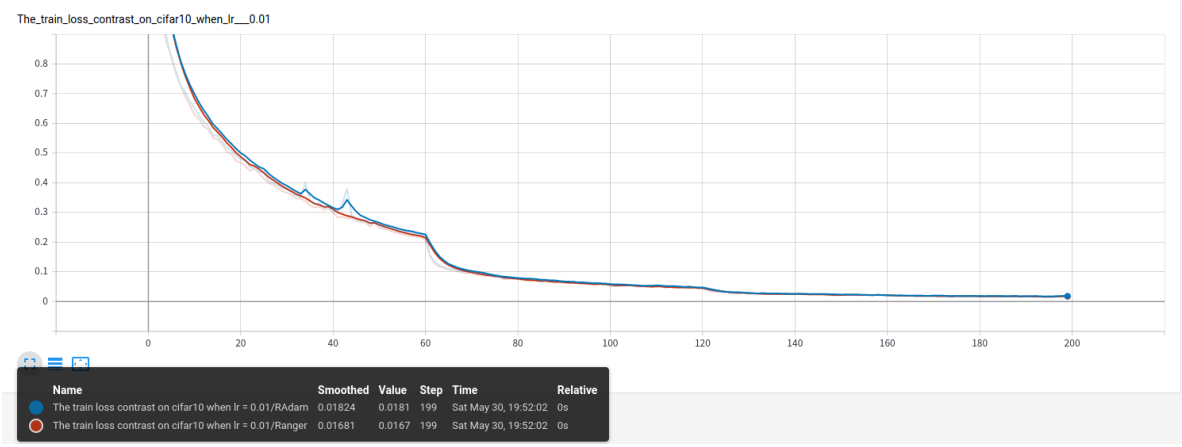
2.3 对学习率鲁棒

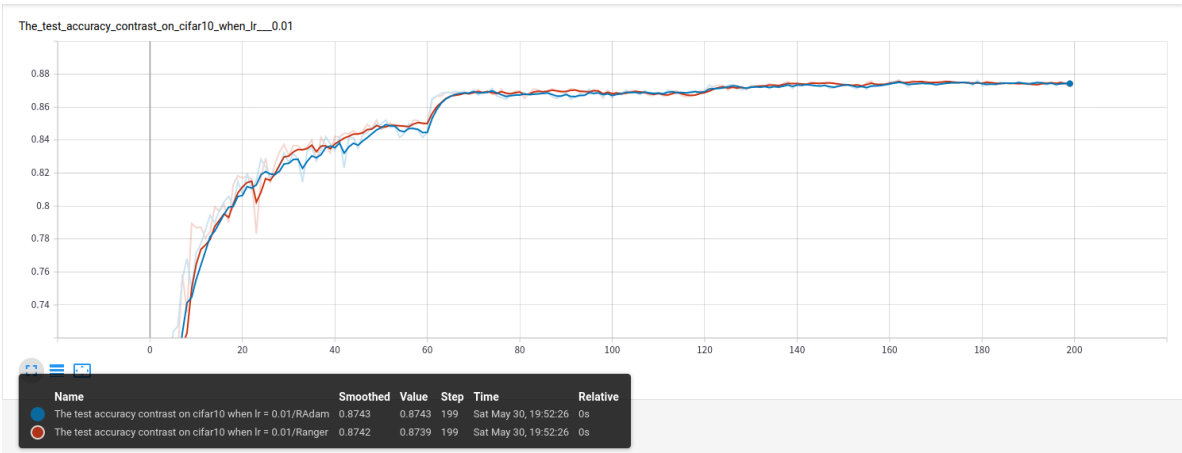
本实验中，我们为了探究学习率的改变对 lookahead 优化 RAdam 的效果的影响，选取了不同的学习率取值和学习率衰减方式进行实验，比较性能变化情况

2.3.1 学习率设置

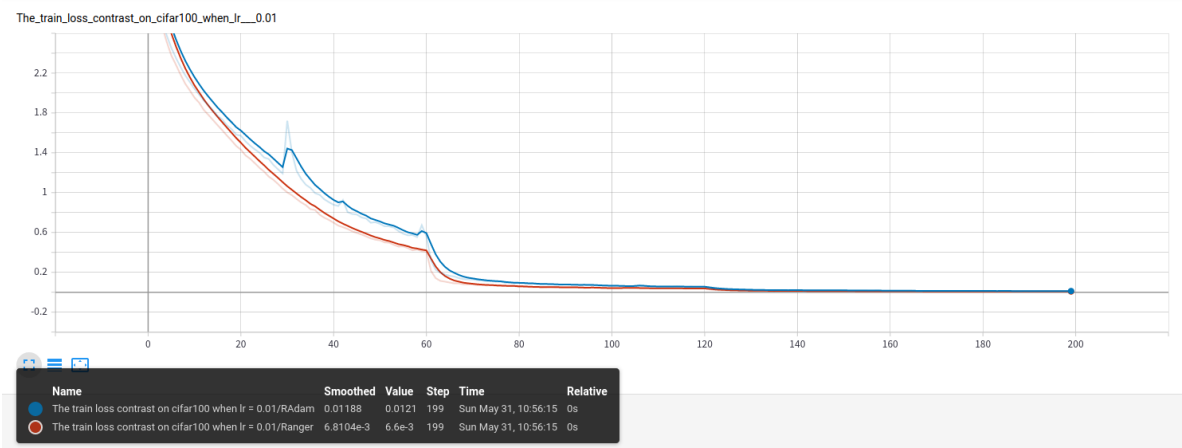
首先，降低学习率 learning_rate = 0.01 (baseline learning_rate = 0.1)

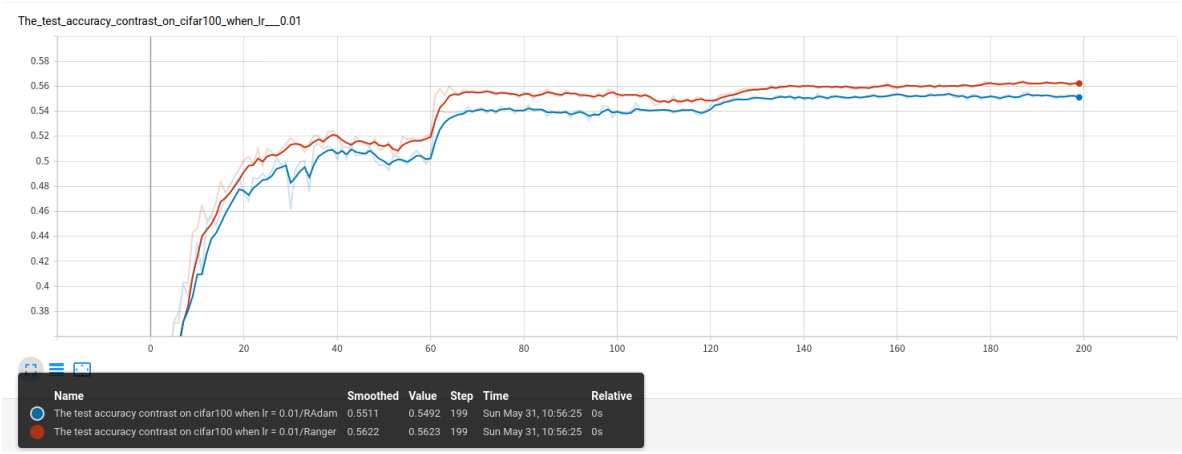
在cifar10数据集上训练和测试时，train_loss 和 test_accuracy 曲线如下：





在cifar100数据集上训练和测试时，train_loss 和 test_accuracy 曲线如下：





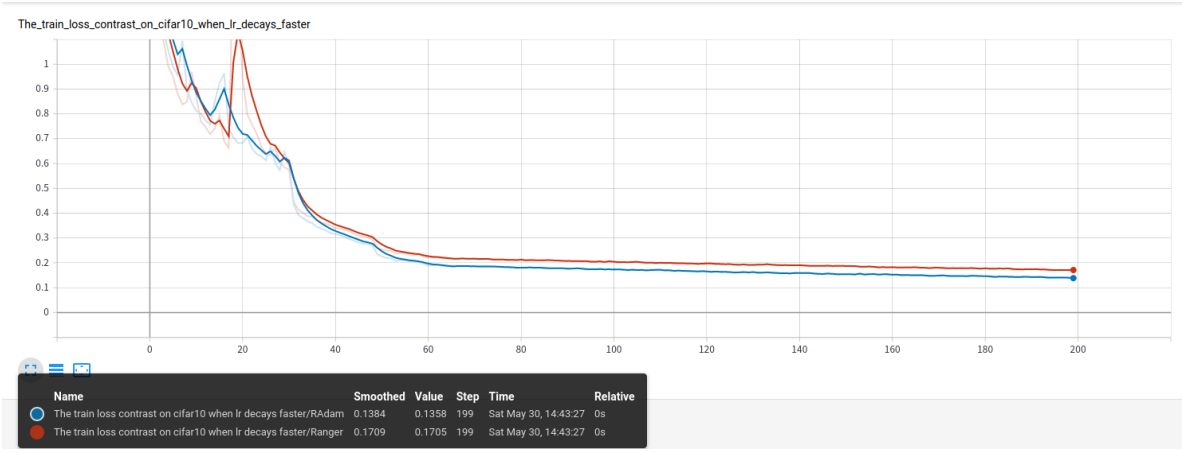
由以上图表可以得出以下结果：

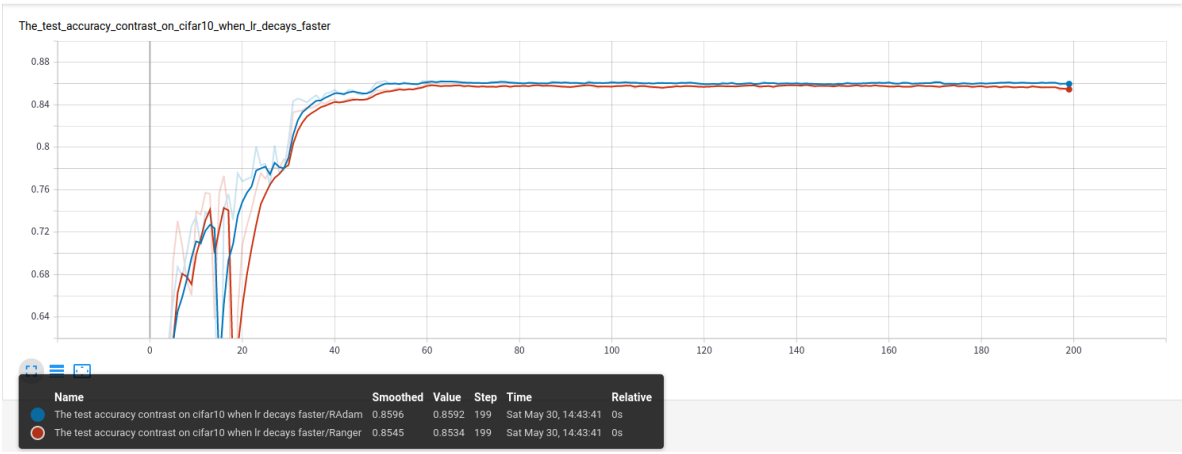
- **在cifar10上，两者的loss和accuracy相当，lookahead并没有明显的优化效果**
- **在cifar100上，Lookahead对RAdam有优化效果，收敛速度和准确率都有所提升**

2.3.2 学习率衰减

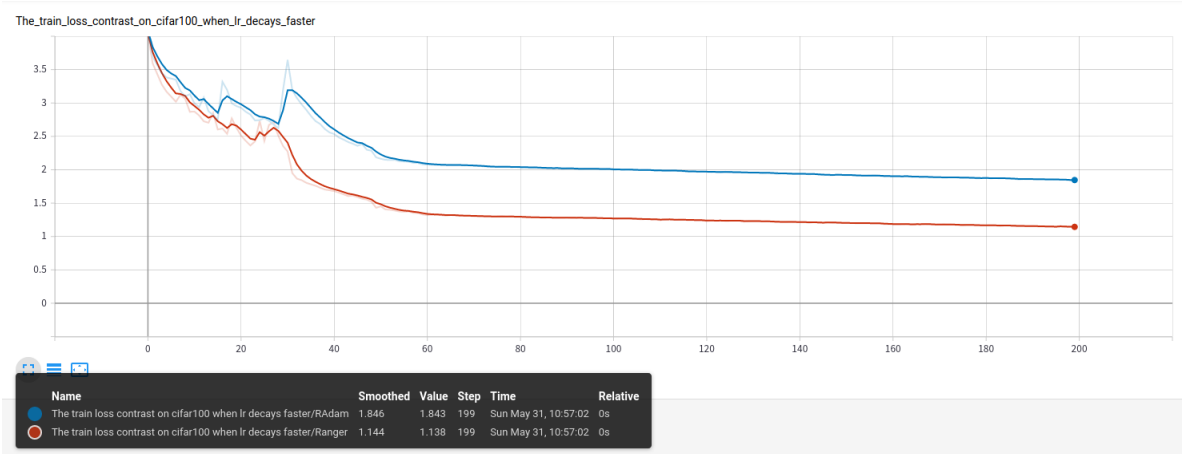
其次，加快学习率衰减速度，baseline中在第60,120,160 epoch时降为原值0.2倍，此处改为在第30,48,58 epoch时降为原值0.2倍

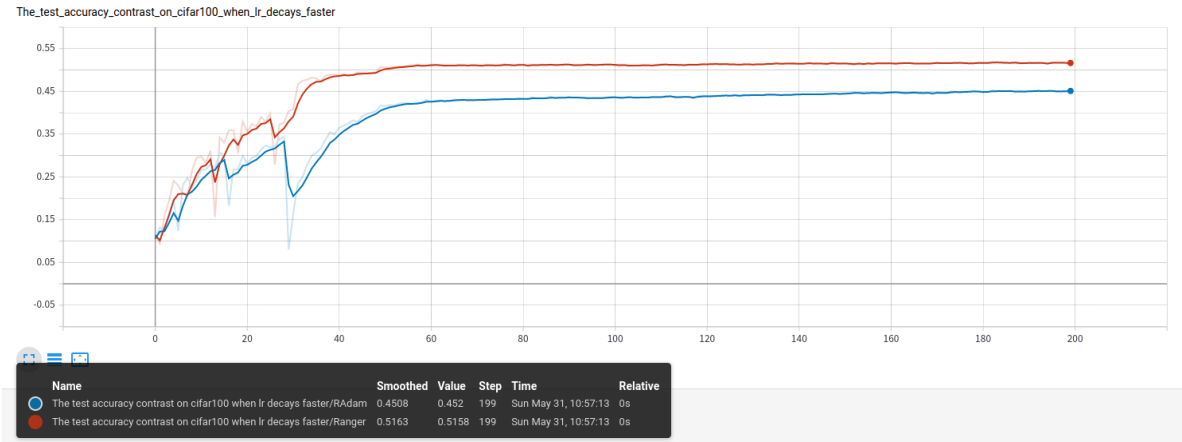
在cifar10数据集上训练和测试时，train_loss 和 test_accuracy 曲线如下：





在cifar100数据集上训练和测试时，train_loss 和 test_accuracy 曲线如下：





由以上图表可以得出以下结果：

- **在cifar10上，两者准确率相当，lookahead甚至会使得loss收敛速度减慢**
- **在cifar100上，lookahead可以显著加快收敛速度和提升准确率，对RAdam的优化性能较好**

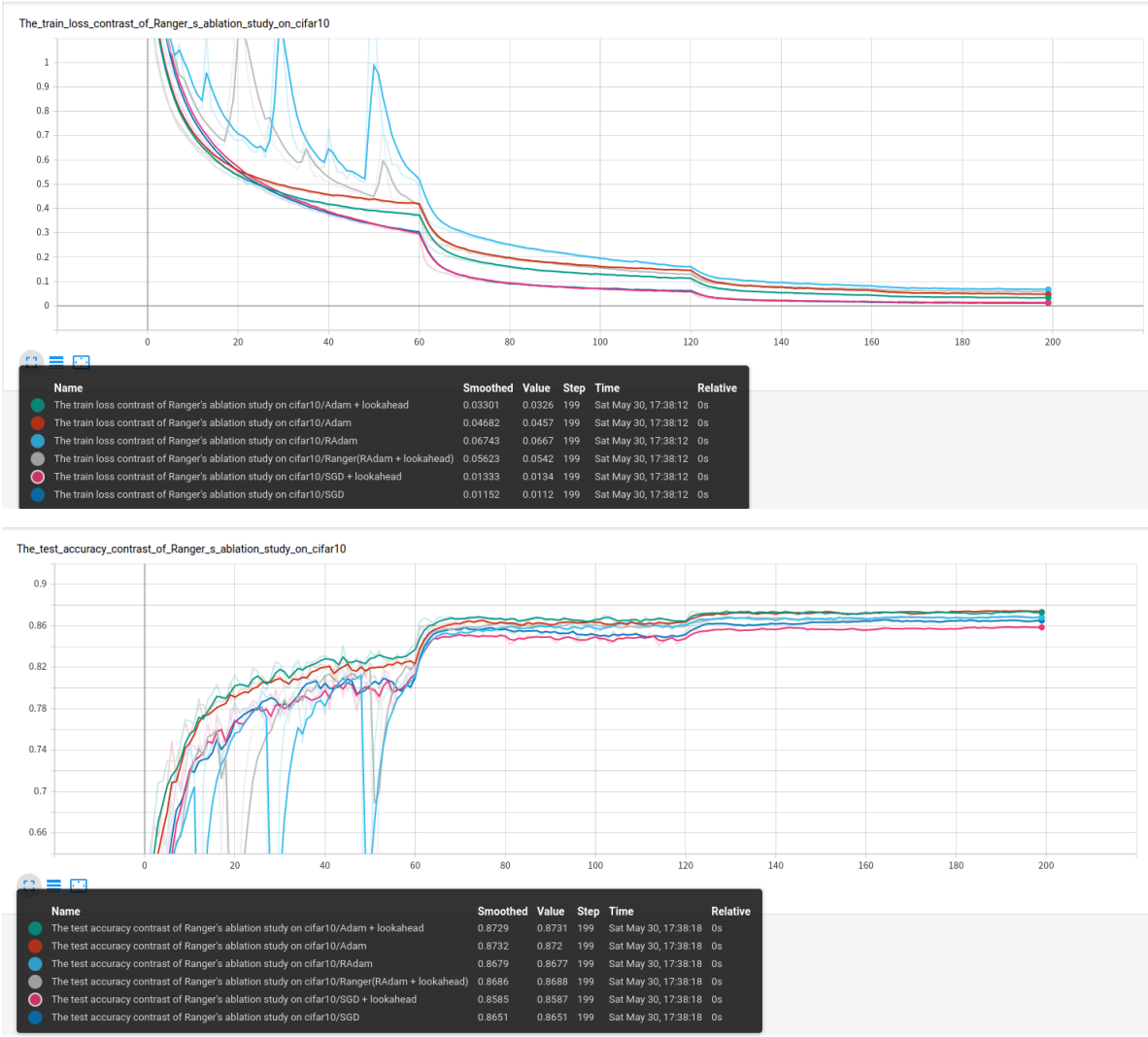
  由以上两个实验可以得出，调整 lookahead 的超参数的效果是因数据集而异的

2.4 Lookahead + other optimizer

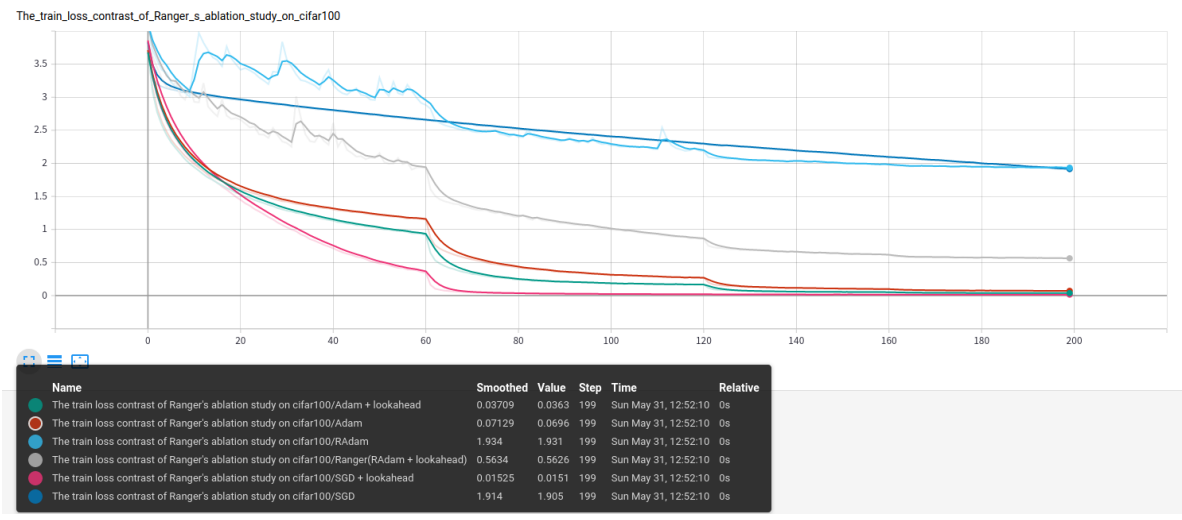
lookahead可以与RAdam组成Ranger优化器，同样可以与SGD，Adam等其它优化器组合，本实验对比了以下三种优化器组合的性能，lookahead的参数与baseline保持一致：

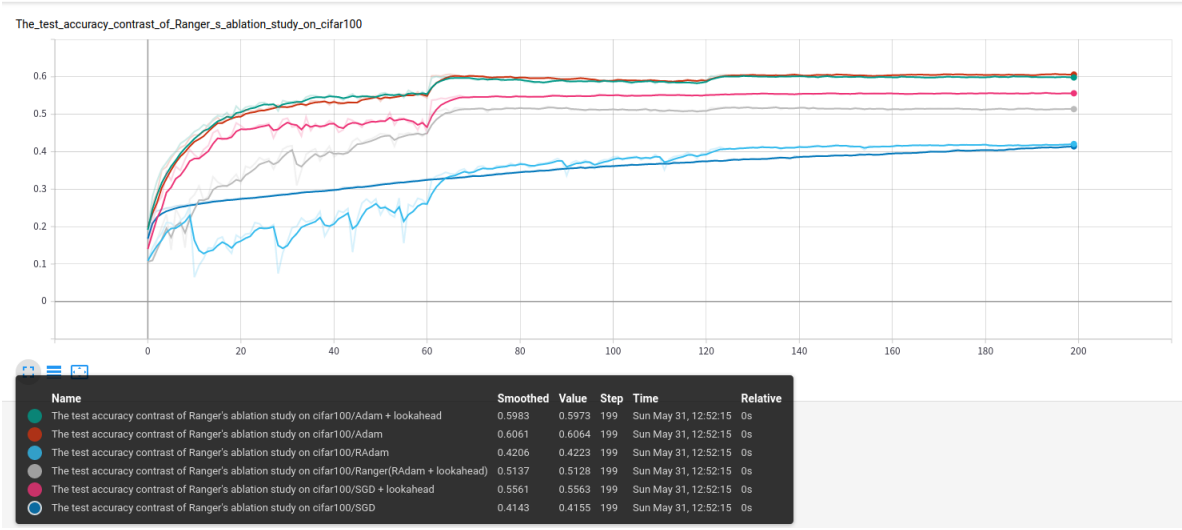
- lookahead + SGD (learning_rate = 0.1)
- lookahead + Adam (learning_rate = 0.001)
- lookahead + RAdam (learning_rate = 0.1)

再与三种优化器未加lookahead时进行比较，作出6条曲线。在cifar10数据集上训练和测试时，train_loss 和 test_accuracy 曲线如下：



在cifar100数据集上训练和测试时，train_loss 和 test_accuracy 曲线如下：





由以上图表可以得出以下结果：

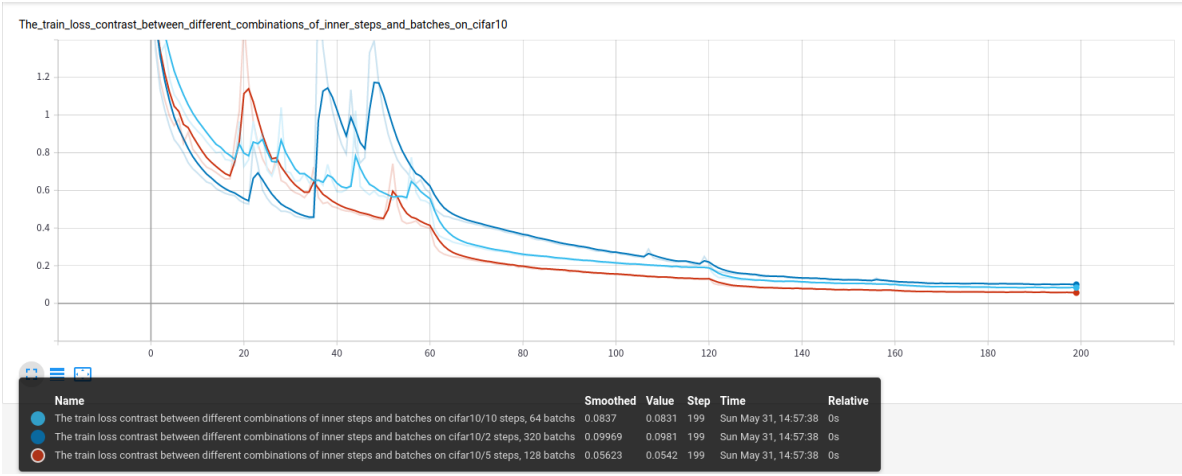
- **在cifar10上，对比 train_loss，各优化器的收敛速度从快到慢为：SGD > Adam > RAdam，lookahead 对 Adam, RAdam 的优化效果较好，但对SGD的优化效果并不明显；各组的 test_accuracy 差别并不大**
- **在cifar100上，对比 train_loss，各优化器的收敛速度从快到慢为：Adam > RAdam, SGD，加上lookahead对三种优化器均有优化效果，其中对 SGD 的优化效果最明显；对比 test_accuracy，lookahead 对 Adam 提升效果不明显，但对 RAdam, SGD 均有提升**

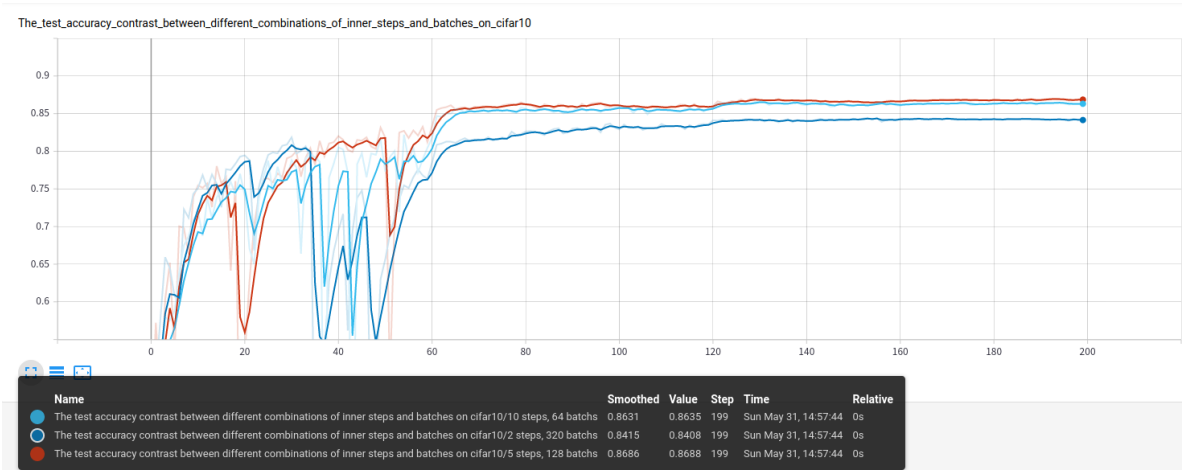
2.5 性能提升是否来源于batch size的提升?

我们注意到，k值越高意味着外层参数更新一次所依赖的batch数越多，因此我们猜测Lookahead的性能提升本质上来源于更高的batch size。如果确实如此，那么该论文的性能提升恐怕难以成立。因此，在本实验中，我们的尝试改变lookahead的参数 batch 和 k 的取值，以对我们的猜测进行验证。我们选取以下三种取值组合进行测试，注意到batch_size * k恒为640：

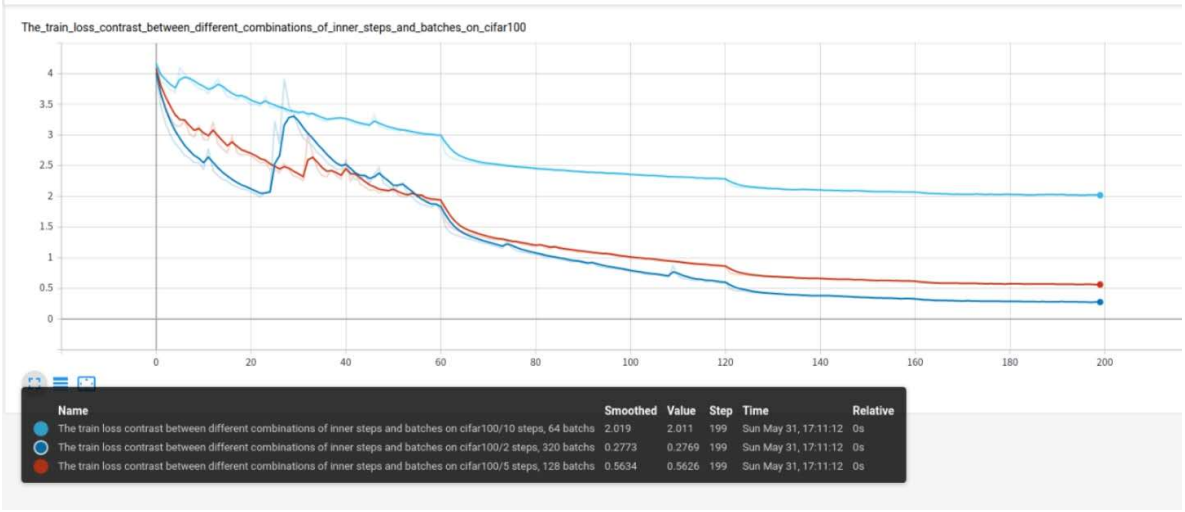
- k = 5，batch = 128
- k = 2，batch = 320
- k = 10，batch = 64

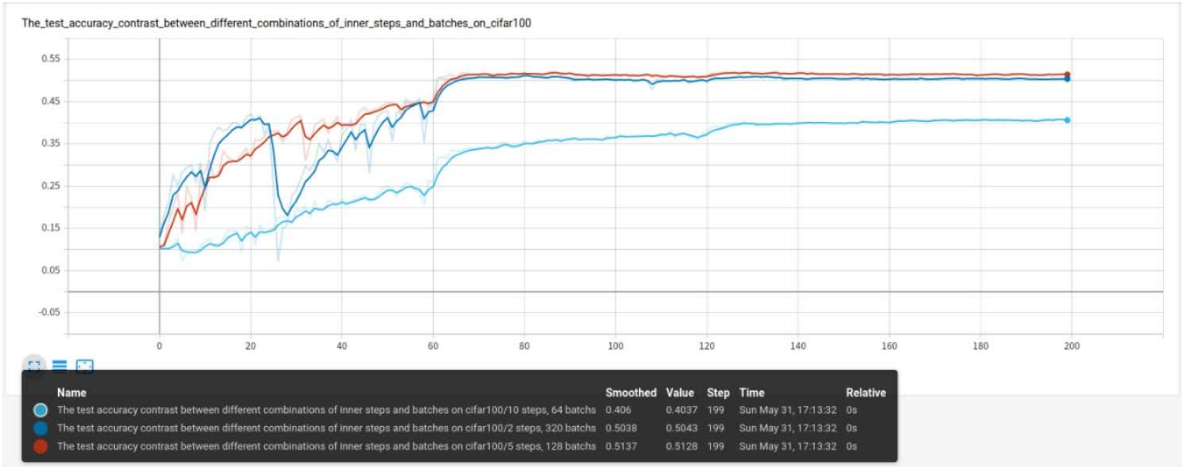
在cifar10数据集上训练和测试时，train_loss 和 test_accuracy 曲线如下：





在cifar100数据集上训练和测试时，train_loss 和 test_accuracy 曲线如下：





由以上图表可以看出，虽然batch_size * k为常量，但各组实验的表现却有较大的差别，说明Lookahead的性能提升并非来源于更大的batch_size，因此论文的性能提高是成立的。

2.6 其他任务/模型上的测试

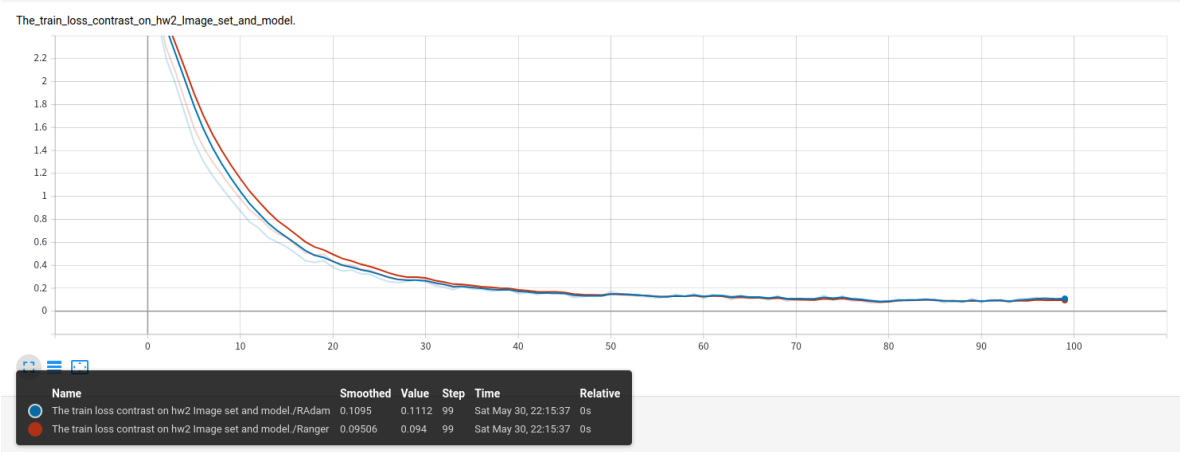
本实验中，在不同的网络模型和数据集下对比 Lookahead的性能提升，选取了两个任务

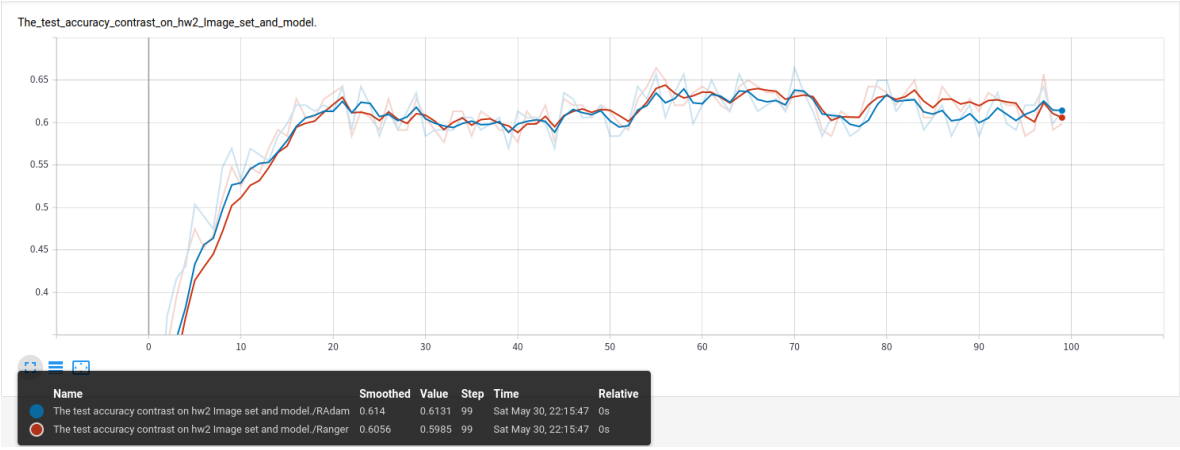
2.6.1 hw2 image classification

在第二次课程作业给出的数据集上进行图像分类任务测试，模型和参数如下：

- 网络模型：三层CNN网络
- 学习率： learning_rate = 0.0001，without decay
- lookahead参数： 内部步长k = 5，内部学习率 α = 0.8

在hw2数据集上训练和测试时，train_loss 和 test_accuracy 曲线如下：





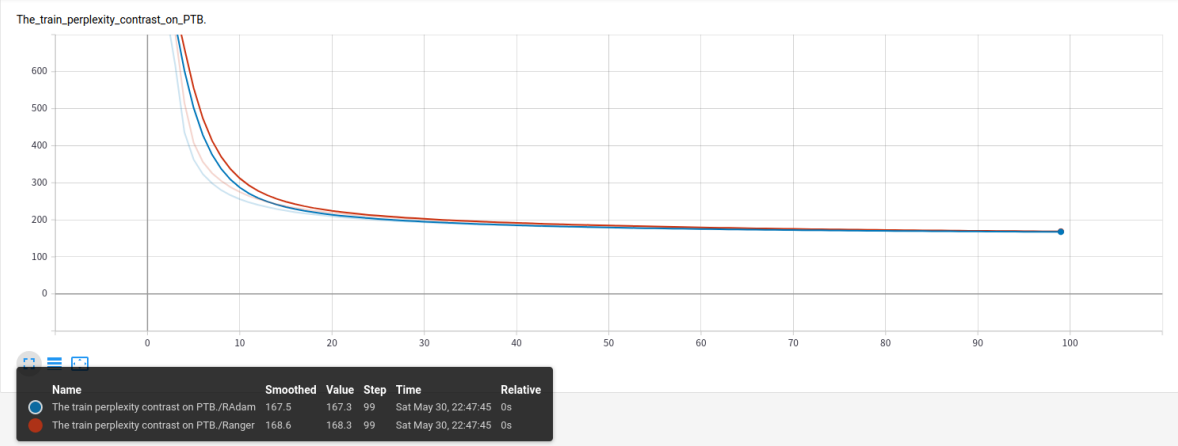
由以上图表可以得出，使用两优化器的收敛速度和准确率相差不多，说明在此任务中 lookahead 并没有有效优化 RAdam

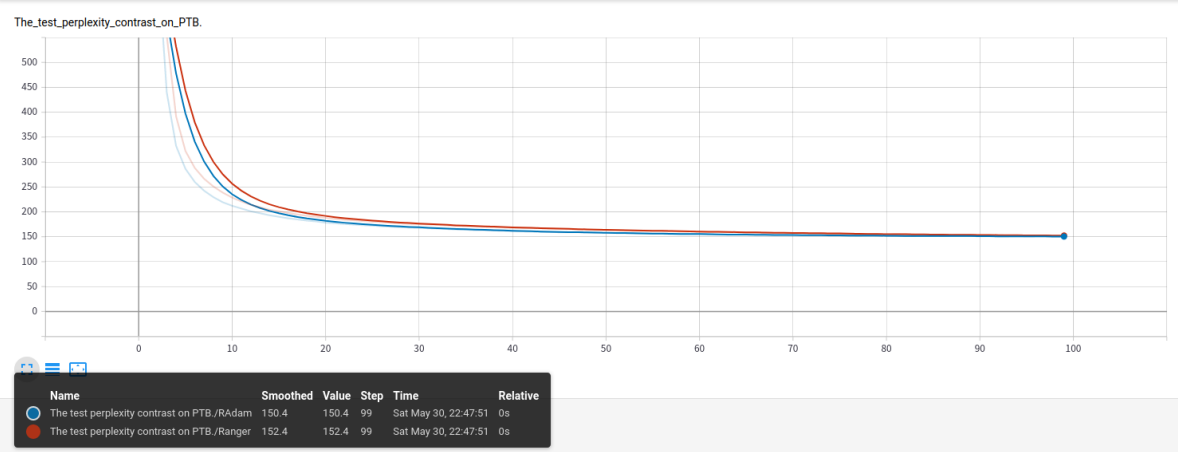
2.6.2 hw3 language model

在第三次课程作业给出的数据集上进行语言模型预测测试，模型和参数如下：

- 网络模型：2层单向GRU网络
- 学习率： learning_rate = 0.001，without decay
- lookahead参数： 内部步长k = 5，内部学习率 α = 0.8

在hw3数据集上训练和测试时，train_perplexity 和 test_perplexity 曲线如下：





由以上图表可以得出，使用两优化器的收敛速度和准确率相差不多，说明在此任务中 lookahead 并没有有效优化 RAdam

3. 总结

我们通过复现验证了论文中提到的部分观点，对Lookahead Optimizer的有效性、鲁棒性进行了验证。此外，我们对性能提升的来源进行了探究，间接证实了论文中关于性能提升原因的相关论述。然而，通过在其他任务上的实验，我们也发现了该方法并不能够在所有任务上都取得性能提升，甚至在许多我们测试过的任务上都没有取得性能提升，说明这个方法或许存在较大的局限性。关于性能没有提升的原因，虽然没有理论支撑，但通过经验我们分析可能与数据集规模有关。根据我们以上展示的实验结果，可以发现当数据集规模较大、复杂度较高（cifar-100）时，Lookahead能够取得的效果非常明显，而复杂度降低到作业数据集时，性能提升就接近于0了。

如果从使用者的角度对Lookahead方法进行感性的评价，Lookahead最大的特点是对超参数非常鲁棒，因此如果是为了验证一个idea，用Lookahead的好处在于实验初期可以减少超参数上的时间投入，利用相对粗糙的超参数获得相对准确的实验结果。然而Lookahead的性能提升在不同数据集上并不稳定，因此实际使用中还是应当具体问题具体分析，比较后选择最好的优化器，以跑出最好的结果。

4. Reference

[1] lookahead source code: https://github.com/michaelrzhang/lookahead

[2] cifar10/100 dataset: https://github.com/uoguelph-mlrg/Cutout/blob/master/train.py

[3] IMDB classification dataset: https://github.com/Cong-Huang/Pytorch-imdb-classification