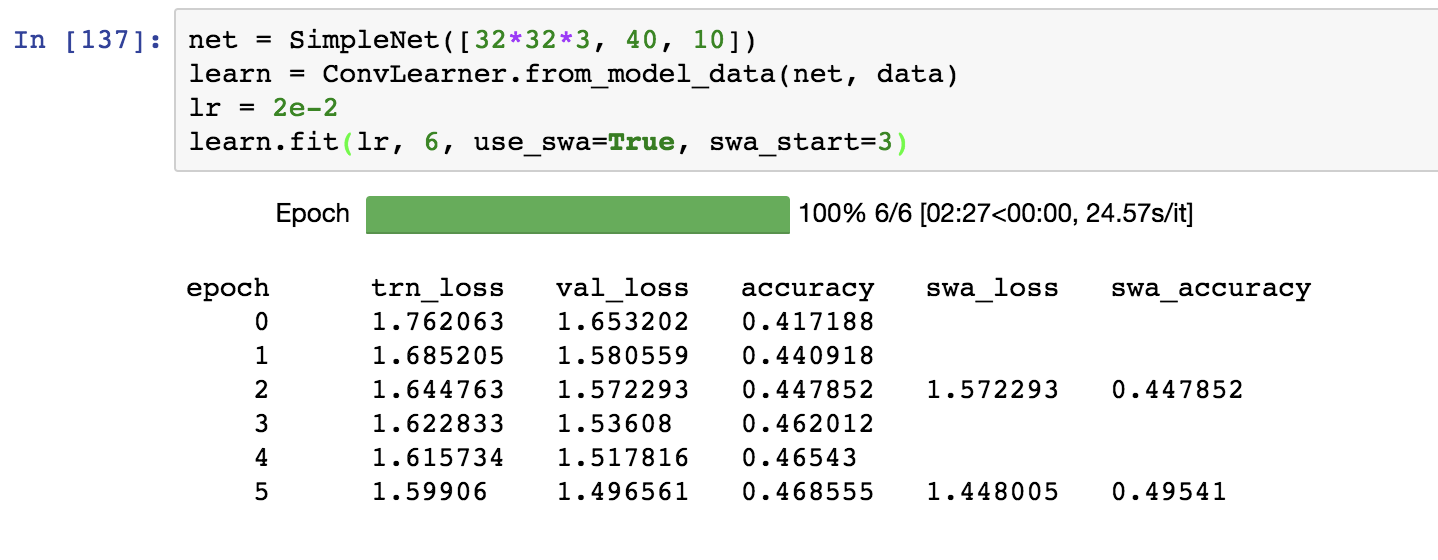
# 在fast.ai库中加入尖端的深度学习训练技术

我如何在Twitter上发现一篇新的深度学习论文，复制其结果，并提交一个开源实现



### 简介

我在二月底被fast.ai国际奖学金录取了。这是一个令人兴奋的机会，因为自从去年发现这些视频以来，我一直是MOOC的粉丝，并利用我从v1课程（与Keras和Tensorflow一起教授）中获得的知识参加了一些Kaggle比赛。

当我花时间准备课程，看去年秋天的视频，尽可能快地学习PyTorch的时候，我在我的feed中看到了一篇新的论文：“平均权重可以得到更广泛的最优值和更好的泛化。”特别是，我看到了一条关于它可能是什么的微博。我想现在我是这个程序的一部分，我可以把它添加到库中！

在我作为软件工程师的职业生涯中，我总是发现学习一项新技术的最好方法是有一个你想应用它的具体项目，所以我认为这是一个很好的机会来磨练我萌芽的PyTorch技能，更好地熟悉fast.a I库，并提高我阅读和理解深入学习论文的能力。

我能够完成一个实现并执行本文中的一些实验，观察到与作者在CIFAR-10数据集上使用SWA训练VGG16和Preactivation-Resnet-110模型时所报告的改进类似的改进。对于VGG，SWA将误差从6.58%降低到6.28%，相对提高了4.5%，而Resnet模型则有更大的优势，将误差从4.47%降低到3.85%，相对提高了13.9%。

### 报纸

背景

SWA方法来自于对集成的研究。Ensembling是一种提高机器学习模型性能的流行技术 - 例如，Neflix奖是由一个集成获得的，Ensembling也是为了提高单个模型的最终性能。

在最简单的形式中，集成可以包括训练具有不同初始化的模型的一定数量的副本，并平均副本的预测以获得集成的预测。这种方法的缺点是，您必须承担培训n个不同副本的成本。为了避免训练所有这些拷贝，研究人员提出了一种称为的方法。使用快照集合，可以训练单个模型，但训练的目的是使模型收敛到多个局部最优解，并保存（或快照）每个局部最优解处的权重。这样，一次训练可以产生n个不同的模型，这些模型的预测值可以取平均值以创建一个集合。

在发表SWA论文之前，一些作者开发了一种称为（FGE）的方法，该方法改进了快照集成论文的结果。FGE论文的创新之处在于找到了“两个局部最优点之间的路径，使得这些路径上的列车损耗和测试误差保持在较低的水平”。也就是说，利用FGE，作者能够发现具有理想特性的损耗面曲线，并沿着这些曲线建立集成模型。

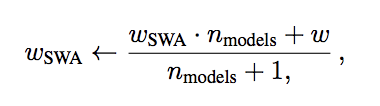
在SWA文件中，作者提供了SWA近似FGE的证据。然而，与FGE相比，SWA的好处是推理成本更低，而对于FGE，您仍然需要生成n个模型的预测，对于SWA，您最终得到的是一个模型，因此推理可以更快。

算法

那么SWA实际上是如何工作的呢？结果证明，算法相对简单。

首先，复制一个你正在训练的模型，它将用于记录平均重量。

每次历次训练后，根据以下等式更新副本的权重：



其中n\_models是已经包含在平均值中的模型数，w\_swarepresents是副本的权重，w表示正在训练的模型的权重。这相当于存储在每个训练阶段结束时看到的模型的运行平均值。

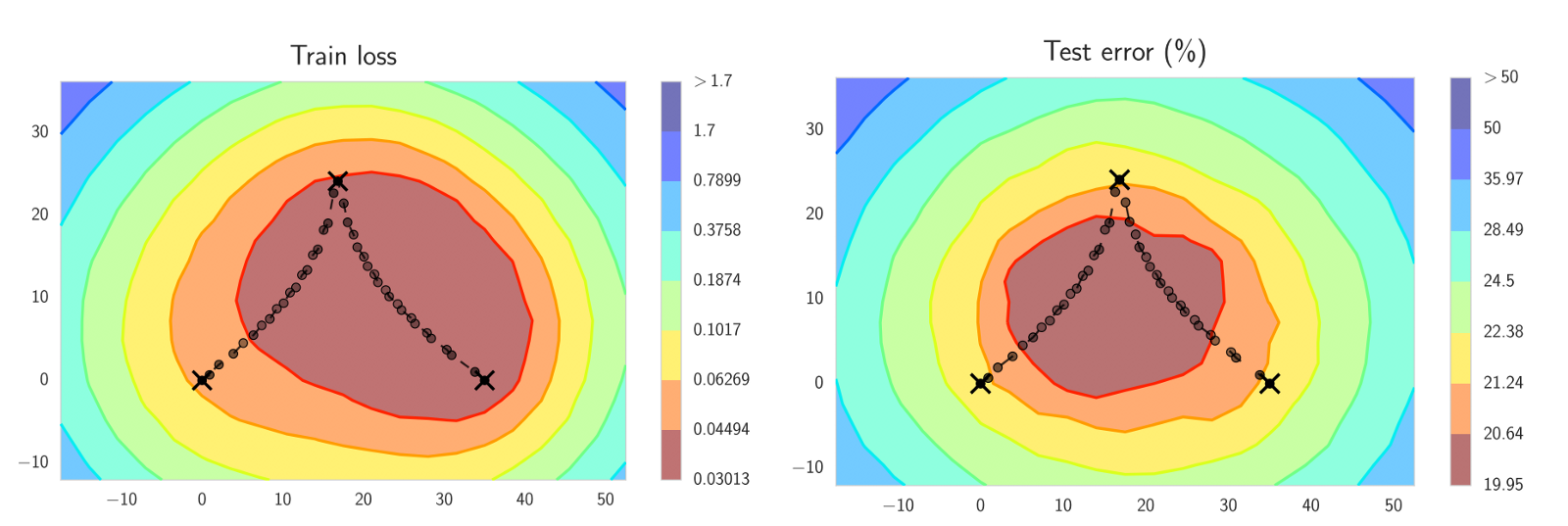
这就是算法的核心！但本文还介绍了一些额外的问题。首先，作者开发了特定的学习速率计划，以确保SGD在开始对模型进行平均化时能够探索出良好的最优值。

此外，您通常希望为一定数量的时间段预先训练网络以启动，而不是从一开始就开始跟踪平均值。此外，如果使用周期性学习率，则希望在每个周期结束时存储平均值，而不是在每个周期结束后存储平均值。

寻找更宽的最优值

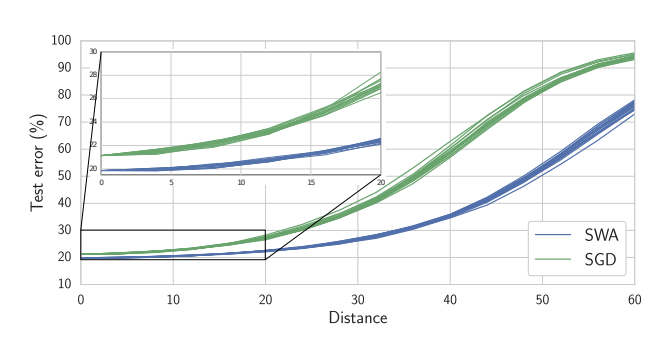
作为SWA工作原理的一个解释，作者提供了证据，证明它导致模型最终达到比SGD更广泛的局部最优。寻找一个更广泛的最优值可以提高模型的泛化能力，因为列车的损失面和试验数据可能并不完全一致。因此，对于训练数据处于更广泛的最优状态使得模型对于测试数据也更可能处于最优状态。

这可以在报纸上的一张图表中看到，这里是：



您可以看到训练损失（左）和测试错误（右）是如何相似的，但并不完全相同。例如，最右边的X在训练损失面上的一个最佳点上，但是距离最佳测试误差有一段距离。正是这些差异使得寻找一个更广泛的最优值变得更好，这个最优值更有可能出现在列车和测试损失的最佳位置。

那么SWA真的会导致更广泛的optima的发现吗？作者提供了实验证据来支持这一观点。在他们的论文的第3.4节，称为Optima Width，他们提出了以下建议：为了比较SGD和SWA所发现的Optima的宽度，可以将损失作为给定方向上距Optima的距离的函数来测量。作者从10个不同的方向取样，测量了在CIFAR-10上用SGD和SWA训练的预激活Resnet的损耗，结果如下：



该图提供了SWA发现的最优值比SGD发现的更宽的证据，因为从SWA最优值到增加相同数量的测试误差需要更大的距离。例如，要达到50%的测试误差，您必须从SGD最佳值行驶约30英里，而SGD的测试误差约为50英里。

实验

作者在不同的数据集和模型结构上进行了大量的实验来验证SWA方法。首先，我将详细描述在算法实现过程中尝试复制的实验的设置，然后我将介绍其他一些关键结果。

我选择的复制实验是在CIFAR-10上进行的，使用VGG16和Preactivation-Resnet-110架构（作者在论文中还包括Wide ResNet28-10和Shake-Shake2x64d）。对于每种架构，都选择了一定的预算，以表示仅使用SGD+动量训练模型收敛所需的阶段数。VGG的预算是200，Resnet是150。这些模型是用SGD+动量为单一预算训练的。然后，为了测试SWA，用SGD+动量训练模型，训练大约75%的预算，然后用SWA训练额外的时间段，达到原来预算的1、1.25和1.5倍。每项测试训练三个模型，并报告平均值和标准差。

除了CIFAR-10上的实验外，作者还对CIFAR-100进行了类似的实验。他们还在ImageNet上测试了预先训练的模型，并使用SWA运行了另外10个阶段，发现预先训练的ResNet-50、ResNet152和DenseNet-161的精度有所提高。最后，作者展示了在固定的学习率下，使用SWA从零开始成功地训练一个宽ResNet-28-10。

关于进一步的细节，你可以找到一篇论文，“平均权重会导致更广泛的优化和更好的泛化”。

### 实施

读了几遍论文并消化之后，我跳进fast.a I库，想知道应该在哪里添加代码来让SWA工作。

因为fast.ai库提供了添加自定义回调的能力，所以我认为这是实现算法的正确位置。如果我用钩子写了一个回调，这个钩子是在每个纪元结束时调用的，我将能够在适当的时间更新权重的运行平均值。这就是我最后得到的代码：

回调包含三个参数：model、swa\_model和swa\_start。前两个只是我们正在训练的模型，以及SWA开始后用来存储平均重量的模型副本。swa\_-start参数是平均开始时的历元，因为在本文中，在开始跟踪平均权重之前，模型总是针对一定数量的具有SGD+动量的历元进行训练。

从这里，您可以看到SWA回调如何将论文中的算法转换为Pytorch代码。如果我们到达SWA开始的时间，我们更新参数的运行平均值，然后增加包含在平均值中的模型数（称为SWA\_n）。

除了执行大部分算法的回调代码之外，我还需要在SWA模型可以进行推理之前包含修复batchnorm的运行平均值的代码。正如作者在文章中所解释的，“如果DNN使用批处理规范化[Ioffe和Szegedy，2015]，我们需要计算训练完成后SWA发现的网络的每一层激活的运行平均值和标准差。”批处理规范层通常在训练期间计算这些运行统计数据，但是由于该模型的权重是作为其他模型的平均值计算的，这些运行状态对于SWA模型的激活将是错误的，因此需要另一个通过数据的单通道来让batchnorm层计算正确的运行统计信息。

修复的代码如下所示：

最后，要确保我为SWA编写的代码放在训练循环的正确位置。这要求我进一步了解fast.ai库是如何在各个时代运行以适应模型的。我不得不对learner.py的fit方法进行一些修改，并在model.py中添加了一些代码，以便在训练的模型之外使用SWA模型运行验证。

对于在fast.ai中工作所必需的更改的完整视图，您可以从我的pull请求中看到差异。

### 测试

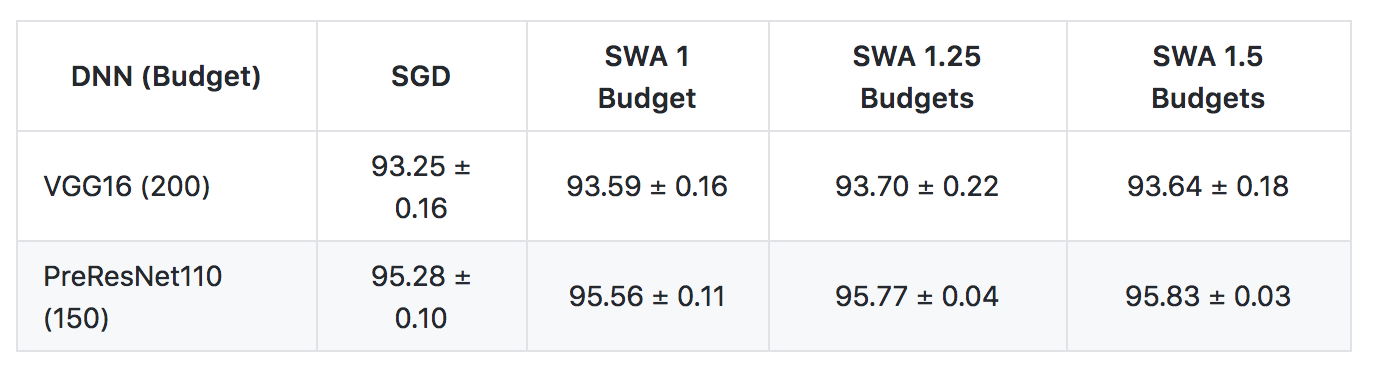
来自软件工程的背景，测试是我非常重视的事情。但是，很难将单元测试之类的东西应用到机器学习代码中，这可能是因为在您的培训中有一些不确定的元素，或者是因为到达您真正想要测试的状态所需的时间。（查看这篇博客文章——关于在机器学习中使用单元测试的详细讨论。）

即使我不能将真正的单元测试应用到我正在编写的代码中，我仍然想要某种方法来确保我所做的工作是实际工作的（并且在这个过程中我没有破坏库的其他部分）。为了做到这一点，我做了两个“测试”笔记本，一个是“功能”测试，是更小的代码块，通常运行在更简单的模型上，这是为了回答一个问题：“这个函数做了我认为它做的吗？”？”. 例如，一项功能测试检查了在经过几次培训后，SWA模型实际上等于所有SGD模型参数的平均值：

这些测试通常可以在30秒内运行，在我编写实现时是一个很大的帮助，在出现故障时提醒我。由于fast.ai库的开发速度目前非常快，这些测试还帮助我在尝试解决与主分支的合并冲突时快速识别问题。

我做的第二个测试笔记本是为了我所谓的“实验”测试。这是为了回答这个问题，“如果我试图使用我的实现和fast.ai库重新创建论文的实验，我观察到的结果是否与论文相同？“一旦有了功能实现，我就运行这些测试，以确定SWA是否会对库做出有用的贡献。这些实验比功能测试花费的时间要长得多（对于我的1080ti上的每个PreResNet110模型，对于12个模型，大约需要3-4个小时），但这是一个很好的最终检查，证明一切都按预期工作。

结果表明，随机加权平均在CIFAR-10上的精度确实比一般的带动量的SGD高，而且随着训练次数的增加，改进的幅度也普遍增大。正如你在下表中看到的，我所有的结果都比原始论文的结果有更好的准确性，这也是我还在调查的。一个因素可能是对CIFAR-10进行数据增强的方式增强包括将每个图像填充4个像素和随机裁剪，我发现fast.a I默认情况下使用不同类型的填充（反射填充）。然而，SWA相对于SGD+动量的改善模式是显而易见的。



（我为测试编写的代码可以在中找到。）

### 结论

我对这个项目的最终结果非常满意，因为我能够从一篇前沿的研究论文中复制一个实验，并为一个机器学习开源项目做出了我的第一个贡献。首先，我非常感谢来自法斯特的杰里米和瑞秋。我从你身上学到了很多，我很感谢你发现了这门课程，这次有机会成为一名国际同行。同时，感谢研究人员帕维尔·伊兹梅洛夫、德米特里·波多皮钦、蒂穆尔·加里波夫、德米特里·维特罗夫、安德鲁·戈登·威尔逊，他们撰写了这篇伟大的论文。

And now I’d like to encourage everyone to download the fastai library and give SWA a try! I’m particularly interested to see people start applying it to their own image datasets (beyond CIFAR and Imagenet), as well as exploring the possibility of using it outside of the domain of computer vision.