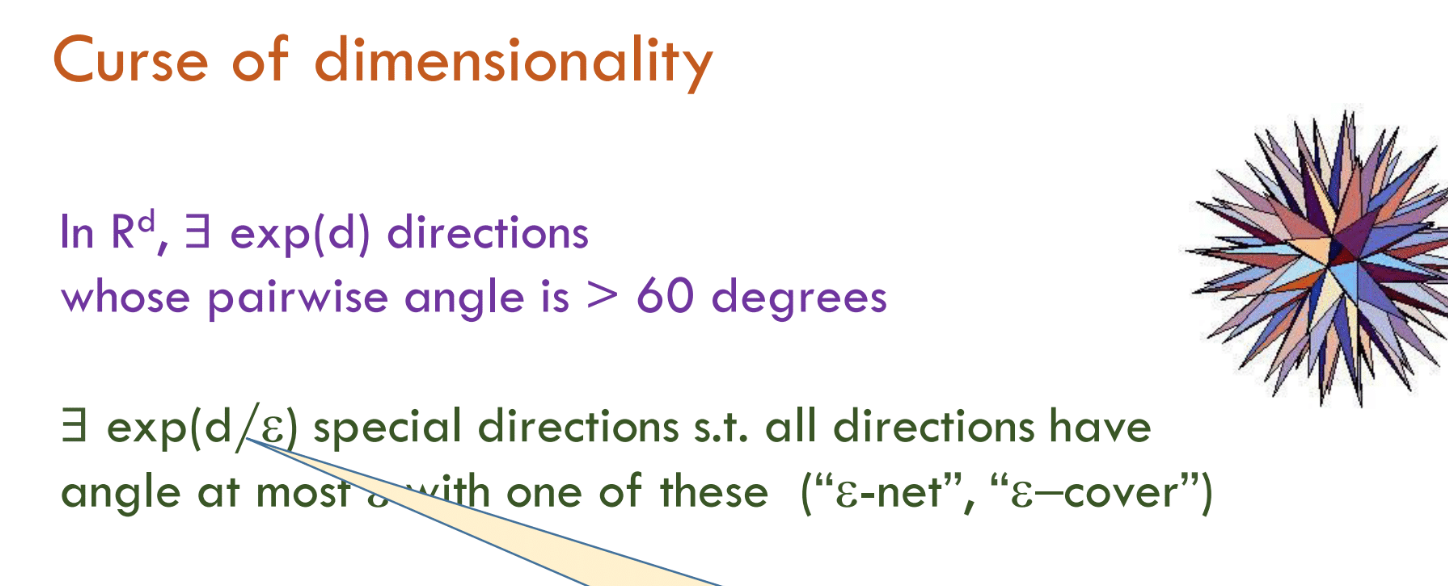
# ICML 2018注释

我在ICML 2018上记下的一些笔记

#### 桑耶夫·阿罗拉：走向深度学习的理论理解

<http://unsupervised.cs.princeton.edu/deeplearningtutorial.html>

Sanjeev Arora有一个技巧，即在严谨的同时，为结果呈现直觉。这里有一个例子，在很多维度上超立方体是以什么样的方式像“尖尖的球体”？



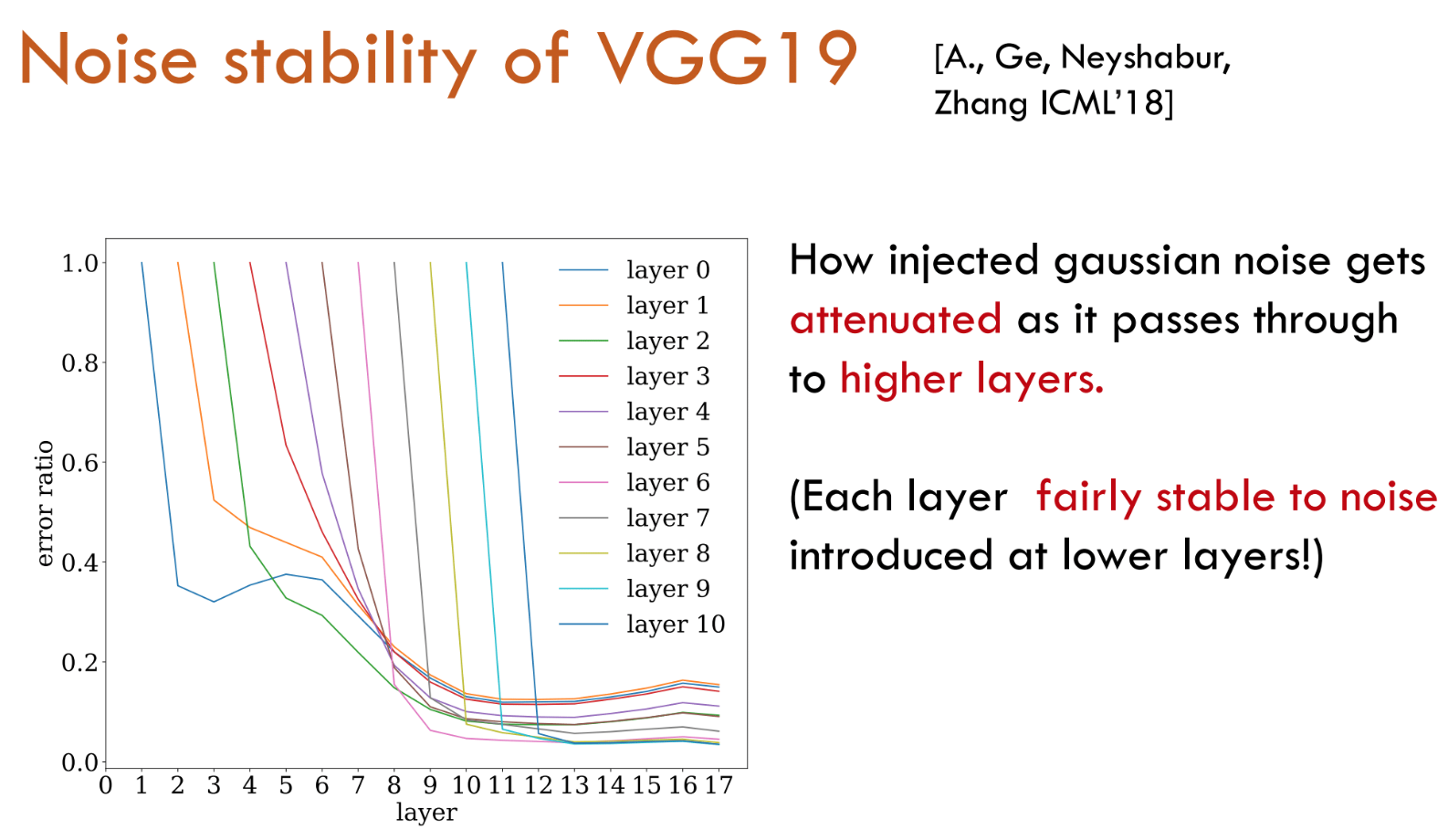
在这篇教程中，他总结了当前（大多是贫瘠的）数学成果在深度学习方面的前景。

一些结果：

* 扰动的SGD可以逃离鞍点
* 扰动SGD可以避开浅层局部极小值
* 尽管存在许多局部极小值，一些特殊的网络结构仍然可以优化为全局极小值
* 在最坏的情况下，甘人是不可能训练的

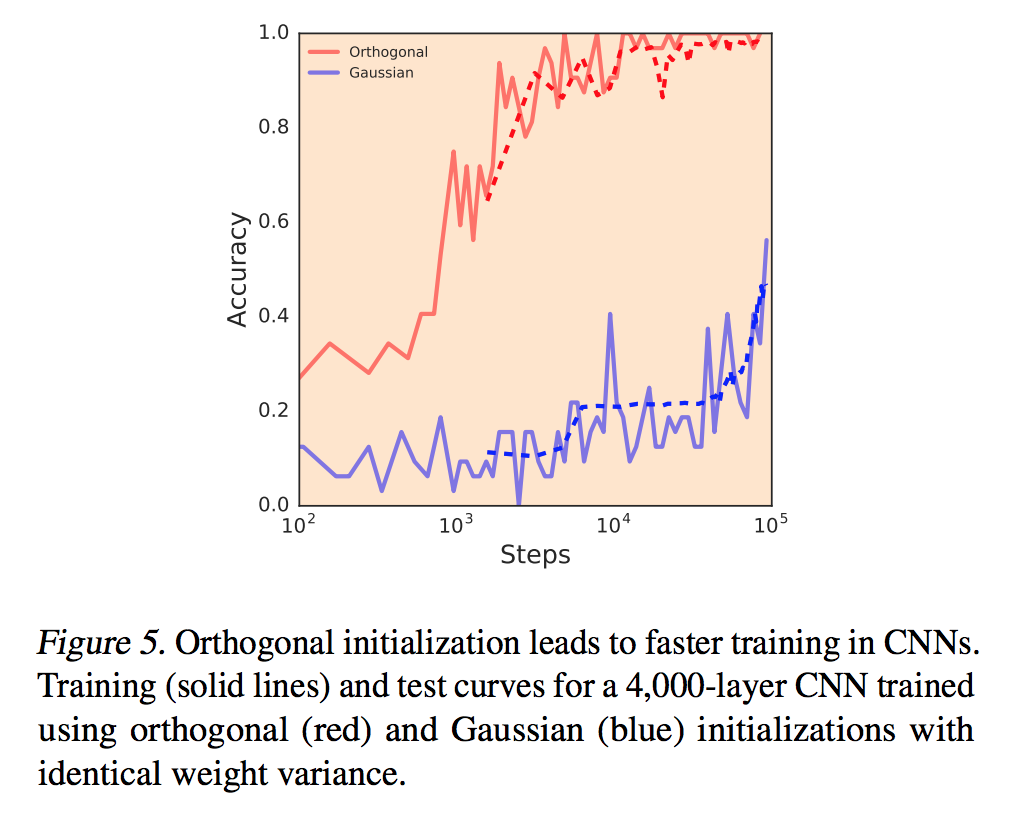
后来，他又去报道他关于量化神经网络泛化能力的工作。1M参数神经网络在100k实例上训练收敛后如何推广？这在经典统计学中是不可能的。

研究结果表明，基于神经网络的“噪声稳定性”，可以将神经网络的容量限制在参数个数以下。经过基本训练的神经网络将“拒绝”注入中间层 - 几层的高斯噪声，注入噪声被衰减，原始激活基本不变。

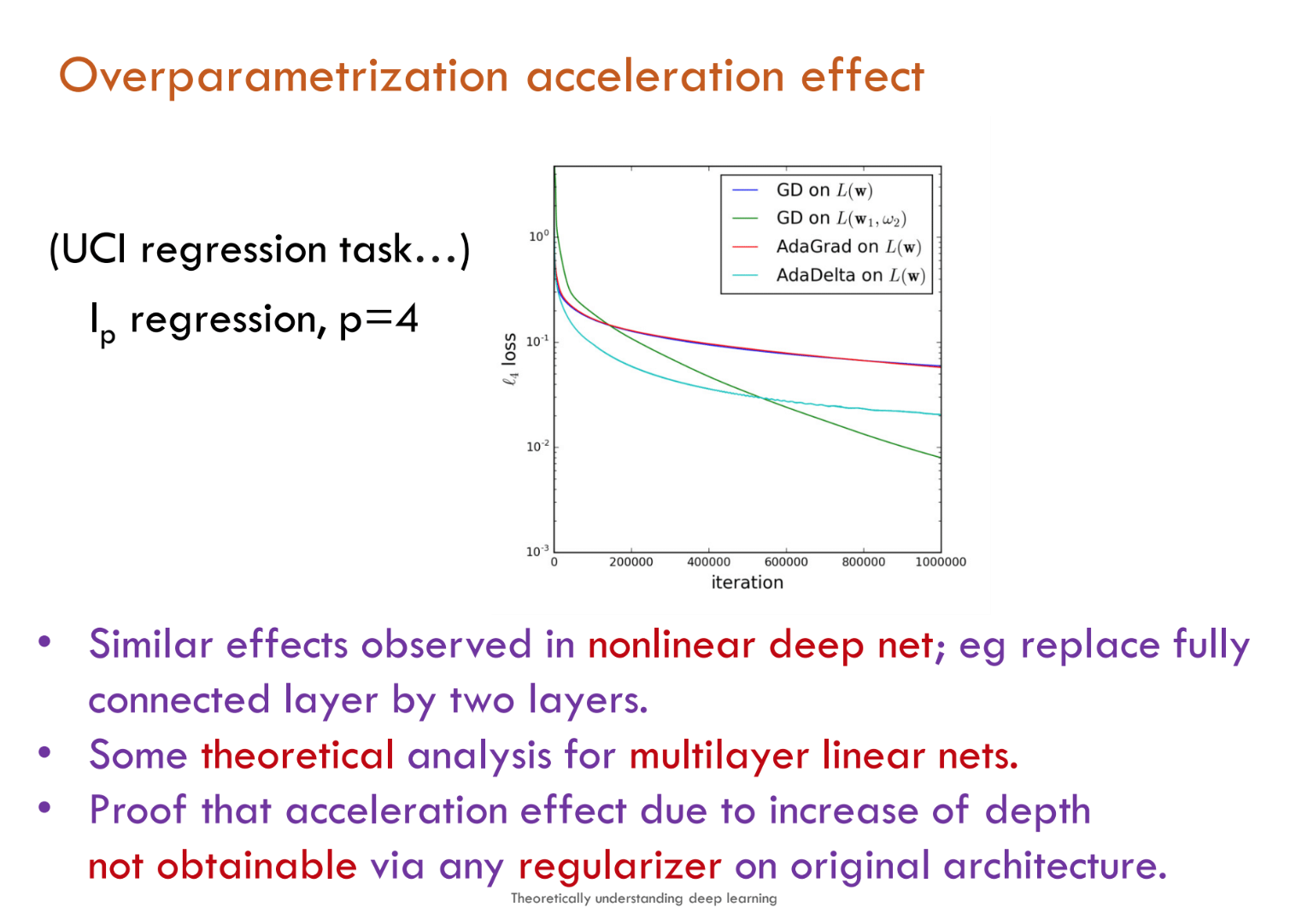


然后你可以用这个“噪声稳定性”来证明神经网络是可压缩的 - ；在不改变很多数值的情况下，抛出很多权重是可能的。可压缩性意味着推广，因为几乎没有可压缩网络。

请注意，虽然训练网络中的噪声稳定性意味着泛化，但您希望未训练网络中的噪声稳定性相反。所有的扰动都应该在没有衰减的情况下传播。用这种方法初始化神经网络，对完全连接的层使用正交矩阵，对conv层使用“delta-orthogonal”，可以在“”中训练10k层tanh网络。



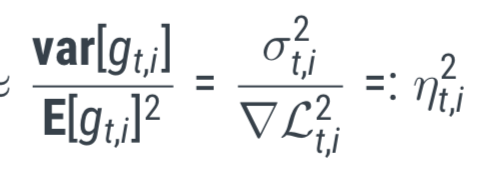
后来他谈到了“深度的作用”。与直觉相反，你可以通过加深网络来加速优化，这里有一个简单的综合优化问题。任务是L4回归。请注意，如果将其替换为二级回归，则额外的隐藏层不再有帮助。



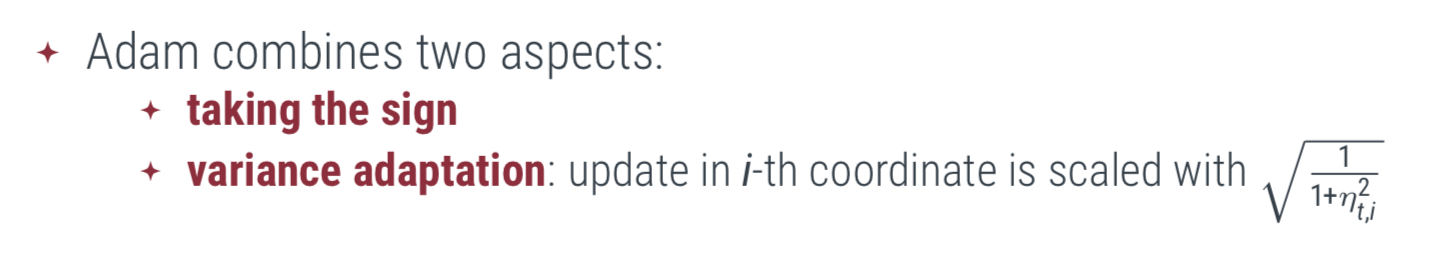
### 解剖亚当

[**[1705.07774] Dissecting Adam: The Sign, Magnitude and Variance of Stochastic Gradients** Abstract: The ADAM optimizer is exceedingly popular in the deep learning community. Often it works very well, sometimes…arxiv.org](https://arxiv.org/abs/1705.07774)

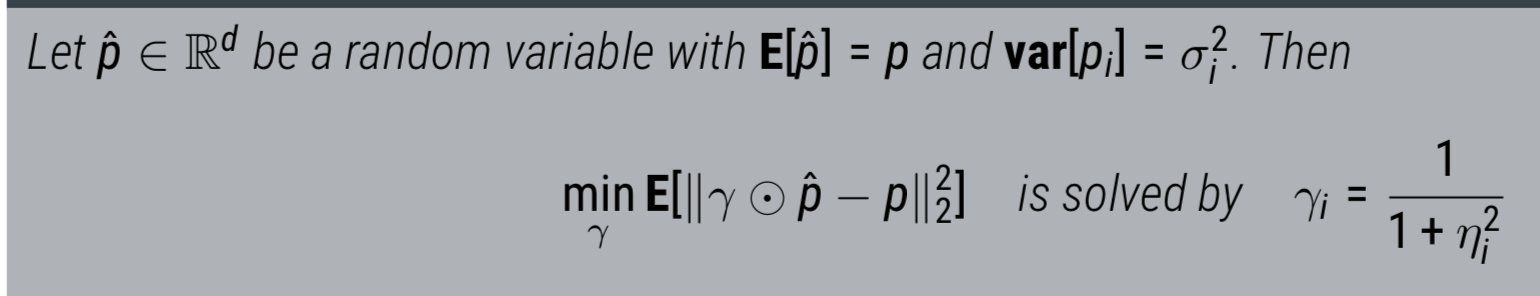
提供了一个很好的理解版本的亚当在“噪音信号比”。定义“噪声信号比”nu为梯度方差与梯度幅度的比值。在完美梯度的极端情况下，这个比率是0。



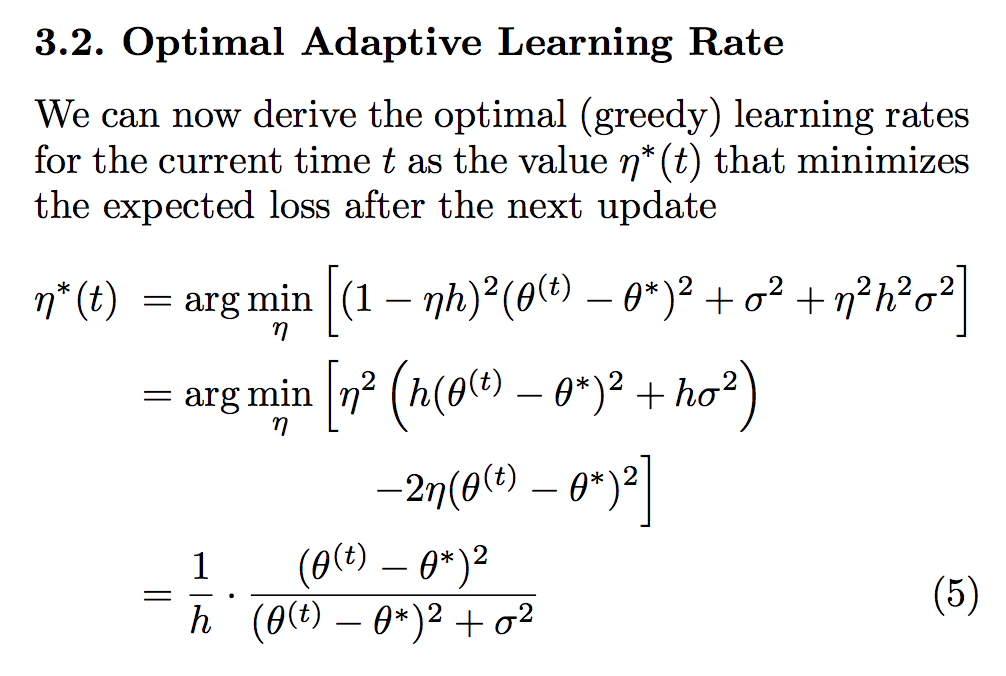
根据这个比率，Adam坐标缩放规则如下



有趣的是，它与每个坐标的“最佳”缩放率有关。



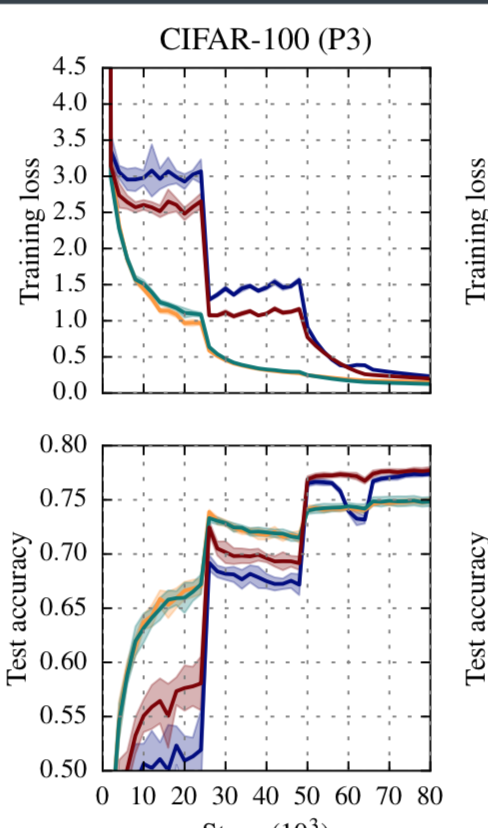
奇怪的是，在3.2的“不再讨厌的学习率”中也得到了类似的表达



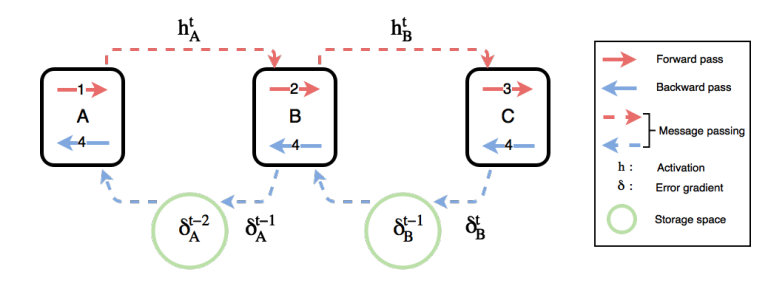
对于二次型，分子是梯度范数的平方，所以除以它得到与Adam相同的形式。

注意，Adam将平方根添加到表达式中。平方根似乎来自亚当所依据的对阿达格勒最坏情况的遗憾分析。在在线学习中，无平方根的变化称为在线牛顿法。

“平方根自由”的Adam被称为“随机方差适应梯度”，他们表明它在一些问题上有帮助。在下图中，绿色是亚当，红色是SVAG。



### 具有收敛保证的解耦并行反向传播

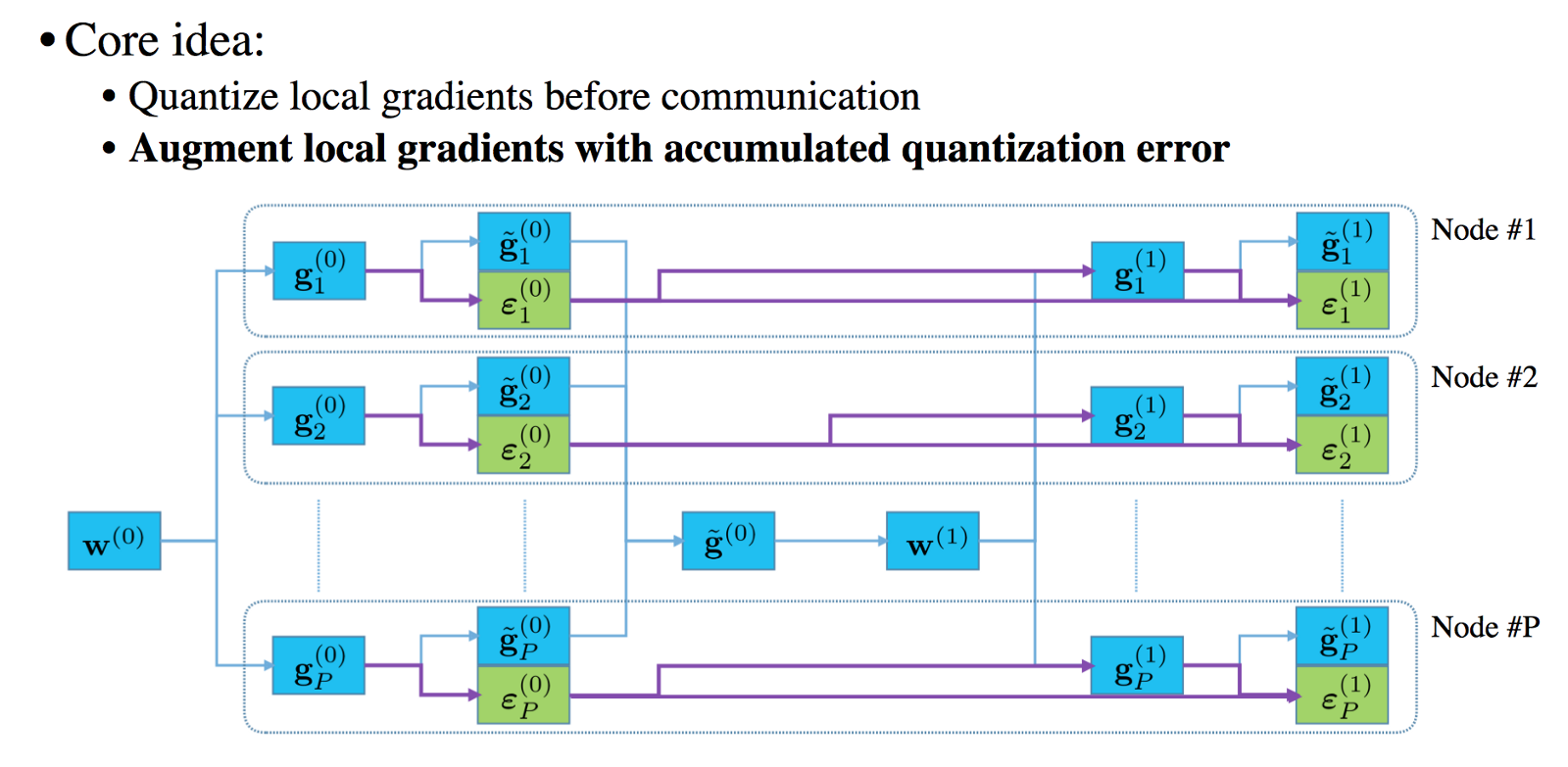


通过在不同的GPU上放置不同的层来实现跨GPU的并行化。为避免锁定，请使用流水线 - 即，当n+1层正在计算到n层的输入时，使用上一步的输入在n层中进行计算。他们看到2个GPU->4个GPU的速度增加了2倍，而且没有精度损失。这提供了并行化 - 的另一种方法，而不是沿着批处理维度并行化，而是跨层并行化。我希望这种形式的并行性会随着批量大小的增加而对模型质量产生类似的影响，因为这两种形式都可以看作是一种梯度陈旧性。

### 误差补偿量化SGD及其在大规模分布式优化中的应用

[**[1806.08054] Error Compensated Quantized SGD and its Applications to Large-scale Distributed…** Abstract: Large-scale distributed optimization is of great importance in various applications. For data-parallel based…arxiv.org](https://arxiv.org/abs/1806.08054)

值得注意的是，他们使用补偿方法来校正量化误差 - 在局部存储误差并将其折叠到下一步的梯度中

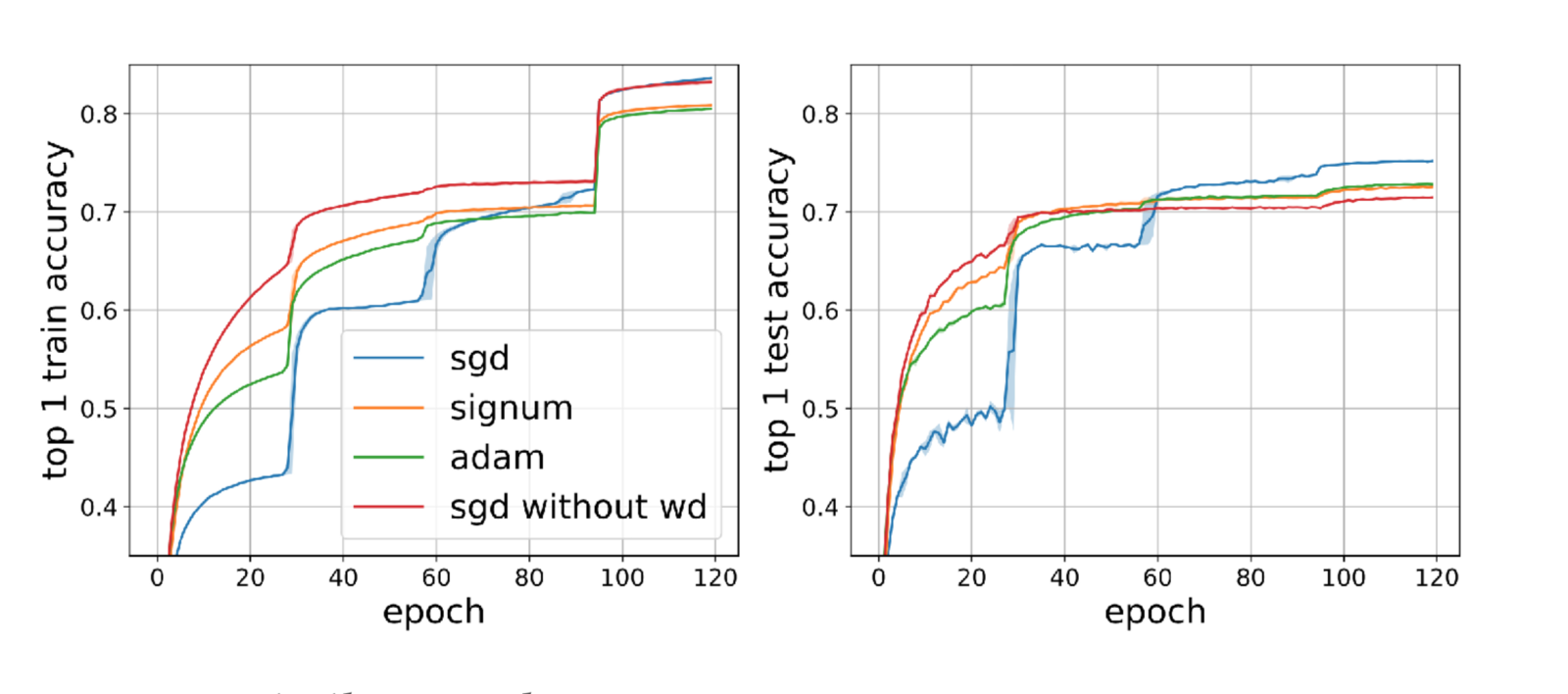


这种补偿方法也是NVidia早先在“深梯度压缩”中使用的，尽管在NVidia的情况下，他们声称只发送最重要的梯度项，就可以实现1000倍的压缩 - 。

### signSGD：非凸问题的压缩优化

[**[1802.04434] signSGD: Compressed Optimisation for Non-Convex Problems** Abstract: Training large neural networks requires distributing learning across multiple workers, where the cost of…arxiv.org](https://arxiv.org/abs/1802.04434)

不是发送渐变，而是将坐标二值化为+1或-1。对于多个工人，工人发送1/-1并进行多数投票以决定合计值。疯狂的是，与亚当相比，这并没有失去准确性。



### 逐步配料l-BFGS

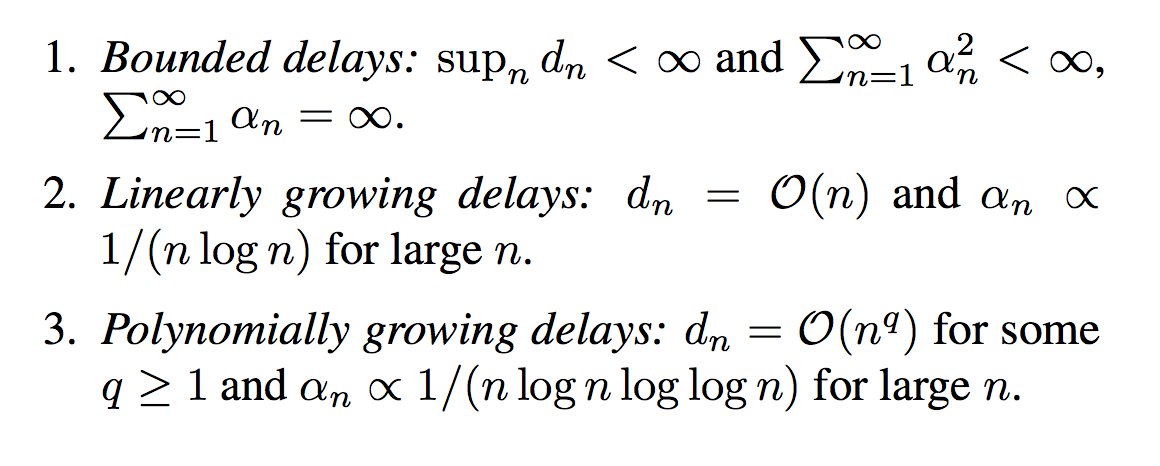
[**[1802.05374] A Progressive Batching L-BFGS Method for Machine Learning** Abstract: The standard L-BFGS method relies on gradient approximations that are not dominated by noise, so that search…arxiv.org](https://arxiv.org/abs/1802.05374)

本文的主要贡献似乎是对如何增加样本量，使l-BFGS步进可靠的试验。基本的直觉是选择足够大的批次大小，以便梯度与全批次梯度足够频繁地对齐。然后，扩展以确保其估计的随机拟牛顿方向与真实拟牛顿方向足够频繁地对齐。它们还用“随机线搜索”代替了拟牛顿法的“线搜索”部分。算法描述起来似乎很复杂，但我认为这两个贡献本身似乎很有用：

* 如何增加批次大小以响应梯度噪声的表达式
* 小批量梯度确定性线搜索的扩展

### 具有无限延迟的分布式异步优化：您能走多慢？

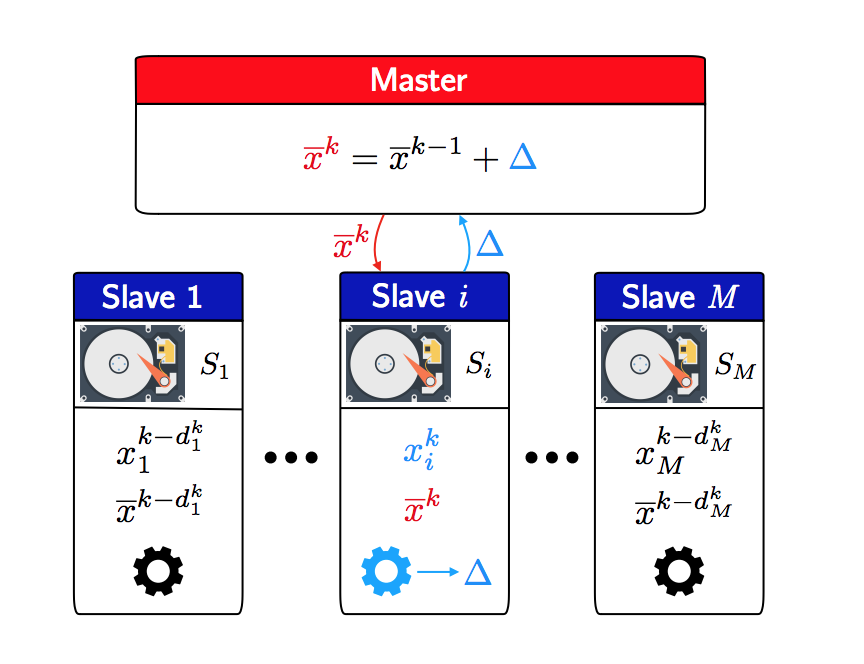
陈旧的梯度是不好的。但是，如果staleness是有界的，它们就不会那么糟糕，这在stalesynchronousparameterserver中得到了证明。而且，如果它们随时间呈次线性增长，也不会那么糟糕，如Bertesekas所示。本文证明了它们甚至可以多项式增长，并且对于一类凸问题仍然收敛到全局最小。



较大的延迟意味着步长必须收缩到零，以便在较大的梯度窗口上聚合。对于线性延迟，步长（alpha）必须比几何速率稍快。令人惊讶的是，多项式增长的时滞不需要比线性增长的时滞更快的收缩，而且多项式的依赖度也不存在。

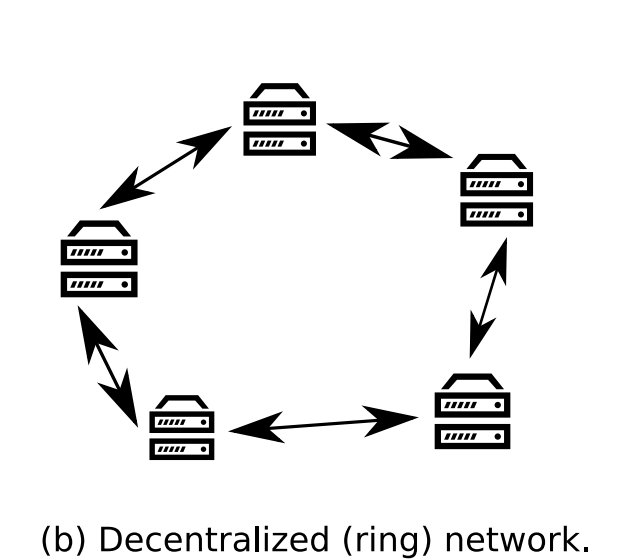
### 一种用于分布式学习的延迟容忍梯度算法

关键的思想是，它不是参数服务器平均梯度，而是平均参数值。这个想法应该是有意义的 - 一个延迟的工人将发送一个在远离当前参数服务器估计值的点上计算的梯度。对于参数服务器估计来说，这是一个不好的步骤，但是对于工作者估计来说，这是一个好的步骤。因此，让worker执行step和average参数值。基本上，组合迭代比组合梯度更稳定。此方案允许在不需要像前面的文章那样大幅缩小步长的情况下实现收敛。

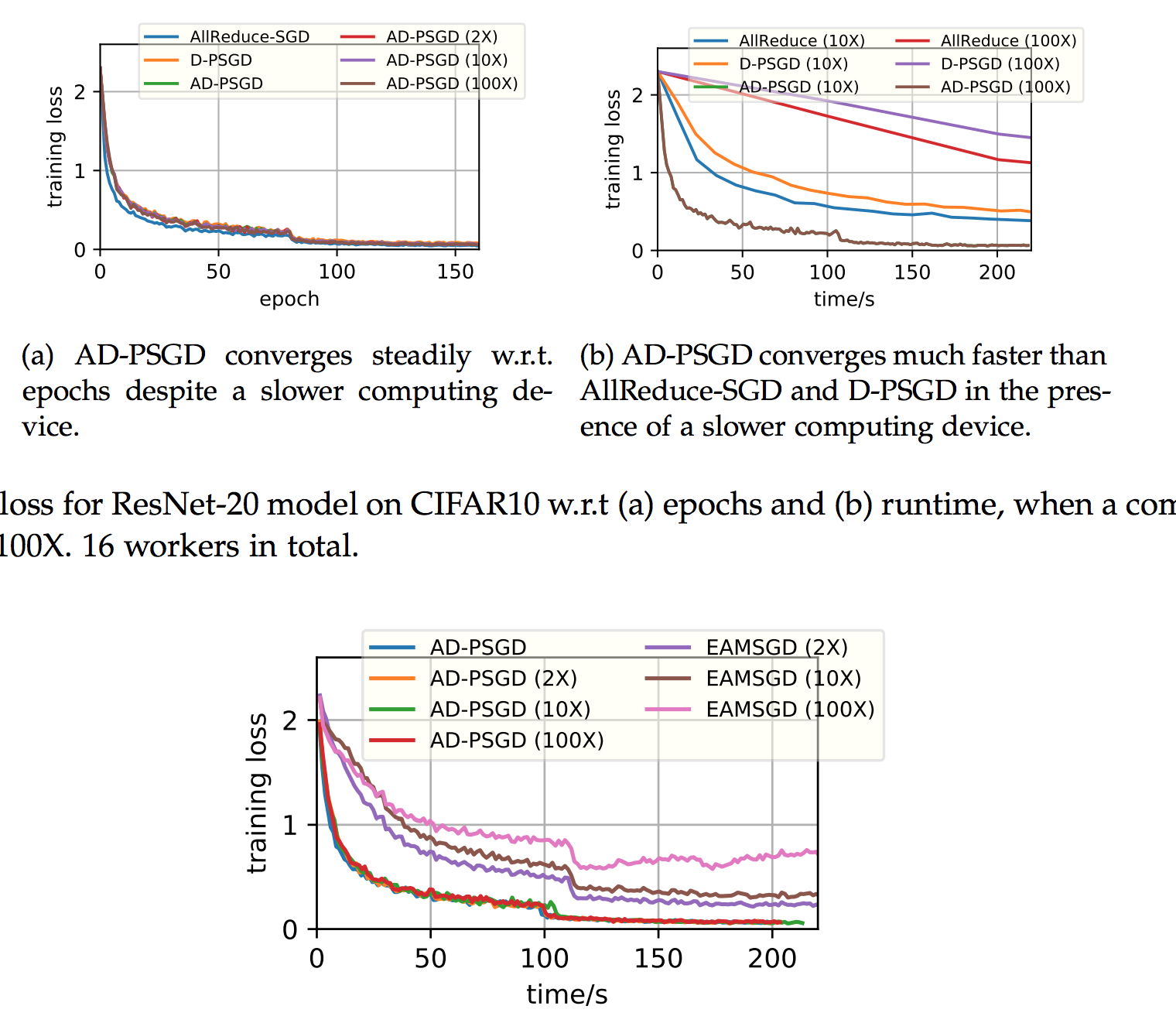


#### 异步分散并行随机梯度下降

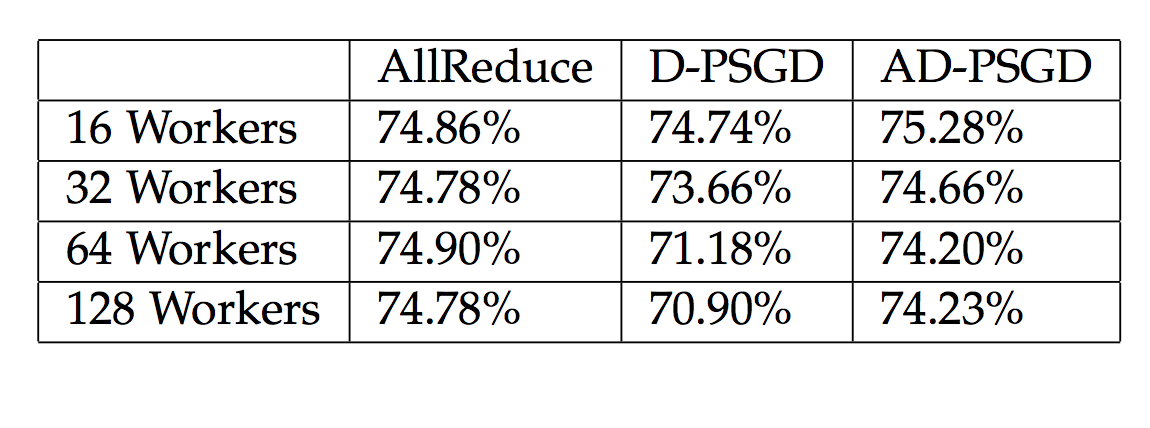
同样的想法 - 平均迭代，而不是梯度。此外，他们的体系结构是分散的：每个工人在每一步将其参数与一个随机邻居进行平均。



他们对16名工人的AllReduce和弹性平均SGD进行了比较，发现在培训吞吐量方面有更好的扩展性。当工作进程缓慢或网络连接速度减慢时，伸缩性差异尤其显著。



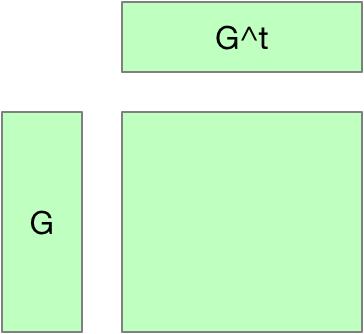
此外，出乎意料的是，在相同的时间段后，分散版本的算法（AD-PSGD）比集中版本（D-PSGD）获得了更高的精度，类似于AllReduce。



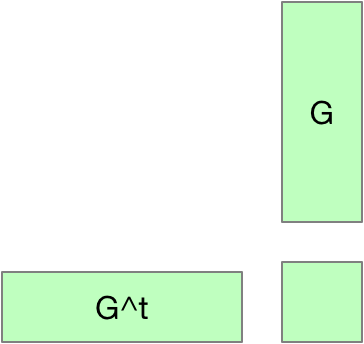
#### 自适应正则化回击

<https://arxiv.org/abs/1806.02958>

AdaGrad的原始推导需要反演历史梯度协方差矩阵。如果参数大小较大（即1米），渐变历史较小（即200米），则矩阵的形状如下所示。G的每一列都是一个梯度向量。



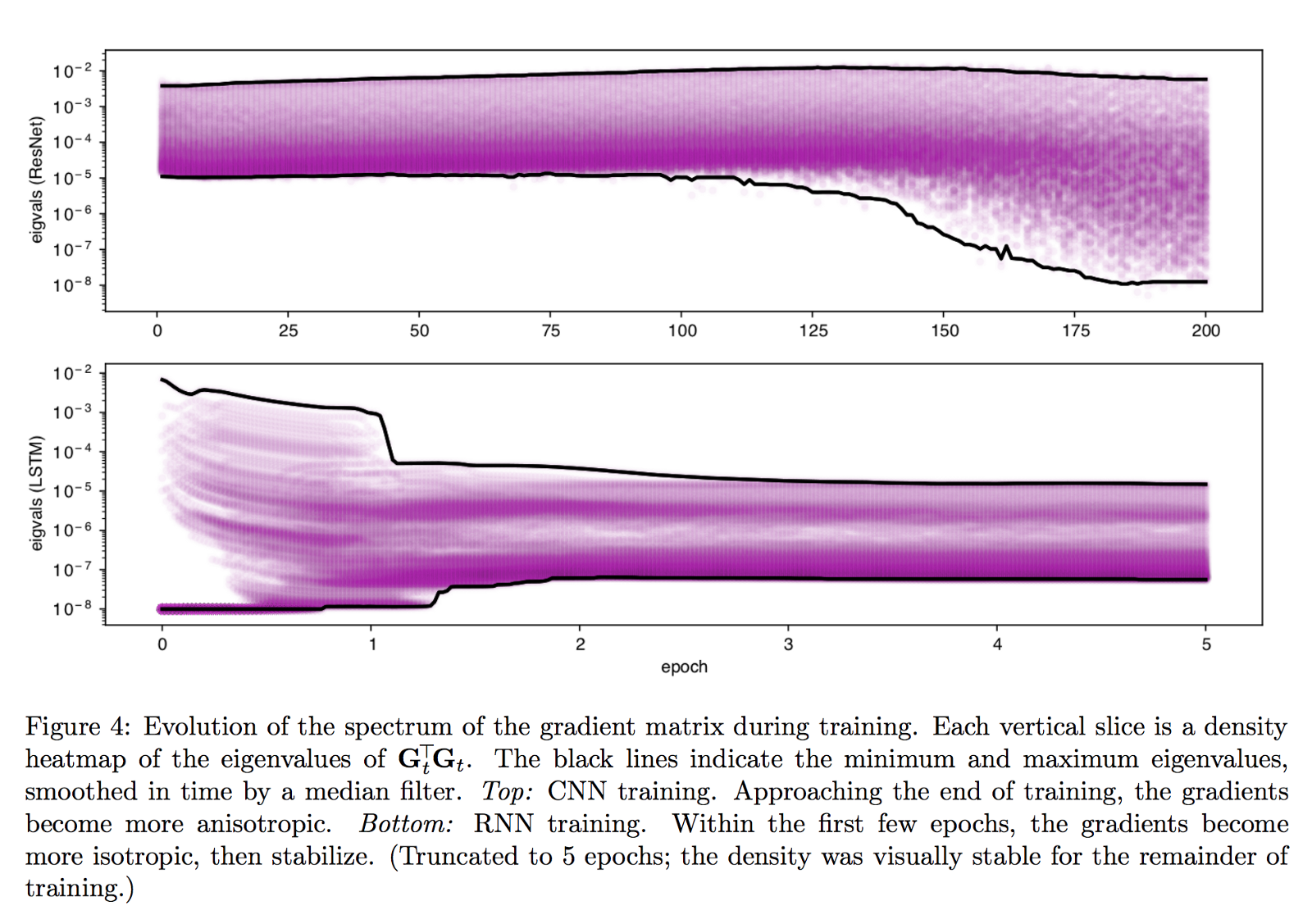
注意，这个矩阵的秩很低，并且这个逆可以用下面的矩阵更有效地计算。



不是反转1Mx1M矩阵，而是反转200x200矩阵。

他们在像PTB这样的RNN任务上获得了显著的更好的结果，在壁时钟时间上优于基线。对于CIFAR-10来说，这一改进似乎更为微不足道，他们假设RNN任务是“更非线性的”，因此从二阶预处理中可以获得更多的好处。

这个技巧还意味着您可以有效地计算（低秩）梯度协方差矩阵的特征值。绘制特征值，结果表明RNN任务梯度的条件差得多，这将使人们期望从二阶方法中获得更多的收益。

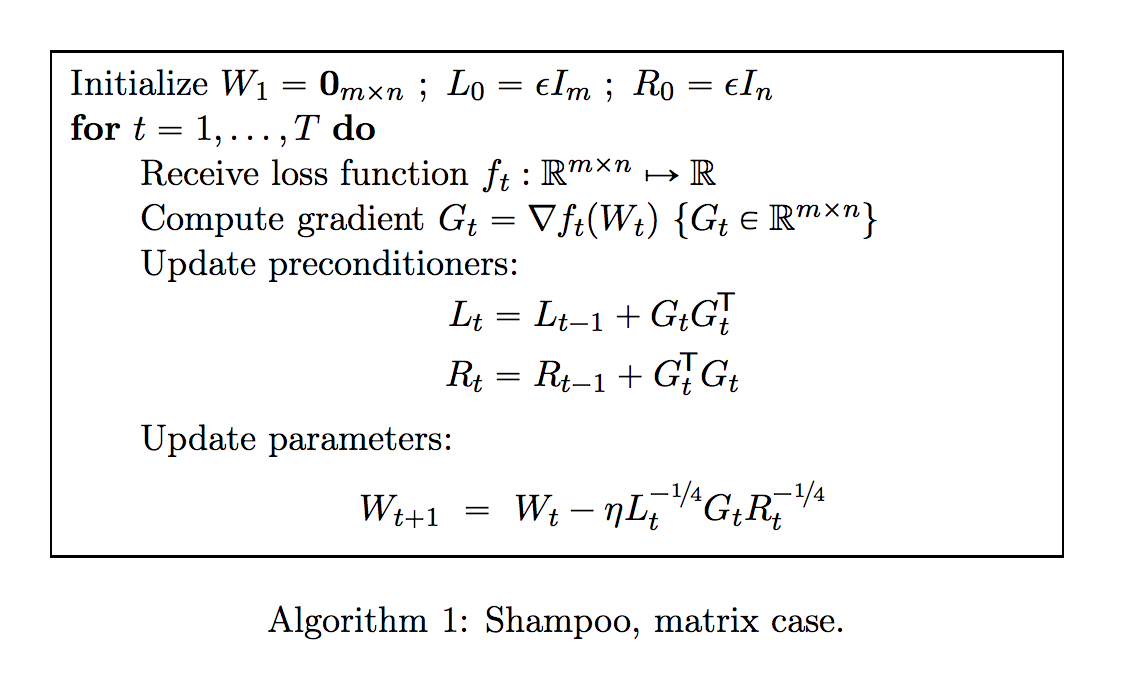


对我来说，这个情节表明RNN优化从一个狭窄的山谷开始，最后在一个碗里结束，而CNN优化从一个碗里开始，最后在一个狭窄的山谷里结束。

#### 洗发水：预条件随机张量优化

另一种使全矩阵AdaGrad有效的方法。其思想是梯度被塑造成矩阵（完全连接层）或4D张量（卷积），并且沿单个维度的预处理比每个元素便宜得多。类似的想法，梯度矩阵在左侧（列预处理）和右侧（行预处理）相乘。

However, KFAC requires access to individual activation and backprop values, whereas here they are able to construct the normalizing factors directly from gradients. The example below, L and R are the column and row normalizing factors respectively.



The impressive result is that using this optimizer they are able to do away with batch-norm layers, and optimize to 5% error on CIFAR using just convolutional layers.

