# Preserving SSD lifetime in deep learning applications with delta snapshots

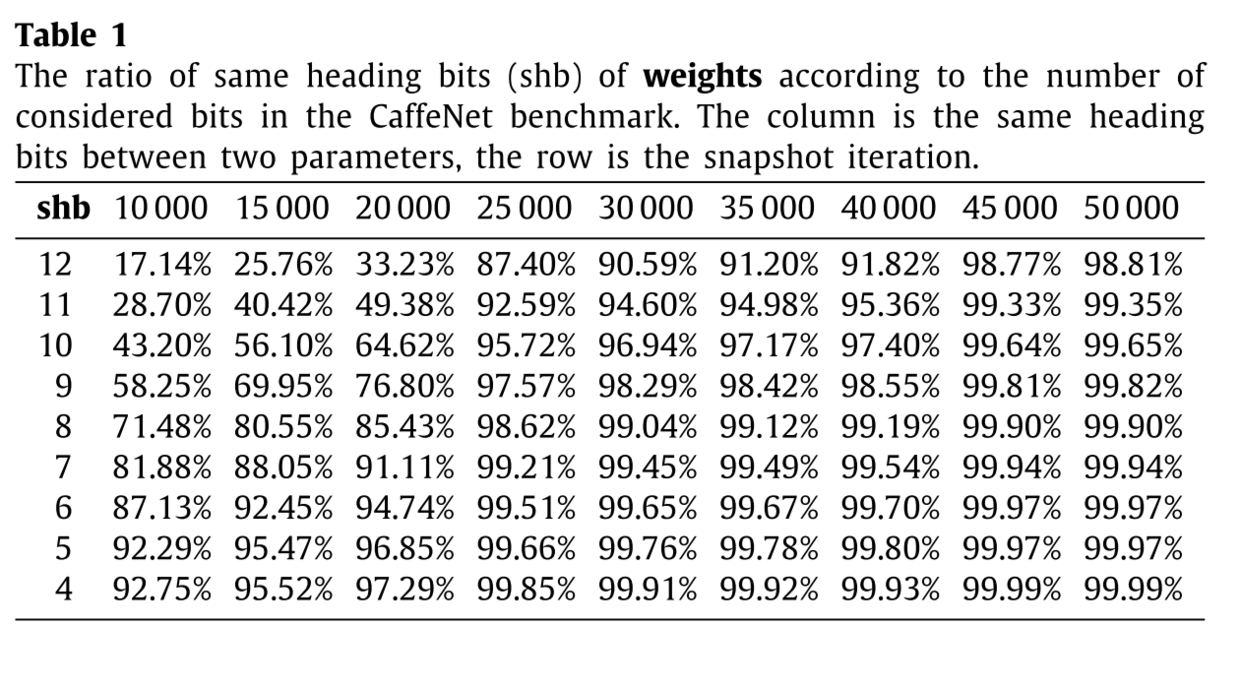
**摘要：**在大规模深度学习应用中，SSD已被广泛采用以加快训练速度。在深度学习应用程序中会定期保存快照，并将大量训练参数（TB级别）写入SSD。因此，这对固态硬盘的寿命提出了严峻的挑战。在本文中，我们首次设计了一种称为增量快照的机制，该机制可以有效减少训练阶段写入SSD的数据总量。增量快照利用快照之间的冗余信息。具体来说，我们观察到两个连续快照之间的尾数的指数部分和最高有效位几乎没有变化。基于此，我们开发了一种有效的机制来压缩快照的冗余位，以减少写入数据的大小。实验结果表明，我们的技术可以减少31％的写入数据总量和27％的擦除操作，而在训练阶段的时间开销却可以忽略不计。

## 一、Motivation

大规模的深度学习训练模型会生成大量的训练中间参数（可到TB级别），如果训练过程中失败会造成很严重的后果。于是在训练的过程中需要将中间过程以Checkpoint的形式保存到文件中。由于SSD基于erase-before-write，写放大问题会影响。

现存的方案均为粗粒度，对Checkpoint的压缩效果并不好。对于浮点变量，两个连续快照之间，尾数的指数部分和最高有效位几乎没有变化。 因此，我们设计了一种技术，可以压缩快照的冗余位以减小写入数据的大小。

表1 随着训练轮数增加相邻weight中相同位数的比例

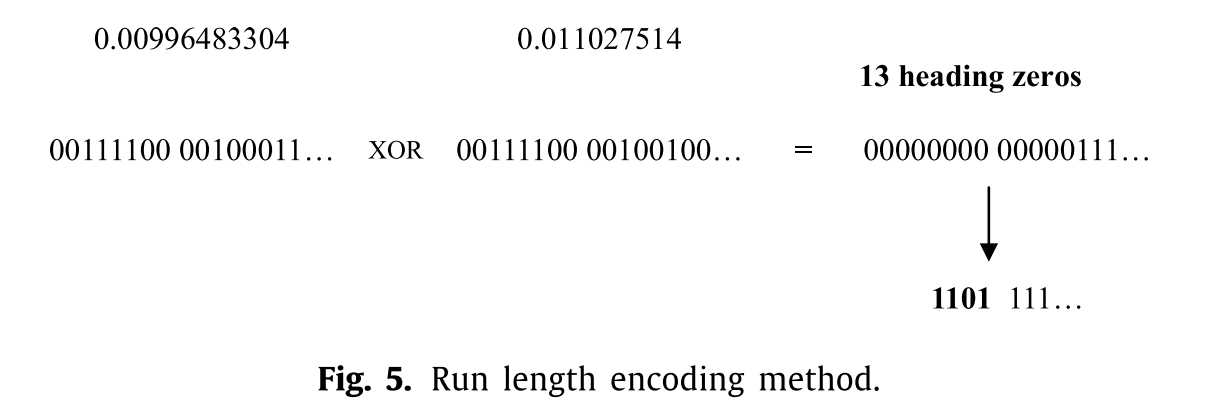


使用CaffeNet进行50000次迭代，并每5000次迭代检查一个快照。在实验中，参数是4字节的单精度浮点数（因此为32位）。两个参数之间的相似性由相同的标题比特数表示（浮点数的shb首先是符号位，指数位然后是尾数的最高有效位）。例如，如果两个权重值分别是“ 01111110”和“ 01111101”，那么会有六个相同的heading位。

由表1可知，随着训练轮数增加，相邻weight中相似的位数增加。

## 二、设计

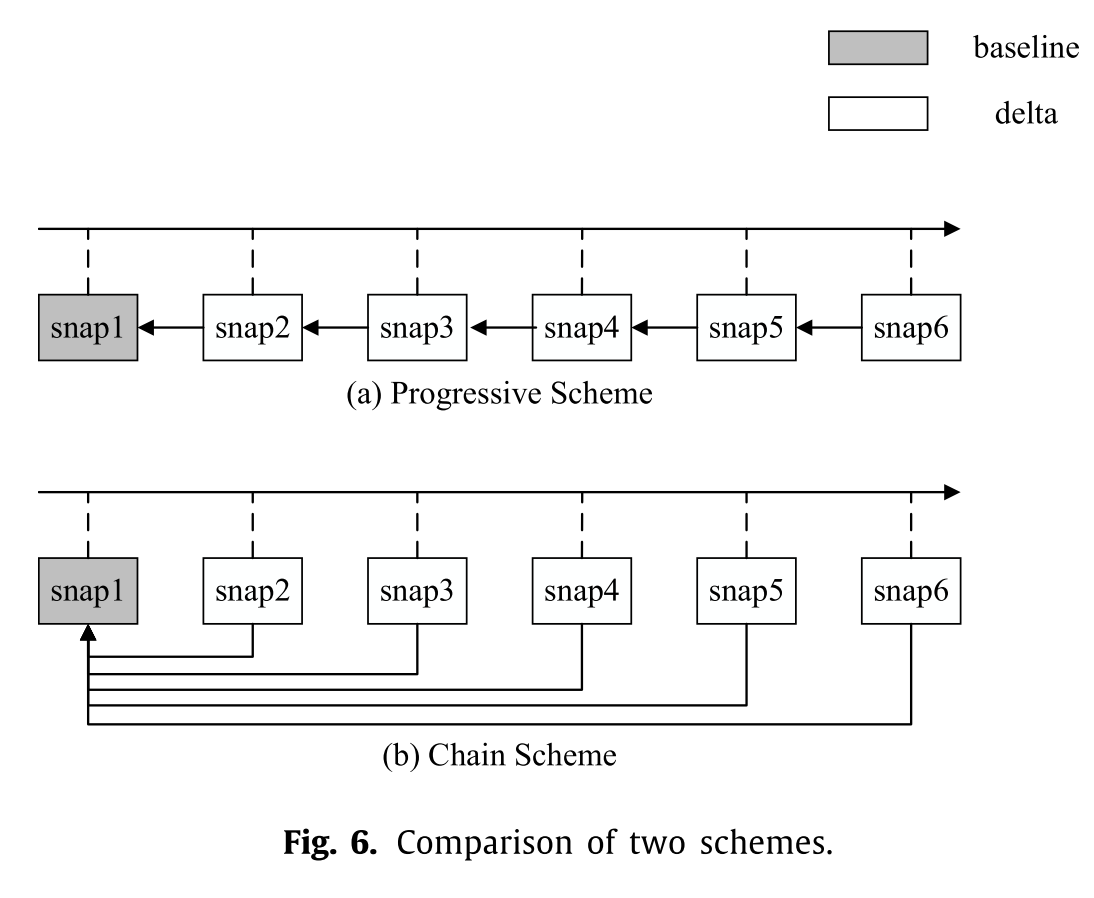
文章使用了Run Length Encoding (RLE)进行压缩（能很好的对0进行压缩）RLE使用开头的“ x”位来表示标题零的数量。在示例中有两个单精度浮点数。 它们都为32位。 然后，对它们进行XOR运算以获得增量值。 该值具有13个零位。 在此示例中，RLE技术使用4位来描述零。 因此它用代码“ 1101”替换了13个标题零，因此节省了9位（从32位开始）



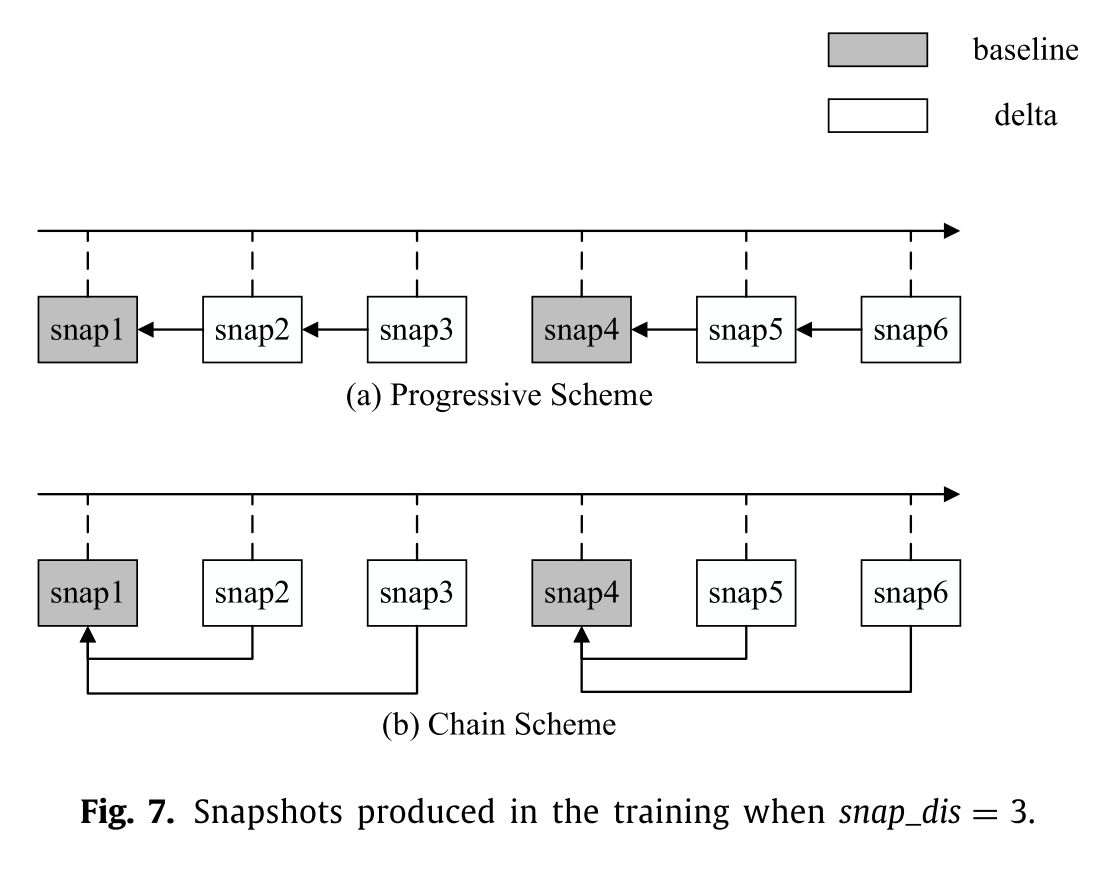
本文还设计了两种快照机制：

Progressive方案：在此方案中，始终基于前一个快照构建快照。 如果后者是基准快照，则直接计算增量值。 如果先前的快照是增量快照，那么需要从基准快照推出来当前快照。

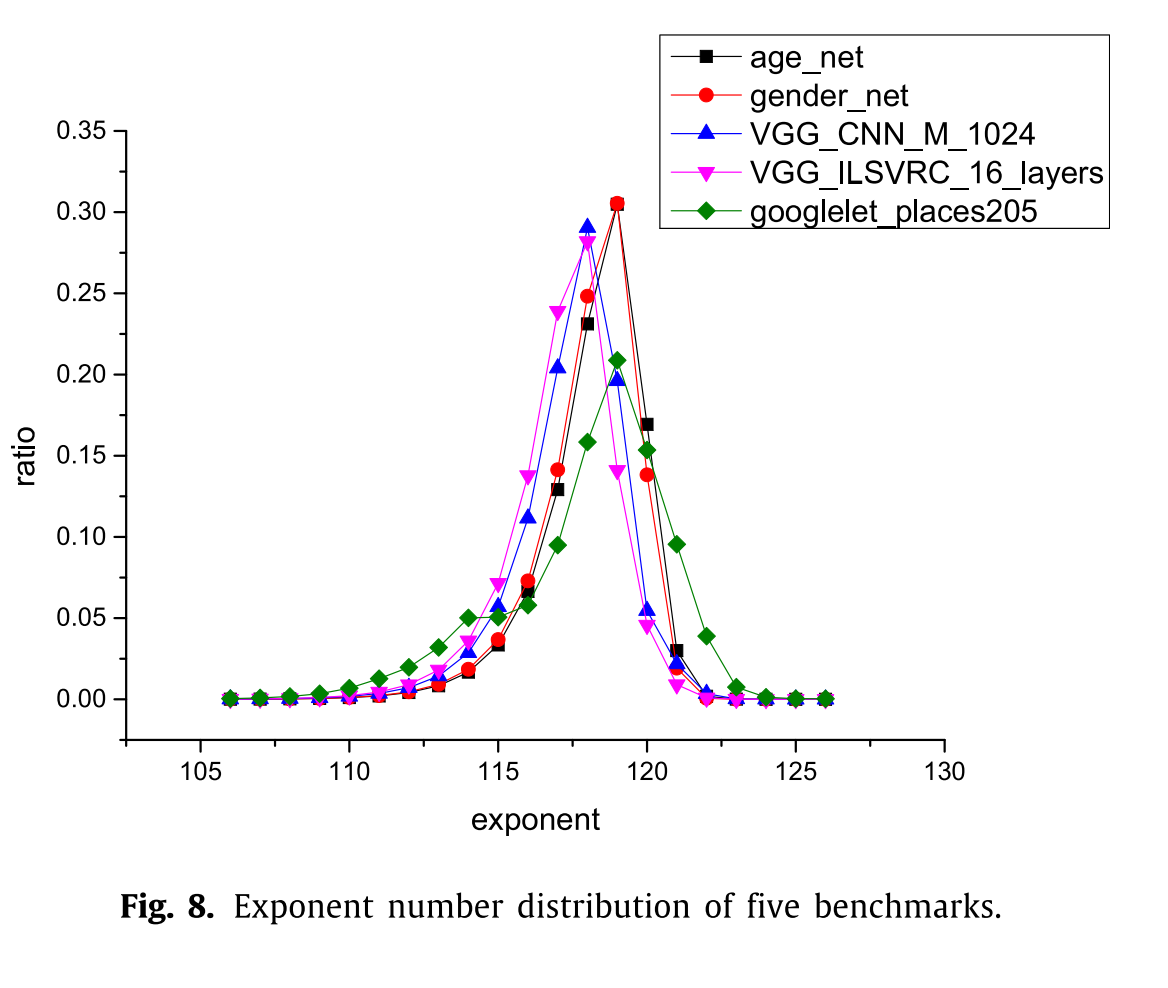
chain方案：为了避免重建先前快照的开销，我们设计了一个链方案。 这个想法是基于最近的基准快照来计算新的增量快照。



对于Progressive快照，当存储的量增加时重建的时间太长；对于chain快照，由于距离baseline太远从而导致压缩效果不好。于是设计了多基准的方案：



而对于多基准的方案来说，当snap\_dis设置的过小会导致存储snapshot过多，从而浪费空间。所以本文设计了一种方案来压缩snapshot。



上图显示了Caffe模型中指数部分的分布。 权重值是一个单精度浮点数，并且具有8位指数。 我们可以观察到大多数指数都落入一个很小的区间（从115到121）。 第一部分由指数数字组成，第二部分由其他两部分组成（符号位和尾数部分）。 我们开发了一种压缩方法来减小第一部分的大小，即使用哈夫曼编码来进行压缩。