# Title：

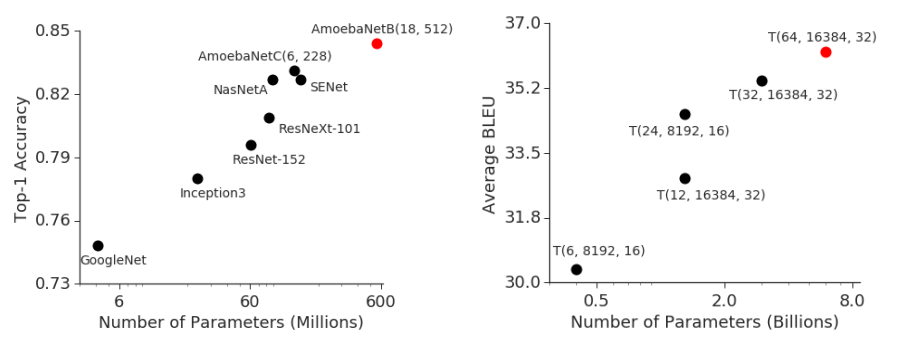
GPipe Easy Scaling with Micro-Batch Pipeline Parallelism

如果一条cpu的指令需要很多的时钟周期来进行，每个时钟周期用到的硬件可能用到的硬件不一样的情况下，可以考虑把多条指令排着序按流水执行，这样每个硬件每个时刻都在做运算，这样可以提高cpu的运行效率。同样的思想来优化在多个gpu上训练。

# Abstract：

将一个模型变大肯定是有好处的，但是与此同时内存的需求就超过了单个加速器（gpu/tpu）内存的容量。把一个模型拓展到多加速器的时候需要特殊的算法或者架构。为了满足高效的、任务无关的模型并行，作者提出了GPip这个流水线并行的包，对任何网络只要能表示成一些层串联而成，都可以使用它：把不同子序列上的层通过流水线的方式放在多个加速器上并行执行，能做到既有灵活性，性能又比较好。他使用了一个把批量分开的流水线算法。

# Introduction

但模型变大的时候，是有好处的：

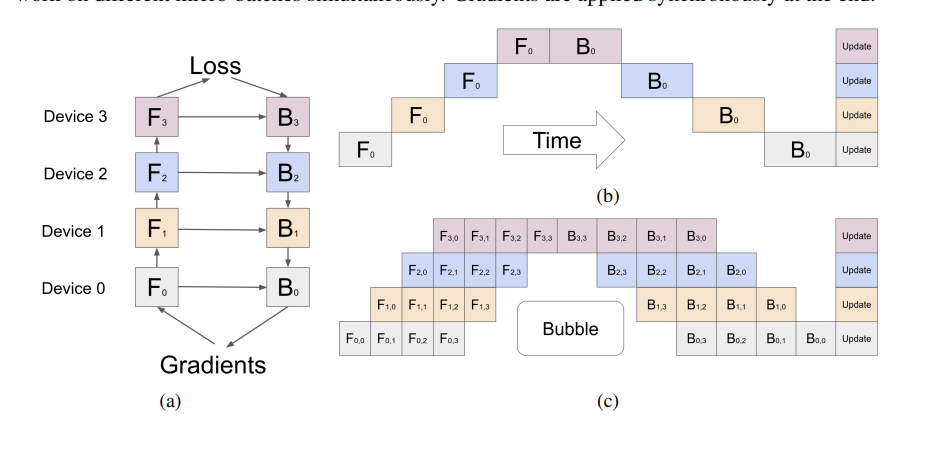
左图是卷积神经网络，右图是Transformer在机器翻译上的效果。

更大的网络遇到的问题：硬件是有局限性的（内存的限制和通信带宽的限制，更多是一个成本的问题）。这样的局限性带来了算法和系统层面的困难。如果要训练一个特别大的网络的时候，一个正常的做法叫做模型并行（把不同的模型分成不同的小块，让不同的小块放在不同的加速器上），但是这样做会带来很多问题。

GPip：re-materialization（把一些中间结果丢掉，要用到的时候再重新计算，减少一些内存的占用率）、微批量（把一个小批量再切成几块做成一个微批量，能带来流水线并行）

作者在一些网络上验证了效果怎么样。

# The GPipe Library

相当是在4个device上做了数据并行和模型并行：将一个批量切成很多微批量，然后运用流水线技术。只要微批量足够小，Bubble还是挺小的，效率可以达到不错的程度。

使用BN的时候要注意一下，算的是总的批量的均值和方差。LN就没问题。

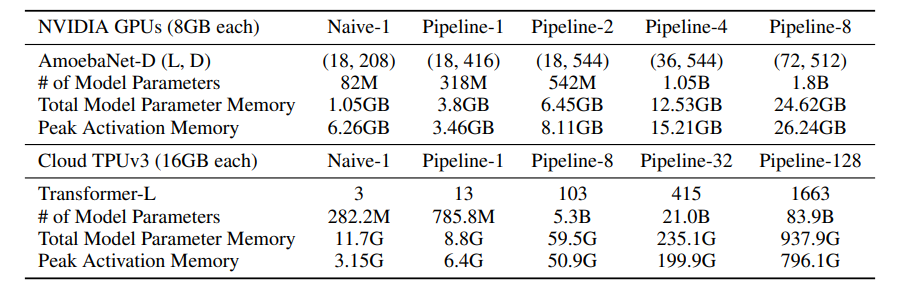
## Performance Optimization

中间结果可否被丢掉？

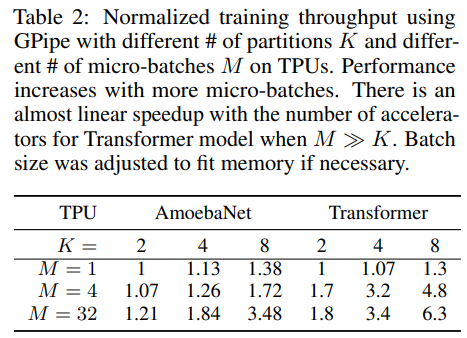
可以，用计算换空间（重新算一遍）：re-materialization

每一个加速器只存在边界处的情况。

计算开销和内存，尽量均匀的切

切的效果。

# Performance Analyses

CNN切起来比较难，Transformer比较简单。

# Conclusion

在这项工作中，我们介绍了GPipe，一个可扩展的模型并行化库，用于训练巨大的神经网络。我们提出了一种新的批处理分割流水线并行算法，该算法使用同步梯度更新，允许高硬件利用率和训练稳定性的模型并行化。强调了GPipe库的三个关键属性：1）效率：通过使用一种新的批处理分割流水线算法，GPipe实现了与设备数量几乎成线性的速度提升。2）灵活性：GPipe支持任何可以表示为一系列层的深度网络。3）可靠性：GPipe利用同步梯度下降，并保证无论分区数量如何，都能保持一致的训练。