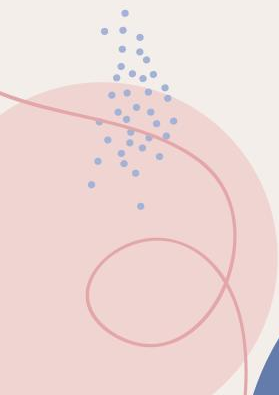


# Virtualized Deep Neural Networks for Scalable, Memory-Efficient Neural Network Design

用于可拓展、高效内存的神经网络设计的虚拟化深度神经网络



# 目录

C O N T E N T S

01

## 作者介绍及文章信息

information about the authors  
and paper

02

## 研究背景

background of this thesis

03

## vDNN的三个主要问题

three main questions

04

## vDNN++

conclusion

# 作者介绍及刊物信息

## 第一作者



**Shriram S. B.** [?](#)

**Affiliation**

Department of Computer Science and Engineering, Indian Institute of Technology, Bombay

**Publication Topics**

graphics processing units, learning (artificial intelligence), neural nets, storage management

Follow This  
Author

| Publications         | Citations? |
|----------------------|------------|
| 1                    | 3          |
| Publications by Year |            |
| <div></div>          |            |
| 2019                 | 2019       |

**Co-Authors:**

Anshuj Garg  
Purushottam Kulkarni

[Show All Co-Authors \(2\)](#)

## Shriram S. B.

毕业院校： Department of Computer Science and Engineering, Indian Institute of Technology, Bombay  
研究方向： graphics processing units,learning (artificial intelligence),neural nets,storage management

## 发表刊物

该文章引用数为28，发表在IPDPS上，属于 CCF B 类。  
IPDPS为IEEE和ACMSIGARCH发起的并行处理国际会议，每年一次，在CCF会议中排名约为49，专注于高性能计算领域。

**Dynamic Memory Management For Gpu-Based Training Of Deep Neural Networks**

+ 收藏

Shriram S. B, Anshuj Garg, Purushottam Kulkarni

Deep learning has been widely adopted for different applications of artificial intelligence speech recognition, natural language processing, computer vision etc. The growing size of Deep Neural Networks (DNNs) has compelled the researchers to design memory efficient and performan...

2019 IEEE 33RD INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM (IPDPS 2019), pp.200-209, (2019)

引用 28 | 浏览 8

下载全文 引用 评论 评分

# 作者介绍及刊物信息

## 第二作者




### Anshuj Garg

H-index: 2

论文数: 2

引用数: 7

右侧是根据AMiner网站获取的他部分论文数据。

 Dynamic Memory Management For Gpu-Based Training Of Deep Neural Networks


Anshuj S. B, Anshuj Garg, Purushottam Kulkarni

Deep learning has been widely adopted for different applications of artificial intelligence speech recognition, natural language processing, computer vision etc. The growing size of Deep Neural Networks (DNNs) has compelled the researchers to design memory efficient and performan...

2019 IEEE 33RD INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM (IPDPS 2019), pp.200-209, (2019)

引用 28 | 浏览 8

人 下载全文 引 用 评 论 评 分

 Catalyst: GPU-assisted rapid memory deduplication in virtualization environments


Anshuj Garg, Debadatta Mishra, Purushottam Kulkarni

Content based page sharing techniques improve memory efficiency in virtualized systems by identifying and merging identical pages. Kernel Same-page Merging (KSM), a Linux kernel utility for page sharing, sequentially scans memory pages of virtual machines to deduplicate pages. Se...

VEE, (2017): 44-59

引用 9 | 浏览 36

人 下载全文 引 用 评 论 评 分

 Gagm: Genome Assembly On Gpu Using Mate Pairs


Ashutosh Jain, Anshuj Garg, Kolin Paul

Genome fragment assembly has long been a time and computation intensive problem in the field of bioinformatics. Many parallel assemblers have been proposed to accelerate the process but there hasn't been any effective approach proposed for GPUs. Also with the increasing power of ...

2013 20TH INTERNATIONAL CONFERENCE ON HIGH PERFORMANCE COMPUTING (HIPCC), pp.176-185, (2013)

引用 8 | 浏览 5

引 用 评 论 评 分

 Catalyst: GPU-assisted rapid memory deduplication in virtualization environments


Anshuj Garg, Debadatta Mishra, Purushottam Kulkarni

Content based page sharing techniques improve memory efficiency in virtualized systems by identifying and merging identical pages. Kernel Same-page Merging (KSM), a Linux kernel utility for page sharing, sequentially scans memory pages of virtual machines to deduplicate pages. Se...

Special Interest Group on Programming Languages, (2017): 44-59

引用 6 | 浏览 32

引 用 评 论 评 分

 Ggake: Gpu Based Genome Assembly Using K-Mer Extension


Anshuj Garg, Ashutosh Jain, Kolin Paul

The genome assembly problem involves constructing the complete genome sequence from the reads generated by the sequencers. The Next Generation Sequencing (NGS) platforms produce a large number of short reads at a very low cost. Many assemblers have been developed to work with NGS...

2013 IEEE 15TH INTERNATIONAL CONFERENCE ON HIGH PERFORMANCE COMPUTING AND COMMUNICATIONS & 2013 IEEE..., pp.1105-1112, (2013)

引用 6 | 浏览 1

引 用 评 论 评 分

 Empirical Analysis Of Hardware-Assisted Gpu Virtualization

Anshuj Garg, Purushottam Kulkarni, Uday Kurkure, Hari Sivaraman, Lan Vu

The increasing use of Graphics Processing Unit (GPUs) for accelerating compute intensive tasks and graphics-related computations has led to their inclusion in High Performance Clusters and Cloud setups. Several cloud vendors provide virtual machine instances with GPU capabilities...

2019 IEEE 33RD INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM (IPDPS 2019), pp.200-209, (2019)

引用 28 | 浏览 8

人 下载全文 引 用 评 论 评 分

# 深度神经网络

deep neural network

基于TensorFlow、PyTorch等框架，利用GPU并行计算大量参数矩阵，通过前向传播和后向传播达到优化模型的作用。



GPU消耗大量增加：  
典型GPU内存为2~24GB，复杂深度神经网络需求在20GB以上



提出vDNN：  
每次只进行一层参数的计算，计算前将数据预取到GPU，将暂时不用的数据从GPU移到CPU



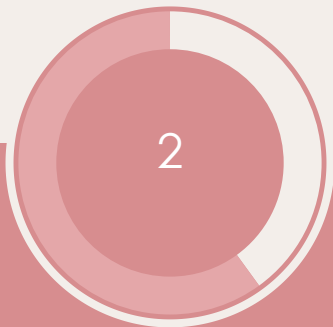
针对vDNN的三个主要问题，提出改进后的vDNN++

# vDNN的三个主要缺陷



## 正逆向传播的停滞

正向传播时，GPU需要等到前一层的数据完全取出后才开始计算；同理，反向传播时只有当前一层的数据取回GPU才开始计算。这样对GPU的计算性能会带来很大损失



## GPU内存碎片化

由于频繁的对GPU中的数据  
进行存入取出操作，导致其  
地址碎片化，且新内存池并  
不会与旧池合并，导致内存  
进一步分散。



## CPU的固定内存需求大

CPU内存有限，如果大量分配  
给GPU，会影响CPU进行原本  
的工作，还会导致其他进程  
停滞。

# vDNN++



改进的异步内存传输



可减少内存碎片化的启发式算法



压缩技术减少对固定内存的需要

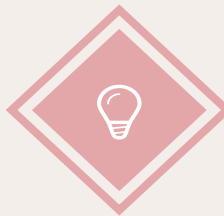
# 改进的异步内存传输

Improved Asynchronous Memory Transfer



## 正逆向传播的停滞

GPU只有当需要用到的前一层数据完全取出或取回后才进行计算，降低了运行效率。



## 异步内存传输

GPU计算与数据取出和取回同步进行。



## 正向传播

第 $n$ 层计算完成后立即计算第 $n+1$ 层，让GPU用不到的数据在计算过程中取出放入CPU内存。



## 反向传播

如果第 $m$ 层的计算需要用到第 $n$ 层的数据，而第 $n$ 层的数据需要从CPU内存取回至GPU，那么在第 $m$ 层的计算前就启动第 $n$ 层的数据的取回工作。



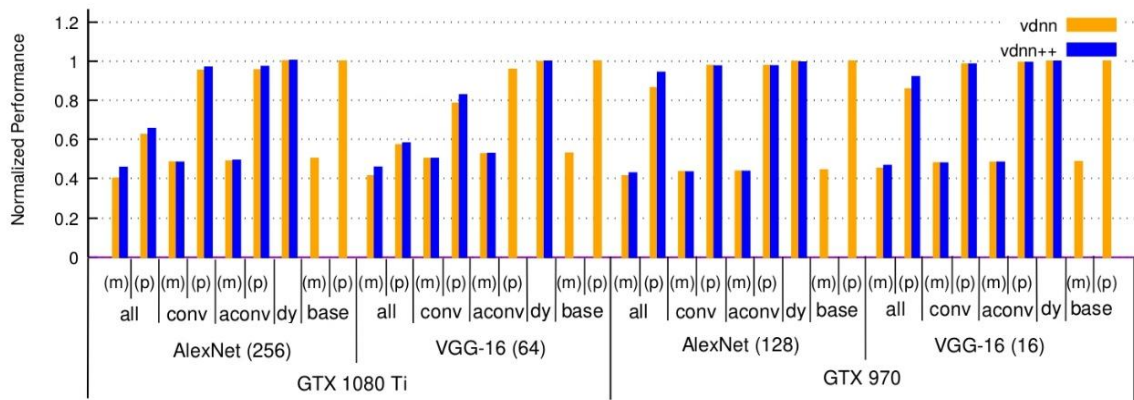
# 改进的异步内存传输

Improved Asynchronous Memory Transfer

EXPERIMENTS ON VGG-116, VGG-191 ON NVIDIA 1080 Ti -  
EXECUTION TIME OF ONE ITERATION OF TRAINING STEP IN MS

| Mode     | VGG-116 (32) |             | VGG-191 (32)  |             |
|----------|--------------|-------------|---------------|-------------|
|          | vDNN         | vDNN++      | vDNN          | vDNN++      |
| dyn      | 2717         | <b>2468</b> | 5139          | <b>5097</b> |
| conv(p)  | 3055         | <b>3018</b> | 5139          | <b>5097</b> |
| all(p)   | 3170         | <b>3129</b> | 5244          | <b>5209</b> |
| aconv(p) | 2717         | <b>2468</b> | Out of Memory |             |

看一下改进的性能提升，通过图中对比我们可以看到vDNN++相较于vDNN确实有一定的提升，经过作者量化总体性能提大概升了60%。



Performance comparison normalized with base memory manager using performance optimal CONV algorithm setup

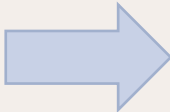
# 可减少内存碎片化的启发式算法

Heuristics to Reduce Memory Fragmentation

2

## GPU内存碎片

分配和释放空间的顺序虽然不影响内存的大小，但是会影响空间碎片化的数量。



## best-fit 启发式算法

用于处理分配请求：寻找满足请求大小的最小块（该块的内存大小肯定大于请求大小），然后将该块从小端开始分配。

## 分配策略

在分配GPU内存时，如果该数据是当前层的输入，就把它分配在大端地址上，在用完后立即释放该地址，减小碎片化数量。如果不是则采用best-fit启发式算法。

结果：数据分配在预留空间的两端，小端为暂时不取出的数据，大端为用完后就取出的数据，中间是空闲区域供分配。

# 可减少内存碎片化的启发式算法

Heuristics to Reduce Memory Fragmentation

实验中作者只处理了128维的网络，需要最小内存656MB，而改进后作者处理了256维的网络最小内存只需要343MB。可以看到，这里改进提升相当明显。

Table V

COMPARISON OF AGGREGATE PEAK MEMORY CONSUMED AND ACTUAL CNMEM POOL SIZE REQUIRED (CALCULATED BY POOL SIZE CALCULATION ALGORITHM STATED IN SECTION IV-B) FOR BEST-FIT AND NON-OFFLOADED HIGH END ALLOCATION HEURISTIC (HIGH-END).

| Network       | Mode     | aggregate (MB) | prefetch-first best-fit (MB) | derivative-first best-fit (MB) | prefetch-first high-end (MB) | derivative-first high-end (MB) |
|---------------|----------|----------------|------------------------------|--------------------------------|------------------------------|--------------------------------|
| AlexNet (256) | dyn      | 1934           | 2011                         | 2011                           | <b>1974</b>                  | <b>1974</b>                    |
|               | conv(p)  | 1934           | <b>2101</b>                  | 2185                           | 2177                         | 2114                           |
|               | conv(m)  | 1184           | 1452                         | 1476                           | <b>1427</b>                  | 1465                           |
|               | all(p)   | 1783           | 2126                         | 2050                           | 2037                         | <b>2020</b>                    |
|               | all(m)   | 1100           | 1375                         | <b>1361</b>                    | 1570                         | 1362                           |
|               | aconv(p) | 1934           | 2162                         | 2162                           | <b>2121</b>                  | 2183                           |
|               | aconv(m) | 1184           | <b>1374</b>                  | 1379                           | 1472                         | 1534                           |
| VGG-16(64)    | dyn      | 7243           | <b>7243</b>                  | <b>7243</b>                    | 7439                         | 7439                           |
|               | conv(p)  | 7243           | 7831                         | 7439                           | 8223                         | <b>7372</b>                    |
|               | conv(m)  | 3715           | 4302                         | 4107                           | 4106                         | <b>3751</b>                    |
|               | all(p)   | 7243           | <b>7341</b>                  | 8105                           | <b>7341</b>                  | 7519                           |
|               | all(m)   | 3715           | <b>3812</b>                  | 4795                           | <b>3812</b>                  | 4795                           |
|               | aconv(p) | 7243           | 8027                         | 7829                           | <b>7635</b>                  | 7831                           |
|               | aconv(m) | 4010           | 4155                         | <b>4073</b>                    | 4302                         | 4106                           |

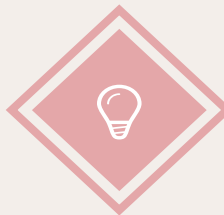
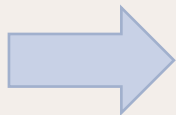
# 压缩技术减少对固定内存的需要

Compression technology reduces the demand for fixed memory

3

## CPU的固定内存需求大

CPU内存有限，如果大量分配给GPU，会影响CPU进行原本的工作，还会导致其他进程停滞。



## 压缩内存

利用ReLU输出层的稀疏特性，在CPU中完成一个硬件来压缩这些数据，以减少转移的数据量



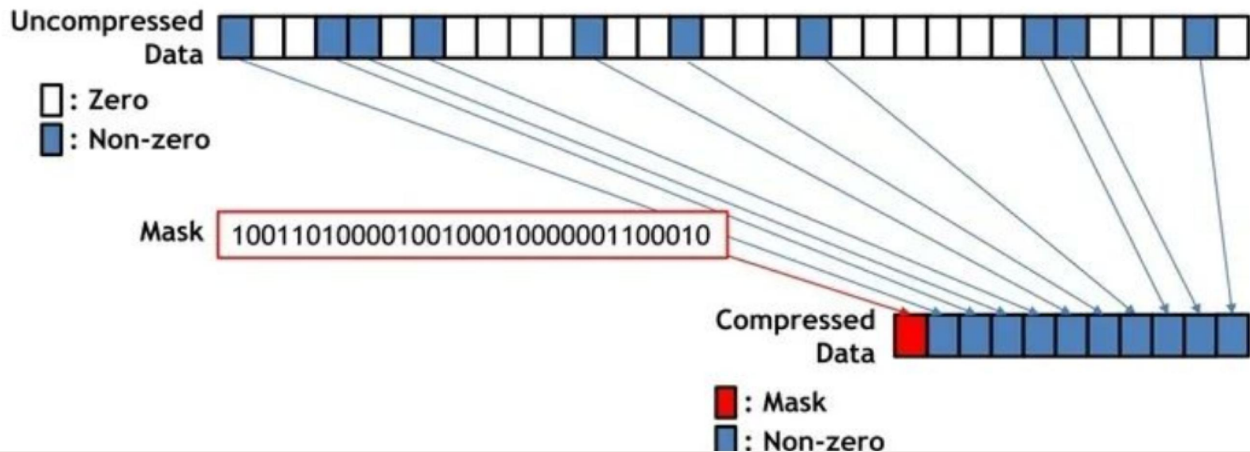
## 具体思路

在训练过程中，经常检查每层的激活密度（非零元素占比），并选择低密度的层进行压缩。

# 压缩技术减少对固定内存的需要

Compression technology reduces the demand for fixed memory

由于硬件是基于模拟器模拟的，压缩算法定制到硬件的形式也比较单一，作者并未深入研究。最后给出的性能提升大约为36%，如下图所示：



# vDNN++



改进的异步内存传输



可减少内存碎片化的启发式算法



压缩技术减少对固定内存的需要



Thank You For Your Leadership And Colleagues

**感谢观看**