Tensor Contractions with Extended BLAS Kernels on CPU and GPU

Yang Shi

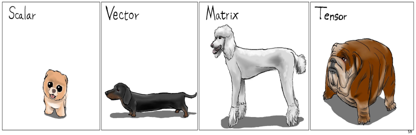
Electrical Engineering and Computer Science
University of California, Irvine

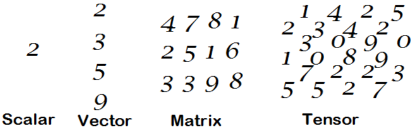
Joint work with U.N. Niranjan, Animashree Anandkumar and Cris Cecka

Southern California Machine Learning Symposium

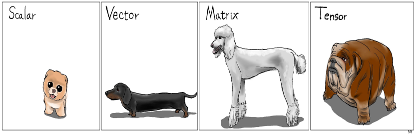
November 18, 2016

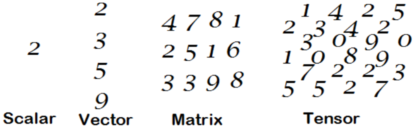
Tensor Contraction-Motivation





Tensor Contraction-Motivation

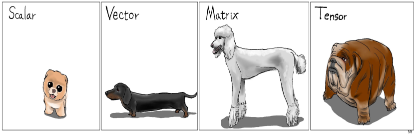


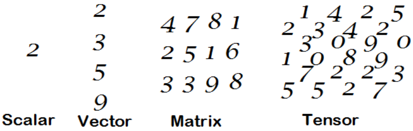


Why we need tensor?

1. Analysis of high dimension data

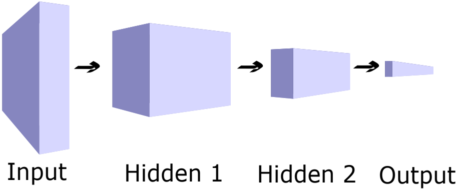
Tensor Contraction-Motivation



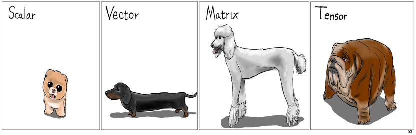


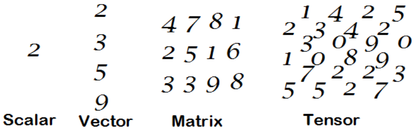
Why we need tensor?

1. Analysis of high dimension data



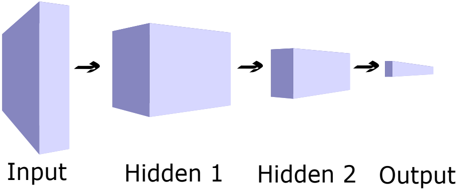
Tensor Contraction-Motivation



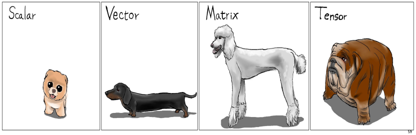


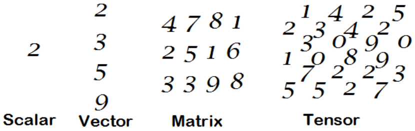
Why we need tensor?

1. Analysis of high dimension data
2. Multi-dimension relationship of low dimension data



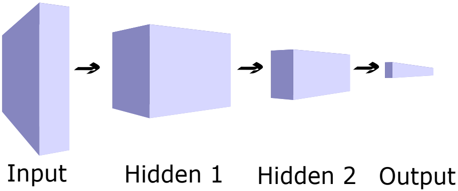
Tensor Contraction-Motivation

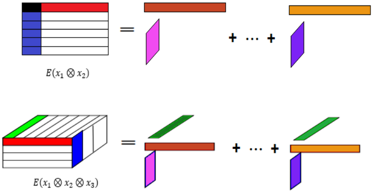




Why we need tensor?

1. Analysis of high dimension data
2. Multi-dimension relationship of low dimension data

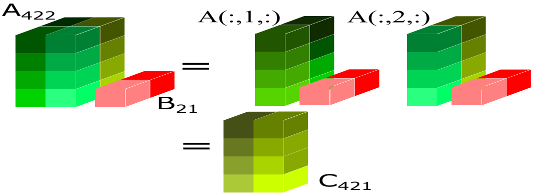




Tensor Contraction-Motivation

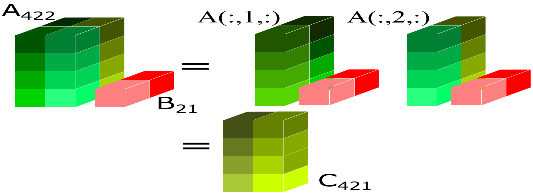
What is tensor contraction

$$C_{\mathcal{C}} = \alpha\, A_{\mathcal{A}}\, B_{\mathcal{B}} + \beta\, C_{\mathcal{C}}$$



What is tensor contraction

$$C_{\mathcal{C}} = \alpha \, A_{\mathcal{A}} \, B_{\mathcal{B}} + \beta \, C_{\mathcal{C}}$$



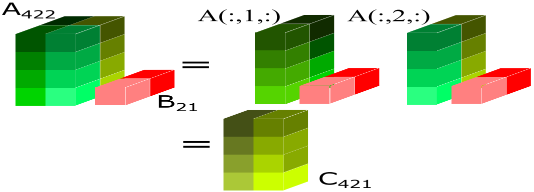
Why we need tensor contraction?

[1]Picture from M. Alex O. Vasilescu

Tensor Contraction-Motivation

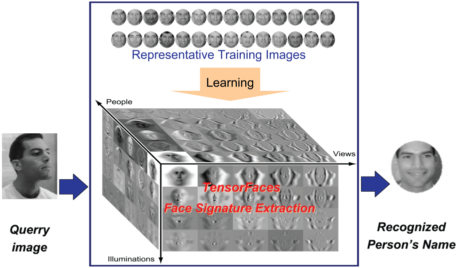
What is tensor contraction

$$C_{\mathcal{C}} = \alpha\, A_{\mathcal{A}}\, B_{\mathcal{B}} + \beta\, C_{\mathcal{C}}$$



Why we need tensor contraction?

1. Machine learning: Tensor decomposition



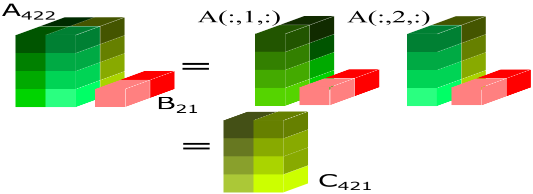
Tensor faces[1]

[1] Picture from M. Alex O. Vasilescu

Tensor Contraction-Motivation

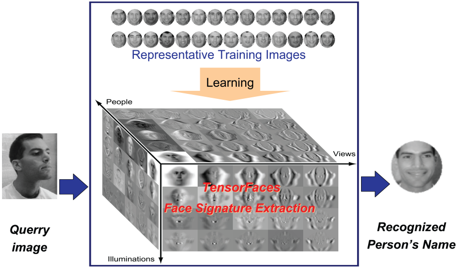
What is tensor contraction

$$C_{\mathcal{C}} = \alpha\, A_{\mathcal{A}}\, B_{\mathcal{B}} + \beta\, C_{\mathcal{C}}$$



Why we need tensor contraction?

1. Machine learning: Tensor decomposition
2. Physics
3. Chemistry



Tensor faces[1]

[1] Picture from M. Alex O. Vasilescu

Tensor Contraction-Motivation

What do we have?

What do we have?

Tensor computation libraries

1. Arbitrary/restricted tensor operation of any order and dimension
2. Such as: Matlab Tensortoolbox, BTAS, FTensor, Cyclops

Tensor Contraction-Motivation

What do we have?

Tensor computation libraries

1. Arbitrary/restricted tensor operation of any order and dimension
2. Such as: Matlab Tensortoolbox, BTAS, FTensor, Cyclops

Efficient computing frame

1. Static analysis solutions: loop reorganization/fusion
2. Parallel and distributed computing system: BatchedGEMM functions in MKL 11.3 β, CuBLAS v4.1

Tensor Contraction-Motivation

What are the limitations?

Tensor Contraction-Motivation

What are the limitations?

1. Explicit permutation takes long time in current tensor libraries:

Tensor Contraction-Motivation

What are the limitations?

1. Explicit permutation takes long time in current tensor libraries: Consider $C_{mnp} = A_{km}B_{pkn}$.
   - $A_{km} \rightarrow A_{mk}$.
   - $B_{pkn} \rightarrow B_{kpn}$.
   - $C_{mnp} \rightarrow C_{mpn}$.
   - $\boxed{C_{mpn} = A_{mk}B_{kpn}.}$
   - $C_{mpn} \rightarrow C_{mnp}$.

Tensor Contraction-Motivation

What are the limitations?

1. Explicit permutation takes long time in current tensor libraries: Consider $C_{mnp} = A_{km}B_{pkn}$.
   - $A_{km} \rightarrow A_{mk}$.
   - $B_{pkn} \rightarrow B_{kpn}$.
   - $C_{mnp} \rightarrow C_{mpn}$.
   - $\boxed{C_{mpn} = A_{mk}B_{kpn}.}$
   - $C_{mpn} \rightarrow C_{mnp}$.

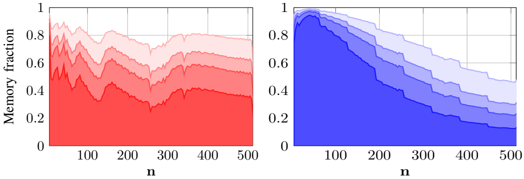


Figure: The fraction of time spent in copies/transpositions when computing $C_{mnp} = A_{mk}B_{pkn}$. Lines are shown with 1, 2, 3, and 6 total transpositions performed on either the input or output. (Left) CPU. (Right) GPU.

Overview

- **Proposed tensor operation kernel: StridedBatchedGEMM**

Overview

- **Proposed tensor operation kernel: StridedBatchedGEMM**
  1. Library-based approaches that avoid memory movement
  2. Constant-strided BatchedGEMM that has more optimization opportunities

Overview

- **Proposed tensor operation kernel: StridedBatchedGEMM**
  1. Library-based approaches that avoid memory movement
  2. Constant-strided BatchedGEMM that has more optimization opportunities
- **Provided evaluation strategies for tensor contractions**

Overview

- **Proposed tensor operation kernel: StridedBatchedGEMM**
  1. Library-based approaches that avoid memory movement
  2. Constant-strided BatchedGEMM that has more optimization opportunities
- **Provided evaluation strategies for tensor contractions**
- **Applied to tensor decomposition**

BLAS Operators

BLAS(Basic Linear Algebra Subprograms): Low-level routines for performing common linear algebra operations.

Level 1	**Level 2**	**Level 3**
$y \leftarrow \alpha x + y$	$y \leftarrow \alpha op(A)x + \beta y$	$C \leftarrow \alpha op(A)op(B) + \beta C$

Table: BLAS operators

BLAS Operators

BLAS(Basic Linear Algebra Subprograms): Low-level routines for performing common linear algebra operations.

Level 1	**Level 2**	**Level 3**
$y \leftarrow \alpha x + y$	$y \leftarrow \alpha op(A)x + \beta y$	$C \leftarrow \alpha op(A)op(B) + \beta C$

Table: BLAS operators

Example:
GEMM(ORDER, TRANSA, TRANSB, M, N, K, α, A, LDA, B, LDB, β, C, LDC)

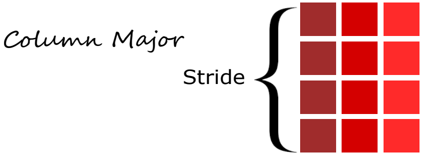
BLAS Operators

BLAS(Basic Linear Algebra Subprograms): Low-level routines for performing common linear algebra operations.

Level 1	**Level 2**	**Level 3**
$y \leftarrow \alpha x + y$	$y \leftarrow \alpha op(A)x + \beta y$	$C \leftarrow \alpha op(A)op(B) + \beta C$

Table: BLAS operators

Example:
GEMM(ORDER, TRANSA, TRANSB, M, N, K, α, A, LDA, B, LDB, β, C, LDC)



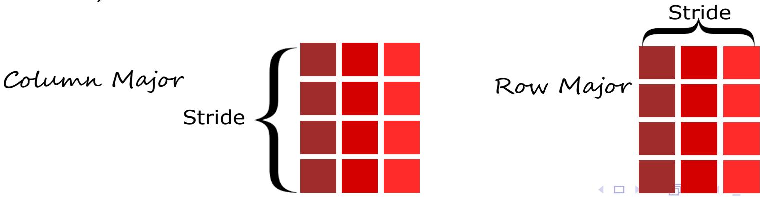
BLAS Operators

BLAS(Basic Linear Algebra Subprograms): Low-level routines for performing common linear algebra operations.

Level 1	**Level 2**	**Level 3**
$y \leftarrow \alpha x + y$	$y \leftarrow \alpha op(A)x + \beta y$	$C \leftarrow \alpha op(A)op(B) + \beta C$

Table: BLAS operators

Example:
GEMM(ORDER, TRANSA, TRANSB, M, N, K, α, A, LDA, B, LDB, β, C, LDC)



Tensor Contraction-Extended BLAS Operators

Extended BLAS Kernel for tensor one-index contraction

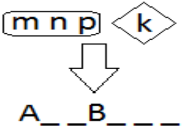
$$C = \alpha op(A) op(B) + \beta C$$

Tensor Contraction-Extended BLAS Operators

Extended BLAS Kernel for tensor one-index contraction

$$C = \alpha op(A) op(B) + \beta C$$

$$C_{mnp} = A_{**} \times B_{***}$$



If fixing C, there are total $3 \times 2 \times 3 \times 2 \times 1 = 36$ cases.

Tensor Contraction-Extended BLAS Operators

Case	Contraction	Kernel1	Kernel2	Kernel3
1.1	$A_{mk}B_{knp}$	$C_{m(np)} = A_{mk}B_{k(np)}$	$C_{mn[p]} = A_{mk}B_{kn[p]}$	$C_{m[n]p} = A_{mk}B_{k[n]p}$

Table: Case 1.1 single index contraction operations between a second-order tensor and a third-order tensor and possible mappings to Level 3 BLAS routines

Tensor Contraction-Extended BLAS Operators

Case	Contraction	Kernel1	Kernel2	Kernel3
1.1	$A_{mk}B_{knp}$	$C_{m(np)} = A_{mk}B_{k(np)}$	$C_{mn[p]} = A_{mk}B_{kn[p]}$	$C_{m[n]p} = A_{mk}B_{k[n]p}$

Table: Case 1.1 single index contraction operations between a second-order tensor and a third-order tensor and possible mappings to Level 3 BLAS routines

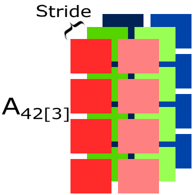
StridedBatchedGEMM(Order, TRANSA, TRANSB, M, N, K, α, A, LDA, LOA, B, LDB, LOB, β, C, LDC, LOC, P)

Tensor Contraction-Extended BLAS Operators

Case	Contraction	Kernel1	Kernel2	Kernel3
1.1	$A_{mk}B_{knp}$	$C_{m(np)} = A_{mk}B_{k(np)}$	$C_{mn[p]} = A_{mk}B_{kn[p]}$	$C_{m[n]p} = A_{mk}B_{k[n]p}$

Table: Case 1.1 single index contraction operations between a second-order tensor and a third-order tensor and possible mappings to Level 3 BLAS routines

StridedBatchedGEMM(Order, TRANSA, TRANSB, M, N, K, α, A, LDA, LOA, B, LDB, LOB, β, C, LDC, LOC, P)

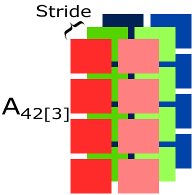


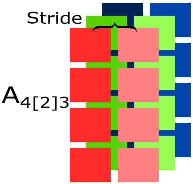
Tensor Contraction-Extended BLAS Operators

Case	Contraction	Kernel1	Kernel2	Kernel3
1.1	$A_{mk}B_{knp}$	$C_{m(np)} = A_{mk}B_{k(np)}$	$C_{mn[p]} = A_{mk}B_{kn[p]}$	$C_{m[n]p} = A_{mk}B_{k[n]p}$

Table: Case 1.1 single index contraction operations between a second-order tensor and a third-order tensor and possible mappings to Level 3 BLAS routines

StridedBatchedGEMM(Order, TRANSA, TRANSB, M, N, K, α, A, LDA, LOA, B, LDB, LOB, β, C, LDC, LOC, P)





Analysis

Flatten V.S. SBGEMM

Batching in last mode V.S. Batching in earlier mode

Mixed mode batching on input/output tensors

Flatten V.S. SBGEMM

1. A single large GEMM is more efficient
2. Flatten modes whenever possible

Batching in last mode V.S. Batching in earlier mode

Mixed mode batching on input/output tensors

Flatten V.S. SBGEMM

1. A single large GEMM is more efficient
2. Flatten modes whenever possible

Batching in last mode V.S. Batching in earlier mode

Mixed mode batching on input/output tensors

Analysis

Flatten V.S. SBGEMM

1. A single large GEMM is more efficient
2. Flatten modes whenever possible

Batching in last mode V.S. Batching in earlier mode

1. On CPU: batching in the third mode is advantageous when the size of the tensor is small
2. On GPU: there is no discernible preference in the choice of batching mode

Mixed mode batching on input/output tensors

Flatten V.S. SBGEMM

1. A single large GEMM is more efficient
2. Flatten modes whenever possible

Batching in last mode V.S. Batching in earlier mode

1. On CPU: batching in the third mode is advantageous when the size of the tensor is small
2. On GPU: there is no discernible preference in the choice of batching mode

Mixed mode batching on input/output tensors

Analysis

Flatten V.S. SBGEMM

1. A single large GEMM is more efficient
2. Flatten modes whenever possible

Batching in last mode V.S. Batching in earlier mode

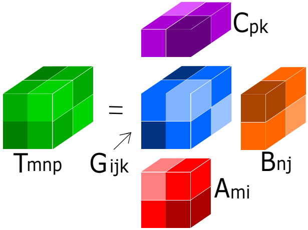
1. On CPU: batching in the third mode is advantageous when the size of the tensor is small
2. On GPU: there is no discernible preference in the choice of batching mode

Mixed mode batching on input/output tensors

1. On CPU: mode of the output tensor is more important than the batching mode of the input tensor.

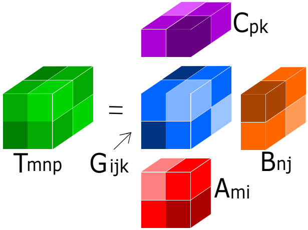
Applications:Tucker Decomposition

$$T_{mnp} = G_{ijk} A_{mi} B_{nj} C_{pk}$$



Applications:Tucker Decomposition

$$T_{mnp} = G_{ijk} A_{mi} B_{nj} C_{pk}$$



Main steps in the algorithm

- $Y_{mjk} = T_{mnp} B^{t}_{nj} C^{t}_{pk}$
- $Y_{ink} = T_{mnp} A^{t+1}_{mi} C^{t}_{pk}$
- $Y_{ijp} = T_{mnp} B^{t+1}_{nj} A^{t+1}_{mi}$

Applications:Tucker Decomposition

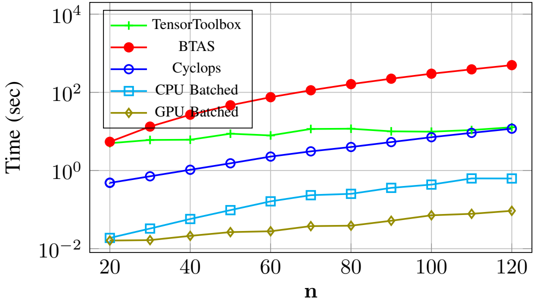


Figure: Performance on Tucker decomposition.

Conclusion

- StridedBatchedGEMM for generalized tensor contractions.
- Avoid explicit transpositions or permutations.
- 10x(GPU) and 2x(CPU) speedup on small and moderate sized tensors.
- Available in CuBLAS 8.0

Conclusion

- StridedBatchedGEMM for generalized tensor contractions.
- Avoid explicit transpositions or permutations.
- 10x(GPU) and 2x(CPU) speedup on small and moderate sized tensors.
- Available in CuBLAS 8.0
- Future works: Multi-index contractions or sparse tensor algebra.

Thank you!

More information: http://shiyangdaisy1.wixsite.com/ucieecs