Tensor Contraction with Extended BLAS Kernels on CPU and GPU

Yang Shi
University of California, Irvine, EECS

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

SIAM-ALA18
Tensor Contraction-Motivation

Why we need tensor?
Tensor Contraction-Motivation

Why we need tensor?

Modern data is inherently multi-dimensional
Tensor Contraction-Motivation

Why we need tensor?

Modern data is inherently multi-dimensional

Neural Networks

Input → Hidden 1 → Hidden 2 → Output
Tensor Contraction-Motivation

Why we need tensor?

Modern data is inherently multi-dimensional

Neural Networks

Input → Hidden 1 → Hidden 2 → Output

Method of Moment

\[ E(x_1 \otimes x_2) \]

\[ = \]

\[ + \ldots + \]

\[ E(x_1 \otimes x_2 \otimes x_3) \]
Tensor Contraction-Motivation

What is tensor contraction?
Tensor Contraction-Motivation

What is tensor contraction?

\[ C_{c} = A_{\mathcal{A}}B_{\mathcal{B}} \]
Tensor Contraction-Motivation

What is tensor contraction?

\[ C_C = A_A B_B \]

Why do we need tensor contraction?
Tensor Contraction—Motivation

What is tensor contraction?

\[ C_C = A_A B_B \]

Why do we need tensor contraction?

- Physics
- Chemistry
Tensor Contraction-Motivation

Why do we need tensor contraction?
Tensor Contraction-Motivation

Why do we need tensor contraction?

- Deep Learning
Tensor Contraction-Motivation

Why do we need tensor contraction?

• Deep Learning

• Learning latent variable model with tensor decomposition
  Example: Topic modeling
Tensor Contraction-Motivation

Why do we need tensor contraction?

- Deep Learning

- Learning latent variable model with tensor decomposition
  Example: Topic modeling
Deep Learning

Why do we need tensor contraction?

- Deep Learning

- Learning latent variable model with tensor decomposition
  
  Example: Topic modeling

  \[ h: \text{Proportion of topics in a document} \]
  
  \[ h = i \text{ with prob. } w_i \]

  \[ A: \text{Topic-word matrix} \]

  \[ A(i, j) = \mathcal{P}(x_m = i | y_m = j) \]
Tensor Contraction-Motivation

Why do we need tensor contraction?

• Deep Learning

• Learning latent variable model with tensor decomposition
  Example: Topic modeling

  $h$: Proportion of topics in a document
  
  $h = i$ with prob. $w_i$

  $A$: Topic-word matrix
  
  $A(i, j) = \mathcal{P}(x_m = i | y_m = j)$

  Third order moment:
  
  $M_3 = \mathbb{E}(x \otimes x \otimes x) = \sum_i w_i a_i \otimes a_i \otimes a_i$
Tensor Contraction-Motivation

What do we have?
Tensor Contraction-Motivation

What do we have?

Tensor computation libraries:

- Arbitrary/restricted tensor operations of any order and dimension
- Such as: Matlab Tensortoolbox, BTAS, FTensor, Cyclops
Tensor Contraction-Motivation

What do we have?

Tensor computation libraries:

• Arbitrary/restricted tensor operations of any order and dimension

• Such as: Matlab TensorToolbox, BTAS, FTensor, Cyclops

Efficient computing frame:

• Static analysis solutions: loop reorganization, fusion

• Parallel and distributed computing system: BatchedGEMM functions in MKL 11.3, CuBLAS v4.1: compute many matrix-matrix multiplies at once.
Tensor Contraction-Motivation

What do we have?

Tensor computation libraries:

• Arbitrary/restricted tensor operations of any order and dimension

• Such as: Matlab Tensortoolbox, BTAS, FTensor, Cyclops

Efficient computing frame:

• Static analysis solutions: loop reorganization, fusion

• Parallel and distributed computing system: BatchedGEMM functions in MKL 11.3, CuBLAS v4.1: compute many matrix-matrix multiplies at once.
Tensor Contraction-Motivation

What are the limitations?
Tensor Contraction-Motivation

What are the limitations?

- Explicit permutation takes long time in current tensor libraries:
Tensor Contraction-Motivation

What are the limitations?

- Explicit permutation takes long time in current tensor libraries:

  Consider \( C_{mnp} = A_{km} B_{pkn} \)
What are the limitations?

• Explicit permutation takes long time in current tensor libraries:

Consider $C_{mnp} = A_{km}B_{pkn}$

$A_{km} \to A_{mk}$.
$B_{pkn} \to B_{kpn}$.
$C_{mnp} \to C_{mpn}$.

$\boxed{C_{mpn} = A_{mk}B_{kpn}}$.
$C_{mpn} \to C_{mnp}$. 
Tensor Contraction-Motivation

What are the limitations?

- Explicit permutation takes long time in current tensor libraries:

Consider \( C_{mnp} = A_{km}B_{pkn} \)

\[
\begin{align*}
A_{km} & \to A_{mk}, \\
B_{pkn} & \to B_{kpn}, \\
C_{mnp} & \to C_{mnp}, \\
C_{mpn} & = A_{mk}B_{kpn}, \\
C_{mpn} & \to C_{mnp}.
\end{align*}
\]

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

• Propose tensor operation kernel: StridedBatchedGEMM
Overview

• Propose tensor operation kernel: StridedBatchedGEMM
  • Library-based approaches that avoid memory movement
Overview

• Propose tensor operation kernel: StridedBatchedGEMM
  • Library-based approaches that avoid memory movement
  • Constant-strided BatchedGEMM that has more optimization opportunities
Overview

• Propose tensor operation kernel: StridedBatchedGEMM
  • Library-based approaches that avoid memory movement
  • Constant-strided BatchedGEMM that has more optimization opportunities

• Provide evaluation strategies for tensor contractions
Overview

- Propose tensor operation kernel: StridedBatchedGEMM
  - Library-based approaches that avoid memory movement
  - Constant-strided BatchedGEMM that has more optimization opportunities

- Provide evaluation strategies for tensor contractions

- Apply to tensor decomposition
Overview

• Propose tensor operation kernel: StridedBatchedGEMM
  • Library-based approaches that avoid memory movement
  • Constant-strided BatchedGEMM that has more optimization opportunities

• Provide evaluation strategies for tensor contractions

• Apply to tensor decomposition

• Introduce TensorLy: Tensor learning in python
BLAS Operations
BLAS Operations

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

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

<table>
<thead>
<tr>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y \leftarrow \alpha x + y$</td>
<td>$y \leftarrow \alpha \text{op}(A)x + \beta y$</td>
<td>$C \leftarrow \alpha \text{op}(A)\text{op}(B) + \beta C$</td>
</tr>
</tbody>
</table>
BLAS Operations

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

<table>
<thead>
<tr>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y \leftarrow \alpha x + y$</td>
<td>$y \leftarrow \alpha \text{op}(A)x + \beta y$</td>
<td>$C \leftarrow \alpha \text{op}(A)\text{op}(B) + \beta C$</td>
</tr>
</tbody>
</table>

Example:
GEMM(ORDER, TRANSA, TRANSB, M, N, K, $\alpha$, A, LDA, B, LDB, $\beta$, C, LDC)
BLAS Operations

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

<table>
<thead>
<tr>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y \leftarrow \alpha x + y$</td>
<td>$y \leftarrow \alpha \text{op}(A)x + \beta y$</td>
<td>$C \leftarrow \alpha \text{op}(A)\text{op}(B) + \beta C$</td>
</tr>
</tbody>
</table>

Example:
GEMM(ORDER, TRANSA, TRANSB, M, N, K, $\alpha$, A, LDA, B, LDB, $\beta$, C, LDC)
Extended BLAS Operator
Extended BLAS Operator

Focusing: one-index contraction

Extended BLAS Kernel for tensor one-index contraction

\[ C = \alpha \text{op}(A) \text{op}(B) + \beta C \]

\[ C_{mnp} = A^{**} \times B^{***} \]
Extended BLAS Operator

Focusing: one-index contraction

Extended BLAS Kernel for tensor one-index contraction

\[ C = \alpha \text{op}(A) \text{op}(B) + \beta C \]

\[ C_{mnp} = A_{**} \times B_{***} \]

If fixing indices of C, there are total \(3 \times 2 \times 3 \times 2 \times 1 = 36\) cases.
Extended BLAS Operator
# Extended BLAS Operator

<table>
<thead>
<tr>
<th>Case</th>
<th>Contraction</th>
<th>Kernel1</th>
<th>Kernel2</th>
<th>Kernel3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>$A_{mk}B_{knp}$</td>
<td>$C_{m(np)} = A_{mk}B_{k(np)}$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn[p]}$</td>
<td>$C_{m[n]}p = A_{mk}B_{k[n]}p$</td>
</tr>
</tbody>
</table>

Table: Example: possible mappings to Level 3 BLAS routines
Extended BLAS Operator

<table>
<thead>
<tr>
<th>Case</th>
<th>Contraction</th>
<th>Kernel1</th>
<th>Kernel2</th>
<th>Kernel3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>$A_{mk}B_{knp}$</td>
<td>$C_{m(np)} = A_{mk}B_{k(np)}$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn[p]}$</td>
<td>$C_{m[n]p} = A_{mk}B_{k[n]p}$</td>
</tr>
</tbody>
</table>

Table: Example: possible mappings to Level 3 BLAS routines

StridedBatchedGEMM(ORDER, TRANSA, TRANSB, M, N, K, $\alpha$, A, LDA, LOA, B, LDB, LOB, $\beta$, C, LDC, LOC, P)
Extended BLAS Operator

<table>
<thead>
<tr>
<th>Case</th>
<th>Contraction</th>
<th>Kernel1</th>
<th>Kernel2</th>
<th>Kernel3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>$A_{mk} B_{knp}$</td>
<td>$C_{m(np)} = A_{mk} B_{k(np)}$</td>
<td>$C_{mn[p]} = A_{mk} B_{kn[p]}$</td>
<td>$C_{m[n]p} = A_{mk} B_{k[n]p}$</td>
</tr>
</tbody>
</table>

Table: Example: possible mappings to Level 3 BLAS routines

StridedBatchedGEMM(ORDER, TRANSA, TRANSB, M, N, K, $\alpha$, A, LDA, LOA, B, LDB, LOB, $\beta$, C, LDC, LOC, P)
<table>
<thead>
<tr>
<th>Case</th>
<th>Contraction</th>
<th>Kernel1</th>
<th>Kernel2</th>
<th>Kernel3</th>
<th>Case</th>
<th>Contraction</th>
<th>Kernel1</th>
<th>Kernel2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>$A_{mk}B_{knp}$</td>
<td>$C_{m(np)} = A_{mk}B_{kn}[n]p$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]$</td>
<td>4.1</td>
<td>$A_{kn}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[p]A_{kn}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[p]A_{kn}$</td>
</tr>
<tr>
<td>1.2</td>
<td>$A_{mk}B_{kpn}$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]n$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]n$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]n$</td>
<td>4.2</td>
<td>$A_{kn}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[p]A_{kn}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[p]A_{kn}$</td>
</tr>
<tr>
<td>1.3</td>
<td>$A_{mk}B_{knk}$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]$</td>
<td>4.3</td>
<td>$A_{kn}B_{kn}$</td>
<td>$C_{mn[p]} = B_{kn}^{T}[p]A_{kn}$</td>
<td>$C_{mn[p]} = B_{kn}^{T}[p]A_{kn}$</td>
</tr>
<tr>
<td>1.4</td>
<td>$A_{mk}B_{pkn}$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]$</td>
<td>4.4</td>
<td>$A_{kn}B_{pkn}$</td>
<td>$C_{mn[p]} = B_{kn}^{T}[p]A_{kn}$</td>
<td>$C_{mn[p]} = B_{kn}^{T}[p]A_{kn}$</td>
</tr>
<tr>
<td>1.5</td>
<td>$A_{mk}B_{nkp}$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]$</td>
<td>4.5</td>
<td>$A_{kn}B_{mpk}$</td>
<td>$C_{mn[p]} = B_{mpk}[p]A_{kn}$</td>
<td>$C_{mn[p]} = B_{mpk}[p]A_{kn}$</td>
</tr>
<tr>
<td>1.6</td>
<td>$A_{mk}B_{nkm}$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn}[p]$</td>
<td>4.6</td>
<td>$A_{kn}B_{mpk}$</td>
<td>$C_{mn[p]} = B_{mpk}[p]A_{kn}$</td>
<td>$C_{mn[p]} = B_{mpk}[p]A_{kn}$</td>
</tr>
<tr>
<td>2.1</td>
<td>$A_{km}B_{kn}$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>5.1</td>
<td>$A_{pk}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[mn]A_{pk}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[mn]A_{pk}$</td>
</tr>
<tr>
<td>2.2</td>
<td>$A_{km}B_{km}$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>5.2</td>
<td>$A_{pk}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[mn]A_{pk}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[mn]A_{pk}$</td>
</tr>
<tr>
<td>2.3</td>
<td>$A_{km}B_{km}$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>5.3</td>
<td>$A_{pk}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[mn]A_{pk}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[mn]A_{pk}$</td>
</tr>
<tr>
<td>2.4</td>
<td>$A_{km}B_{km}$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>5.4</td>
<td>$A_{pk}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[mn]A_{pk}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[mn]A_{pk}$</td>
</tr>
<tr>
<td>2.5</td>
<td>$A_{km}B_{km}$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>5.5</td>
<td>$A_{pk}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[mn]A_{pk}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[mn]A_{pk}$</td>
</tr>
<tr>
<td>2.6</td>
<td>$A_{km}B_{km}$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>$C_{mn[p]} = A_{km}^{T}[mn][p]$</td>
<td>5.6</td>
<td>$A_{pk}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[mn]A_{pk}$</td>
<td>$C_{mn[p]} = B_{km}^{T}[mn]A_{pk}$</td>
</tr>
<tr>
<td>3.1</td>
<td>$A_{mk}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>6.1</td>
<td>$A_{kp}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
</tr>
<tr>
<td>3.2</td>
<td>$A_{mk}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>6.2</td>
<td>$A_{kp}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
</tr>
<tr>
<td>3.3</td>
<td>$A_{mk}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>6.3</td>
<td>$A_{kp}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
</tr>
<tr>
<td>3.4</td>
<td>$A_{mk}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>6.4</td>
<td>$A_{kp}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
</tr>
<tr>
<td>3.5</td>
<td>$A_{mk}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>6.5</td>
<td>$A_{kp}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
</tr>
<tr>
<td>3.6</td>
<td>$A_{mk}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>6.6</td>
<td>$A_{kp}B_{km}$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
<td>$C_{mn[p]} = B_{km}[p]A_{mk}^{T}[n]k$</td>
</tr>
</tbody>
</table>
Table: List of 36 possible single mode contraction operations between a second-order tensor and a third-order tensor and possible mappings to Level-3 BLAS routines

<table>
<thead>
<tr>
<th>Case</th>
<th>Contraction</th>
<th>Kernel1</th>
<th>Kernel2</th>
<th>Kernel3</th>
<th>Case</th>
<th>Contraction</th>
<th>Kernel1</th>
<th>Kernel2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>$A_{km}B_{kp}$</td>
<td>$C_{m(n)p} = A_{km}B_{k(n)p}$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn[p]}$</td>
<td>$C_{m[n]p} = A_{mk}B_{k[n]p}$</td>
<td>4.1</td>
<td>$A_{kn}B_{kp}$</td>
<td>$C_{m[n]p} = B_{kn[m]}^T$ $A_{kn}$</td>
<td>$C_{m[n]p} = B_{kn[m]}^T$ $A_{kn}$</td>
</tr>
<tr>
<td>1.2</td>
<td>$A_{km}B_{kp}$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn[p]}$</td>
<td>$C_{m[n]p} = A_{mk}B_{k[n]p}$</td>
<td>$C_{m[n]p} = A_{mk}B_{k[n]p}$</td>
<td>4.2</td>
<td>$A_{kn}B_{km}$</td>
<td>$C_{m[n]p} = B_{kn[m]}^T$ $A_{kn}$</td>
<td>$C_{m[n]p} = B_{kn[m]}^T$ $A_{kn}$</td>
</tr>
<tr>
<td>1.3</td>
<td>$A_{km}B_{kn}$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn[p]}$</td>
<td>$C_{m[n]p} = A_{mk}B_{kn[p]}$</td>
<td>$C_{m[n]p} = A_{mk}B_{kn[p]}$</td>
<td>4.3</td>
<td>$A_{kn}B_{km}$</td>
<td>$C_{m[n]p] = B_{mk[p]}^T$ $A_{kn}$</td>
<td>$C_{m[n]p] = B_{mk[p]}^T$ $A_{kn}$</td>
</tr>
<tr>
<td>1.4</td>
<td>$A_{km}B_{pk}$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn[p]}$</td>
<td>$C_{m[n]p] = B_{pk[m]}^T$ $A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]}^T$ $A_{kn}$</td>
<td>4.4</td>
<td>$A_{kn}B_{pk}$</td>
<td>$C_{m[n]p] = B_{pk[m]}^T$ $A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]}^T$ $A_{kn}$</td>
</tr>
<tr>
<td>1.5</td>
<td>$A_{km}B_{np}$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn[p]}$</td>
<td>$C_{m[n]p] = B_{pk[m]}^T$ $A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]}^T$ $A_{kn}$</td>
<td>4.5</td>
<td>$A_{kn}B_{mp}$</td>
<td>$C_{m[n]p] = B_{m[p]}^T$ $A_{kn}$</td>
<td>$C_{m[n]p] = B_{m[p]}^T$ $A_{kn}$</td>
</tr>
<tr>
<td>1.6</td>
<td>$A_{km}B_{pk}$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn[p]}$</td>
<td>$C_{m[n]p] = B_{pk[m]}^T$ $A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]}^T$ $A_{kn}$</td>
<td>4.6</td>
<td>$A_{kn}B_{pm}$</td>
<td>$C_{m[n]p] = B_{m[p]}^T$ $A_{kn}$</td>
<td>$C_{m[n]p] = B_{m[p]}^T$ $A_{kn}$</td>
</tr>
<tr>
<td>2.1</td>
<td>$A_{km}B_{kp}$</td>
<td>$C_{mn[p]} = A_{km}B_{kn[p]}^T$</td>
<td>$C_{m[n]p] = B_{kn[m]}^T$ $A_{kn}$</td>
<td>$C_{m[n]p] = B_{kn[m]}^T$ $A_{kn}$</td>
<td>5.1</td>
<td>$A_{kp}B_{km}$</td>
<td>$C_{(mn)p} = B_{k(mn)k}^T$ $A_{kp}$</td>
<td>$C_{(mn)p} = B_{k(mn)k}^T$ $A_{kp}$</td>
</tr>
<tr>
<td>2.2</td>
<td>$A_{km}B_{kp}$</td>
<td>$C_{mn[p]} = A_{km}B_{kn[p]}^T$</td>
<td>$C_{m[n]p] = B_{kn[m]}^T$ $A_{kn}$</td>
<td>$C_{m[n]p] = B_{kn[m]}^T$ $A_{kn}$</td>
<td>5.2</td>
<td>$A_{kp}B_{km}$</td>
<td>$C_{m[n]p] = B_{k(mn)k}^T$ $A_{kp}$</td>
<td>$C_{m[n]p] = B_{k(mn)k}^T$ $A_{kp}$</td>
</tr>
<tr>
<td>2.3</td>
<td>$A_{km}B_{kn}$</td>
<td>$C_{mn[p]} = A_{km}B_{kn[p]}^T$</td>
<td>$C_{m[n]p] = B_{kn[m]}^T$ $A_{kn}$</td>
<td>$C_{m[n]p] = B_{kn[m]}^T$ $A_{kn}$</td>
<td>5.3</td>
<td>$A_{kp}B_{kn}$</td>
<td>$C_{m[n]p] = B_{k(mn)k}^T$ $A_{kp}$</td>
<td>$C_{m[n]p] = B_{k(mn)k}^T$ $A_{kp}$</td>
</tr>
<tr>
<td>2.4</td>
<td>$A_{km}B_{pk}$</td>
<td>$C_{mn[p]} = A_{km}B_{kn[p]}^T$</td>
<td>$C_{m[n]p] = B_{kn[m]}^T$ $A_{kn}$</td>
<td>$C_{m[n]p] = B_{kn[m]}^T$ $A_{kn}$</td>
<td>5.4</td>
<td>$A_{kp}B_{km}$</td>
<td>$C_{m[n]p] = B_{k(mn)k}^T$ $A_{kp}$</td>
<td>$C_{m[n]p] = B_{k(mn)k}^T$ $A_{kp}$</td>
</tr>
<tr>
<td>2.5</td>
<td>$A_{km}B_{np}$</td>
<td>$C_{mn[p]} = A_{km}B_{kn[p]}^T$</td>
<td>$C_{m[n]p] = B_{kn[m]}^T$ $A_{kn}$</td>
<td>$C_{m[n]p] = B_{kn[m]}^T$ $A_{kn}$</td>
<td>5.5</td>
<td>$A_{kp}B_{mk}$</td>
<td>$C_{(mn)p} = B_{m[n]k} A_{kp}$</td>
<td>$C_{(mn)p} = B_{m[n]k} A_{kp}$</td>
</tr>
<tr>
<td>2.6</td>
<td>$A_{km}B_{pk}$</td>
<td>$C_{mn[p]} = A_{km}B_{kn[p]}^T$</td>
<td>$C_{m[n]p] = B_{kn[m]}^T$ $A_{kn}$</td>
<td>$C_{m[n]p] = B_{kn[m]}^T$ $A_{kn}$</td>
<td>5.6</td>
<td>$A_{kp}B_{mk}$</td>
<td>$C_{m[n]p] = B_{m[n]k} A_{kp}$</td>
<td>$C_{m[n]p] = B_{m[n]k} A_{kp}$</td>
</tr>
<tr>
<td>3.1</td>
<td>$A_{kn}B_{km}$</td>
<td>$C_{mn[p]} = B_{kn[m]}^T A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]} A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]} A_{kn}$</td>
<td>6.1</td>
<td>$A_{kp}B_{km}$</td>
<td>$C_{(mn)p} = B_{k(mn)k}^T A_{kp}$</td>
<td>$C_{(mn)p} = B_{k(mn)k}^T A_{kp}$</td>
</tr>
<tr>
<td>3.2</td>
<td>$A_{kn}B_{km}$</td>
<td>$C_{mn[p]} = B_{kn[m]}^T A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]} A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]} A_{kn}$</td>
<td>6.2</td>
<td>$A_{kp}B_{km}$</td>
<td>$C_{m[n]p] = B_{k(mn)k} A_{kp}$</td>
<td>$C_{m[n]p] = B_{k(mn)k} A_{kp}$</td>
</tr>
<tr>
<td>3.3</td>
<td>$A_{kn}B_{km}$</td>
<td>$C_{mn[p]} = B_{kn[m]}^T A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]} A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]} A_{kn}$</td>
<td>6.3</td>
<td>$A_{kp}B_{km}$</td>
<td>$C_{m[n]p] = B_{k(mn)k} A_{kp}$</td>
<td>$C_{m[n]p] = B_{k(mn)k} A_{kp}$</td>
</tr>
<tr>
<td>3.4</td>
<td>$A_{kn}B_{km}$</td>
<td>$C_{mn[p]} = B_{kn[m]}^T A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]} A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]} A_{kn}$</td>
<td>6.4</td>
<td>$A_{kp}B_{km}$</td>
<td>$C_{m[n]p] = B_{k(mn)k} A_{kp}$</td>
<td>$C_{m[n]p] = B_{k(mn)k} A_{kp}$</td>
</tr>
<tr>
<td>3.5</td>
<td>$A_{kn}B_{km}$</td>
<td>$C_{mn[p]} = B_{kn[m]}^T A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]} A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]} A_{kn}$</td>
<td>6.5</td>
<td>$A_{kp}B_{km}$</td>
<td>$C_{m[n]p] = B_{k(mn)k} A_{kp}$</td>
<td>$C_{m[n]p] = B_{k(mn)k} A_{kp}$</td>
</tr>
<tr>
<td>3.6</td>
<td>$A_{kn}B_{km}$</td>
<td>$C_{mn[p]} = B_{kn[m]}^T A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]} A_{kn}$</td>
<td>$C_{m[n]p] = B_{pk[m]} A_{kn}$</td>
<td>6.6</td>
<td>$A_{kp}B_{km}$</td>
<td>$C_{m[n]p] = B_{k(mn)k} A_{kp}$</td>
<td>$C_{m[n]p] = B_{k(mn)k} A_{kp}$</td>
</tr>
</tbody>
</table>
Analysis

Flatten v.s. SBGEMM

<table>
<thead>
<tr>
<th>Case</th>
<th>Contraction</th>
<th>Kernel1</th>
<th>Kernel2</th>
<th>Kernel3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>$A_{mk}B_{knp}$</td>
<td>$C_{mn(p)} = A_{mk}B_{k(np)}$</td>
<td>$C_{mn(p)} = A_{mk}B_{kn[p]}$</td>
<td>$C_{m[n]p} = A_{mk}B_{k[n]p}$</td>
</tr>
</tbody>
</table>

Flattening Speedup (Batch / Flat)

Prefer flatten than SBGEMM
Analysis

Batching in last mode v.s. middle mode

<table>
<thead>
<tr>
<th>Case</th>
<th>Contraction</th>
<th>Kernel1</th>
<th>Kernel2</th>
<th>Kernel3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>$A_{mk}B_{knp}$</td>
<td>$C_{m(np)} = A_{mk}B_{kn(p)}$</td>
<td>$C_{mn[p]} = A_{mk}B_{kn[p]}$</td>
<td>$C_{m[n]p} = A_{mk}B_{kn[p]}$</td>
</tr>
<tr>
<td>2.1</td>
<td>$A_{km}B_{knp}$</td>
<td>$C_{m(np)} = A_{km}^T B_{kn(p)}$</td>
<td>$C_{mn[p]} = A_{km}^T B_{kn[p]}$</td>
<td>$C_{m[n]p} = A_{km}^T B_{kn[p]}$</td>
</tr>
</tbody>
</table>

On CPU, it’s better to batch in last mode when tensor size is small/moderate
Application: Tucker Decomposition
Application: Tucker Decomposition

\[ T_{mnp} = G_{ijk} A_{mi} B_{nj} C_{pk} \]
Application: Tucker Decomposition

\[ T_{mnp} = G_{ijk} A_{mi} B_{nj} C_{pk} \]
Application: Tucker Decomposition

\[ T_{mnp} = G_{ijk} A_{mi} B_{nj} C_{pk} \]

Main Steps:
Application: Tucker Decomposition

\[ T_{mnp} = G_{ijk} A_{mi} B_{nj} C_{pk} \]

Main Steps:

- \[ Y_{mjk} = T_{mnp} B_{nj}^t C_p^t \]
- \[ Y_{ink} = T_{mnp} A_{mi}^{t+1} C_p^t \]
- \[ Y_{ijp} = T_{mnp} B_{nj}^{t+1} A_{mi}^{t+1} \]
Application: 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.
Introduction of TensorLy

by Jean Kossaifi, Imperial College London
Yannis Panagakis, Imperial College London
Anima Anandkumar, Caltech
Introduction of TensorLy

by Jean Kossaifi, Imperial College London
Yannis Panagakis, Imperial College London
Anima Anandkumar, Caltech

• Open source

Homepage: http://tensorly.org/dev/

Github: https://github.com/tensorly/tensorly

Suitable for academic / industrial applications
Introduction of TensorLy

by Jean Kossaifi, Imperial College London
Yannis Panagakis, Imperial College London
Anima Anandkumar, Caltech

• Open source

  Homepage: http://tensorly.org/dev/

  Github: https://github.com/tensorly/tensorly

  Suitable for academic / industrial applications

• Reliability and easy to use

  Depends only on NumPy, SciPy [Optionally Matplotlib, MXNet and PyTorch]

  Exhaustive documentation, Unit-testing for all functions

  Fast
User-friendly API

Tensor decomposition

Tensor regression

Deep learning

Basic tensor operations

Unified backend

NumPy
SciPy
mxnet
PyTorch
TensorFlow
TensorLy Operators

- Kronecker
- Khatri-rao
- Hadamard products
- Tensor unfolding/folding/vectorization
- N-mode product

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kron_rao (matrices[, skip_matrix, reverse])</td>
<td>Kronecker product of a list of matrices</td>
</tr>
<tr>
<td>kronecker (matrices[, skip_matrix, reverse])</td>
<td>Kronecker product of a list of matrices</td>
</tr>
<tr>
<td>mode_dot (tensor, matrix_or_vector, mode)</td>
<td>n-mode product of a tensor and a matrix or vector at the specified mode</td>
</tr>
<tr>
<td>multi_mode_dot (tensor, matrix_or_vec_list[, ...])</td>
<td>n-mode product of a tensor and several matrices or vectors over several modes</td>
</tr>
<tr>
<td>proximal.soft_thresholding (tensor, threshold)</td>
<td>Soft-thresholding operator</td>
</tr>
<tr>
<td>proximal.svd_thresholding (matrix, threshold)</td>
<td>Singular value thresholding operator</td>
</tr>
<tr>
<td>proximal.procrustes (matrix)</td>
<td>Procrustes operator</td>
</tr>
<tr>
<td>inner (tensor1, tensor2[, n_modes])</td>
<td>Generalised inner products between tensors</td>
</tr>
</tbody>
</table>

- CANONICAL-POLYADIC (CP)
- Non-negative CP Tucker (HO-SVD)
- Non-negative Tucker
- Robust Tensor PCA

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parafac (tensor, rank[, n_iter_max, init, ...])</td>
<td>CANDECOMP/PARAFAC decomposition via alternating least squares (ALS)</td>
</tr>
<tr>
<td>non_negative_parafac (tensor, rank[, ...])</td>
<td>Non-negative CP decomposition</td>
</tr>
<tr>
<td>tucker (tensor[, rank, ranks, n_iter_max, ...])</td>
<td>Tucker decomposition via Higher Order Orthogonal Iteration (HOI)</td>
</tr>
<tr>
<td>partial_tucker (tensor, modes[, rank, ...])</td>
<td>Partial Tucker decomposition via Higher Order Orthogonal Iteration (HOI)</td>
</tr>
<tr>
<td>non_negative_tucker (tensor, rank[, ...])</td>
<td>Non-negative Tucker decomposition</td>
</tr>
<tr>
<td>robust_pca (X[, mask, tol, reg_E, reg_J, ...])</td>
<td>Robust Tensor PCA via ALM with support for missing values</td>
</tr>
</tbody>
</table>
TensorLy Example

```python
from tensorly.decomposition import parafac

factors = parafac(image, rank=50, init='random')
cp_reconstruction = tl.kruskal_to_tensor(factors)

from tensorly.decomposition import tucker

core, factors = tucker(image, ranks=(50, 50, 3), init='random')
tucker_reconstruction = tl.tucker_to_tensor(core, factors)
```
TensorLy Backend

tl.set_backend('numpy') # or 'mxnet' or 'pytorch'

```python
import tensorly as tl

T = tl.tensor([[1, 2, 3], [4, 5, 6]])
tl.tenalg.kronecker([T, T])
tl.clip(T, a_min=2, a_max=5)

tl.set_backend('mxnet')
T = tl.tensor([[1, 2, 3], [4, 5, 6]])

tl.set_backend('pytorch')
T = tl.tensor([[1, 2, 3], [4, 5, 6]])
```

NumPy ndarray
MXNet NDArray
PyTorch FloatTensor
TensorLy Example

Back-propagate through tensor operations with PyTorch

```python
import tensorly as tl
from tensorly.random import tucker_tensor

tl.set_backend('pytorch')
core, factors = tucker_tensor((5, 5, 5),
                               rank=(3, 3, 3))
core = Variable(core, requires_grad=True)
factors = [Variable(f, requires_grad=True) for f in factors]

optimiser = torch.optim.Adam([core]+factors, lr=lr)

for i in range(1, n_iter):
    optimiser.zero_grad()
    rec = tucker_to_tensor(core, factors)
    loss = (rec - tensor).pow(2).sum()
    for f in factors:
        loss = loss + 0.01*f.pow(2).sum()

    loss.backward()
    optimiser.step()
```
Contribute to TensorLy

Contributions welcome!

• If you have a cool tensor method you want to add

• If you spot a bug
Thank you!

Questions?