Figure 1: Visualization of random dense and random block-sparse weight matrices, where white indicates a weight of zero. Our new kernels allow efficient usage of block-sparse weights in fully connected and convolutional layers, as illustrated in the middle figure. For convolutional layers, the kernels allow for sparsity in input and output feature dimensions; the connectivity is still dense in the spatial dimensions. The sparsity is defined at the level of blocks (right figure), with block size of at least $8 \times 8$. At the block level, the sparsity pattern is completely configurable. Since the kernels skip computations of blocks that are zero, the computational cost is only proportional to the number of weights, not the number of input/output features.

Figure 2: Dense linear layers (left) can be replaced with layers that are sparse and wider (center) or sparse and deeper (right) while approximately retaining computational cost and memory cost. Note these costs are, in principle, proportional to the number of non-zero weights (edges). The shown networks have an equal number of edges. However, the sparse and wide network has the potential advantage of a larger information bandwidth, while the deeper network has the potential benefit of fitting nonlinear functions.

Block-sparsity unlocks various research directions (see section 6). One application we explore in experiments is the widening or deepening of neural networks, while increasing sparsity, such that the computational cost remains approximately equal as explained in figure 2. In experiments we have only scratched the surface of the applications of block-sparse linear operations; by releasing our kernels in the open, we aim to spur further advancement in model and algorithm design.

2 Capabilities

The two main components of this release are a block-sparse matrix multiplication kernel and a block-sparse convolution kernel. Both are wrapped in Tensorflow [Abadi et al., 2016] ops for easy use and the kernels are straightforward to integrate into other frameworks, such as PyTorch. Both kernels support an arbitrary block size and are optimized for 8x8, 16x16, and 32x32 block sizes. The matrix multiplication kernel supports an arbitrary block layout which is specified via a masking matrix. In addition, the feature axis is configurable. The convolution kernel supports non-contiguous input/output feature blocks of any uniform or non-uniform size specified via a configuration format (see API) though multiples of 32x32 perform best. Arbitrary dense spatial filter sizes are supported in addition to dilation, striding, padding, and edge biasing.
Sparse Tensors are Everywhere

Data Analytics
- Netflix: Movie ratings
- Amazon: Product Reviews
- Facebook: Social interactions

Machine Learning
- Convolutional Networks
- Sparse Networks [GRK 2017]
- Graph Convolutional Network [KW 2017]

Science and Engineering
- QCD
- Finite Element Method
- Quantitative Trait Loci
Sparse Tensors are Everywhere

Data Analytics
- Netflix
  - Movie ratings
- Amazon
  - Product Reviews
- Facebook
  - Social interactions

Machine Learning
- Convolutional Networks
- Sparse Networks [GRK 2017]
- Graph Convolutional Network [KW 2017]

Science and Engineering
- QCD
- Finite Element Method
- Quantitative Trait Loci
Figure 1: Visualization of random dense and random block-sparse weight matrices, where white indicates a weight of zero. Our new kernels allow efficient usage of block-sparse weights in fully connected and convolutional layers, as illustrated in the middle figure. For convolutional layers, the kernels allow for sparsity in input and output feature dimensions; the connectivity is still dense in the spatial dimensions. The sparsity is defined at the level of blocks (right figure), with block size of at least $8 \times 8$. At the block level, the sparsity pattern is completely configurable. Since the kernels skip computations of blocks that are zero, the computational cost is only proportional to the number of weights, not the number of input/output features.

Figure 2: Dense linear layers (left) can be replaced with layers that are sparse and wider (center) or sparse and deeper (right) while approximately retaining computational cost and memory cost. Note these costs are, in principle, proportional to the number of non-zero weights (edges). The shown networks have an equal number of edges. However, the sparse and wide network has the potential advantage of a larger information bandwidth, while the deeper network has the potential benefit of fitting nonlinear functions.

Block-sparsity unlocks various research directions (see section 6). One application we explore in experiments is the widening or deepening of neural networks, while increasing sparsity, such that the computational cost remains approximately equal as explained in figure 2. In experiments we have only scratched the surface of the applications of block-sparse linear operations; by releasing our kernels in the open, we aim to spur further advancement in model and algorithm design.

2 Capabilities

The two main components of this release are a block-sparse matrix multiplication kernel and a block-sparse convolution kernel. Both are wrapped in Tensorflow [Abadi et al., 2016] ops for easy use and the kernels are straightforward to integrate into other frameworks, such as PyTorch. Both kernels support an arbitrary block size and are optimized for 8x8, 16x16, and 32x32 block sizes. The matrix multiplication kernel supports an arbitrary block layout which is specified via a masking matrix. In addition, the feature axis is configurable. The convolution kernel supports non-contiguous input/output feature blocks of any uniform or non-uniform size specified via a configuration format (see API) though multiples of 32x32 perform best. Arbitrary dense spatial filter sizes are supported in addition to dilation, striding, padding, and edge biasing.

Sparse Tensors are Everywhere

Scalars have 0 dimensions
Vectors have 1 dimension
Matrices have 2 dimensions
An n-tensor has n dimensions
Sparse Tensors are Everywhere

Data Analytics

- Netflix
  - Movie ratings

Machine Learning

- Convolutional Networks
- Sparse Networks [GRK 2017]

- Graph Convolutional Network [KW 2017]

Science and Engineering

- QCD
- Finite Element Method
- Quantitative Trait Loci

- Convolutions
- Sparse Tensors
- Deep Learning

Product Reviews

- Amazon
  - Product Reviews

- Peter
  - Amazing
  - Performance

Social interactions

- Facebook
  - Social interactions
Sparse Tensors are Everywhere

Data Analytics

- Movie ratings

Machine Learning

- Convolutional Networks
  - Sparse Networks [GRK 2017]
  - Graph Convolutional Network [KW 2017]

Science and Engineering

- QCD
  - Finite Element Method

- Extremely sparse
  - Dense storage: 107 Exabytes
  - Sparse storage: 13 Gigabytes

---

**Figure 1:** Visualization of random dense and random block-sparse weight matrices, where white indicates a weight of zero. Our new kernels allow efficient usage of block-sparse weights in fully connected and convolutional layers, as illustrated in the middle figure. For convolutional layers, the kernels allow for sparsity in input and output feature dimensions; the connectivity is still dense in the spatial dimensions. The sparsity is defined at the level of blocks (right figure), with block size of at least $8\times8$. At the block level, the sparsity pattern is completely configurable. Since the kernels skip computations of blocks that are zero, the computational cost is only proportional to the number of weights, not the number of input/output features.

**Figure 2:** Dense linear layers (left) can be replaced with layers that are sparse and wider (center) or sparse and deeper (right) while approximately retaining computational cost and memory cost. Note these costs are, in principle, proportional to the number of non-zero weights (edges). The shown networks have an equal number of edges. However, the sparse and wide network has the potential advantage of a larger information bandwidth, while the deeper network has the potential benefit of fitting nonlinear functions.

Block-sparsity unlocks various research directions (see section 6). One application we explore in experiments is the widening or deepening of neural networks, while increasing sparsity, such that the computational cost remains approximately equal as explained in figure 2. In experiments we have only scratched the surface of the applications of block-sparse linear operations; by releasing our kernels in the open, we aim to spur further advancement in model and algorithm design.

**2 Capabilities**

The two main components of this release are a block-sparse matrix multiplication kernel and a block-sparse convolution kernel. Both are wrapped in Tensorflow [Abadi et al., 2016] ops for easy use and the kernels are straightforward to integrate into other frameworks, such as PyTorch. Both kernels support an arbitrary block size and are optimized for 8x8, 16x16, and 32x32 block sizes. The matrix multiplication kernel supports an arbitrary block layout which is specified via a masking matrix. In addition, the feature axis is configurable. The convolution kernel supports non-contiguous input/output feature blocks of any uniform or non-uniform size specified via a configuration format (see API) though multiples of 32x32 perform best. Arbitrary dense spatial filter sizes are supported in addition to dilation, striding, padding, and edge biasing.

**Extremely sparse**

- Dense storage: 107 Exabytes
- Sparse storage: 13 Gigabytes
Too many tensor kernels for a fixed-function library
Too many tensor kernels for a fixed-function library

\[ a = Bc + a \]
\[ a = Bc + b \quad A = B + C \quad a = \alpha Bc + \beta a \]
\[ a = B^Tc \quad A = \alpha B \quad a = B(c + d) \]
\[ a = B^Tc + d \quad A = B + C + D \quad A = BC \]
\[ A = B \odot C \quad a = b \odot c \quad A = 0 \quad A = B \odot (CD) \]
\[ A = BCd \quad A = B^T \quad a = B^T Bc \]
\[ a = b + c \quad A = B \quad K = A^T CA \]

\[ A_{ij} = \sum_{kl} B_{ikl} C_{lj} D_{kj} \quad A_{kj} = \sum_{il} B_{ikl} C_{lj} D_{ij} \]
\[ A_{lj} = \sum_{ik} B_{ikl} C_{ij} D_{kj} \quad A_{ij} = \sum_{ik} B_{ikl} c_k \]

\[ A_{ijk} = \sum_{l} B_{ikl} C_{lj} \quad A_{ik} = \sum_{j} B_{ijk} c_j \]
\[ A_{jk} = \sum_{i} B_{ijk} c_i \quad A_{ijl} = \sum_{k} B_{ikl} C_{kj} \]

\[ C = \sum_{ijkl} M_{ij} P_{jk} M_{lk} P_{il} \]
\[ a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} M_{nm} P_{no} M_{po} P_{ip} \]
\[ \tau = \sum_{i} z_i \left( \sum_{j} z_j \theta_{ij} \right) \left( \sum_{k} z_k \theta_{ik} \right) \]
Too many tensor kernels for a fixed-function library

\[ a = Bc + a \quad a = Bc \]
\[ a = Bc + b \quad A = B + C \quad a = \alpha Bc + \beta a \]
\[ a = B^Tc \quad A = \alpha B \quad a = B(c + d) \]
\[ a = B^Tc + d \quad A = B + C + D \quad a = BC \]
\[ A = B \odot C \quad a = b \odot c \quad A = 0 \quad A = B \odot (CD) \]
\[ A = BCd \quad A = B^T \quad a = B^TBc \]
\[ a = b + c \quad A = B \quad K = A^TCA \]

\[
A_{ij} = \sum_{kl} B_{ikl} C_{lj} D_{kj} \quad A_{kj} = \sum_{il} B_{ikl} C_{lj} D_{ij} \\
A_{ij} = \sum_{ik} B_{ikl} C_{ij} D_{kj} \quad A_{ij} = \sum_{ik} B_{ikl} c_k \\
A_{ijk} = \sum_{l} B_{ikl} C_{lj} \quad A_{ik} = \sum_{k} B_{ikl} c_j \\
A_{jk} = \sum_{i} B_{ijk} c_i \quad A_{ijl} = \sum_{k} B_{ikl} C_{kj} \\
C = \sum_{ijkl} M_{ij} P_{jk} M_{lk} P_{il} \\
a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} M_{nm} P_{no} M_{po} P_{ip} \]

\[ \tau = \sum_i z_i \left( \sum_j z_j \theta_{ij} \right) \left( \sum_k z_k \theta_{ik} \right) \]
Too many tensor kernels for a fixed-function library

\[
\begin{align*}
a &= Bc + a \\
a &= Bc + b \quad A &= B + C \quad a &= \alpha Bc + \beta a \\
a &= B^Tc \quad A &= \alpha B \quad a &= B(c + d) \\
a &= B^Tc + d \quad A &= B + C + D \quad A &= BC \\
A &= B \odot C \quad a &= b \odot c \quad A = 0 \quad A &= B \odot (CD) \\
A &= BCD \quad A &= B^T \quad a &= B^T Bc \\
a &= b + c \quad A &= B \quad K &= A^T CA
\end{align*}
\]

\[
\begin{align*}
A_{ij} &= \sum_{kl} B_{ikl} C_{lj} D_{kj} \quad A_{kj} = \sum_{il} B_{ikl} C_{lj} D_{ij} \\
A_{lj} &= \sum_{ik} B_{ikl} C_{ij} D_{kj} \quad A_{ij} = \sum_{ik} B_{ikl} c_k \\
A_{ik} &= \sum_{i} B_{ikl} C_{lj} \quad A_{ik} = \sum_{i} B_{ikl} c_j \\
A_{ij} &= \sum_{i} B_{ijl} c_i \quad A_{ij} = \sum_{i} B_{ikl} C_{kj} \\
C &= \sum_{ijkl} M_{ij} P_{jk} M_{lk} P_{il} \\
a &= \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} M_{nm} P_{no} M_{po} P_{ip}
\end{align*}
\]

Data analytics (tensor factorization)
Too many tensor kernels for a fixed-function library

\[
a = Bc + a \\
a = Bc + b \quad A = B + C \quad a = \alpha Bc + \beta a \\
a = B^T c \quad A = \alpha B \quad a = B(c + d) \\
a = B^T c + d \quad A = B + C + D \quad A = BC \\
A = B \odot C \quad a = b \odot c \quad A = 0 \quad A = B \odot (CD) \\
A = BCd \quad A = B^T \quad a = B^T Bc \\
a = b + c \quad A = B \quad K = A^T CA
\]

\[
A_{ij} = \sum_{kl} B_{ikl} C_{lj} D_{kj} \quad A_{kj} = \sum_{il} B_{ikl} C_{lj} D_{ij} \\
A_{lj} = \sum_{ik} B_{ikl} C_{ij} D_{kj} \quad A_{ij} = \sum_{ik} B_{ikl} C_{kj} \\
A_{ik} = \sum_{l} B_{ikl} C_{lj} \quad A_{ijl} = \sum_{k} B_{ikl} C_{kj} \\
A_{jk} = \sum_{i} B_{ijk} c_i \quad A_{ijk} = \sum_{j} B_{ijk} c_j
\]

\[
C = \sum_{ijkl} M_{ij} P_{jk} M_{kl} P_{li} \\
a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} M_{nm} P_{no} M_{po} P_{pi}
\]

Quantum Chromodynamics
Too many tensor kernels for a fixed-function library

\[
\begin{align*}
\text{Eigen (SpMV)} \\
\quad a &= Bc + a \\
\quad a &= Bc + b 
A &= B + C 
\quad a &= \alpha Bc + \beta a \\
\quad a &= B^Tc 
A &= \alpha B 
\quad a &= B(c + d) \\
\quad a &= B^Tc + d 
A &= B + C + D 
\quad A &= BC \\
A &= B \odot C 
\quad a &= b \odot c 
A &= 0 
\quad A &= B \odot (CD) \\
\quad A &= BCd 
A &= B^T 
\quad a &= B^TBc \\
\quad a &= b + c 
A &= B 
\quad K &= A^TCA \\
\quad A_{ij} &= \sum_{kl} B_{ikl} C_{lj} D_{kj} 
A_{kj} &= \sum_{il} B_{ikl} C_{lj} D_{ij} \\
A_{lj} &= \sum_{ik} B_{ikl} C_{ij} D_{kj} 
A_{ij} &= \sum_{ik} B_{ikl} c_k \\
A_{ijk} &= \sum_{l} B_{ikl} C_{lj} 
A_{ik} &= \sum_{l} B_{ikl} c_j \\
A_{jk} &= \sum_{i} B_{ijk} c_i 
A_{ijl} &= \sum_{k} B_{ikl} C_{kj} \\
C &= \sum_{ijkl} M_{ij} P_{jk} M_{lk} P_{il} 
\quad \tau &= \sum_{ij} z_i (\sum_{j} z_j \theta_{ij}) (\sum_{k} z_k \theta_{ik}) \\
\quad a &= \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} M_{lm} M_{nm} P_{no} M_{po} P_{ip} 
\end{align*}
\]
Too many tensor kernels for a fixed-function library

CSparse

\[ a = Bc + a \]

Eigen (SpMV)

\[ a = Bc \]

OSKI

\[ a = Bc + b \]

\[ A = B + C \]

\[ a = \alpha Bc + \beta a \]

PETSc

\[ a = B^Tc \]

\[ A = \alpha B \]

\[ a = B(c + d) \]

\[ A = B \odot C \]

\[ a = b \odot c A = 0 \]

\[ A = BC \]

\[ A = BCD \]

\[ A = B^T \]

\[ a = B^T Bc \]

\[ a = b + c \]

\[ A = B \]

\[ K = A^TCA \]

\[ A_{ij} = \sum_{kl} B_{ikl} C_{lj} D_{kj} \]

\[ A_{kj} = \sum_{il} B_{ikl} C_{lj} D_{ij} \]

\[ A_{lj} = \sum_{ik} B_{ikl} C_{ij} D_{kj} \]

\[ A_{ij} = \sum_{k} B_{ijk} c_k \]

\[ A_{ijk} = \sum_{l} B_{ikl} C_{lj} \]

\[ A_{ik} = \sum_{l} B_{ilk} c_l \]

\[ A_{ijl} = \sum_{k} B_{ikl} C_{kj} \]

\[ A_{jkl} = \sum_{i} B_{ijk} c_i \]

\[ C = \sum_{ijkl} M_{ij} P_{jk} M_{lk} P_{it} \]

\[ a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} M_{nm} P_{no} M_{po} P_{ip} \]
Too many tensor kernels for a fixed-function library

\[
\begin{align*}
C_{	ext{Sparse}} & : \quad a = Bc + a \\
\quad A = B + C & : \quad a = \alpha Bc + \beta a \\
\quad a = Bc + b & : \quad A = B + C \\
\quad a = Bc + c & : \quad A = B + C + D \\
\quad A = B \odot C & : \quad a = b \odot c \quad A = 0 \\
\quad A = BCD & : \quad A = B^T Bc \\
\quad A = B + c & : \quad A = B \quad K = A^T C A \\
\quad A_{ij} = \sum_{k,l} B_{ikl} C_{lj} D_{kj} & : \quad a_{ij} = \sum_{il} B_{ikl} C_{lj} D_{ij} \\
\quad A_{ij} = \sum_{ik} B_{ikl} C_{ij} D_{kj} & : \quad a_{ij} = \sum_{ik} B_{ijk} c_k \\
\quad A_{ijk} = \sum_{l} B_{ikl} C_{lj} & : \quad A_{ik} = \sum_{j} B_{ijk} c_j \\
\quad A_{ikl} = \sum_{i} B_{ijkl} c_i & : \quad A_{ijk} = \sum_{k} B_{ijkl} c_k \\
\quad C = \sum_{ijkl} M_{ij} P_{jk} M_{lk} P_{it} & : \quad a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{im} M_{nm} P_{no} M_{po} P_{ip} \\
\quad \tau = \sum_{ij} z_i \left( \sum_{j} z_j \theta_{ij} \right) \left( \sum_{k} z_k \theta_{ik} \right) \\
\end{align*}
\]
Too many tensor kernels for a fixed-function library

\[ a = Bc + a \]

\[ a = Bc + b \]
\[ A = B + C \]
\[ a = \alpha Bc + \beta a \]
\[ a = B^T c \]
\[ A = \alpha B \]
\[ a = B(c + d) \]
\[ a = B^T c + d \]
\[ A = B + C + D \]
\[ A = BC \]
\[ A = B \odot C \]
\[ a = b \odot c A = 0 \]
\[ A = B \odot (CD) \]
\[ A = BCd \]
\[ A = B^T \]
\[ a = B^T Bc \]
\[ a = b + c \]
\[ A = B \]
\[ K = A^T CA \]

\[ A_{ij} = \sum_{kl} B_{ikl} C_{lj} D_{kj} \]
\[ A_{kj} = \sum_{il} B_{ikl} C_{lj} D_{ij} \]
\[ A_{lj} = \sum_{ik} B_{ikl} C_{ij} D_{kj} \]
\[ A_{ij} = \sum_{ik} B_{ikl} C_{lk} \]
\[ A_{ij} = \sum_{ik} B_{ikl} C_{lj} \]

\[ A_{ijk} = \sum_{l} B_{ikl} C_{lj} \]
\[ A_{ik} = \sum_{l} B_{ikl} C_{lj} \]
\[ A_{ij} = \sum_{k} B_{ikl} C_{kj} \]
\[ A_{ij} = \sum_{k} B_{ikl} C_{lk} \]

\[ C = \sum_{ijkl} M_{ij} P_{jk} M_{lk} P_{il} \]
\[ \tau = \sum_{i} z_i \left( \sum_{j} z_j \theta_{ij} \right) \left( \sum_{k} z_k \theta_{ik} \right) \]

\[ a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} M_{nm} P_{no} M_{po} P_{ip} \]
Too many tensor kernels for a fixed-function library

\[
\begin{align*}
  a &= Bc + a \\  a &= Bc + b \\  A &= B + C \\  A &= \alpha Bc + \beta a \\  a &= B^T c \\  A &= \alpha B \\  a &= B(c + d) \\  a &= B^T c + d \\  A &= B + C + D \\  A &= BC \\  a &= b \odot c \\  a &= 0 \\  A &= B \odot (CD) \\  A &= BCD \\  A &= B^T \\  a &= B^T Bc \\  a &= b + c \\  A &= B \\  K &= A^T CA \\
  A_{ij} &= \sum_{kl} B_{ikl} C_{lj} D_{kj} \\  A_{kj} &= \sum_{il} B_{ikl} C_{lj} D_{ij} \\  A_{lj} &= \sum_{ik} B_{ikl} C_{lj} D_{kj} \\  A_{ij} &= \sum_{lk} B_{ijl} c_k \\  A_{iik} &= \sum_{j} B_{ijk} c_j \\  A_{ijk} &= \sum_{i} B_{ijk} c_i \\  A_{ik} &= \sum_{j} B_{ijk} c_j \\  A_{ikl} &= \sum_{j} B_{ijk} c_j \\  C &= \sum_{ijkl} M_{ij} P_{jk} M_{kl} P_{il} \\  a &= \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} M_{nm} P_{no} M_{po} P_{ip} \\
\end{align*}
\]

Dense Matrix

- CSR
- DCSR
- BCSR
- Thermal Simulation
- COO
- ELLPACK
- CSB
- Blocked COO
- CSC
- DIA
- Blocked DIA
- DCSC

\[\times\]
Too many tensor kernels for a fixed-function library

\[ a = Bc + a \quad a = Bc \]
\[ a = Bc + b \quad A = B + C \quad a = \alpha Bc + \beta a \]
\[ a = B^T c \quad A = \alpha B \quad a = B(c + d) \]
\[ a = B^T c + d \quad A = B + C + D \quad A = BC \]
\[ A = B \otimes C \quad a = b \otimes c \quad A = 0 \quad A = B \otimes (CD) \]
\[ A = BCd \quad A = B^T \quad a = B^T Bc \]
\[ a = b + c \quad A = B \quad K = A^T C A \]

\[ A_{ij} = \sum_{kl} B_{ikl}C_{lj}D_{kj} \quad A_{kj} = \sum_{il} B_{ikl}C_{lj}D_{ij} \]
\[ A_{lj} = \sum_{ik} B_{ikl}C_{ij}D_{kj} \quad A_{ij} = \sum_{lk} B_{ijk}c_k \]
\[ A_{ijk} = \sum_{l} B_{ikl}C_{lj} \quad A_{ik} = \sum_{k} B_{ijk}c_j \]
\[ A_{jk} = \sum_{i} B_{ijk}c_i \quad A_{ijl} = \sum_{k} B_{ikl}C_{kj} \]

\[ C = \sum_{ijkl} M_{ij}P_{jk}M_{lk}P_{il} \]
\[ C = \sum_{ijklmnop} M_{ij}P_{jk}M_{kl}P_{lm}M_{nm}P_{no}M_{po}P_{ip} \]

<table>
<thead>
<tr>
<th>Dense Matrix</th>
<th>Web matrix [BG 2008]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSR DCSR BCSR COO ELLPACK CSB Blocked COO CSC DIA Blocked DIA DCSC</td>
<td></td>
</tr>
</tbody>
</table>
Too many tensor kernels for a fixed-function library

\[
a = Bc + a
\]

\[
a = Bc + b
\]

\[
A = B + c
\]

\[
a = \alpha B c + \beta a
\]

\[
A = B^T c
\]

\[
a = B(c + d)
\]

\[
A = B \oplus C
\]

\[
a = \alpha B
\]

\[
A = B \oplus (CD)
\]

\[
A = BC
\]

\[
A = BC d
\]

\[
a = B^T Bc
\]

\[
a = b + c
\]

\[
A = B
\]

\[
K = A^T C A
\]

\[
A_{ij} = \sum_{kl} B_{ikl} C_{lj} D_{kj}
\]

\[
A_{kj} = \sum_{il} B_{ikl} C_{lj} D_{ij}
\]

\[
A_{lj} = \sum_{ik} B_{ikl} C_{lj} D_{kj}
\]

\[
A_{ij} = \sum_{k} B_{ijk} c_k
\]

\[
A_{ik} = \sum_{l} B_{ikl} C_{lj}
\]

\[
A_{i} = \sum_{k} B_{ijk} c_j
\]

\[
A_{jk} = \sum_{i} B_{ijk} c_i
\]

\[
A_{ijl} = \sum_{k} B_{ikl} C_{kj}
\]

\[
C = \sum_{ijkl} M_{ij} P_{jk} M_{lk} P_{il}
\]

\[
a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} M_{nm} P_{no} M_{po} P_{ip}
\]

\[
\tau = \sum_{i} z_{i} (\sum_{j} z_{j} \theta_{ij}) (\sum_{k} z_{k} \theta_{ik})
\]

Dense Matrix

CSR  DCSR  BCSR

Finite Elements Method, Block-Sparse NN Weights [GRK 2017]

COO  ELLPACK  CSB

Blocked COO  CSC

DIA  Blocked DIA  DCSC
Too many tensor kernels for a fixed-function library

\[
a = Bc + a \\
a = Bc + b \\
a = B^Tc + d \\
A = B + C \\
a = \alpha Bc + \beta a \\
a = \alpha B \\
a = B(c + d) \\
A = B + C + D \\
A = BC \\
a = b \odot c \\
A = B \odot (CD) \\
A = BCD \\
A = B^T \\
a = B^TBc \\
A = B + c \\
A = B \\
K = A^TCA
\]

\[
A_{ij} = \sum_{kl} B_{ijkl}C_{lj}D_{kj} \quad A_{kj} = \sum_{il} B_{ijkl}C_{lj}D_{ij} \\
A_{lj} = \sum_{ik} B_{ijkl}C_{ij}D_{kj} \quad A_{ij} = \sum_{ik} B_{ijk}c_k \\
A_{ijk} = \sum_{l} B_{ijkl}C_{lj} \quad A_{ik} = \sum_{k} B_{ijk}c_j \\
A_{jk} = \sum_{i} B_{ijk}c_i \quad A_{ijl} = \sum_{k} B_{ikl}C_{kj} \\
\tau = \sum_{i} z_i \left( \sum_{j} z_j \theta_{ij} \right) \left( \sum_{k} z_k \theta_{ik} \right) \\
C = \sum_{ijkl} M_{ij}P_{jk}M_{kl}P_{il} \\
a = \sum_{ijklmnop} M_{ij}P_{jk}M_{kl}P_{lm}M_{nm}P_{no}M_{po}P_{ip}
\]

Dense Matrix

CSR DCSR BCSR COO ELLPACK CSB Data Analytics

Blocked COO CSC DIA Blocked DIA DCSC
Too many tensor kernels for a fixed-function library

\[a = Bc + a\]  
\[a = Bc + b\]  
\[A = B + C\]  
\[a = \alpha Bc + \beta a\]  
\[a = B^T c\]  
\[A = \alpha B\]  
\[a = B(c + d)\]  
\[a = B^T c + d\]  
\[A = B + C + D\]  
\[A = BC\]  
\[A = B \odot C\]  
\[a = b \odot c A = 0\]  
\[A = B \odot (CD)\]  
\[A = BCd\]  
\[A = B^T\]  
\[a = B^T Bc\]  
\[a = b + c\]  
\[A = B\]  
\[K = A^T CA\]  
\[A_{ij} = \sum_{kl} B_{ikl}C_{lj}D_{kj}\]  
\[A_{kj} = \sum_{il} B_{ikl}C_{lj}D_{ij}\]  
\[A_{lji} = \sum_{ik} B_{ikl}C_{ij}D_{kj}\]  
\[A_{ij} = \sum_{k} B_{ijk}c_k\]  
\[A_{ijk} = \sum_{l} B_{ikl}C_{lj}\]  
\[A_{ik} = \sum_{k} B_{ijk}c_j\]  
\[A_{jki} = \sum_{l} B_{ijl}c_i\]  
\[A_{ijl} = \sum_{k} B_{ikl}C_{kj}\]  
\[A_{ijk} = \sum_{l} B_{ikl}C_{lj}\]  
\[A_{ij} = \sum_{k} B_{ijk}c_k\]  
\[A_{ijkl} = \sum_{k} B_{ikl}C_{kj}\]  
\[\tau = \sum_{i} z_i (\sum_{j} z_j \theta_{ij}) (\sum_{k} z_k \theta_{ik})\]  
\[C = \sum_{ijkl} M_{ij}P_{jk}M_{lk}P_{il}\]  
\[a = \sum_{ijklmnop} M_{ij}P_{jk}M_{kl}P_{lm}M_{nm}P_{no}M_{po}P_{ip}\]  

Dense Matrix

CSR  DCSR  BCSR

COO  ELLPACK  CSB  Mesh Simulations on GPUs [BG 2009]

Blocked COO  CSC

DIA  Blocked DIA  DCSC

3
Too many tensor kernels for a fixed-function library

\[ a = Bc + a \quad a = Bc \]

\[ a = Bc + b \quad A = B + C \quad a = \alpha Bc + \beta a \]

\[ a = B^Tc \quad A = \alpha B \quad a = B(c + d) \]

\[ a = B^Tc + d \quad A = B + C + D \quad A = BC \]

\[ A = B \odot C \quad a = b \odot c \quad A = 0 \quad A = B \odot (CD) \]

\[ A = BCd \quad A = B^T \quad a = B^T Bc \]

\[ a = b + c \quad A = B \quad K = A^T C A \]

\[ A_{ij} = \sum_{kl} B_{ijkl} C_{lj} D_{kj} \quad A_{kj} = \sum_{il} B_{ijkl} C_{lj} D_{ij} \]

\[ A_{lj} = \sum_{ik} B_{iklj} C_{ij} D_{kj} \quad A_{ij} = \sum_{k} B_{ijkc_k} \]

\[ A_{iij} = \sum_{l} B_{lijk} C_{lj} \quad A_{ik} = \sum_{k} B_{ijkc_k} \]

\[ A_{jkl} = \sum_{i} B_{lijk} C_{kj} \quad A_{ijkl} = \sum_{k} B_{ijkl} C_{kj} \]

\[ C = \sum_{ijkl} M_{ij} P_{jk} M_{lk} P_{il} \]

\[ \tau = \sum_{i} z_i \left( \sum_{j} z_j \theta_{ij} \right) \left( \sum_{k} z_k \theta_{ik} \right) \]

\[ a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} M_{nm} P_{no} M_{po} P_{ip} \]
Too many tensor kernels for a fixed-function library

\[
\begin{align*}
    a &= Bc + a \\
    a &= Bc + b & A &= B + C & a &= \alpha Bc + \beta a \\
    a &= B^Tc & A &= \alpha B & a &= B(c + d) \\
    a &= B^Tc + d & A &= B + C + D & A &= BC \\
    A &= B \odot C & a &= b \odot c & A &= 0 & A &= B \odot (CD) \\
    A &= BCd & A &= B^T & a &= B^T Bc \\
    a &= b + c & A &= B & K &= A^T CA
\end{align*}
\]

\[
\begin{align*}
    A_{ij} &= \sum_{kl} B_{ikl} C_{lj} D_{kj} & A_{kj} &= \sum_{il} B_{ikl} C_{lj} D_{ij} \\
    A_{lj} &= \sum_{ik} B_{ikl} C_{ij} D_{kj} & A_{ij} &= \sum_{k} B_{ijk} c_k \\
    A_{ijk} &= \sum_{l} B_{ikl} C_{lj} & A_{ik} &= \sum_{j} B_{ijk} c_j \\
    A_{jk} &= \sum_{i} B_{ijk} c_i & A_{ijl} &= \sum_{k} B_{ikl} C_{kj} \\
    C &= \sum_{ijkl} M_{ij} P_{jk} M_{kl} P_{li} \\
    a &= \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} M_{nm} P_{no} M_{po} P_{ip}
\end{align*}
\]
Too many tensor kernels for a fixed-function library

\[ a = Bc + a \quad a = Bc \]
\[ a = Bc + b \quad A = B + C \quad a = \alpha Bc + \beta a \]
\[ a = B^Tc \quad A = \alpha B \quad a = B(c + d) \]
\[ a = B^Tc + d \quad A = B + C + D \quad A = BC \]
\[ A = B \odot C \quad a = b \odot c A = 0 \quad A = B \odot (CD) \]
\[ A = BCd \quad A = B^T \quad a = B^TBc \]
\[ a = b + c \quad A = B \quad K = A^TCA \]
\[ A_{ij} = \sum_{kl} B_{ikl} C_{lj} D_{kj} \quad A_{kj} = \sum_{il} B_{ikl} C_{lj} D_{ij} \]
\[ A_{lj} = \sum_{ik} B_{ikl} C_{ij} D_{kj} \quad A_{ij} = \sum_{k} B_{ijk} c_k \]
\[ A_{ijk} = \sum_{l} B_{ikl} C_{lj} \quad A_{ik} = \sum_{l} B_{ikl} c_j \]
\[ A_{jk} = \sum_{i} B_{ijk} c_i \quad A_{ijl} = \sum_{k} B_{ikl} C_{kj} \]
\[ C = \sum_{ijkl} M_{ij} P_{jk} M_{lk} P_{il} \]
\[ a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} M_{nm} P_{no} M_{po} P_{ip} \]
Too many tensor kernels for a fixed-function library

\[ a = Bc + a \quad a = Bc \]

\[ a = Bc + b \quad A = B + C \quad a = \alpha Bc + \beta a \]

\[ a = B^Tc \quad A = \alpha B \quad a = B(c + d) \]

\[ a = B^Tc + d \quad A = B + C + D \quad a = BC \]

\[ A = B \odot C \quad a = b \odot c A = 0 \quad A = B \odot (CD) \]

\[ A = BCd \quad A = B^T \quad a = B^T Bc \]

\[ a = b + c \quad A = B \quad K = A^T CA \]

\[ A_{ij} = \sum_{kl} B_{ikl} C_{lj} D_{kj} \quad A_{kj} = \sum_{il} B_{ikl} C_{lj} D_{ij} \]

\[ A_{lj} = \sum_{ik} B_{ikl} C_{ij} D_{kj} \quad A_{ij} = \sum_k B_{ijk} c_k \]

\[ A_{ijk} = \sum_l B_{ikl} C_{ij} \quad A_{ik} = \sum_j B_{ijk} c_j \]

\[ A_{jk} = \sum_i B_{ijk} c_i \quad A_{ijl} = \sum_k B_{ikl} C_{kj} \]

\[ C = \sum_{ijkl} M_{ij} P_{jk} M_{kl} P_{il} \]

\[ a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} M_{nm} P_{no} M_{po} P_{ip} \]

Dense Matrix
CSR DCSR BCSR
COO ELLPACK CSB
Blocked COO CSC
DIA Blocked DIA DCSC
Sparse vector Hash Maps
Coordinates
CSF Dense Tensors
Block Tensors
Too many tensor kernels for a fixed-function library

\[ a = Bc + a \quad a = Bc \]
\[ a = Bc + b \quad A = B + C \quad a = \alpha Bc + \beta a \]
\[ a = B^T c \quad A = \alpha B \quad a = B(c + d) \]
\[ a = B^T c + d \quad A = B + C + D \quad A = BC \]
\[ A = B \odot C \quad a = b \odot c A = 0 \quad A = B \odot (CD) \]
\[ A = BCd \quad A = B^T \quad a = B^T Bc \]
\[ a = b + c \quad A = B \quad K = A^T CA \]

\[ A_{ij} = \sum_{kl} B_{ikl} C_{lj} D_{kj} \quad A_{kj} = \sum_{il} B_{ikl} C_{lj} D_{ij} \]
\[ A_{lj} = \sum_{ik} B_{ikl} C_{ij} D_{kj} \quad A_{ij} = \sum_{i k} B_{i j k} c_k \]
\[ A_{ijk} = \sum_{i l} B_{i k l} C_{lj} \quad A_{ik} = \sum_{i k} B_{i j k} c_j \]
\[ A_{j k} = \sum_{i k} B_{i j k} c_i \quad A_{ijkl} = \sum_{k} B_{ikl} C_{kj} \]

\[ C = \sum_{ijkl} M_{ij} P_{jk} \overline{M_{ikl}} P_{il} \]
\[ a = \sum_{ijklmnop} M_{ij} P_{jk} M_{kl} P_{lm} \overline{M_{nm}} P_{n o} \overline{M_{po}} P_{ip} \]

Dense Matrix
- CSR
- DCSR
- BCSR

COO
- ELLPACK
- CSB

Blocked COO
- CSC

DIA
- Blocked DIA
- DCSC

Sparse vector
- Hash Maps

Coordinates
- CSF
- Dense Tensors

Blocked Tensors

Linked Lists
- Database

Compression Schemes
- Cloud Storage
Too many tensor kernels for a fixed-function library

\[ a = Bc + a \]
\[ a = Bc + b \]
\[ A = B + C \]
\[ a = \alpha Bc + \beta a \]
\[ a = B^Tc \]
\[ A = \alpha B \]
\[ a = B(c + d) \]
\[ a = B^Tc + d \]
\[ A = B + C + D \]
\[ A = BC \]
\[ A = B \odot C \]
\[ a = b \odot c A = 0 \]
\[ A = B \odot (CD) \]
\[ A = BCd \]
\[ A = B^T \]
\[ a = B^TBc \]

\[ a = b + c \]
\[ A = B \]
\[ K = A^TCA \]

\[ A_{ij} = \sum_{kl} B_{ikl}C_{lj}D_{kj} \]
\[ A_{kj} = \sum_{il} B_{ikl}C_{lj}D_{ij} \]
\[ A_{ij} = \sum_{k} B_{ijk}c_k \]
\[ A_{kj} = \sum_{l} B_{ikl}C_{lj} \]
\[ A_{ik} = \sum_{j} B_{ijk}c_j \]
\[ A_{ij} = \sum_{k} B_{ikl}C_{kj} \]

\[ C = \sum_{ijkl} M_{ij}P_{jk}M_{lk}P_{it} \]
\[ a = \sum_{ijklmnop} M_{ij}P_{jk}M_{kl}P_{lm}M_{nm}P_{no}M_{po}P_{ip} \]

---

Dense Matrix

CSR  DCSR  BCSR

COO  ELLPACK  CSB

Blocked COO  CSC

DIA  Blocked DIA  DCSC

Sparse vector  Hash Maps

Coordinates

CSF  Dense Tensors

Blocked Tensors

Linked Lists  Database

Compression Schemes

Cloud Storage

---

CPU

GPUs  TPUs

FPGA

Sparse Tensor Hardware

Cloud Computers

Supercomputers
Logistics

Sessions

Sparse Algorithms and Data Structures I
Sparse Algorithms and Data Structures II
Sparse Polyhedral Optimization
Tensor Compilers I
Tensor Compilers II
Tensors and Graphs
Hardware for Sparse Tensor Algebra
Applications in Science and Engineering

Session Format

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 min</td>
<td>Speaker</td>
</tr>
<tr>
<td>10 min</td>
<td>Speaker</td>
</tr>
<tr>
<td>10 min</td>
<td>Speaker</td>
</tr>
<tr>
<td>20 min</td>
<td>Discussion</td>
</tr>
<tr>
<td>10/15 min</td>
<td>Break</td>
</tr>
</tbody>
</table>
Dinner and Posters

Dinner (Sri Lankan)

Posters

Eddie Davis
Suzy Mueller
Mahdi Mohammadi
Ryan Senanayake
Srinivas Eswar
Stephen Chou
Peter Ahrens
Patricio Noyola
Matt Fishman
Yunming Zhang