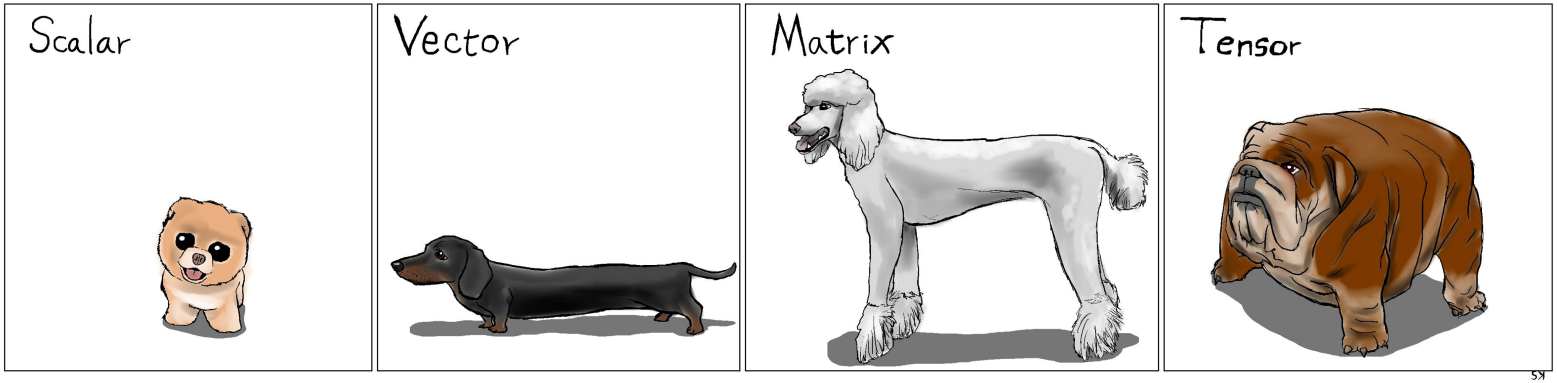
# Role of Tensors in Machine Learning

**Anima Anandkumar**

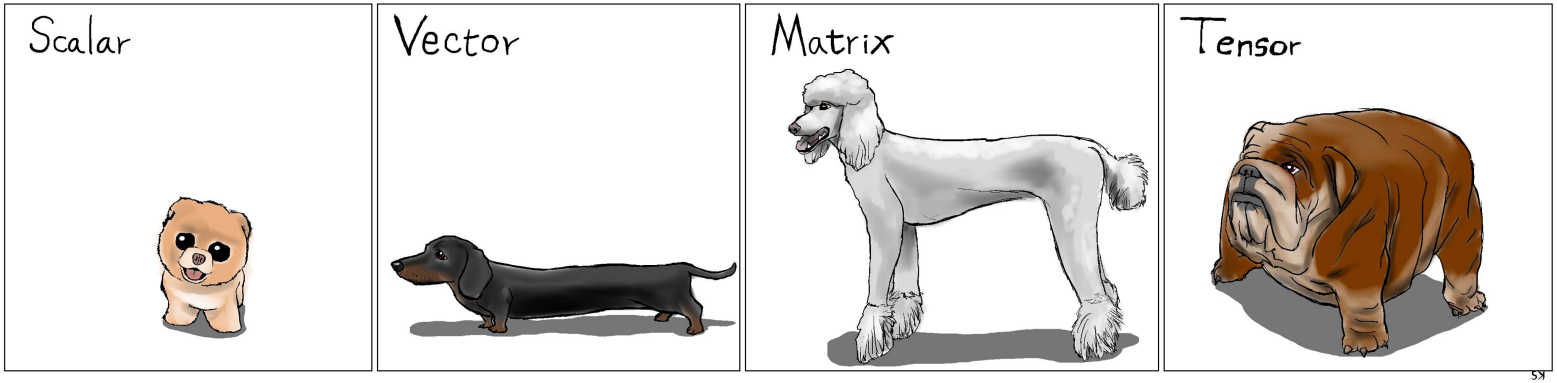
Amazon AI & Caltech

# Tensors: Beyond 2D world

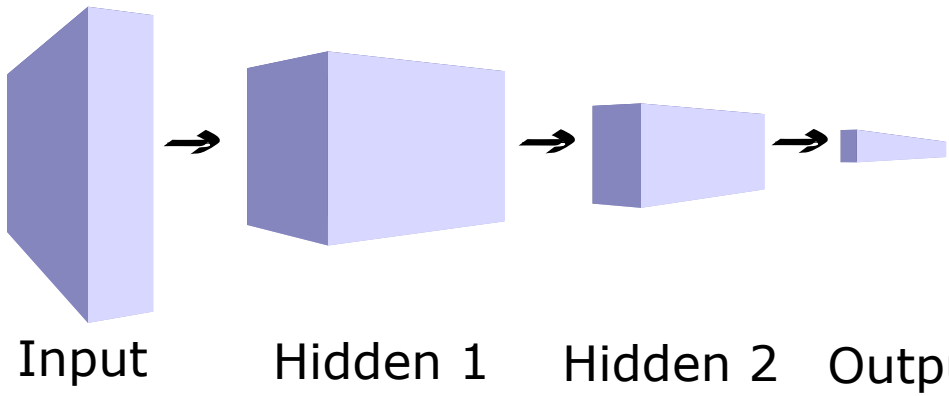



**Modern data is inherently multi-dimensional**

# Tensors: Beyond 2D world




**Modern data is inherently multi-dimensional**

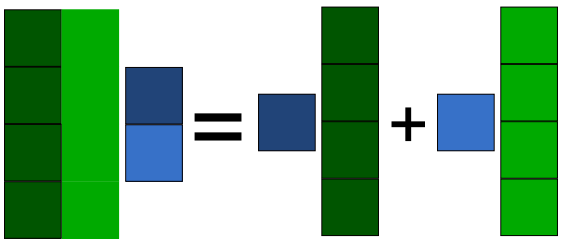


# Tensor Contraction

Extends the notion of matrix product

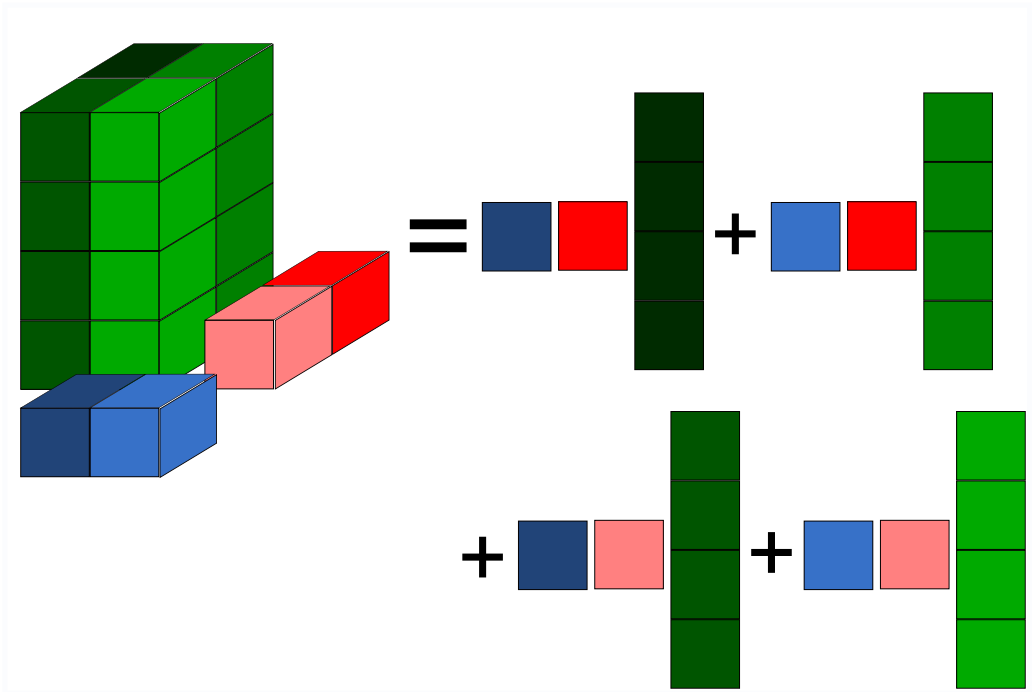
## Matrix product

$$Mv = \sum_j v_j M_j$$



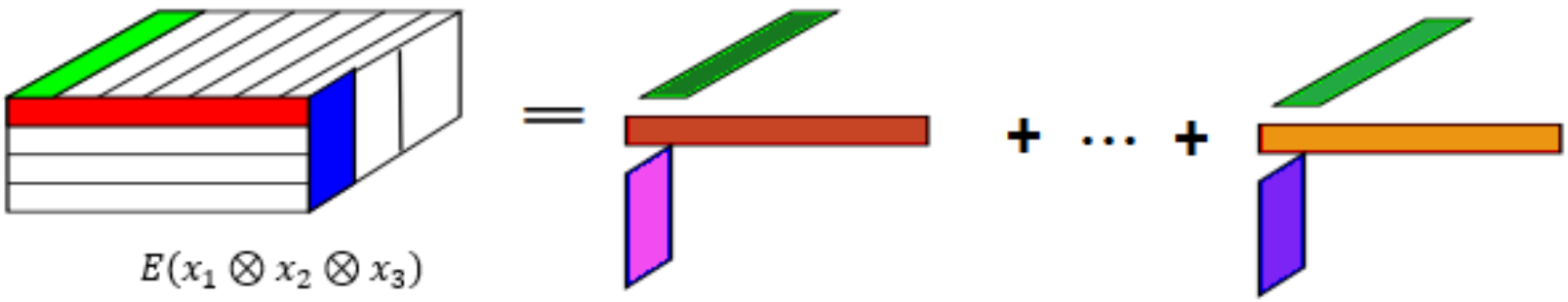
## Tensor Contraction

$$T(u, v, \cdot) = \sum_{i,j} u_i v_j T_{i,j,:}$$



# Tensor Decompositions

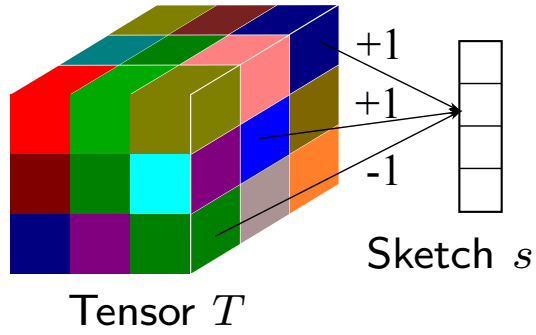
$E(x_1 \otimes x_2)$



$E(x_1 \otimes x_2 \otimes x_3)$

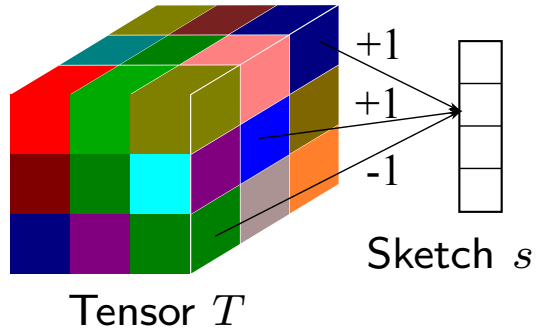
# Tensor Sketches

- Dimensionality reduction through sketching.
  - Complexity independent of tensor order: exponential gain!



# Tensor Sketches

- Randomized dimensionality reduction through sketching.
  - Complexity independent of tensor order: exponential gain!

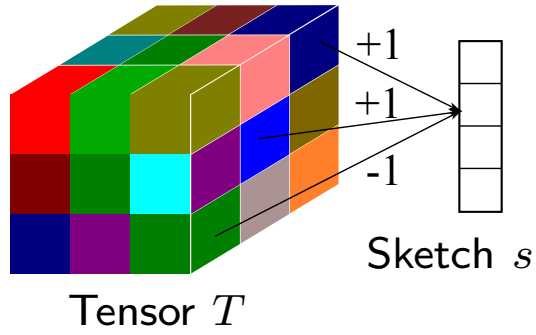



## Applications

- Tensor Decomposition via Sketching by Wang, Tung, Smola, A. NIPS'15.
- Compact Tensor Pooling for Visual Question and Answering by Shi, Anubhai, Furlanello, A, CVPR 2017 VQA workshop.

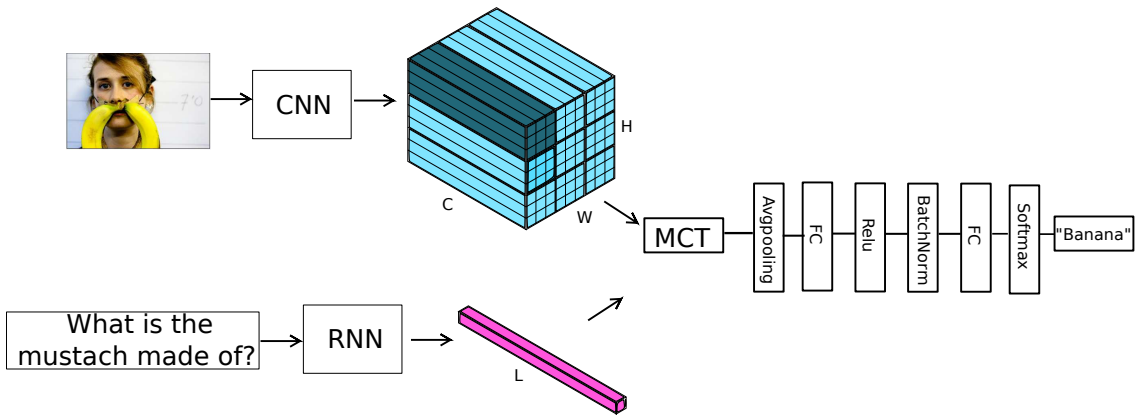
# Tensor Sketches

- Randomized dimensionality reduction through sketching.
  - Complexity independent of tensor order: exponential gain!




## Applications

- Tensor Decomposition via Sketching by Wang, Tung, Smola, A. NIPS'15.
- Compact Tensor Pooling for Visual Question and Answering by Shi, Anubhai, Furlanello, A, CVPR 2017 VQA workshop.



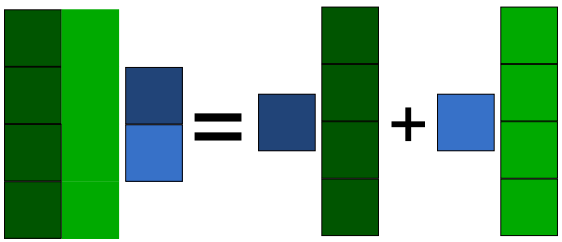
# Outline

# Tensor Contraction

Extends the notion of matrix product

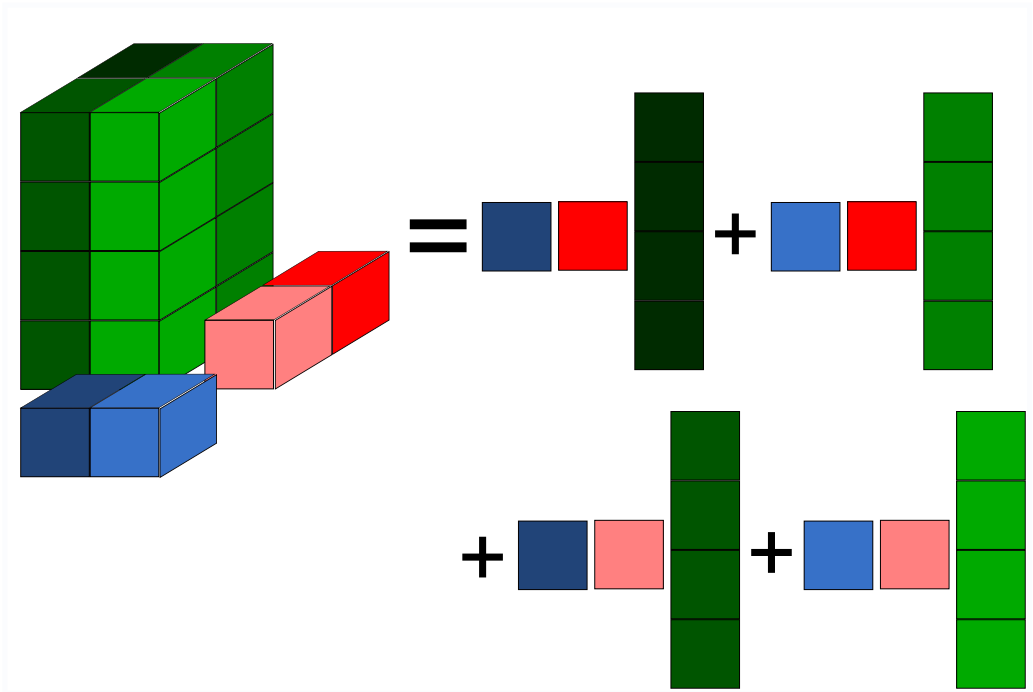
## Matrix product

$$Mv = \sum_j v_j M_j$$

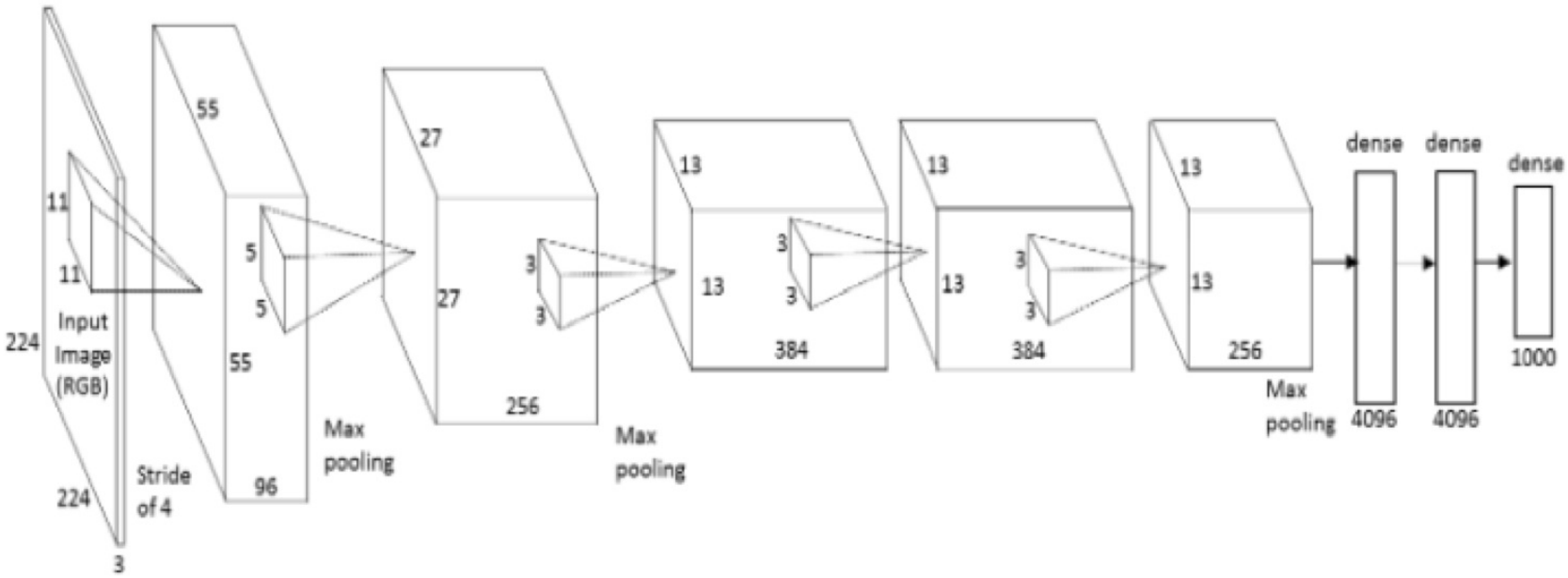


## Tensor Contraction

$$T(u, v, \cdot) = \sum_{i,j} u_i v_j T_{i,j,:}$$

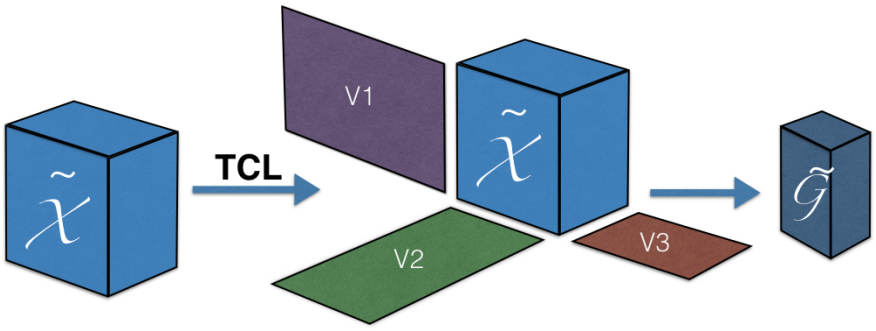


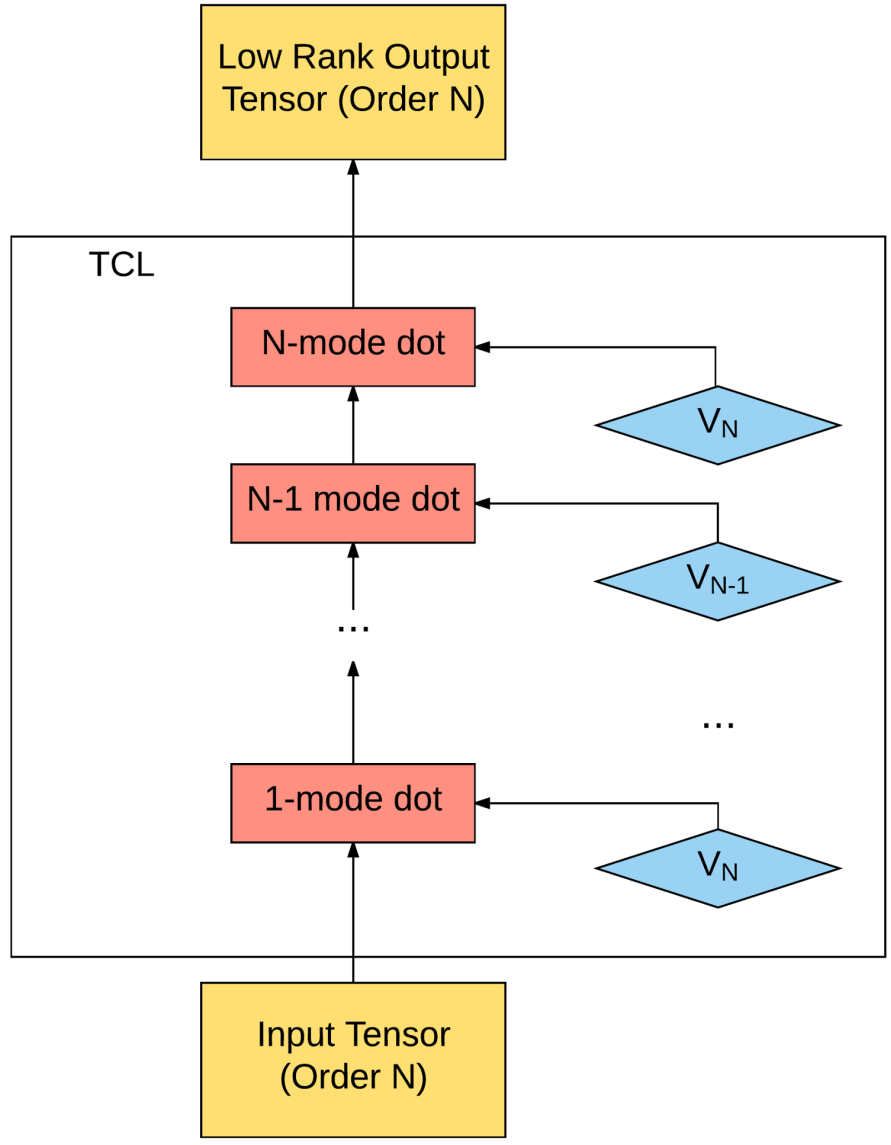
# Employing Tensor Contractions in Alexnet




Replace fully connected layer with tensor contraction layer

# Enabling Tensor Contraction Layer in Mxnet

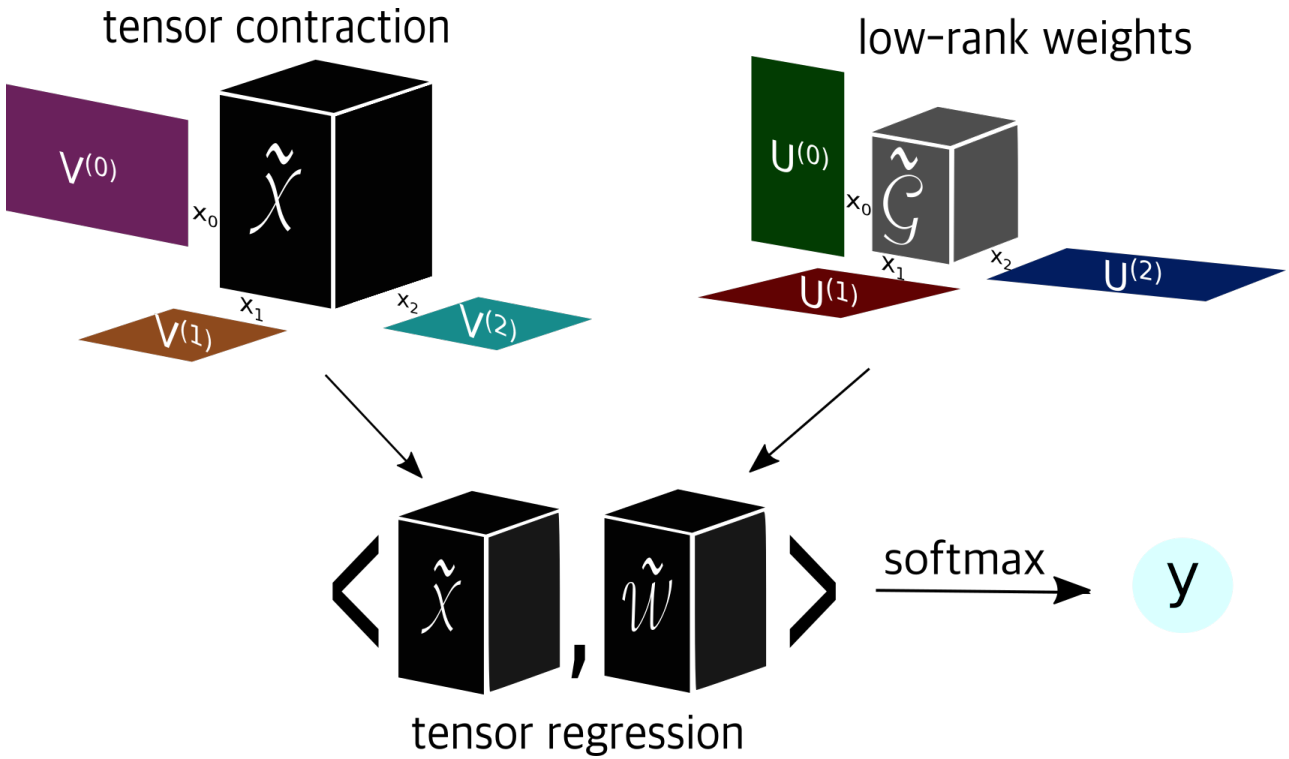


# Performance of the TCL

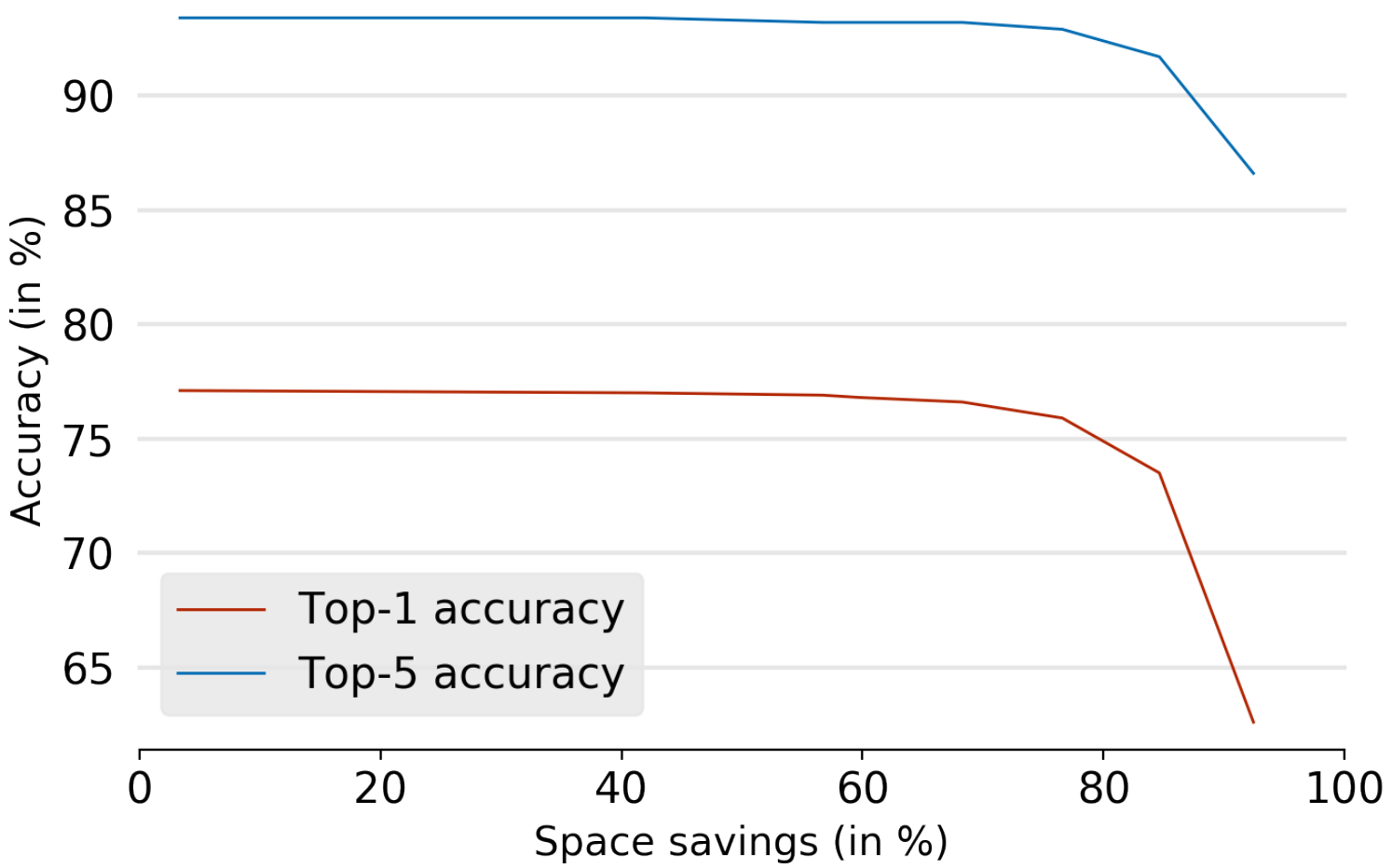
- Trained end-to-end
- On ImageNet with VGG:
  - 65.9% space savings
  - performance drop of 0.6% only
- On ImageNet with AlexNet:
  - 56.6% space savings
  - Performance improvement of 0.5%

# Low-rank tensor regression




**Tensor Regression Networks**, *J. Kossaifi, Z.C.Lipton, A.Khanna, T.Furlanello and A.Anandkumar*, ArXiv pre-publication

# Performance and rank



# Outline

# Speeding up Tensor Contractions

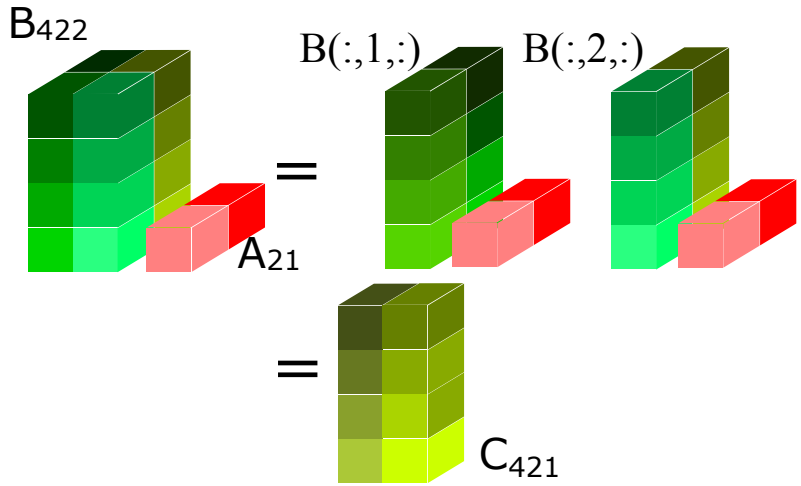
1. Tensor contractions are a core primitive of multilinear algebra.
2. BLAS 3: Unbounded compute intensity (no. of ops per I/O)

Consider single-index contractions: $C_{\mathcal{C}} = A_{\mathcal{A}}\, B_{\mathcal{B}}$

# Speeding up Tensor Contractions

1. Tensor contractions are a core primitive of multilinear algebra.
2. BLAS 3: Unbounded compute intensity (no. of ops per I/O)

Consider single-index contractions: $C_{\mathcal{C}} = A_{\mathcal{A}}\, B_{\mathcal{B}}$



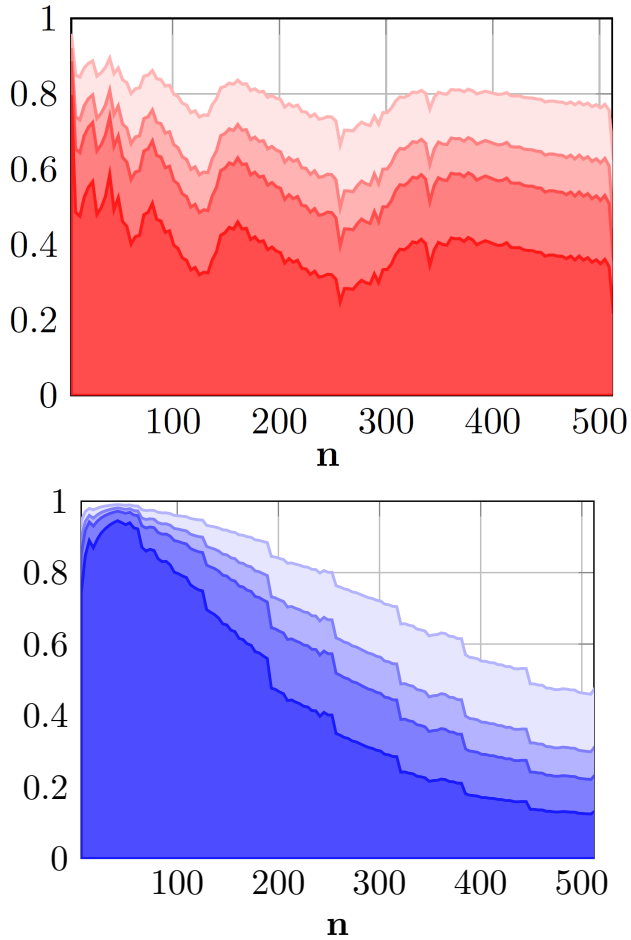

e.g. $C_{mnp} = \mathrm{B}_{mnk}\, \mathrm{A}_{kp}$

# Speeding up Tensor Contraction

Explicit permutation dominates, especially for small tensors.

Consider $C_{mnp} = A_{km}\, B_{pkn}$.

1. $A_{km} \rightarrow A_{mk}$
2. $B_{pkn} \rightarrow B_{kpn}$
3. $C_{mnp} \rightarrow C_{mpn}$
4. $\boxed{C_{m(pn)} = A_{mk}\, B_{k(pn)}}$
5. $C_{mpn} \rightarrow C_{mnp}$




(Top) CPU. (Bottom) GPU. The fraction of time spent in copies/transpositions. Lines are shown with 1, 2, 3, and 6 transpositions.

# Existing Primitives

## GEMM

- Suboptimal for many small matrices.

## Pointer-to-Pointer BatchedGEMM

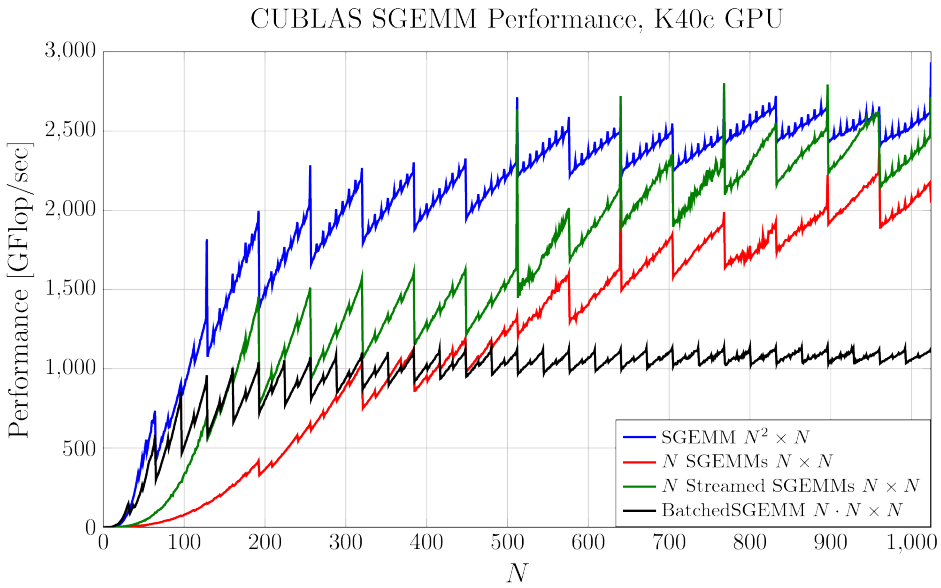
- Available in MKL 11.3$\beta$ and cuBLAS 4.1

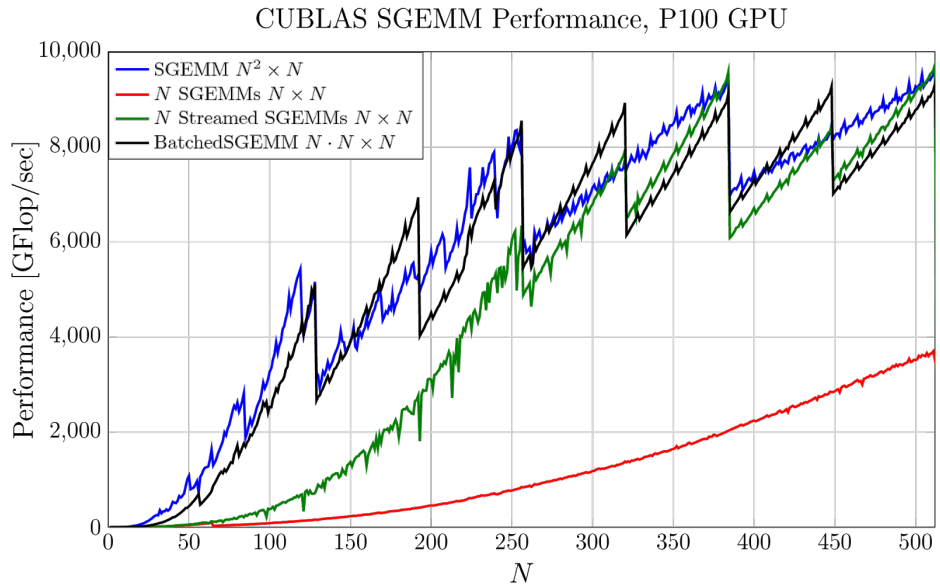
$$C[p] = \alpha\,\mathrm{op}(A[p])\,\mathrm{op}(B[p]) + \beta\,C[p]$$

```
cublas<T>gemmBatched(cublasHandle_t handle,
                     cublasOperation_t transA, cublasOperation_t transB,
                     int M, int N, int K,
                     const T* alpha,
                     const T** A, int ldA,
                     const T** B, int ldB,
                     const T* beta,
                     T** C, int ldC,
                     int batchCount)
```

# Existing Primitives

## Pointer-to-Pointer BatchedGEMM



# A new primitive: StridedBatchedGEMM

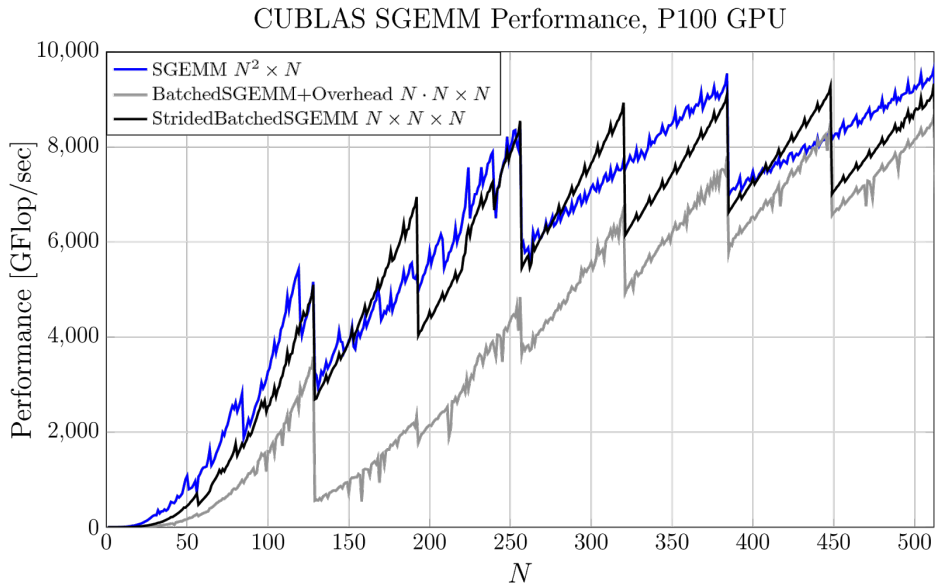
$$C[p] = \alpha\,\mathrm{op}(A[p])\,\mathrm{op}(B[p]) + \beta\,C[p]$$

- Pointer-to-Pointer BatchedGEMM requires memory allocation and pre-computation.
- Solution: StridedBatchedGEMM with fixed strides.
  - Special case of Pointer-to-pointer BatchedGEMM.
  - No Pointer-to-pointer data structure or overhead.

```
cublas<T>gemmStridedBatched(cublasHandle_t handle,
                            cublasOperation_t transA, cublasOperation_t transB,
                            int M, int N, int K,
                            const T* alpha,
                            const T* A, int ldA1, int strideA,
                            const T* B, int ldB1, int strideB,
                            const T* beta,
                            T* C, int ldC1, int strideC,
                            int batchCount)
```

# A new primitive: StridedBatchedGEMM

- Performance on par with pure GEMM (P100 and beyond).



# StridedBatchedGEMM

## Documentation in cuBLAS 8.0:

```
$$ grep StridedBatched -A 17 /usr/local/cuda/include/cublas_api.h
2320:CUBLASAPI cublasStatus_t cublasSgemmStridedBatched (cublasHandle_t handle,
2321-                                                    cublasOperation_t transa,
2322-                                                    cublasOperation_t transb,
2323-                                                    int m,
2324-                                                    int n,
2325-                                                    int k,
2326-                                                    const float *alpha,  // host or device pointer
2327-                                                    const float *A,
2328-                                                    int lda,
2329-                                                    long long int strideA,   // purposely signed
2330-                                                    const float *B,
2331-                                                    int ldb,
2332-                                                    long long int strideB,
2333-                                                    const float *beta,   // host or device pointer
2334-                                                    float *C,
2335-                                                    int ldc,
2336-                                                    long long int strideC,
2337-                                                    int batchCount);
...
```

# Tensor Contraction with Extended BLAS Primitives

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

$$C_{mnp} \equiv C[m + n \cdot \mathsf{ldC1} + p \cdot \mathsf{ldC2}]$$

| Case | Contraction | Kernel1 | Kernel2 | Case | Contraction | Kernel1 | Kernel2 |
|---|---|---|---|---|---|---|---|
| 1.1 | $A_{mk}B_{knp}$ | $C_{m(np)} = A_{mk}B_{k(np)}$ | $C_{mn[p]} = A_{mk}B_{kn[p]}$ | 4.1 | $A_{kn}B_{kmp}$ | $C_{mn[p]} = B^{\top}_{km[p]}A_{kn}$ | |
| 1.2 | $A_{mk}B_{kpn}$ | $C_{mn[p]} = A_{mk}B_{k[p]n}$ | $C_{m[n]p} = A_{mk}B_{kp[n]}$ | 4.2 | $A_{kn}B_{kpm}$ | $C_{mn[p]} = B^{\top}_{k[p]m}A_{kn}$ | |
| 1.3 | $A_{mk}B_{nkp}$ | $C_{mn[p]} = A_{mk}B^{\top}_{nk[p]}$ | | 4.3 | $A_{kn}B_{mkp}$ | $C_{mn[p]} = B_{mk[p]}A_{kn}$ | |
| 1.4 | $A_{mk}B_{pkn}$ | $C_{m[n]p} = A_{mk}B^{\top}_{pk[n]}$ | | 4.4 | $A_{kn}B_{pkm}$ | | |
| 1.5 | $A_{mk}B_{npk}$ | $C_{m(np)} = A_{mk}B^{\top}_{(np)k}$ | $C_{mn[p]} = A_{mk}B^{\top}_{n[p]k}$ | 4.5 | $A_{kn}B_{mpk}$ | $C_{mn[p]} = B_{m[p]k}A_{kn}$ | |
| 1.6 | $A_{mk}B_{pnk}$ | $C_{m[n]p} = A_{mk}B^{\top}_{p[n]k}$ | | 4.6 | $A_{kn}B_{pmk}$ | | |
| 2.1 | $A_{km}B_{knp}$ | $C_{m(np)} = A^{\top}_{km}B_{k(np)}$ | $C_{mn[p]} = A^{\top}_{km}B_{kn[p]}$ | 5.1 | $A_{pk}B_{kmn}$ | $C_{(mn)p} = B^{\top}_{k(mn)}A^{\top}_{pk}$ | $C_{m[n]p} = B^{\top}_{km[n]}A^{\top}_{pk}$ |
| 2.2 | $A_{km}B_{kpn}$ | $C_{mn[p]} = A^{\top}_{km}B_{k[p]n}$ | $C_{m[n]p} = A^{\top}_{km}B_{kp[n]}$ | 5.2 | $A_{pk}B_{knm}$ | $C_{m[n]p} = B^{\top}_{k[n]m}A^{\top}_{pk}$ | |
| 2.3 | $A_{km}B_{nkp}$ | $C_{mn[p]} = A^{\top}_{km}B^{\top}_{nk[p]}$ | | 5.3 | $A_{pk}B_{mkn}$ | $C_{m[n]p} = B_{mk[n]}A^{\top}_{pk}$ | |
| 2.4 | $A_{km}B_{pkn}$ | $C_{m[n]p} = A^{\top}_{km}B^{\top}_{pk[n]}$ | | 5.4 | $A_{pk}B_{nkm}$ | | |
| 2.5 | $A_{km}B_{npk}$ | $C_{m(np)} = A^{\top}_{km}B^{\top}_{(np)k}$ | $C_{mn[p]} = A^{\top}_{km}B^{\top}_{n[p]k}$ | 5.5 | $A_{pk}B_{mnk}$ | $C_{(mn)p} = B_{(mn)k}A^{\top}_{pk}$ | $C_{m[n]p} = B_{m[n]k}A^{\top}_{pk}$ |
| 2.6 | $A_{km}B_{pnk}$ | $C_{m[n]p} = A^{\top}_{km}B^{\top}_{p[n]k}$ | | 5.6 | $A_{pk}B_{nmk}$ | | |
| 3.1 | $A_{nk}B_{kmp}$ | $C_{mn[p]} = B^{\top}_{km[p]}A^{\top}_{nk}$ | | 6.1 | $A_{kp}B_{kmn}$ | $C_{(mn)p} = B^{\top}_{k(mn)}A_{kp}$ | $C_{m[n]p} = B^{\top}_{km[n]}A_{kp}$ |
| 3.2 | $A_{nk}B_{kpm}$ | $C_{mn[p]} = B^{\top}_{k[p]m}A^{\top}_{nk}$ | | 6.2 | $A_{kp}B_{knm}$ | $C_{m[n]p} = B^{\top}_{k[n]m}A_{kp}$ | |
| 3.3 | $A_{nk}B_{mkp}$ | $C_{mn[p]} = B_{mk[p]}A^{\top}_{nk}$ | | 6.3 | $A_{kp}B_{mkn}$ | $C_{m[n]p} = B_{mk[n]}A_{kp}$ | |
| 3.4 | $A_{nk}B_{pkm}$ | | | 6.4 | $A_{kp}B_{nkm}$ | | |
| 3.5 | $A_{nk}B_{mpk}$ | $C_{mn[p]} = B_{m[p]k}A^{\top}_{nk}$ | | 6.5 | $A_{kp}B_{mnk}$ | $C_{(mn)p} = B_{(mn)k}A_{kp}$ | $C_{m[n]p} = B_{m[n]k}A_{kp}$ |
| 3.6 | $A_{nk}B_{pmk}$ | | | 6.6 | $A_{kp}B_{nmk}$ | | |

# Tensor Contraction with Extended BLAS Primitives

| Case | Contraction | Kernel1 | Kernel2 | Kernel3 |
|---|---|---|---|---|
| 1.1 | $A_{mk}B_{knp}$ | $C_{m(np)} = A_{mk}B_{k(np)}$ | $C_{mn[p]} = A_{mk}B_{kn[p]}$ | $C_{m[n]p} = A_{mk}B_{k[n]p}$ |
| 6.1 | $A_{kp}B_{kmn}$ | $C_{(mn)p} = B^{\top}_{k(mn)}A_{kp}$ | $C_{m[n]p} = B^{\top}_{km[n]}A_{kp}$ | |

Example: Mappings to Level 3 BLAS routines

- Case 1.1, Kernel2: $C_{mn[p]} = A_{mk}B_{kn[p]}$

```
cublasDgemmStridedBatched(handle,
                          CUBLAS_OP_N, CUBLAS_OP_N,
                          M, N, K,
                          &alpha,
                          A, ldA1, 0,
                          B, ldB1, ldB2,
                          &beta,
                          C, ldC1, ldC2,
                          P)
```

# Tensor Contraction with Extended BLAS Primitives

| Case | Contraction | Kernel1 | Kernel2 | Kernel3 |
|---|---|---|---|---|
| 1.1 | $A_{mk}B_{knp}$ | $C_{m(np)} = A_{mk}B_{k(np)}$ | $C_{mn[p]} = A_{mk}B_{kn[p]}$ | $C_{m[n]p} = A_{mk}B_{k[n]p}$ |
| 6.1 | $A_{kp}B_{kmn}$ | $C_{(mn)p} = B^{\top}_{k(mn)}A_{kp}$ | $C_{m[n]p} = B^{\top}_{km[n]}A_{kp}$ | |

Example: Mappings to Level 3 BLAS routines

- Case 6.1, Kernel2: $C_{m[n]p} = B^{\top}_{km[n]}A_{kp}$

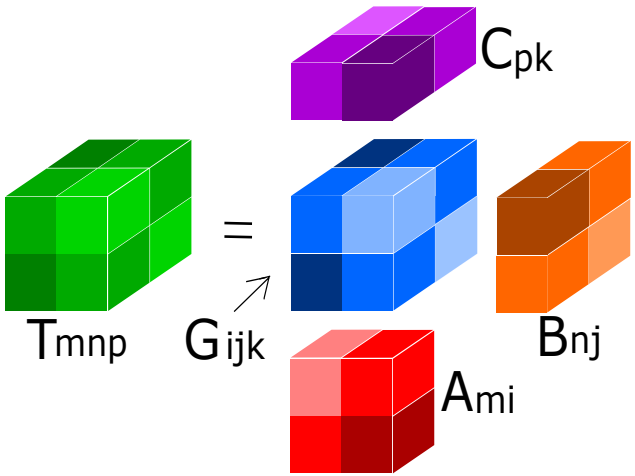
```
cublasDgemmStridedBatched(handle,
                          CUBLAS_OP_T, CUBLAS_OP_N,
                          M, P, K,
                          &alpha,
                          B, ldB1, ldB2,
                          A, ldA1, 0,
                          &beta,
                          C, ldC2, ldC1,
                          N)
```

---

Y. Shi, U.N. Niranjan, **A.**, C. Cecka "Tensor Contractions with Extended BLAS Kernels on CPU and GPU" in HiPC2016.

# Applications: Tucker Decomposition

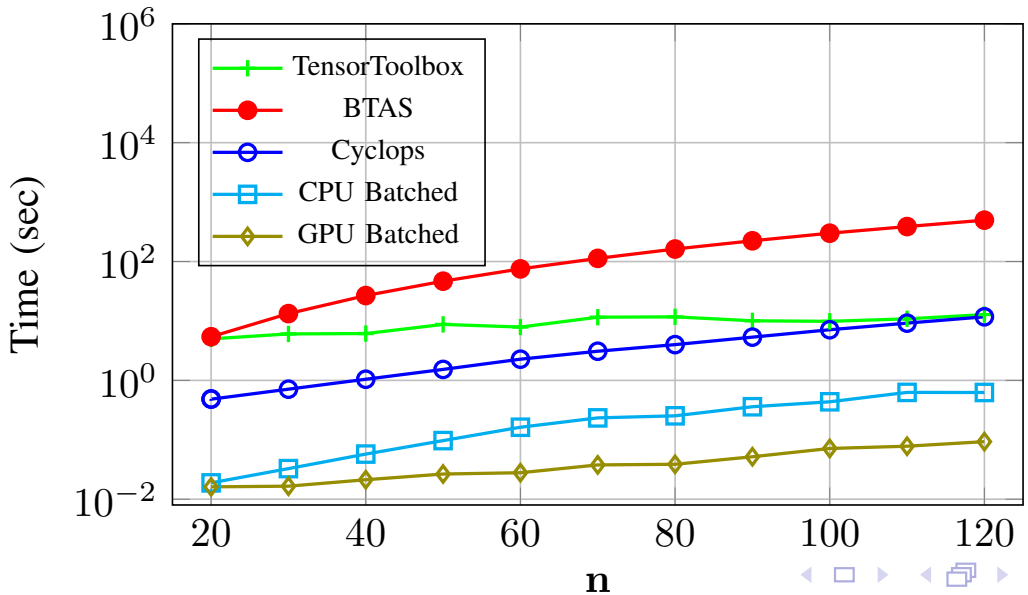
$$T_{mnp} = G_{ijk} A_{mi} B_{nj} C_{pk}$$




**Main steps in the algorithm**

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

Performance on Tucker decomposition:



# Applications: FFT

Low-Communication FFT for multiple GPUs involves tensor contractions.

$$
\begin{aligned}
T_{pib} &= S2T^{(p)}_{ijs}\, S_{pj(b+s)} && \Longrightarrow && T_{pib} &= S2T^{(p)}_{i(js)}\, S_{p(js)b} \\
M_{pqb} &= S2M_{qi}\, S_{pib} && \Longrightarrow && M_{pq[b]} &= S_{pi[b]}\, S2M^{T}_{qi} \\
M_{pqb'} &= M2M^{-}_{qm}\, M_{pmb^-} + M2M^{+}_{qm}\, M_{pmb^+} && \Longrightarrow && M_{pq[b']} &= M_{pM[b]}\, M2M^{T}_{qM} \\
r_p &= 1_{ib}\, S_{pib} = 1_{qb}\, M_{pqb} && \Longrightarrow && r_p &= 1_{(qb)}\, M_{p(qb)} \\
L_{pnb} &= M2L^{(p)}_{nms}\, M_{pm(b+s)} && \Longrightarrow && L_{pnb} &= M2L^{(p)}_{n(ms)}\, M_{p(ms)b} \\
L_{pqb^\pm} &= L2L^{\pm}_{qm}\, L_{pmb'} && \Longrightarrow && L_{pq[b]} &= L_{pM[b']}\, M2M_{qM} \\
T_{pib} &= L2T_{iq}\, L_{pqb} && \Longrightarrow && T_{pi[b]} &= L_{pq[b]}\, S2M_{qi}
\end{aligned}
$$

- StridedBatchedGEMM composes 75%+ of the runtime.
  - Essential to the performance.
  - Two custom kernels are precisely interleaved GEMMs.

- 2 P100 GPUs: 1.3x over cuFFTXT.
- 8 P100 GPUs: 2.1x over cuFFTXT.

---

C. Cecka "Low-Communication FFT with Fast Multipole Method" GTC 2017.

# Summary of this Section

- Tensor contractions provide significant space savings in deep learning.
- Fast tensor contractions using extended BLAS primitive: StridedBatchedGEMM
- Avoids explicit transpositions or permutations.
- **10x**(GPU) and **2x**(CPU) speedup on small/moderate sized tensors.
- **Available in cuBLAS 8.0**

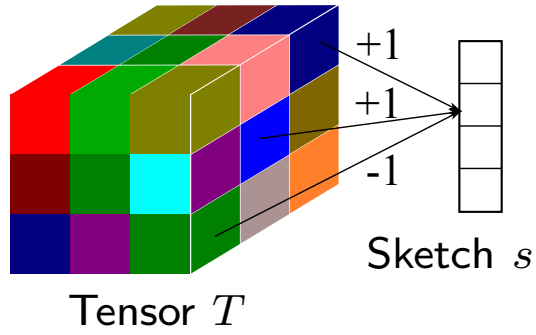
# Summary of this Section

- Tensor contractions provide significant space savings in deep learning.
- Fast tensor contractions using extended BLAS primitive: StridedBatchedGEMM
- Avoids explicit transpositions or permutations.
- **10x**(GPU) and **2x**(CPU) speedup on small/moderate sized tensors.
- **Available in cuBLAS 8.0**
- Future work:
  - Exceptional case kernels/performance/interface??
  - Library Optimizations
    - ★ Matrix stride zero – Persistent Matrix Strided Batched GEMM

# Outline

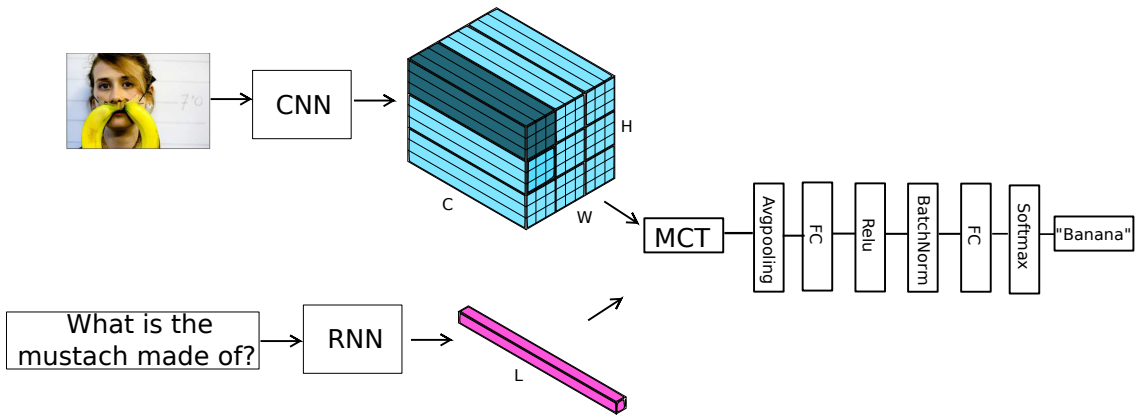
# Tensor Sketches

- Randomized dimensionality reduction through sketching.
  - Complexity independent of tensor order: exponential gain!




## Applications

- Tensor Decomposition via Sketching
- Visual Question and Answering

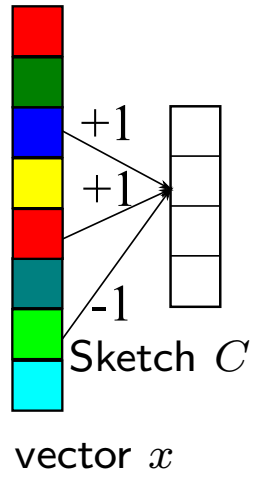


# Tensor Sketching

- Dimensionality reduction through sketching.

## Count Sketch for vector $x$

- $C[h[i]] += s[i]x[i]$, for $s[i] \in \{-1, +1\}^n$

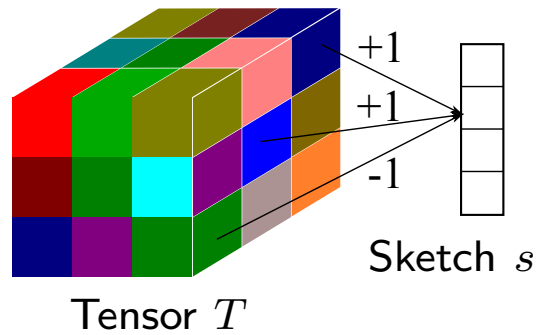



---

## Count Sketch for outer products $x \otimes y$

- Convolution of count sketches

$$C(x \otimes y, h, s) = C(x, h, s) * C(y, h, s)$$
$$= FFT^{-1}(FFT(C(x, h, s))FFT(C(y, h, s)))$$



# Accelerated tensor low-rank decomposition

## Symmetric tensor CP decomposition

For symmetric tensor $T \in \mathbb{R}^{n \times n \times n}$, find $\{(\lambda_i, u_i)\}_{i=1}^k$ to minimize

$$\| T - \sum_{i=1}^k \lambda_i u_i^{\otimes 3} \|_F^2.$$

- Wide application in data mining and latent variable models.
- **Tensor power iteration**: $u^{(t+1)} = T(I, u^{(t)}, u^{(t)}) / \| T(I, u^{(t)}, u^{(t)}) \|_2$.
- Accelerated tensor power iteration via sketching:
  - TensorSketch: $s(T) \in \mathbb{R}^b$, for $n < b \ll n^3$.

$$\begin{aligned}
[T(I, u, u)]_i &\approx \langle s(T), \quad s(u \otimes u \otimes e_i) \rangle \\
&= \langle \mathcal{F}(s(T)), \quad \mathcal{F}(s(u)) .* \mathcal{F}(s(u)) .* \mathcal{F}(s(e_i)) \rangle \\
&= \langle \mathcal{F}^{-1}(\mathcal{F}(s(T)) .* \overline{\mathcal{F}(s(u))} .* \overline{\mathcal{F}(s(u))}), \quad s(e_i) \rangle.
\end{aligned}$$

  - Time complexity: $O(n^3) \to O(n + b \log b)$.

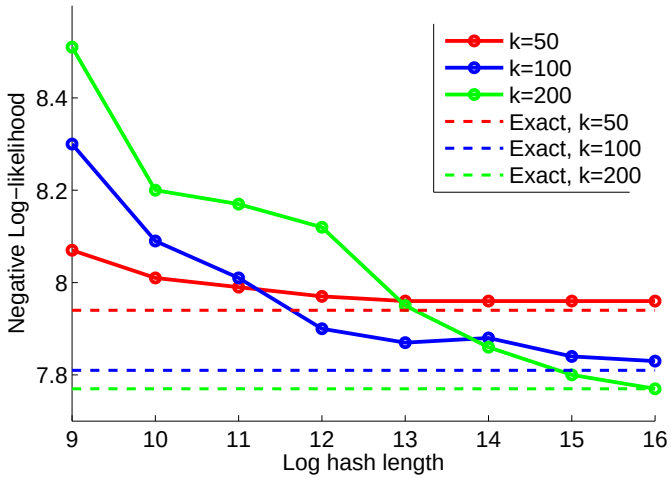
# Efficient spectral method for topic modeling

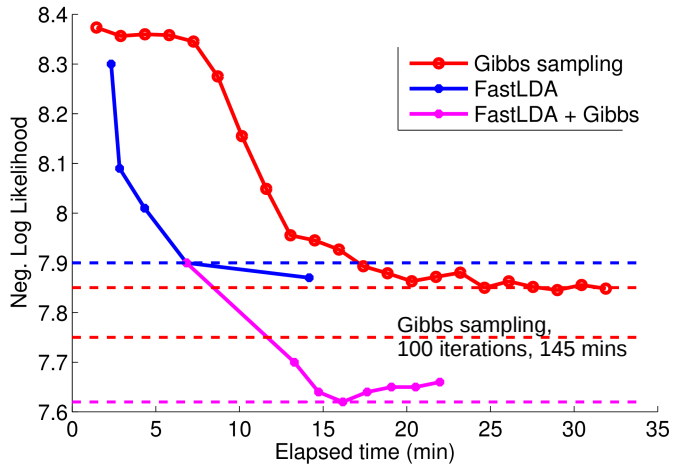
## Topic modeling

$V$: vocabulary size; $k$: number of topics. Recover topic distributions $\mu_1, \cdots, \mu_k \in \mathbb{R}^V$ from $N$ unlabeled documents.

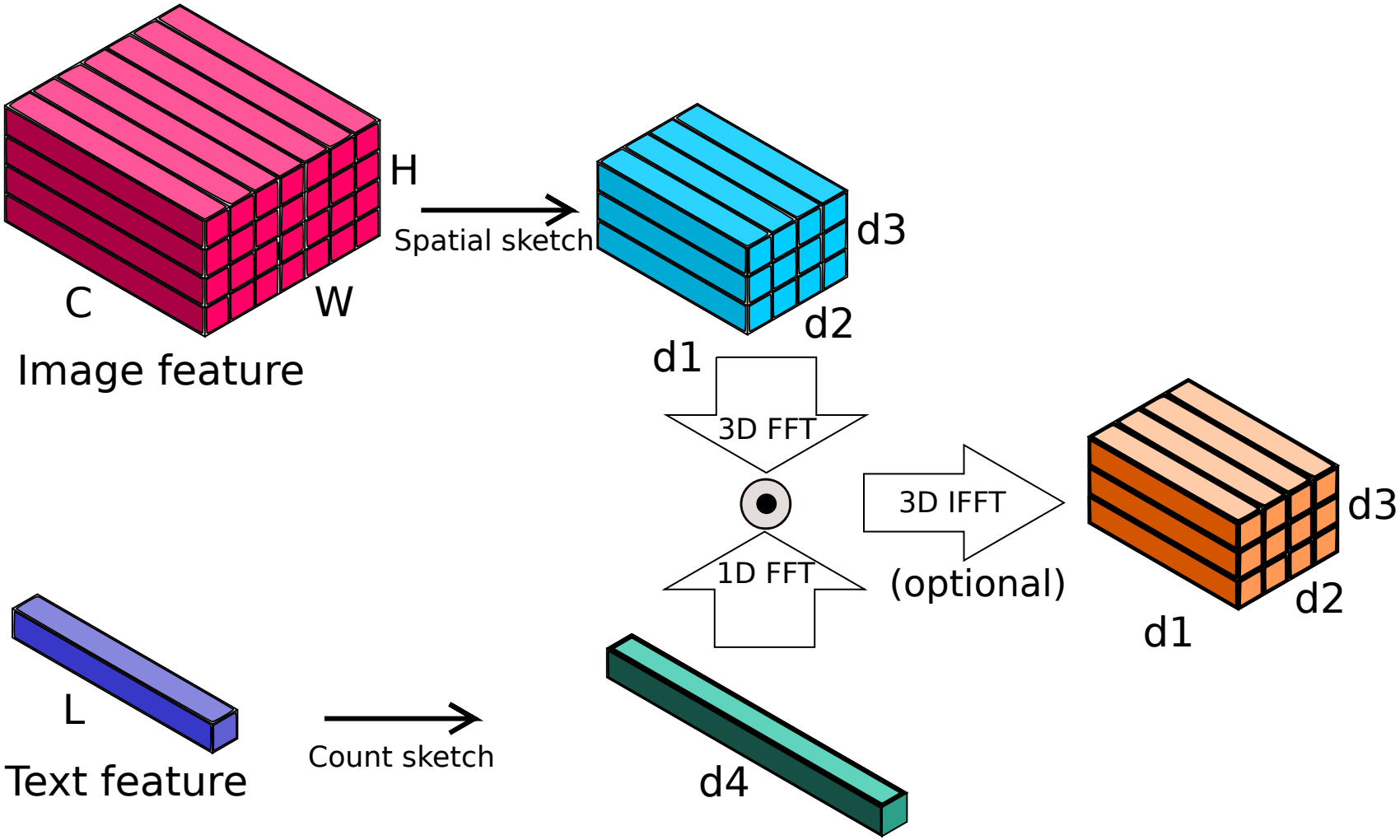
Figure 1: Negative log-likelihood and running time (min) on Wikipedia dataset.

| $k$ | | like. | time | $\log_2 b$ | iters | $k$ | like. | time | $\log_2 b$ | iters |
|---|---|---|---|---|---|---|---|---|---|---|
| 200 | Spectral | 7.49 | **34** | 12 | - | 300 | 7.39 | **56** | 13 | - |
| | Gibbs | 6.85 | 561 | - | 30 | | 6.38 | 818 | - | 30 |
| | Hybrid | **6.77** | 144 | 12 | 5 | | **6.31** | 352 | 13 | 10 |

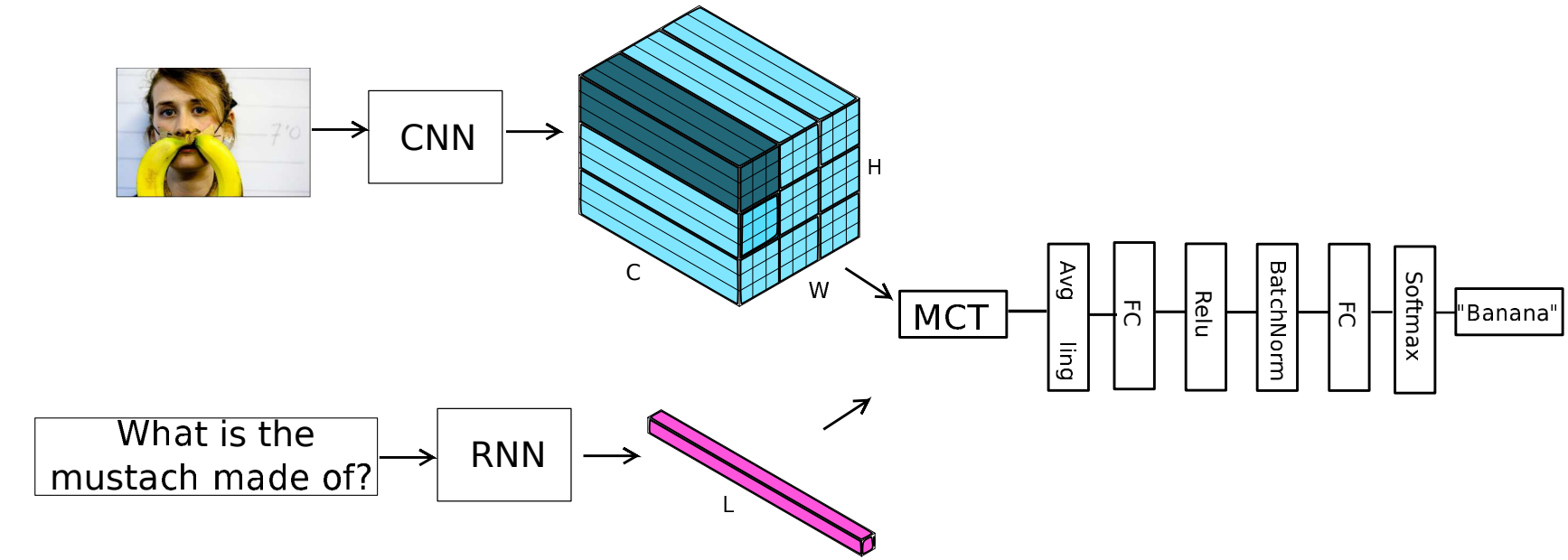


# Multimodal Tensor Pooling

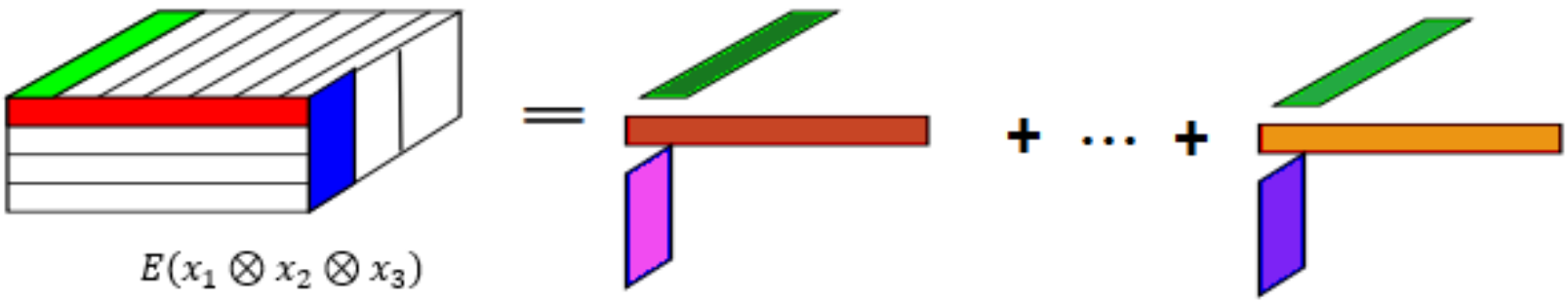


# MCT in Visual Question & Answering



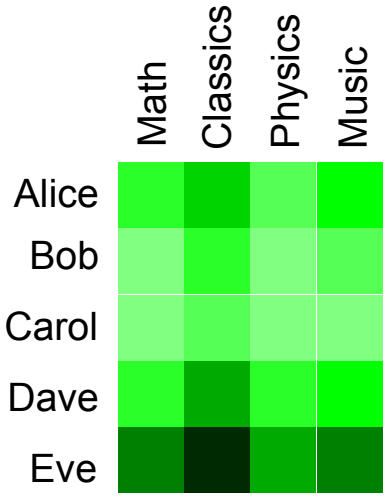
# Tensor Decompositions

$E(x_1 \otimes x_2)$



$E(x_1 \otimes x_2 \otimes x_3)$

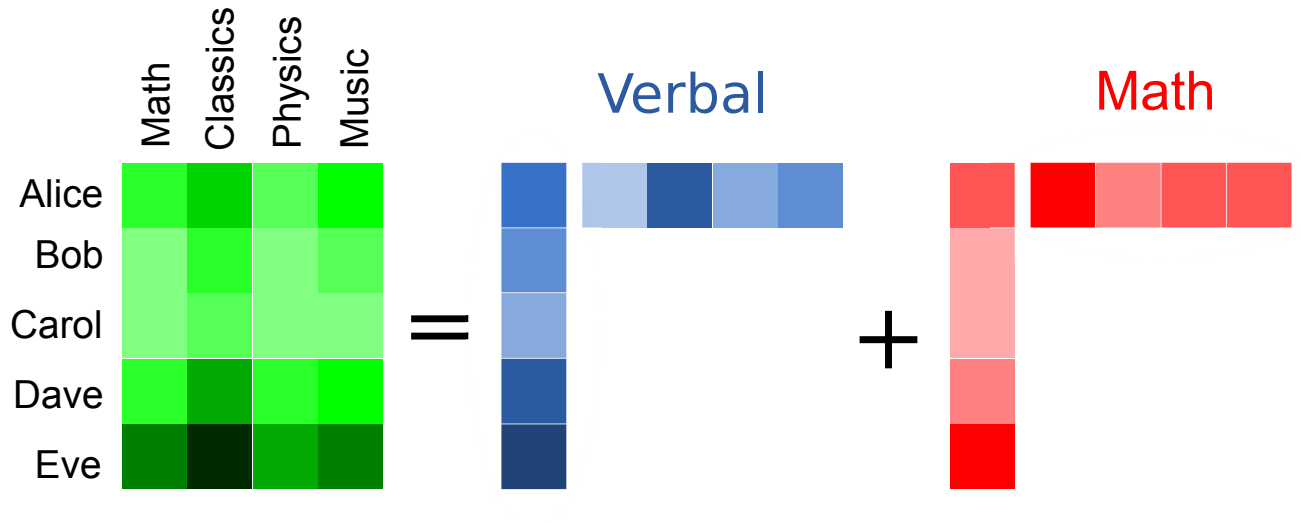
# Example: Discovering Latent Factors




- List of scores for students in different tests
- Learn hidden factors for Verbal and Mathematical Intelligence [C. Spearman 1904]

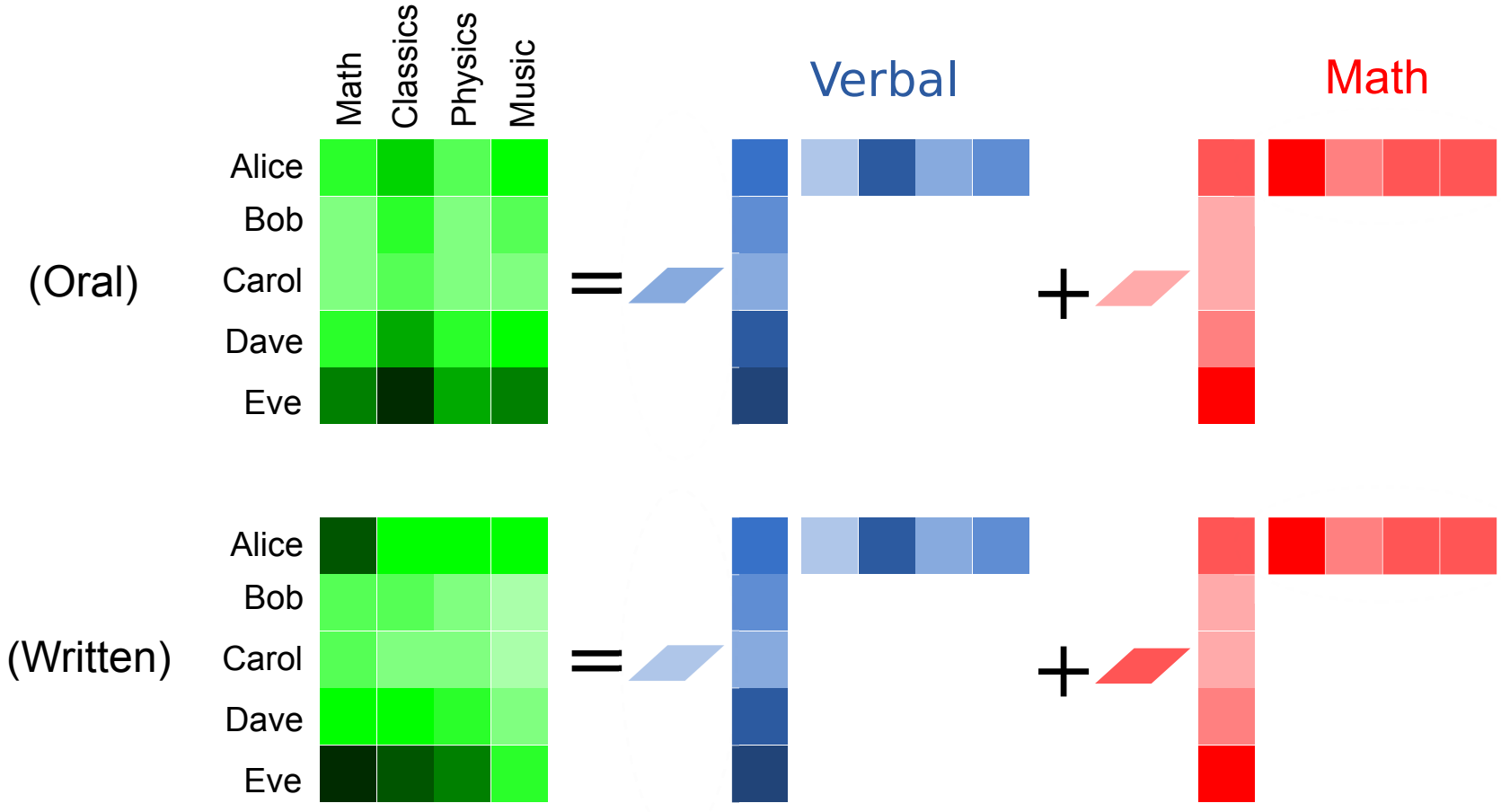
$$\text{Score}\ (\text{student,test}) = \text{student}_{\text{verbal-intlg}} \times \text{test}_{\text{verbal}} + \text{student}_{\text{math-intlg}} \times \text{test}_{\text{math}}$$

# Matrix Decomposition: Discovering Latent Factors



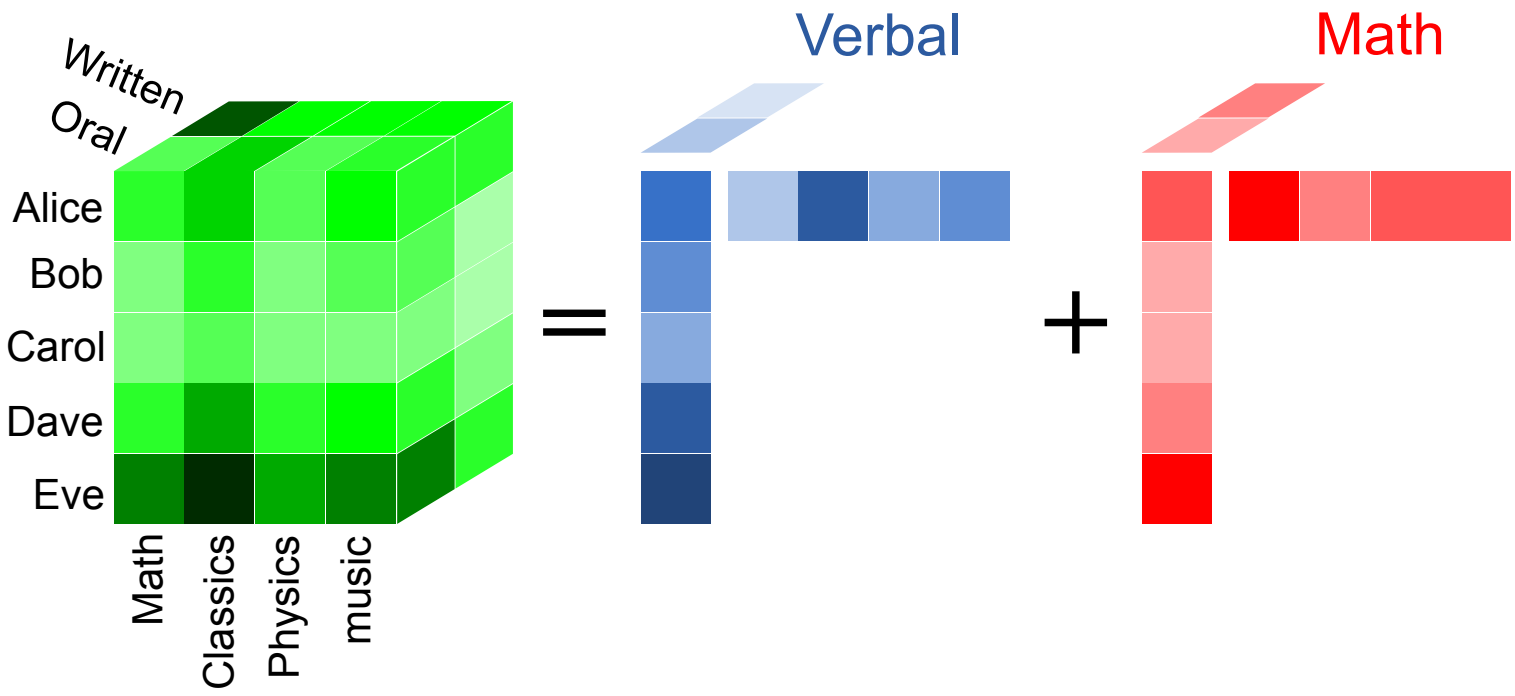

- Identifying hidden factors influencing the observations
- Characterized as matrix decomposition

# Tensor: Shared Matrix Decomposition




- Shared decomposition with different scaling factors
- Combine matrix slices as a tensor

# Tensor Decomposition



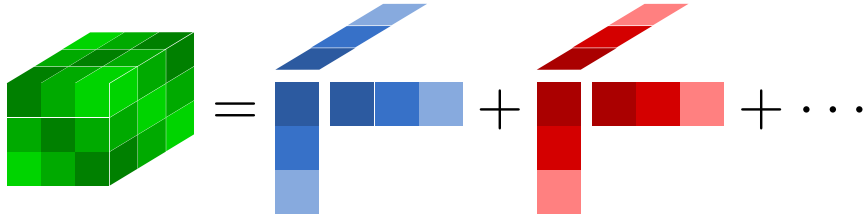

- Outer product notation:

$$T = u \otimes v \otimes w + \tilde{u} \otimes \tilde{v} \otimes \tilde{w}$$

$$\Updownarrow$$

$$T_{i_1,i_2,i_3} = u_{i_1} \cdot v_{i_2} \cdot w_{i_3} + \tilde{u}_{i_1} \cdot \tilde{v}_{i_2} \cdot \tilde{w}_{i_3}$$

# Identifiability under Tensor Decomposition

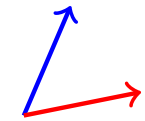


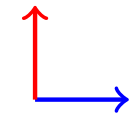
$$T = \lambda_1 a_1{}^{\otimes 3} + \lambda_2 a_2{}^{\otimes 3} + \cdots ,$$

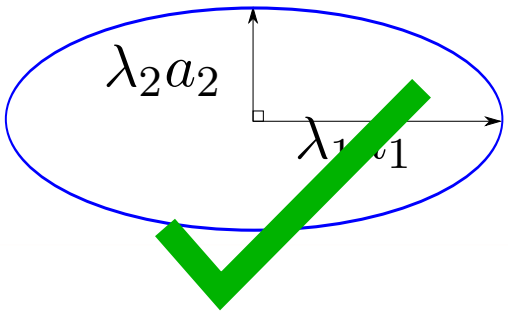
---

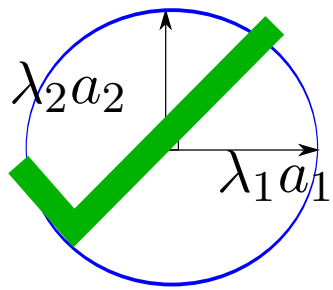
## Uniqueness of Tensor Decomposition [J. Kruskal 1977]

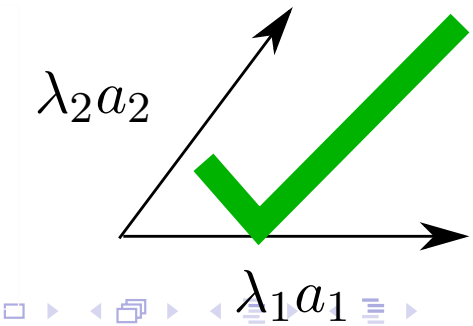
- Above tensor decomposition: unique when rank one pairs are linearly independent
- Matrix case: when rank one pairs are orthogonal





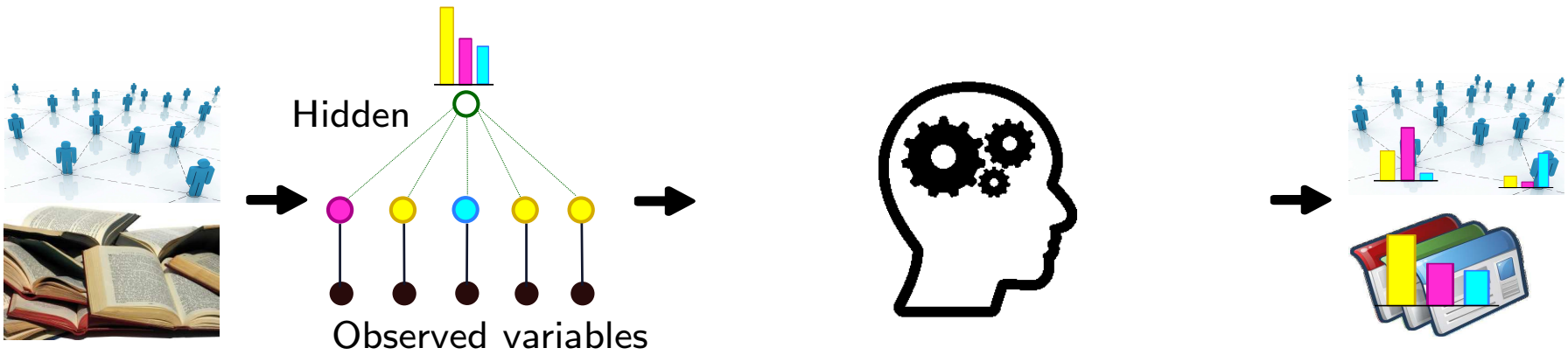


# Unsupervised Learning via Probabilistic Models

Data $\rightarrow$ Model $\rightarrow$ Learning Algorithm $\rightarrow$ Predictions

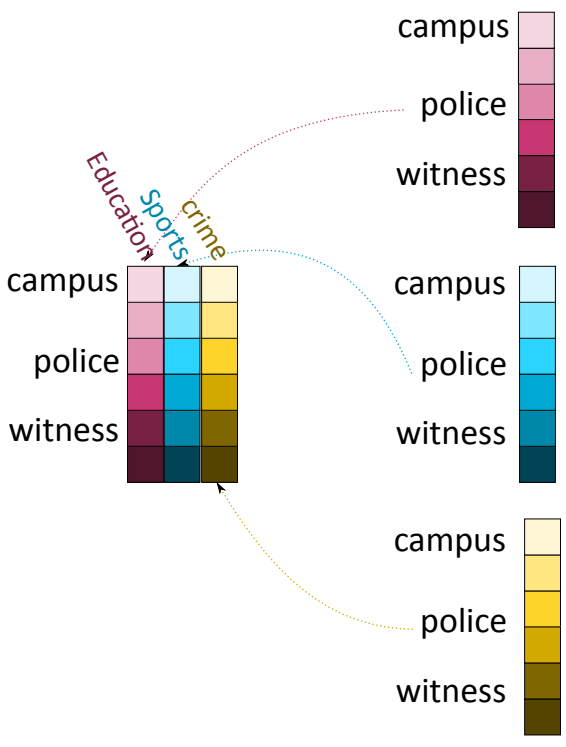


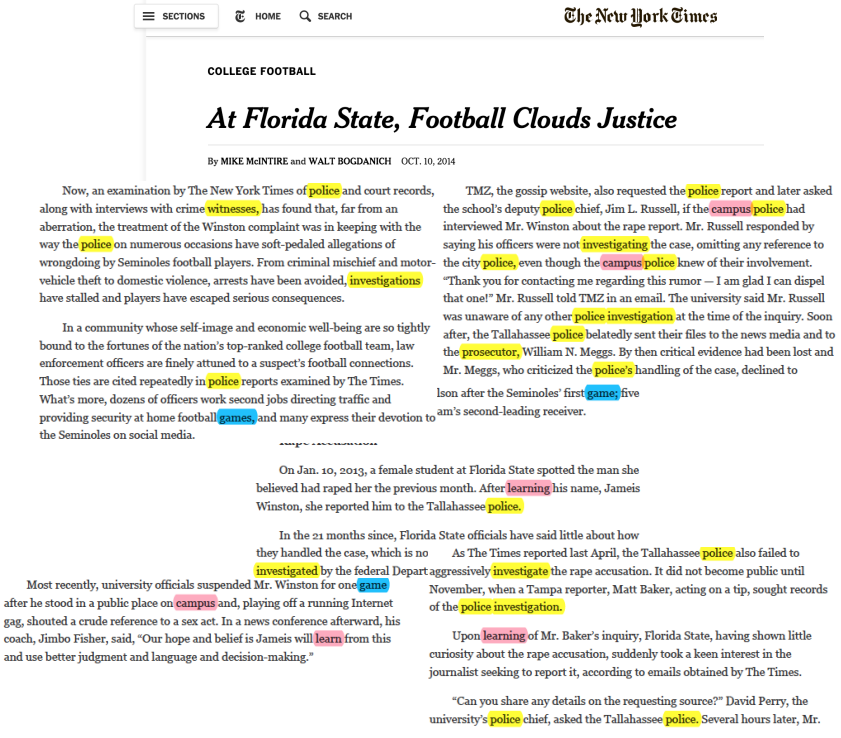

---

## Challenges in High dimensional Learning

- Dimension of $x \gg$ dim. of latent variable $h$.
- Learning is like finding needle in a haystack.
- Computationally & statistically challenging.

# Extracting Topics from Documents





SECTIONS HOME SEARCH The New York Times

COLLEGE FOOTBALL

*At Florida State, Football Clouds Justice*

By MIKE McINTIRE and WALT BOGDANICH OCT. 10, 2014

Now, an examination by The New York Times of police and court records, along with interviews with crime witnesses, has found that, far from an aberration, the treatment of the Winston complaint was in keeping with the way the police on numerous occasions have soft-pedaled allegations of wrongdoing by Seminoles football players. From criminal mischief and motor-vehicle theft to domestic violence, arrests have been avoided, investigations have stalled and players have escaped serious consequences.

In a community whose self-image and economic well-being are so tightly bound to the fortunes of the nation's top-ranked college football team, law enforcement officers are finely attuned to a suspect's football connections. Those ties are cited repeatedly in police reports examined by The Times. What's more, dozens of officers work second jobs directing traffic and providing security at home football games, and many express their devotion to the Seminoles on social media.

TMZ, the gossip website, also requested the police report and later asked the school's deputy police chief, Jim L. Russell, if the campus police had interviewed Mr. Winston about the rape report. Mr. Russell responded by saying his officers were not investigating the case, omitting any reference to the city police, even though the campus police knew of their involvement. "Thank you for contacting me regarding this rumor — I am glad I can dispel that one!" Mr. Russell told TMZ in an email. The university said Mr. Russell was unaware of any other police investigation at the time of the inquiry. Soon after, the Tallahassee police belatedly sent their files to the news media and to the prosecutor, William N. Meggs. By then critical evidence had been lost and Mr. Meggs, who criticized the police's handling of the case, declined to

lson after the Seminoles' first game; five
am's second-leading receiver.

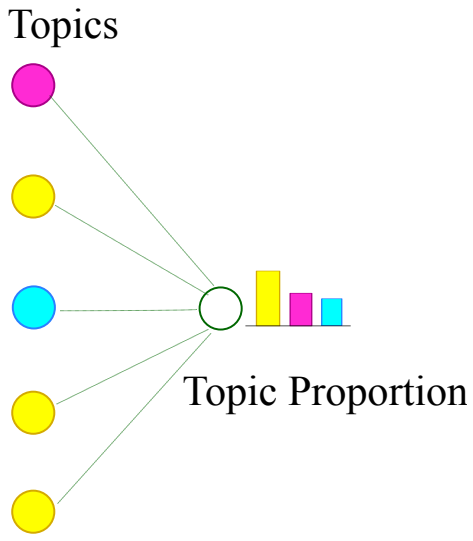
On Jan. 10, 2013, a female student at Florida State spotted the man she believed had raped her the previous month. After learning his name, Jameis Winston, she reported him to the Tallahassee police.

In the 21 months since, Florida State officials have said little about how they handled the case, which is no As The Times reported last April, the Tallahassee police also failed to investigated by the federal Depart aggressively investigate the rape accusation. It did not become public until November, when a Tampa reporter, Matt Baker, acting on a tip, sought records of the police investigation.

Most recently, university officials suspended Mr. Winston for one game after he stood in a public place on campus and, playing off a running Internet gag, shouted a crude reference to a sex act. In a news conference afterward, his coach, Jimbo Fisher, said, "Our hope and belief is Jameis will learn from this and use better judgment and language and decision-making."

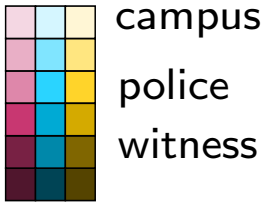
Upon learning of Mr. Baker's inquiry, Florida State, having shown little curiosity about the rape accusation, suddenly took a keen interest in the journalist seeking to report it, according to emails obtained by The Times.

"Can you share any details on the requesting source?" David Perry, the university's police chief, asked the Tallahassee police. Several hours later, Mr.




A., D. P. Foster, D. Hsu, S.M. Kakade, Y.K. Liu. "Two SVDs Suffice: Spectral decompositions for probabilistic topic modeling and latent Dirichlet allocation," NIPS 2012.

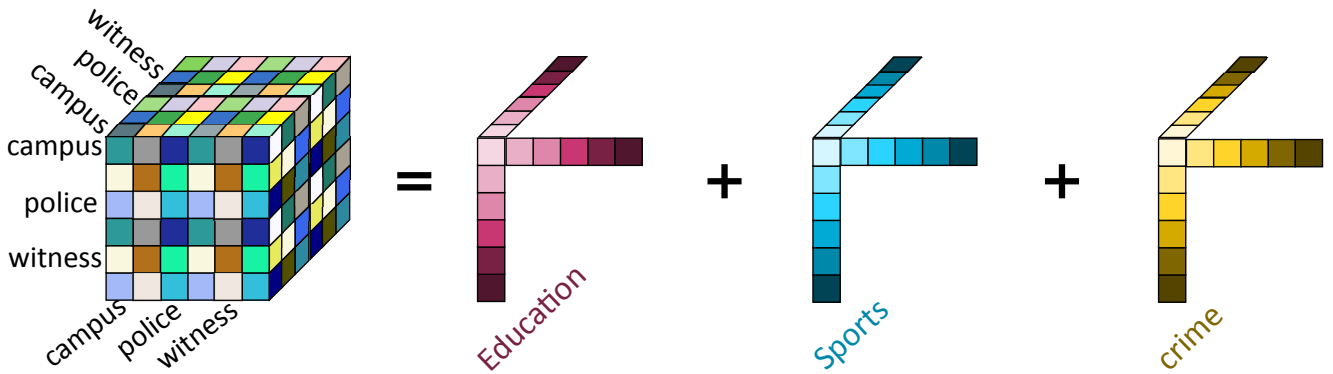
# Tensor Methods for Topic Modeling




- Topic-word matrix $\mathbb{P}[\text{word} = i | \text{topic} = j]$
- Linearly independent columns

---

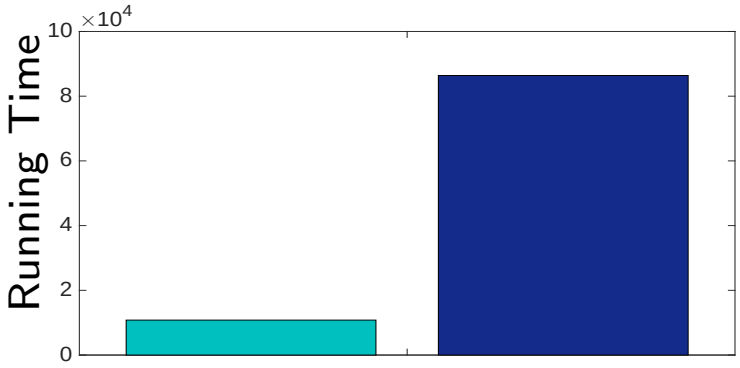
## Moment Tensor: Co-occurrence of Word Triplets

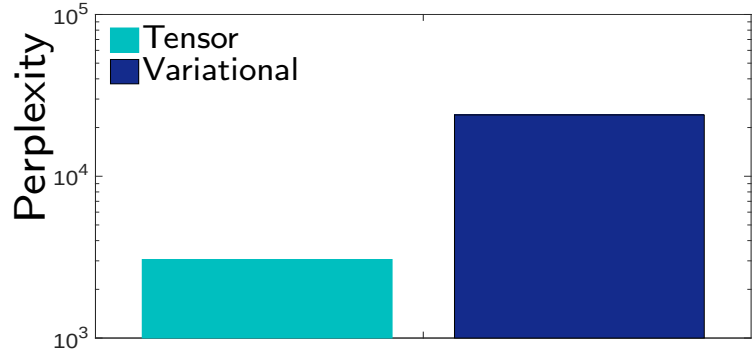


# Tensors vs. Variational Inference

Criterion: Perplexity $= \exp[-\text{likelihood}]$.

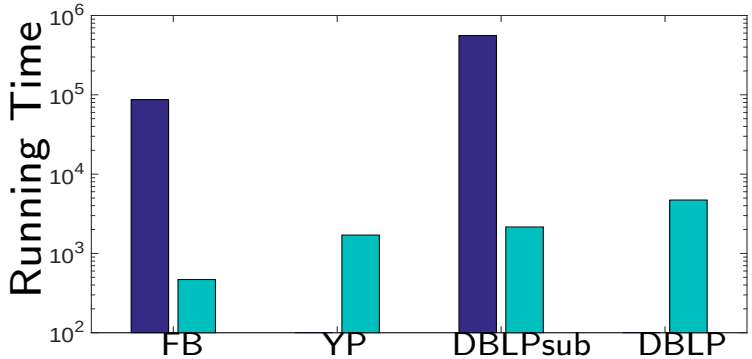
## Learning Topics from PubMed on Spark, 8mil articles

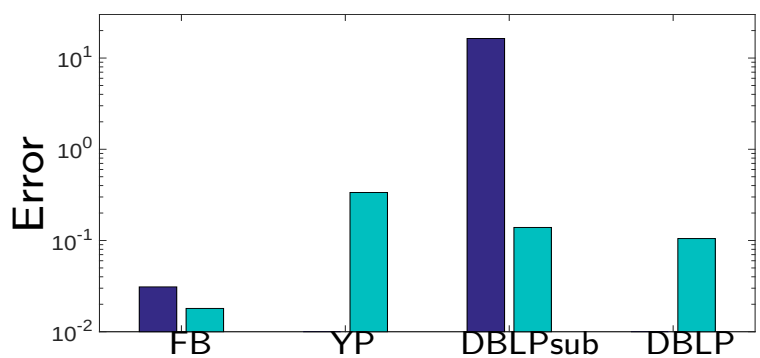



## Learning network communities from social network data

Facebook $n \sim 20k$, Yelp $n \sim 40k$, DBLP-sub $n \sim 1e5$, DBLP $n \sim 1e6$.



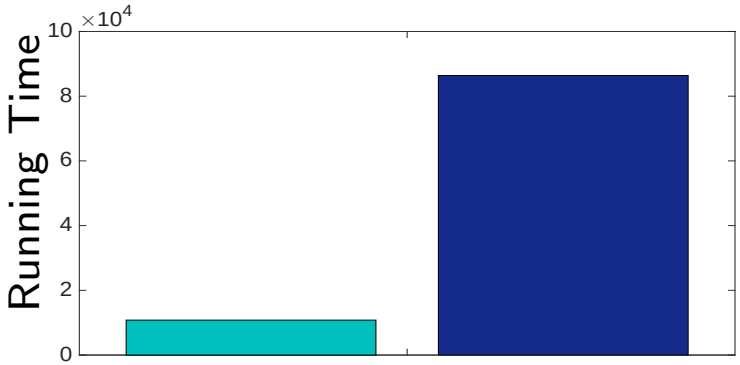



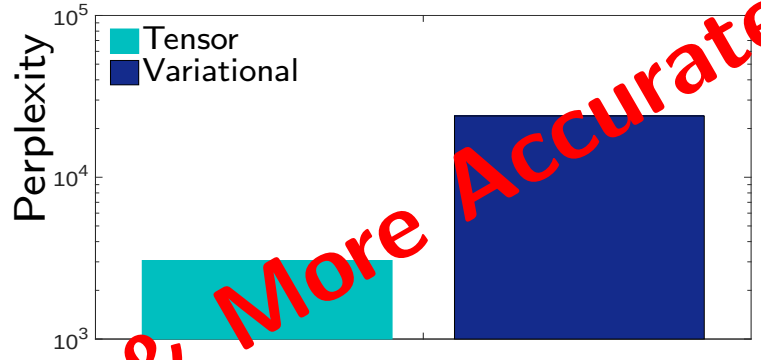

F. Huang, U.N. Niranjan, M. Hakeem, A, "Online tensor methods for training latent variable models," JMLR 2014.

# Tensors vs. Variational Inference

Criterion: Perplexity $= \exp[-\text{likelihood}]$.

## Learning Topics from PubMed on Spark, 8mil articles

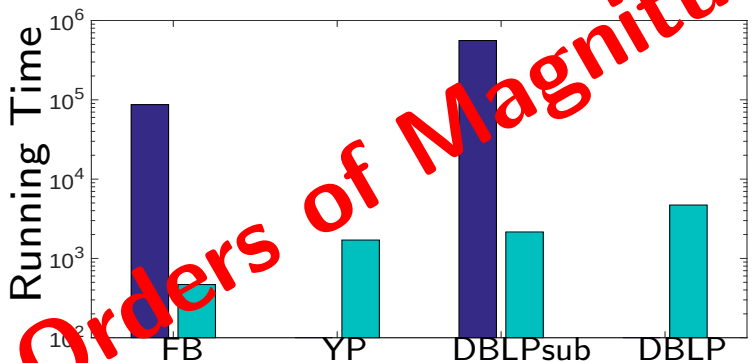


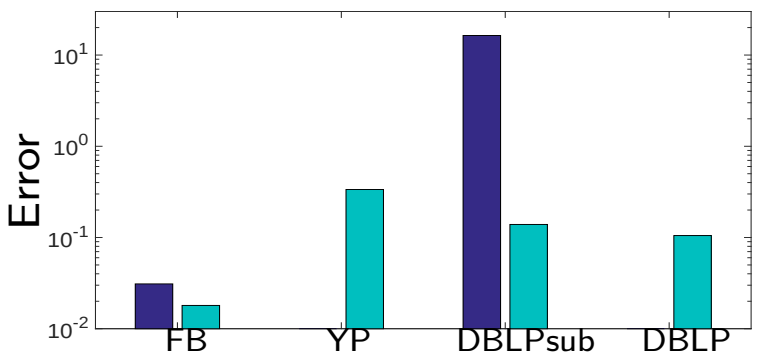




Orders of Magnitude Faster & More Accurate

## Learning network communities from social network data

Facebook $n \sim 20k$, Yelp $n \sim 40k$, DBLP-sub $n \sim 1e5$, DBLP $n \sim 1e6$.




F. Huang, U.N. Niranjan, M. Hakeem, A, "Online tensor methods for training latent variable models," JMLR 2014.

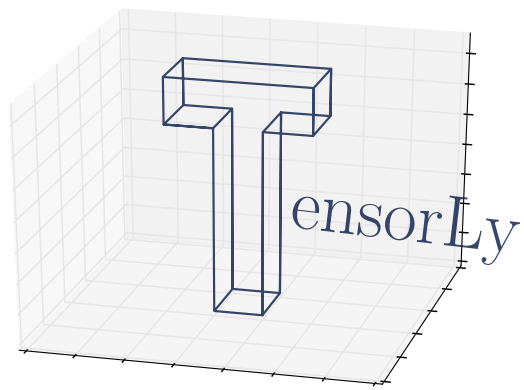
# TensorLy: Tensor Learning in Python

- Pure Python
- Integrated in the Python ecosystem
- Minimal dependencies (NumPy, SciPy and Matplotlib)
- Easy to use and extend
- Fast
- Extensively tested (unit-tests)
- Exhaustive documentation
- Open source, BSD licensed

# Tensorly yours,

Try it: pip install tensorly

`https://tensorly.github.io`



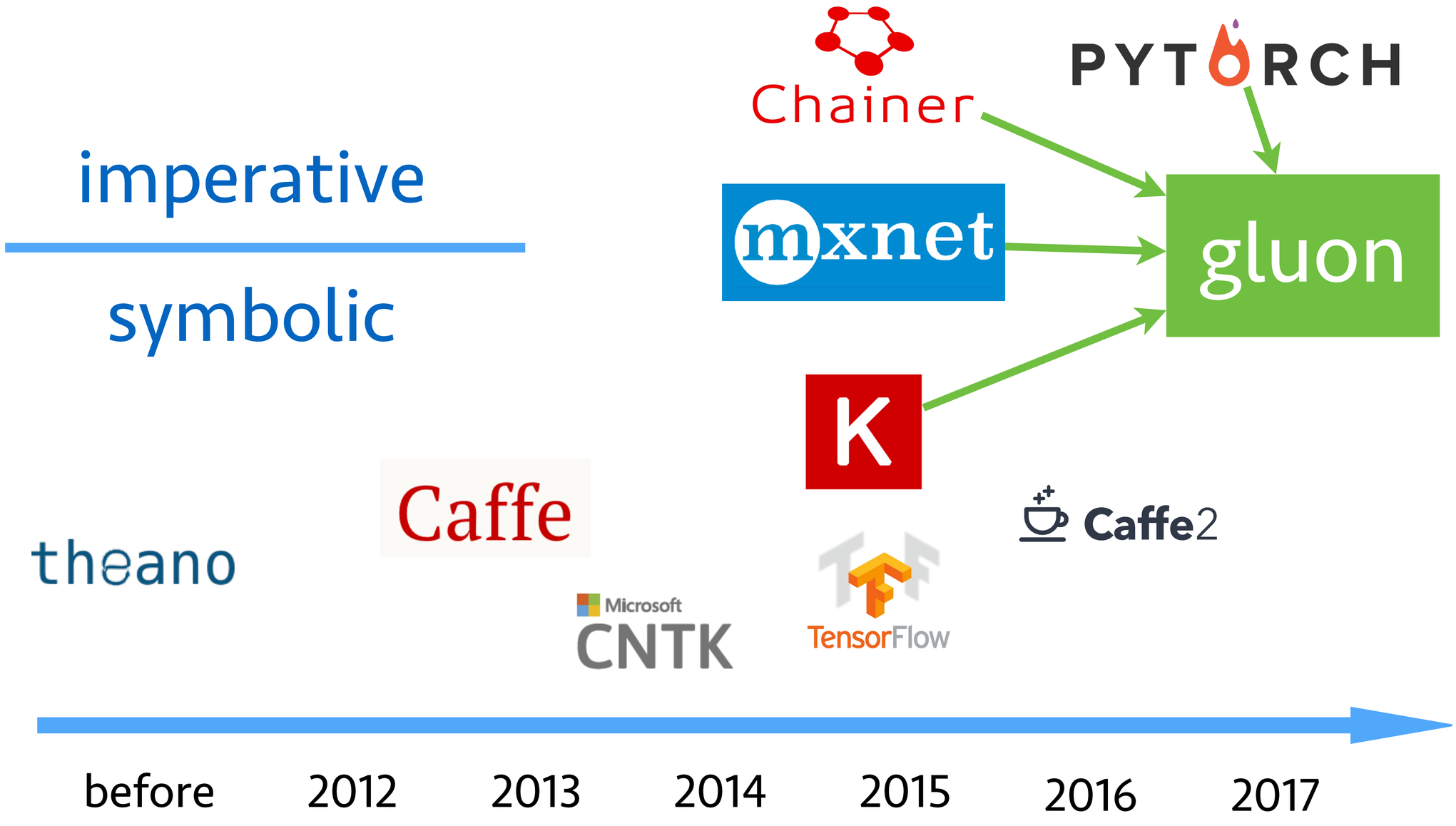

Contributions welcome!

# 3. MXNet

- Imperative and Declarative Programming
- Language Support
- Backend and Automatic Parallelization

image credit - wikipedia

amazon web services


imperative
symbolic
Chainer
PYTORCH
mxnet
gluon
K
Caffe
theano
Caffe2
Microsoft
CNTK
TensorFlow
before
2012
2013
2014
2015
2016
2017

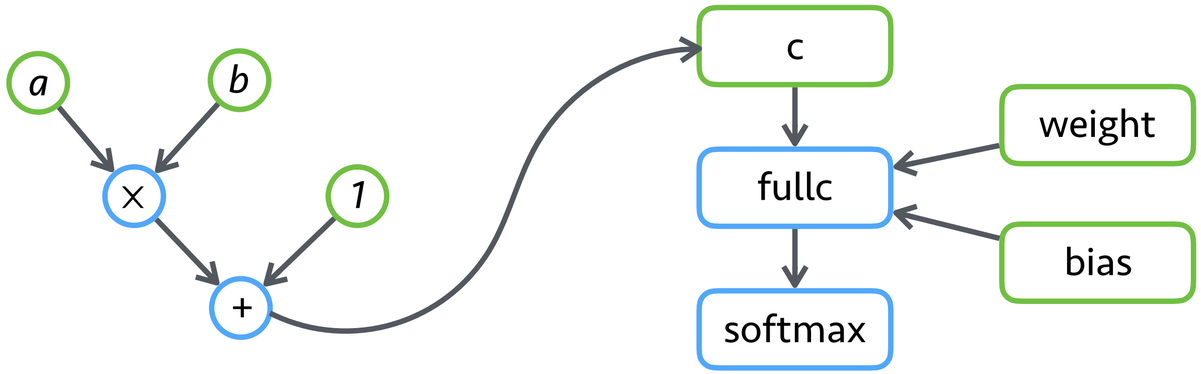
# Back-end System

Front-end

```
import mxnet as mx
a = mx.nd.zeros((100, 50))
b = mx.nd.ones((100, 50))
c = a * b
c += 1
```

```
import mxnet as mx
net = mx.symbol.Variable('data')
net = mx.symbol.FullyConnected(
        data=net, num_hidden=128)
net = mx.symbol.SoftmaxOutput(data=net)
texec = mx.module.Module(net)
texec.forward(data=c)
texec.backward()
```

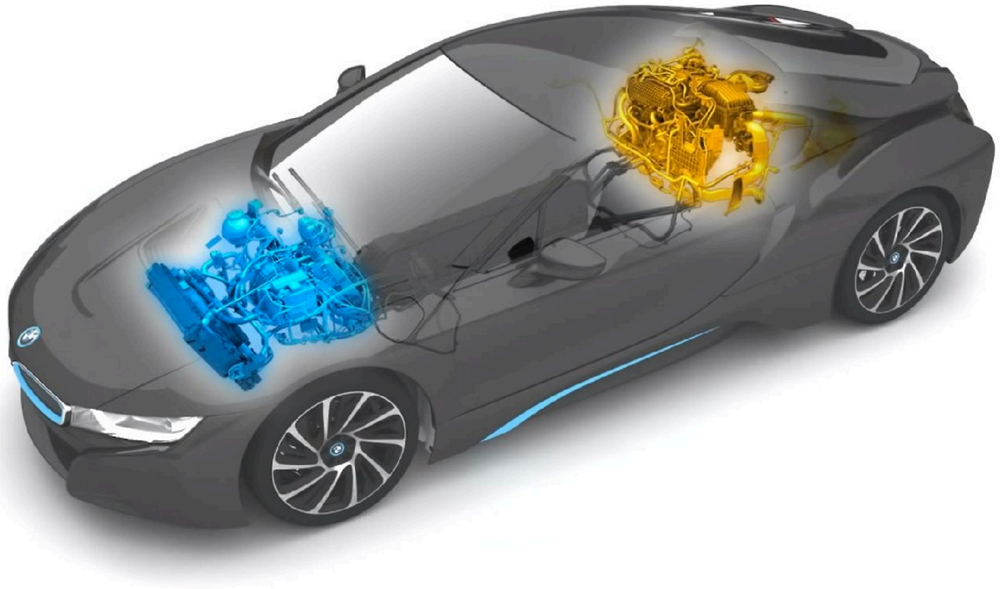
Back-end




- ✧ Optimization
  - ✓ Memory optimization
  - ✓ Operator fusion
- ✧ Scheduling
  - ✓ Auto-parallelization

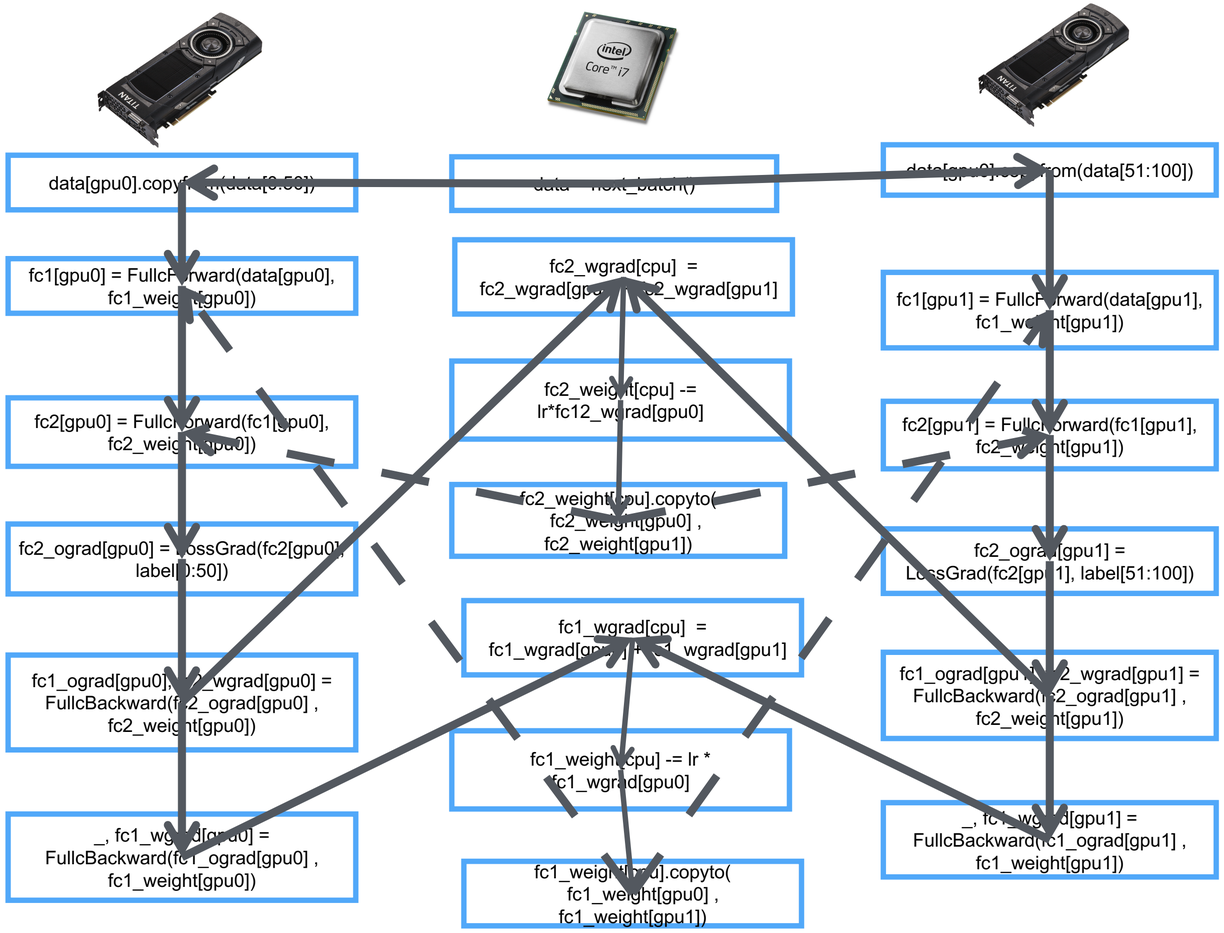
# In summary

- Symbolic
  - efficient & portable
  - but hard to use
- Gluon
  - imperative  for developing
  - symbolic for deploying
- Imperative
  - flexible
  - may be slow

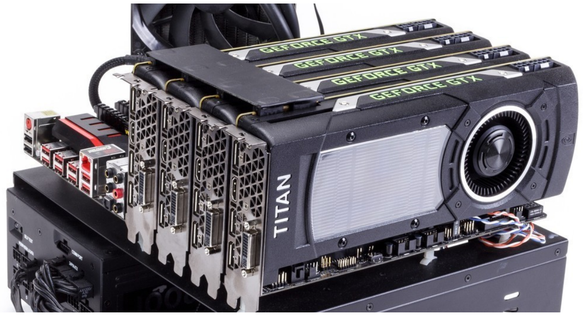



# Writing Parallel Programs is Painful

## Dependency graph for 2-layer neural networks with 2 GPUs




Each forward-backward-update involves O(num_layer), which is often 100—1,000, tensor computations and communications

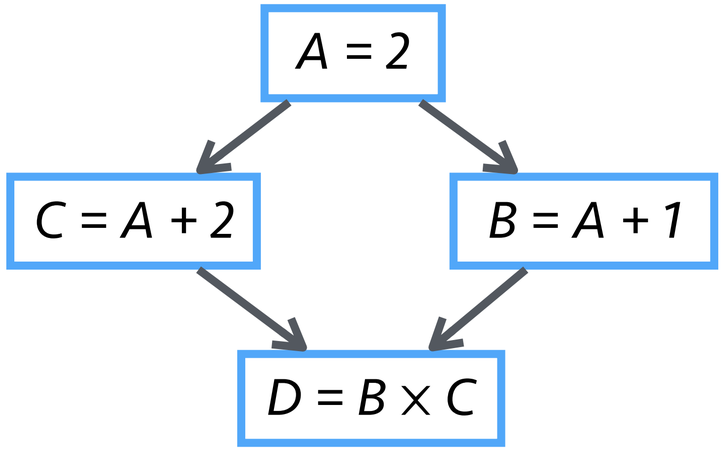


# Auto Parallelization

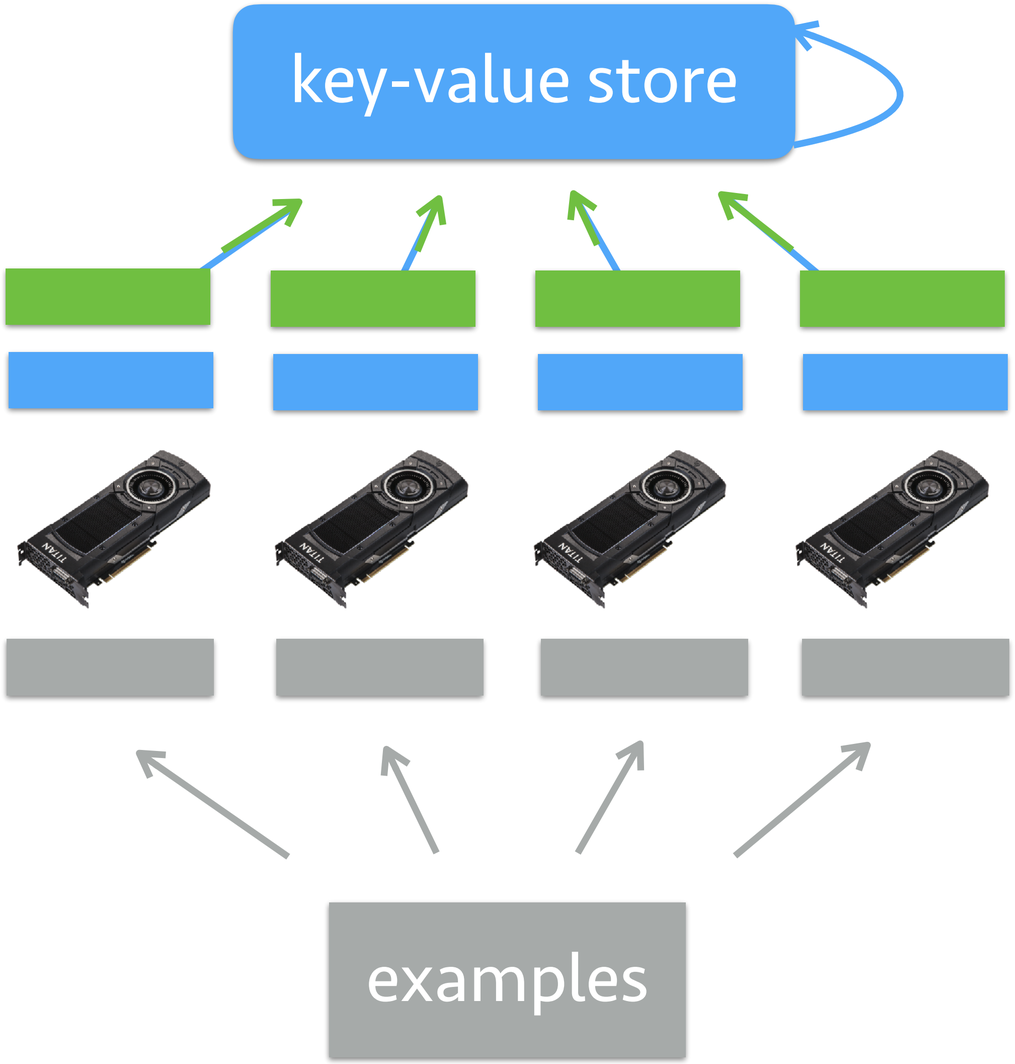
Write **serial** programs

```
>>> import mxnet as mx
>>> A = mx.nd.ones((2,2)) *2
>>> C = A + 2
>>> B = A + 1
>>> D = B * C
>>> D.wait_to_read()
```

Run in **parallel**



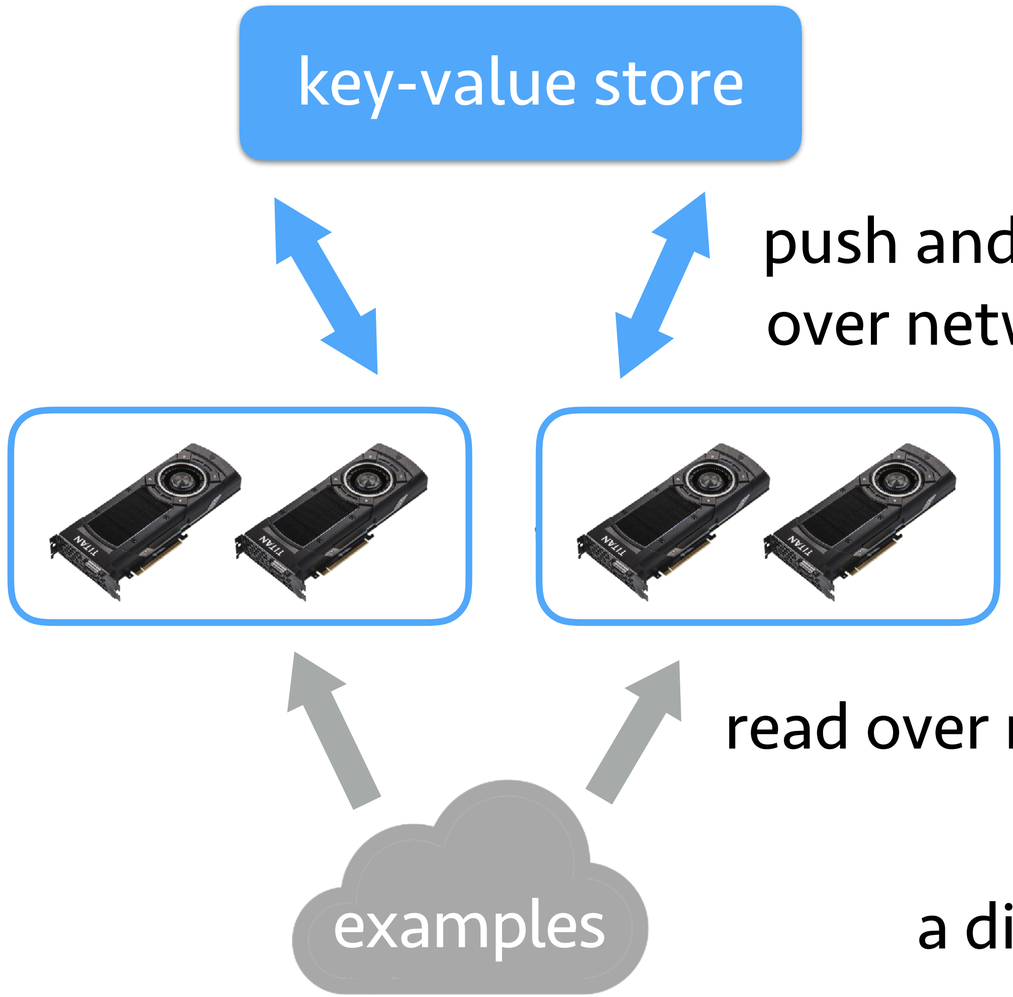
# Data Parallelism



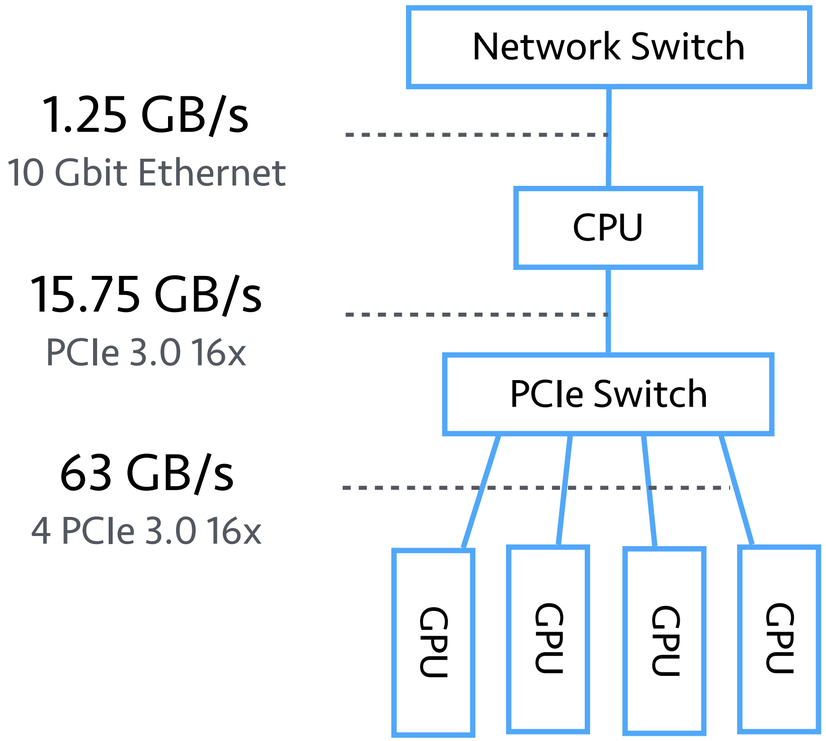

1. Read a data partition
2. Pull the parameters
3. Compute the gradient
4. Push the gradient
5. Update the parameters

# Distributed Computing

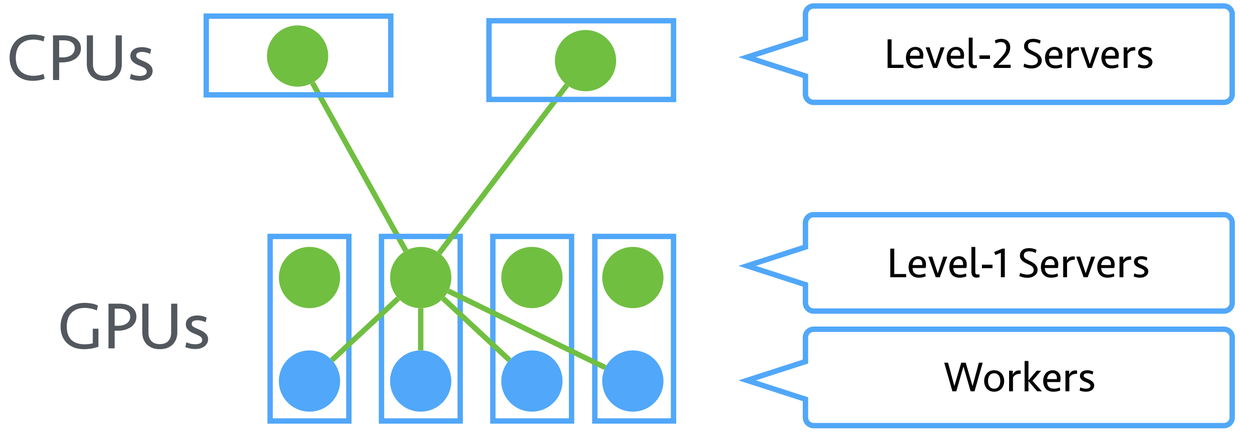
A user does not need to change the codes when using multiple machines



# Scale to Multiple GPU Machines



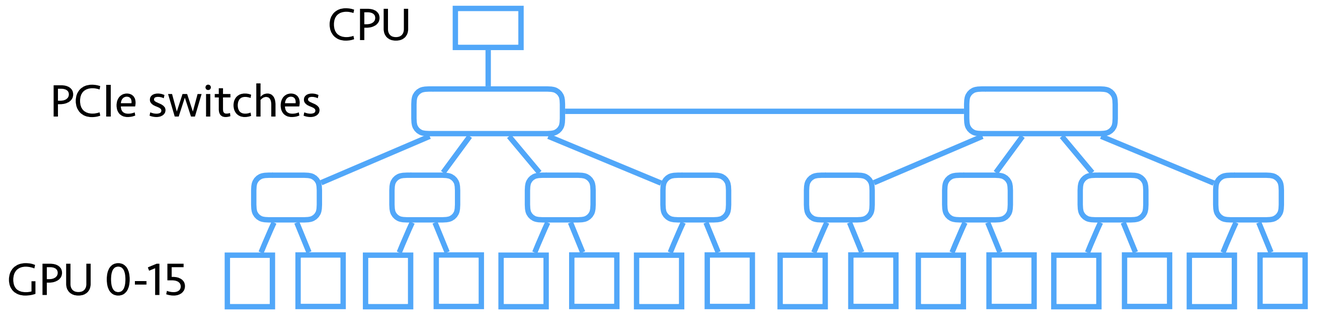

Hierarchical parameter server



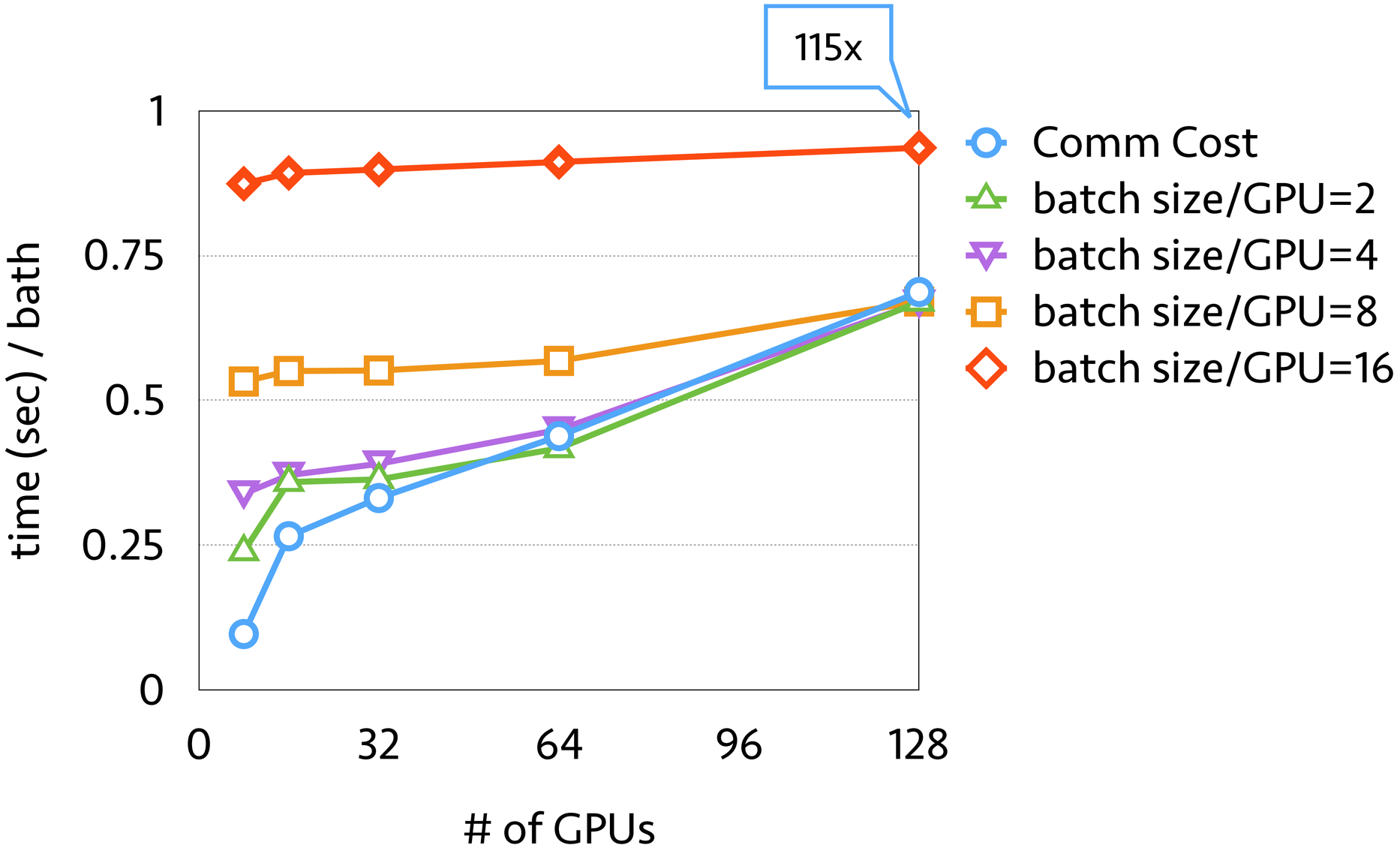
# Experiment Setup

- IMAGENET
  - ✓ 1.2 million images with 1000 classes
- Resnet 152-layer model
- EC2 P2.16xlarge

- Minibatch SGD
- Synchronized Updating

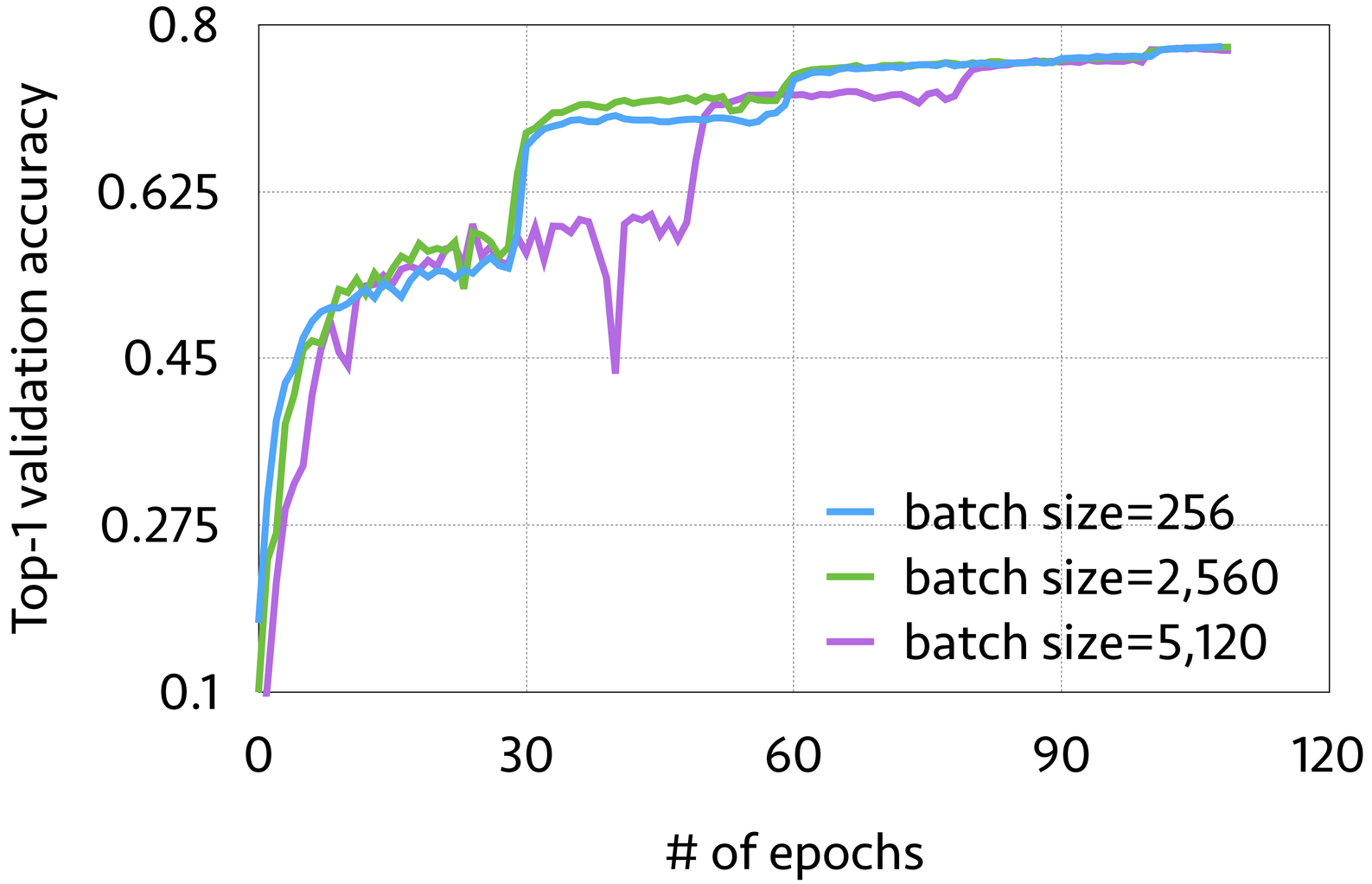


# Scalability over Multiple Machines

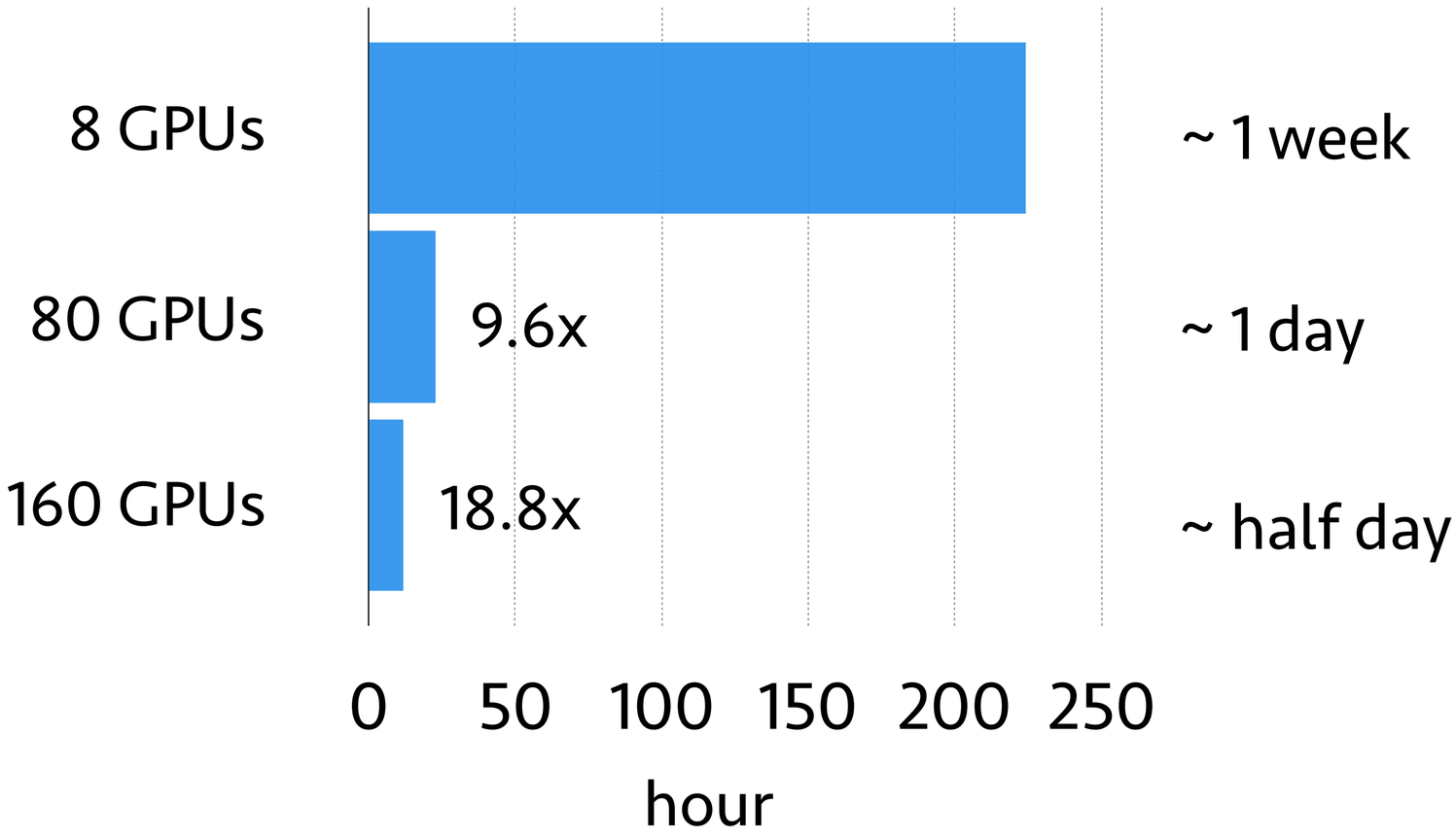


# Convergence

- Increase learning rate by 5x
- Increase learning rate by 10x, decrease it at epoch 50, 80



# Time to achieve 22.5% top-1 accuracy



# Outline

# Conclusion

## Tensors are the future of ML

- Tensor contractions: space savings in deep architectures.

- New primitives speed up tensor contractions: extended BLAS
- Tensor sketches compress efficiently while preserving information
- Tensor sketches enable multi-modal tasks such as visual Q&A