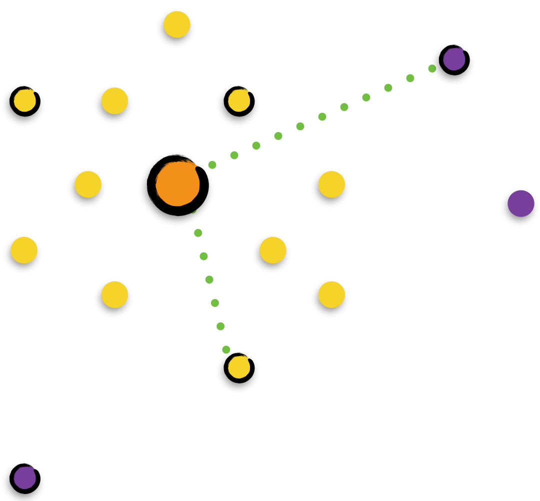


High-Performance Machine Learning Primitives

High Performance Computing Kernels in N-body Problems

Chenhan D. Yu

Sep 18, 2017
The 5th BLIS retreat!!!
Austin, TX

Hook 'em Horns

This year ...

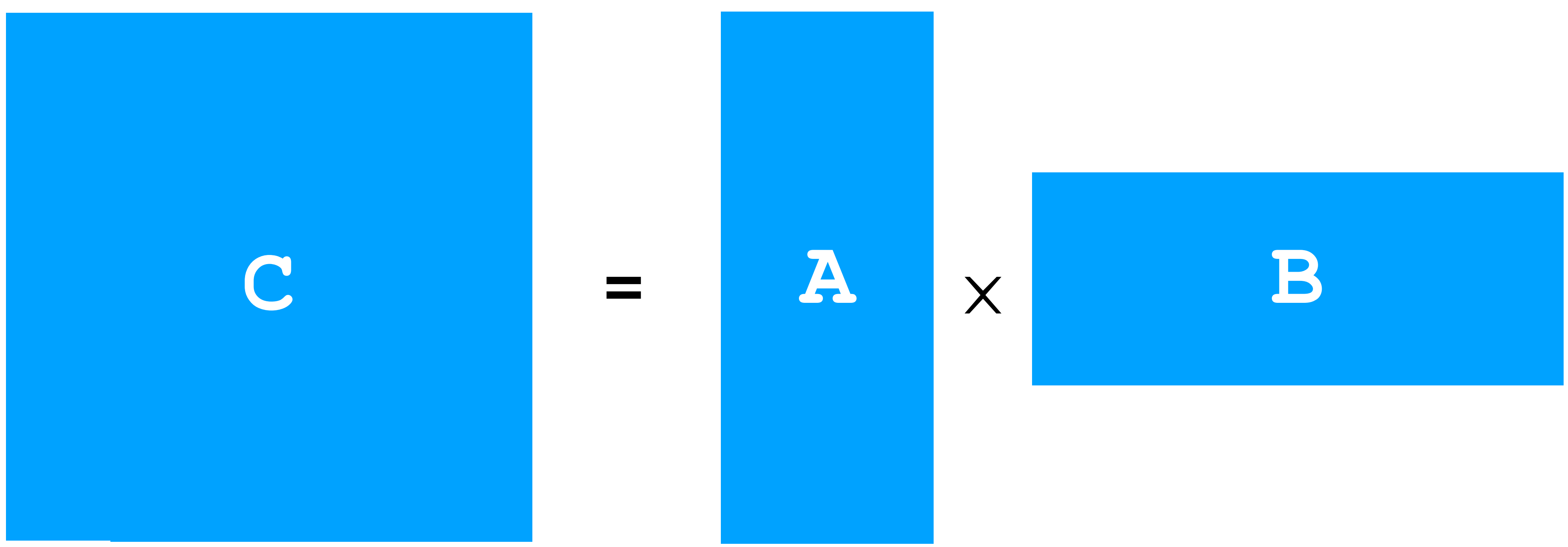
I am on the job market.

Both academia and industry positions are very welcome!

The Spirit of HMLP

O(2mnk) FLOPS, O(mn+mk+kn) MOPS,

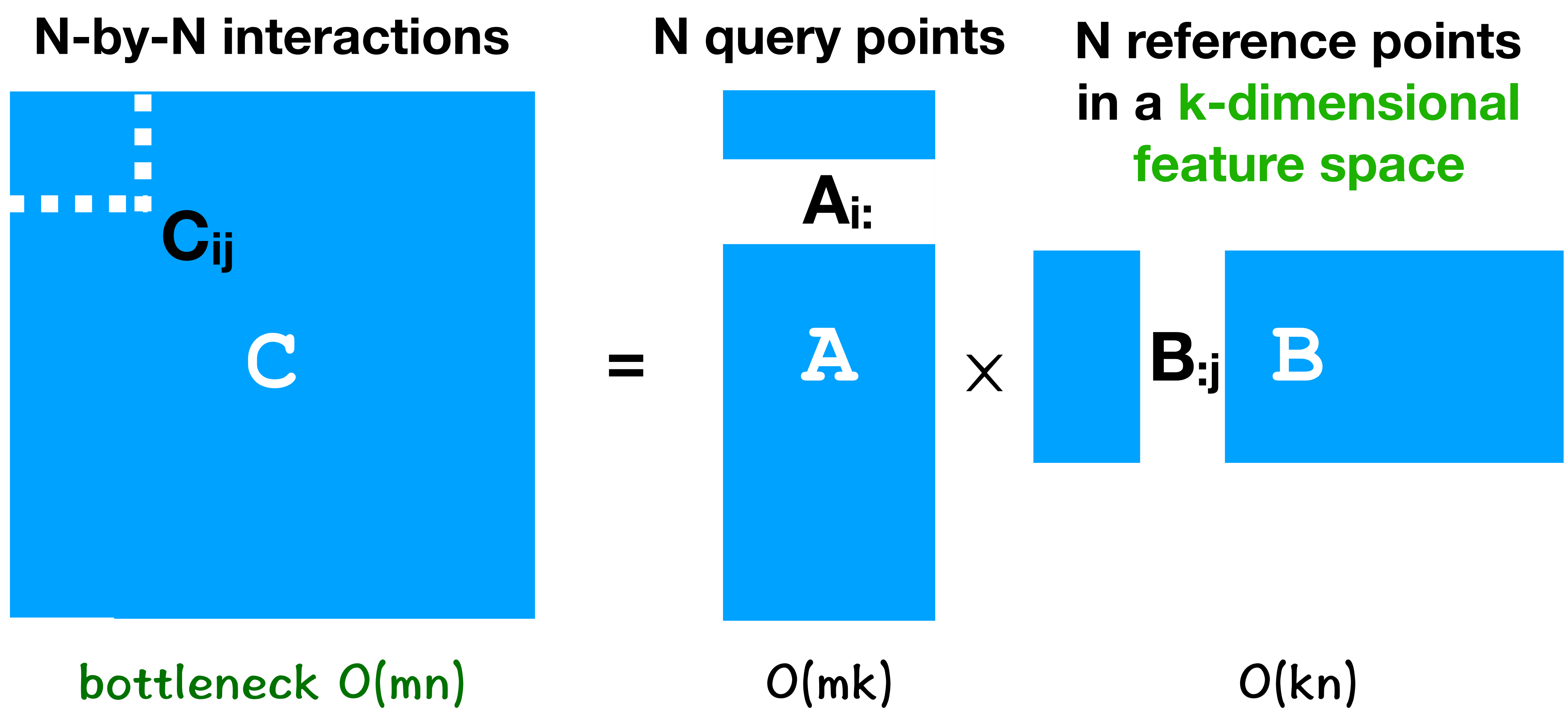
~ 95% PEAK, if k is large enough (k > 1*KC)



bottleneck O(mn)   O(mk)   O(kn)

N-body Operators [A. Gray, ’03]

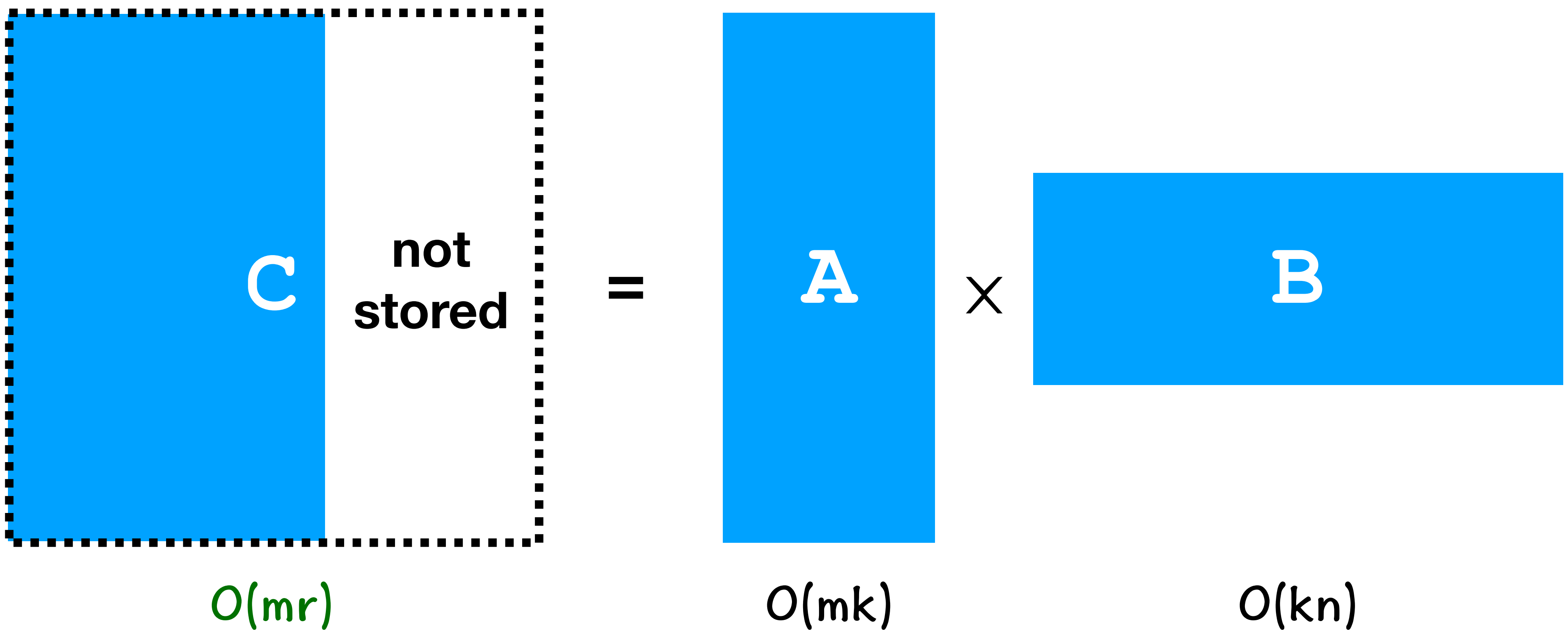
Describing the “interactions × “ between data points



Learning = Less Outputs

Instead, we need some kind of reduction of C.

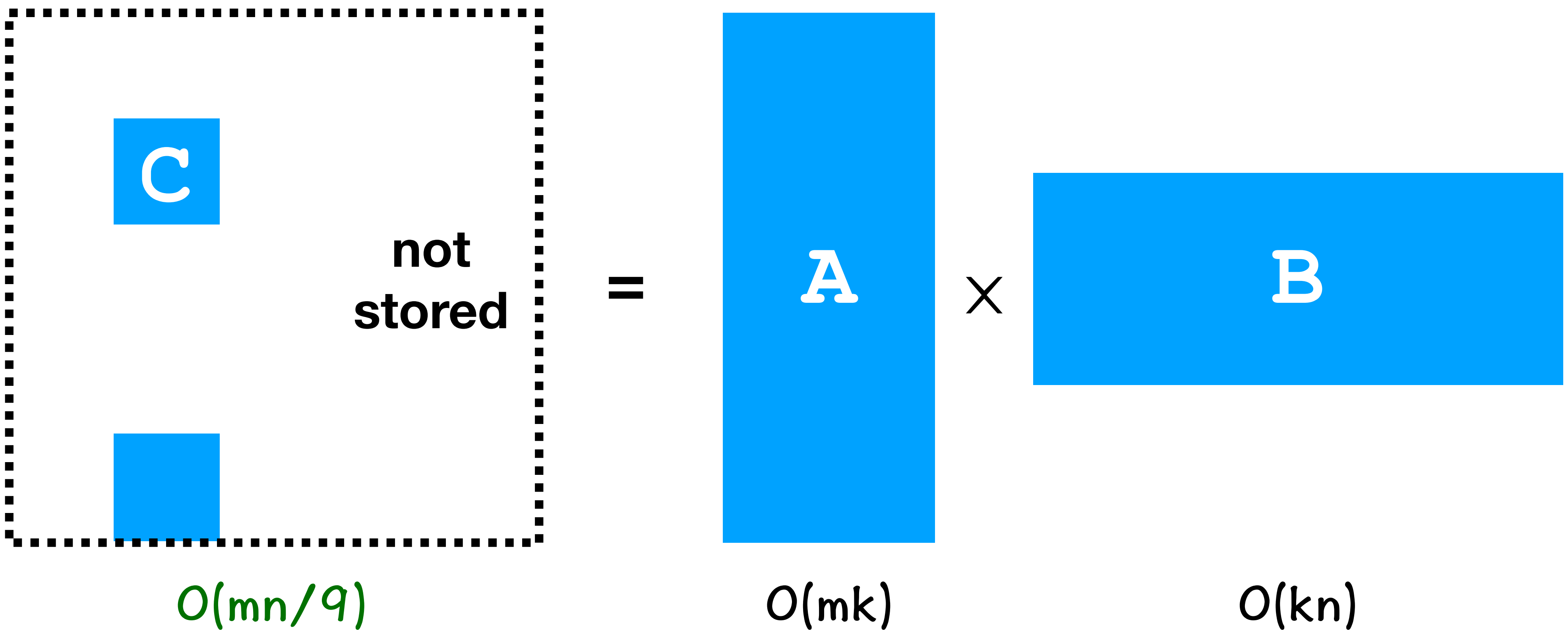
e.g. select r columns (nearest neighbors) [SC'15]



Spatial Reduction

Instead, we need some kind of reduction of C.

e.g. pool each 3-by-3 block (convolution + pooling layer)



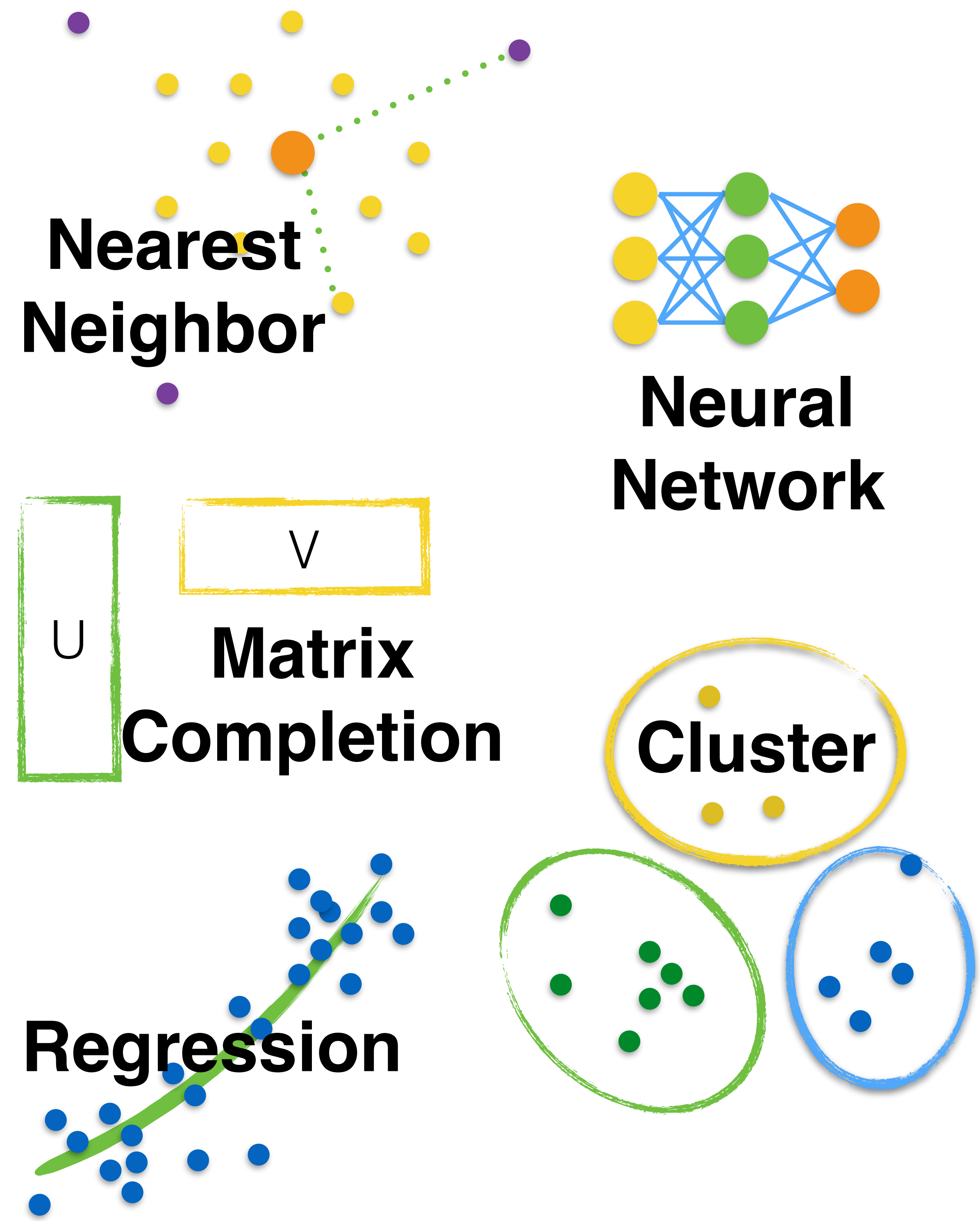
Significants

ML tasks

- Supervised Regression / Classification
- Clustering
- Dimensionality Reduction
- Neural Networks

Primitives

- Nearest Neighbors
- Kernel Methods
- K-mean Partitioning
- Matrix Compression PCA / CUR / Nystrom
- Convolutional Networks



Generalization

These memory reduction schemes require a more flexible interface than GEMM

$$\mathcal{C}_{ij} = \sum_k \mathcal{A}_{ik} \times \mathcal{B}_{kj}$$

$$\theta(\mathcal{C})_{i:} = \bigodot_j \mathcal{K}\Big(\bigoplus_k \alpha(\mathcal{A})_{ik} \bigotimes \beta(\mathcal{B})_{kj}\Big)$$

GEMM-like generalization [GraphBLAS]

N-body Computation Primitives

<typename TC> <typename TA> <typename TB>

$$\theta(\mathcal{C})_{i:} = \bigodot_{j} \mathcal{K}\Big(\bigoplus_{k} \alpha(\mathcal{A})_{ik} \bigotimes \beta(\mathcal{B})_{kj}\Big)$$

packing routines to extract (reorder) A, B, C into matrices

semi-ring operators <TA, TB> -> <TV>

kernel function <TV> -> <TC>

reduction operator <TC, TC> -> <TC>

BLIS (Framework + Kernel)

<typename TC> <typename TA> <typename TB>

$$\theta(\mathcal{C})_{i:} = \bigodot_{j} \mathcal{K}\Big(\bigoplus_{k} \alpha(\mathcal{A})_{ik} \bigotimes \beta(\mathcal{B})_{kj}\Big)$$

packing routines to extract (reorder) A, B, C into matrices

microkernel

semi-ring operators <TA, TB> -> <TV>

kernel function <TV> -> <TC>

reduction operator <TC, TC> -> <TC>

Preserve the BLIS structure (the Goto algorithm)

Worry About Optimization?

<typename TC> <typename TA> <typename TB>

$$\theta(\mathcal{C})_{i:} = \bigodot_{j} \mathcal{K}\Big(\bigoplus_{k} \alpha(\mathcal{A})_{ik} \bigotimes \beta(\mathcal{B})_{kj}\Big)$$

`packing routines` to extract (reorder) A, B, C into matrices

Reduce storage and slow memory complexity by O(mk+kn)

BLIS microkernel

semi-ring operators <TA, TB> -> <TV>

kernel function <TV> -> <TC>

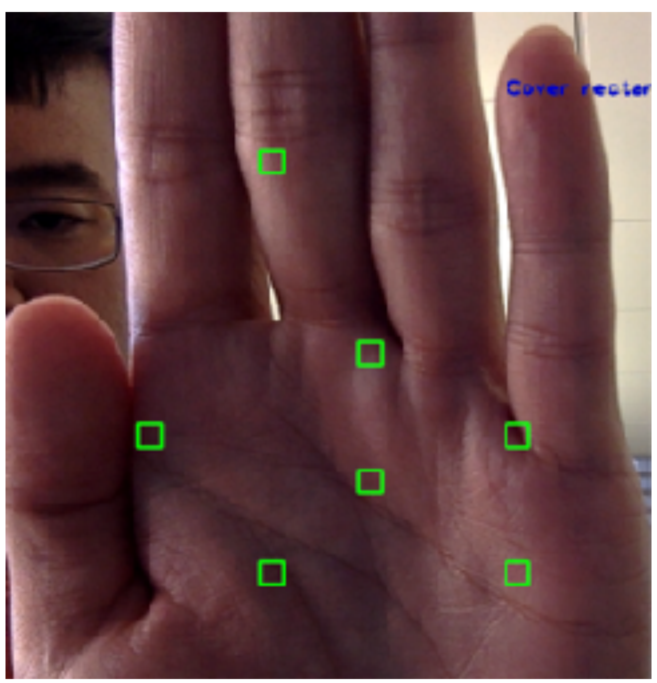
Reuse registers C

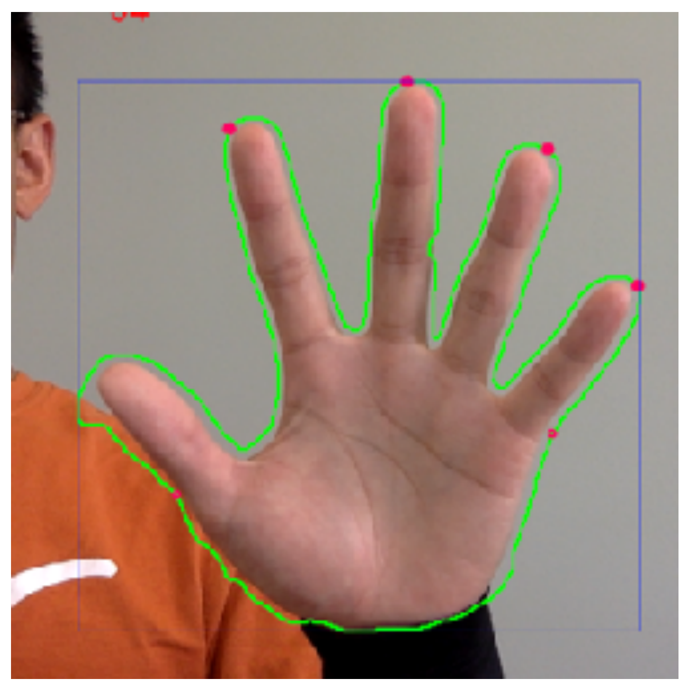
reduction operator <TC, TC> -> <TC>

Reduce storage and slow memory complexity by O(mn)

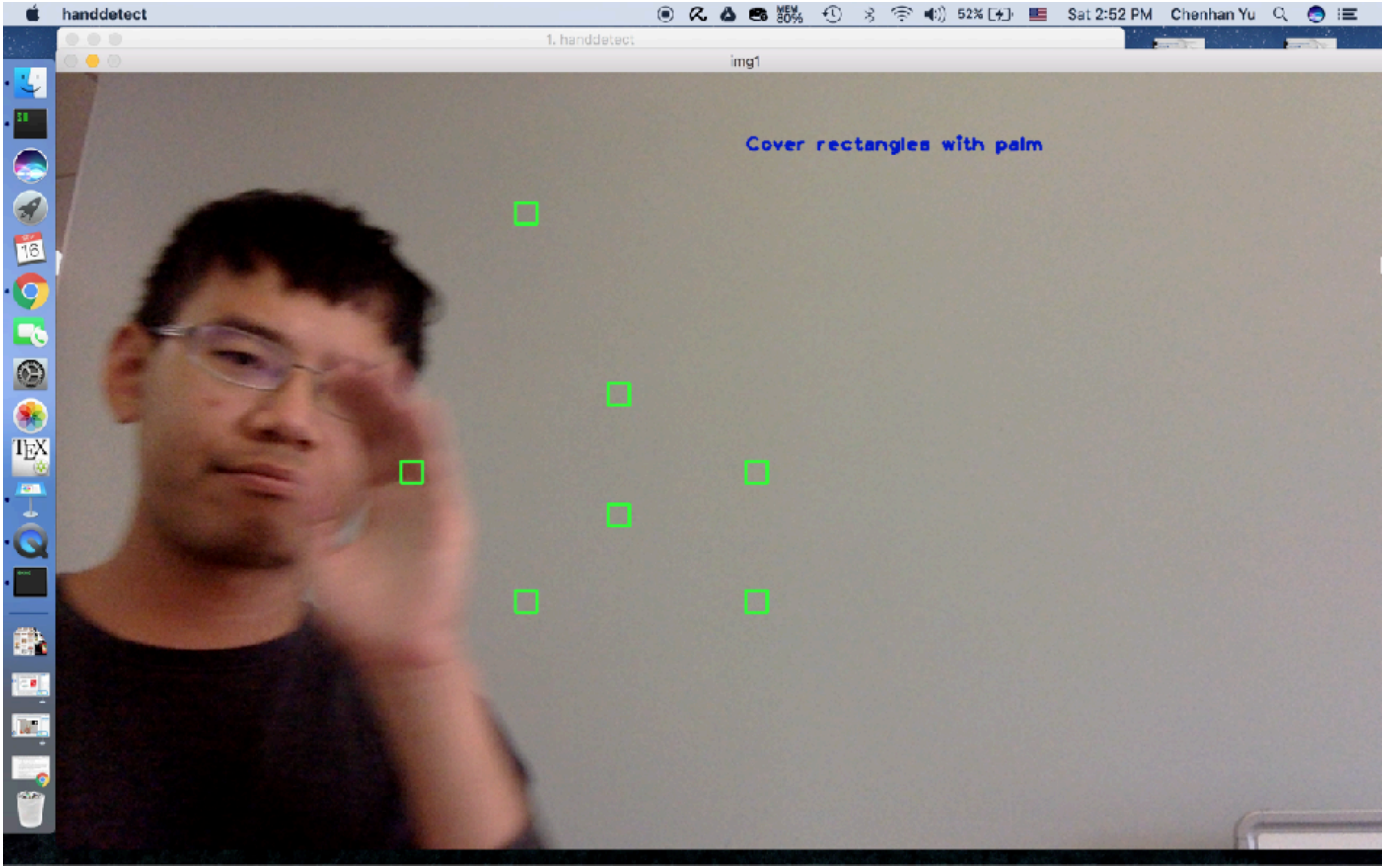
Reduce loads/stores from O(mc*nc) to O(mc)

Gesture Recognition

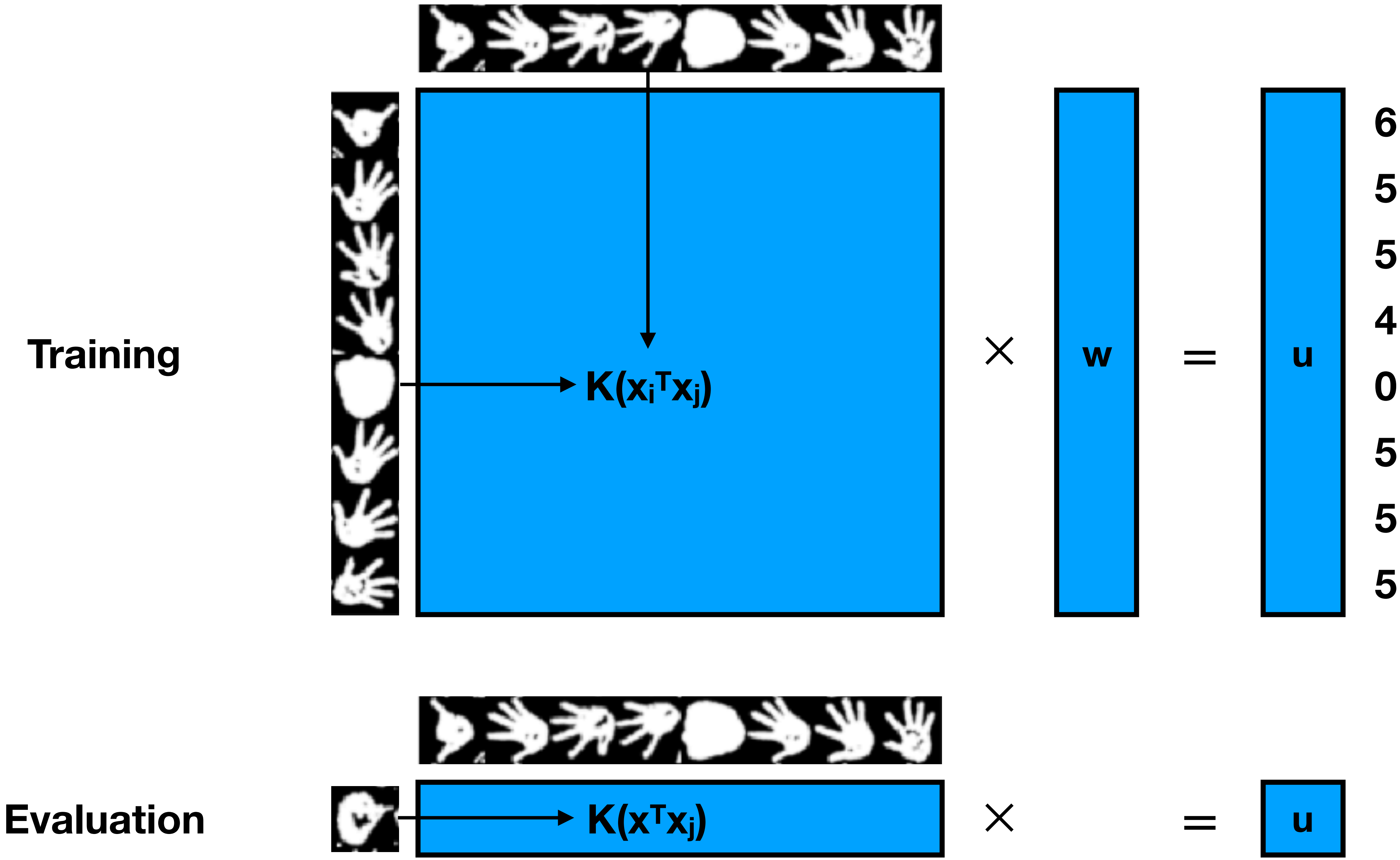




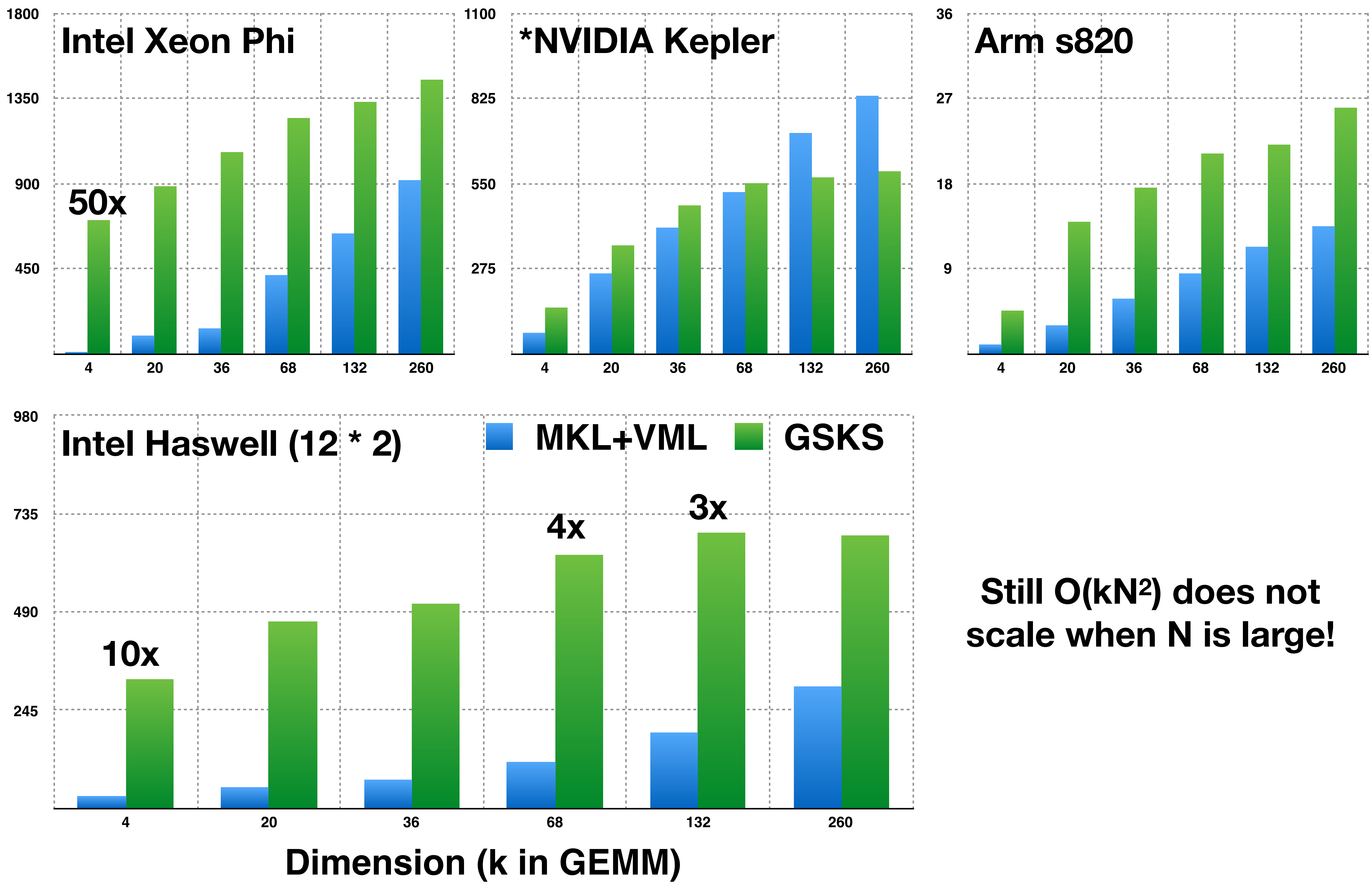
Classification



Kernel Density Estimation

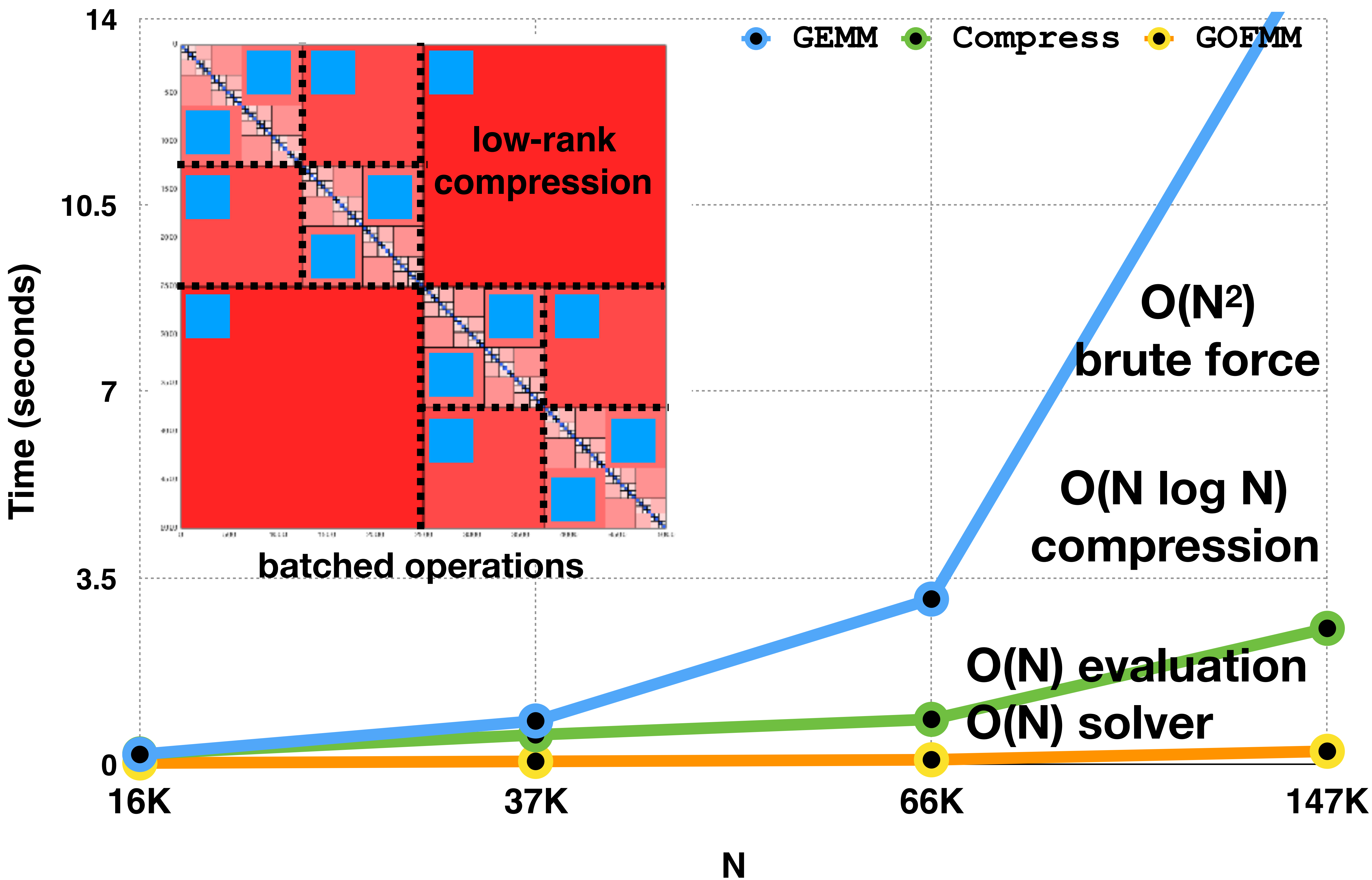


Portable Performance*



Still O(kN²) does not scale when N is large!

Approximation [SC'15,'17, KDD'15, IPDPS'15-'17, SISC]

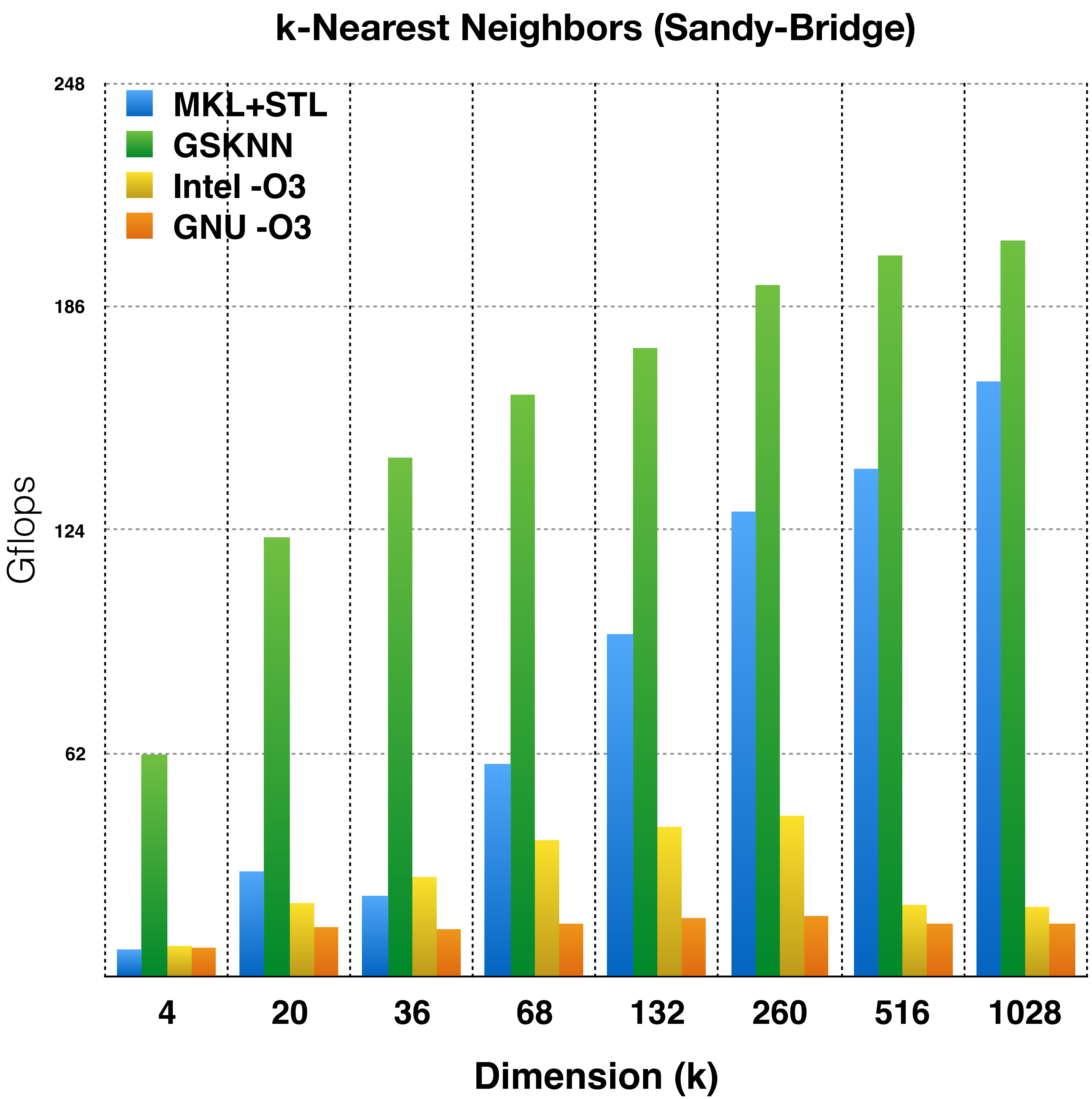


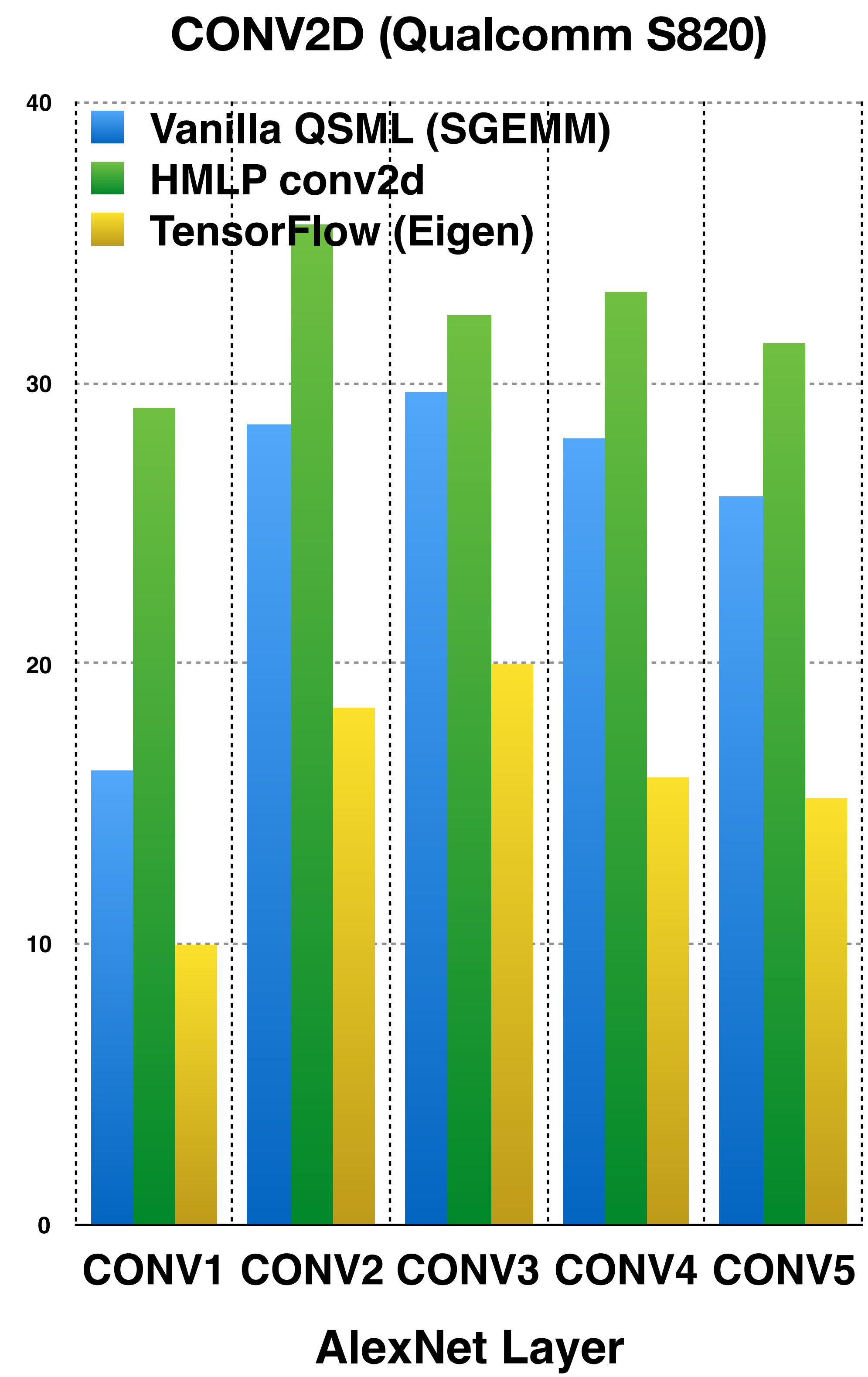
The Largest Problem?

For example, I systematically discover low-rank and sparse matrix structures such that I can invert a 32M-by-32M kernel matrix in 10 seconds but not 3 years.

*Note: Direct **MATVEC** on a 32Mx32M matrix takes 120 minutes using 3,072 Haswell cores. Cholesky factorization takes 2.8 years to complete.

More Primitives





Thank You!