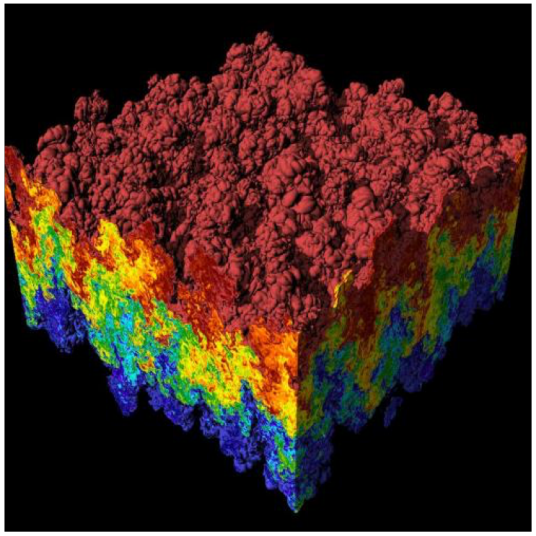
Enabling On-the-fly Storage Format Prediction and Optimization for SpMV

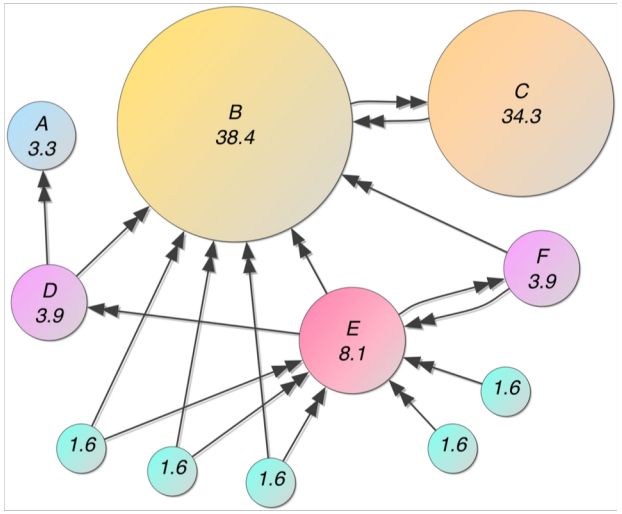
Weijie Zhou, Xipeng Shen
CSC, North Carolina State University

Sparse Matrix Multiplication

- Sparse matrix vector multiplication (SpMV) -- core of many HPC applications.



Scientific applications, e.g, PDE, solvers



Large scale graph algorithm, e.g., PageRank

Sparse Matrix Formats

A =

1	5	0	0
0	2	6	0
8	0	3	7
0	9	0	4

SpMV: **y** = **Ax**

Sparse Matrix Formats

A =

1	5	0	0
0	2	6	0
8	0	3	7
0	9	0	4

SpMV: **y** = **Ax**

CSR

ptr = | 0 | 2 | 4 | 7 |

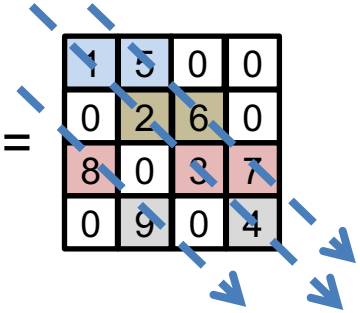
cols = | 0 | 1 | 1 | 2 | 0 | 2 | 3 | 1 | 3 |

data = | 1 | 5 | 2 | 6 | 8 | 3 | 7 | 9 | 4 |

```
for(i = 0; i < m; ++i) {
    for(j = ptr[i]; j < ptr[i+1]; ++j) {
        y[i] += data[j] * x[cols[j]];
    }
}
```

Sparse Matrix Formats

A =



SpMV: **y** = **Ax**

DIA

offsets =

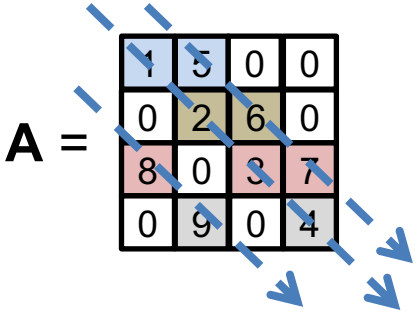
-2	0	1

data =

0	0	8	9
1	2	3	4
5	6	7	0

```
for(d = 0; d < ndiags; ++d) {
    k = offsets[d];
    istart = max(0, -k); jstart = max(0, k);
    L = min(m-istart, n-jstart);
    for(i = 0; i < L; ++i) {
        y[istart+i] +=
            data[istart+d*MAXDIA+i] * x[jstart+i];
    }
}
```

Sparse Matrix Formats



SpMV: **y** = **Ax**

Storage formats

- CSR
- COO
- DIA
- ELL
- HYB
- …

DIA

offsets = | -2 | 0 | 1 |

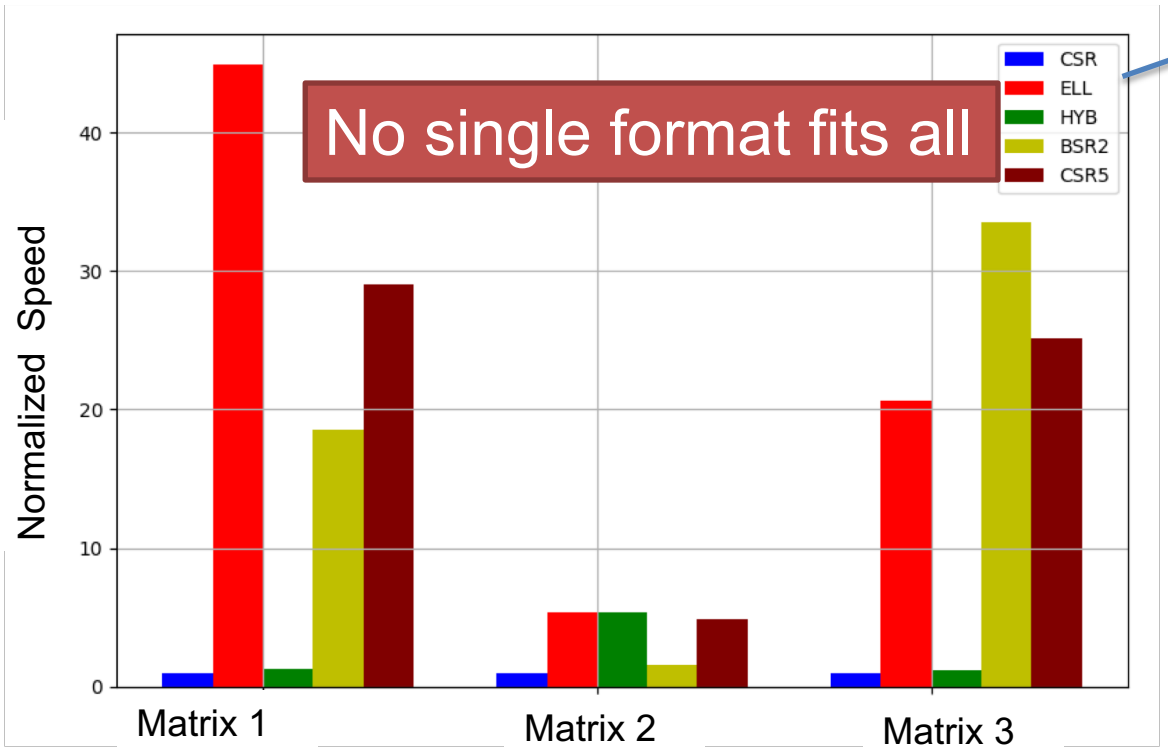
data =

0	0	8	9
1	2	3	4
5	6	7	0

```
for(d = 0; d < ndiags; ++d) {
    k = offsets[d];
    istart = max(0, -k); jstart = max(0, k);
    L = min(m-istart, n-jstart);
    for(i = 0; i < L; ++i) {
        y[istart+i] +=
            data[istart+d*MAXDIA+i] * x[jstart+i];
    }
}
```

Format Selection for SpMV

- Formats may give significant different performance
- How to select the best format for a given matrix?



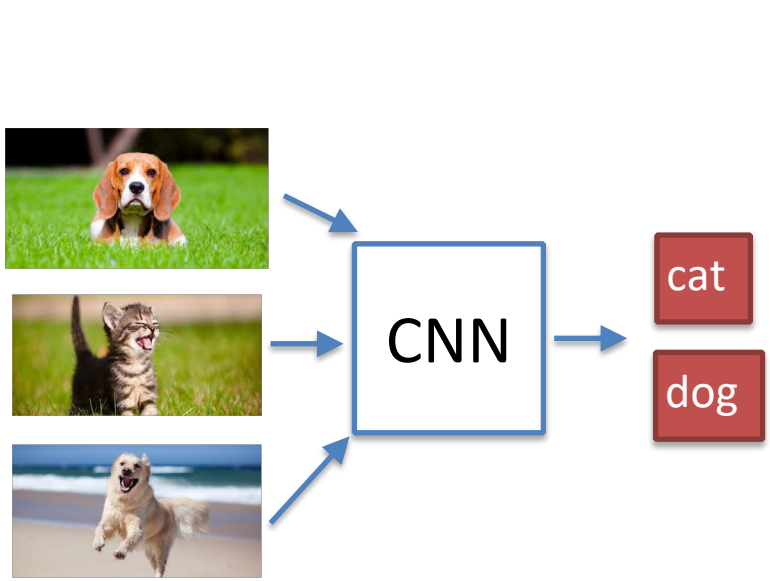
Previous work

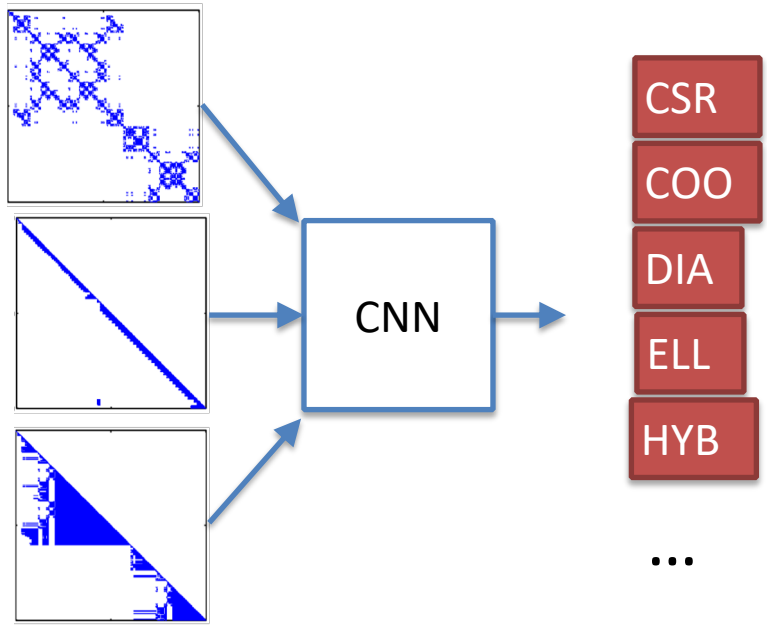
- Jiajia Li+:PLDI ’13, decision tree
- N. Sedaghati+:ICS’15, decision tree
- A. Benatia+:ICPP,16, SVM
- B. Yilmaz+:TACO’16, decision tree

Limitations: accuracy, manual feature design, …

Our Inspiration

- Treat matrix as an image, use *image recognition* methods for selection.



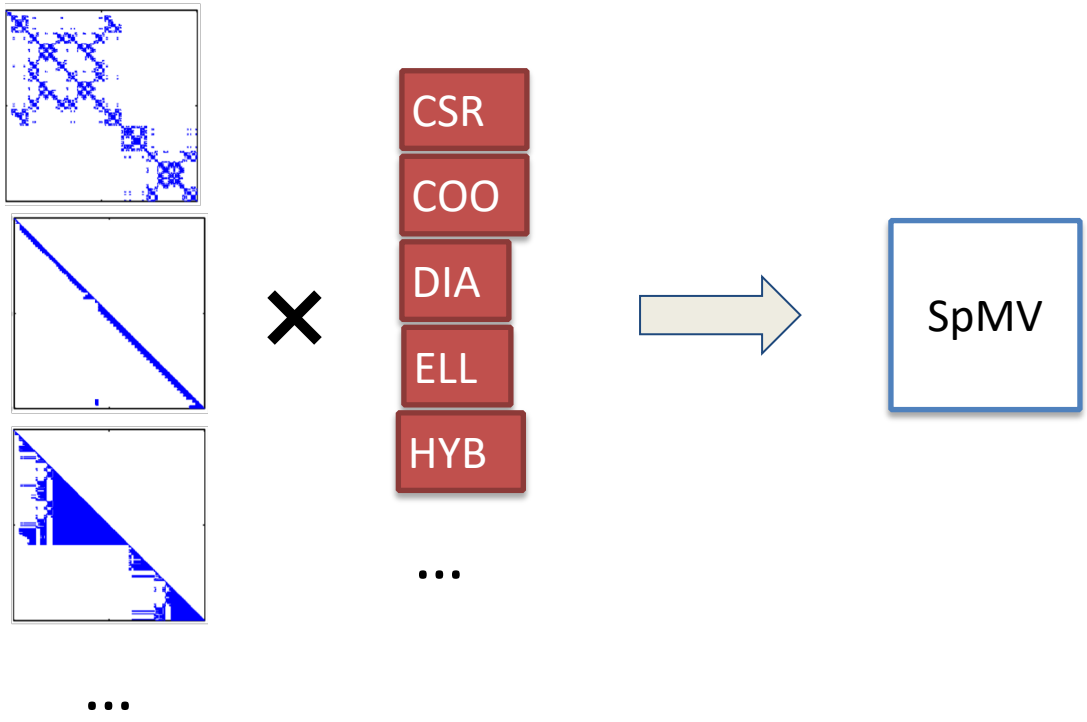


Our Contributions

- Bridging the gap between deep learning and format selection
- Three key questions
  - How to represent sparse matrices for DNN?
  - What deep learning structure to use?
  - How to address the architecture sensitivity?

Collecting Labels For Training

Run SpMV on the combination of matrix and format



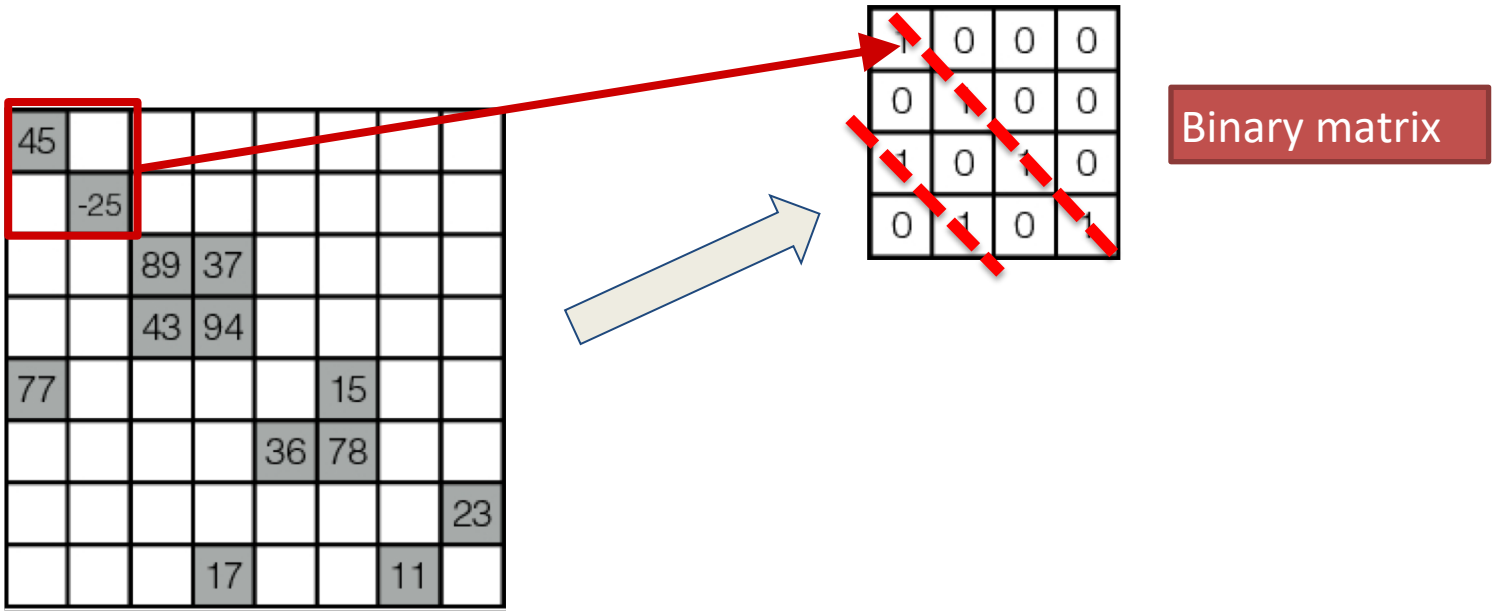
Label = ***argmax*** Performance(format)

Special Challenge I

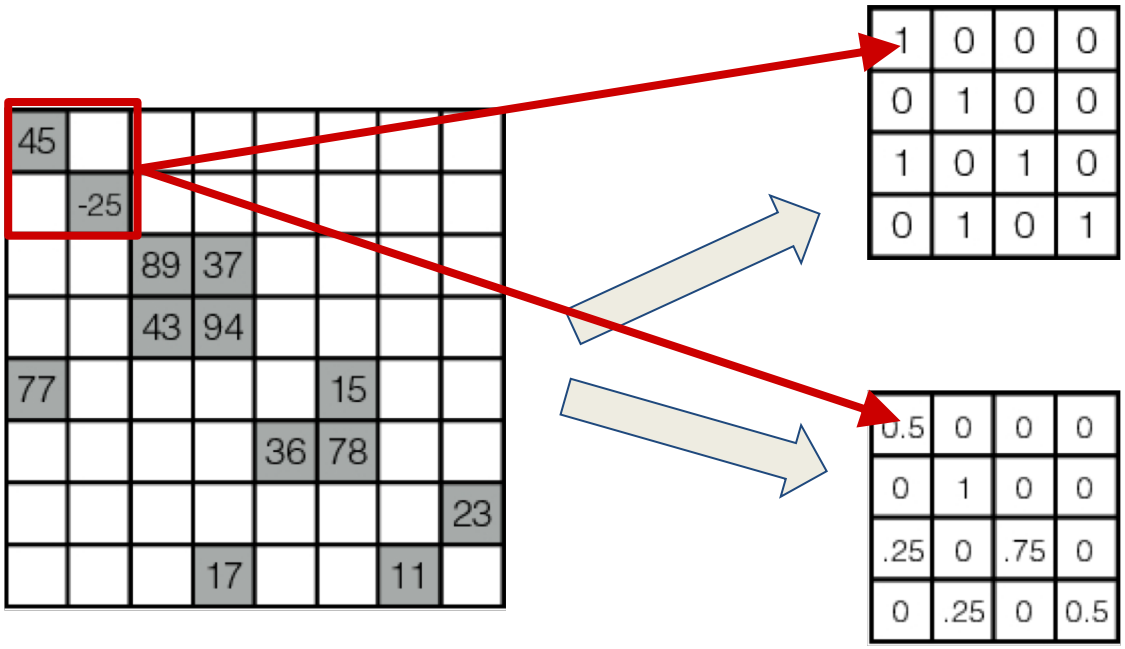
- **Input representation**: fixed size required

Special Challenge I

- Method in image processing: Image scaling

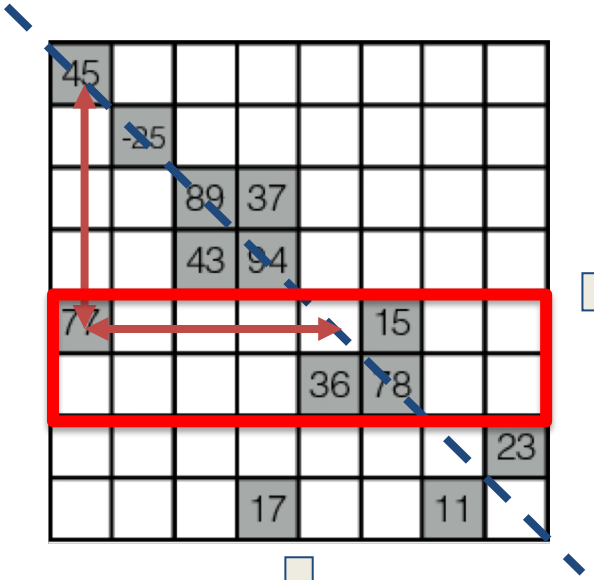


Proposal I: Augmented with Density Matrix



Proposal II: Distance histogram

- Dist. hist. between nonzeros and the diagonal
- Two representations: row-wise, col-wise



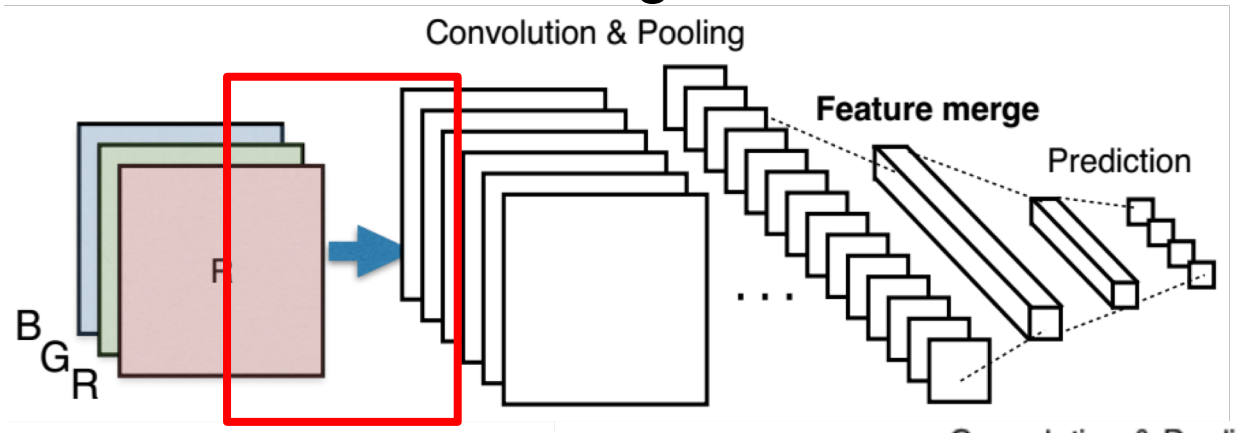
2	0	0	0
4	0	0	0
3	0	1	0
2	0	1	0

Row-wise

Bin = 2
One (bin) distant: 3 nz
Three (bin) distant: 1 nz

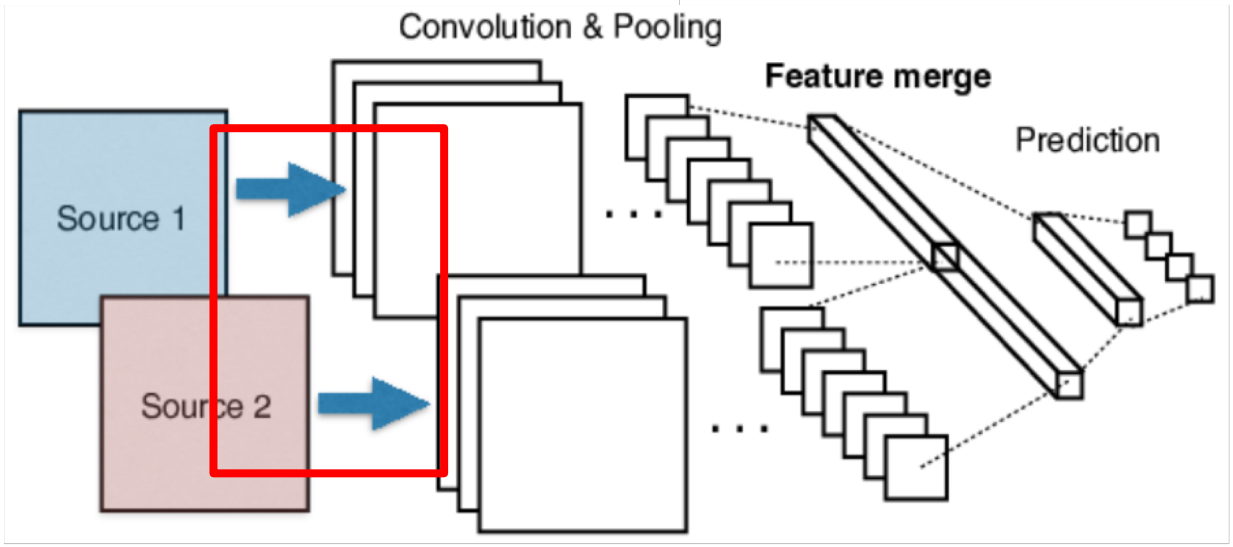
Special Challenge II

- CNN structure design



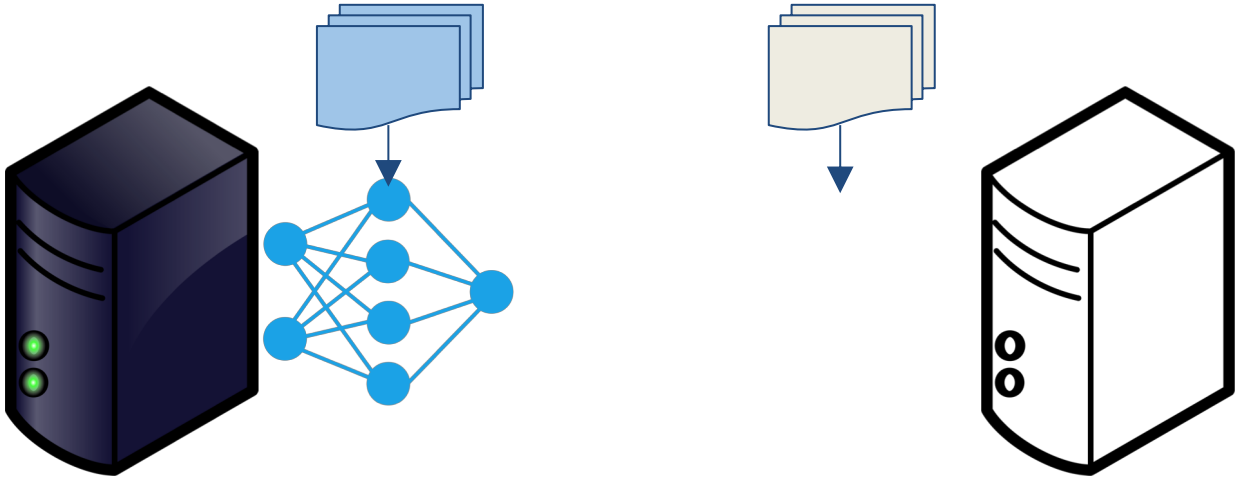
Early Merging

Late Merging



Special Challenge III

- Architecture Sensitivity
  - Best formats for a matrix differ across machines
  - Model cannot be reused across machines



Transfer Learning

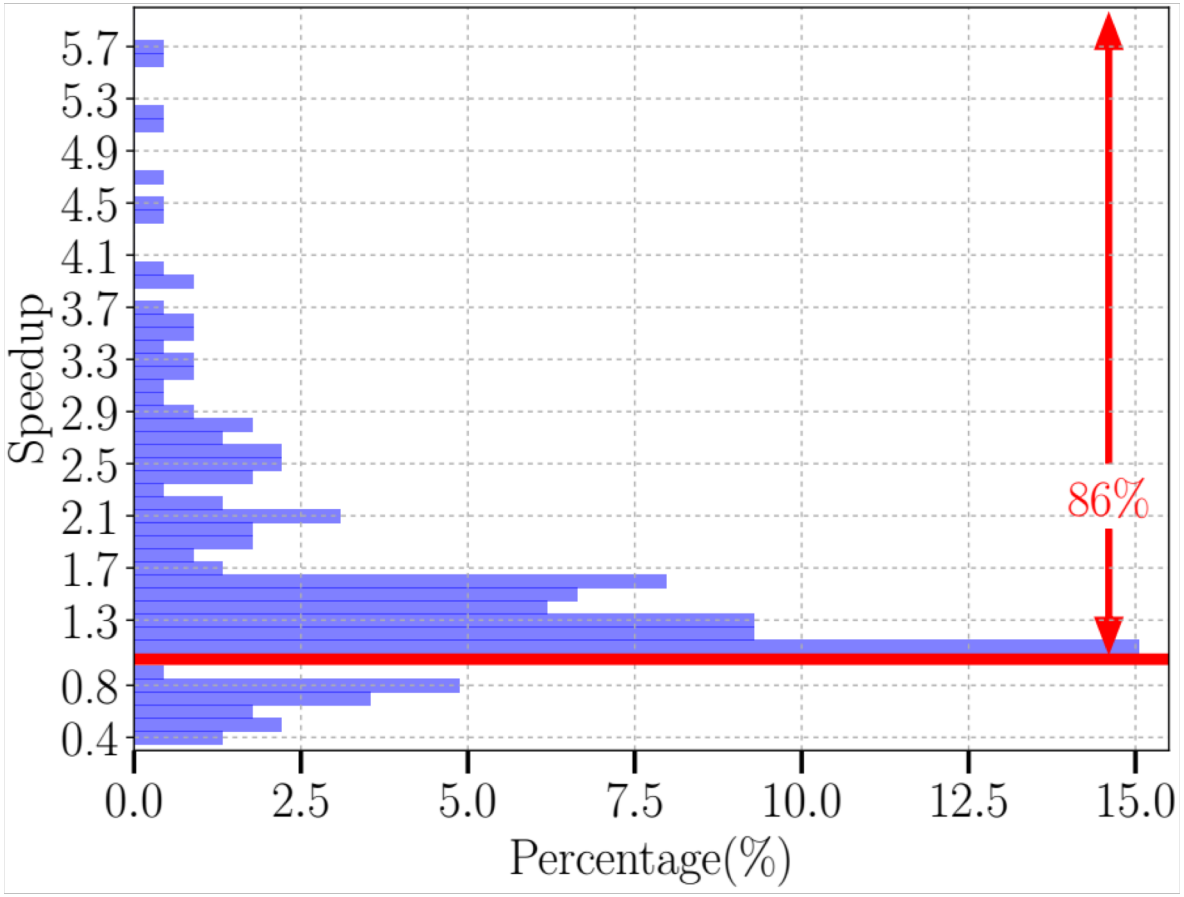
- An idea in ML for cross-domain model migration
  - Mostly across datasets in the domains
  - Train a model on one dataset in domain A
  - Refine the model on datasets in domain B
- *Questions* w.r.t the architecture sensitivity problem
  - How to effectively apply it?
  - How much help can it bring?

Evaluation -- Setup

- Dataset (9200 matrices)
  - The SuiteSparse Matrix Collection (2757 matrices)
  - Derived 6443 matrices
- 5-fold cross validation is used
- Evaluated formats
  - CSR, COO, ELL, HYB, BSR, CSR5
- Three platforms
  - Intel Xeon E5, AMD A8, Nvidia Titan X GPU

Speedup (Intel CPU)

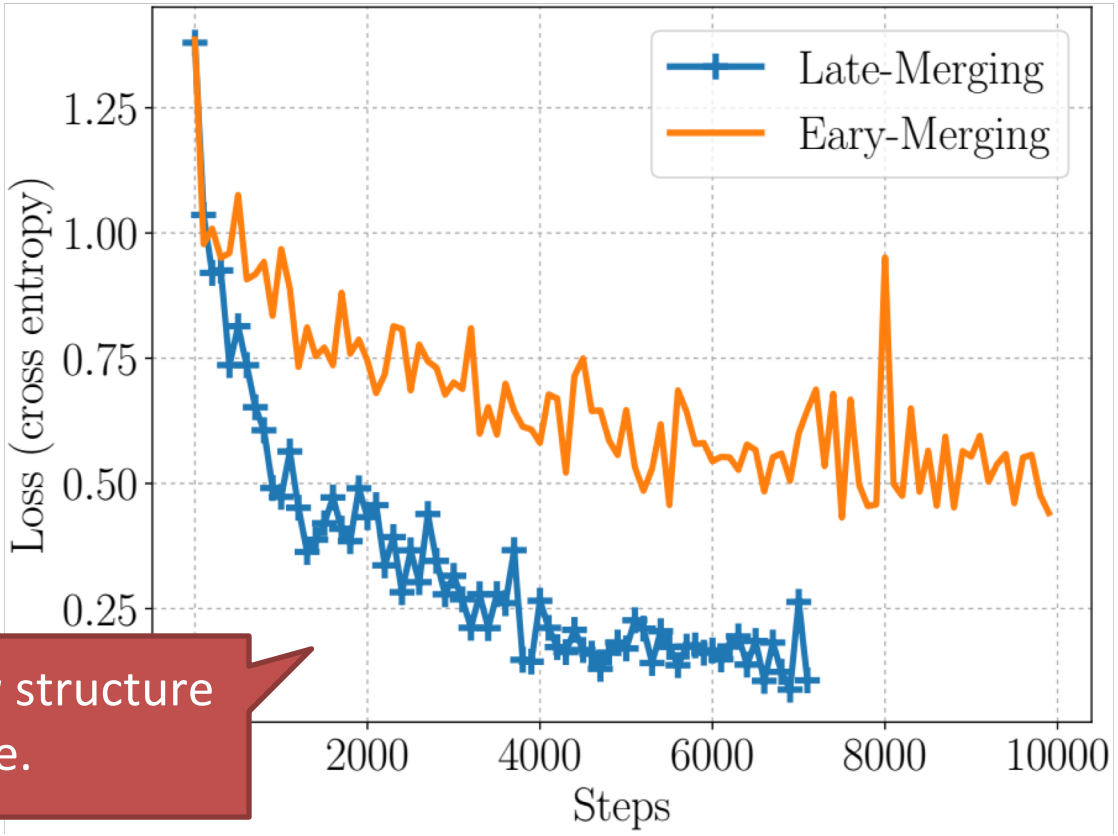
Baseline: SMAT [PLDI'13]



Average 1.73X over SMAT; 2.23X over all-CSR (max 15X)

CNN Structure Impact

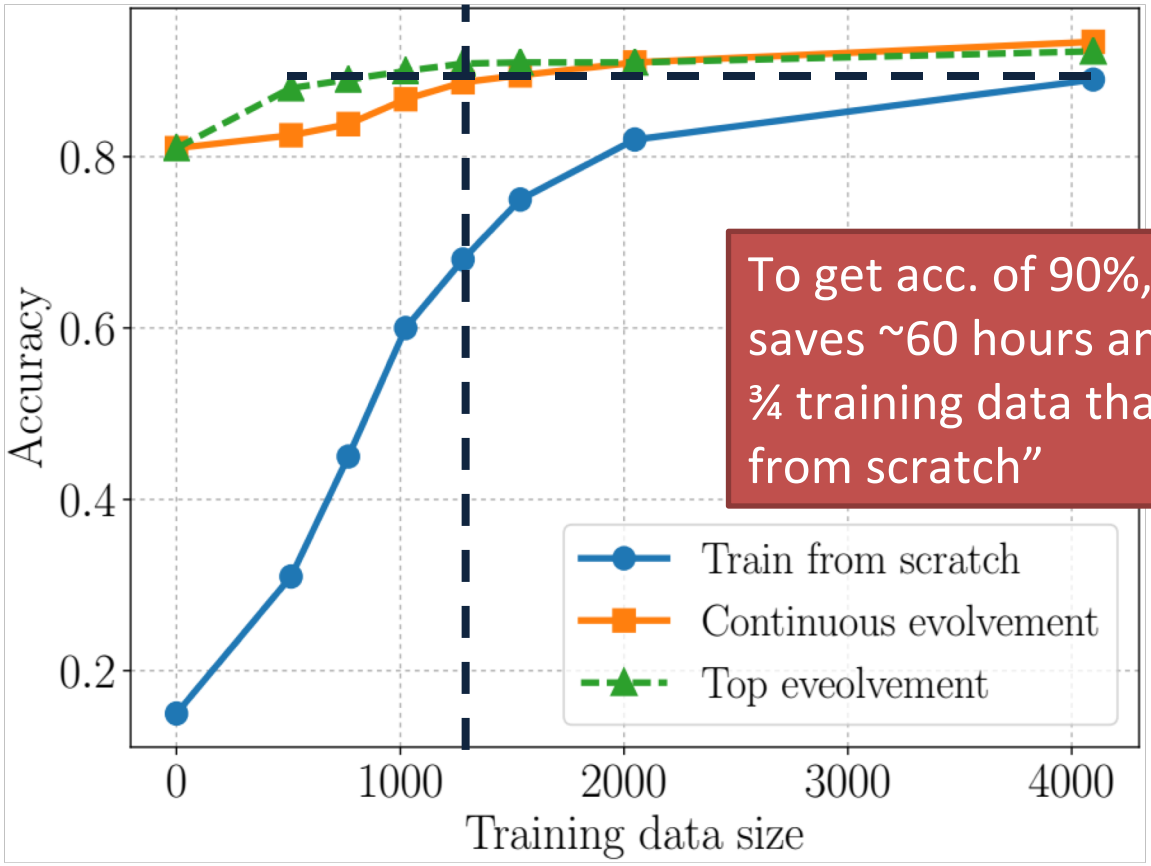
- *Early Merging* vs *Late Merging*



Late Merging structure is more stable.

Transfer Learning

- From Intel Xeon E5 to AMD Radeon A8



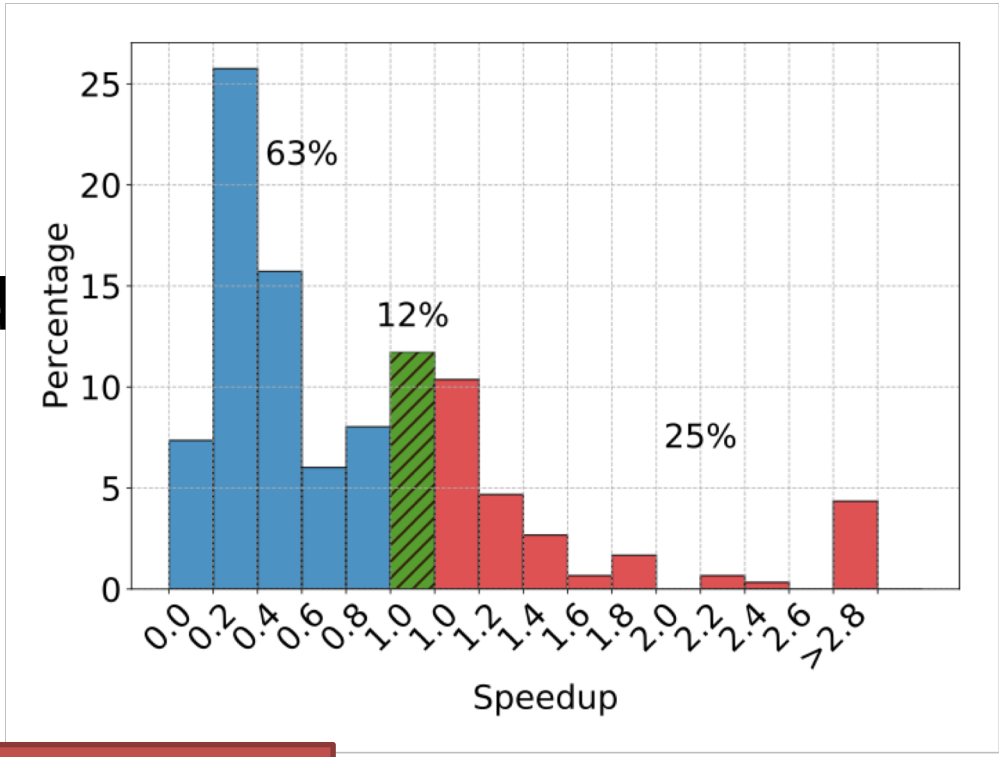
To get acc. of 90%, "Top Evo." saves ~60 hours and ¾ training data than "Train from scratch"

Discussion on Overhead

- Prediction and conversion overhead not considered
  - Focused on cases where SpMV runs repeatedly on a matrix for many times
- For other cases
  - Conversion overhead can outweigh the benefit or new format

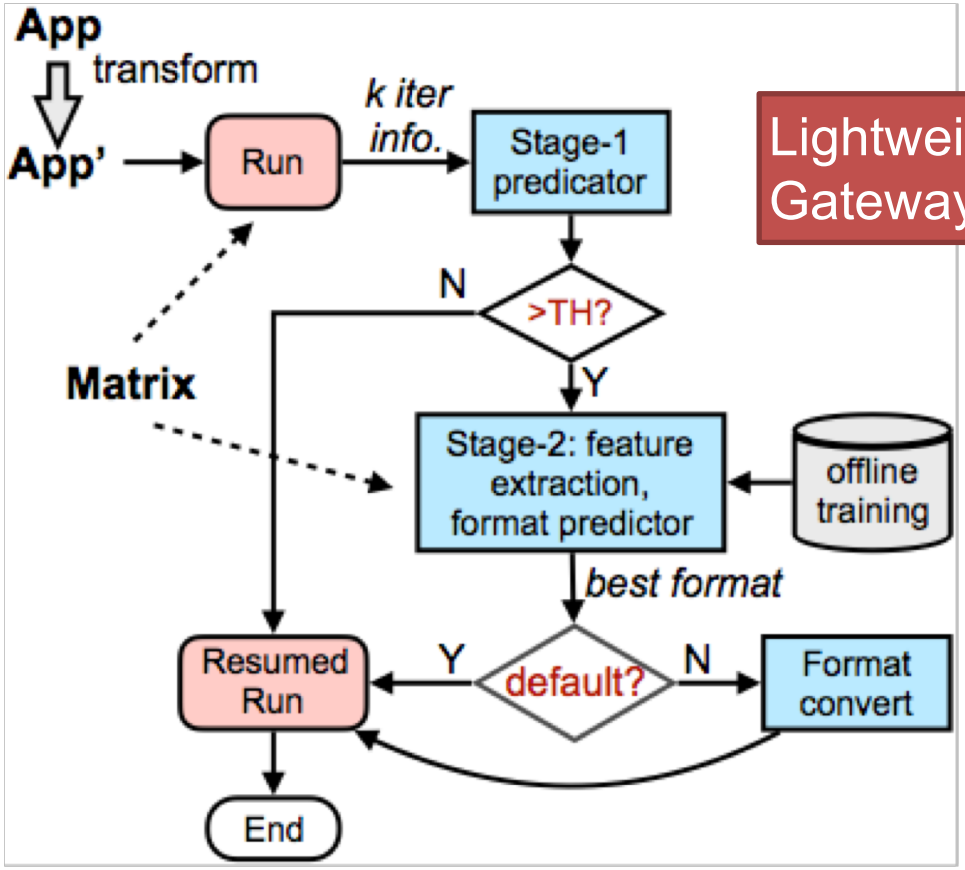
Impact of Conversion Time

- The conversion can be 50X of a SpMV operation
- Conversion overhead can outweigh the benefit of new format
- an overhead-conscious prediction model



Conversion time matters

Solution: Two-Stage Lazy-and-Light Scheme



Lightweight loop count predictor
Gateway for the second stage

More sophisticated predictions for best format

Summary: Overall speedup of applications: 1.14X to 1.43X vs. 0.82X to 1.24X upper-bound with overhead-oblivious

Final Takeaways

- Deep learning is effective for SpMV format selection
  - Important to treat the special challenges
- One step to relate deep learning with prog. optimizations
- Many potential uses to explore
- Considering conversion time is essential for sparse matrix format selection

Publication

- **[PPoPP'18]** *Bridging the Gap between Deep Learning and Sparse Matrix Format Selection*
- **[IPDPS'18]** *Overhead-Conscious Format Selection for SpMV-Based Applications*
- **[TPDS submitted]** *Enabling Runtime SpMV Format Selection through an Overhead Conscious Method*

Q&A