Towards Holistic Redundancy Exploitation for Data-centric ML Pipelines

Matthias Boehm

Technische Universität Berlin

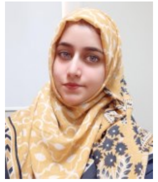
Berlin Institute for the Foundations of Learning and Data

Big Data Engineering (DAMS Lab)

Data-centric ML Pipelines

Key observation: SotA data engineering based on ML

Data Engineering



Top-K Cleaning Pipelines
[under submission]

Alignment of Multi-modal Data

I/O for Custom Data Formats
[under submission]

Parallel Feature Transformations
[**PVLDB'22**]

Data Integration & Data Cleaning

Data Programming & Augmentation

Model and Feature Selection

Hyper-parameter Tuning + CV

Hierarchical Composition as Library Functions on top of ML systems

Data Preparation
(e.g., one-hot, bins)

FX

```
train()
```

Model Training

X **y**

```
while(!converged) {
  … q = X %*% v …
}
```

```
predict()
```

Model Scoring

85%
Accuracy

SliceLine
[**SIGMOD'21c**]

Validation & Debugging

Deployment & Scoring

Apache SystemDS

[https://github.com/apache/systemds]

Apache SystemML™

07/2020 Renamed to SystemDS
05/2017 Apache Top-Level Project
11/2015 Apache Incubator Project
08/2015 Open Source Release

APIs: Command line, JMLC, Spark MLContext, Spark ML, (20+ Scalable Algorithms)

DML Scripts

Language

Compiler

Runtime

Write Once, Run Anywhere

[SIGMOD'15,'17,'19,'21abc,'23ab]
[PVLDB'14,'16ab,'18,'22]
[ICDE'11,'12,'15]
[CIDR'17,'20]
[VLDBJ'18]
[CIKM'22]
[DEBull'14]
[PPoPP'15]

In-Memory Single Node
(scale-up)

Hadoop or Spark Cluster
(scale-out)

Federated
(LA progs, PS)

In-Progress:

GPU

since 2014/16

Java

since 2012

hadoop

since 2010/11

Apache Spark

since 2015

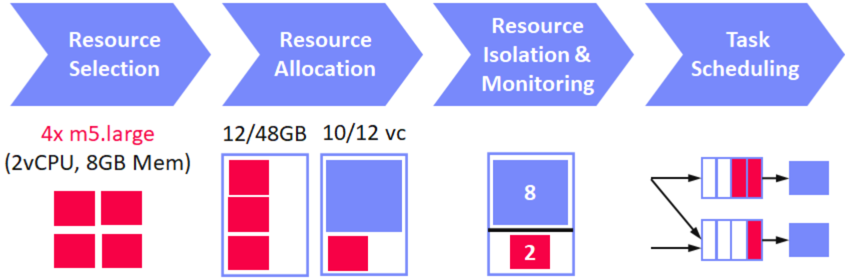
exdra

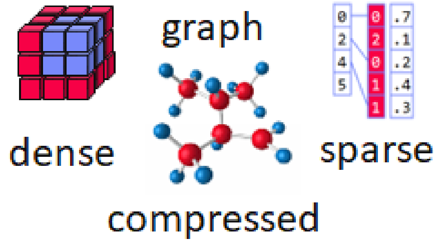
since 2019

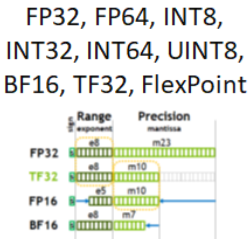
Redundancy-exploiting Techniques for data-centric ML Pipelines

Isolated Application, Exploration, and Tuning; Trial-and-Error Process

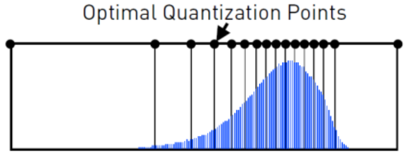
- **Resource Allocation and Elasticity**
- **Data Sampling and Composition**
  (sampling, distillation, augmentation-as-a-kernel, factorization)
- **Sparsity Exploitation**
  (algorithms, op pipelines, data/weights, kernels, HW)
- **Lossy and Lossless Compression**
- **Weight Pruning and Connection Sampling**



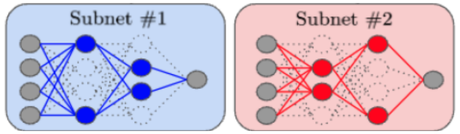




[Credit: NVIDIA A100 Whitepaper]



[**Credit:** Ce Zhang]



[**Credit:** Chris Jermaine]

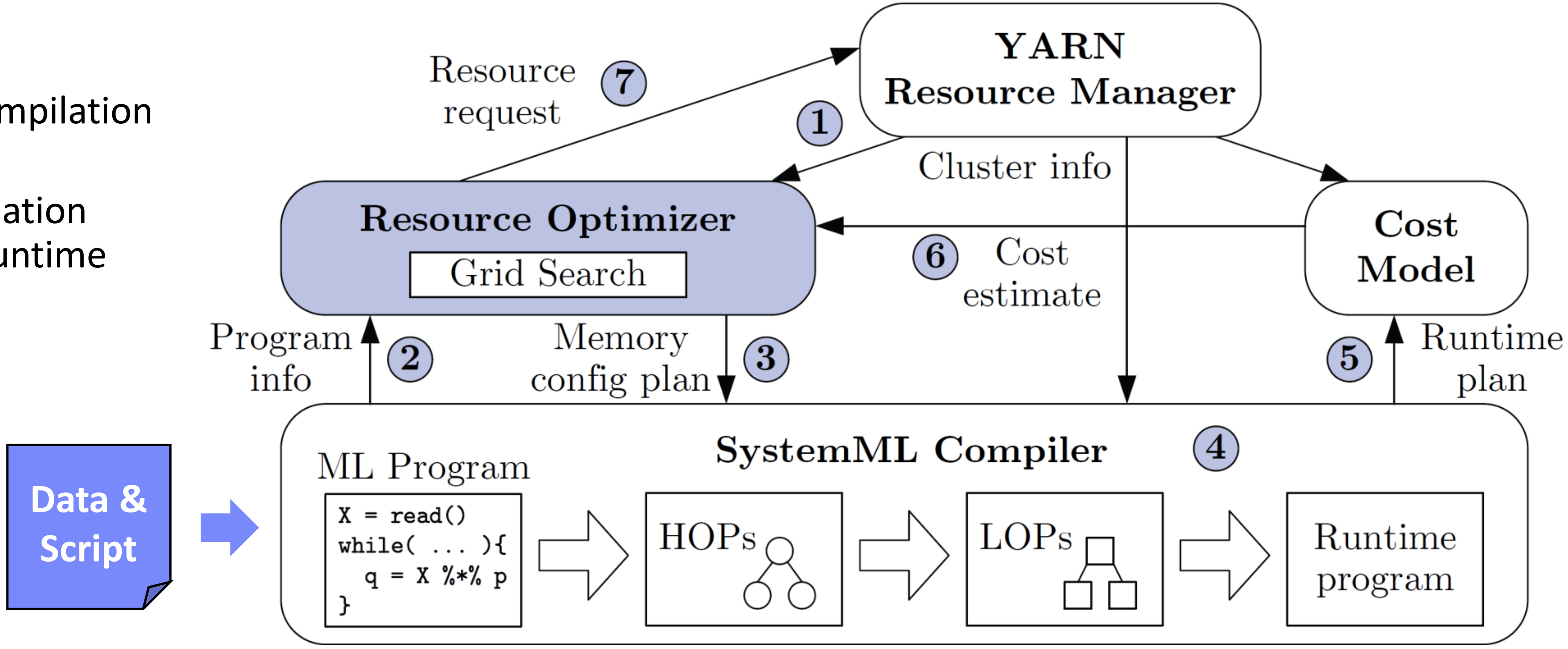
#1 Resource Elasticity

[SIGMOD'15]

[Botong Huang et al.: Resource Elasticity for Large-Scale Machine Learning. **SIGMOD 2015**]

- **Resource Optimizer for ML Workloads**
  - Optimize ML program resource configurations via online **what-if analysis and plan generation**
  - **Minimize cost w/o unnecessary overprovisioning**, program-aware enumeration (e.g., mem estimates)

- **Deployment**
  - Initial Compilation
  - Dynamic Recompilation during Runtime



#2 Sparsity-exploiting Operator Fusion

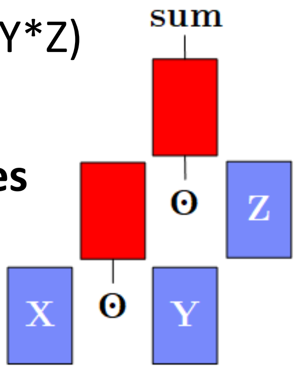
[PPoPP'15, PVLDB'16b, CIDR'17, PVLDB'18]

[Matthias Boehm et al.: On Optimizing Operator Fusion Plans for Large-Scale Machine Learning in SystemML. **PVLDB 2018**]

- **Motivation:** DAGs of linear algebra (LA) operations and statistical functions with materialized intermediates → ubiquitous fusion opportunities
- **Examples:**

sum(X*Y*Z)

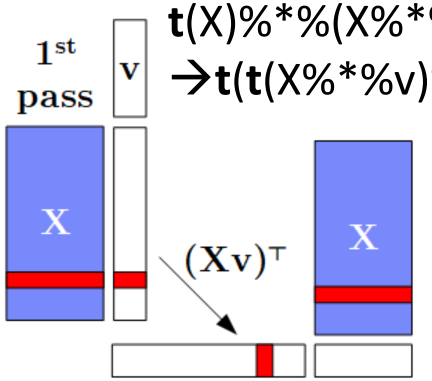
a) Intermediates



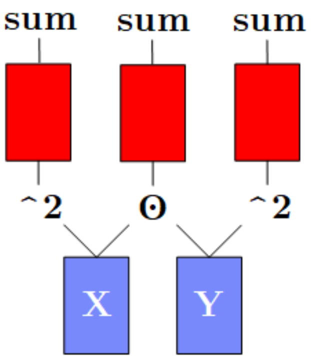
b) Single-Pass

t(X)%*%(X%*%v)

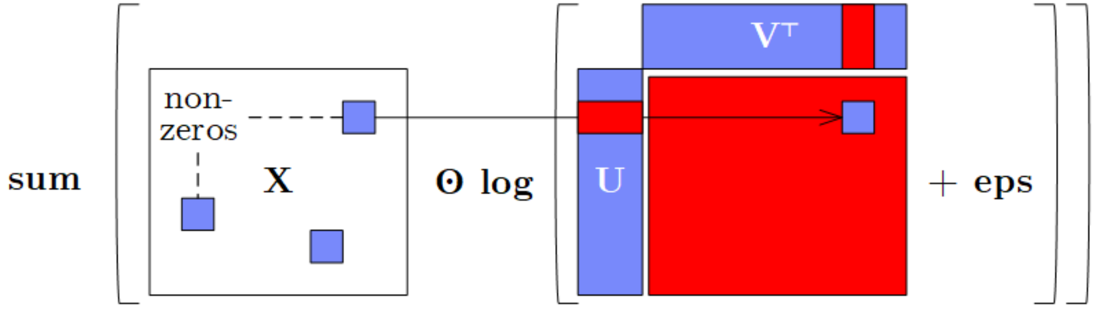
→t(t(X%*%v)%*%X)

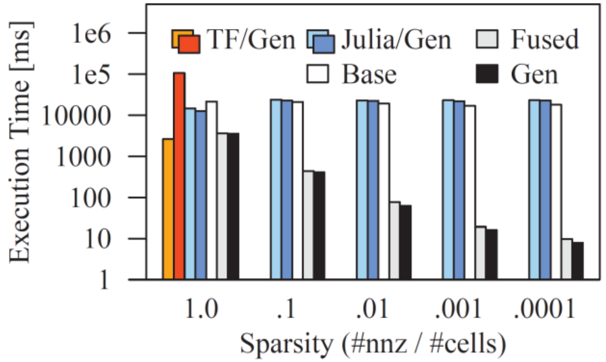


c) Multi-Aggregates



d) Sparsity Exploitation





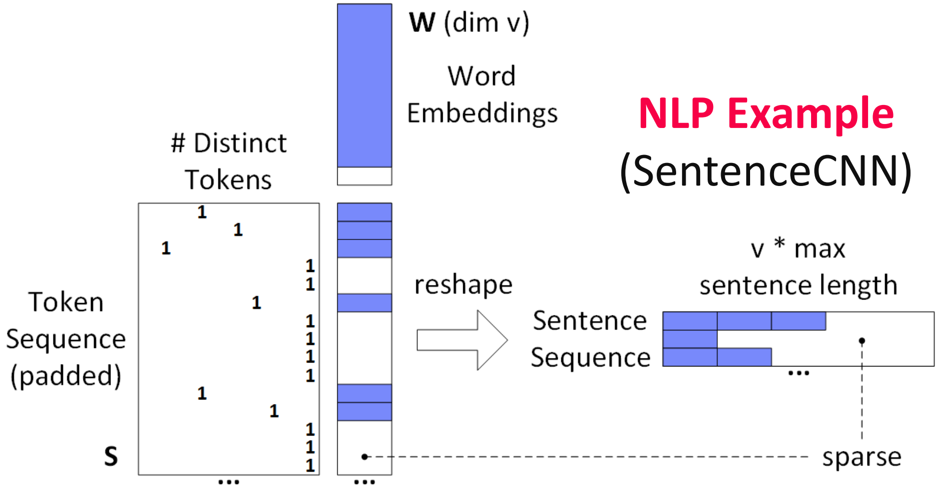
#3 Sparsity Estimation

[DEBull'14, SIGMOD'19]

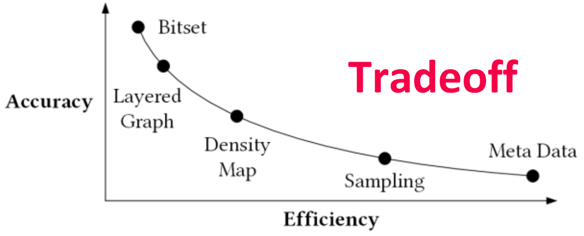
[Johanna Sommer et al.: **MNC:** Structure-Exploiting Sparsity Estimation for Matrix Expressions. **SIGMOD 2019**]

- **Motivation**

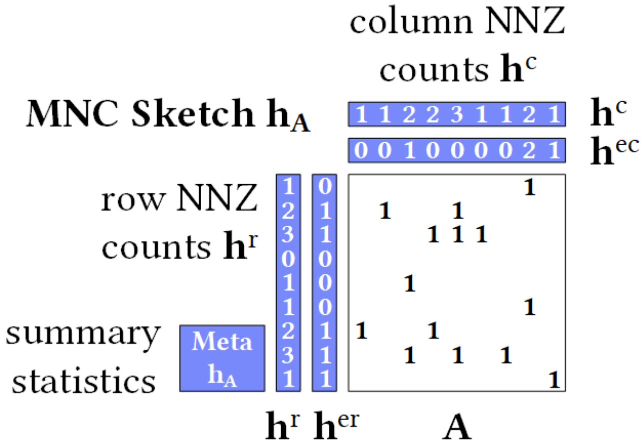
  - Sparse input matrices from NLP, graphs analytics, RecSys, HPC
  - Sparse intermediates (transform, dropout), and weights
  - Selection/permutation matrices



- **Sparsity Estimation**
  - Assumptions: no cancellation / no NaNs → Boolean matmult
  - Existing estimators: Naïve, Bitset, Sample, Hash, DMap, LGraph



- **MNC Sketch (Matrix Non-zero Count)**
  - Create MNC sketch for inputs A and B
  - Exploitation of structural properties (e.g., 1 non-zero per row, row sparsity)
  - Support for matrix expressions (reorganizations, elementwise ops)
  - Sketch propagation and estimation

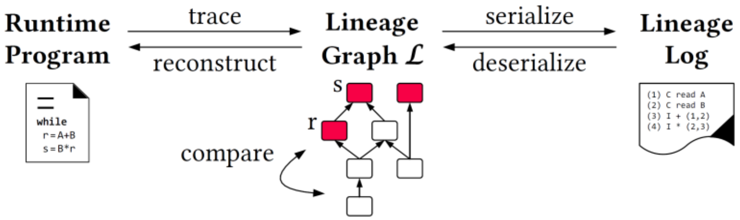


$s_C = \hat{s}_C = h_A^c h_B^r/(ml)$ if $\max(h_A^r) \le 1 \vee \max(h_B^c) \le 1$

#4 Multi-level Lineage Tracing & Reuse

[CIDR'20, SIGMOD'21]


- **Lineage as Key Enabling Technique**
  - Trace lineage of ops (incl. non-determinism), dedup for loops/funcs
  - Model versioning, data reuse, incr. maintenance, autodiff, debugging

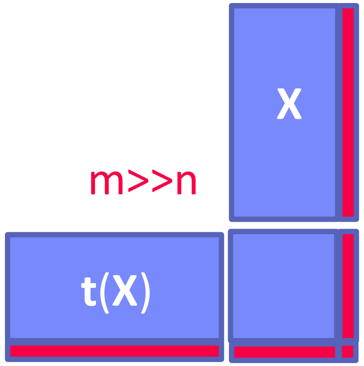


- **Full Reuse of Intermediates**
  - Before executing instruction, probe output lineage in cache `Map<Lineage, MatrixBlock>`
  - Cost-based/heuristic caching and eviction decisions (compiler-assisted)

```
for( i in 1:numModels )
  R[,i] = lm(X, y, lambda[i,], ...)
```

```
m_lmDS = function(...) {
  l = matrix(reg,ncol(X),1)
  A = t(X) %*% X + diag(l)
  b = t(X) %*% y
  beta = solve(A, b) ...}
```

- **Partial Reuse of Intermediates**
  - **Problem:** Often partial result overlap
  - Reuse partial results via dedicated rewrites (compensation plans)
  - Example: steplm



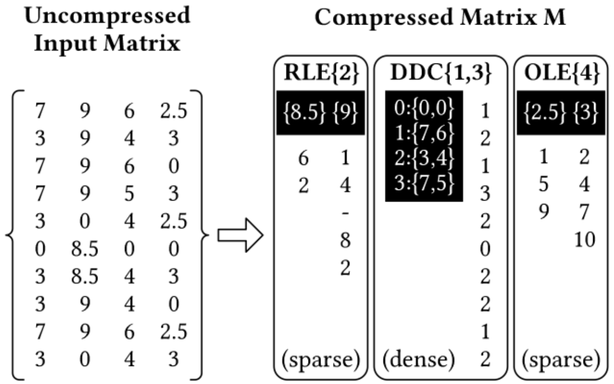

```
m_steplm = function(...) {
  while( continue ) {
    parfor( i in 1:n ) {
      if( !fixed[1,i] ) {
        Xi = cbind(Xg, X[,i])
        B[,i] = lm(Xi, y, ...)
    } }
    # add best to Xg (AIC)
} }
```

- **Next Steps:** multi-backend, unified mem mgmt

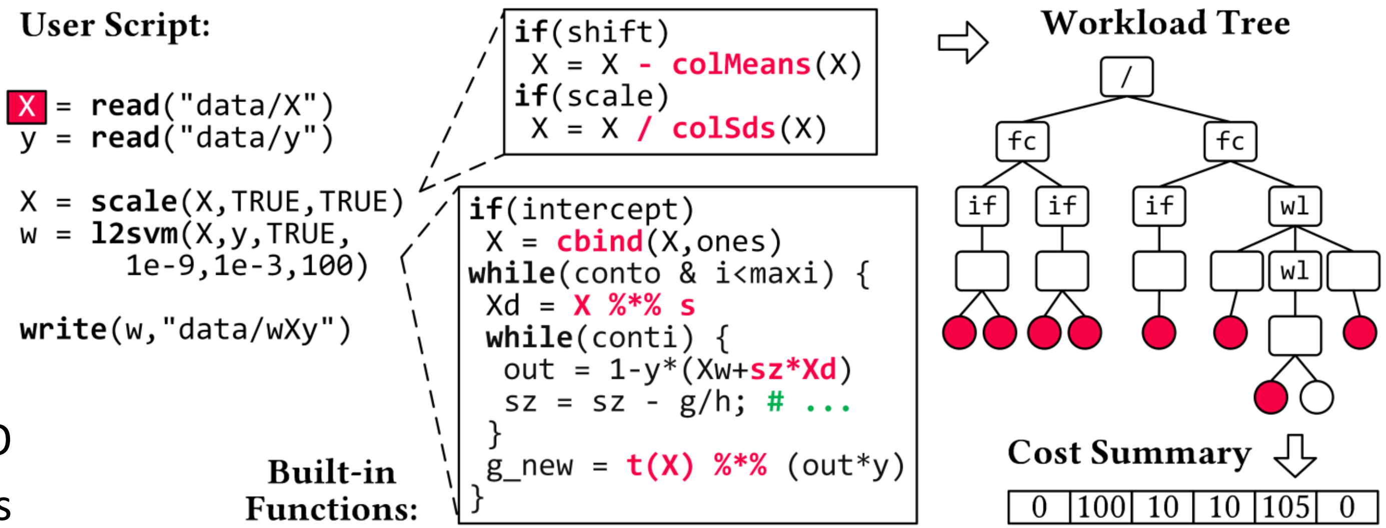
#5 Compressed Linear Algebra Extended

[PVLDB’16a, VLDBJ’18, SIGMOD’23a]

- **Lossless Matrix Compression**
  - Improved general applicability (adaptive compression time, new compression schemes, new kernels, intermediates, workload-aware)
  - Sparsity → Redundancy exploitation (data redundancy, structural redundancy)



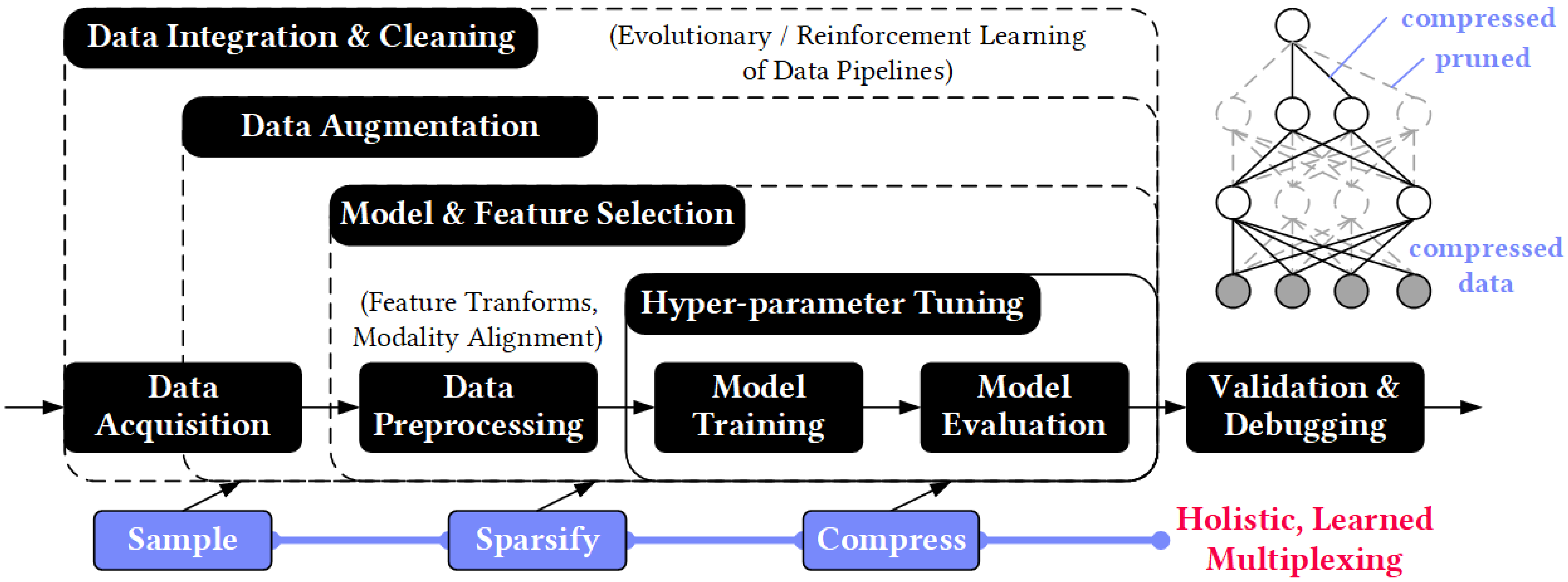
- **Workload-aware Compression**
  - Workload summary → compression
  - Compressed Representation → execution planning



- **Next Steps**
  - Frame compression, compressed I/O
  - Compressed feature transformations
  - Morphing of compressed data

Holistic Redundancy Exploitation

- **Overall Vision**
  - **Learned multiplexing of redundancy-exploiting techniques** (application and parameterization)
  - More effective sparsity- and redundancy-exploiting techniques for **changing data characteristics**
  - Robust ML system integration for **end-to-end improvements** (runtime, memory/energy)



Holistic Redundancy Exploitation, cont.

- **Overall Approach**
  - End-to-end learning of a holistic multiplexing of redundancy-exploiting techniques
  - **Lossy decisions learned at algorithm level** (sampling, sparsification, lossy compression), combined with **lossless sparsity exploitation and compression at systems level**

$$W' = arg \min_{W} \; E_D(W) + \lambda \cdot R(W) + \cdots + \lambda_S \cdot \Sigma_{i=1}^{n}(W_i \neq 0) + \lambda_C \cdot |W|$$

#non-zeros #distinct

- **Multi-objective Optimization with Hierarchical Multiplexing**

```
while(!convergedOuter) {
  X1 = sample(X, ...)
  while(!convergedInner) {
    X2 = compress(X2, |X|)
    … q = X2 %*% w …
  }
}
```

(**proxy models** sufficient?)

Automatic Redundancy Exploitation (**foundational advancements** for sparsity/error estimators, new sparse/compressed data types and kernels, workload awareness)

Conclusions and Q&A

Thanks

- **#1 Data-centric ML Pipelines**
  - Increasingly complex, composite ML pipelines
  - State-of-the-art data engineering methods based on ML
  - Partial **resource, operational, and data redundancy**

→ **Optimizing Compiler and Runtime Infrastructure**

- **#2 Holistic Redundancy Exploitation (codename LAURYN)**
  - **Learned multiplexing of redundancy-exploiting techniques** (application and parameterization)
  - More effective sparsity- and redundancy-exploiting techniques for **changing data characteristics**
  - Robust ML system integration for **end-to-end improvements** (runtime, memory/energy)

- **TU Berlin – Big Data Engineering (DAMS Lab)**
  - **#1 Integrated Data Analysis Pipelines** (specialized for workload & HW)
  - **#2 Automatic Data Reorganization** (specialized for data characteristics)
  - **#3 Data Engineering and Model Debugging** (specialized for domain)
  - **#4 Data Platforms, Federated and Cloud Infra** (specialized deployment)

  → Needs appropriate **Abstractions** and inter-disciplinary **Collaborations**

https://github.com/apache/systemds
https://github.com/daphne-eu/daphne