Architecture of ML Systems (AMLS)
13 Model Deployment and Serving

Prof. Dr. Matthias Boehm
Technische Universität Berlin
Berlin Institute for the Foundations of Learning and Data
Big Data Engineering (DAMS Lab)

Last update: Jul 16, 2025

Announcements / Org

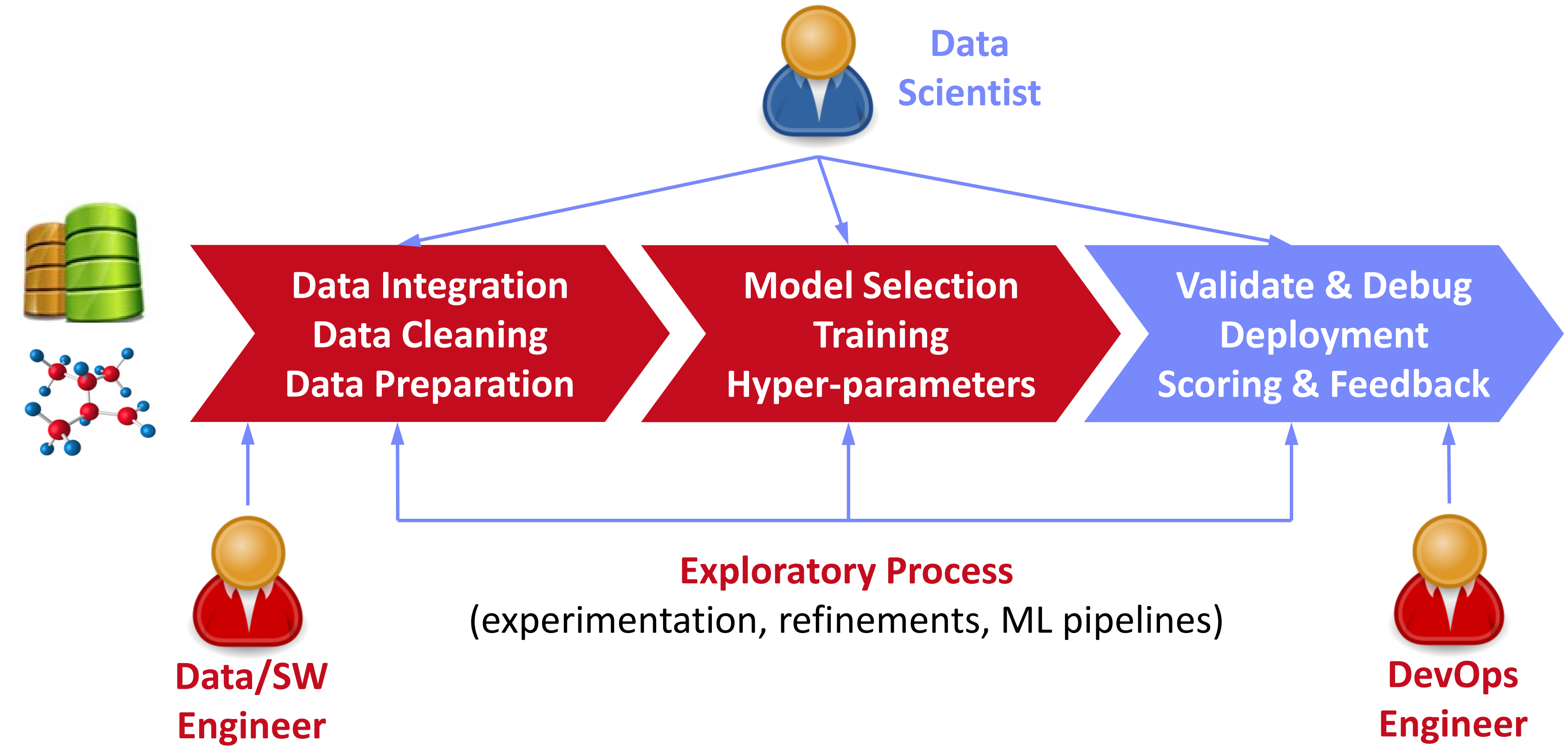
- **#1 Hybrid & Video Recording**
  - Hybrid lectures (in-person, zoom) with optional attendance
    https://tu-berlin.zoom.us/j/9529634787?pwd=R1ZsN1M3SC9BOU1OcFdmem9zT202UT09
  - Zoom **video recordings**, links from website
    https://mboehm7.github.io/**teaching**/ss25_amls/index.htm

- **#2 Exercise/Project Submissions**
  - **64** submissions alternative exercise
  - **6+7** submissions SystemDS/DAPHNE projects

- **#3 Written Exams**
  - Thu **July 24, 4-6pm** (A 151, **max 50**) → 24 registrations
  - Thu **July 31, 4-6pm** (EW 201, **max 47**) → 48 registrations
  - Thu **Aug 14, 4-6pm** (A 151, **max 50**) → 34 registrations

Recap: The Data Science Lifecycle (aka KDD Process, aka CRISP-DM)

Data-centric View:
Application perspective
Workload perspective
System perspective



Agenda

- **Model Exchange and Serving**
- **Model Monitoring and Updates**

Model Exchange and Serving

Model Exchange Formats

- **Definition Deployed Model**
  - **#1 Trained ML model** (weight/parameter matrix)
  - **#2 Trained weights AND operator graph** / entire ML pipeline
    ➔ especially for DNN (many weight/bias tensors, hyper parameters, etc)

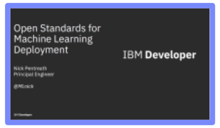
- **Recap: Data Exchange Formats** (model + meta data)
  - General-purpose formats: **CSV**, **JSON**, **XML**, **Protobuf**
  - Sparse matrix formats: **matrix market**, **libsvm**
  - Scientific formats: **NetCDF**, **HDF5**
  - ML-system-specific binary formats (e.g., SystemDS, PyTorch serialized)

```
%%MatrixMarket matrix coordinate real general
% ---------------------------------------------
% 0 or more comment lines
% ---------------------------------------------
5  5  8
1 1 1.000e+00
2 2 1.050e+01
3 3 1.500e-02
1 4 6.000e+00
4 2 2.505e+02
4 4 -2.800e+02
4 5 3.332e+01
5 5 1.200e+01
```

- **Problem ML System Landscape**
  - Different languages and frameworks, including versions
  - Lack of standardization → **DSLs for ML is wild west**

Model Exchange Formats, cont.

- **Why Open Standards?**
  - Open source allows inspection but no control
  - Open governance necessary for open standard
  - Cons: needs adoption, moves slowly



[Nick Pentreath: Open Standards for Machine Learning Deployment, **bbuzz 2019**]

- **#1 Predictive Model Markup Language (PMML)**
  - Model exchange format in XML, created by Data Mining Group 1997
  - Package model weights, hyper parameters, and **limited set of algorithms**

- **#2 Portable Format for Analytics (PFA)**
  - Attempt to fix limitations of PMML, created by Data Mining Group
  - JSON and AVRO exchange format
  - **Minimal functional math language** → arbitrary custom models
  - Scoring in JVM, Python, R

Model Exchange Formats, cont.

- **#3 Open Neural Network Exchange (ONNX)**
  - Model exchange format (data and operator graph) via Protobuf
  - First Facebook and Microsoft, then IBM, Amazon → PyTorch, MXNet
  - Focused on **deep learning and tensor operations**
  - ONNX-ML: support for traditional ML algorithms
  - Scoring engine: https://github.com/Microsoft/onnxruntime
  - Cons: **low level** (e.g., fused ops), **DNN-centric** → ONNX-ML

`python/systemds/onnx_systemds`

- **TensorFlow Saved Models**
  - TensorFlow-specific exchange format for model and operator graph
  - Freezes input weights and literals, for additional optimizations (e.g., constant folding, quantization, etc)
  - Cloud providers may not be interested in open exchange standards

ML Systems for Serving

- **#1 Embedded ML Serving**
  - TensorFlow Lite and new language bindings (small footprint, dedicated HW acceleration, APIs, and models: MobileNet, SqueezeNet)
  - TorchScript: Compile Python functions into ScriptModule/ScriptFunction
  - SystemML JMLC (Java ML Connector), IREE (data centers / edge – small footprint)
- **#2 ML Serving Services**
  - Motivation: Complex DNN models, ran on dedicated HW
  - RPC/REST interface for applications
  - TensorFlow Serving: configurable serving w/ batching
  - TorchServe: Specialized model for HW, batching/parallelism
  - Clipper: Decoupled multi-framework scoring, w/ batching and result caching
  - Pretzel: Batching and multi-model optimizations in ML.NET
  - Rafiki: Optimizations for accuracy s.t. latency constraints, batching, multi-model opt

Google Translate
140B words/day
→ **82K GPUs** in 2016

PyTorch TorchServe Config

```
models={
 "resnet-152": {"1.0": {
  "minWorkers": 1,
  "maxWorkers": 1,
  "batchSize": 8,
  "maxBatchDelay": 50,
  "responseTimeout": 120
}}}
```

[Christopher Olston et al: TensorFlow-Serving: Flexible, High-Performance ML Serving. **ML Systems**@NeurIPS **2017**]

[Daniel Crankshaw et al: Clipper: A Low-Latency Online Prediction Serving System. **NSDI 2017**]

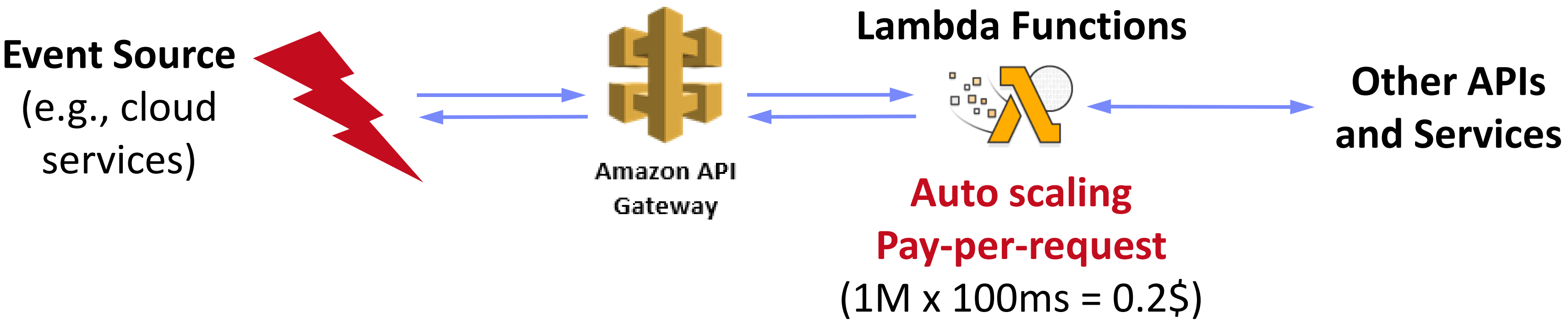
[Yunseong Lee et al.: PRETZEL: Opening the Black Box of Machine Learning Prediction Serving Systems. **OSDI 2018**]

[Wei Wang et al: Rafiki: Machine Learning as an Analytics Service System. **PVLDB 2018**]

Serverless Computing

[Joseph M. Hellerstein et al: Serverless Computing: One Step Forward, Two Steps Back. **CIDR 2019**]

- **Definition Serverless**
  - **FaaS:** functions-as-a-service (event-driven, stateless input-output mapping)
  - Infrastructure for deployment and auto-scaling of APIs/functions
  - Examples: **Amazon Lambda**, **Microsoft Azure Functions**, etc



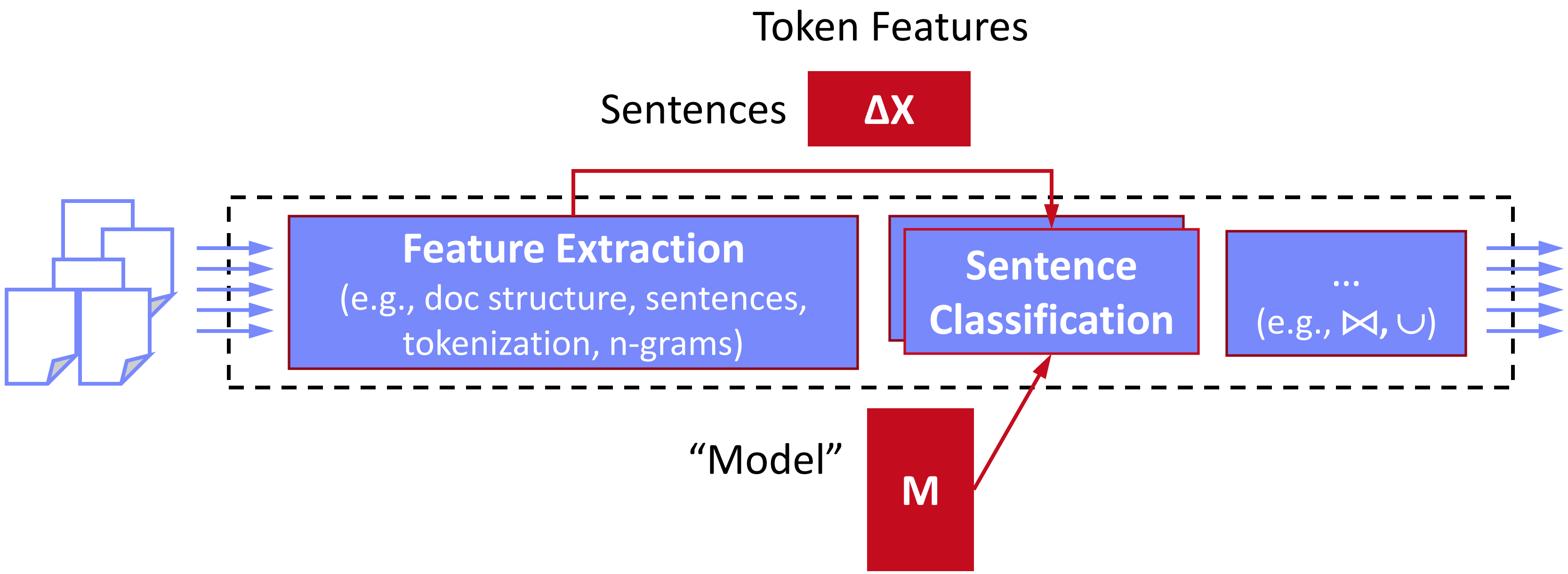
- **Example**

```java
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class MyHandler implements RequestHandler<Tuple, MyResponse> {
    @Override
    public MyResponse handleRequest(Tuple input, Context context) {
       return expensiveModelScoring(input); // with read-only model
    }
}
```

Example SystemDS JMLC

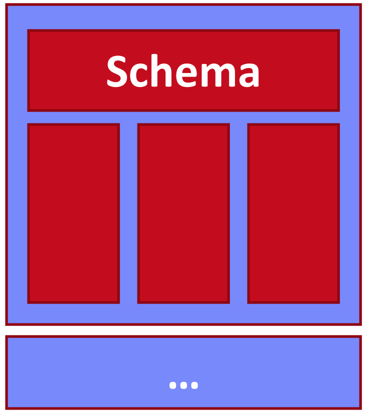
- **Example Scenario**



- **Challenges**
  - Scoring part of larger **end-to-end pipeline** → **Embedded scoring**
  - External parallelization w/o materialization
  - Simple **synchronous scoring** → **Latency ⇒ Throughput**
  - **Data size** (tiny ΔX, huge model M) → **Minimize overhead per ΔX**
  - **Seamless integration** & model consistency → **Token inputs & outputs**

Example SystemDS JMLC, cont.

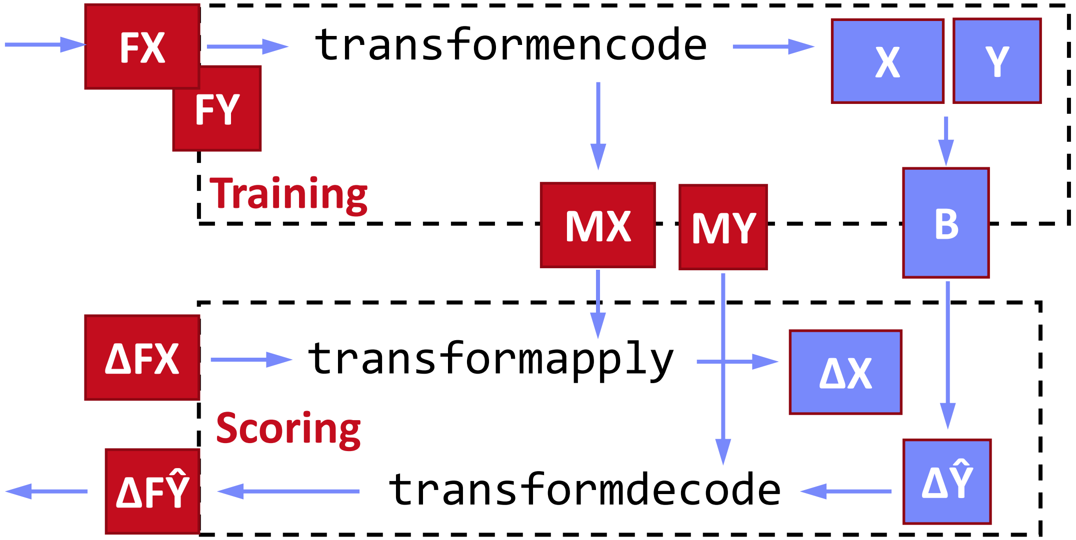
- **Background: Frame**
  - **Abstract data type with schema** (BIN, INT64, FP64, STR)
  - Column-wise block layout, with ragged arrays
  - Local and distributed operations



Distributed representation:
? x ncol(F) blocks

(shuffle-free conversion of csv / datasets)

- **Data Preparation via Transform**



Example SystemML JMLC, cont.

- **Motivation**
  - ➔ Embedded scoring
  - ➔ Latency ⇒ Throughput
  - ➔ Minimize overhead per ΔX

Typical compiler/runtime overheads:

Script parsing and config:	~100ms
Validation, compile, IPA:	~10ms
HOP DAG (re-)compile:	~1ms
Instruction execute:	<0.1µs

- **Example**

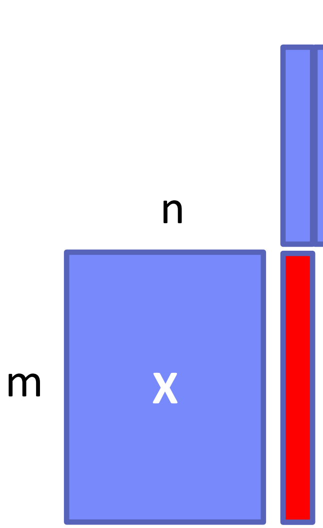
```
1: Connection conn = new Connection();
2: PreparedScript pscript = conn.prepareScript(
       getScriptAsString("glm-predict-extended.dml"),
       new String[]{"FX","MX","MY","B"}, new String[]{"FY"});
3: // ... Setup constant inputs
4: for( Document d : documents ) {
5:     FrameBlock FX = ...; //Input pipeline
6:     pscript.setFrame("FX", FX);
7:     FrameBlock FY = pscript.executeScript().getFrame("FY");
8:     // ... Remaining pipeline
9: }
```

// single-node, no evictions,
// no recompile, no multithread.

// execute precompiled script
// many times

Serving Optimizations – Batching

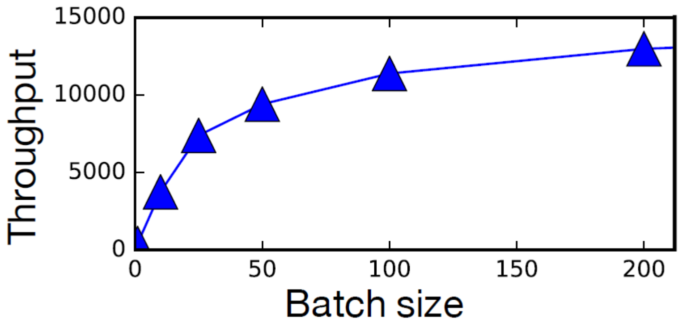
- **Recap: Model Batching** (see 08 Data Access)
  - One-pass evaluation of multiple configurations
  - EL, CV, feature selection, hyper parameter tuning
  - E.g.: TUPAQ [SoCC'16], Columbus [SIGMOD'14]



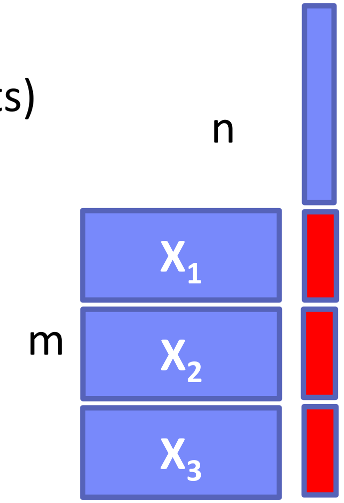
O(m*n) read
O(m*n***k**) compute
m >> n >> k

- **Data Batching**
  - Batching to utilize the HW more efficiently under SLA
  - **Use case:** multiple users use the same model (wait and collect requests)
  - **Adaptive:** additive increase, multiplicative decrease

[Clipper @ NSDI'17]



Fewer kernel launches, Parallelization



Benefits for multi-class / complex models

Serving Optimizations – Quantization

08 Data Access Methods

- **Quantization**
  - **Lossy compression via ultra-low precision / fixed-point**
  - Ex.: **62.7% energy** spent on data movement

[Amirali Boroumand et al.: Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks. **ASPLOS 2018**]

- **Quantization for Model Scoring**
  - Usually **much smaller data types** (e.g., **UINT8**)
  - Quantization of model weights, and sometimes also activations
    → reduced memory requirements and better latency / throughput (SIMD)

```
import tensorflow as tf
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
converter.optimizations = [tf.lite.Optimize.OPTIMIZE_FOR_SIZE]
tflite_quant_model = converter.convert()
```

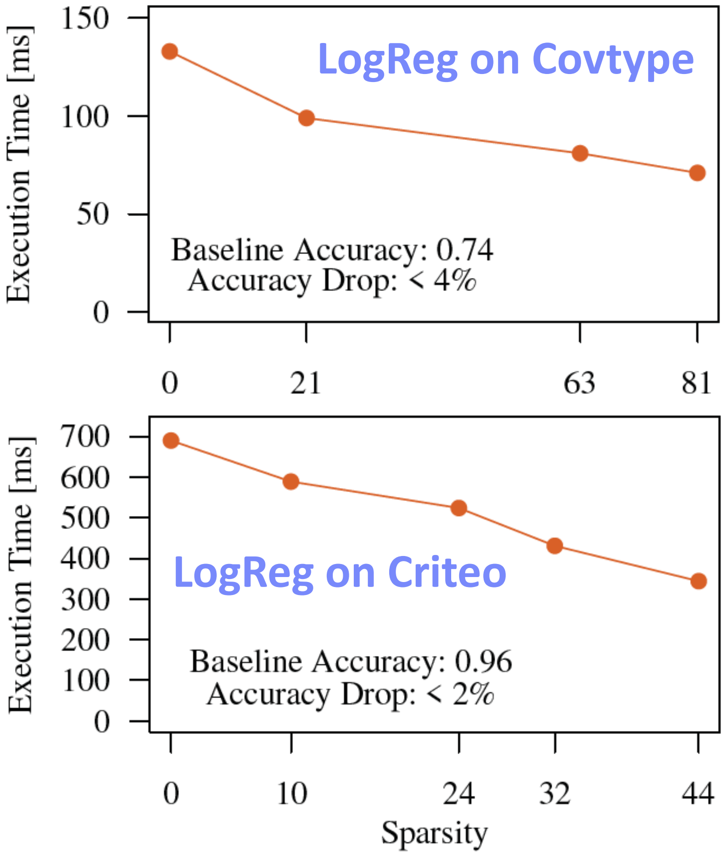
[**Credit:** https://www.tensorflow.org/lite/performance/post_training_quantization]

Operation	Energy
Load from **DRAM**	640 pJ
Load from **large SRAM**	50 pJ
Move 10mm across chip	32 pJ
Load from **local SRAM**	5 pJ
64-bit FMA	5 pJ
32-bit FMA	1.2 pJ
16-bit IMUL	0.26 pJ
8-bit IADD	0.01 pJ

[Jonathan Ragan-Kelly: The Future of Fast Code: Giving Hardware What It Wants, **PLDI 2024** Keynote (inspired by Bill Dally on 14nm)]

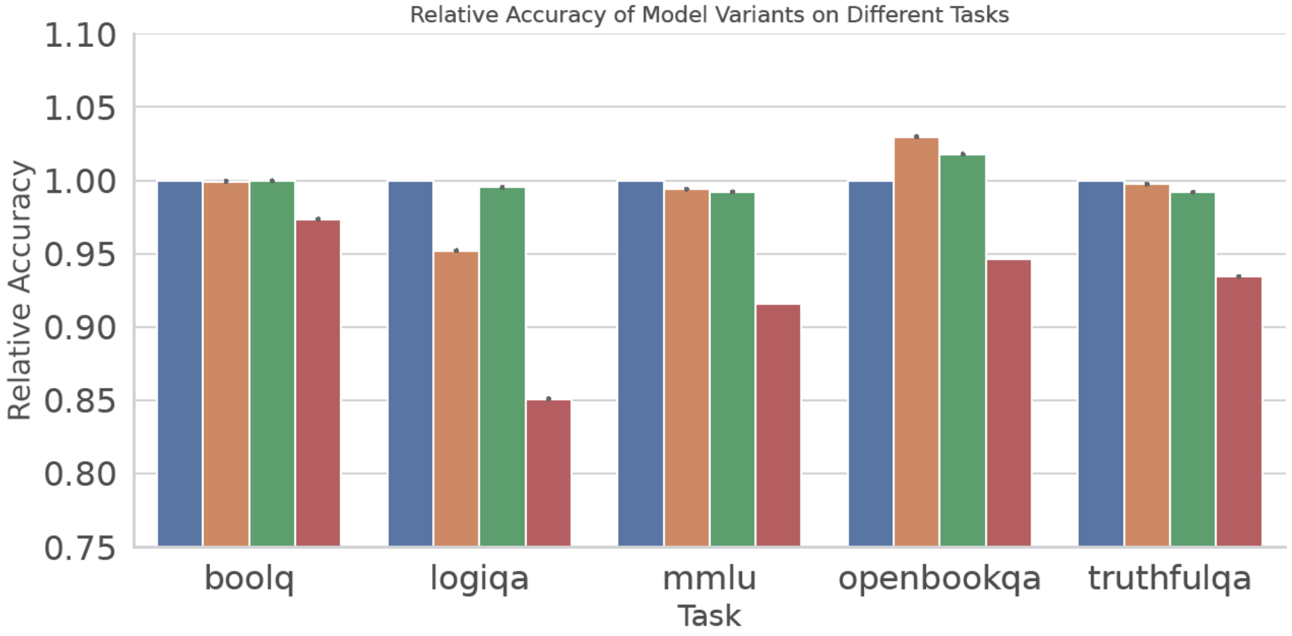
Serving Optimizations – Sparsification / Quantization, cont.

Sparsification



[**Credit:** Arnab Phani (DEEM@TU Berlin)]

Quantization



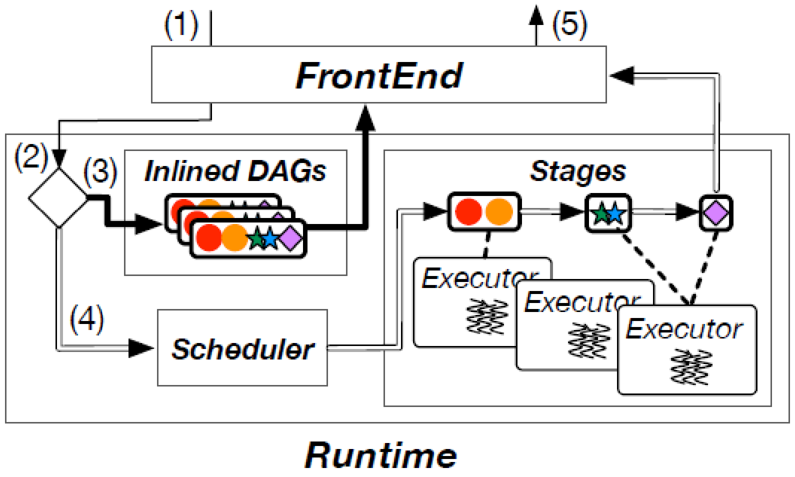
[**Credit:** Xiaozhe Yao (ETH Zurich)]

Serving Optimizations – MQO

- **Result Caching**
  - Establish a function cache for X → Y (memoization of deterministic function evaluation)
  - E.g., translation use case

```
Predict(m: ModelId, x: X) -> y: Y
```

- **Multi Model Optimizations**
  - Same input fed into multiple partially redundant model evaluations
  - Common subexpression elimination between prediction programs
  - In PRETZEL, programs compiled into physical stages and registered with the runtime + caching for stages (decided based on hashing the inputs)



[Yunseong Lee et al.: PRETZEL: Opening the Black Box of Machine Learning Prediction Serving Systems. **OSDI 2018**]

Serving Optimizations – Compilation

- **TensorFlow `tf.compile`**
  - Compile entire TF graph into binary function w/ low footprint
  - **Input:** Graph, config (feeds+fetches w/ fixes shape sizes)
  - **Output:** x86 binary and C++ header (e.g., inference)
  - **Specialization for frozen model and sizes**

[Chris Leary, Todd Wang: XLA – TensorFlow, Compiled!, **TF Dev Summit 2017**]

- **PyTorch Compile**
  - Compile Python functions into ScriptModule/ScriptFunction
  - Lazily collect operations, optimize, and JIT compile
  - Explicit `jit.script` call or `@torch.jit.script`

```
a = torch.rand(5)
def func(x):
  for i in range(10):
    x = x * x # unrolled into graph
  return x

jitfunc = torch.jit.script(func) # JIT
jitfunc.save("func.pt")
```

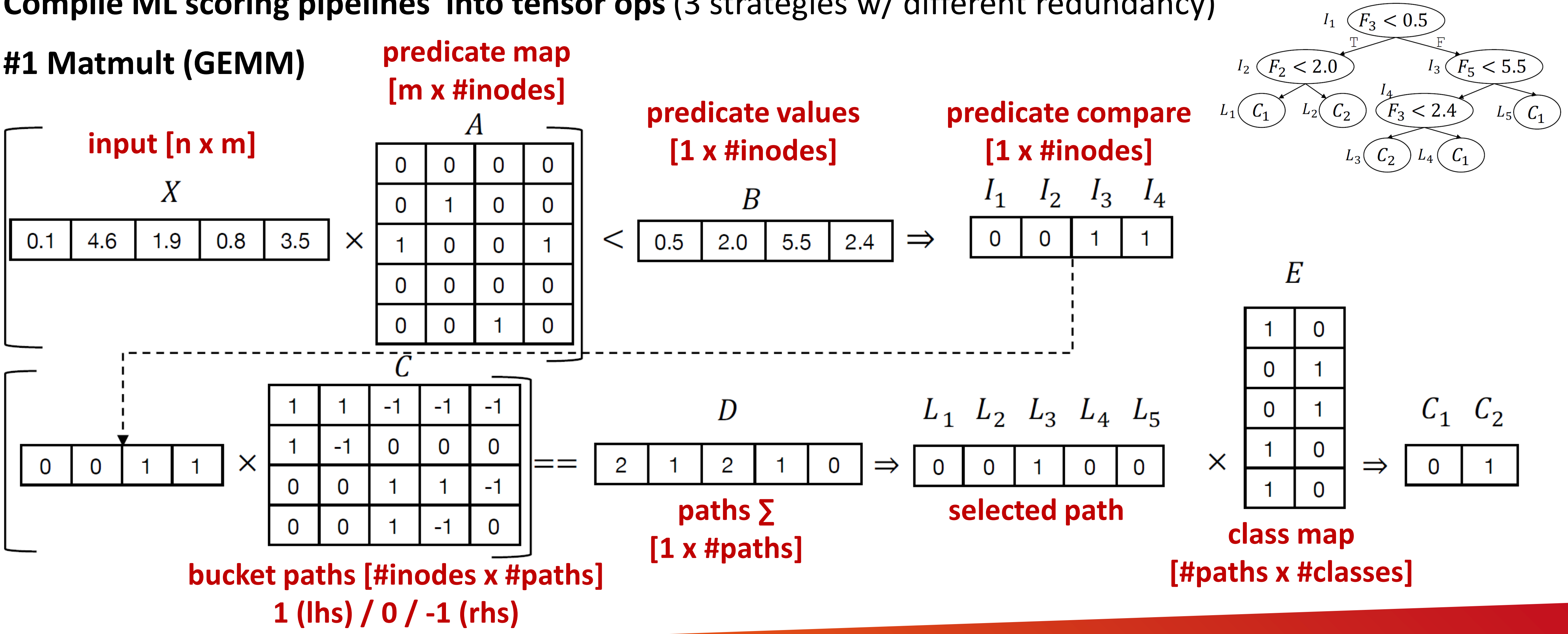
[Vincent Quenneville-Bélair: How PyTorch Optimizes Deep Learning Computations, **Guest Lecture Stanford 2020**]

Serving Optimizations – Model Vectorization

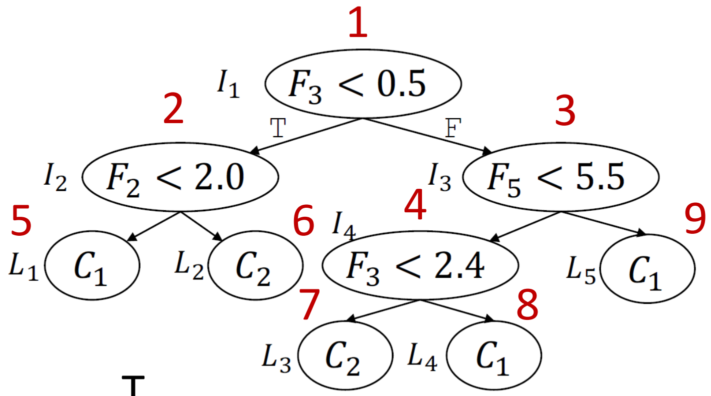
[Supun Nakandala et al: A Tensor Compiler for Unified Machine Learning Prediction Serving. **OSDI 2020,** https://github.com/microsoft/hummingbird]

- **Compile ML scoring pipelines into tensor ops** (3 strategies w/ different redundancy)

- **#1 Matmult (GEMM)**



Serving Optimizations – Model Vectorization, cont.



- **#2 Tree Traversal (TT)**
  - Traversal for batch of records via value indexing / table() and `ifelse(Tv<Tt, Tl, Tr)`

Algorithm 2 TreeTraversal Strategy (Notation in Tables 5)

Input : $X \in \mathbb{R}^{n \times |F|}$, Input records
Output : $R \in \{0,1\}^{n \times |C|}$, Predicted class labels

/* Initialize all records to point to k, with k the index of Root node. */
$T_I \leftarrow \{k\}^n$ // $T_I \in \mathbb{Z}^n$
for $i \leftarrow 1$ **to** TREE_DEPTH **do**
  /* Find the index of the feature evaluated by the current node. Then find its value. */
  $T_F \leftarrow$ Gather(N_F, T_I) // $T_F \in \mathbb{Z}^n$
  $T_V \leftarrow$ Gather(X, T_f) T_F // $T_V \in \mathbb{R}^n$
  /* Find the threshold, left child and right child */
  $T_T \leftarrow$ Gather(N_T, T_I) // $T_T \in \mathbb{R}^n$
  $T_L \leftarrow$ Gather(N_L, T_I) // $T_L \in \mathbb{Z}^n$
  $T_R \leftarrow$ Gather(N_R, T_I) // $T_R \in \mathbb{Z}^n$
  /* Perform logical evaluation. If true pick from T_L; else from T_R. */
  $T_I \leftarrow$ Where$(T_V < T_T, T_L, T_R)$ // $I \in \mathbb{Z}^n$
end
/* Find label for each leaf node */
$R \leftarrow$ Gather(N_C, T_I) // $R \in \mathbb{Z}^n$

Input data

F1	F2	F3	F4	F5
F1	F2	F3	F4	F5
F1	F2	F3	F4	F5

T_I

1
1
1

Nodes position of individual tuples

N_L	2	5	4	7	5	6	7	8	9
N_R	3	6	9	8	5	6	7	8	9
N_F	3	2	5	3	1	1	1	1	1
N_T	0.5	2.0	5.5	2.4	0	0	0	0	0
$t(N_C)$	0	0	0	0	1	0	0	1	1
	0	0	0	0	0	1	1	0	0

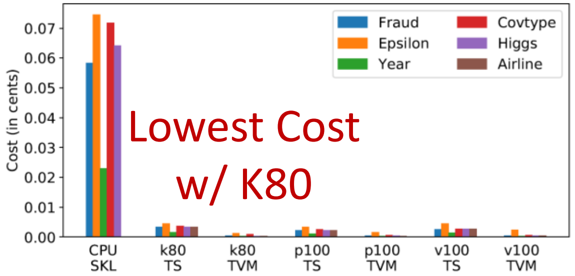
Serving Optimizations – Model Vectorization, cont.
Batch Scoring Experiments

Forest Inference Library (FIL)

Algorithm	Dataset	Baselines (CPU)		HB CPU			Baselines (GPU)	HB GPU	
		Sklearn	ONNX-ML	PyTorch	TorchScript	TVM	RAPIDS FIL	TorchScript	TVM
Rand. Forest	Fraud	**2.5**	7.1	8.0	7.8	3.0	not supported	0.044	**0.015**
	Epsilon	9.8	18.7	14.7	13.9	**6.6**	not supported	**0.13**	**0.13**
	Year	1.9	6.6	7.8	7.7	**1.4**	not supported	0.045	**0.026**
	Covtype	**5.9**	18.1	17.22	16.5	6.8	not supported	0.11	**0.047**
	Higgs	**102.4**	257.6	314.4	314.5	118.0	not supported	1.84	**0.55**
	Airline	1320.1	timeout	timeout	timeout	**1216.7**	not supported	18.83	**5.23**
LightGBM	Fraud	3.4	5.9	7.9	7.6	**1.7**	**0.014**	0.044	**0.014**
	Epsilon	10.5	18.9	14.9	14.5	**4.0**	0.15	0.13	**0.12**
	Year	5.0	7.4	7.7	7.6	**1.6**	**0.023**	0.045	0.025
	Covtype	51.06	126.6	79.5	79.5	**27.2**	not supported	0.62	**0.25**
	Higgs	198.2	271.2	304.0	292.2	**69.3**	0.59	1.72	**0.52**
	Airline	1696.0	timeout	timeout	timeout	**702.4**	5.55	17.65	**4.83**
XGBoost	Fraud	1.9	5.5	7.7	7.6	**1.6**	**0.013**	0.44	0.015
	Epsilon	7.6	18.9	14.8	14.8	**4.2**	0.15	0.13	**0.12**
	Year	3.1	8.6	7.6	7.6	**1.6**	**0.022**	0.045	0.026
	Covtype	42.3	121.7	79.2	79.0	**26.4**	not supported	0.62	**0.25**
	Higgs	126.4	309.7	301.0	301.7	**66.0**	0.59	1.73	**0.53**
	Airline	1316.0	timeout	timeout	timeout	**663.3**	5.43	17.16	**4.83**

Azure NC6 v2
(6 vcores, 112GB, P1 GPU)

Batch of 10K records
[seconds]



Serving Optimizations – Model Distillation

- **Model Distillation**
  - Ensembles of models → **single NN model**
  - Specialized models for different classes (found via differences to generalist model)
  - Trained on soft targets (softmax w/ temperature T)

[Geoffrey E. Hinton, Oriol Vinyals, Jeffrey Dean: Distilling the Knowledge in a Neural Network. **CoRR 2015**]

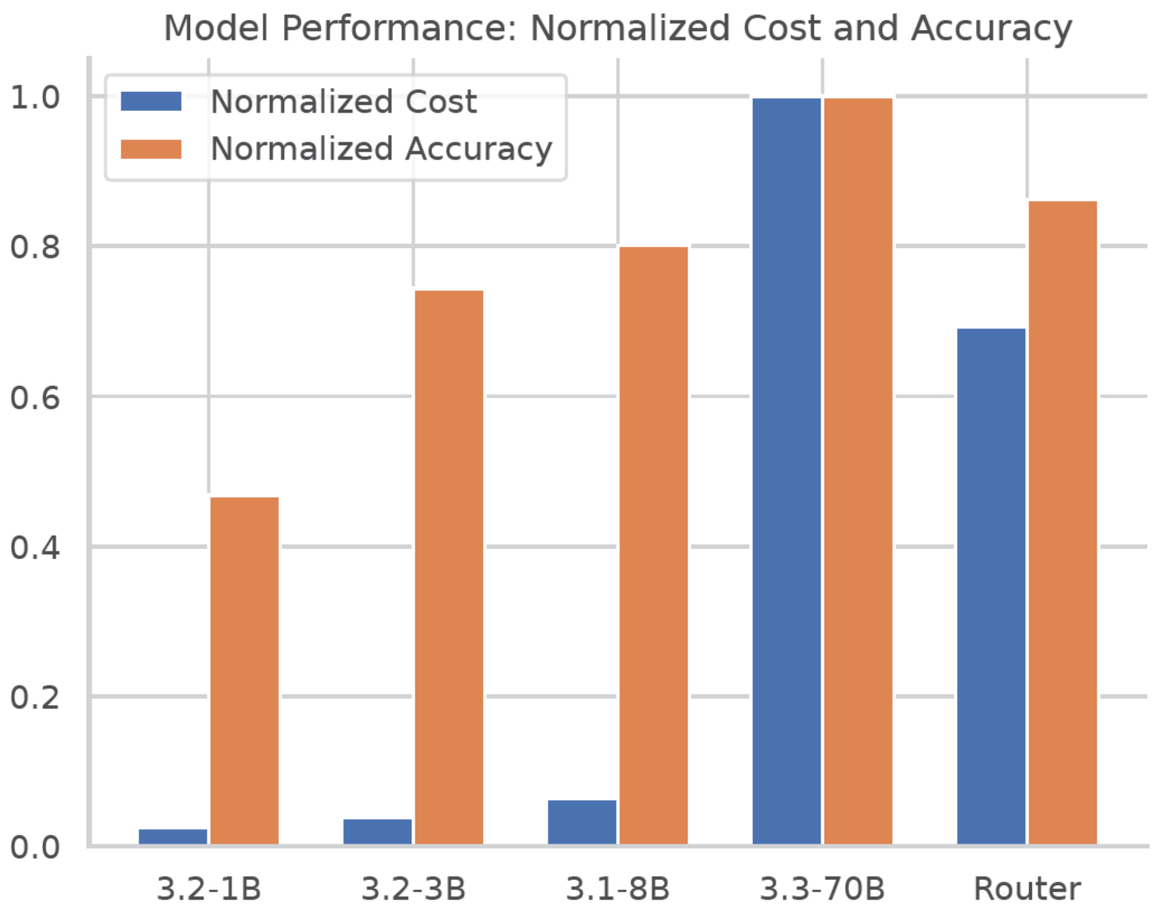
$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

- **Example Experiments**
  - Automatic Speech Recognition
  - Frame classification accuracy, and word error rate

System	Test Frame Accuracy	Word Error Rate
Baseline	58.9%	10.9%
10x Ensemble	61.1%	10.7%
Distilled 1x Model	60.8%	10.7%

Serving Optimizations – Model Distillation, cont.

- **LLaMA 2 Model Variants**
  - mmlu dataset
  - Tradeoff normalized accuracy and costs
  - Simple router model for improved tradeoff

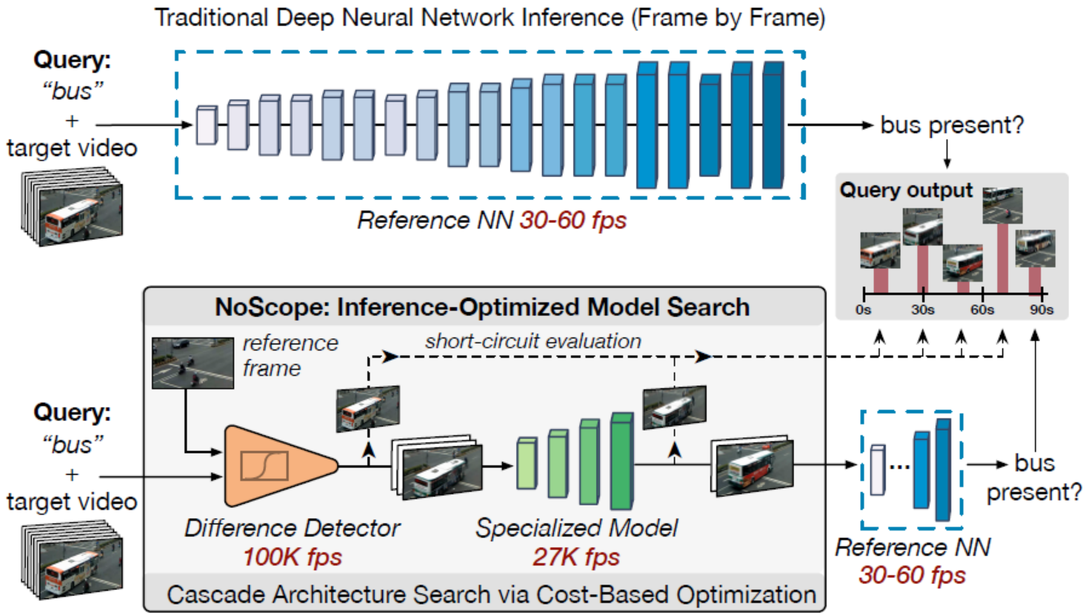


[**Credit:** Runsheng Benson Guo (UWaterloo)]

Serving Optimizations – Specialization

- **NoScope Architecture**
  - Baseline: YOLOv2 on 1 GPU per video camera @30fps
  - **Optimizer to find filters**

[Daniel Kang et al: NoScope: Optimizing Deep CNN-Based Queries over Video Streams at Scale. **PVLDB 2017**]



- **#1 Model Specialization**
  - Given query and baseline model
  - Trained shallow NN (based on AlexNet) on output of baseline model
  - Short-circuit if prediction with high confidence
- **#2 Difference Detection**
  - Compute difference to ref-image/earlier-frame
  - Short-circuit w/ ref label if no significant difference

Model Monitoring and Updates

Part of Model Management and MLOps
(see 10 Model Selection & Management)

Model Deployment Workflow

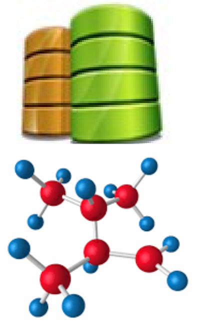
Data Integration
Data Cleaning
Data Preparation

Model Selection
Training
Hyper-parameters

#1 Model Deployment

MX MY B

Prediction Requests



#2 Continuous Data Validation / Concept Drift Detection

Model Serving

#3 Model Monitoring

#4 Periodic / Event-based Re-Training & Updates
(automatic / semi-manual)

DevOps Engineer

Monitoring Deployed Models


Google
Data Management Challenges in Production Machine Learning
Neoklis Polyzotis, Sudip Roy, Steven Whang, Martin Zinkevich

[Neoklis Polyzotis, Sudip Roy, Steven Whang, Martin Zinkevich: Data Management Challenges in Production Machine Learning, **SIGMOD 2017**]

- **Goals:** Robustness (e.g., data, latency) and **model accuracy**

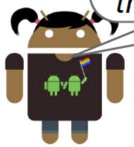
- **#1 Check Deviations Training/Serving Data**
  - Different data distributions, distinct items → impact on model accuracy?

  → See **09 Data Acquisition and Preparation** (Data Validation)

- **#2 Definition of Alerts**
  - Understandable and actionable
  - Sensitivity for alerts (**ignored if too frequent**)

- **#3 Data Fixes**
  - Identify problematic parts
  - Impact of fix on accuracy
  - How to backfill into training data

`age` should have a Kolmogorov distance of less than 0.1 from the previous day..



During serving: 0.11?

"The question is not whether something is 'wrong'. The question is whether it gets fixed"

Monitoring Deployed Models, cont.

[Neoklis Polyzotis, Sudip Roy, Steven Whang, Martin Zinkevich: Data Management Challenges in Production Machine Learning, **SIGMOD 2017**]

- **Alert Guidelines**
  - **Make them actionable**
    missing field,
    field has new values,
    distribution changes

    less actionable
  - **Question** data AND constraints
  - Combining repairs:
    principle of minimality

[George Beskales et al: On the relative trust between inconsistent data and inaccurate constraints. **ICDE 2013**]

[Xu Chu, Ihab F. Ilyas: Qualitative Data Cleaning. Tutorial, **PVLDB 2016**]

- **Complex Data Lifecycle**
  - Adding new features to production ML pipelines is a **complex process**
  - Data does not live in a DBMS; data often resides in **multiple storage systems** that have **different characteristics**
  - Collecting data for training can be **hard and expensive**

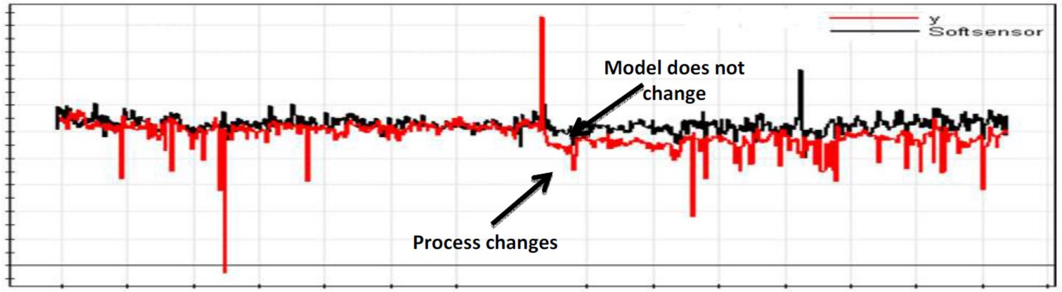
Concept Drift

[A. Bifet, J. Gama, M. Pechenizkiy, I. Žliobaitė: Handling Concept Drift: Importance, Challenges & Solutions, **PAKDD 2011**]

- **Recap Concept Drift** (features → labels)
  - Change of statistical properties / dependencies (features-labels)
  - Requires re-training, parametric approaches for deciding when to retrain

- **#1 Input Data Changes**
  - Population change (gradual/sudden), but also new categories, data errors
  - Covariance shift p(x) with constant p(y|x)

- **#2 Output Data Changes**
  - Label shift p(y)
  - Constant conditional feature distributed p(x|y)



source: Evonik Industries

- **Goals:** Fast adaptation; noise vs change, recurring contexts, small overhead

Concept Drift, cont.

[A. Bifet, J. Gama, M. Pechenizkiy, I. Žliobaitė: Handling Concept Drift: Importance, Challenges & Solutions, **PAKDD 2011**]

- **Approach 1: Periodic Re-Training**
  - Training: **window of latest data** + data selection/weighting
  - Alternatives: incremental maintenance, warm starting, online learning

- **Approach 2: Event-based Re-Training**
  - **Change detection** (supervised, unsupervised)
  - Often model-dependent, specific techniques for time series
  - **Drift Detection Method:** binomial distribution, if error outside scaled standard-deviation → raise warnings and alerts
  - **Adaptive Windowing (ADWIN):** window W, append data to W, drop old values until avg windows W=W1-W2 similar (below epsilon), raise alerts
  - **Kolmogorov-Smirnov distance / Chi-Squared:** univariate statistical tests training/serving

[Albert Bifet, Ricard Gavaldà: Learning from Time-Changing Data with Adaptive Windowing. **SDM 2007**]

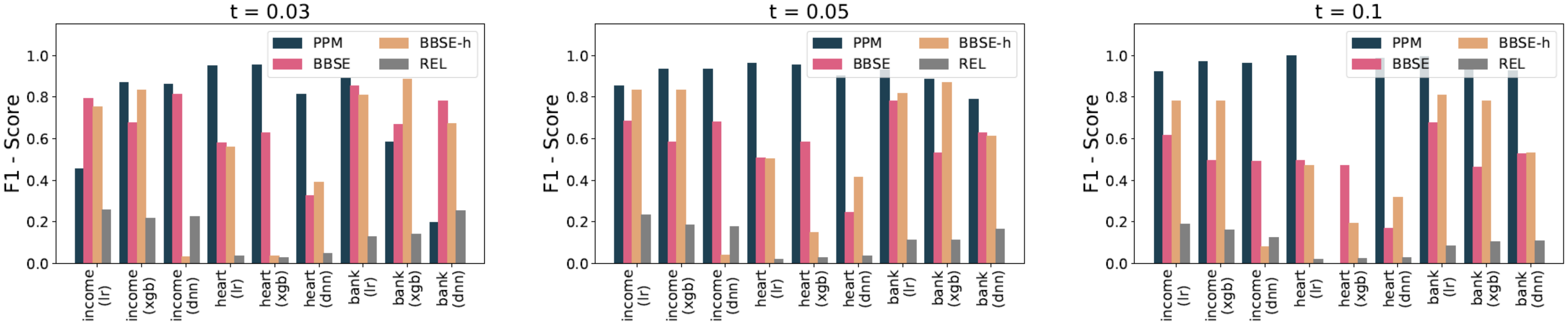
[https://scikitmultiflow.readthedocs.io/en/stable/api/generated/skmultiflow.drift_detection.ADWIN.html]

Concept Drift, cont.

[Sebastian Schelter, Tammo Rukat, Felix Bießmann: Learning to Validate the Predictions of Black Box Classifiers on Unseen Data. **SIGMOD 2020**]

- **Model-agnostic Performance Predictor**
  - Approach 2: Event-based Re-Training
  - User-defined error generators
  - Synthetic data corruption → impact on black-box model
  - Train performance predictor (regression/classification at threshold t) for expected prediction quality on **percentiles of target variable ŷ**

- **Results PPM**

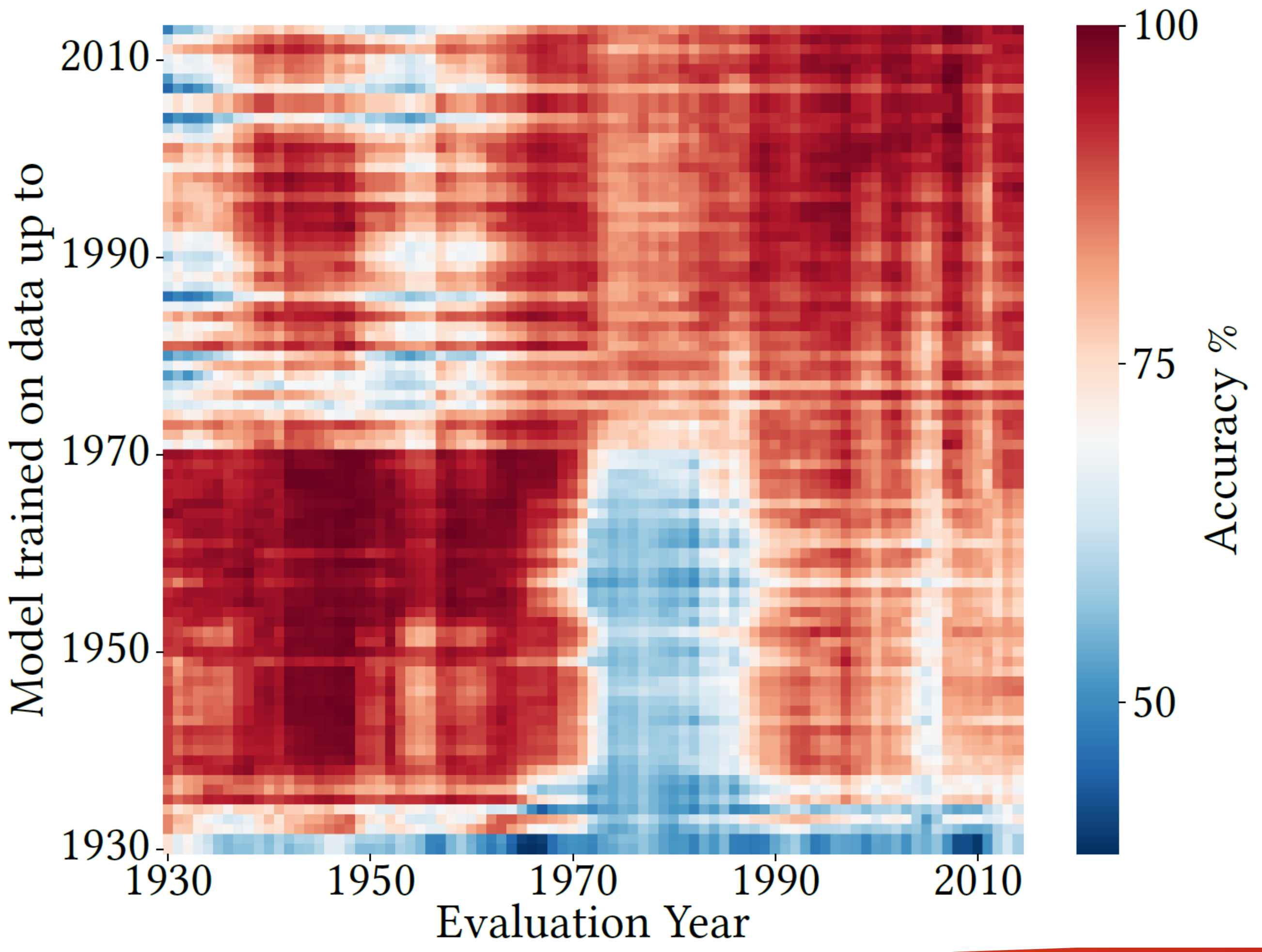


Concept Drift, cont.

[Maximilian Böther, Ties Robroek, Viktor Gsteiger, Robin Holzinger, Xianzhe Ma, Pinar Tözün, Ana Klimovic: Modyn: Data-Centric Machine Learning Pipeline Orchestration, **SIGMOD 2025**]

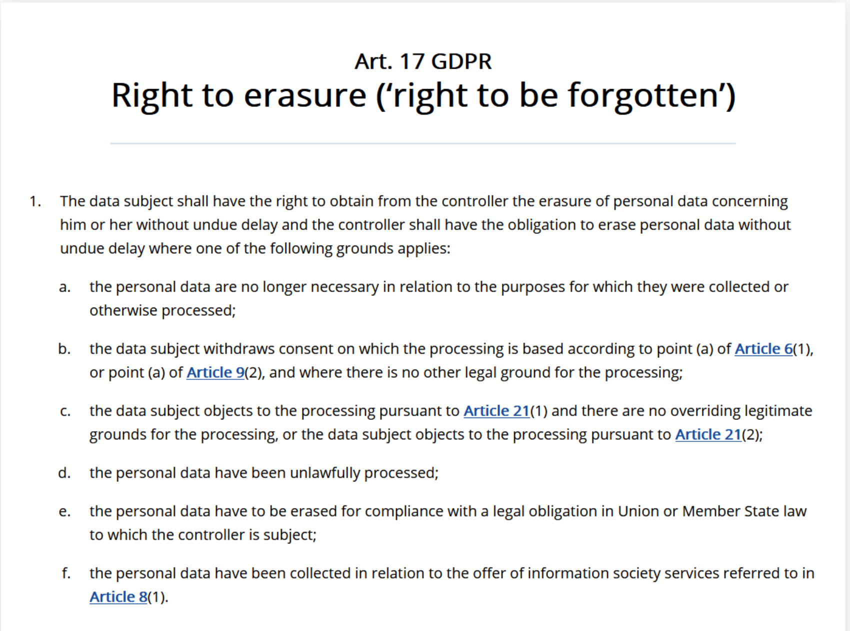
- **Yearbook Dataset**
  - Frontal-facing American high-school seniors
  - 1905 - 2013
  - Classification: male/female, smiles hair-styles

[https://shiry.ttic.edu/projects/yearbooks/yearbooks.html]



GDPR (General Data Protection Regulation)

[https://gdpr.eu/article-17-right-to-be-forgotten/]



Art. 17 GDPR

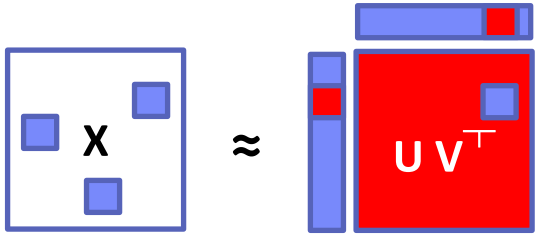
Right to erasure ('right to be forgotten')

1. The data subject shall have the right to obtain from the controller the erasure of personal data concerning him or her without undue delay and the controller shall have the obligation to erase personal data without undue delay where one of the following grounds applies:
   a. the personal data are no longer necessary in relation to the purposes for which they were collected or otherwise processed;
   b. the data subject withdraws consent on which the processing is based according to point (a) of Article 6(1), or point (a) of Article 9(2), and where there is no other legal ground for the processing;
   c. the data subject objects to the processing pursuant to Article 21(1) and there are no overriding legitimate grounds for the processing, or the data subject objects to the processing pursuant to Article 21(2);
   d. the personal data have been unlawfully processed;
   e. the personal data have to be erased for compliance with a legal obligation in Union or Member State law to which the controller is subject;
   f. the personal data have been collected in relation to the offer of information society services referred to in Article 8(1).

- **GDPR "Right to be Forgotten"**
  - Recent laws such as GDPR require companies and institutions to **delete user data upon request**
  - Personal data must not only be deleted from primary data stores but also from **ML models** trained on it (Recital 75)

- **Example Deanonymization**
  - Recommender systems: models **retain user similarly**
  - Social network data / clustering / KNN
  - Large language models (e.g., GPT-3)

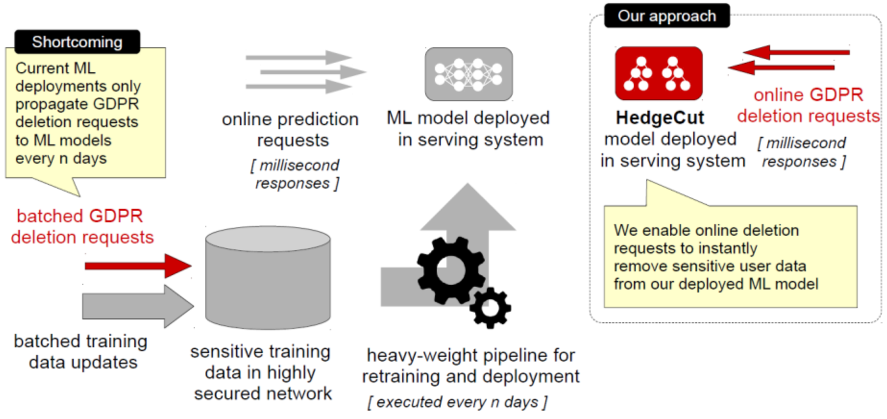
[Sebastian Schelter: "Amnesia" - Machine Learning Models That Can Forget User Data Very Fast. **CIDR 2020**]



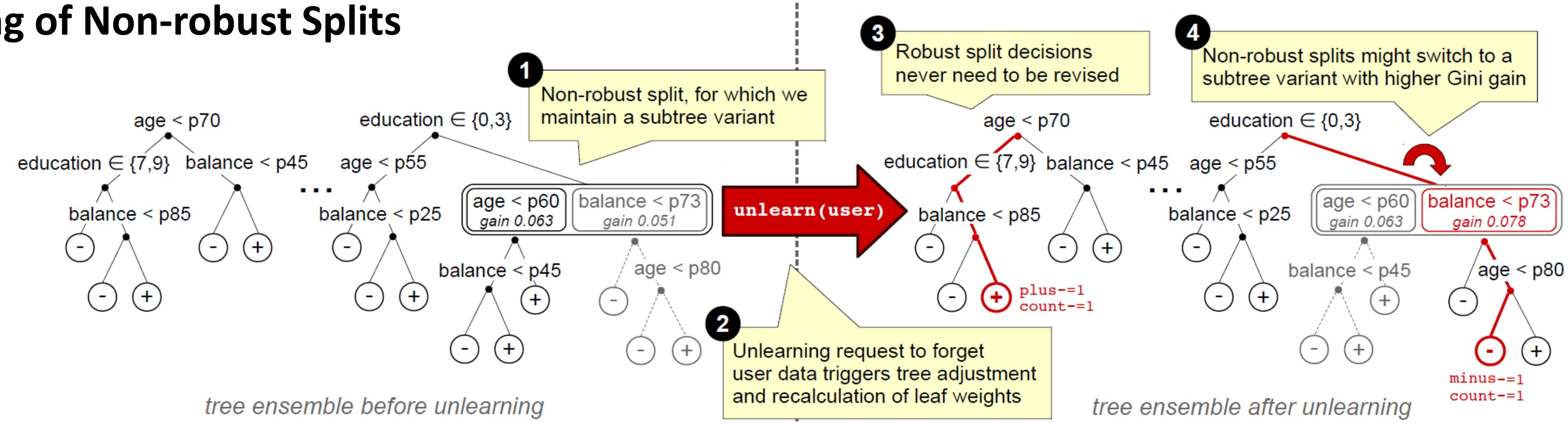
See incremental computations in **03 Sizes Inferences and Rewrites**

[Sebastian Schelter, Stefan Grafberger, Ted Dunning: HedgeCut: Maintaining Randomised Trees for Low-Latency **Machine Unlearning**, **SIGMOD 2021**]

- **HedgeCut Overview**
  - Extremely Randomized Trees (ERT): ensemble of DTs w/ randomized attributes and cut-off points
  - **Online unlearning requests** < 1ms w/o retraining for few points



- **Handling of Non-robust Splits**



Summary & QA

- **Model Exchange and Serving**
- **Model Monitoring and Updates**

Thanks

- **#1 Exam Preparation – Ask Questions in the Forum**
- **#2 Written Exams**
  - Thu **July 24, 4-6pm** (A 151, **max 50**) → 24 registrations
  - Thu **July 31, 4-6pm** (EW 201, **max 47**) → 48 registrations
  - Thu **Aug 14, 4-6pm** (A 151, **max 50**) → 34 registrations

Example AMLS Exams (90min for 100/100 points)
https://mboehm7.github.io/teaching/ss24_amls/ExamAMLS24_v1.pdf
https://mboehm7.github.io/teaching/ss24_amls/ExamAMLS24_v2.pdf
https://mboehm7.github.io/teaching/ss24_amls/ExamAMLS24_v3.pdf

No Lecture Materials or Mobile Devices

TECHNISCHE UNIVERSITÄT BERLIN

Architecture of ML Systems (AMLS)
14 Q&A and Exam Preparation [continues 5.45pm]

Prof. Dr. Matthias Boehm
Technische Universität Berlin
Berlin Institute for the Foundations of Learning and Data
Big Data Engineering (DAMS Lab)

Last update: Jul 15, 2025

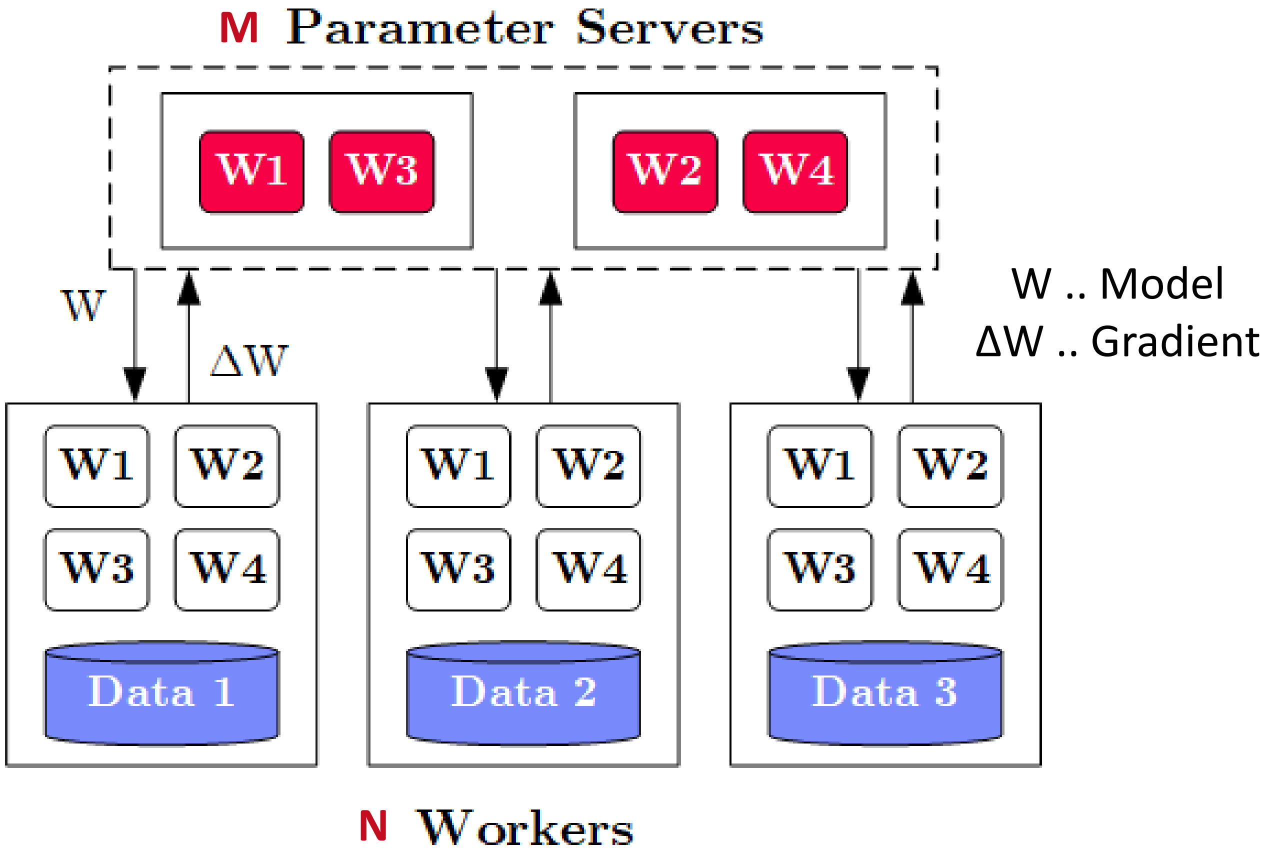
BIFOLD

Task 1 Parameter Servers [question appeared in every exam]

10/100

- **Task 1a: Describe the overall system architecture of data-parallel parameter servers, explain its components and interaction among these components [10/100 points]**

- **System Architecture**
  - M Parameter Servers w/ model
  - N Workers w/ data partitions
  - Optional Coordinator
- **Interactions**
  - Workers pull model from parameter servers, slice a mini-batch of data, run a forward and backward pass to compute gradients, which are pushed back to parameter serves
  - Parameter servers wait for gradients, aggregate the gradients/models, and perform a global model update



Task 1 Parameter Servers, cont. [question appeared in every exam]

- **Task 1b: Describe synchronous (BSP) and asynchronous (ASP) update strategies in data-parallel parameter servers and name their advantages and disadvantages. [6/100 points]**

	Synchronous	Asynchronous
Description	Per-batch or -n-batches synchronization barrier (wait for all workers before update)	Every pushed gradient updates the model, workers obtain model immediately
Advantages	**Consistent learning** process and model updates	**No waiting** for stragglers
Disadvantages	**Workers wait** for slowest worker (repeatedly)	Workers use stale models, **potential divergence**

Task 2 Data Preparation

- **Task 2a: Given the raw input data below, apply recoding and one-hot encoding to all categorical columns, and binning with 3 equi-width bins to all numerical columns. [10/100 points]**

A	B	C
Low	0	S
High	3	M
Med	7	L
Low	9	XL
Low	15	M
Low	7	M
Med	4	L
High	12	XL
High	13	L

→

ALow	AMed	AHigh	B	CS	CM	CL	CXL
1	0	0	1	1	0	0	0
0	0	1	1	0	1	0	0
0	1	0	2	0	0	1	0
1	0	0	2	0	0	0	1
1	0	0	3	0	1	0	0
1	0	0	2	0	1	0	0
0	1	0	1	0	0	1	0
0	0	1	3	0	0	0	1
0	0	1	3	0	0	1	0

Task 2 Data Preparation, cont.

- **Task 2b: What is feature hashing and what is its advantage over recoding? [3/100 points]**
  - **Hash values** and compute modulo with **user-provided k**
  - Reduces the number of distinct items, and thus columns in one-hot-encoded representation

- **Task 2c: Describe the text encodings bag-of-word and word-embeddings. [6/100 points]**
  - **Bag-of-word:** encode sentence as a **vector of token counts** (how often every distinct token appeared in the sentence)
  - **Word Embedding:** continuous bag-of-words, learned numerical vectors for **predicting the context words** from a word or a word from its context

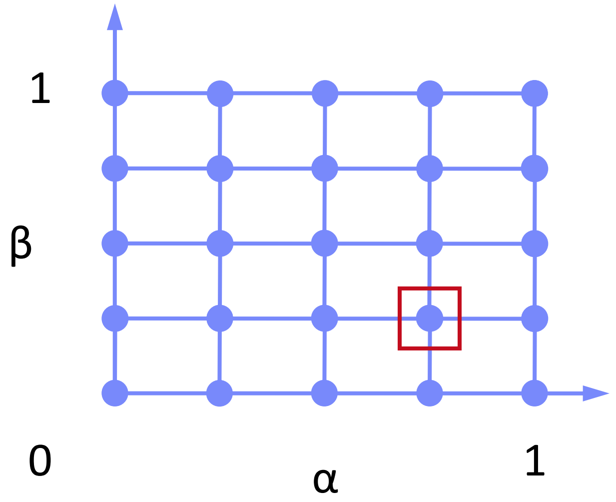
A	B	C	D	E
2	2	1	0	1

- **Task 2d: What is data augmentation and name 2 concrete techniques. [3/100 points]**
  - **Synthetically generate** labeled examples from small real labeled dataset through transformations
  - **Examples:** rotations, reflections, shearing, noise modulation

Task 3 Model Selection

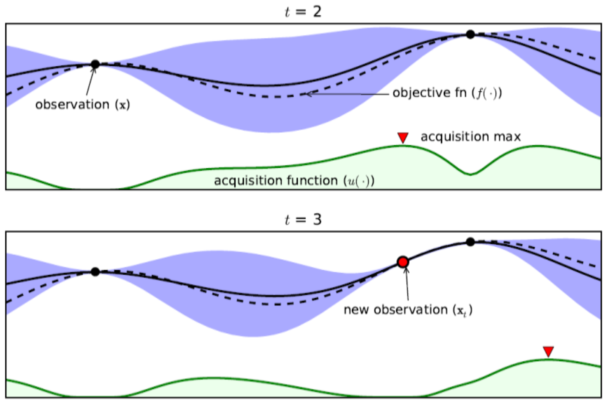
- **Task 3a: Describe the task of hyper-parameter tuning by example of GridSearch. Assume three hyper-parameters with 10 discretized values each, how many models do we need to train? [8/100 points]**

- **Hyper Parameter Tuning**
  - Given a model and dataset, **find best hyper parameter values** (e.g., learning rate, regularization, kernel parameters, tree params) by training the model and evaluating it on the validation set.
- **Grid Search**
  - Discretize continuous parameters (linearly or exponentially)
  - Every hyper-parameter is a dimension of a hyper-cube
  - For all combinations, train and evaluate the model
- **Example: 10^3 = 1000 trained models**



Task 3 Model Selection, cont.

- **Task 3b: Explain Bayesian Optimization as a more directed search strategy, and how it balances exploitation and exploration? [5/100 points]**
  - Use **lightweight ML** models like Gaussian Processes to find next points
  - **Acquisition function** to balance exploitation (expected mean) and exploration (uncertainty, expected variance)



- **Task 3c: Describe the problem of neural architecture search, and how to deal with multiple optimization objectives (e.g., accuracy and runtime). [5/100 points]**
  - Automatically compose neural network architectures from building blocks
  - **Search strategies: evolutionary algorithms** and Bayesian optimization
  - **Multi-objective optimization:** (1) **linearization**, (2) **pareto front to user**, (3) **primary objective w/ constraints** on other dims

Task 4 Model Debugging

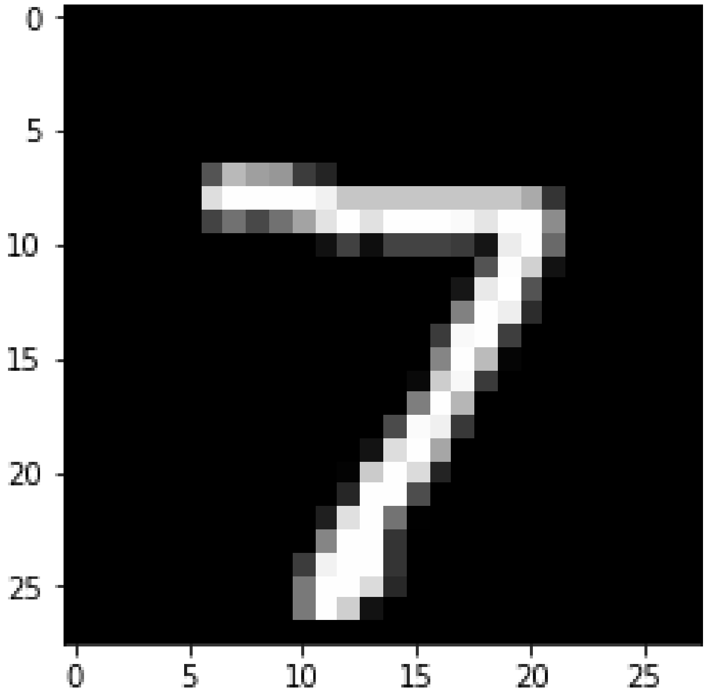
- **Task 4a: Describe sources of bias in machine learning and name examples how to ensure fairness when building ML models with examples. [4/100 points]**
  - **Sources:** selection bias, sample bias, data bias (e.g., NMAR), confirmation bias
  - **Fairness constraints:** monotonicity, group fairness constraints

- **Task 4b: Explain the concept of a confusion matrix and describe it in detail. [4/100 points]**
  - **Matrix of correct versus predicted labels**
  - Cells contain counts or relative frequencies of correct/predicted pair occurrences
  - Enables understanding which classes are "confused" with each other

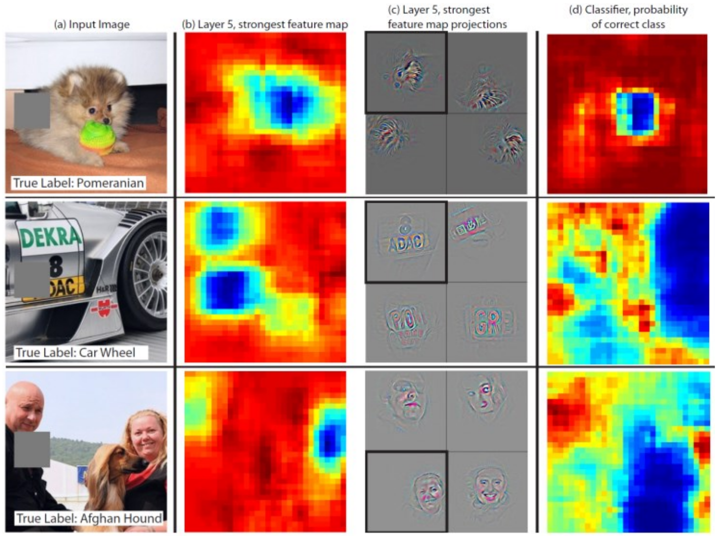
	predicted label									
correct label	0	1	2	3	4	5	6	7	8	9
0	21									
1		25								
2			15							
3				76						
4					23					12
5						36				
6							24			
7								31		37
8									42	
9					8			11		53

Task 4 Model Debugging, cont.

- **Task 4c: Explain the concept of occlusion-based explanations by example of classifying below hand-written digit as a seven [4/100].**

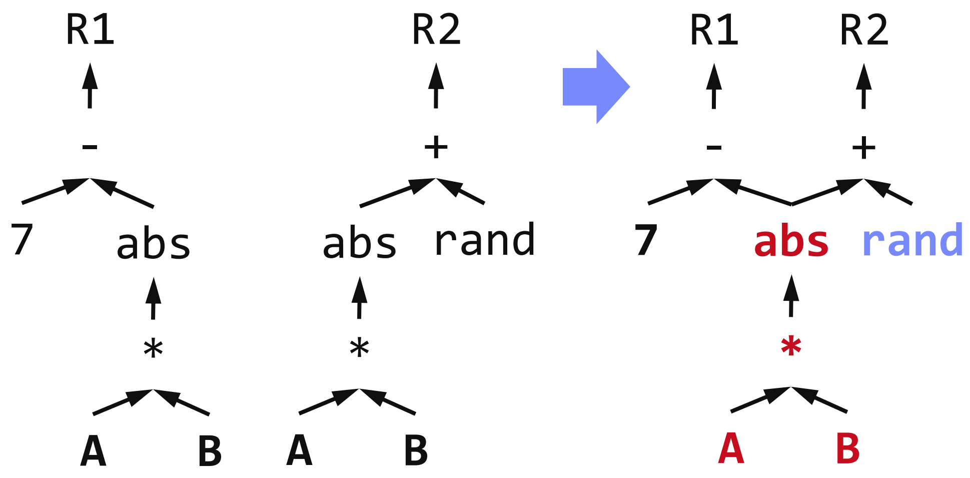


- **Slide black/gray square over input image**
- **Measure how feature maps (layer activation) and classifier output change**
- **Show a heat map of these changes**



Task 5 Compilation Techniques

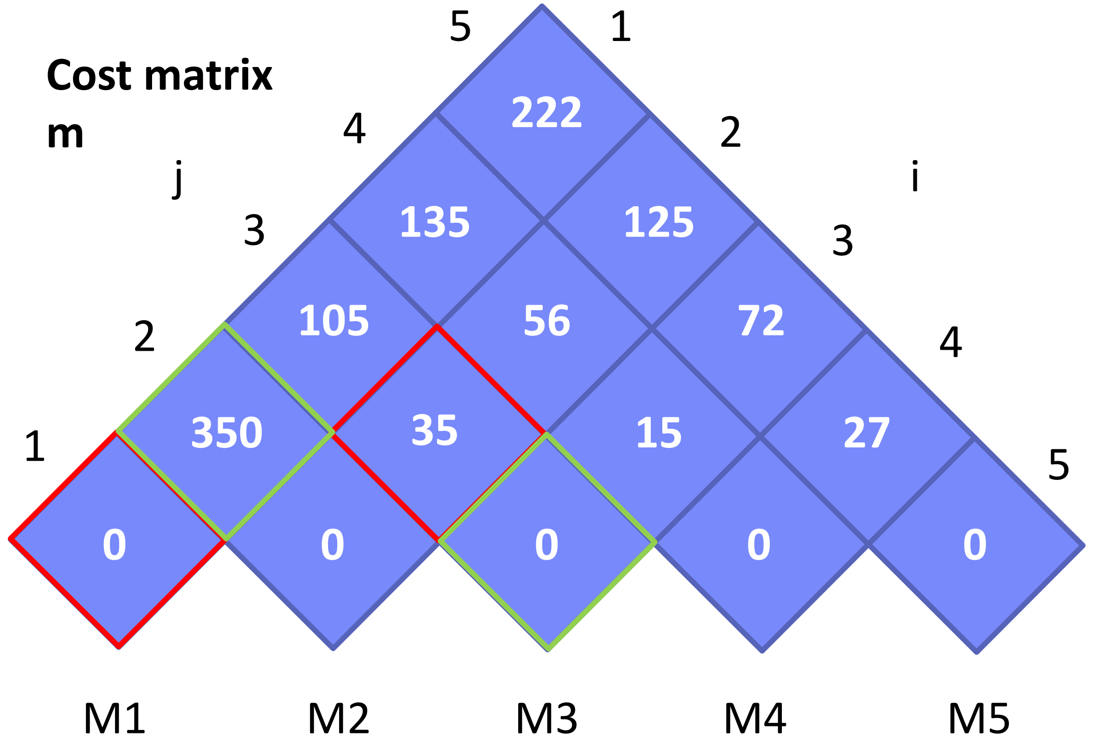
- **Task 5a: Describe the purpose of the rewrite common subexpression elimination (CSE) and sketch an algorithm to perform CSE on a directed acyclic graph (DAG) of operators. [5/100 points]**
  - Convert tree/graph of operators with redundant common subexpressions into redundancy-free operator graph
  - **Step 1:** Collect and **replace leaf nodes** (variable reads and literals)
  - **Step 2:** recursively **remove CSEs bottom-up** starting at the leaves by merging nodes with same inputs (**beware non-determinism**)



- **Task 5b: Explain the concept of operator fusion and how it can improve runtime performance [3/100 points]**
  - Merge sequence or sub-DAG of data-dependent operators into a single operator
  - **Performance Improvements: avoid unnecessary allocation**, reduced write/read **memory bandwidth** requirements / cache locality, **additional specialization** (e.g., data types, dimensions)

- **Task 5c: Assume an example chain of matrix multiplications (A B C D E), describe the problem of matrix multiplication chain optimization, and a dynamic programming algorithm for solving it efficiently [7/100 points]**

- **Matrix Multiplication is associative, mmchain opt aims to find optimal parenthesization**
- **Dynamic programming applies because (1) optimal substructure, and (2) overlapping subproblems**
- **Bottom-up sub-chain optimization via composition from solved subproblems**
- **Top-down read-out of optimal split matrix**

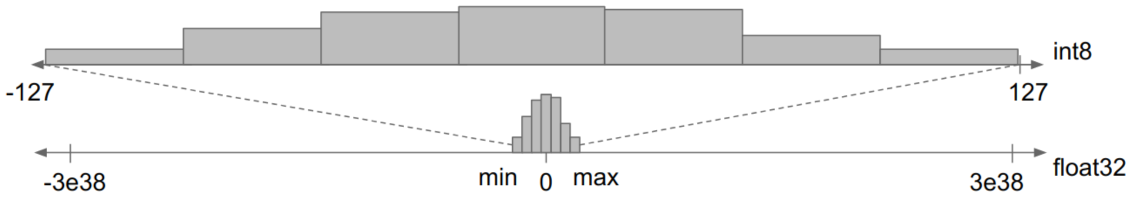


Task 6 Data Access Optimizations

- **Task 6a: Describe min-max quantization of an FP64 (floating point) representation into UINT8 (integer). Why does such an encoding increase training and/or inference performance? [8/100 points]**

FP64 →

Min/Max Quantization



- **Determine min/max range of matrix**
- **Split range into 2^8 = 256 buckets/bins**
- **Encode FP64 values in a bin via binID (lossy, but order-preserving)**

→ **Performance Improvements**

- (1) Reduced memory bandwidth requirements
- (2) Increased instruction parallelism (e.g., AVX512: 8 FP64 → 64 UINT8 ops)
- (3) Reduced energy consumption

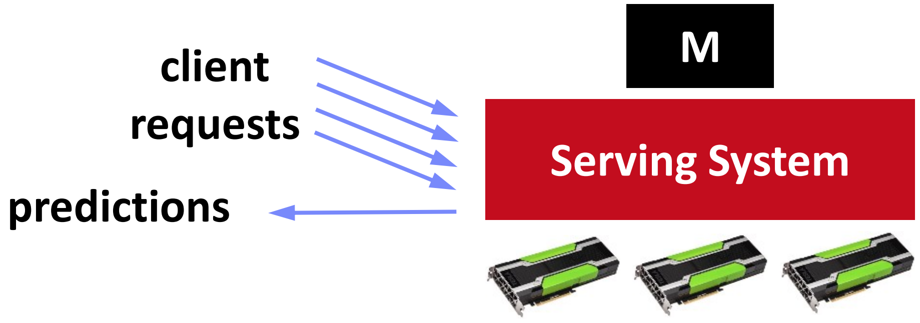
Operation	Energy
Load from DRAM	640 pJ
Load from large SRAM	50 pJ
Move 10mm across chip	32 pJ
Load from local SRAM	5 pJ
64-bit FMA	5 pJ
32-bit FMA	1.2 pJ
16-bit IMUL	0.26 pJ
8-bit IADD	0.01 pJ

[Jonathan Ragan-Kelly: The Future of Fast Code: Giving Hardware What It Wants, **PLDI 2024** Keynote (inspired by Bill Dally on 14nm)]

Task 7 Model Deployment

100/100

- Task 7a: Consider a deployed model M in a cloud serving environment and assume 1000s of clients. Explain three strategies for improving model scoring throughput at the serving site. [9/100 points]



- Caching / Reuse of input-prediction pairs (fewer model invocations)
- Batching of client requests / vectorization
  (fewer kernel launches, utilize compute better, less sync barriers)
- Quantization of input data (data transfer to serving systems, instruction parallelism)
- Inference Program Compilation
- Specialized, Smaller Models

THANKS
and
GOOD LUCK!