Architecture of ML Systems
12 Model Deployment & Serving

Matthias Boehm, <u>Arnab Phani</u>

Graz University of Technology, Austria
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMK endowed chair for Data Management

Last update: June 10, 2022

Announcements/Org

- **#1 Video Recording**
  - Link in **TeachCenter** & **TUbe** (lectures will be public)
  - Hybrid: HS i5 / https://tugraz.webex.com/meet/m.boehm
- **#2 Projects and Oral Exams**
  - **Precondition:** completed exercise/project by **Jun 17 EOD** (so far: 3x SIGMOD, 2x SystemDS, 3x Exercise completed)
  - Doodle for exam slot selection (~ 35/35) by **Jun 17 EOD** https://doodle.com/meeting/participate/id/eER4P10a

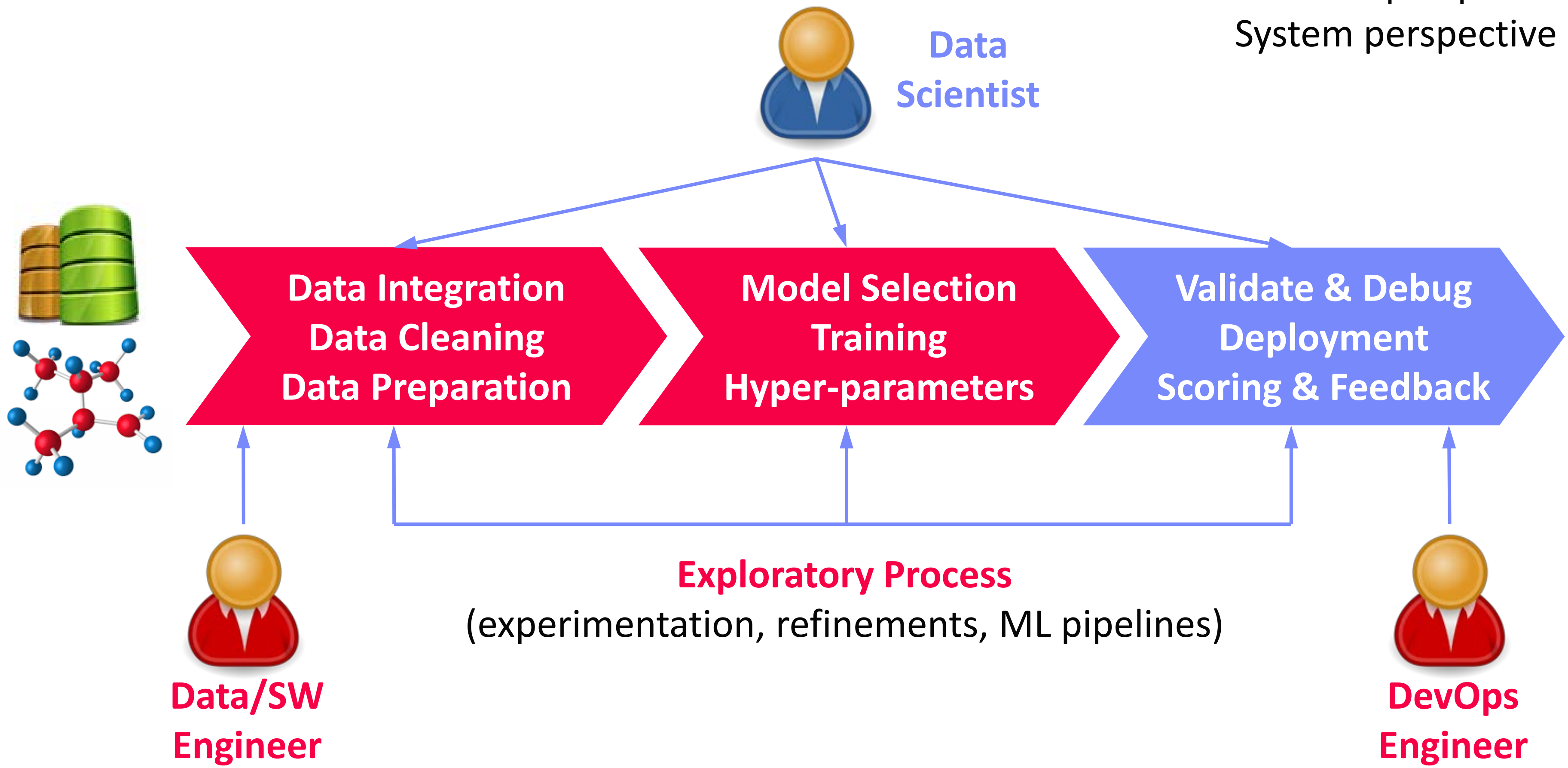
Q&A

- **#3 Course Evaluation**
  - Please participate; open period: **June 1 – July 15**

- **#4 Open Position**
  - Exploratory data analysis on vehicle video/time series data
  - **4 months for 20h/week**, preferred start **July 1**

Recap: The Data Science Lifecycle

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

Lukas Timpl
`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)
  - SystemML JMLC (Java ML Connector)

- **#2 ML Serving Services**
  - Motivation: Complex DNN models, ran on dedicated HW
  - RPC/REST interface for applications
  - TensorFlow Serving: configurable serving w/ batching
  - Clipper: Decoupled multi-framework scoring, w/ batching and result caching
  - Pretzel: Batching and multi-model optimizations in ML.NET
  - Rafiki: Optimization for accuracy under latency constraints, and batching and multi-model optimizations

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

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

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

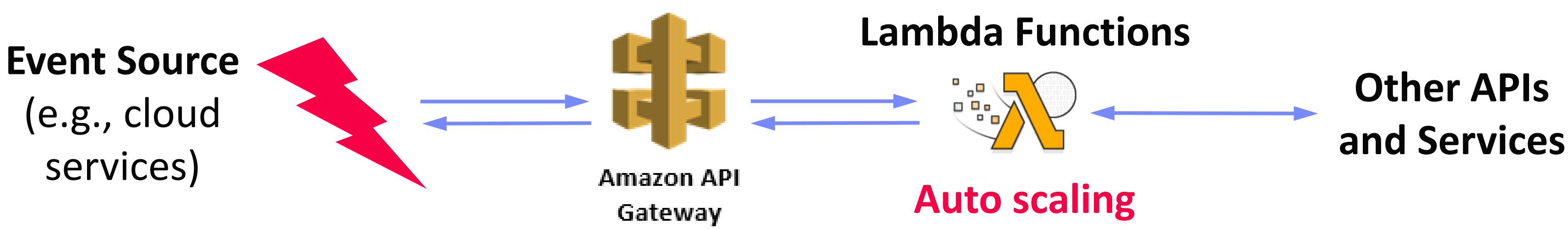
[Yunseong Lee et al.: PRETZEL: Opening the Black Box of Machine Learning Prediction Serving Systems. **OSDI 2018**]

[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



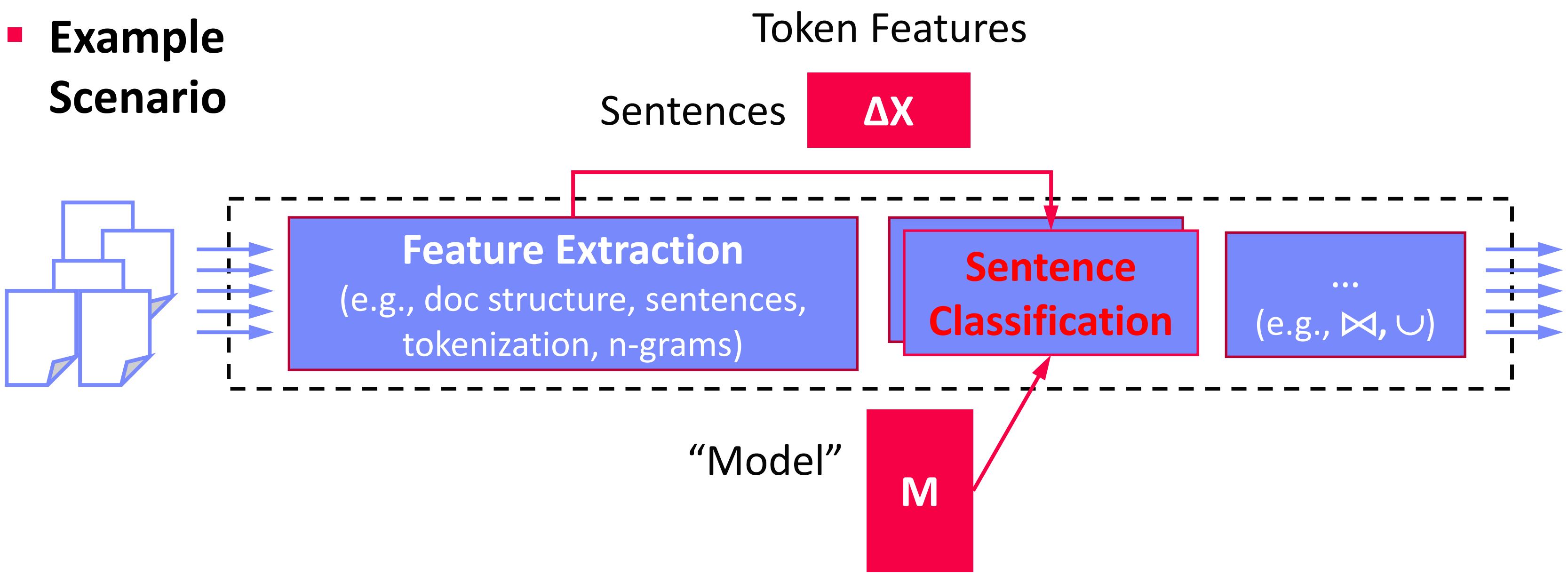
Auto scaling
Pay-per-request
(1M x 100ms = 0.2$)

- **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

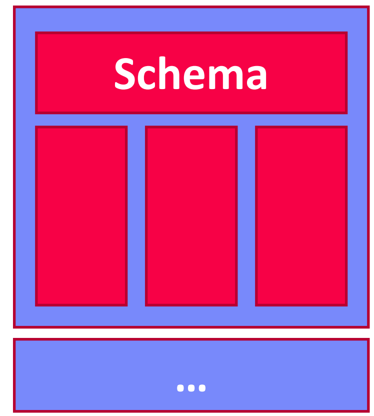
- **Example Scenario**



- **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.

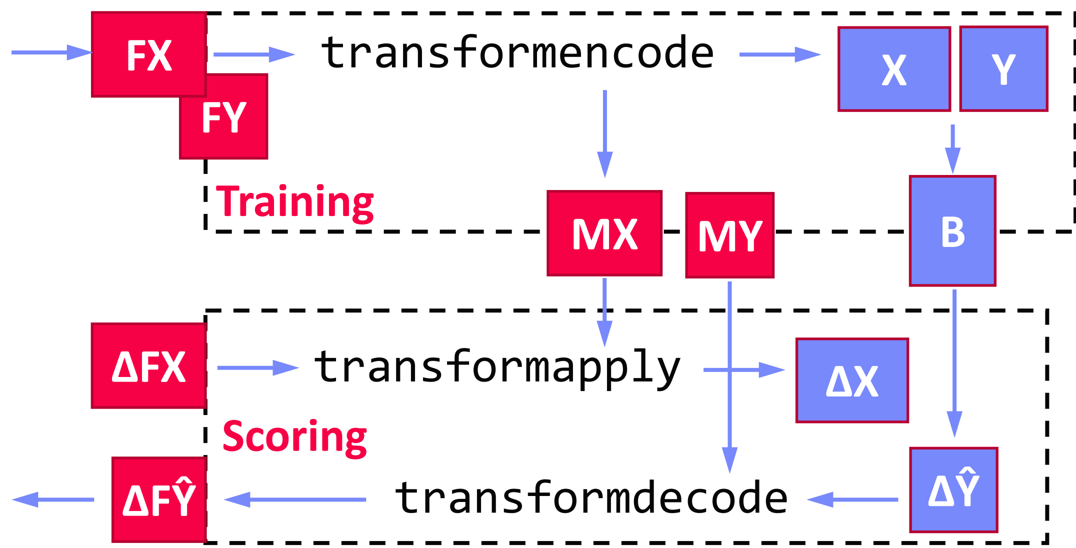
- **Background: Frame**
  - **Abstract data type with schema** (boolean, int, double, string)
  - Column-wise block layout
  - Local/distributed operations: e.g., indexing, append, **transform**



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:

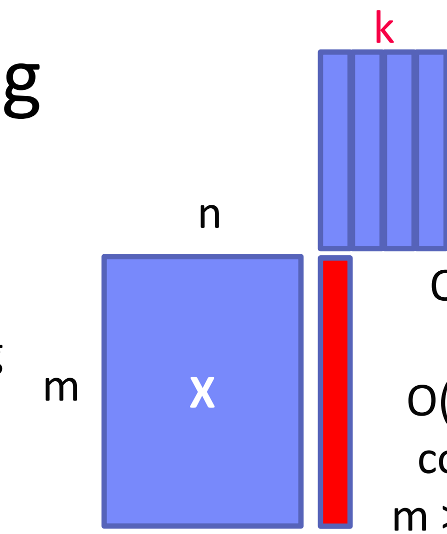
Script parsing and config:	**~100ms**
Validation, compile, IPA:	**~10ms**
HOP DAG (re-)compile:	**~1ms**
Instruction execute:	**<0.1µs**

- **Example**

```
                                          // single-node, no evictions,
1: Connection conn = new Connection();    // no recompile, no multithread.
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: }
                                          // execute precompiled script
                                          // many times
```

Serving Optimizations – Batching

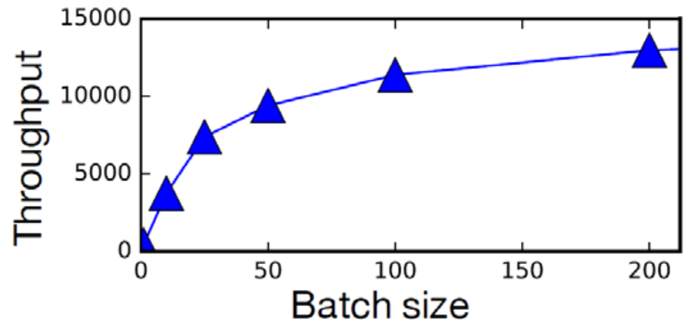
- **Recap: Model Batching** (see 08 Data Access)
  - One-pass evaluation of multiple configurations
  - EL, CV, feature selection, hyper parameter tuning
  - E.g.: TUPAQ [SoCC'16], Columbus [SIGMOD'14

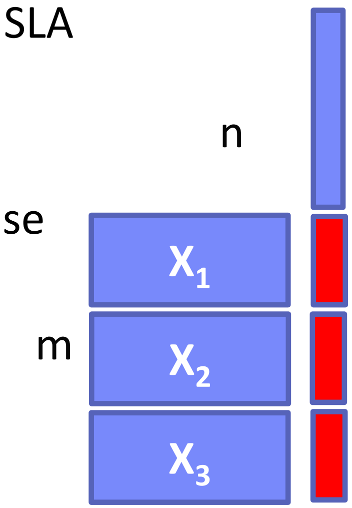


O(m*n) read
O(m*n***k**) compute
m >> n >> k

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

[Clipper @ NSDI'17]





Benefits for multi-class / complex models

Serving Optimizations – Quantization

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

08 Data Access Methods

[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()
```

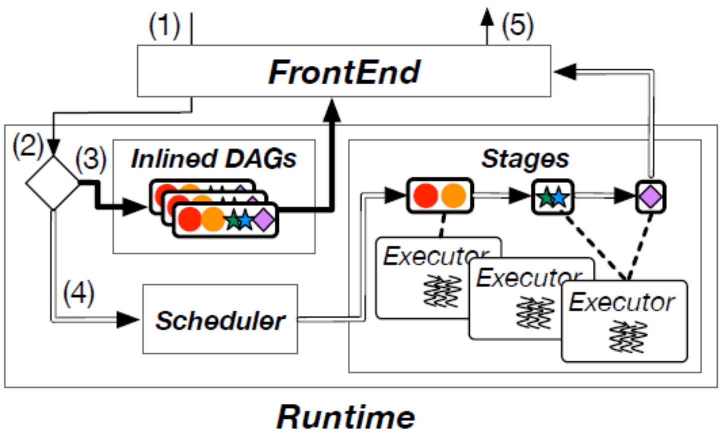
[**Credit:** https://www.tensorflow.org/lite/performance/post_training_quantization]

Serving Optimizations – MQO

- **Result Caching**
  - Establish a function cache for X → Y
    (memoization of deterministic function evaluation)

```
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
  - Done during compilation or runtime
  - 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

04 Adaptation, Fusion, and JIT

- **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`

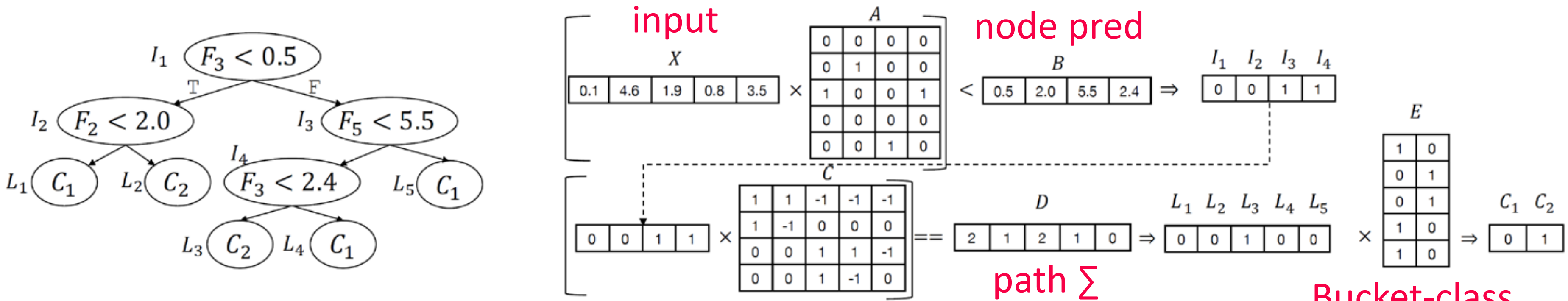
```
a = torch.rand(5)
def func(x):
  for i in range(10):
    x = x * x # unrolled into graph
  return x

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

- **HummingBird** [https://github.com/microsoft/hummingbird]
  - Compile ML scoring pipelines into tensor ops
  - Tree-based models (**GEMM**, 2x tree traversal)

[Supun Nakandala et al: A Tensor Compiler for Unified Machine Learning Prediction Serving. **OSDI 2020**]



- **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)

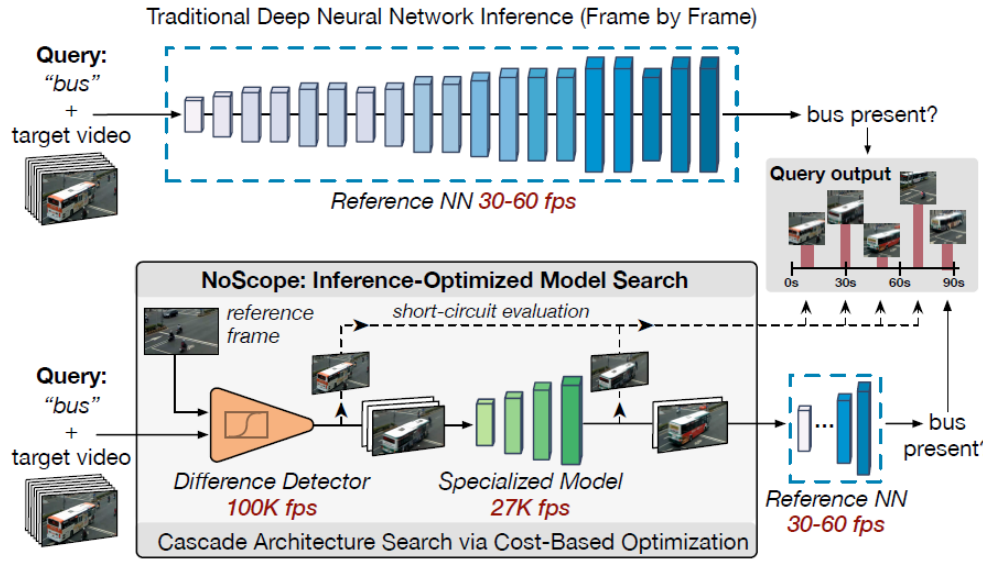
[Geoffrey E. Hinton, Oriol Vinyals, Jeffrey Dean: Distilling the Knowledge in a Neural Network. **CoRR 2015**]

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

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

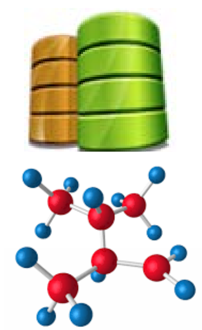
Data Integration
Data Cleaning
Data Preparation

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

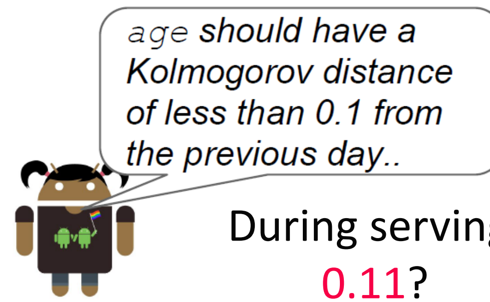
- **Goals:** Robustness (e.g., data, latency) and **model accuracy**

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

- **#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**)



During serving: 0.11?

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

"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**

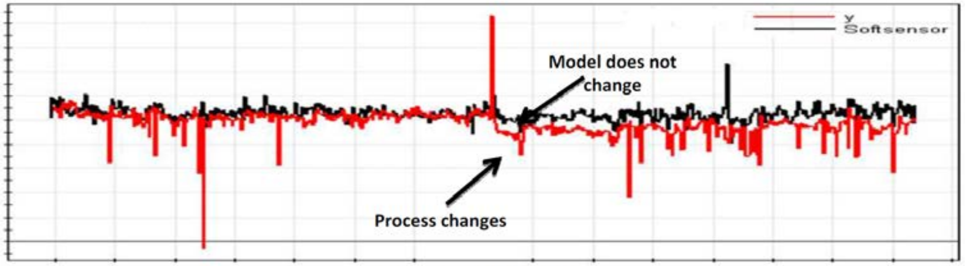
Concept Drift

[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**]


Handling Concept Drift:
Importance, Challenges & Solutions

- **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 alters
  - **Adaptive Windowing (ADWIN):** window W, append data to W, drop old values until avg windows W=W1-W2 similar (below epsillon), 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**]

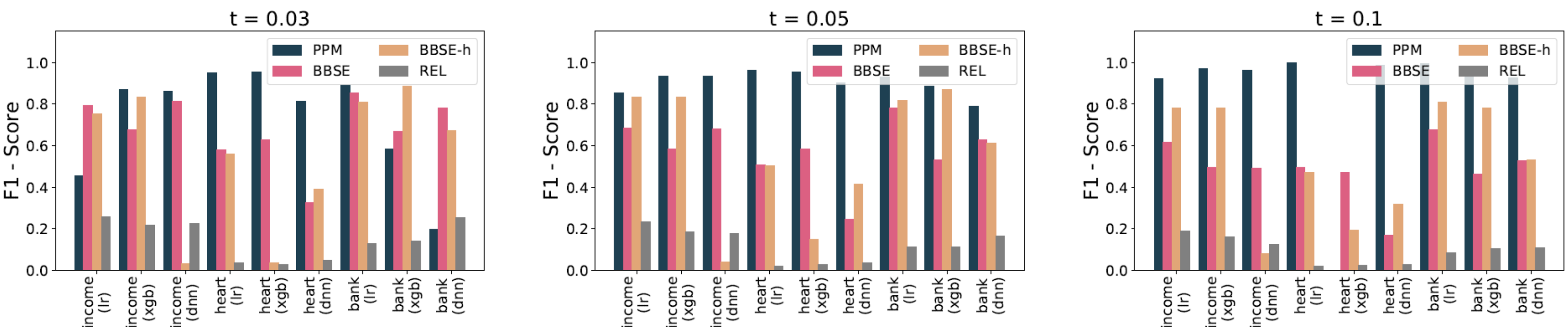
[https://scikitmultiflow.readthedocs.io/en/stable/api/generated/skmultiflow.drift_detection.ADWIN.html]

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**



GDPR (General Data Protection Regulation)

- **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)

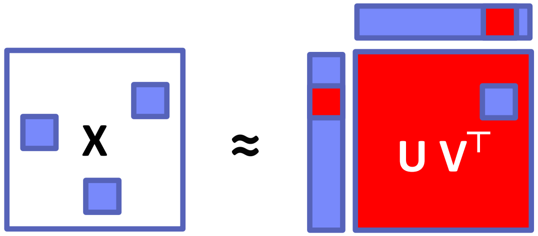
  [https://gdpr.eu/article-17-right-to-be-forgotten/]

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).

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

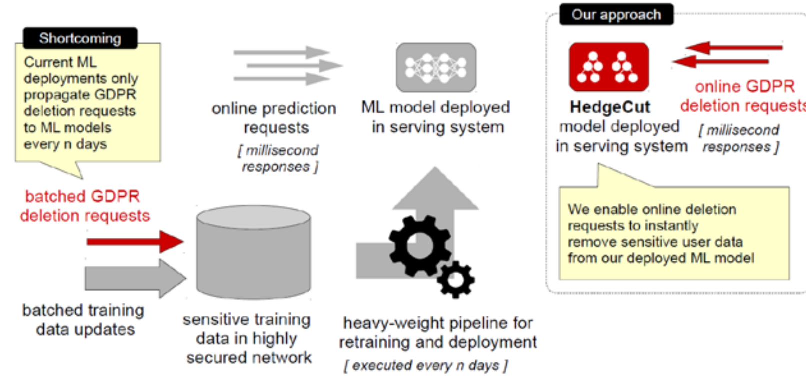


[Sebastian Schelter: "Amnesia" - Machine Learning Models That Can Forget User Data Very Fast. **CIDR 2020**]

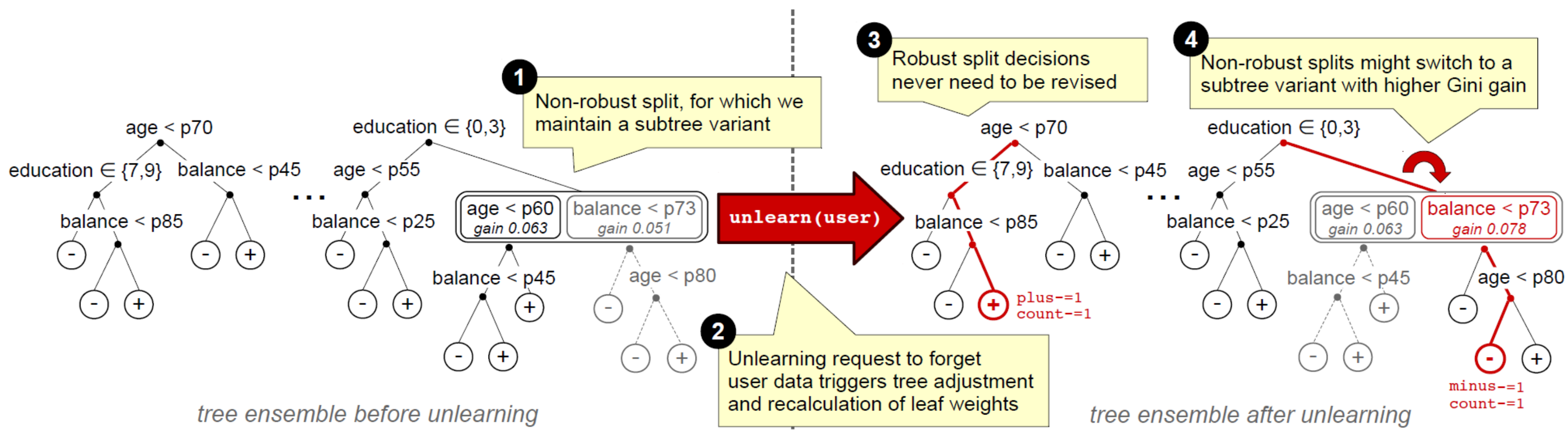
GDPR, cont.

[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 and Conclusions

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

- **#1 Finalize Programming Projects by Jun 17 EOD**
- **#2 Oral Exam**
  - Doodle for oral exam slots until **Jun 17 EOD**
  - **Part 1:** Describe you programming project, warm-up questions
  - **Part 2:** Questions on 2-3 topics of lectures 02 - 12
    (basic understanding of the discussed topics / techniques)