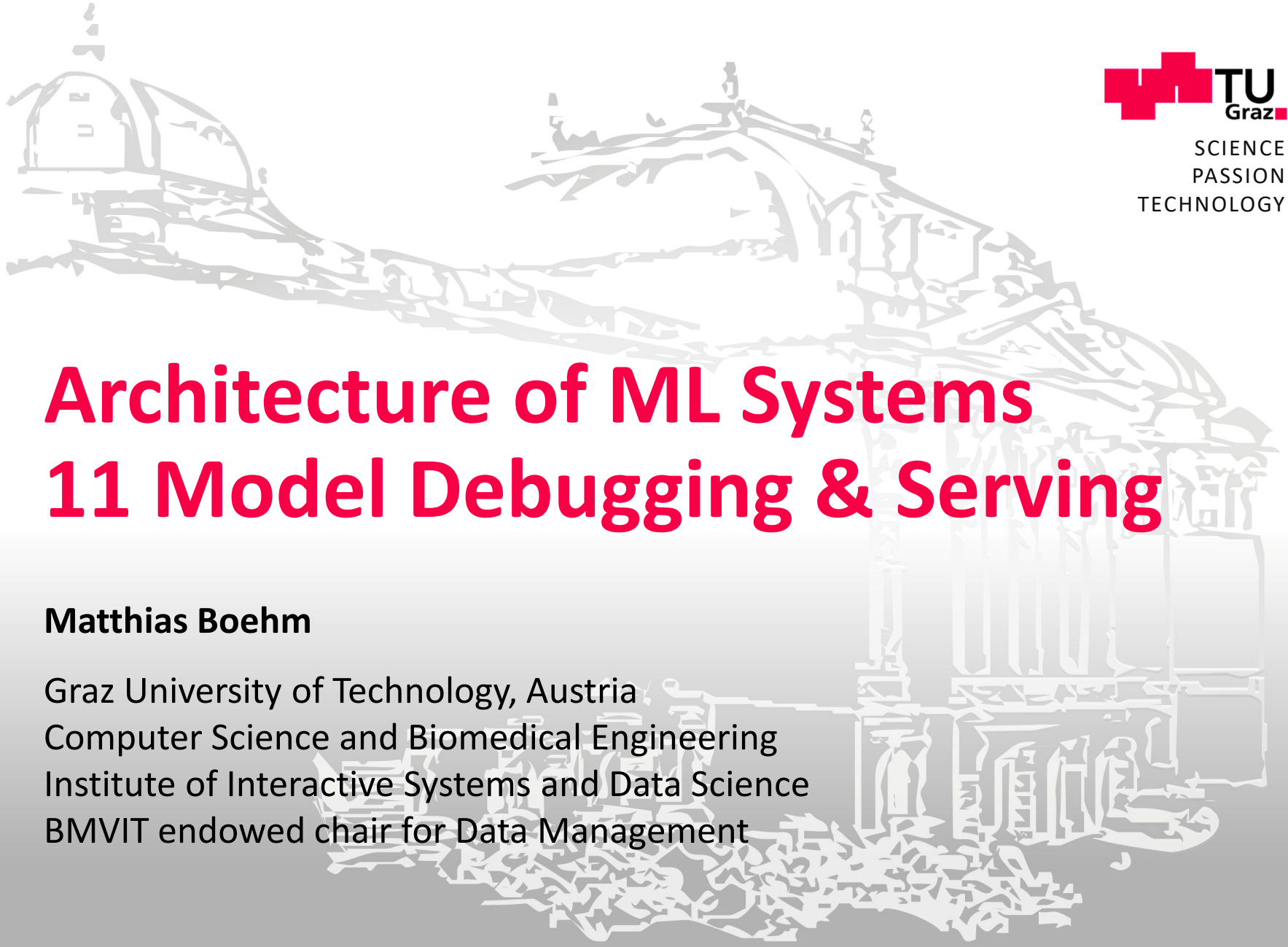


SCIENCE
PASSION
TECHNOLOGY

Architecture of ML Systems
11 Model Debugging & Serving

Matthias Boehm

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

Last update: June 18, 2020

ISDS

Announcements/Org

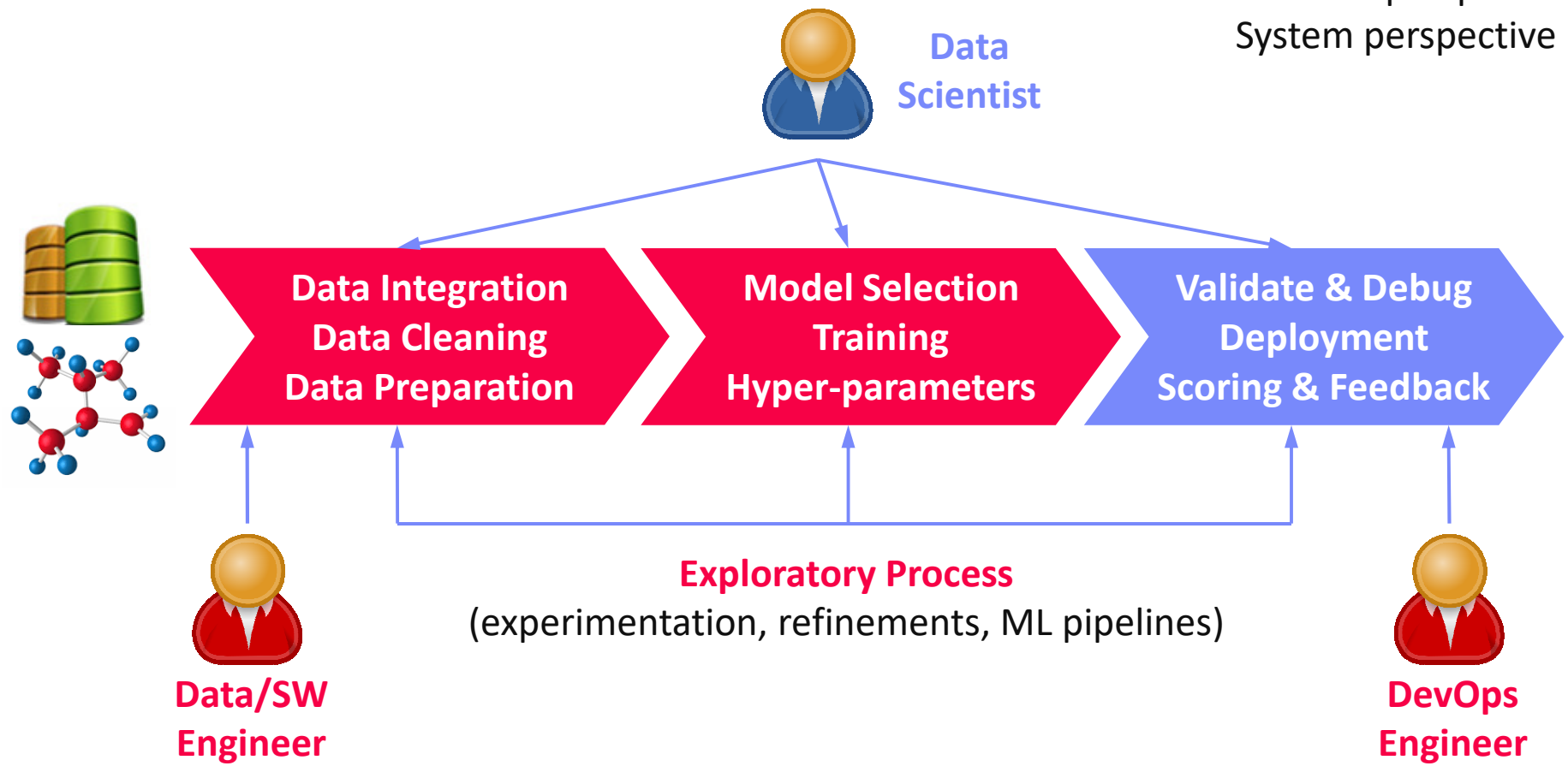
- **#1 Video Recording**
  - Link in TeachCenter & TUbe (lectures will be public)
  - **Live streaming through TUbe**, since May 08
  - Questions: https://tugraz.webex.com/meet/m.boehm

- **#2 AMLS Programming Projects**
  - **Status:** all project discussions w/ **15 students** (~**10 PRs**)
  - Soft deadline: **June 30**
  - If unable to complete: email to m.boehm@tugraz.at
  - **Doodle for oral exam slots** July 2/3 (1 done + 8 scheduled)

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

Recap: The Data Science Lifecycle

Data-centric View:
Application perspective
Workload perspective
System perspective



Agenda

- **Model Debugging and Validation**
- **Model Deployment and Serving**

Model Debugging and Validation

Similar to Software Testing

Focus on Benchmarks, Assessment, Monitoring, Trust, Finding Room for Improvements

Recap: Data Validation

Sanity checks on expected shape before training first model

[Neoklis Polyzotis, et al: Data Management Challenges in Production Machine Learning. Tutorial, **SIGMOD 2017**]

(**Google Research**)

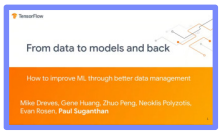
- **Check a feature's min, max, and most common value**
  - Ex: Latitude values must be within the range [-90, 90] or [-π/2, π/2]
- **The histograms of continuous or categorical values are as expected**
  - Ex: There are similar numbers of positive and negative labels
- **Whether a feature is present in enough examples**
  - Ex: Country code must be in at least 70% of the examples
- **Whether a feature has the right number of values (i.e., cardinality)**
  - Ex: There cannot be more than one age of a person

- **Others**

[Sebastian Schelter et al: Automating Large-Scale Data Quality Verification. **PVLDB 2018**]

(**Amazon Research**)

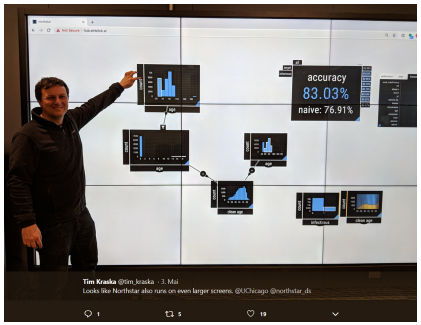
[Mike Dreves et al: From Data to Models and Back **DEEM@SIGMOD 2020,** http://deem-workshop.org/videos/2020/8_dreves.mp4]



(**Google**)

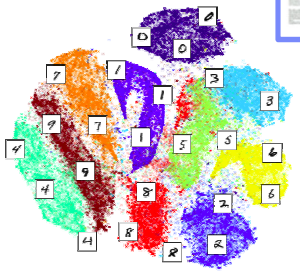
Overview Model Debugging

[**Credit:** twitter.com/tim_kraska]



[Andrew Crotty et al: Vizdom: Interactive Analytics through Pen and Touch. **PVLDB 2015**]

- **#1 Understanding via Visualization**
  - Plotting of predictions / interactions
  - Combination with dimensionality reduction into 2D:
    - **Autoencoder**
    - **PCA** (principal component analysis)
    - **t-SNE** (T-distributed Stochastic Neighbor Embedding)
  - Input, intermediate, and output layers of DNNs



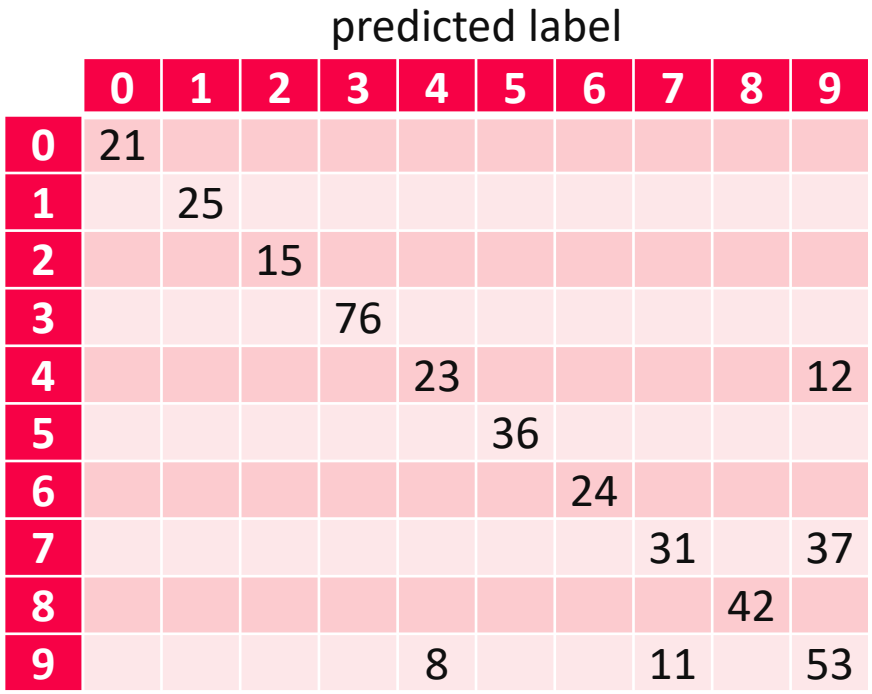
[**Credit:** nlml.github.io/in-raw-numpy/in-raw-numpy-t-sne/]

- **#2 Fairness, Explainability, and Validation via Constraints**
  - Impose constraints like monotonicity for ensuring fairness
  - Generate succinct representations (e.g., rules) as explanation
  - Establish assertions and thresholds for automatic validation and alerts

Basic Model-Specific Statistics

- **Regression Statistics**
  - Average response and stddev, average residuals stddev residuals
  - R2 (coeff of determination) with and without bias, etc

- **Classification Statistics**
  - Classical: recall, precision, F1-score
  - Visual: **confusion matrix** (correct vs predicated classes) ➔ understand performance wrt individual classes
  - Example Mnist
  - Mispredictions might also be visualized via dimensionality reduction



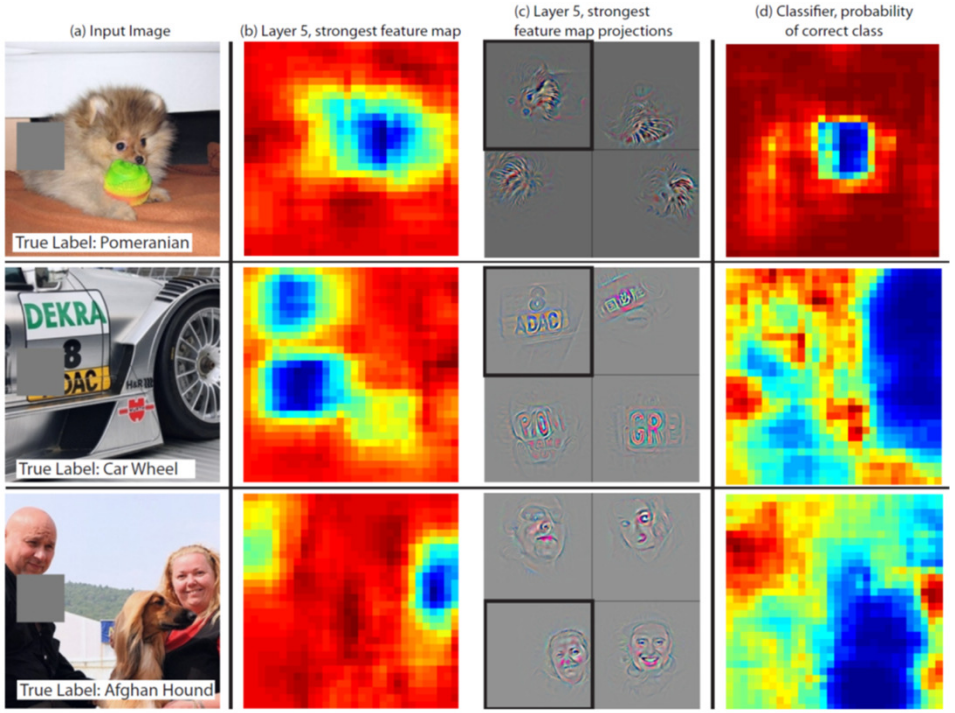
Understanding Other Basic Issues

- **Overfitting / Imbalance**
  - Compare train and test performance
  - ➔ **Algorithm-specific techniques:** regularization, pruning, loss, etc
- **Data Leakage**
  - Example: time-shifted external time series data (e.g., weather)
  - **Compare performance train/test vs production setting**
- **Covariance Shift** (features)
  - Distribution of features in training/test data different from production data
  - Reasons: **out-of-domain prediction, sample selection bias**
  - Examples: NLP, speech recognition, face/age recognition
- **Concept Drift** (features → labels)
  - **Gradual change of statistical properties** / dependencies (features-labels)
  - Requires re-training, parametric approaches for deciding when to retrain

Occlusion-Based Explanations

- **Occlusion Explanations**
  - Slide gray square over inputs
  - Measure how feature maps and classifier output changes

[Matthew D. Zeiler, Rob Fergus: Visualizing and Understanding Convolutional Networks. **ECCV 2014**]



- **Incremental Computation of Occlusion Explanations**
  - View CNN as white-box operator graph and operators as views
  - Materialize intermediate tensors and apply **incremental view maintenance**

[Supun Nakandala, Arun Kumar, and Yannis Papakonstantinou: Incremental and Approximate Inference forFaster Occlusion-based Deep CNN Explanations, **SIGMOD 2019**]

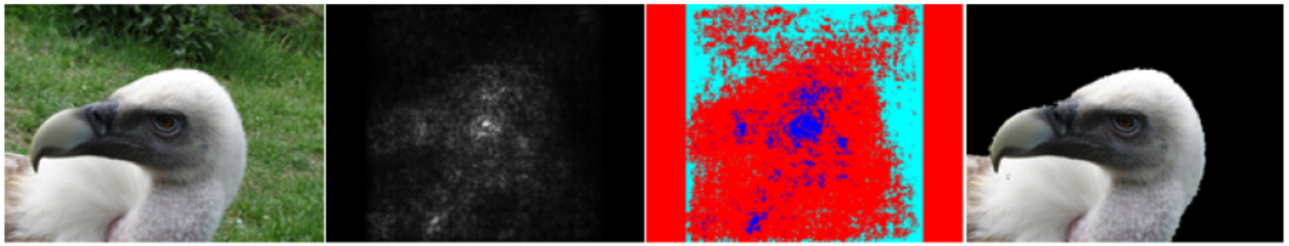
SIGMOD 2020 Research Highlight

Saliency Maps

[Karen Simonyan, Andrea Vedaldi, Andrew Zisserman: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. **ICLR Workshop 2014**]

- **Saliency Map**
  - Given input image and specific class
  - Compute saliency map of **class derivatives wrt input image**
  - Approximated w/ a linear function (Taylor expansion)

- **Unsupervised Image Segmentation**

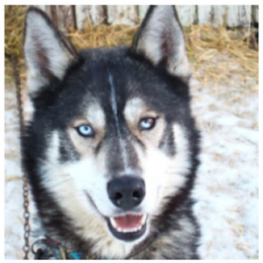


Example Model Anomalies

"silent but severe problems"

- **#1 Wolf Detection based on snow cover**

[Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin: Why Should I Trust You?: Explaining the Predictions of Any Classifier, **KDD 2016**]



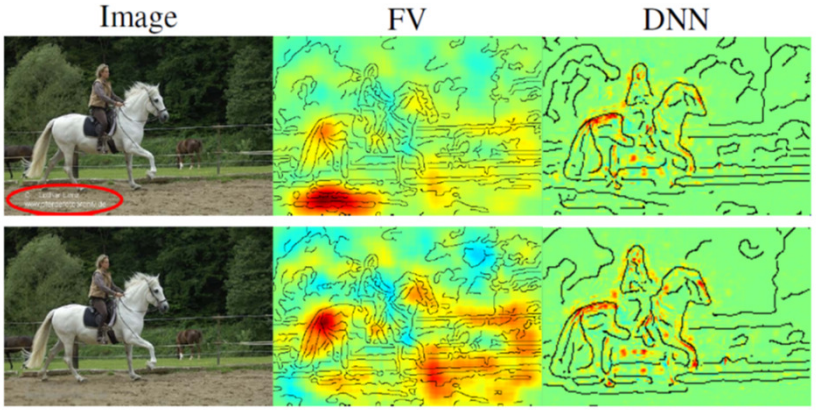
(a) Husky classified as wolf

(b) Explanation

12/27 → 25/27

- **#2 Horse Detection based on image watermarks**
  - Layer-wise relevance propagation

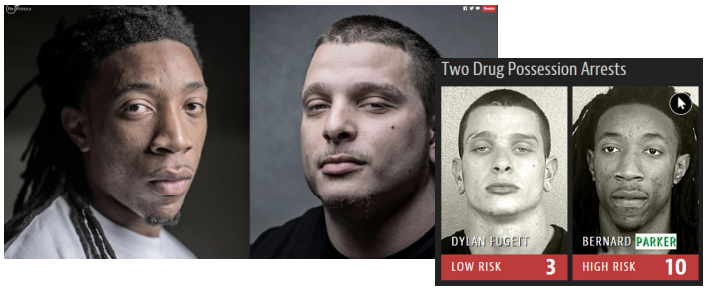
[Sebastian Lapuschkin et al.: Analyzing Classifiers: Fisher Vectors and Deep Neural Networks, **CVPR 2016**]



- **#3 Race-biased Jail Risk Assessment**

#BlackLivesMatter

[Julia Angwin et al: Machine Bias – There's software used across the country to predict future criminals. And it's biased against blacks, **2016**, https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing]

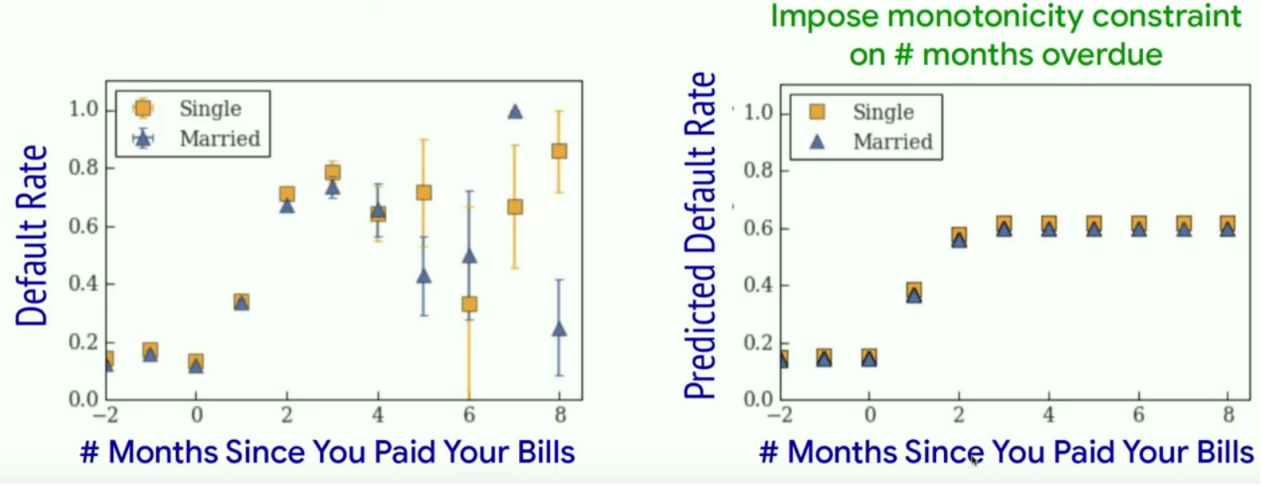


Debugging Bias and Fairness

- **Fairness**
  - Validate and ensure fairness with regard to sensitive features (unbiased)
  - Use **occlusion and saliency maps** to characterize and compare groups

- **Enforcing Fairness**
  - Use constraints to enforce certain properties (e.g., monotonicity, smoothness)
  - Example: late payment → credit score

[Maya Gupta: How Do We Make AI Fair? **SysML 2019**]



Explanation Tables

- **Motivation**
  - Generate **succinct decision rules** from data
  - **Problem:** Decision tree rules do not overlap by def
  - Example athlete's exercise log: "Goal met" → 7 vs 7
- **Explanation Tables**
  - **Find smallest explanation** table subject to max KL divergence threshold
  - Greedy and sampling algorithms

[Kareem El Gebaly, Parag Agrawal, Lukasz Golab, Flip Korn, Divesh Srivastava: Interpretable and Informative Explanations of Outcomes. **PVLDB 2014**]

id	day	time	meal	goal met?
1	Fri	Dawn	Banana	Yes
2	Fri	Night	Green salad	Yes
3	Sun	Dusk	Oatmeal	Yes
4	Sun	Morning	Banana	Yes
5	Mon	Afternoon	Oatmeal	Yes
6	Mon	Midday	Banana	Yes
7	Tue	Morning	Green salad	No
8	Wed	Night	Burgers	No
9	Thu	Dawn	Oatmeal	Yes
10	Sat	Afternoon	Nuts	No
11	Sat	Dawn	Banana	No
12	Sat	Dawn	Oatmeal	No
13	Sat	Dusk	Rice	No
14	Sat	Midday	Toast	No

day	time	meal	goal met=Yes?	count
*	*	*	.5	14
Sat	*	*	0	5
*	*	Banana	.75	4
*	*	Oatmeal	.75	4

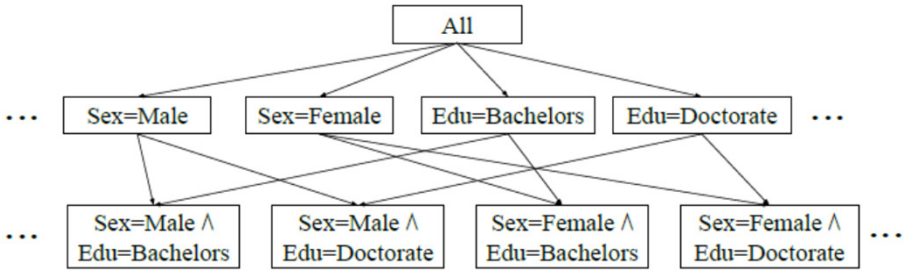
Data Slices and Automatic Slice Finding

- **Problem**
  - **Find top-k data slices where model performs worse** than average
  - Data slice: S^{DG} := D=PhD AND G=female (subsets of features)
  - Objective: **w** * err(S^{DG})/err(S^{*}) + (**1-w**) * |S^{DG}|

[Yeounoh Chung et al.: Slice Finder: Automated Data Slicing for Model Validation. CoRR 2018/**ICDE2019**]

"find largest error vs find large slices"

- **Existing Algorithms**
  - Preparation: Binning + One-Hot Encoding of features
  - Lattice search with heuristic, level-wise termination



- **Extensions**
  - **#1** Lower/upper bounds → pruning & termination
  - **#2** Large-scale task-/data-parallel computation

`SystemDS/scripts/staging/slicing`

Model Assertions

[Daniel Kang, Deepti Raghavan, Peter Bailis, Matei Zaharia: Model Assertions for Debugging Machine Learning, **NIPS Workshop ML Systems, 2018**]

- **Motivation**
  - ML models might fail in complex ways that are not captured in loss function
  - Inspired by assertions in SW dev → Model assertions via Python rules

Example: Flickering of object detection

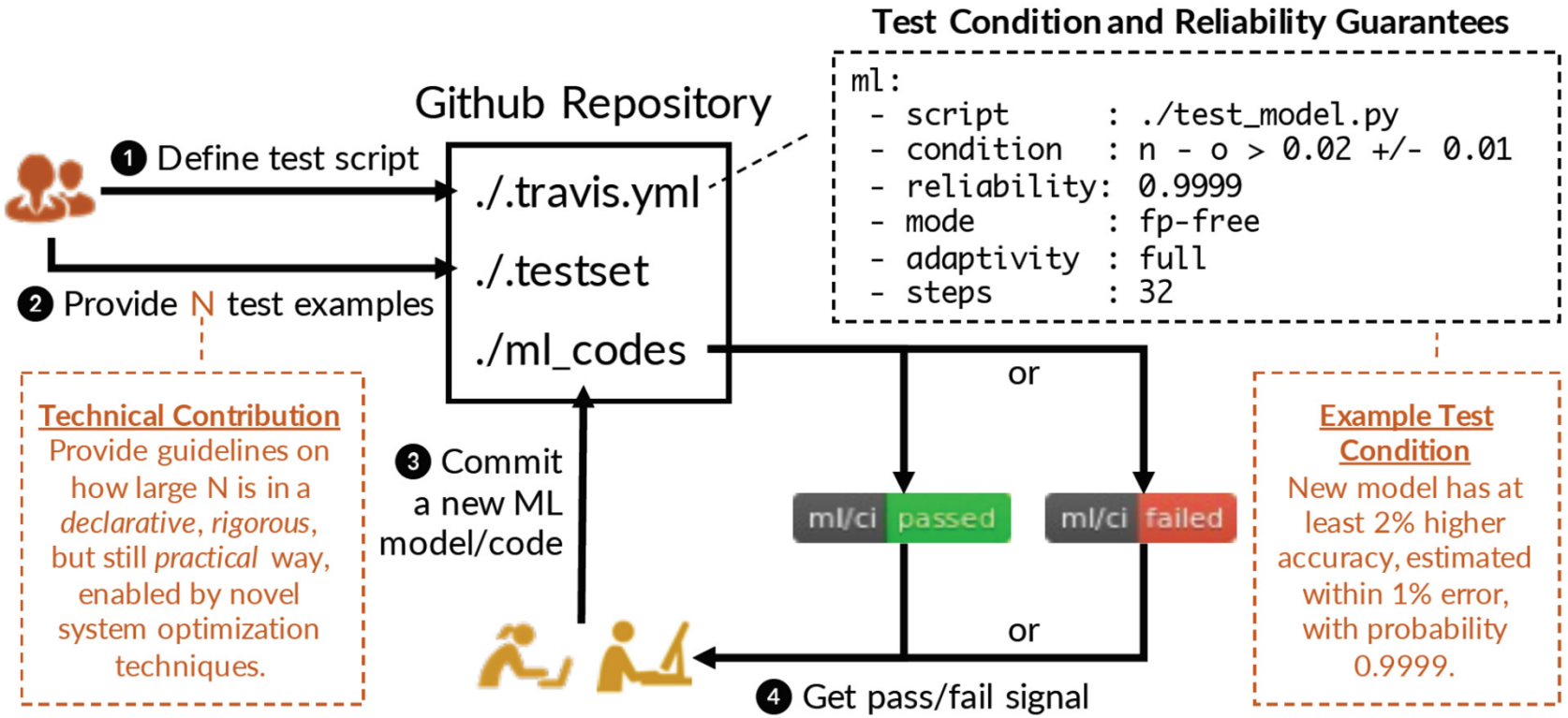
(a) Frame 1, base SSD (b) Frame 2, base SSD (c) Frame 3, base SSD

- **Assertion Use Cases**
  - **#1 Runtime monitoring** (collect statistics on incorrect behavior)
  - **#2 Corrective Action** (trigger corrections at runtime) → **but how in retrospect?**
  - **#3 Active Learning** (decide which difficult data points to give to user)
  - **#4 Weak supervision** (propose alternative labels and use for retraining)

Continuous Integration

- **System Architecture**
  ease.ml/ci

[Cedric Renggli, Bojan Karlaš, Bolin Ding, Feng Liu, Kevin Schawinski, Wentao Wu, Ce Zhang: Continuous Integration of Machine Learning Models with ease.ml/ci: Towards a Rigorous Yet Practical Treatment, **SysML 2019**]



Model Deployment and Serving

How to Exchange, Deploy, and Update Trained Models

How to Efficiently Serve Prediction Tasks

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., SystemML **binary block**)

```
%%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**]

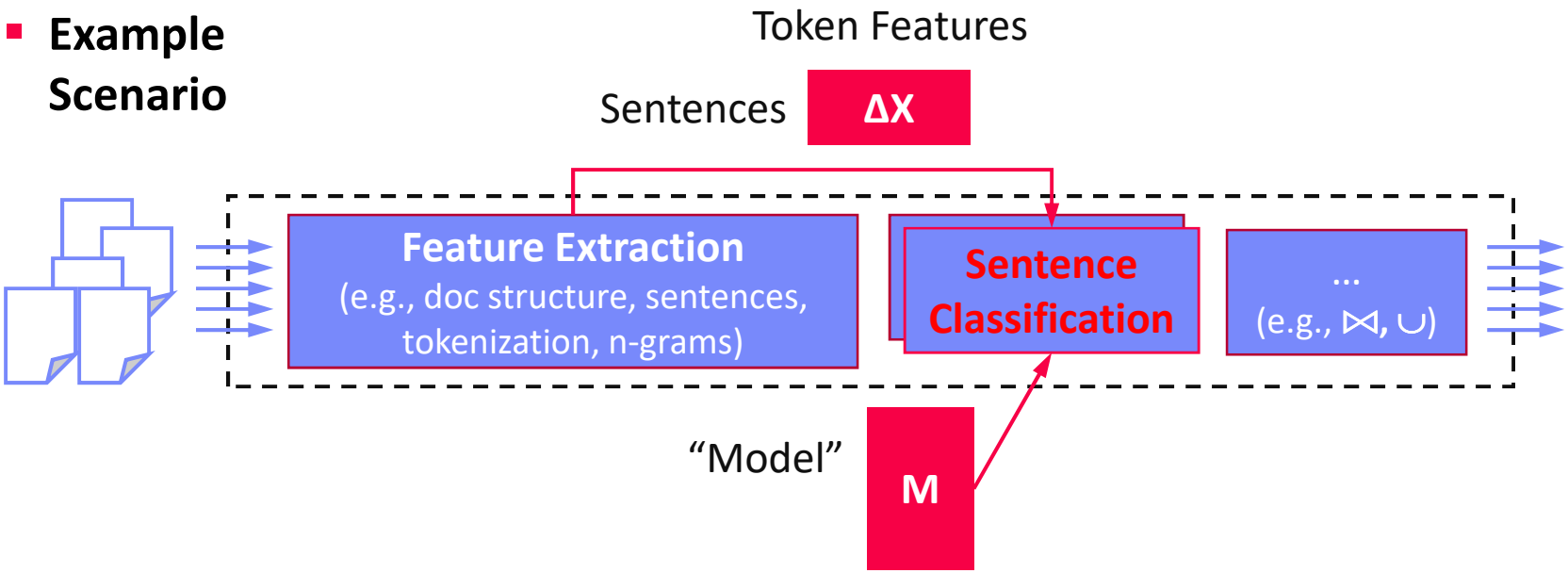
[Daniel Crankshaw et al: Clipper: A Low-Latency Online Prediction Serving System. **NSDI 2017**]

[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**]

Example SystemDS JMLC

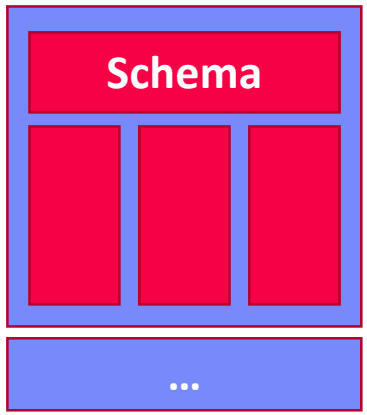
- **Example Scenario**



- **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.

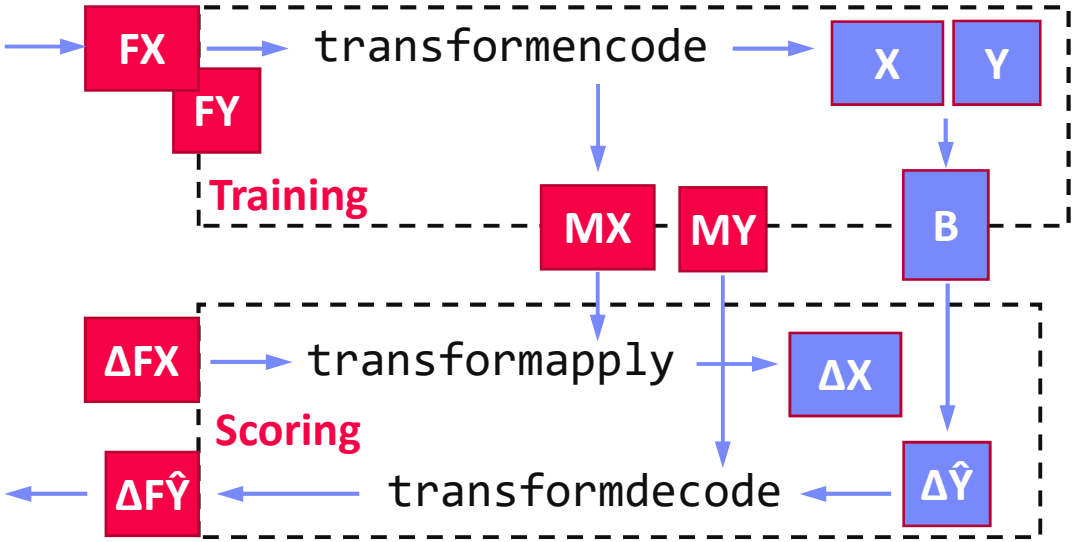
- **Background: Frame**
  - Abstract data type with schema (boolean, int, double, string)
  - Column-wise block layout
  - Local/distributed operations: e.g., indexing, append, transform



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
- **Example**

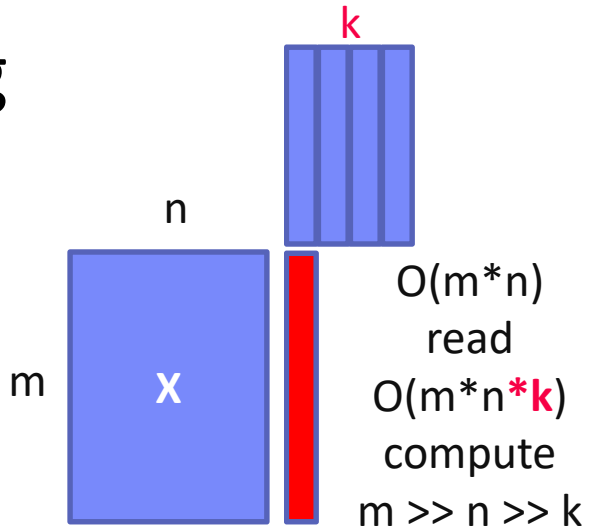
Typical compiler/runtime overheads:

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

```
1: Connection conn = new Connection();   // single-node, no evictions,
                                         // 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
```

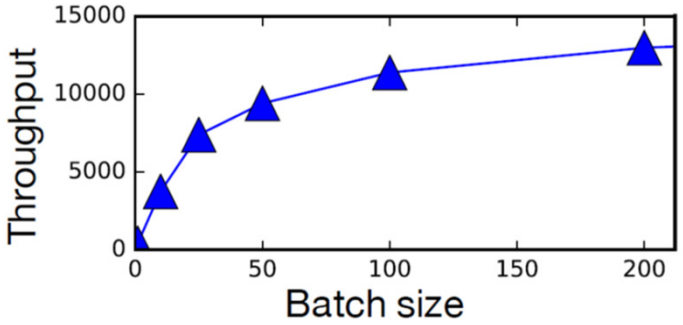
Serving Optimizations – Batching

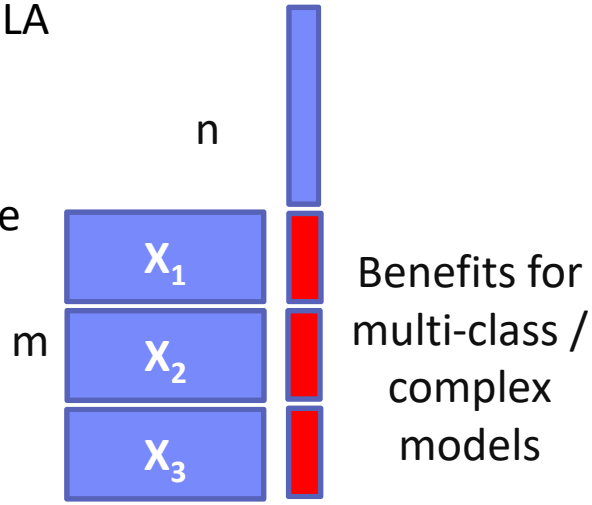
- **Recap: Model Batching** (see **08 Data Access**)
  - One-pass evaluation of multiple configurations
  - EL, CV, feature selection, hyper parameter tuning
  - E.g.: **TUPAQ** [SoCC'16], **Columbus** [SIGMOD'14



- **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]



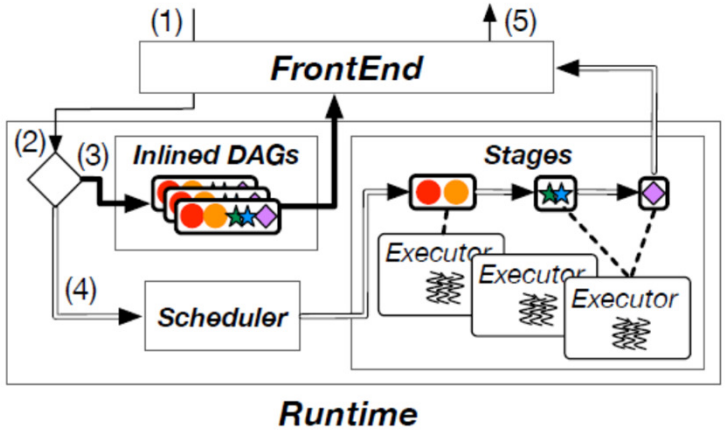


Serving Optimizations – MQO

- **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 – 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 – 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`

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

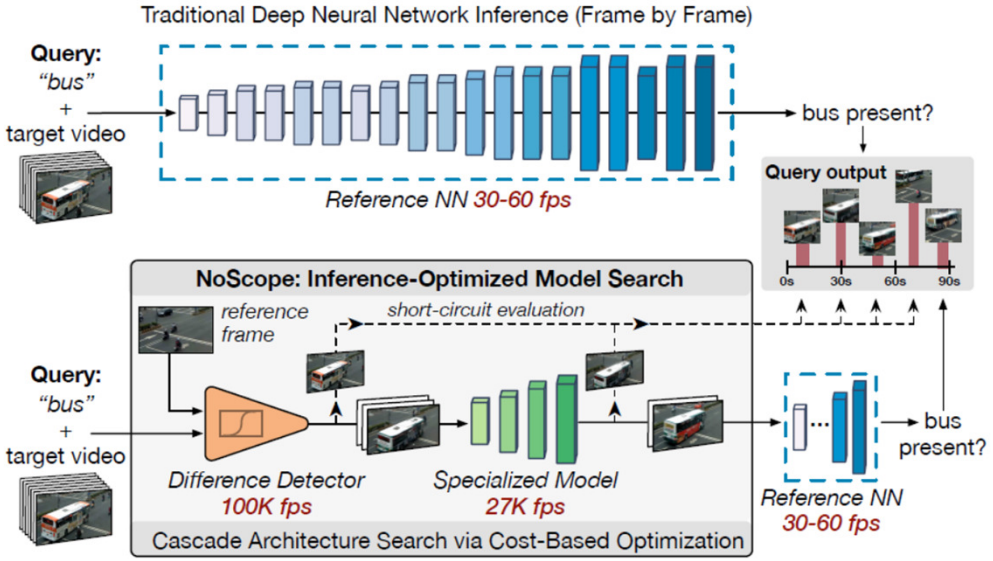
```
a = torch.rand(5)
def func(x):
  for i in range(10):
    x = x * x # unrolled into graph
  return x

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

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

Summary and Conclusions

- **Summary 11 Model Deployment**
  - Model Debugging and Validation
  - Model Deployment and Serving

- **#1 Finalize Programming Projects by ~June 30**

- **#2 Oral Exam**
  - Doodle for **July 2/3**, 45min each (done via skype/webex)
  - **Part 1:** Describe you programming project, ram-up questions
  - **Part 2:** Questions on 2-3 topics of 11 lectures
    (basic understanding of the discussed topics / techniques)