From Raw to Ready: Optimizing Data Curation for Machine Learning

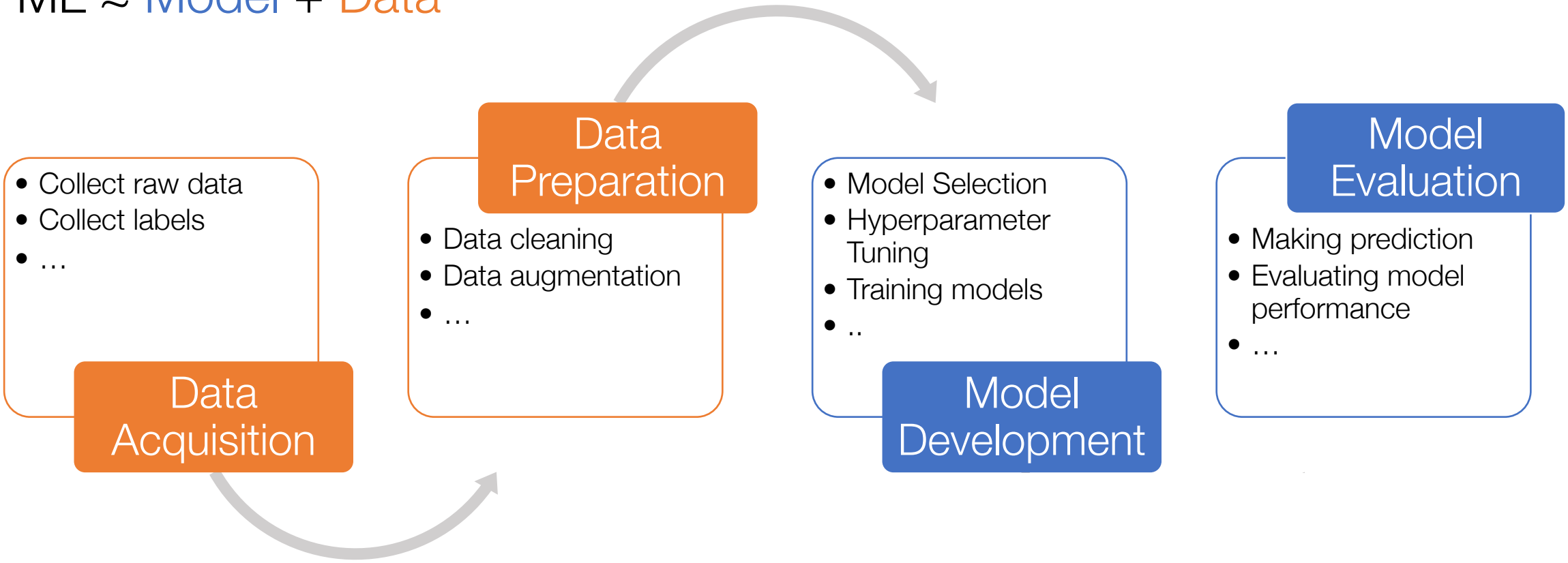
Kexin Rong

VRG, 07/11/24

The ML lifecycle in a bird's eye view

"Only a fraction of real-world ML systems is composed of ML code" [1]

ML ≈ Model + Data



[1] Sculley, David, et al. "Hidden technical debt in machine learning systems." NeurIPS 2015

Data is the Bottleneck for ML

ML ≈ Model + Data

Model is gradually commoditized

- Transformers for “all” tasks
- Out-of-the-box invocation of ML libraries gives decent results

Data remains the bottleneck

- Collecting and storing raw da
- Turning them into ML-ready d

What is hard about data?

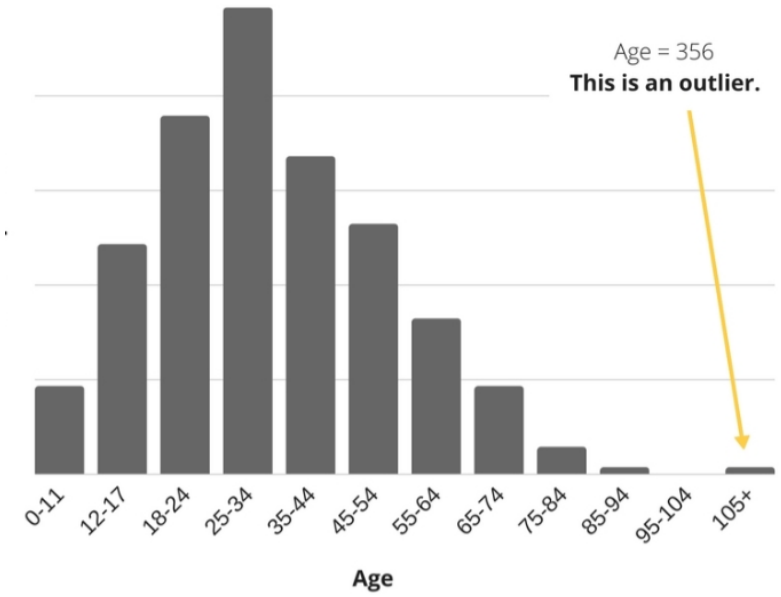
Sources:
https://www.techspot.com/news/98600-chatgpt-powered-hidden-army-contractors-making-15-hour.html

Challenge #1: Raw data often requires significant human cleaning efforts

ML demands high quality data, but real-world data often contains errors

Country	UN R/P 10%[4]	UN R/P 20%[5]	World Bank Gini (%)[6]
Seychelles			65.8
Comoros			64.3
Namibia	106.6	56.1	63.9
South Africa	33.1	17.9	63.1
Botswana	43.0	20.4	61.0
Haiti	54.4	26.6	59.2
Angola			58.6
Honduras	59.4	17.2	57.0

Missing Values



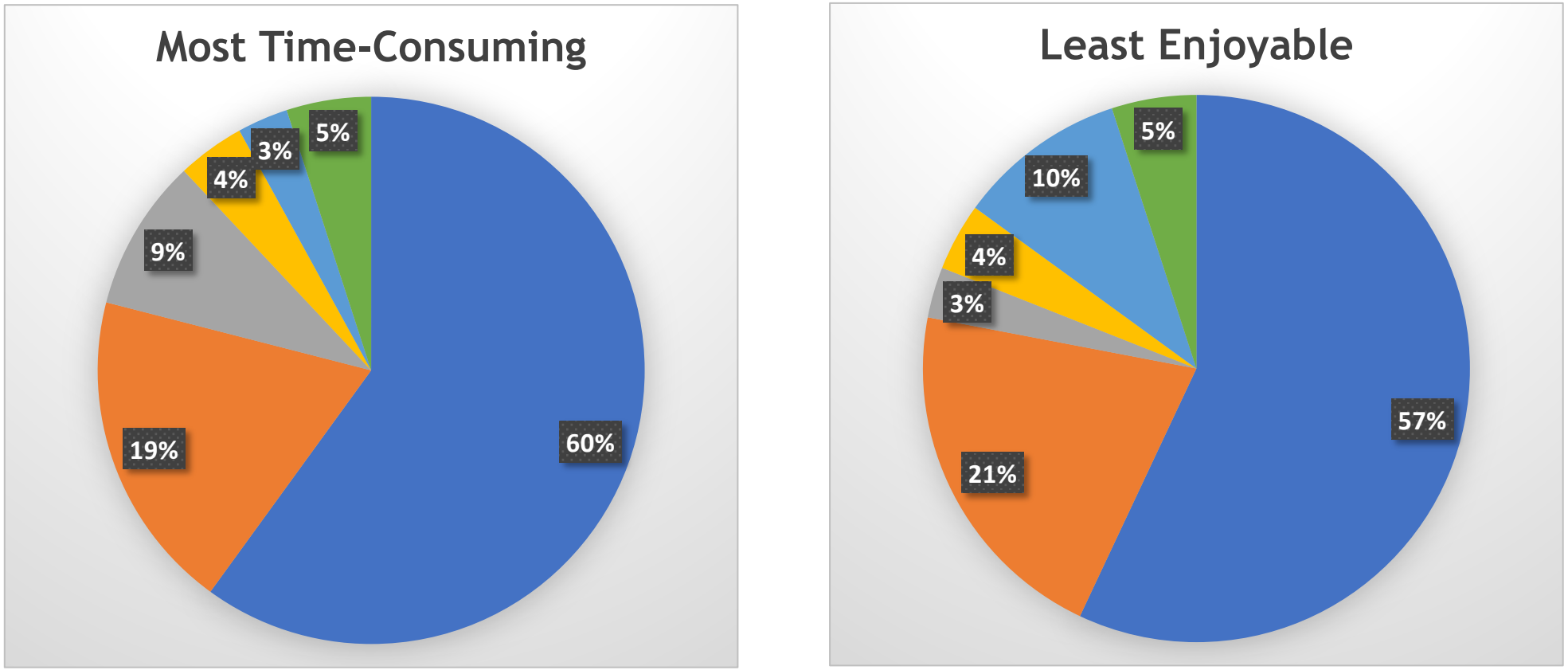
Outliers

Label: Fox

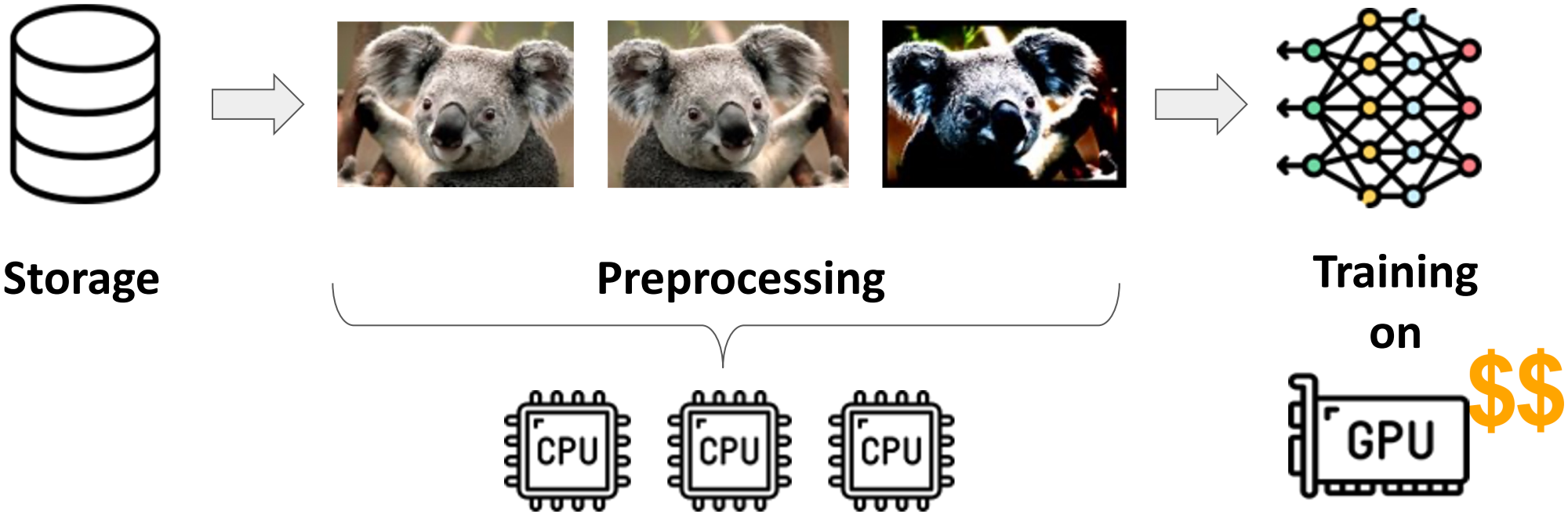
Mislabels

Data Cleaning: Most Time-Consuming, Least Enjoyable Data Science Task

Forbes, 2016



Challenge #2: Data preprocessing can also be a computation bottleneck



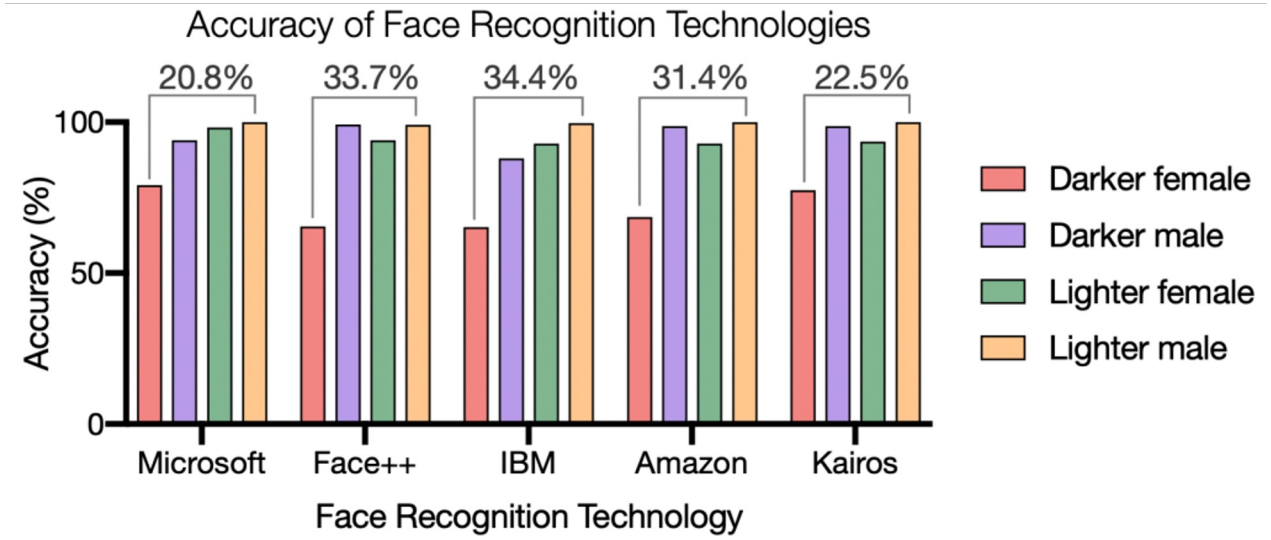
65% of epoch time

Analyzing and Mitigating Data Stalls in DNN Training, VLDB 2021

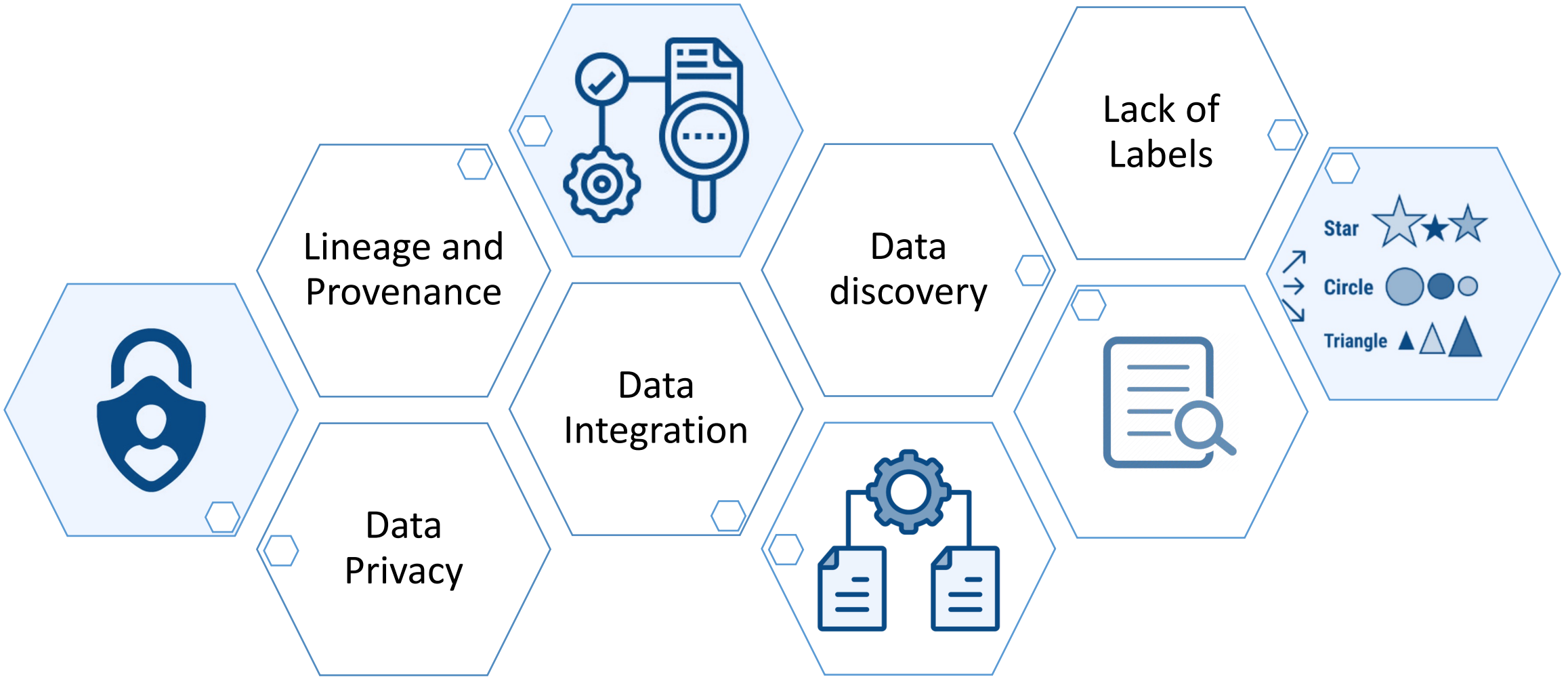
Challenge #3: Model Amplifies Data Biases

The model perpetuates and amplify human biases by aggregating the data, resulting in a model that only performs well for the majority.

Example: Buolamwini and Gebru (2018). Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification



And many more…



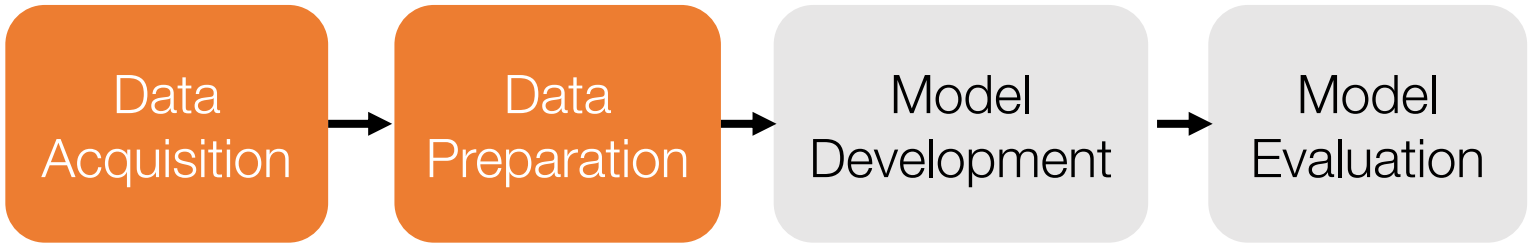
The data-centric perspective

Data curation is becoming a critical bottleneck in the ML lifecycle

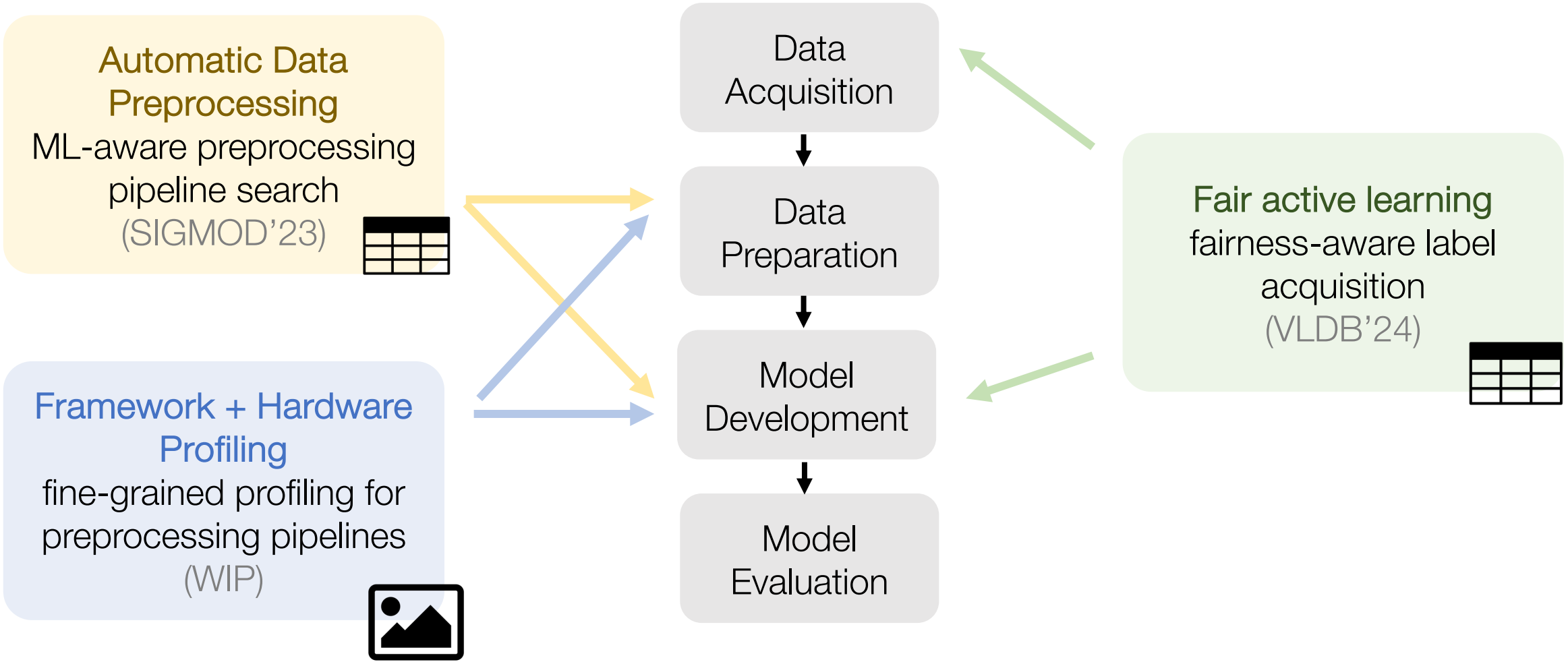
- High-quality, ML-ready datasets

By optimizing data curation, we have opportunities to

- Save human time
- Save compute resources
- Even get a better model

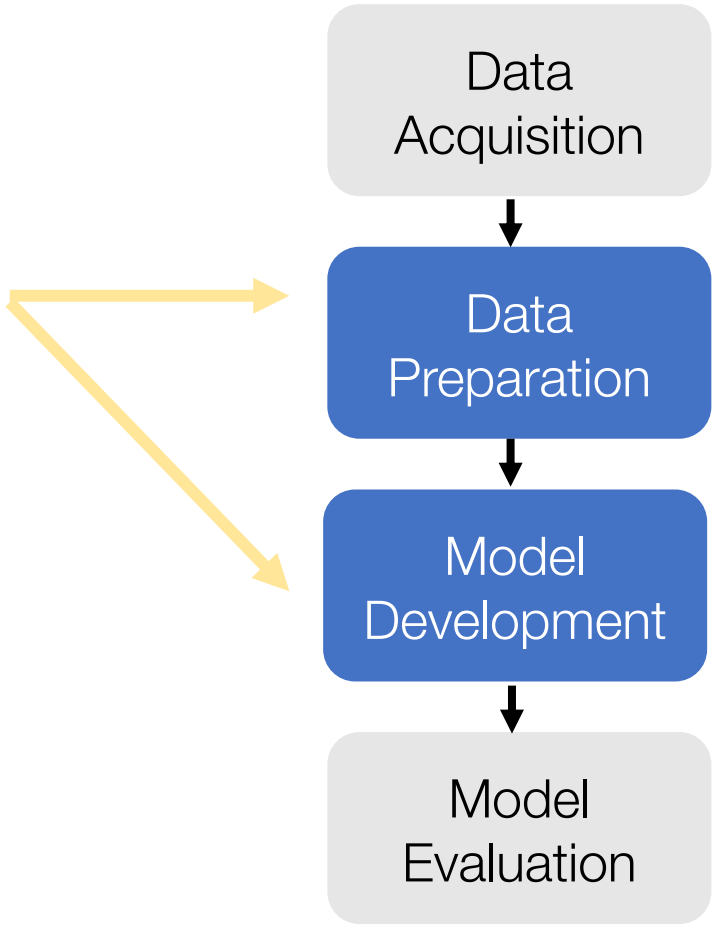


This talk: Data curation for ML



This talk: Data curation for ML

Automatic Data Preprocessing
ML-aware preprocessing pipeline search
(SIGMOD'23)



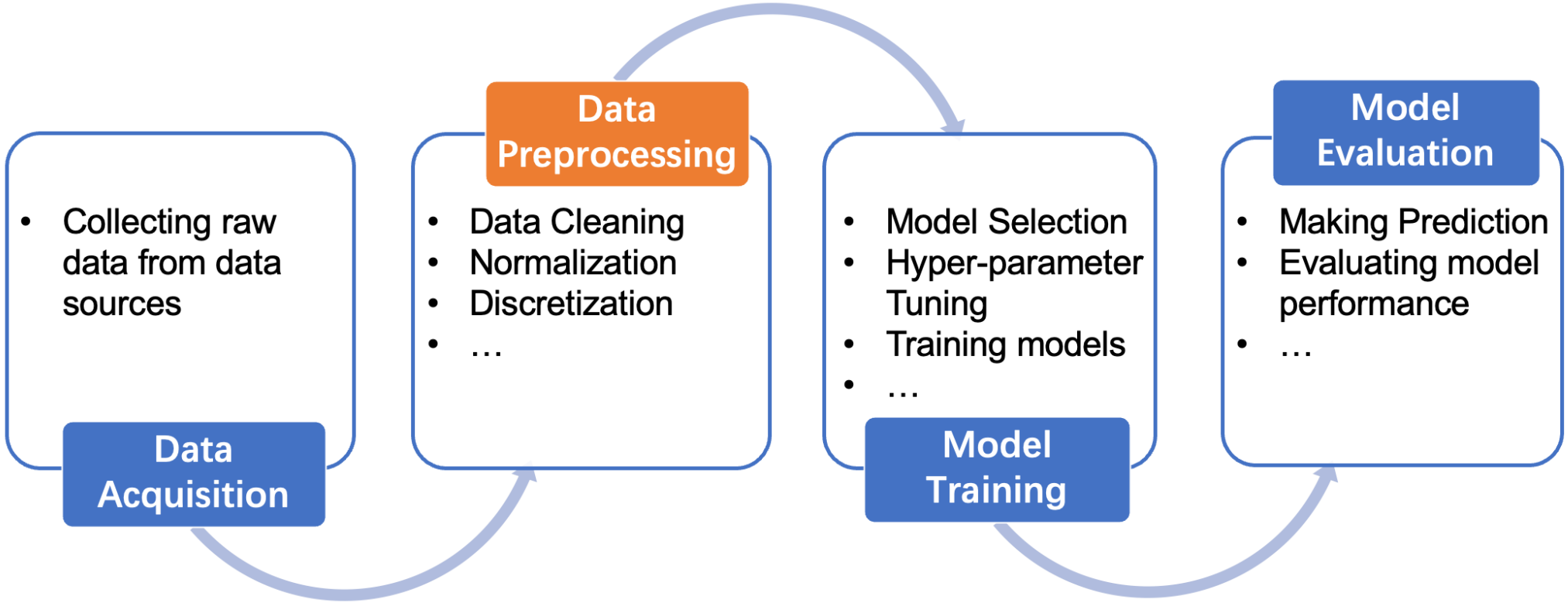
DiffPrep: Differentiable Data Preprocessing Pipeline Search for Learning over Tabular Data

With Peng Li, Zhiyi Chen and Xu Chu

Georgia Tech

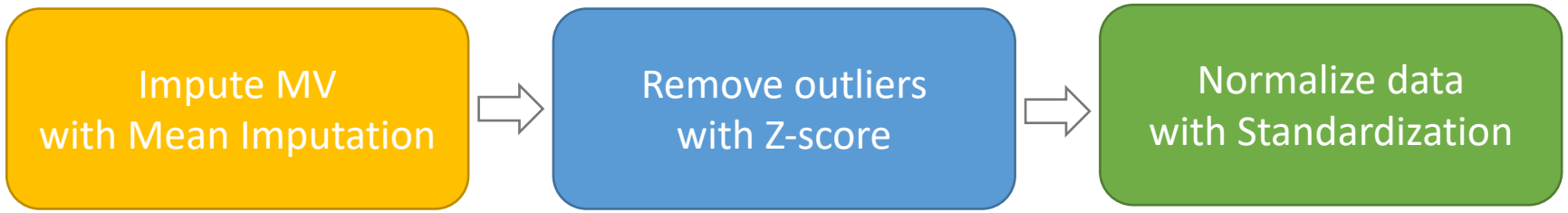
Data preprocessing is an essential step in ML

Raw data collected from data sources can rarely be used directly by ML models due to the existence of data issues (e.g., data errors, different feature scales).



Designing a data preprocessing pipeline is challenging

Data Preprocessing Pipeline is a sequence of operators, where each operator tackles one specific data issue.



Complex design decisions

- **Types**: should we include outlier removal?
- **Operators**: should we use standardization or min-max normalization?
- **Order**: should we remove outlier before or after normalization?
- **Feature-wise**: different features use different pipelines?

↑

Exponentially increases the design space

Limitations in Existing Methods

Practioners

- Use a default pipeline or trial-and-error methods.

Systems	Operator	Types	Order	Feature-wise	Optimization Method
H2O	✗	✗	✗	✗	Random Search
Azure	✓	✗	✗	✗	Bayesian Optimization
Auto-Sklearn	✓	✓	✗	✗	Bayesian Optimization
Learn2Clean	✓	✓	✓	✗	Q-Learning
DiffPrep-Fix	✓	✓	✗	✓	**Bi-level Optimization with Gradient Descent**
DiffPrep-Flex	✓	✓	✓	✓	

Traditional Data Cleaning Work

- Design pipelines that optimize data quality independently of ML.
- Data quality may not be accessible, and it may not lead to the optimal ML performance.

Existing AutoML Systems

- Limited search space.
- Train model multiple times.

DiffPrep: Automate Data Preprocessing

Goal: automatically and efficiently select a data preprocessing pipeline from the search space such that the model performance (validation accuracy) is maximized.

Outer level: $\underset{pipline}{\text{argmin}}\, Loss\,(D_{val}, pipeline, model^*)$

Inner level: $s.t.\ model^* = \underset{model}{\text{argmin}}\, Loss\,(D_{train}, pipeline, model)$

"Bi-level Optimization"

Compared with existing AutoML systems:

- Explore the entire design space of data preprocessing pipelines (types, operators, order, feature-wise).
- Only need to train ML model once.

Solving the bi-level optimization problem

Outer level: $\underset{pipline}{\operatorname{argmin}} Loss\ (D_{val}, pipeline, model^*)$

Inner level: $s.t.\ model^* = \underset{model}{\operatorname{argmin}} Loss\ (D_{train}, pipeline, model)$

Naive Approach: Train a ML model with every possible pipeline and select the best one. This does not work as the space is very large -- O(m^{sc}) with m choices, s transformations, c feature.

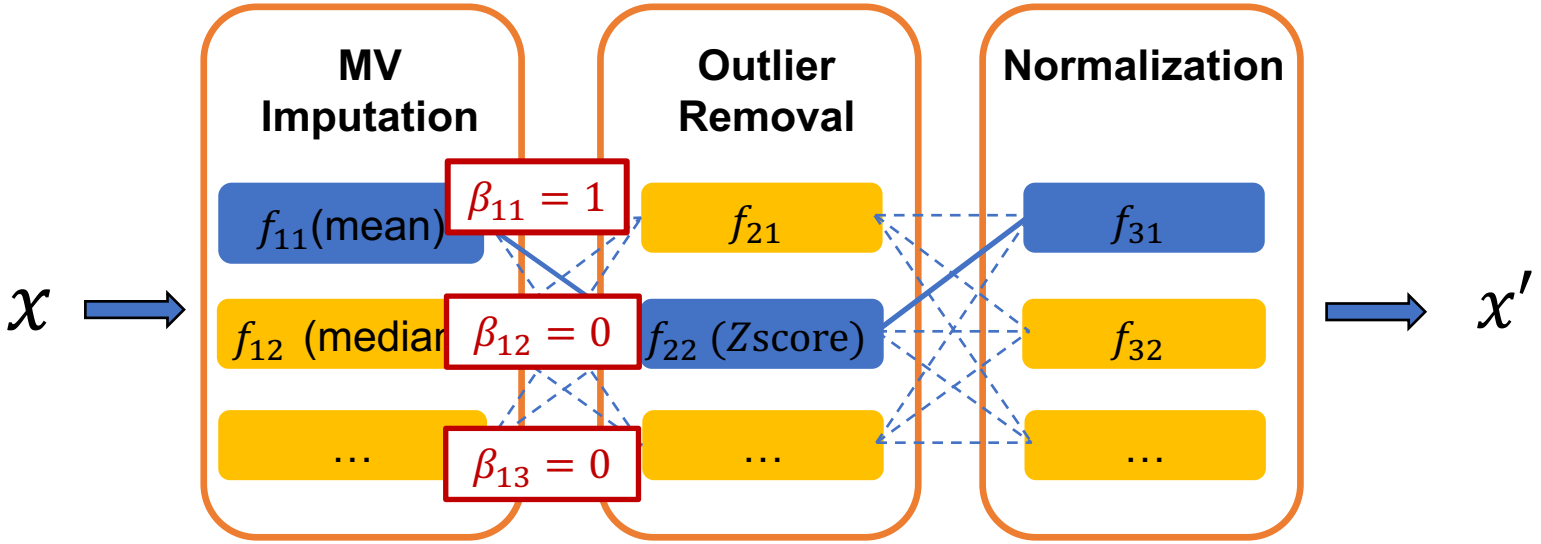
Key challenge: The search space of pipelines is discrete.

Can we convert the discrete search space into a continuous and differentiable space? -- Then we can use gradient descent

Step 1: Parameterization

Goal: Represent each choice of pipeline using a set of binary parameters

Let‘s first assume we have a predefined order of transformation.



$$x_i = \sum_i \beta_{ij} f_{ij}(x_{i-1})$$

$$x'(x, \boldsymbol{\beta})$$

$$L\,(D_{val}, \boldsymbol{\beta})$$

$$L\,(D_{train}, \boldsymbol{\beta})$$

Associate each operator with a $\beta_{ij} \in \{0,1\}$

$$\beta_{ij} = \begin{cases} 1 & f_{ij} \text{ is selected} \\ 0 & \text{Otherwise} \end{cases}$$

$$\sum_i \beta_{ij} = 1$$

Step 2: Relaxation

Associate each operator with a $\beta_{ij} \in \{0,1\}$ —Relax→ $\boldsymbol{\beta}$: $\beta_{ij} \in [0,1]$

$$\beta_{ij} = \begin{cases} 1 & f_{ij} \text{ is selected} \\ 0 & \text{Otherwise} \end{cases}$$

To retain constraints $\sum_i \beta_{ij} = 1$, use **softmax function**, $\tau_{ij} \in \mathbb{R}$

$$\beta_{ij} = \frac{\exp(\tau_{ij})}{\sum_k \exp(\tau_{ik})}$$

* Order selection can be supported using a similar methodology of parameterization (permutation matrix) + relaxation (Sinkhorn normalization)

Gradient-based Bi-level Optimization

Iteratively and alternatively

- update pipeline parameters using gradient of validation loss
- update model parameters using gradient of training loss

Algorithm 1 Learning Model and Pipeline Parameters via Gradient Descent

Require: S, D_{train}, D_{val}
Ensure: $\boldsymbol{\beta}$, $\mathbf{w}$

1: **while** *not converged* **do**
2: Update underlying parameters for $\boldsymbol{\beta}$: $\boldsymbol{\tau} = \boldsymbol{\tau} - \eta_1 \nabla_{\boldsymbol{\tau}} L_{val}(\boldsymbol{\beta}, \boldsymbol{w} - \eta_2 \nabla_{\boldsymbol{w}} L_{train}(\boldsymbol{\beta}, \boldsymbol{w}))$
3: Update model parameters: $\boldsymbol{w} = \boldsymbol{w} - \eta_2 \nabla_{\boldsymbol{w}} L_{train}(\boldsymbol{\beta}(\boldsymbol{\tau}), \boldsymbol{w})$
4: **return** $\boldsymbol{\alpha}$, $\mathbf{w}$

We can learn pipeline and model parameters simultaneously by training model only once!

Experiment Setup

Datasets: 18 real-world datasets

Model: Logistic regression

Search Space:

Missing value imputation, Outlier removal, Discretization, Normalization

Transformation Types	Transformation Operators	
Missing Value Imputation	Numerical Features	Mean Median Mode
	Categorical Features	Most Frequent Value Dummy Variable
Normalization	Standardization Min-Max Scaling Robust Scaling Max Absolute Scaling	
Outlier Removal	Z-Score (k) MAD (k) IQR (k)	
Discretization	Uniform (n) Quantile (n)	

Methods compared:

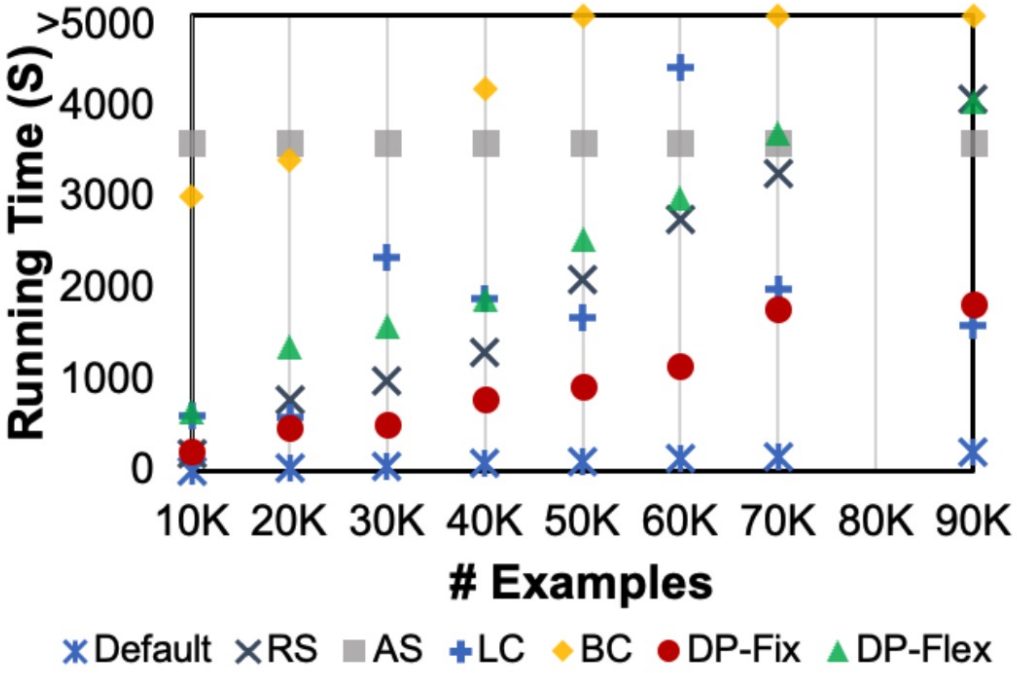
Our methods	Practical Methods	AutoML Systems	Advanced Data Cleaning Methods
DiffPrep-Fix (DP-Fix) DiffPrep-Flex (DP-Flex)	Default (DEF) Random Search (RS)	AutoSklearn (AS)	BoostClean (Clean) Learn2Clean (LC)

Experiment Results – Model Accuracy

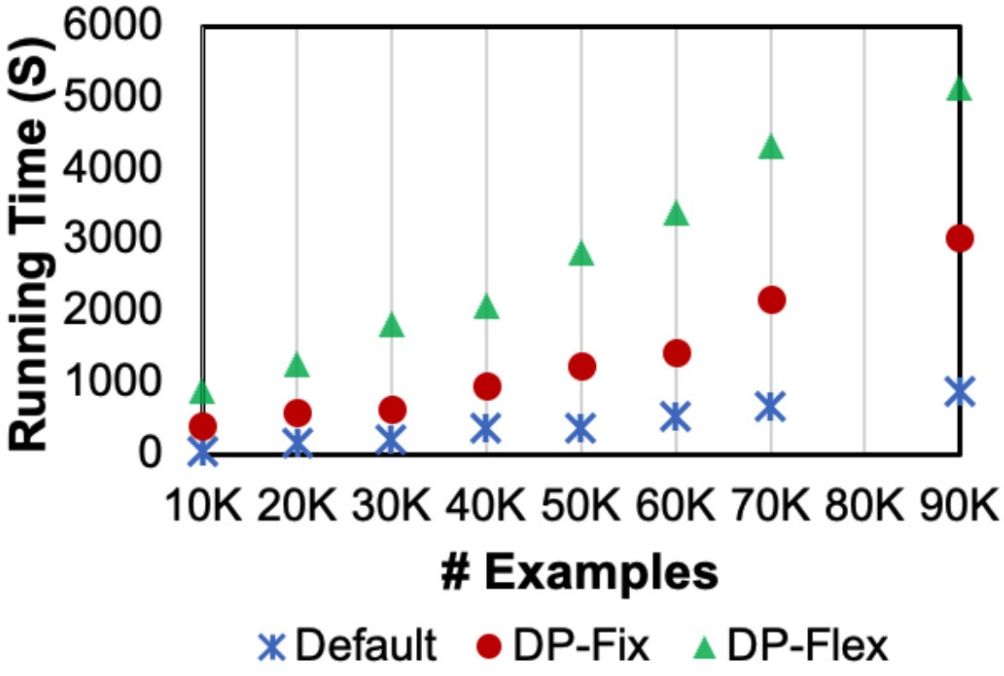
Dataset	Data Characteristics					Test Accuracy						
	#Ex.	#Feat.	#classes	#MVs	#Out.	DEF	RS	AS	LC	BC	DP-Fix	DP-Flex
abalone	4177	9	28	0	200	0.24	0.243	0.216	0.186	0.168	0.238	**0.255**
ada_prior	4562	15	2	88	423	0.848	0.844	0.853	0.816	0.848	**0.854**	0.846
avila	20867	11	12	0	4458	0.553	0.598	0.615	0.597	0.585	**0.638**	0.63
connect-4	67557	43	3	0	45873	0.659	0.671	0.667	0.658	0.69	**0.732**	0.701
eeg	14980	15	2	0	209	0.589	0.658	0.657	0.641	0.659	**0.678**	0.677
google	9367	9	2	1639	109	0.586	0.627	**0.664**	0.549	0.616	0.645	0.641
house	1460	81	2	6965	617	0.928	0.938	**0.945**	0.812	0.928	0.932	**0.945**
jungle_chess	44819	7	3	0	0	0.668	0.669	0.678	0.676	0.667	**0.682**	**0.682**
micro	20000	21	5	0	8122	0.564	0.579	0.584	0.582	0.561	0.586	**0.588**
mozilla4	15545	6	2	0	290	0.855	0.922	**0.931**	0.854	0.93	0.923	0.922
obesity	2111	17	7	0	25	0.775	0.841	0.737	0.723	0.652	0.893	**0.896**
page-blocks	5473	11	5	0	1011	0.942	0.959	**0.969**	0.92	0.951	0.957	0.967
pbcseq	1945	19	2	1445	99	0.71	0.73	0.712	0.704	0.72	0.725	**0.743**
pol	15000	49	2	0	8754	0.884	0.879	0.877	0.737	0.903	0.904	**0.919**
run_or_walk	88588	7	2	0	8548	0.719	0.829	0.851	0.728	0.835	0.907	**0.917**
shuttle	58000	10	7	0	5341	0.964	0.996	0.998	0.997	0.997	**0.998**	0.997
uscensus	32561	15	2	4262	2812	0.848	0.84	0.851	0.786	0.848	**0.857**	0.852
wall-robot-nav	5456	25	4	0	1871	0.697	0.872	0.869	0.69	0.9	0.898	**0.914**

DiffPrep achieves the best test accuracy on 15 out of 18 datasets!

Experiment Results – Runtime



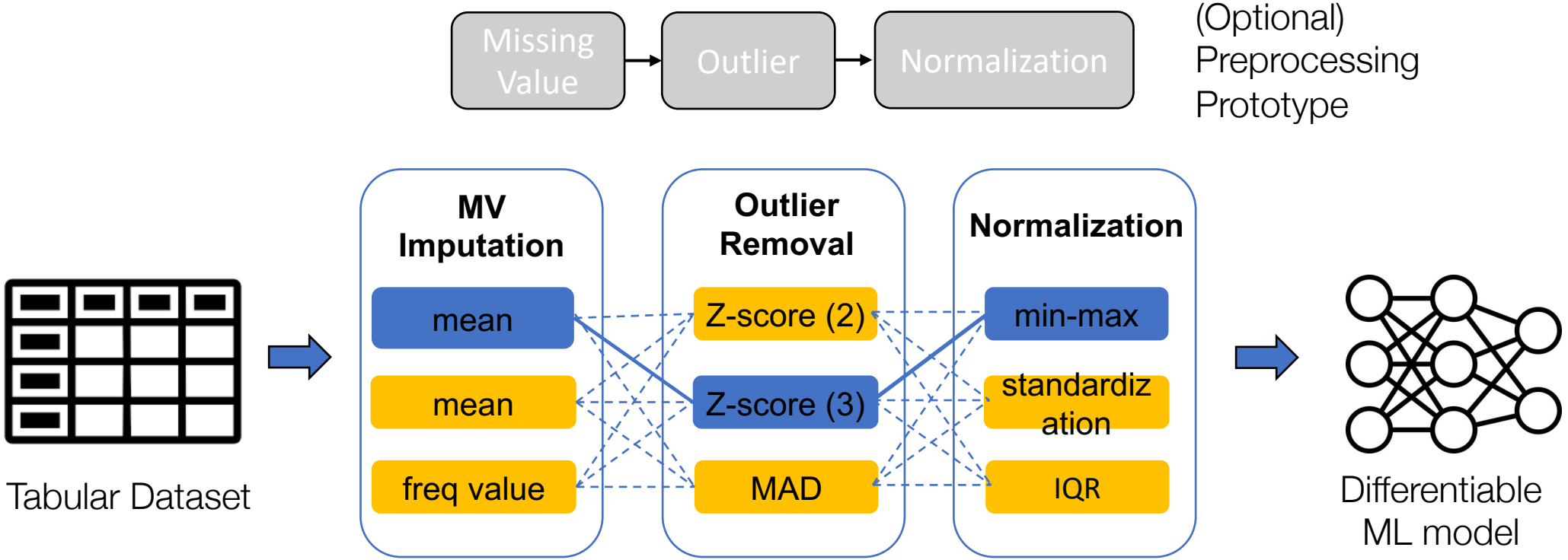
(a) Logistic Regression



(b) Two-layer Neural Network

DiffPrep is faster than other approaches like AutoSklearn and RandomSearch

DiffPrep: Automate Data Preprocessing

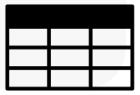


Novel bi-level optimization problem that enables:

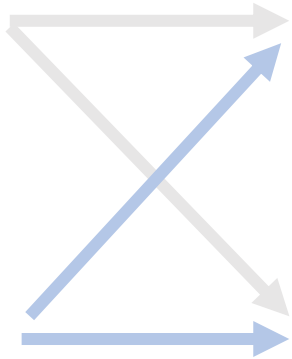
- Large pipeline search space
- Train model only once

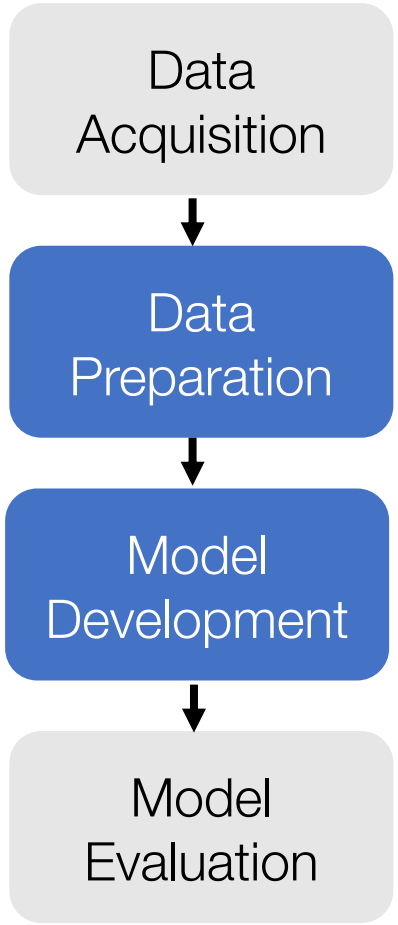
This talk: Data curation for ML

Automatic Data Preprocessing
ML-aware preprocessing pipeline search
(SIGMOD'23)



Framework + Hardware Profiling
fine-grained profiling for preprocessing pipelines
(WIP)

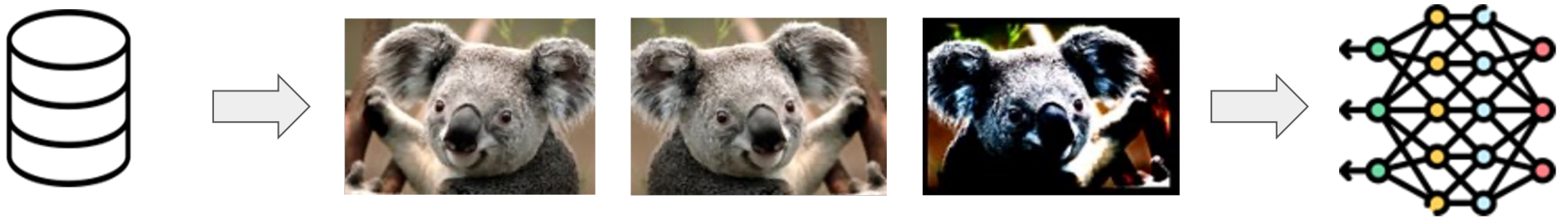




LOTUS: Characterization of Machine Learning Preprocessing Pipelines via Framework and Hardware Profiling

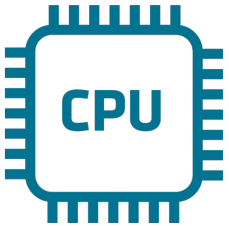
With Rajveer Bachkaniwala, Harshith Lanka, and Ada Gavrilovska

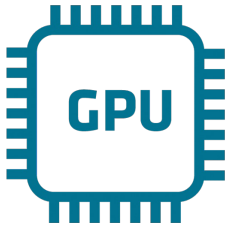
Preprocessing in ML training jobs



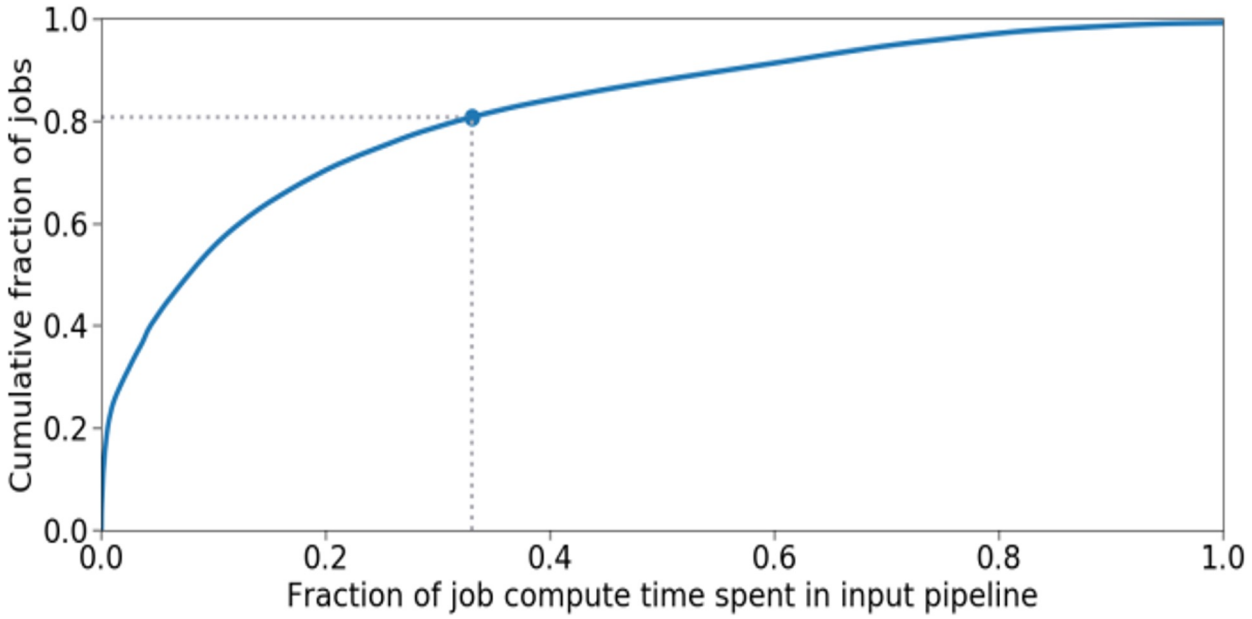
Storage **Preprocessing** **Training**

```
transforms.Compose([
        transforms.RandomResizedCrop(…),
        transforms.RandomHorizontalFlip(…),
        transforms.Normalize(...)])
```





Preprocessing performance matters



Google's profile numbers:

- Low latency batch generation required (< 1 ms)
- 20% jobs spend > 33% compute time in ingestion

tf.data, VLDB 2021| cedar, arxiv 2024

Especially in

- ML training jobs that demands low latency
- Systems with a CPU-to-accelerator ratio imbalance

Many preprocessing optimization

Parallelization
(Plumber [MLSys'22], tf.data [VLDB'21])

Accelerator Offloading
(NVIDIA DALI)

Caching Optimizations
(tf.data service, Cachew [ATC'22], FFCV [CVPR'23], Where Is My Training Bottleneck? [SIGMOD'22])

Disaggregated Preprocessing
(GoldMiner [SIGMOD'23], cedar [arXiv'24])

Co-location and scheduling
(Revamper [ATC'21], SiloD [EuroSys'23])

Optimization rely on understanding of the performance bottlenecks!

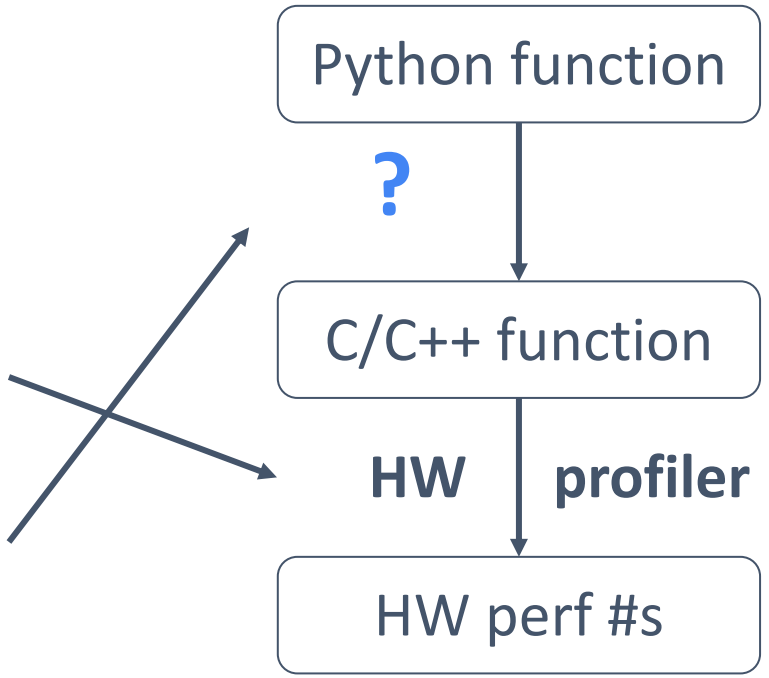
Limitation of Current Profiling Tools

Connecting python functions to their performance on CPU hardware

Preprocessing operations are declared in Python!

For hardware profilers (e.g., Intel Vtune, AMD uProf):

- Hardware profilers collect performance numbers for C/C++ functions
- Exact stack trace from Python function to C/C++ functions called is missing



Limitation of Current Profiling Tools

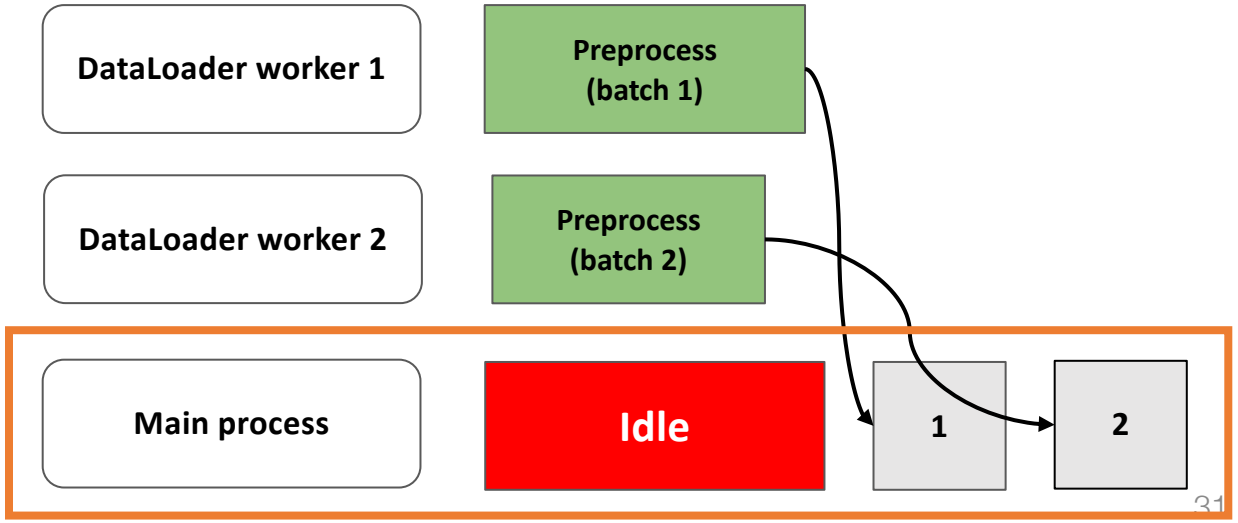
Fine-grained tracing of preprocessing stage with low overhead

Sampling-based Python profilers: Scalene, py-spy, austin

- Limited by sampling interval (e.g., 1-10ms by default)
- ”Fine-grained” events (e.g., duration of individual operator) as short as 100us

PyTorch Profiler

- Does not capture actual preprocessing operations on the worker processes



LOTUS: Profiling Tool for Preprocessing Pipelines*

Enable reasoning of performance of preprocessing pipelines at a hardware level

LotusTrace

Fine-grained tracing of preprocessing stage with low overhead

LotusMap

Connecting python functions to their performance on CPU hardware

* Current implementation targets PyTorch's DataLoader preprocessing library

LotusTrace: Fine-grained tracing with low overhead

Declarative specification =>
Instrumentation while
ensuring generalizability

```
1   import torchvision.transforms as transforms
2   import torchvision.datasets as datasets
3   custom_log_file = <To use our instrumentation>
4   train_dataset = datasets.ImageFolder(
5       traindir,
6       transforms.Compose([
7           transforms.RandomResizedCrop(224),
8           transforms.RandomHorizontalFlip(),
9           transforms.ToTensor(),
10          transforms.Normalize(mean=[0.485, 0.456, 0.406],
11                               std=[0.229, 0.224, 0.225])
12      ], log_transform_elapsed_time=custom_log_file),
13      log_file=custom_log_file
14  )
```

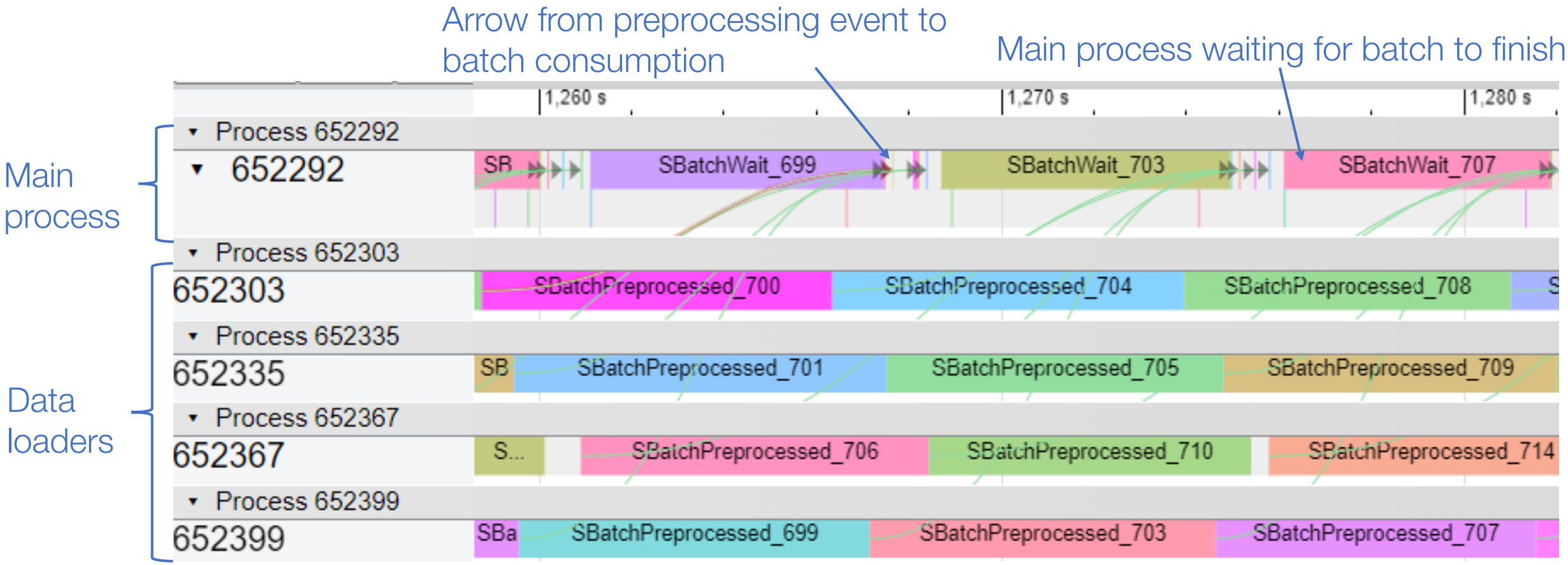
[T1] Total preprocessing time for a specific batch ~ per batch variance

[T2] Time taken by each preprocessing operation in a batch ~ dominant ops

[T3] Time the main process spent waiting for a specific batch to finish being preprocessed by a DataLoader worker ~ GPU idle time

Tracing visualization

Example **image classification pipeline** from MLPerf



Bottleneck: prepeocessing

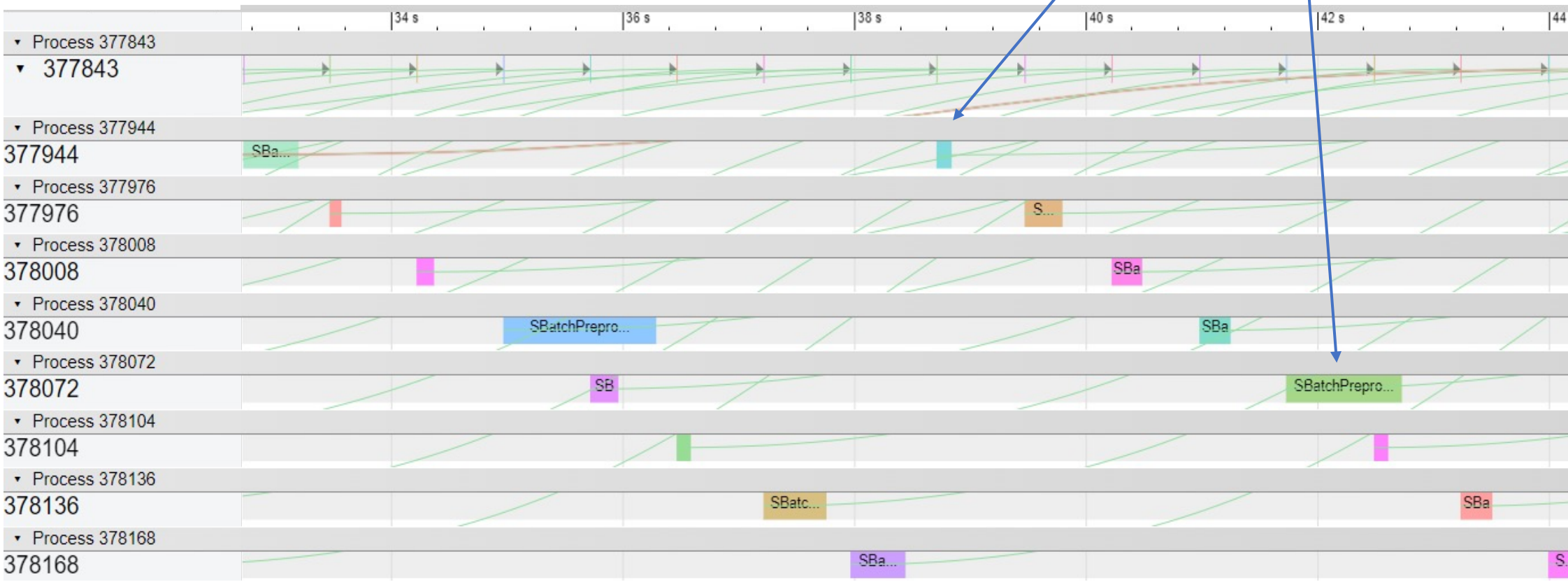
Tracing visualization

Example **image segmentation pipeline** from MLPerf

variation in per batch preprocessing time

Main process

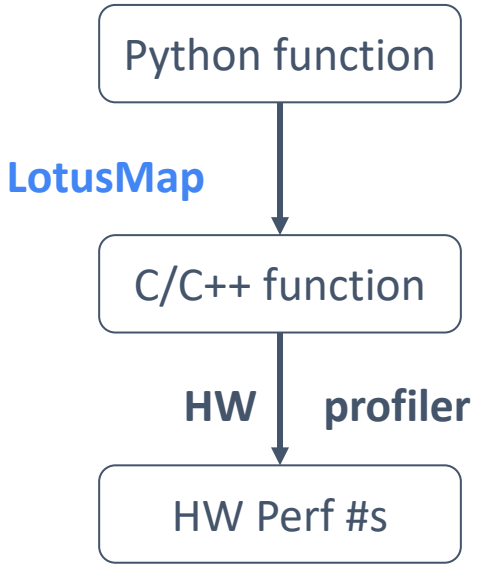
Data loaders



Bottleneck: GPU

LotusMap: Python func <=> hardware events

Missing piece: mapping from Python functions to C++ function



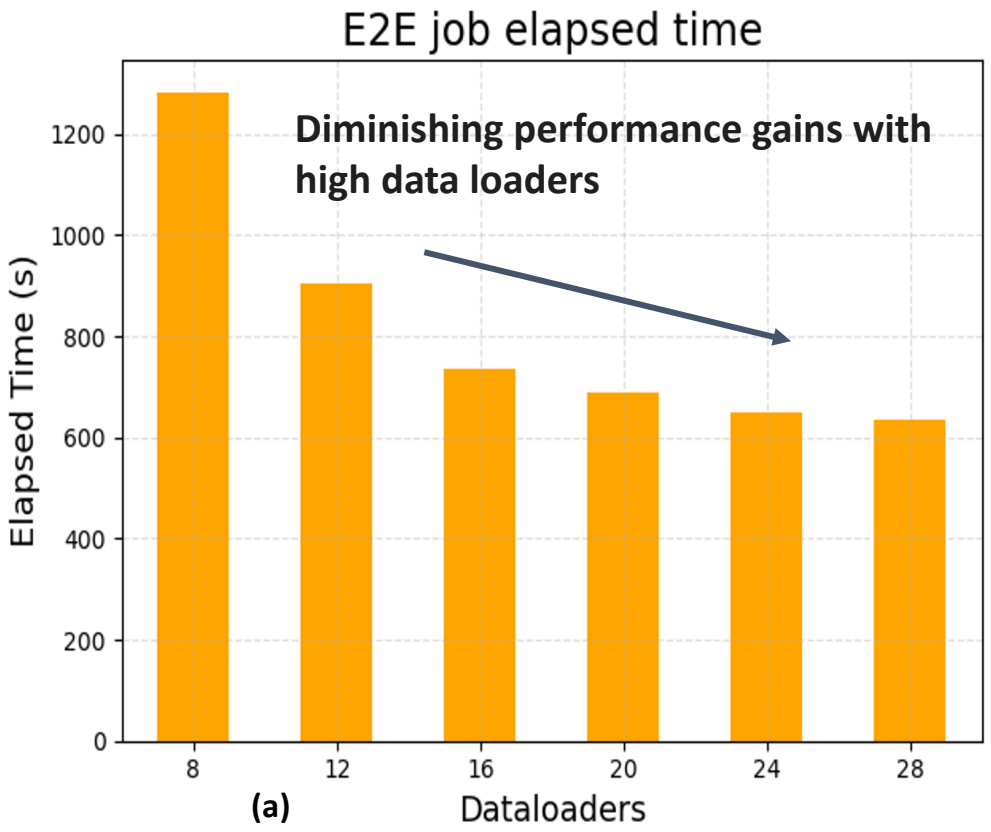
Transformation	Function
Image.convert (Loader)	decompress_onepass jpeg_idct_islow jpeg_idct_16x16 ycc_rgb_convert decode_mcu ImagingUnpackRGB __memset_avx2_unaligned_erms __memmove_avx_unaligned_erms __libc_calloc
RandomResizedCrop	__memmove_avx_unaligned_erms _int_free ImagingResampleHorizontal_8bpc ImagingResampleVertical_8bpc

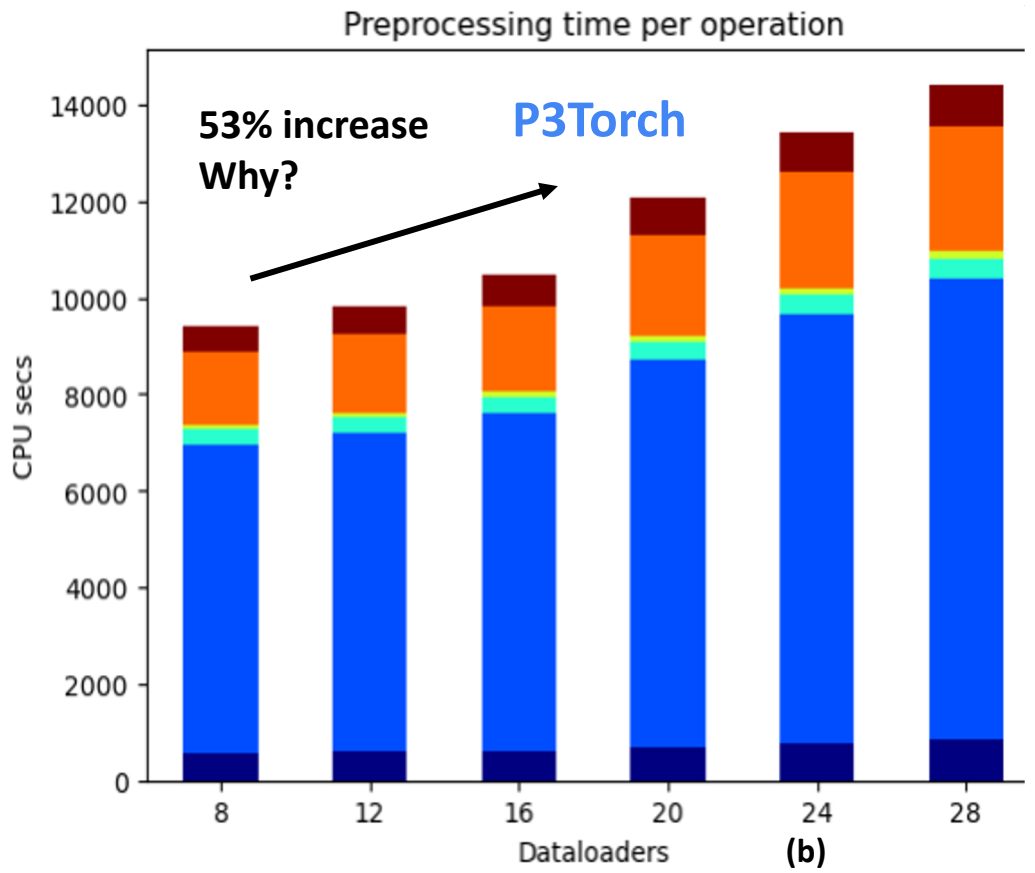
Python funcs

C/C++ funcs

Mapping can be precomputed offline

Case study: Impact of #DataLoaders



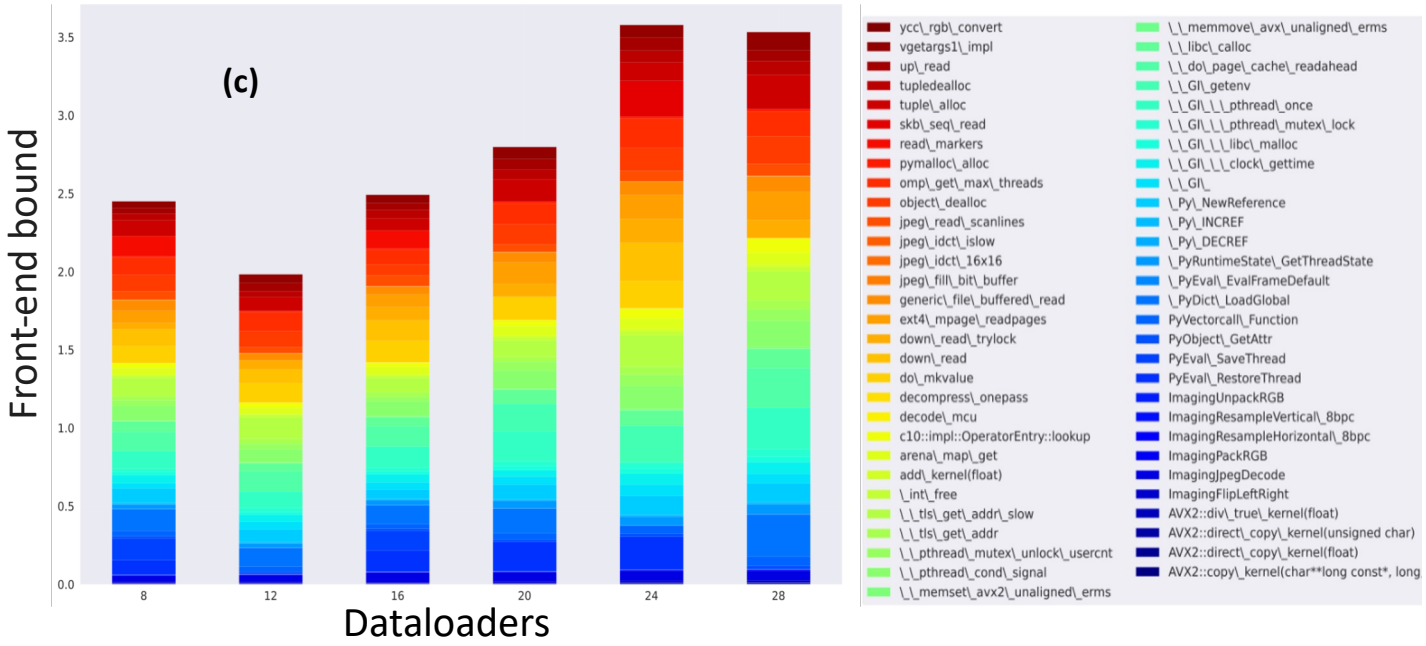


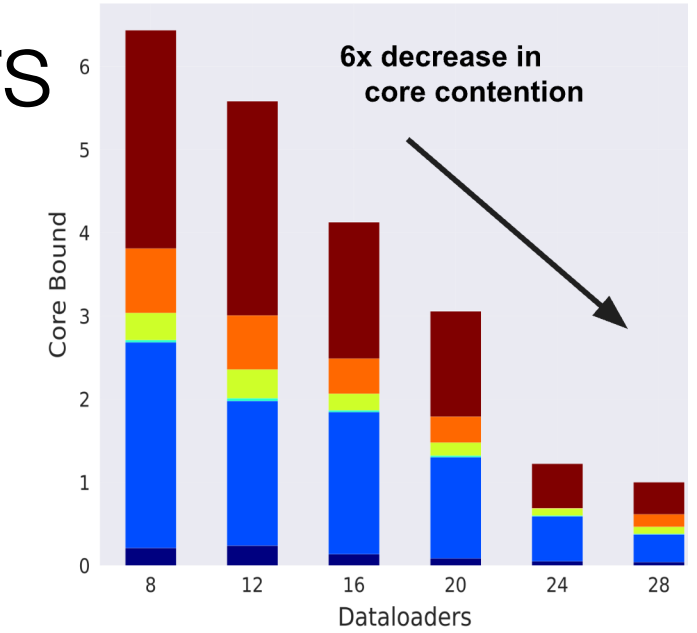
Legend for (b)

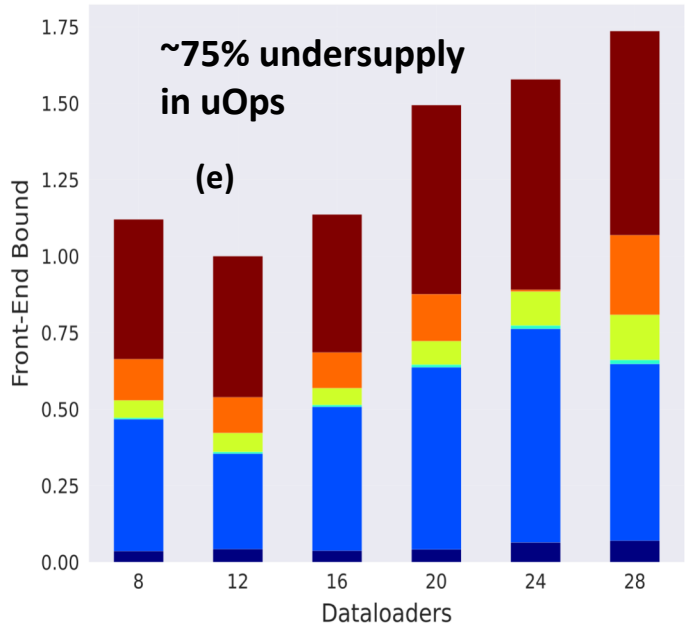
Case study: Impact of #DataLoaders

Preprocessing stage becomes front-end bound with more cores (dataloaders)

Filter out C/C++ functions irrelevant to preprocessing







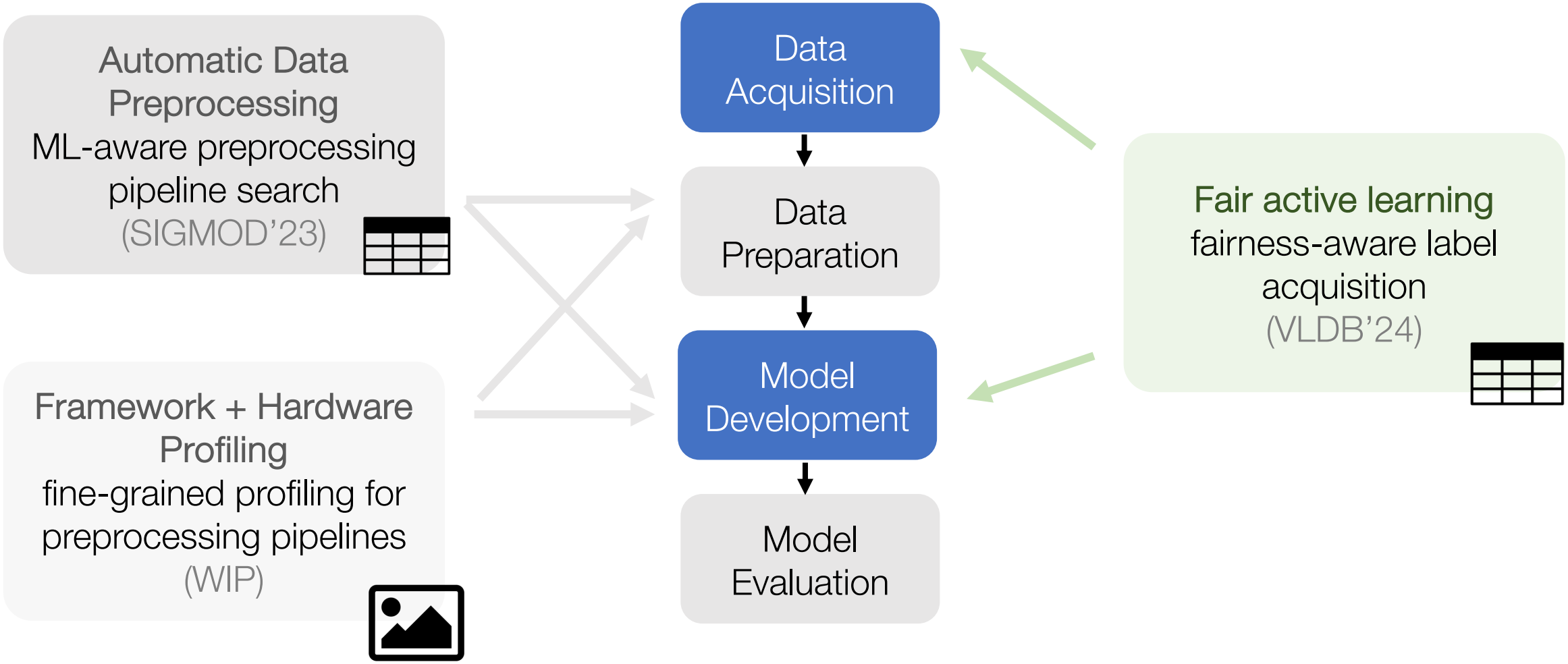
Legend for (e)

Collation | Loader | Normalize | RandomHorizontalFlip | RandomResizedCrop | ToTensor

LOTUS: Profiling Preprocessing Pipelines

- LotusTrace enables insights into the high level behavior of an ML pipeline through finer granularity trace
- LotusMap enables insight into the HW performance of preprocessing operations through a mapping methodology
- More workload characterization in the paper

This talk: Data curation for ML



FALCON: Fair Active Learning using Multi-armed Bandits

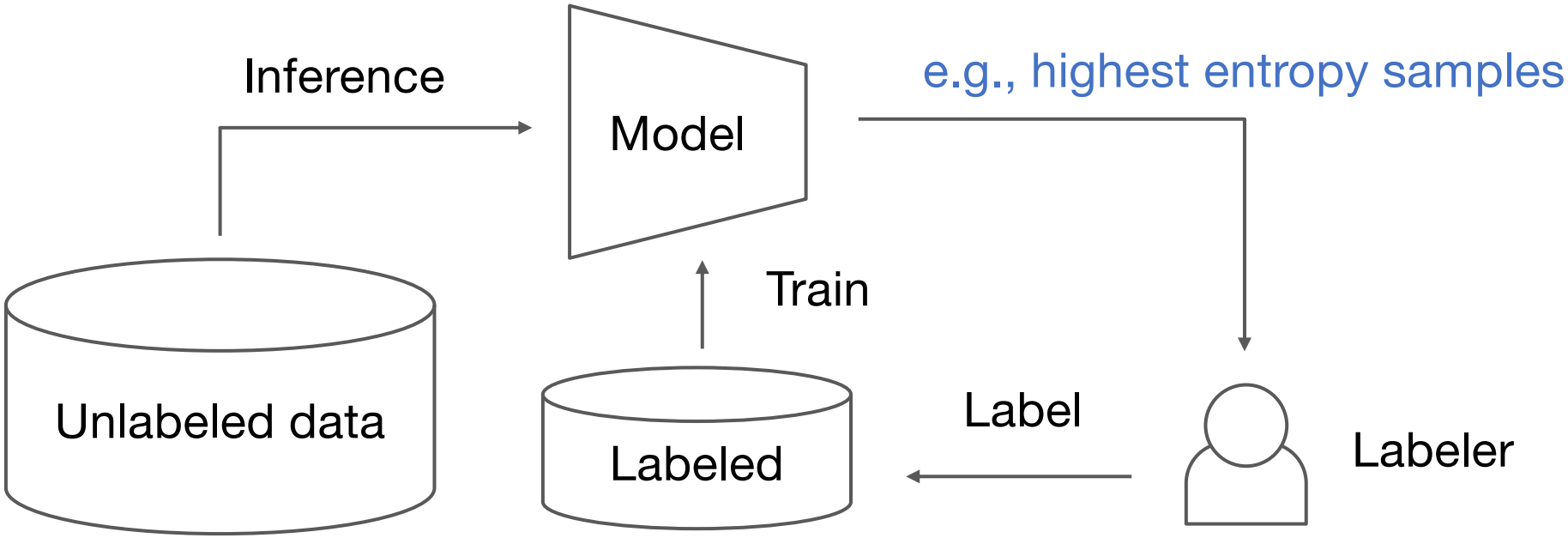
With Ki Hyun Tae, Hantian Zhang, Jaeyoung Park, and Steven Euijong Whang

KAIST

Georgia Tech

Active Learning: Reduce data annotation cost

- Given an unlabeled dataset
- Selects samples to label for maximizing accuracy under a fixed budget



Impact of Active Learning on Fairness

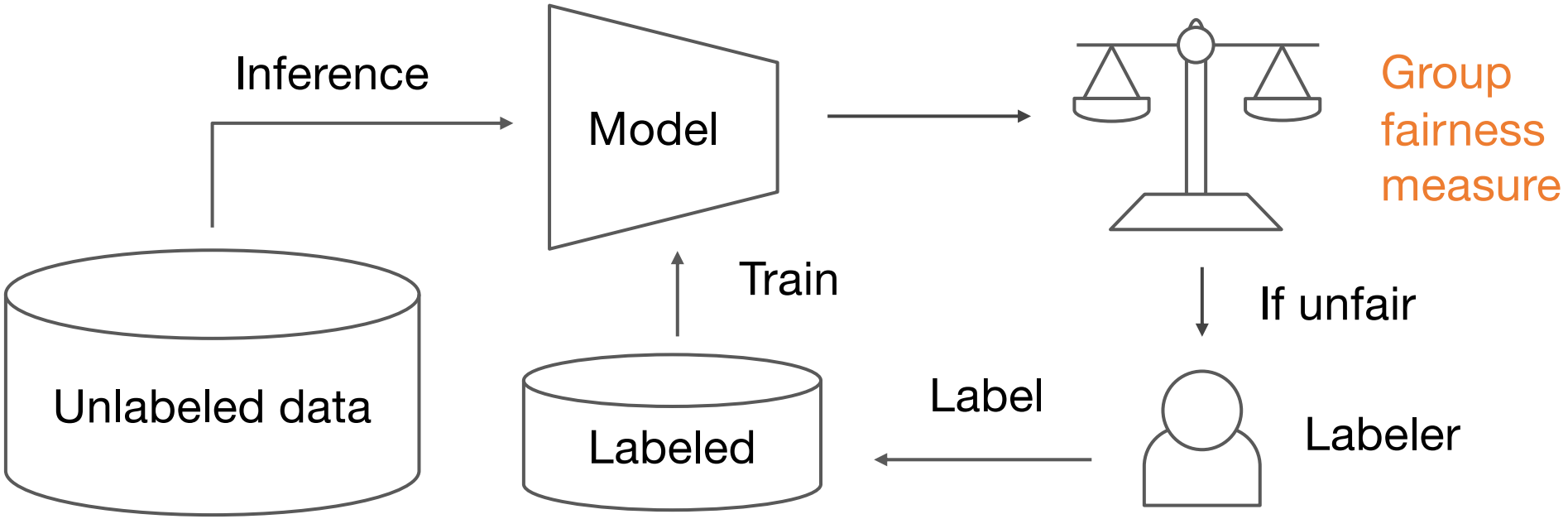
- Labeling more samples could worsen fairness
  - To improve DP, we want more samples from the target subgroup *(attribute=female, label=positive)*
  - What if the sample has a different label (attribute=female, label=negative)?
  - It decreases the positive prediction rate of Female and thus worsens DP

Demographic Parity (DP): Similar positive prediction rate across sensitive groups

$$p(\hat{y} = 1 | z = female) < p(\hat{y} = 1 | z = male)$$

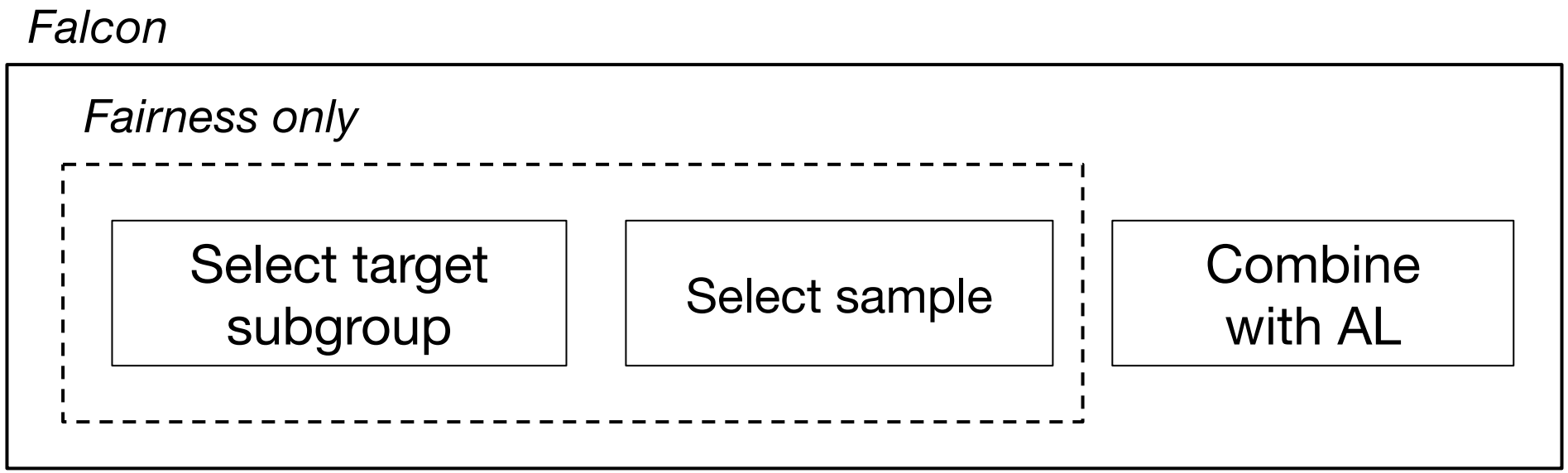
Our Setup: Fair Active Learning

- Selects samples to label for maximizing fairness under a fixed budget
- Supports any group fairness of binary classification models



Falcon: Fair Active Learning

1. Select subgroups to label and uses a trial-and-error method to handle unknown ground-truth labels
2. Identify the most informative samples for fairness using adversarial MABs
3. Balance fairness and accuracy by alternating with traditional AL



Subgroup Labeling for Fairness

- Key strategy: increase the labeling of specific subgroups
  - Subgroup is defined using attributes and labels, e.g., (attribute=female, label=positive)
  - Any group fairness measure can be expressed as a function of subgroup accuracies

Step 1

Subgroup Labeling for Fairness

Key strategy: increase the labeling of specific subgroups

- Subgroup is defined using attributes and labels, e.g., (attribute=female, label=positive)
- Any group fairness measure can be expressed as a function of subgroup accuracies

*See paper for other measures

Demographic Parity (DP): Similar positive prediction rate across sensitive groups

$$p(\hat{y} = 1 | z = female) < p(\hat{y} = 1 | z = male)$$

Target Subgroups

(attribute=female, label=positive)

(attribute=male, label=negative)

Handling Unknown Ground Truth Labels

- However, ground truth labels are not available in an AL setting
- Adding samples with undesired labels can negatively affect fairness

$$p(\hat{y} = 1|z = female) < p(\hat{y} = 1|z = male)$$

(attribute=female, label=~~positive~~)

negative

Trial-and-error Strategy

- Select samples in the target sensitive group to label, but postpone using them in model training when they turn out to have undesirable labels
- Postponing undesired samples is critical for improving fairness

$$p(\hat{y} = 1|z = female) < p(\hat{y} = 1|z = male)$$

(attribute=female, label=positive)

Informativeness for Fairness

- Which sample is the most informative for fairness when using trial-and-error?
- Improves the target group's accuracy the most and also has a desired label

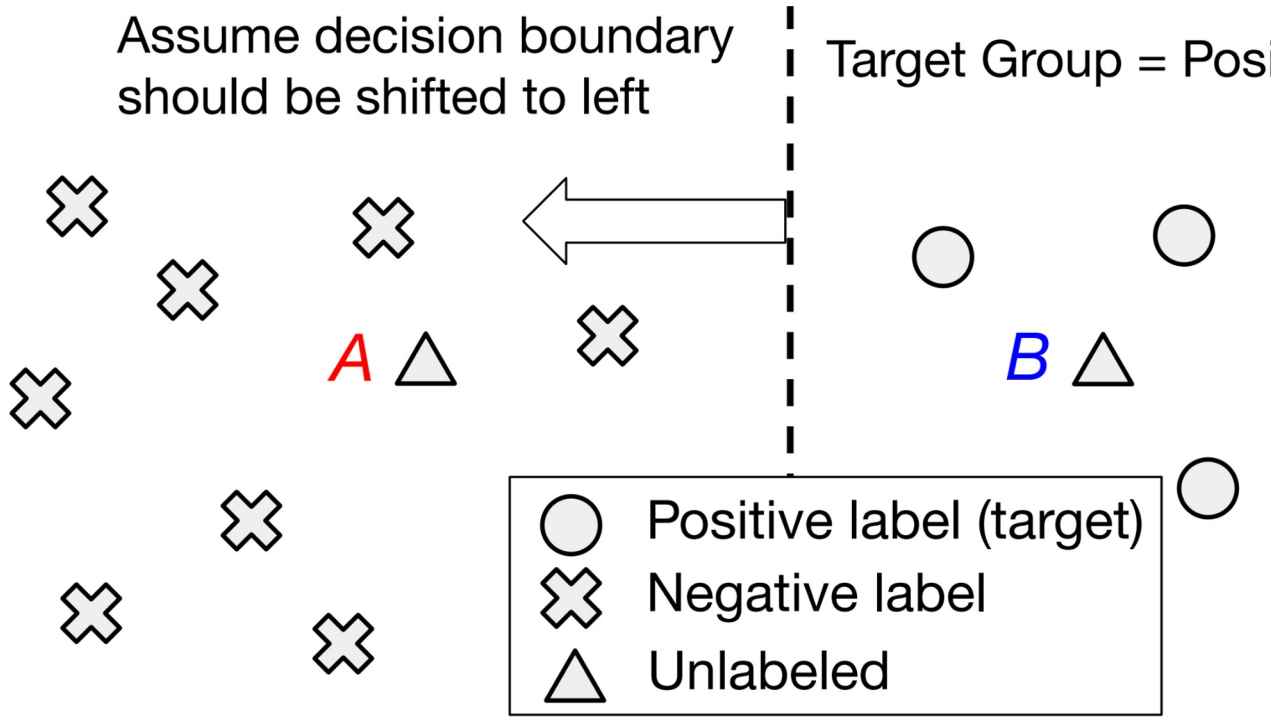
Step 2

Select Target Subgroup | **Select Sample** | Combined with AL

Trade-off b/w Informativeness and Postpone Rate

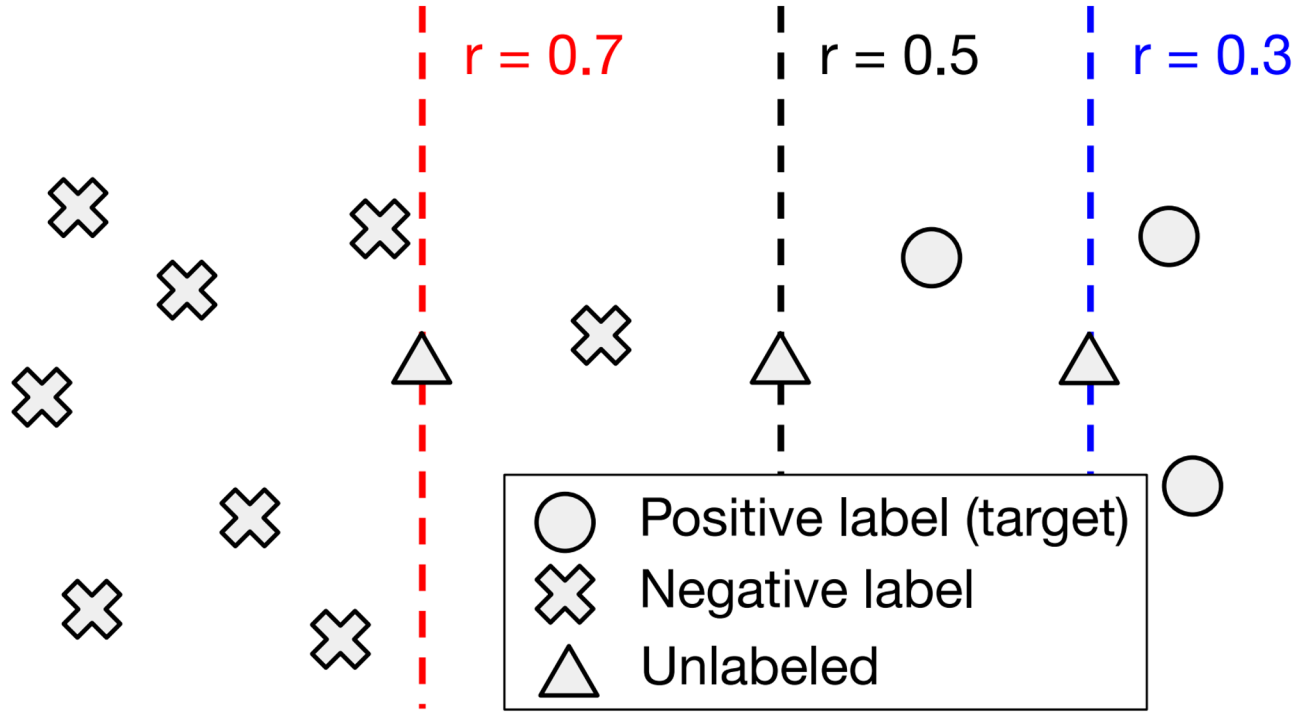
Key observation: the more informative a sample is for improving the target group's accuracy, the less likely it has the target label

- Sample A increases the target group accuracy more than B if positively labeled, but is less likely to have a positive label



Policies: amount of risk taken

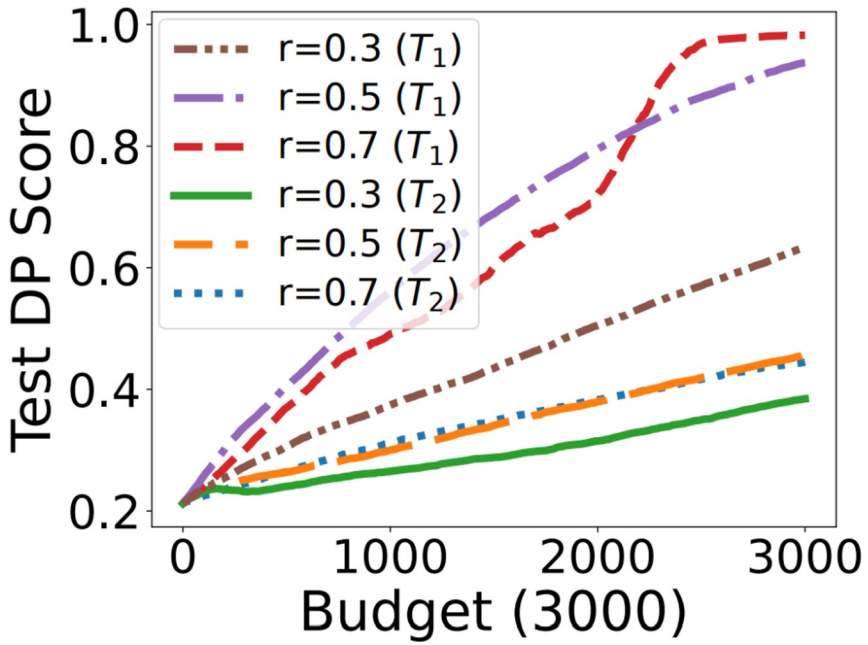
- The more “risk” we are willing to take for finding an informative sample, the less likely it has the desired label
- We capture this risk taking as a policy “r” = c for each target group
  - Selects a sample whose predicted probability for the target label closest to (1 - c)



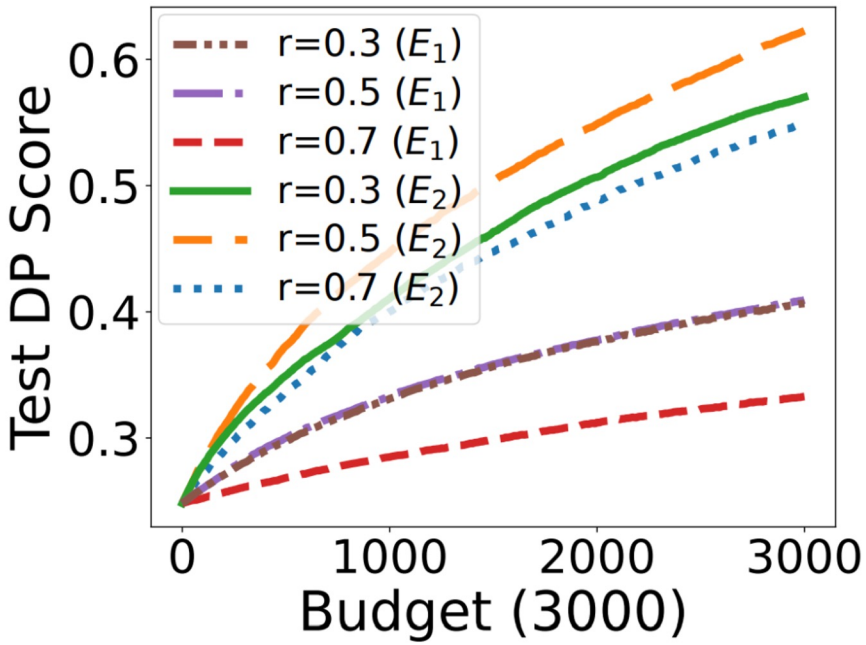
Challenge: Optimal policy changes over time

The optimal policy varies as we label more samples

- There is no clear trend across the datasets



TravelTime (DP)



Employ (DP)

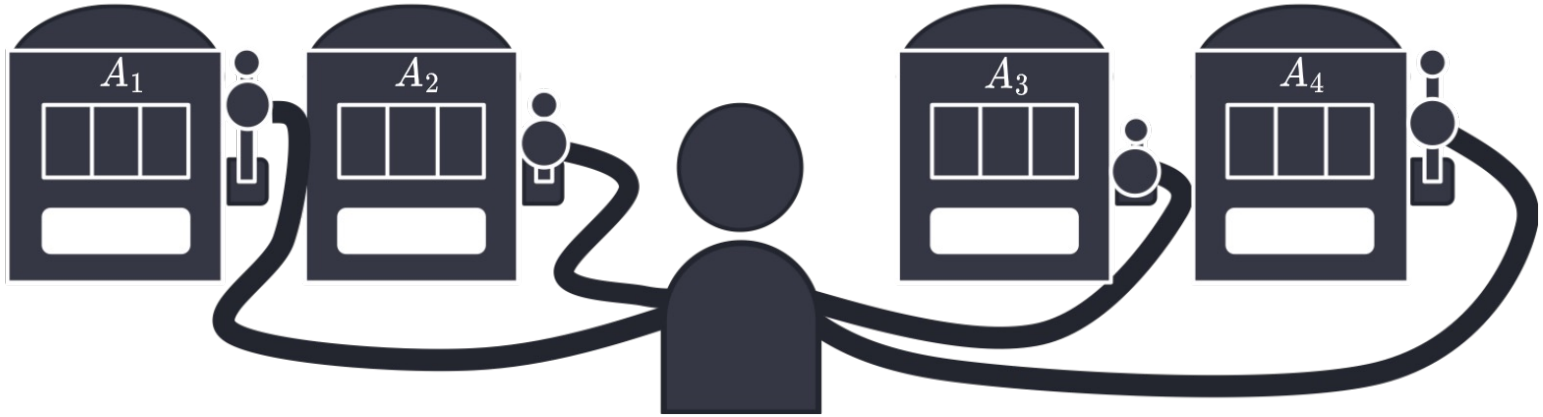
Multi-armed Bandit (MAB) for Policy Search

Adversarial MABs: No assumptions about the reward distribution

- More conservative, but have theoretical guarantees

We use EXP3 as a representative algorithm

- Key idea: some arms may later be useful, keep on giving each arm a chance to be selected
- Selection probability = Accumulated reward + Uniform distribution



Arm: policy
Reward: fairness improv.

Combined with AL for Accuracy

Alternates between fair and accurate labeling probabilistically

- Improves fairness with λ probability and accuracy with (1 – λ) probability
- A higher λ indicates better fairness

No modifications for the AL methods

Step 3

Select Target Subgroup

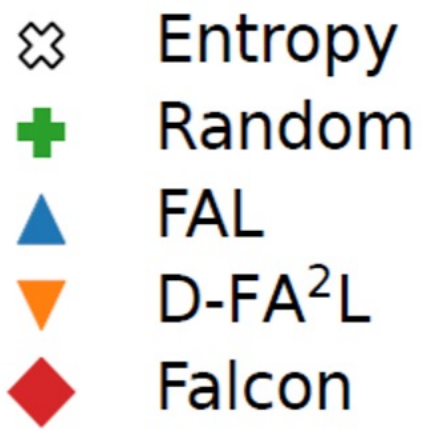
Select Sample

Combined with AL

Experiment Setup

Methods compared:

- Entropy: Standard AL
- Random: Uniform random samples
- FAL: First fair active learning algorithm
- D-FA^{2}L: Disagreement-based AL algorithm

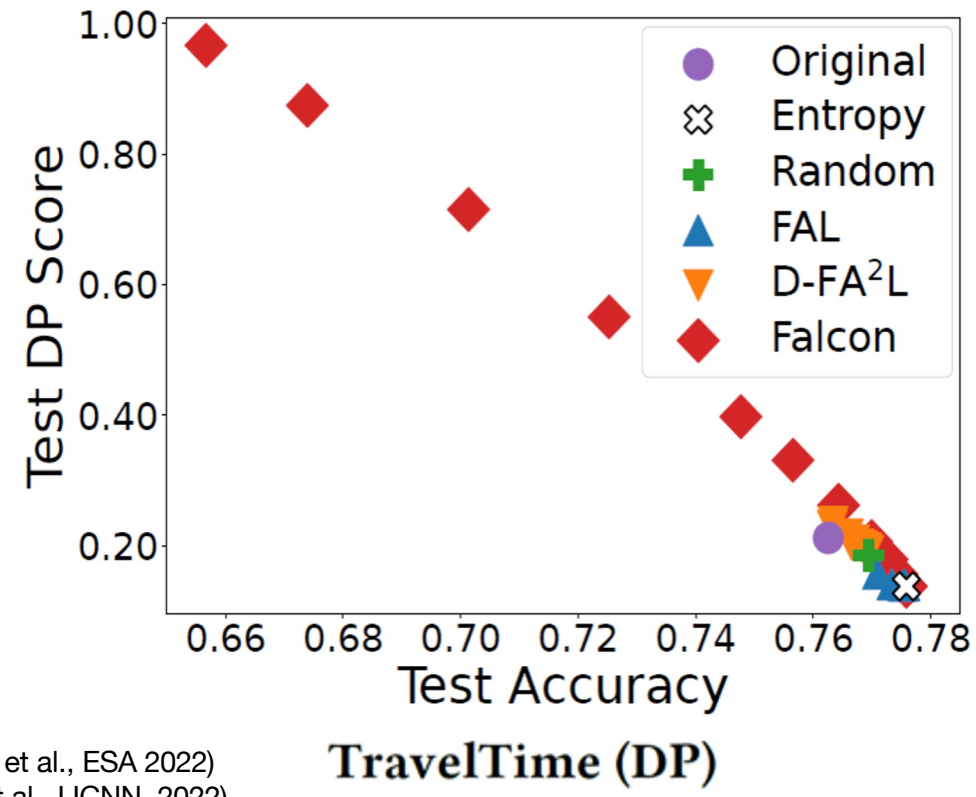


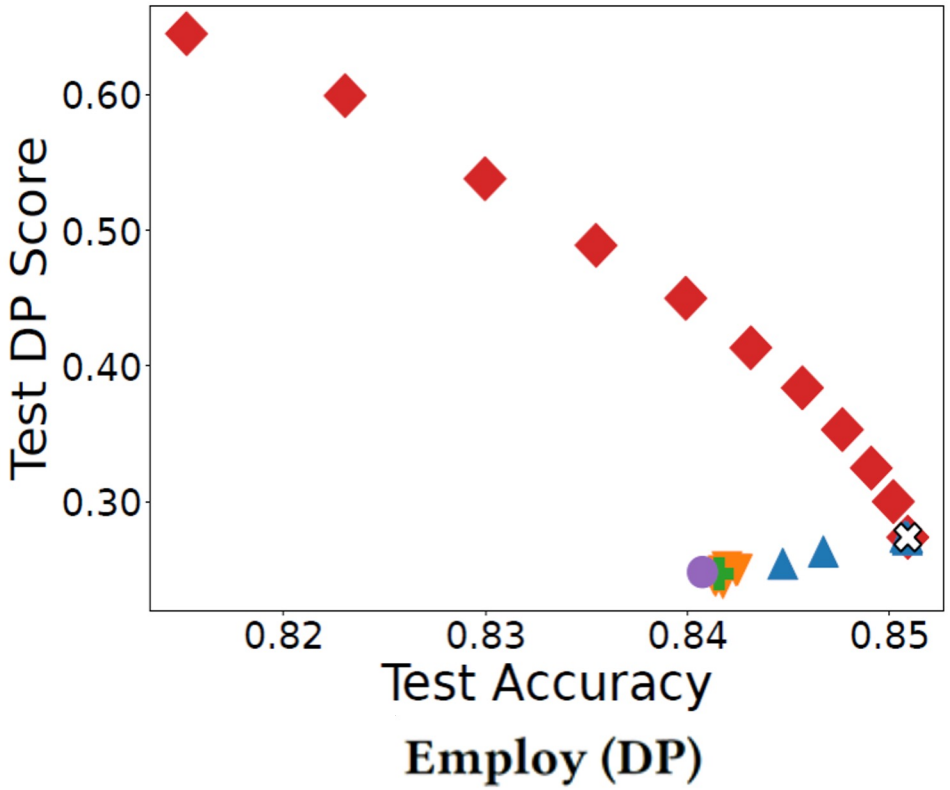
Datasets

Dataset	$\lvert D_{train}\rvert/\lvert D_{un}\rvert/\lvert D_{test}\rvert/\lvert D_{val}\rvert$	Sen. Attr	Batch	B
TravelTime	2,446/48,940/24,470/2,446	gender	10	4K
Employ	5,432/162,960/81,480/5,432	disability	10	4K
Income	3,188/63,760/31,880/3,188	race	10	4K
COMPAS	294/2,356/1,178/294	gender	1	200

Accuracy and Fairness Tradeoff

- Falcon shows the best accuracy and fairness trade-off
  - Also, similar results for other datasets, fairness measures, and ML models

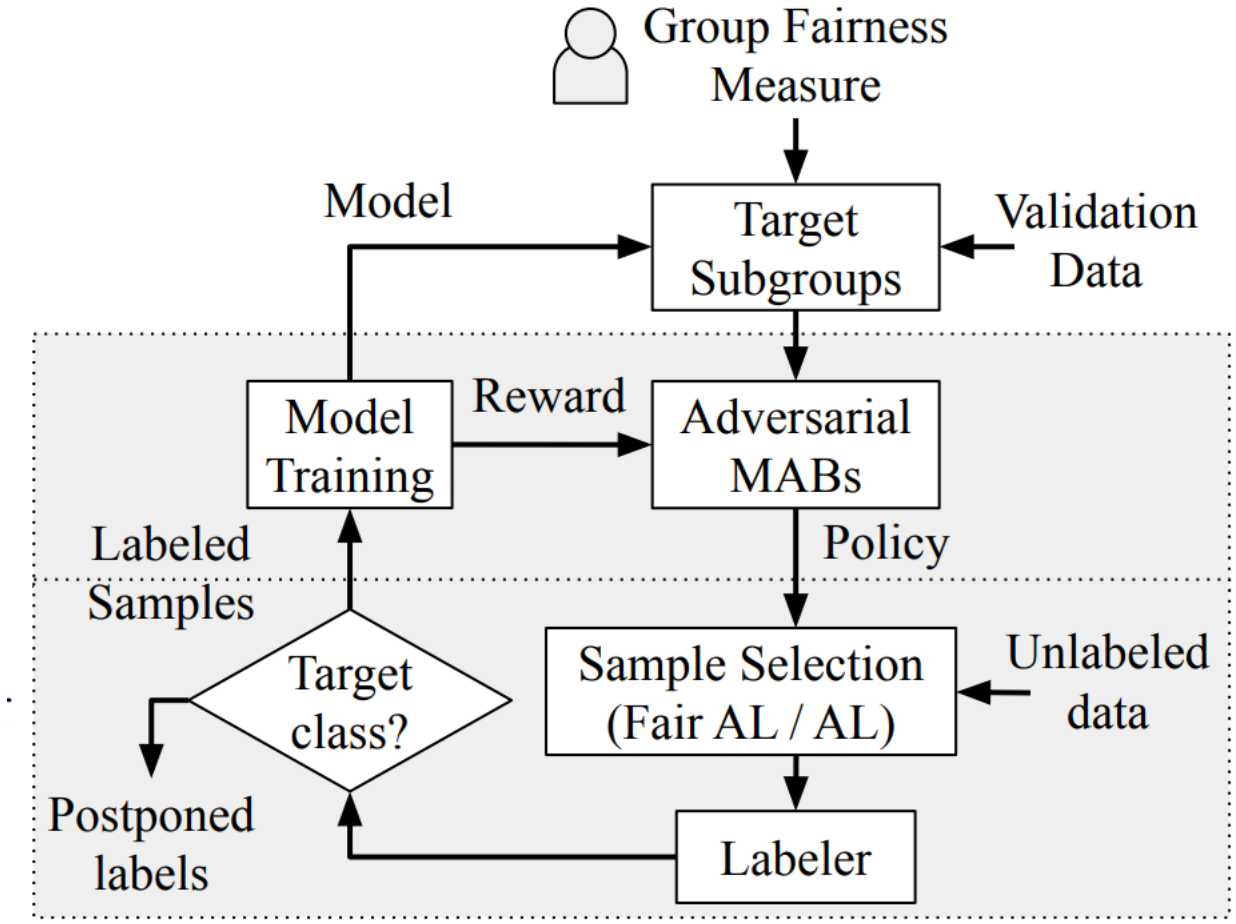




[1] FAL (Anahideh et al., ESA 2022)
[2] D-FA²L (Cao et al., IJCNN 2022)

Falcon Summary

1. Select subgroups to label and uses a trial-and-error method to handle unknown ground-truth labels
2. Automatically selects the best sampling policy using adversarial MABs
3. Balances fairness and accuracy by alternating its selection for fairness with traditional AL



This talk: Data curation for ML