Special Topics: Machine Learning (ML) for Networking

COL867

Holi, 2025

Robustness

Tarun Mangla

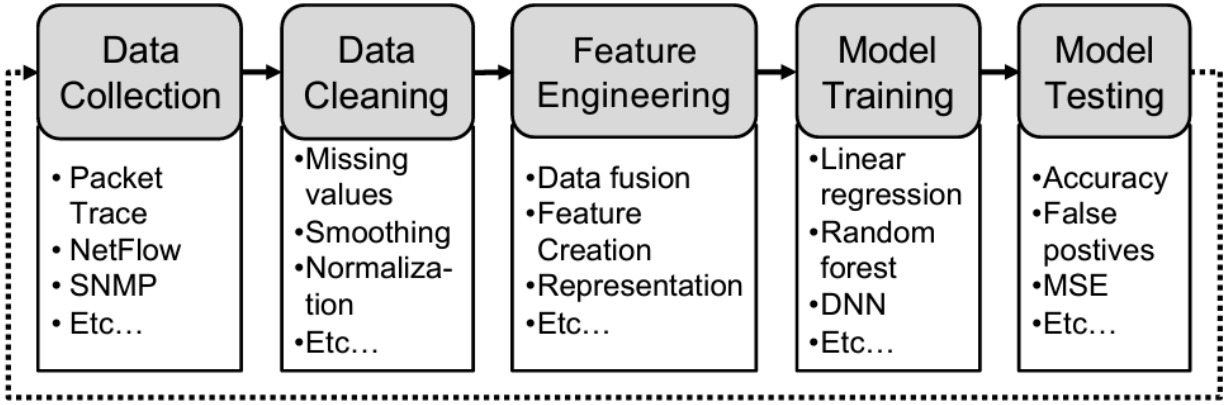
ML for Networks

Module 1: Case studies of specific network learning tasks

Module 2: Task-agnostic automatic ML pipelines for networks

Module 3: Beyond feature engineering and modeling

- Network telemetry
- Robustness
- Explainability
- Synthetic data generation*
- Data imputation
- Formal verification
- ..



Robust ML

- What is robustness in ML?

capacity of a model to sustain stable predictive performance in the face of variations and changes in the input data

Examples of variations or changes:

- Difficulty in adapting to (underrepresented) edge-case scenarios
- Data drift
- Concept drift
- Adversarial inputs
- ..

Robust Network ML

- ML robustness studied extensively
- Not so much in the networking domain
- Most papers evaluate models using in-lab data
- Big gap between prototyping and making models production-ready

Next Few Classes

Case studies around robustness in networking

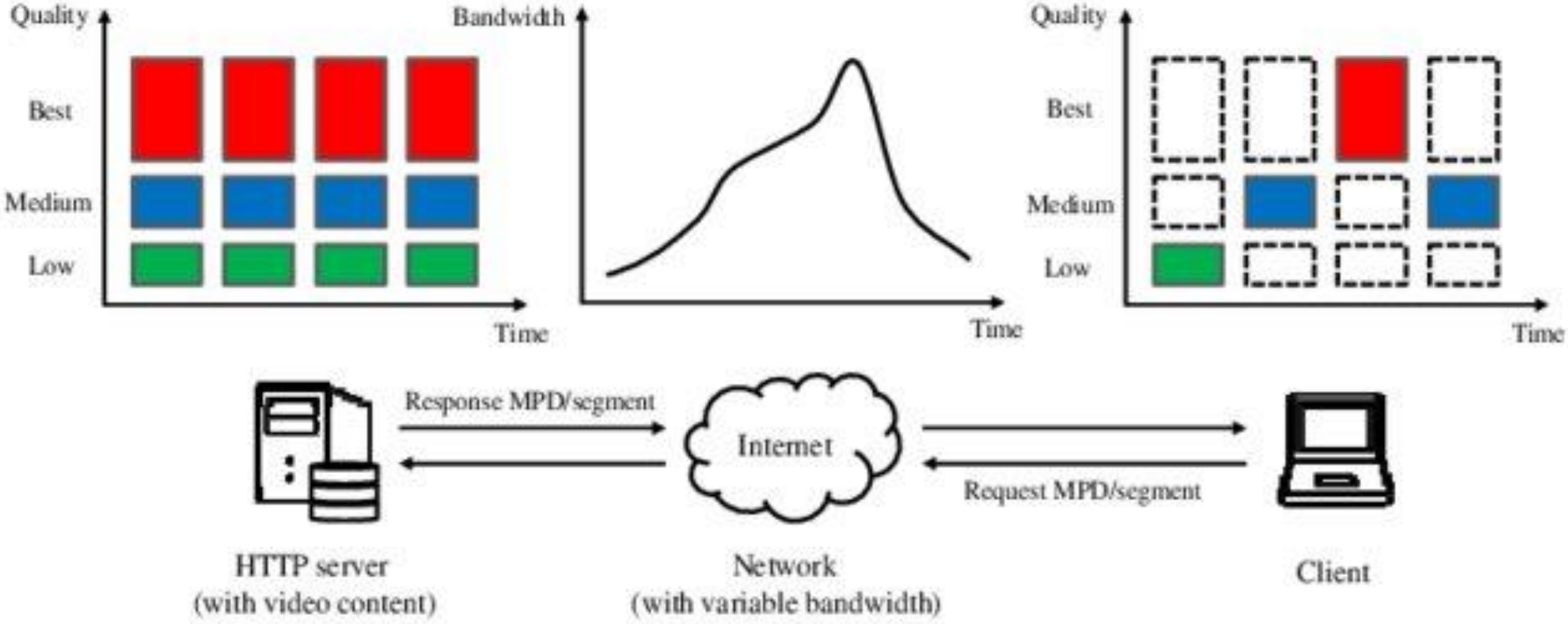
- How to train a robust network ML?
  - Puffer – evaluating robustness & also training for robustness
  - Genet – training robust RL models
- How to handle drift?

Puffer: Key Research Question

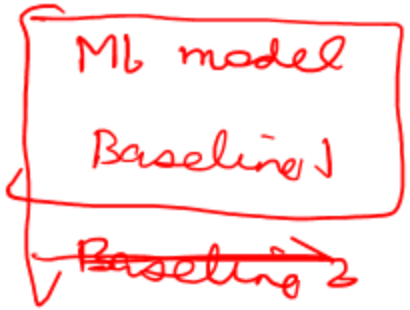
Pensieve

What does it take to create a ML-based bitrate adaptation algorithm that **robustly** performs well over the **wild Internet**?

Bitrate Adaptation



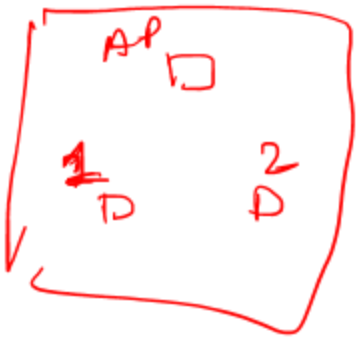
Puffer: Key Research Question

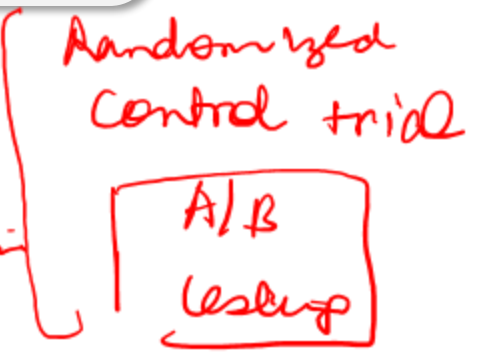


QoE

What does it take to create a ML-based bitrate adaptation algorithm that **robustly** performs well over the **wild Internet**?

How would you answer this question?





Randomized control trial

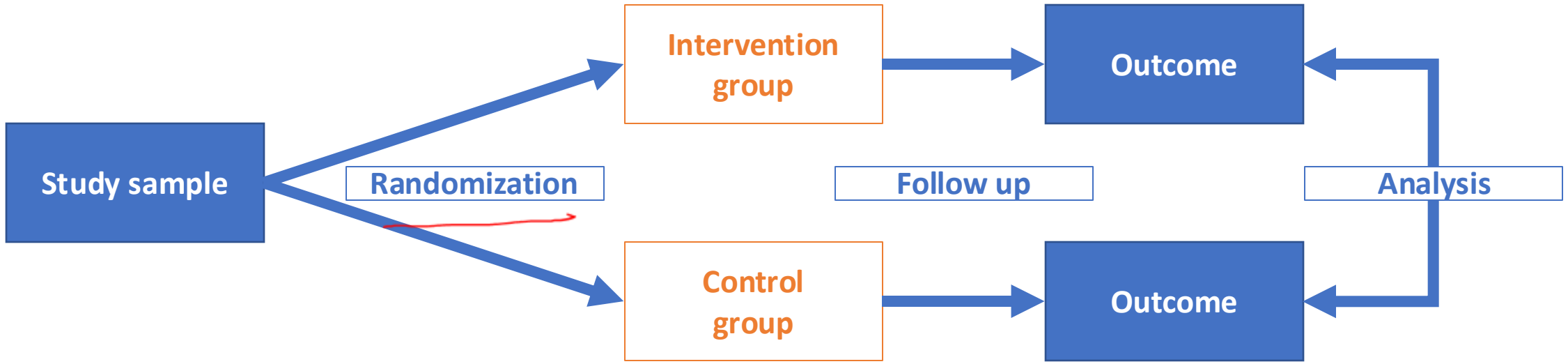
① Controlled network condition evaluation

② Evaluate it over the wild Internet

Causal Inference: $X \rightarrow Y$

Randomized Control Trial (RCT)

- Tests if an intervention works
- Participants randomly assigned to intervention or control group
- Outcome differences reflect treatment effect



How would you design an RCT?

Puffer Demo

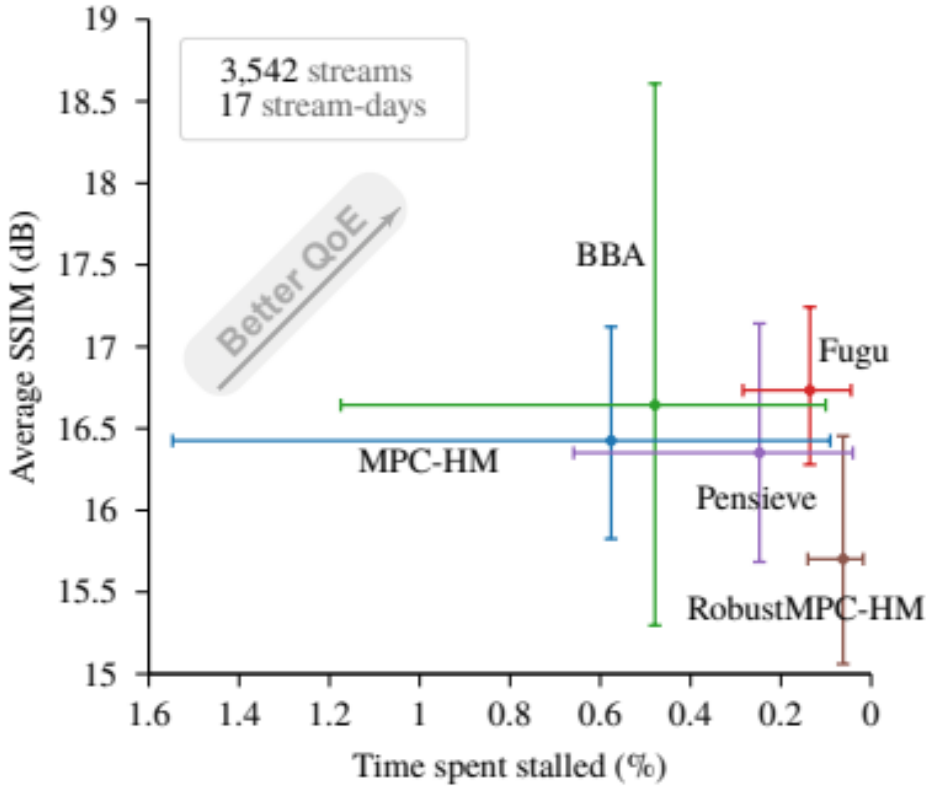
Experiment Design

- Compare it with a few baselines through Randomized Control Trials
- Consider popular bitrate adaptation algorithms: BBA, MPC DASH, Pensieve and their own bitrate adaptation algorithm, Fugu [will cover later]
- Collect QoE data: Structural Similarity Index Measure (SSIM) and Stall durations for each session, bitrate adaptation algorithm
- Puffer study: Collected 38.6 client years of streaming video data to 63,508 distinct users

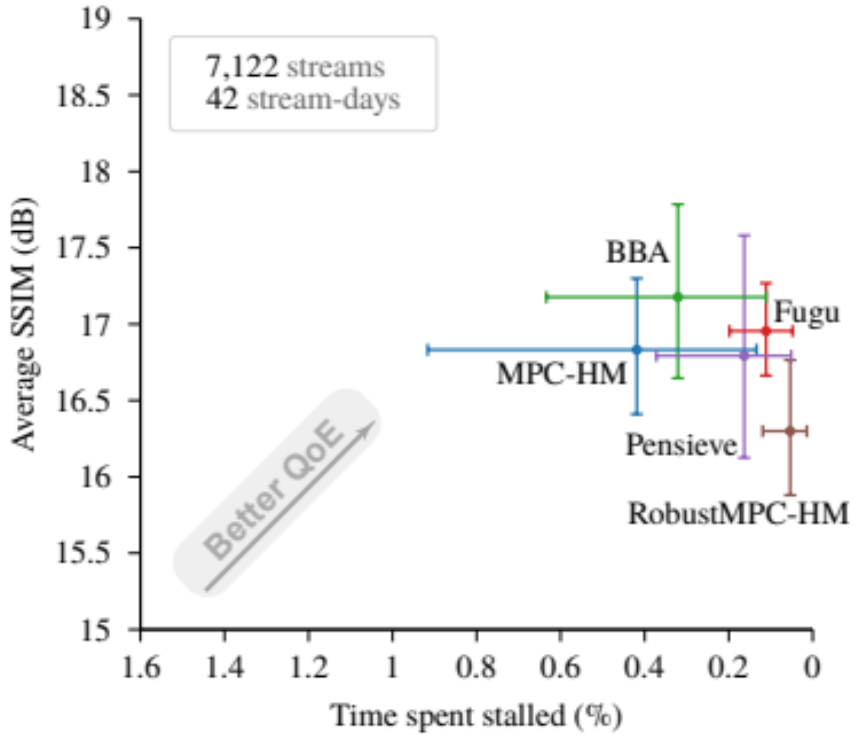
Takeaways

- **Confidence intervals in video streaming are bigger than expected**
  - Calculated using bootstrapping

Confidence intervals in video streaming are bigger than expected

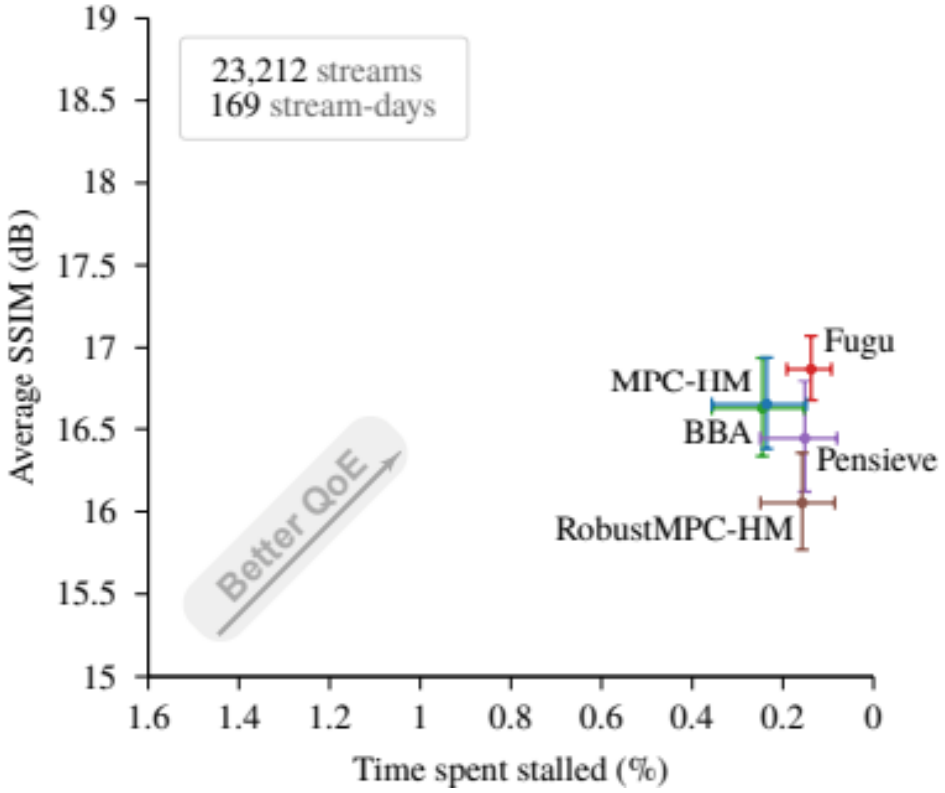


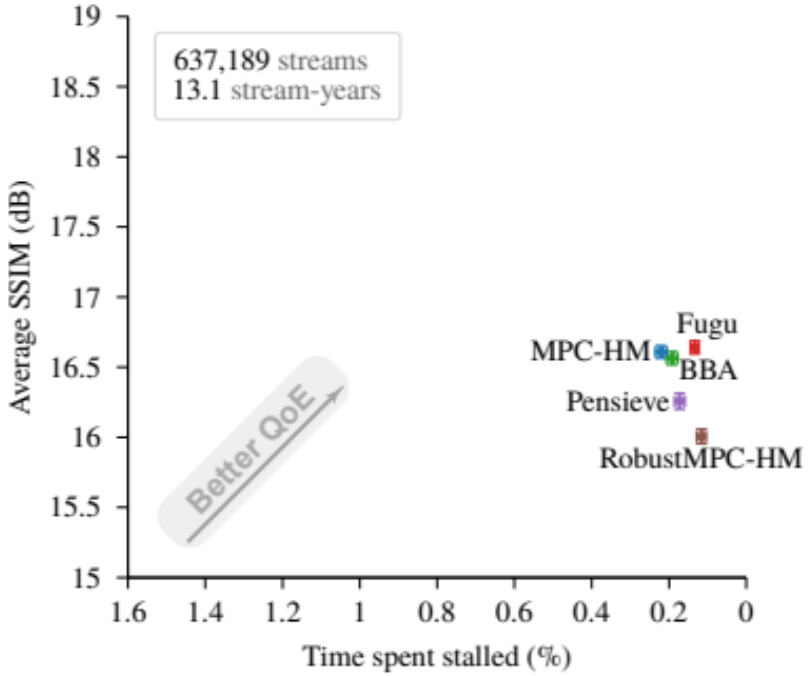
Results on the day of Jan. 26, 2019, with 17 days of video streamed to 600 users



Results in the week starting from Jan. 26, 2019, streaming 42 days of video

Confidence intervals in video streaming are bigger than expected





- Results in the month starting from Jan. 26, 2019, streaming 169 days of video

- Results in an eight-month period after Jan. 26, 2019, streaming > 13 years of video

Discussion

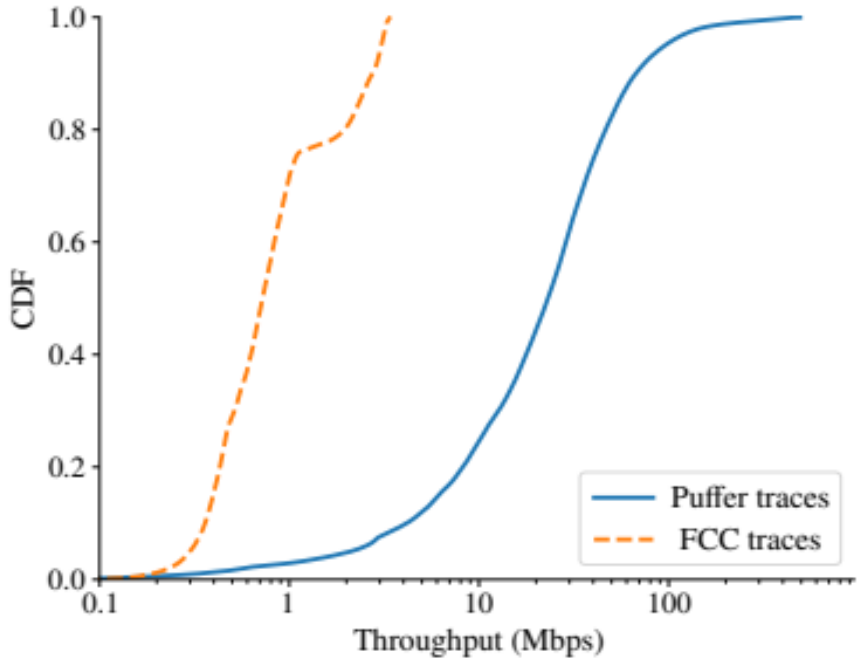
- ML systems usually tested in contained or modeled version of the problem
- Real world data heavy tailed
  - Challenge: How do we collect the right data that resembles the wild Internet

Takeaways

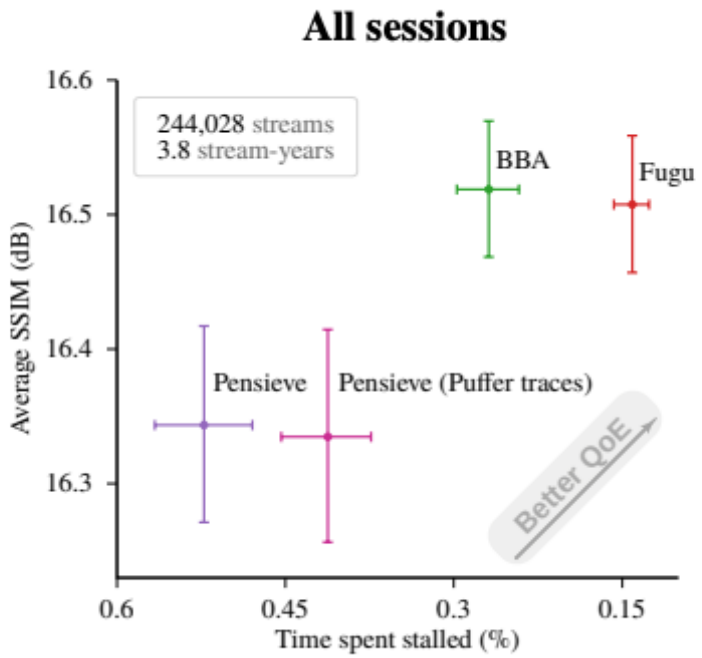
- **Confidence intervals in video streaming are bigger than expected**
  - Calculated using bootstrapping
- **In situ training on real data is useful**

Why Does Pensieve Underperform?

- Penseive trained using a simulator which may not have faithfully captured the heavy-tailed behavior of Internet
- Moreover, the training and test distributions differ



How About Training Pensieve In Situ?



In situ training helps but it is not the end of the story

Special Topics: Machine Learning (ML) for Networking

COL867

Holi, 2025

Robustness

Tarun Mangla

Recap

- Evaluating robustness of network ML

Research Question

What does it take to create a ML-based bitrate adaptation algorithm that **robustly** performs well over the **wild Internet**?

- Puffer: Designed a randomized control trial

Recap

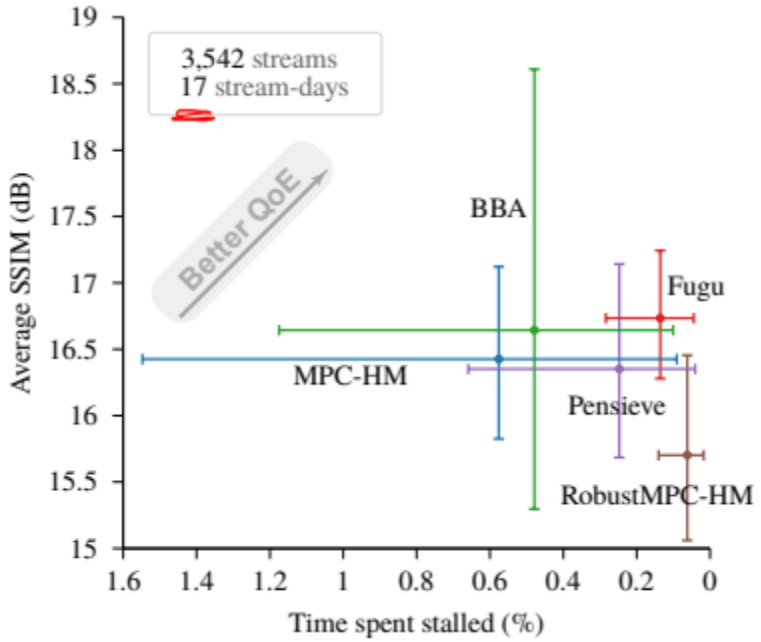
- Evaluating robustness of network ML

Research Question

What does it take to create a ML-based bitrate adaptation algorithm that **robustly** performs well over the **wild Internet**?

- Puffer: Designed a randomized control trial
- Takeaways from RCT:
  1. Confidence intervals are bigger than expected

Insight 1



Recap

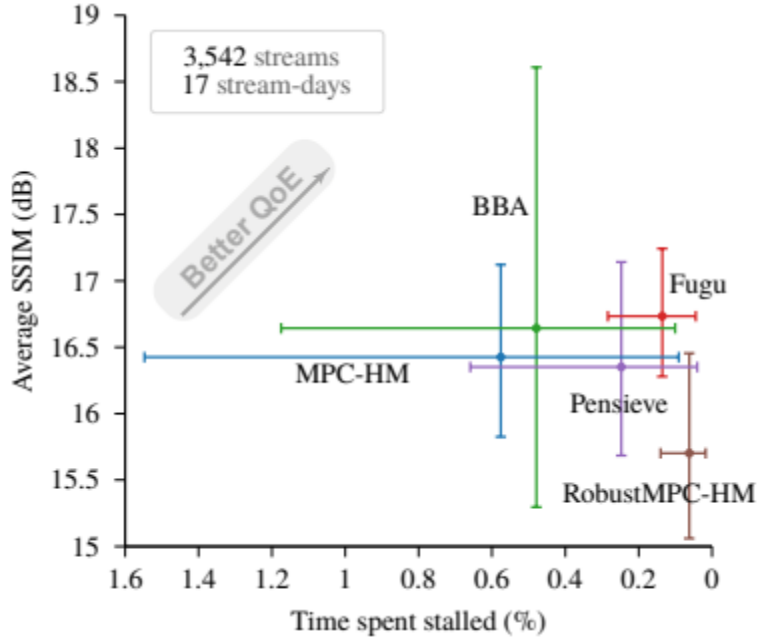
- Evaluating robustness of network ML

Research Question

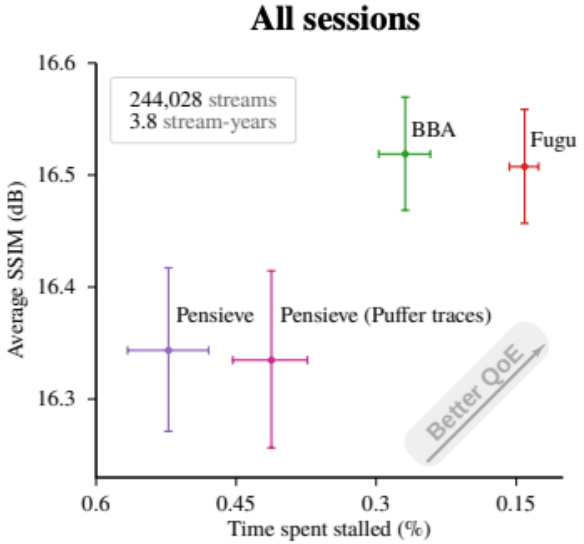
What does it take to create a ML-based bitrate adaptation algorithm that **robustly** performs well over the **wild Internet**?

- Puffer: Designed a randomized control trial
- Takeaways from RCT:
  1. Confidence intervals are bigger than expected
  2. In-situ training on real data is useful
  3. ML model matters*

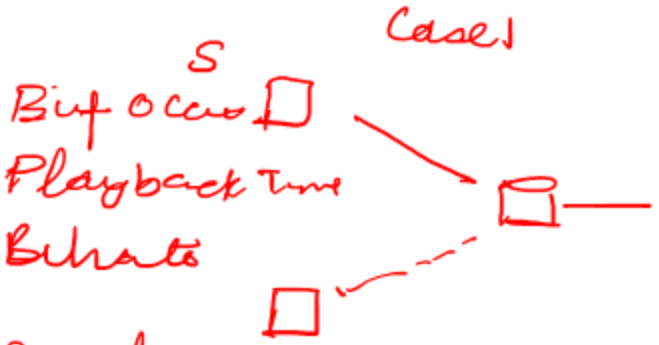
Insight 1



Insight 2



Why avoid RL models?



- Reinforcement learning is inherently hard compared to supervised learning
- Even harder for networked systems
  - Networked systems are input-driven environments
  - Reward depends not only on the local system state but exogenous input events
- RL models have high variance in such systems

Solution
1. train on large amt of data
2. Avoid RL

How to model bitrate adaptation as supervised learning problem?

Model Predictive Control

MPC DASH

- Can be formulated as a stochastic optimization problem given throughput prediction
  - Target: QoE function
  - Constraints

- Throughput prediction can be formulated as a supervised learning problem

$$\max_{R_1,\cdots,R_K,T_s} \quad QoE_1^K \tag{6}$$

$$s.t. \qquad t_{k+1} = t_k + \frac{d_k(R_k)}{C_k} + \Delta t_k, \tag{7}$$

$$C_k = \frac{1}{t_{k+1} - t_k - \Delta t_k} \int_{t_k}^{t_{k+1}-\Delta t_k} C_t \, dt, \tag{8}$$

$$B_{k+1} = \left(\left(B_k - \frac{d_k(R_k)}{C_k} \right)_+ + L - \Delta t_k \right)_+, \tag{9}$$

$$B_1 = T_s, \quad B_k \in [0, B_{max}] \tag{10}$$

$$R_k \in \mathcal{R}, \quad \forall k = 1, \cdots, K. \tag{11}$$

Bitrate Adaptation Problem

- Can be formulated as a stochastic optimization problem given throughput prediction
  - Target: QoE function
  - Constraints

- ~~Throughput prediction~~ Time to download chunk prediction can be formulated as a supervised learning problem

$$\max_{R_1,\cdots,R_K,T_s} \quad QoE_1^K \tag{6}$$

$$s.t. \qquad t_{k+1} = t_k + \frac{d_k(R_k)}{C_k} + \Delta t_k, \tag{7}$$

$$C_k = \frac{1}{t_{k+1} - t_k - \Delta t_k} \int_{t_k}^{t_{k+1}-\Delta t_k} C_t \, dt, \tag{8}$$

$$B_{k+1} = \left(\left(B_k - \frac{d_k(R_k)}{C_k} \right)_+ + L - \Delta t_k \right)_+, \tag{9}$$

$$B_1 = T_s, \quad B_k \in [0, B_{max}] \tag{10}$$

$$R_k \in \mathcal{R}, \quad \forall k = 1, \cdots, K. \tag{11}$$

Fugu's Transmission Time Predictor

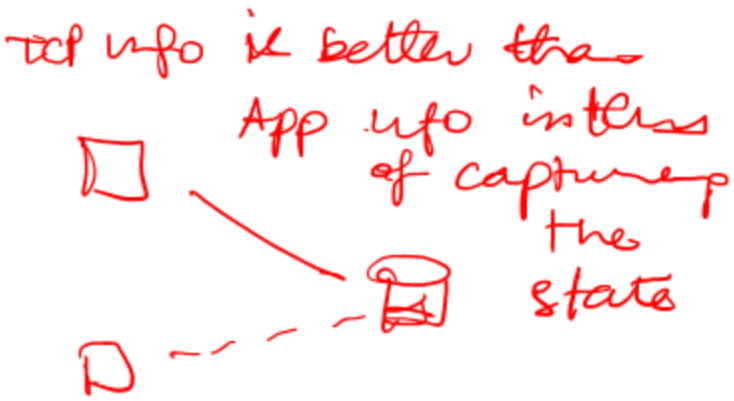
- Neural Network predicts "how long would each chunk take?"

- Input:
  - ~~Size of~~ transmission times of past chunks
  - Size of a chunk to be transmitted
  - Low-level TCP statistics

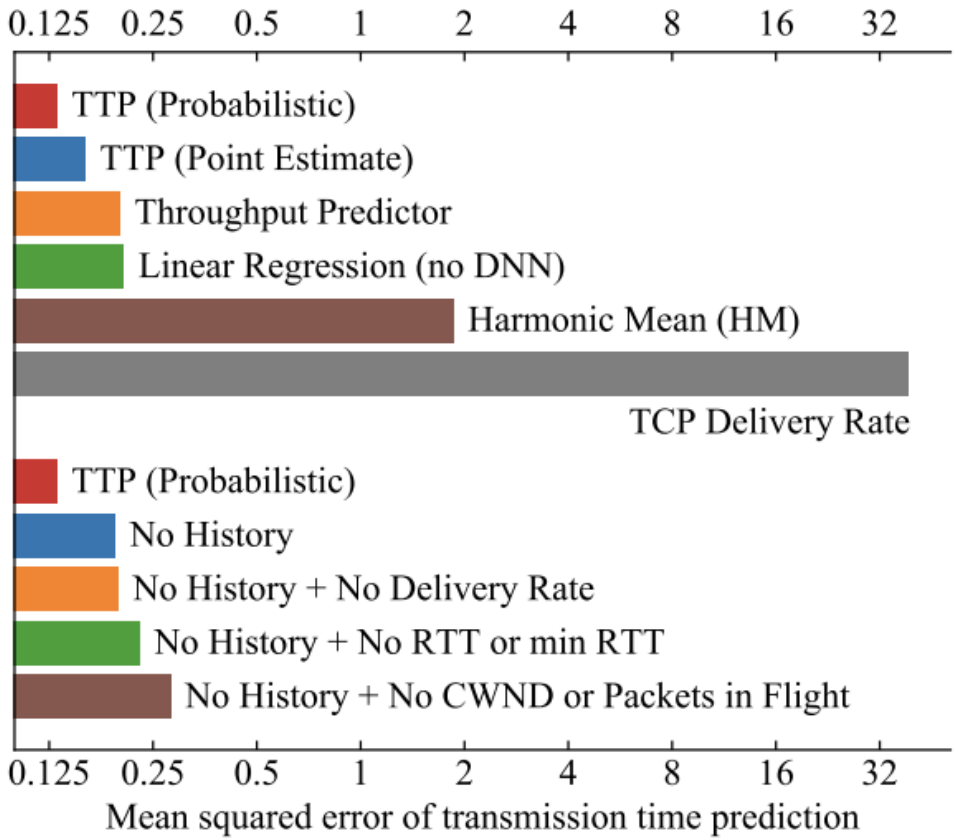
- Output
  - Probability distribution over transmission time

Takeaways

- **Confidence intervals in video streaming are bigger than expected**
- **In situ training on real data is useful**
- **ML model matters**

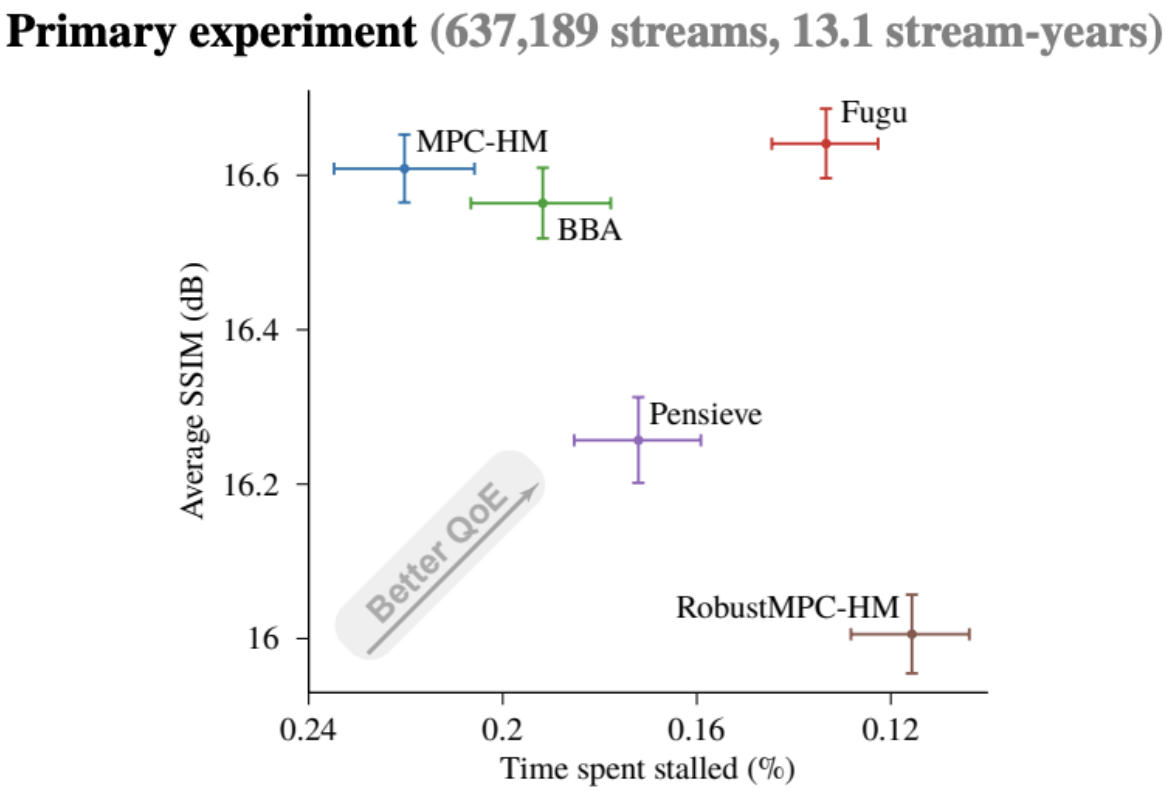


Error in Transmission Time Prediction



Directly predicting transmission time is better than using throughput prediction

Fugu Performance



Next Few Classes

Case studies around robustness in networking

- How to train a robust network ML?
  - Puffer – evaluating robustness
  - **Genet – training robust RL models**
- How to handle drift?

What if I need to use RL?

- Why would I need to use RL? Why not use SL?
  - ↳ Continuous learning
  - ↳ Labeled data is not present

Bitrate Adaptation

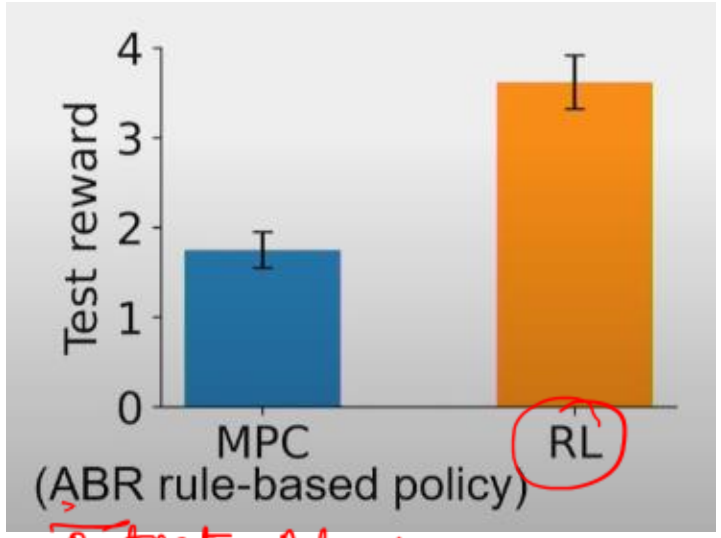
(i) Not clear what is optimal

- Let's revisit the challenges associated with RL

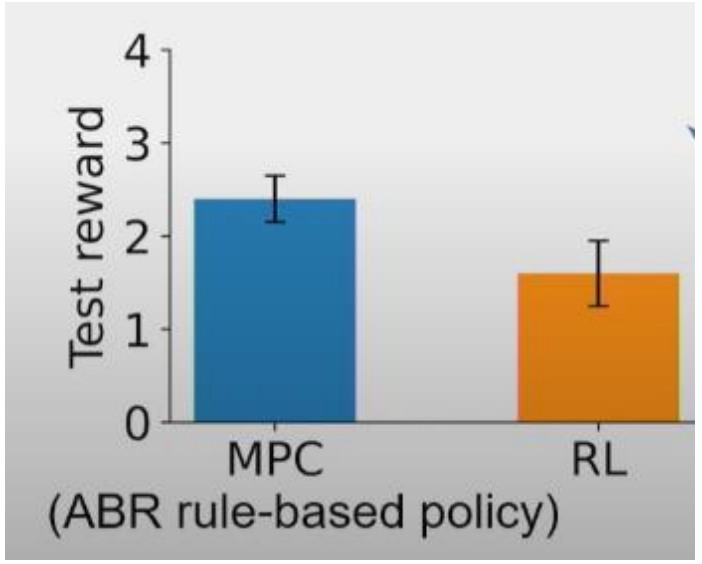
Challenge 1: Poor Generalizability

- Policy may generalize poorly to unseen network environments

Test: BW range 1-10 Mbps
Train: BW range 1-10 Mbps



Test: BW range 1-100 Mbps
Train: BW range 1-10 Mbps



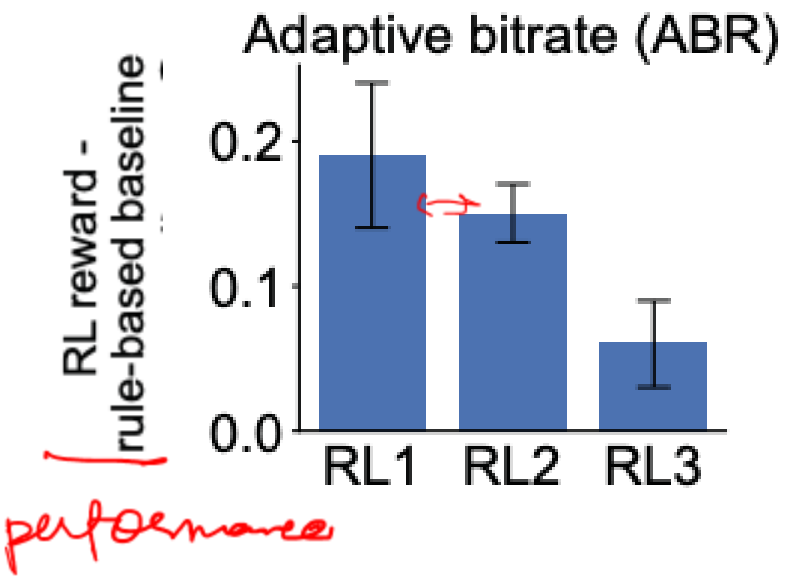
Challenge 2: Poor Converged Performance for Wide Training Distribution

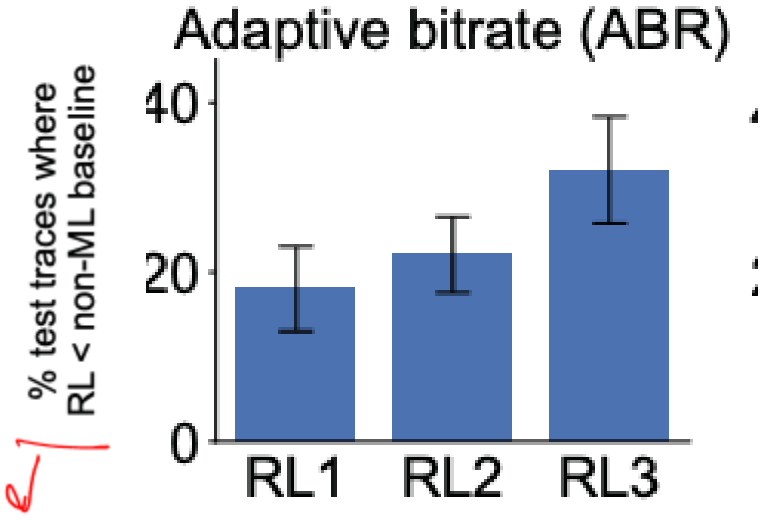
- If training distribution is wide, RL may perform poorly even if test data is from the same distribution

ABR Parameter	RL1	RL2	RL3	Default	Original
Max playback buffer (s)	[2, 10]	[2, 50]	[2, 100]	60	60
Video chunk length (s)	[1, 4]	[1, 6]	[1, 10]	4	4
Min link RTT (ms)	[20, 30]	[20, 220]	[20, 1000]	80	80
Video length (s)	[40, 45]	[40, 200]	[40, 400]	196	196
Bandwidth change interval (s)	[2, 2]	[2, 20]	[2, 100]	5	
Max link bandwidth (Mbps)	[2, 5]	[2, 100]	[2, 1000]	5	

Challenge 2: Poor Converged Performance for Wide Training Distribution

- If training distribution is wide, RL may perform poorly even if test data is from the same distribution





Possible Issue: RL Training

- Training done in iteration
- Each iteration, uniformly sample subset of environment
- Update the policy using backpropagation

- Uniform sampling has low efficiency
- Stuck in hard-to-improve environments

Algorithm 1 Traditional Reinforecment Learning (RL)

Input: Ω: space of configurations, θ: initial policy parameters, N_{iters}: # of iterations

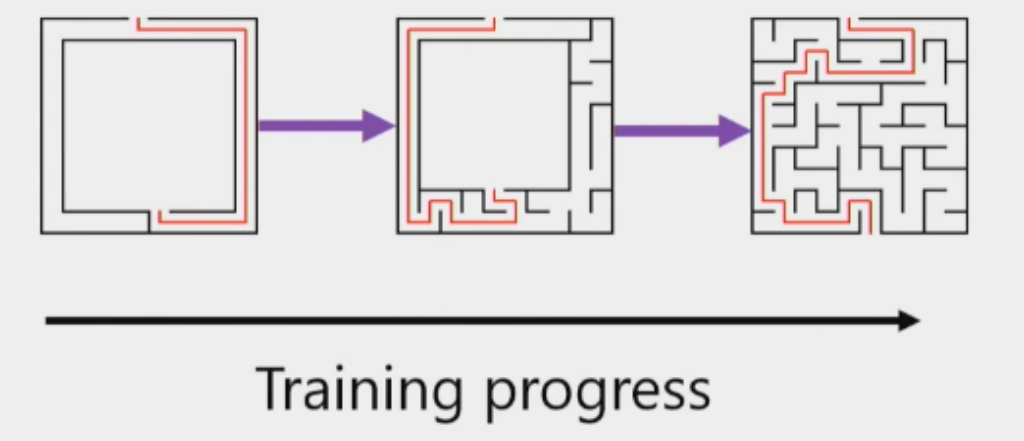
Output: θ: returned policy parameters

for i from 1 to N_{iters} **do**
  $\Phi_{rand} \leftarrow \emptyset$
  for 1 **to** K **do** ▷ K: # configs per iteration
    $p_i \sim Random(\Omega)$ ▷ Uniformly sampled config in Ω
    for 1 **to** N **do** ▷ N: # random envs per config
      $E \leftarrow S(p_i)$ ▷ Create a simulated env by p_i
      rollout $\phi \sim \pi_\theta(\cdot; E)$ ▷ Rollout policy π_θ on E
      $\Phi_{rand} \leftarrow \Phi_{rand} \cup \phi$
    end for
  end for
  with Φ_{rand} update:
  $\theta \leftarrow \theta + \nu \nabla_\theta J(\pi_\theta)$ ▷ Gradient update with rate ν
end for
return θ

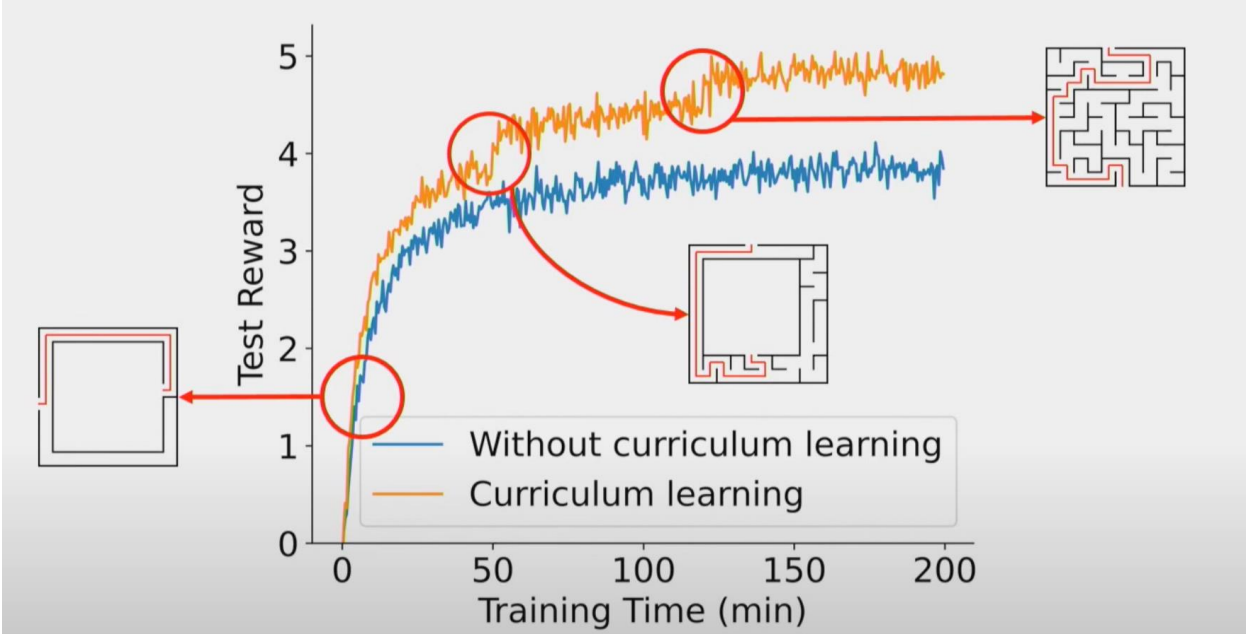
Curriculum Learning

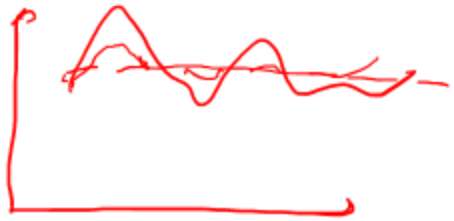
- Over time, iteratively introduce rewarding environments for the current RL model

- A rewarding environment has high potential improvements for RL training
  - i.e., after introducing the new environment to RL training, RL performance can be much better



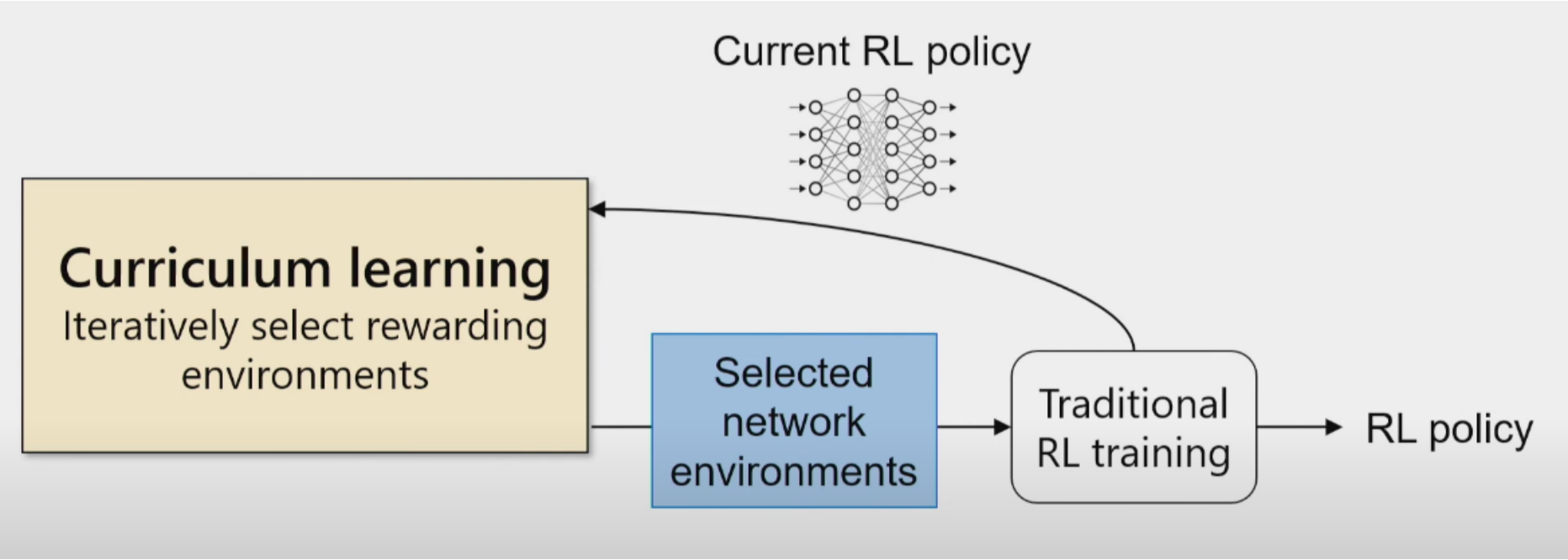
How Does Curriculum Learning Benefit RL?





Intuition: Optimize a family of gradually less smooth loss functions; prevents it from being trapped in local minima

Curriculum Learning in Networking



Challenge: How to define rewarding environment in networked systems?

① . Domain knowledge

↳ In-lab easier → Real world Int

↳ low variance → High variance

② Non-ML baseline

Bitrate Adaptive : RL

BBA → calculate Reward

calcul Reward (E_i)

↓ E_i

Pick ~~Reward~~ Max Reward

Gap-to-baseline

(E_1 . . . . E_n)

Strawman Approaches

- Inherent properties
  - E.g., high bandwidth variance → more rewarding
- Gap to optimum
  - Higher the gap → more rewarding
- Performance of rule-based baselines
  - Lower performance → more rewarding

① → calculating optimum is challenge

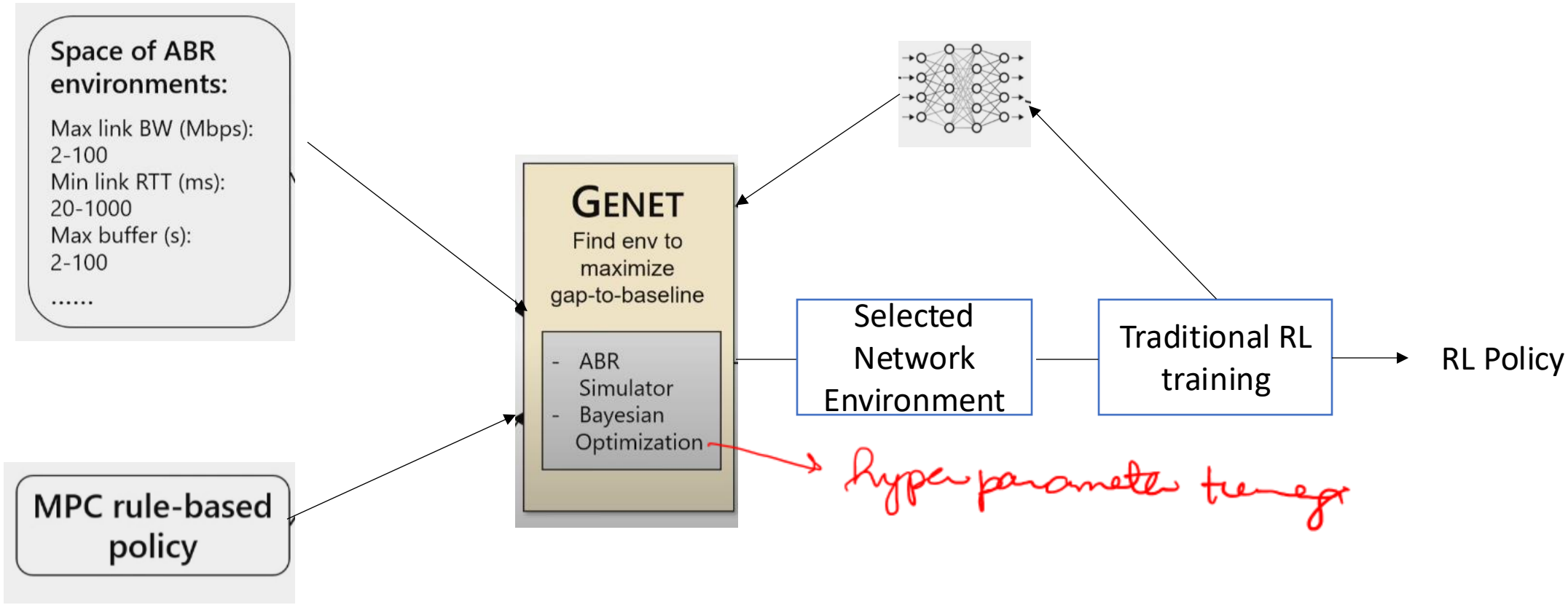
② optimum might be hard to achieve

Can we find a better indicator?

GENET Indicator: Gap-to-baseline

- Gap-to-baseline = baseline reward – current RL reward
- Baseline estimates how difficult an environment is
- Gap indicates room for RL improvement

GENET Workflow



Evaluation Setup

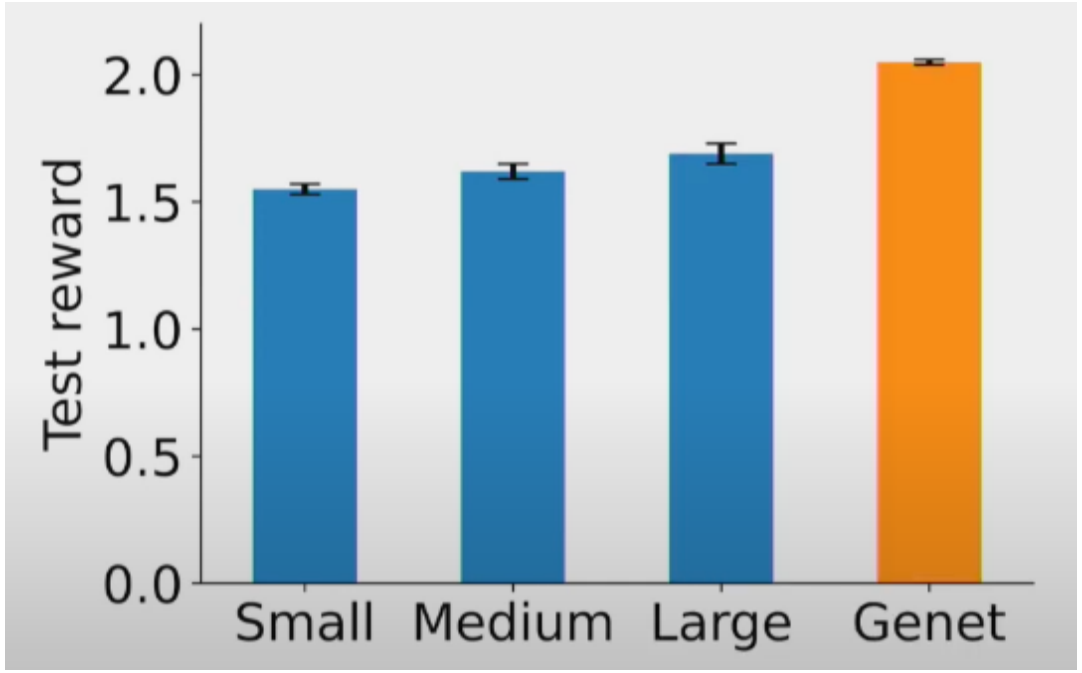
- Application: ABR

- Traces
  - Synthetic: BW (0-100 Mbps), BW frequency (2-10 seconds)
  - Real: FCC broadband, Puffer

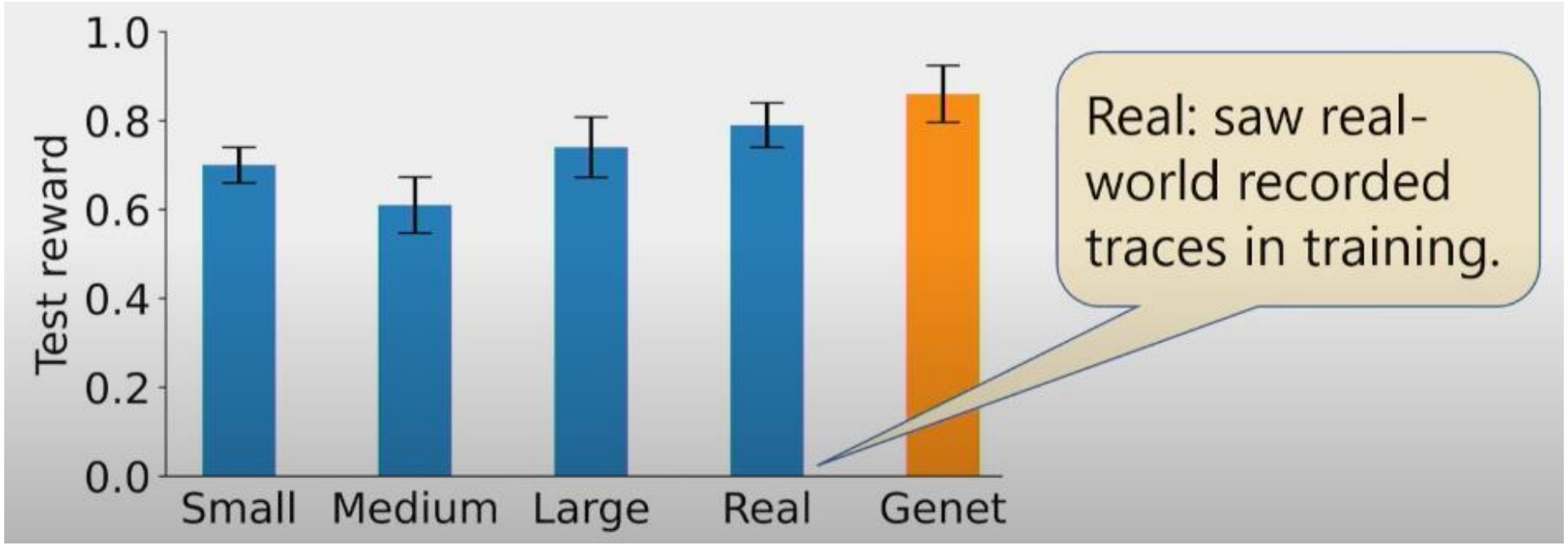
- Baseline:
  - Traditional RL with small, medium, and large training distribution
  - Rule-based baselines: MPC, BBA etc.

Result: Train on Synthetic, Test on Synthetic

ABR Parameter	RL1	RL2	RL3	Default	Original
Max playback buffer (s)	[2, 10]	[2, 50]	[2, 100]	60	60
Video chunk length (s)	[1, 4]	[1, 6]	[1, 10]	4	4
Min link RTT (ms)	[20, 30]	[20, 220]	[20, 1000]	80	80
Video length (s)	[40, 45]	[40, 200]	[40, 400]	196	196
Bandwidth change interval (s)	[2, 2]	[2, 20]	[2, 100]	5	
Max link bandwidth (Mbps)	[2, 5]	[2, 100]	[2, 1000]	5	



Result: Train on Synthetic, Test on Real World



TCP

Congestion control

R: Throughput ↑ α Latency − β Loss

Baseline : TCP CUBIC

E: BW [0 - 100]

Loss: [0 - 10]

Latency [0 - 100]

Takeaways

- Gap between training in-lab to real-world testing
- Internet exhibits heavy-tailed distribution
- Training in-situ is beneficial
- RL models are particularly at disadvantage –> higher variance
  - Careful modeling of the problem (Use SL instead of RL)
  - Use curriculum learning for more efficient training

Next Class

Case studies around robustness in networking

- How to train a robust network ML?
  - Puffer – evaluating robustness
  - Genet – training robust RL models
- **How to handle drift?**

Overall → Accuracy

Netflix → α

Youtube → β

Phone → γ

$w_1\alpha + w_2\beta + w_3\gamma$

shift in $P(X)$: Data drift
input distribution

shift in $P(Y|X)$: Concept drift
↳ Always an issue

Retrain ? N/w security

Special Topics: Machine Learning (ML) for Networking

COL867

Holi, 2025

Robustness

Tarun Mangla

Today's Class

Task: App Classification

Hulu (α) / Netflix (β) / Youtube

10 / 50 / 40

After Drift

40 / 30 / 30

Case studies around robustness in networking

- How to train a robust network ML?
  - Puffer – evaluating robustness
  - Genet – training robust RL models
- **How to handle drift?**

Overall accuracy:

$0.6\alpha + 0.4\beta$ $\alpha = \beta$

$0.3\alpha + 0.7\beta$

Drift in

↳ Data drift : $P(X)$: $P(Y|X)$ has changed (Not true)

↳ Concept drift : $P(Y|X)$

Re-train :

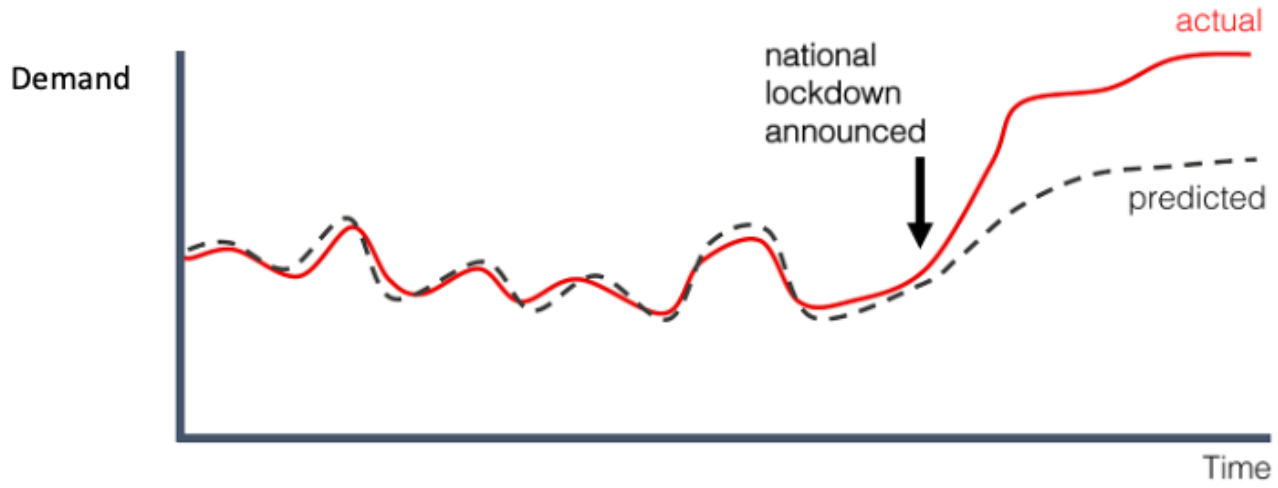
(1) Computationally expensive

(2) labeled data is expensive

What is Concept Drift?

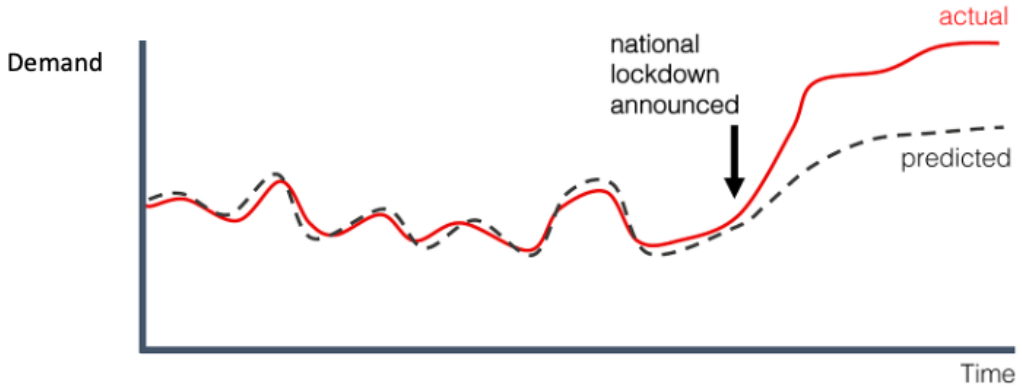
Data drift led to concept drift

- Change in relationship between input and output data in the underlying learning problem

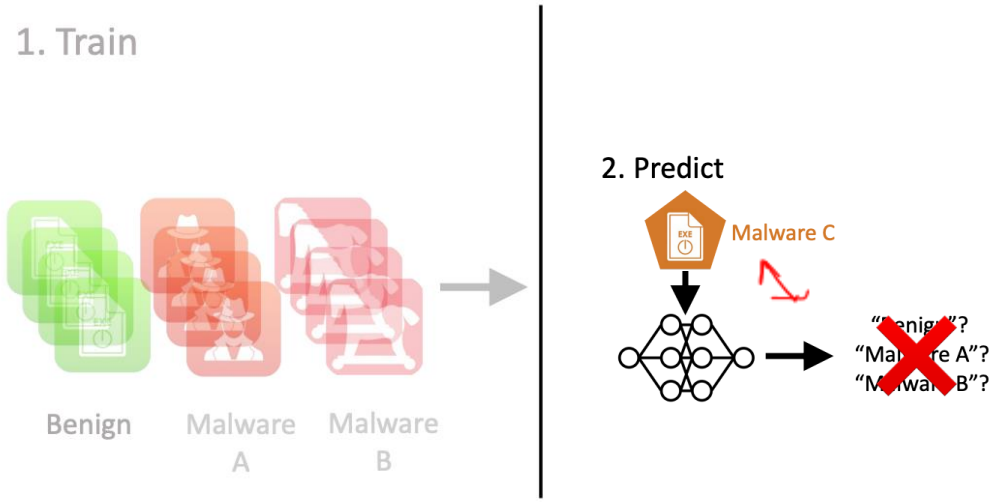


What is Concept Drift?

- Change in relationship between input and output data in the underlying learning problem



Exogenous events



Zero-day attacks

An Ideal ML System

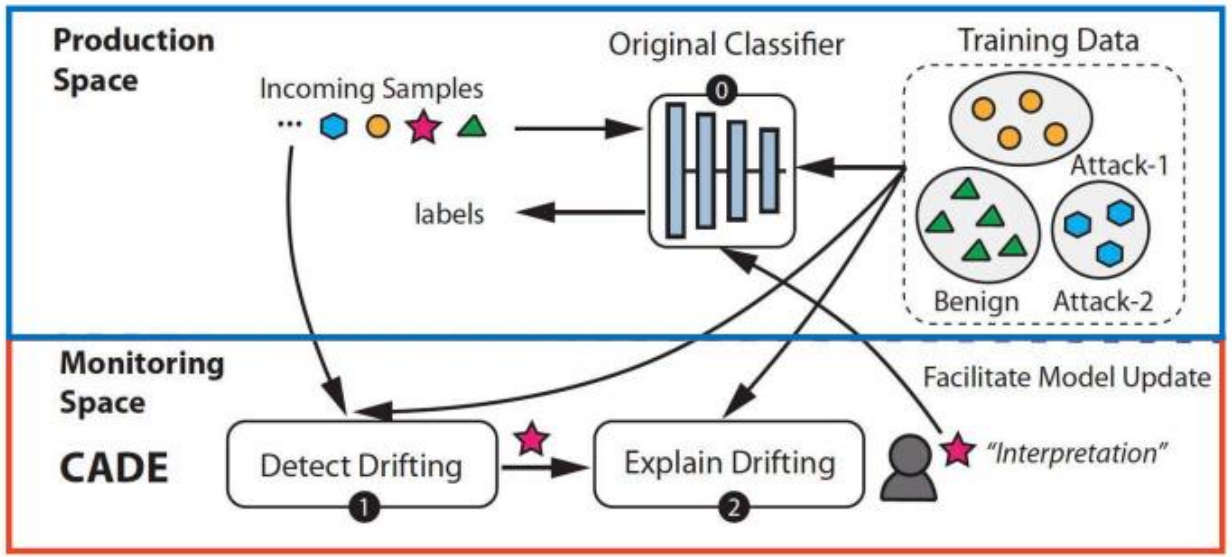
- Quickly and accurately able to detect concept drift

CADE: Detecting and Explaining Concept Drift Samples for Security Applications

Limin Yang*, Wenbo Guo†, Qingying Hao*, Arridhana Ciptadi‡
Ali Ahmadzadeh‡, Xinyu Xing†, Gang Wang*
*University of Illinois at Urbana-Champaign †The Pennsylvania State University ‡Blue Hexagon
liminy2@illinois.edu, wzg13@ist.psu.edu, qhao2@illinois.edu, {arri, ali}@bluehexagon.ai, xxing@ist.psu.edu, gangw@illinois.edu

- Adapt itself to concept drift

CADE: When NOT to Predict?



Goals

1. Detect drifting samples
2. Find a small subset of important features that explain why the drifting sample is different from training data

Context:

- Supervised learning, classification
- Goal is to detect drift due to a new class

Model M → Classification into N classes

Instance x_i

Pick all example

→ Class 1

↳ Centroid of Class 1

↳ 'Distance' from centroid

D > Threshold

D > α (Variance of distance in cluster)

or

α (95% of distance)

Challenges

① Feature might be a mix of continuous & discrete

② If the dimensionality is large., curse of dimensionality

② Classification model give confidence value

If low confidence then defer

Classify & verify later

→ High latency

→ Cumbersome

How to Detect Drifting Samples in a Classification Problem?

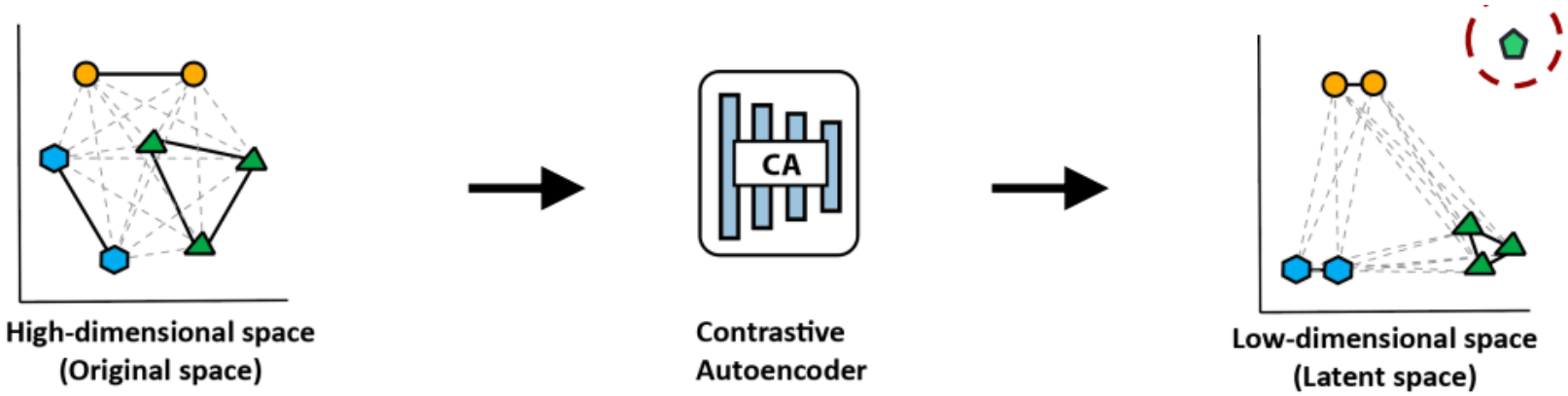
- Check errors post-prediction
  - Verification latency

- Check confidence of prediction
  - Problematic for a new class

- Distance between training data and test sample
  - Distance loses effectiveness in high-dimensional data

Contrastive Learning

- Use contrastive learning to learn a compressed representation of training data by contrasting with existing samples



- Loss function: $\min_{\theta,\phi} \mathbb{E}_{\boldsymbol{x}} \|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2^2 + \lambda \mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j}\left[(1-y_{ij})d_{ij}^2 + y_{ij}(m-d_{ij})_+^2\right]$

$y_{ij} = 1$ if x_i & x_j belong to diff class

0 otherwise

case 1: $\lambda \mathbb{E}_{x_i x_j}(d_{ij}^2)$

case 2: $\lambda \mathbb{E}_{x_i x_j}[(m-d_{ij})_+^2]$

How to detect drifting sample using contrastive learning?

- Calculate centroids for each class in the latent representation
- A sample that is far away from ANY existing families' centroids is a potential drifting sample
- Rank samples based on the distance from the centroid
  - How to define a threshold?

$$d_i^y = \| \hat{x}^y - c_i \|_2$$

$$d_i^y - d_i^{median} \geq T$$

$$\frac{d_i^y - d_i^{mean}}{d_i^{std\text{-}dev}} : \text{outlier}$$

Median Absolute Deviation

$$\frac{d_i^y - d_i^{median}}{\text{Median Absolute Dev}} \geq T \to 3$$

$$\text{Median}\left(\| d_i^j - d_i^{median} \| \right)$$

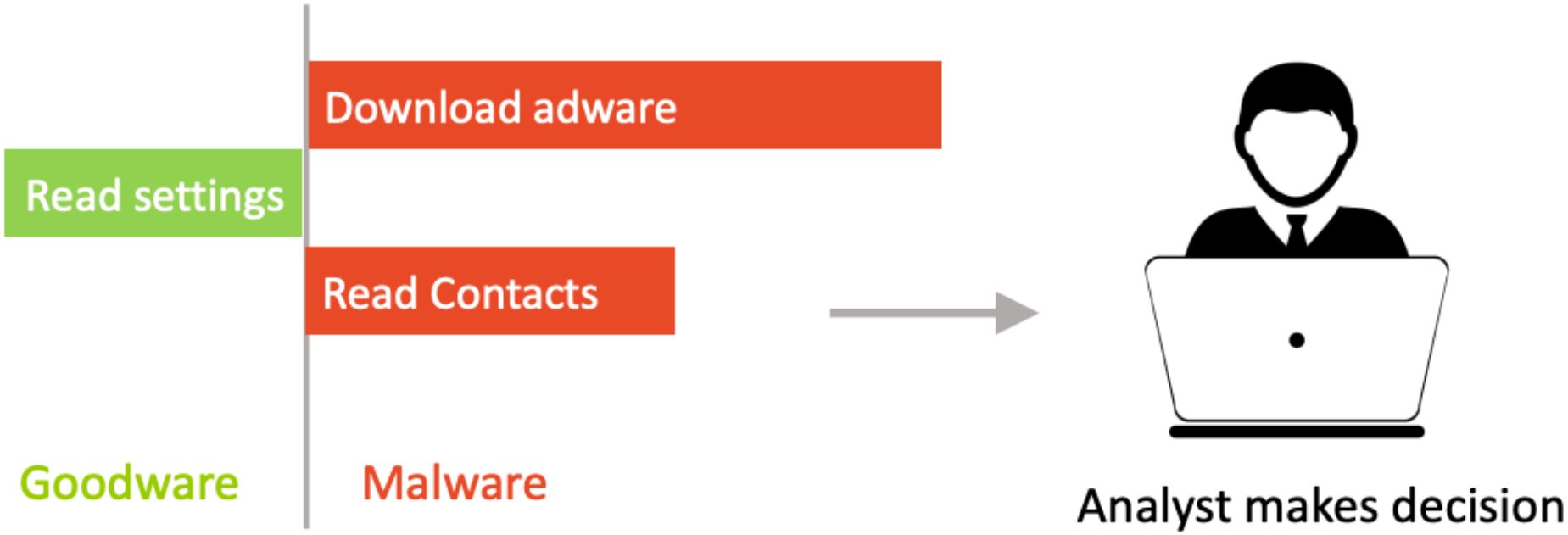
For each class i

Check if distance in latent space > Threshold

Rank all the instance based on distance

↳ Explain why it is drifting

How to Explain Drifting Samples?



- High dimensionality makes it difficult for the analyst to make decisions

Can we make the task easier?

Explanation Methods

- Identify a small set of important features that make the drifting sample an outlier

- **Idea:** Use distance-based explanation
  - Which features lead to a higher distance between drifting sample and centroid?
  - **Challenge**: Distance is calculated in the latent dimension

- **Solution:** Perturb the original features and observe the distance changes in latent space

Distance-based Explanation

- **Problem:** Given a drifting instance, perturbing which minimal set of features will bring the instance closer to the centroid
- **Challenge:** How much to perturb?
- **Solution:** Replace the feature values with the instance that is closest to the centroid

$$\mathbb{E}_{\boldsymbol{m}\sim Q(\boldsymbol{p})}\|\hat{\boldsymbol{z}}_t - \boldsymbol{c}_{y_t}\|_2 + \lambda_1 R(\boldsymbol{m},\boldsymbol{b}),$$

$$\hat{\boldsymbol{z}}_t = f(\boldsymbol{x}_t \odot (1-\boldsymbol{m}\odot\boldsymbol{b}) + \boldsymbol{x}_{y_t}^{(c)} \odot (\boldsymbol{m}\odot\boldsymbol{b})),$$

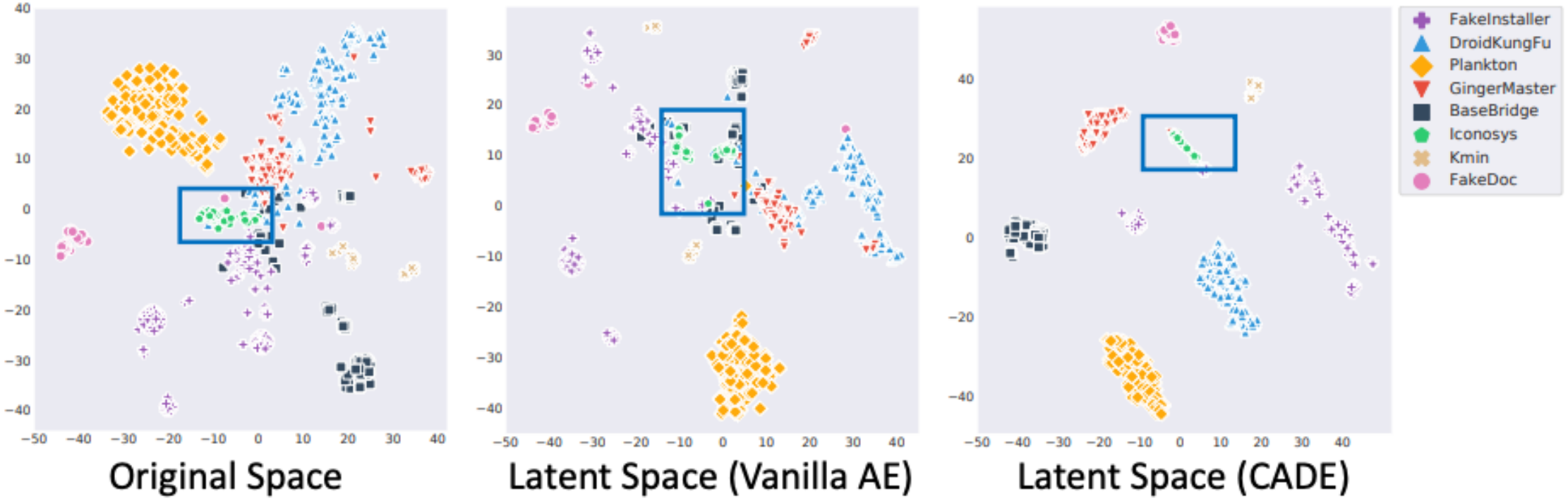
$$R(\boldsymbol{m},\boldsymbol{b}) = \|\boldsymbol{m}\odot\boldsymbol{b}\|_1 + \|\boldsymbol{m}\odot\boldsymbol{b}\|_2, \quad Q(\boldsymbol{p}) = \prod_{i=1}^{q} p(\boldsymbol{m}_i|p_i).$$

Results

Iteratively choose a family as the unseen family and report the average results here.

Method	Drebin (Avg±Std)		IDS2018 (Avg±Std)	
	F_1	Norm. Effort	F_1	Norm. Effort
Vanilla AE	0.72±0.15	1.48±0.31	0.74±0.12	1.74±0.40
Transcend	0.80±0.12	1.29±0.45	0.65±0.46	1.45±0.57
CADE	**0.96±0.03**	**1.00±0.09**	**0.96±0.06**	**0.95±0.07**

Why CADE Works?



T-SNE visualization for Drebin dataset (Unseen family: Iconosys)

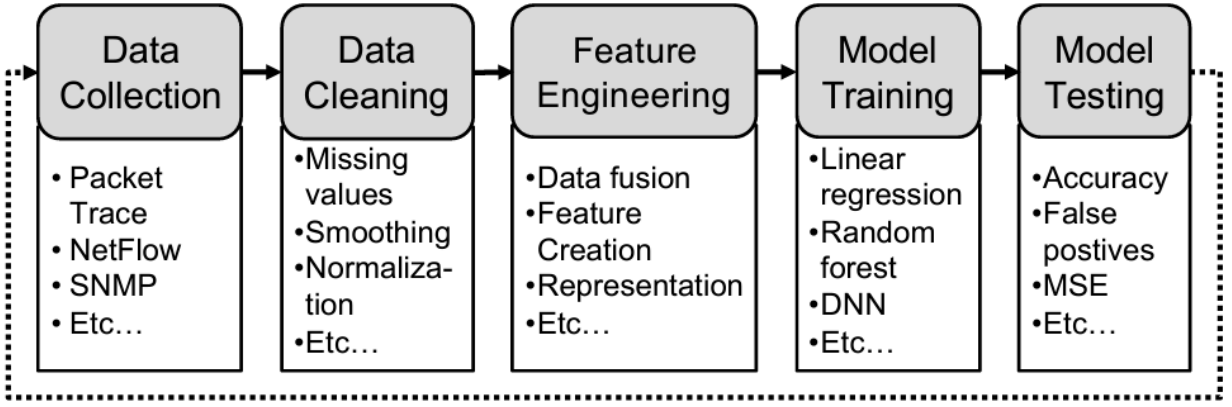
Next Class

Module 1: Case studies of specific network learning tasks

Module 2: Task-agnostic automatic ML pipelines for networks

Module 3: Beyond feature engineering and modeling

- Network telemetry
- Robustness
- **Explainability**
- Synthetic data generation*
- Data imputation
- Formal verification
- ..



Special Topics: Machine Learning (ML) for Networking

COL867

Holi, 2025

Robustness

Tarun Mangla

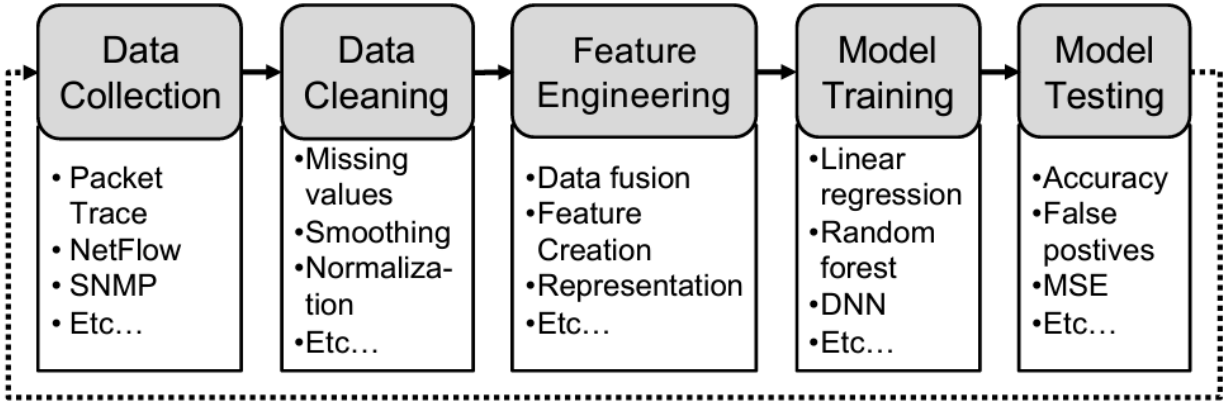
ML for Networks

Module 1: Case studies of specific network learning tasks

Module 2: Task-agnostic automatic ML pipelines for networks

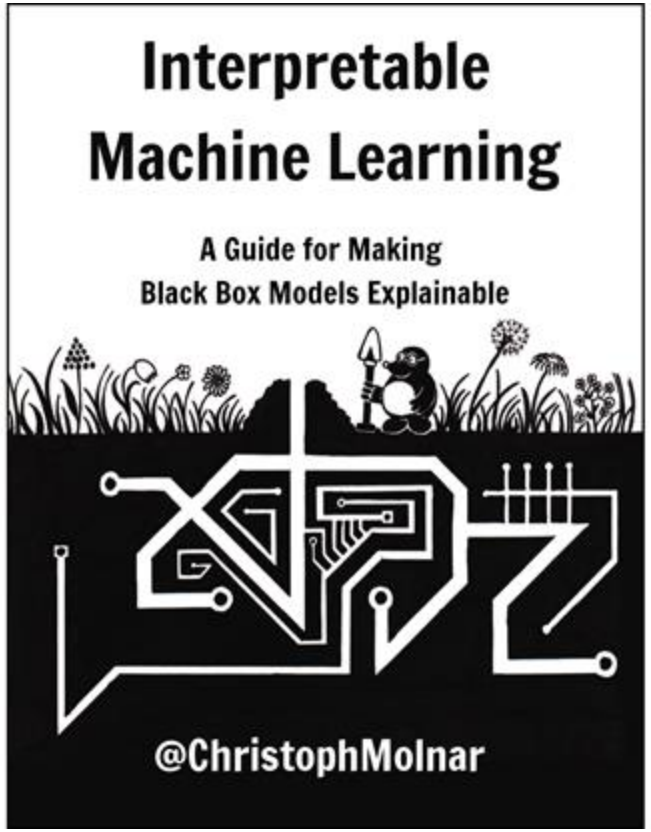
Module 3: Beyond feature engineering and modeling

- Network telemetry
- Robustness
- **Explainability**
- Synthetic data generation*
- Data imputation
- Formal verification



What is Explainable AI?

- **Explainable Machine Learning** refers to methods and models that make the behavior and predictions of machine learning systems understandable to humans
- **Explainability vs interpretability**:
  - Interpretability: Describe the internals of a system in a way which is understandable to humans
  - Explainability: produce insights about the causes of model's decisions

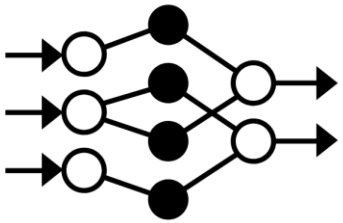


Source: https://christophm.github.io/interpretable-ml-book/

Why we need Explainable ML?

① Sanity check
② Human-in-the-loop
↳ helpful in that case
② Trust

12/02	Timestamp
⋮	⋮
7	Avg. length of sent packets
15 pkt/s	Avg. packet (pkt) sending frequency



Intrusion

① Ablation study

② ~~Bubbl~~ Use white box model
Black-box model
Neural network

Diagnose errors

Build Trust

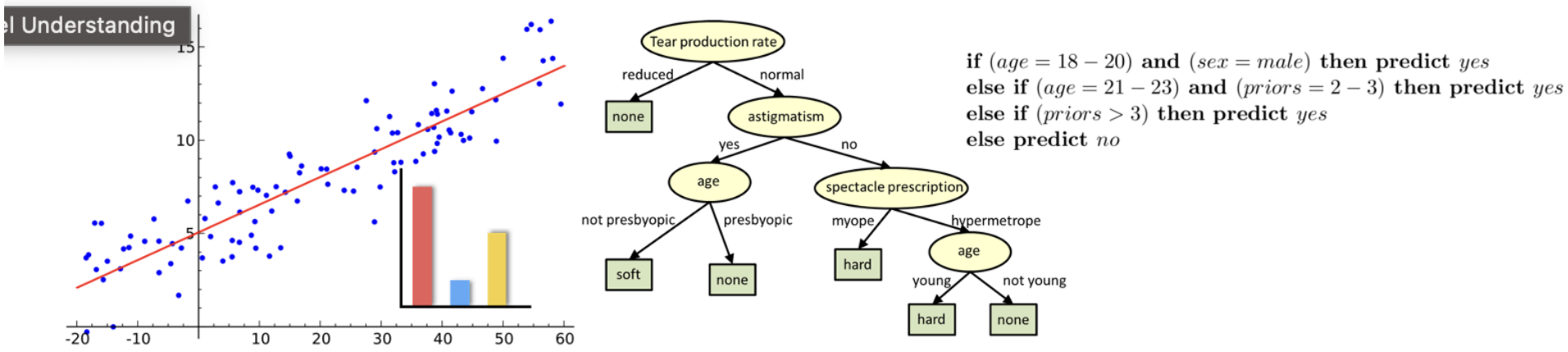
Patch errors

Learn new knowledge

How to build explainable AI?

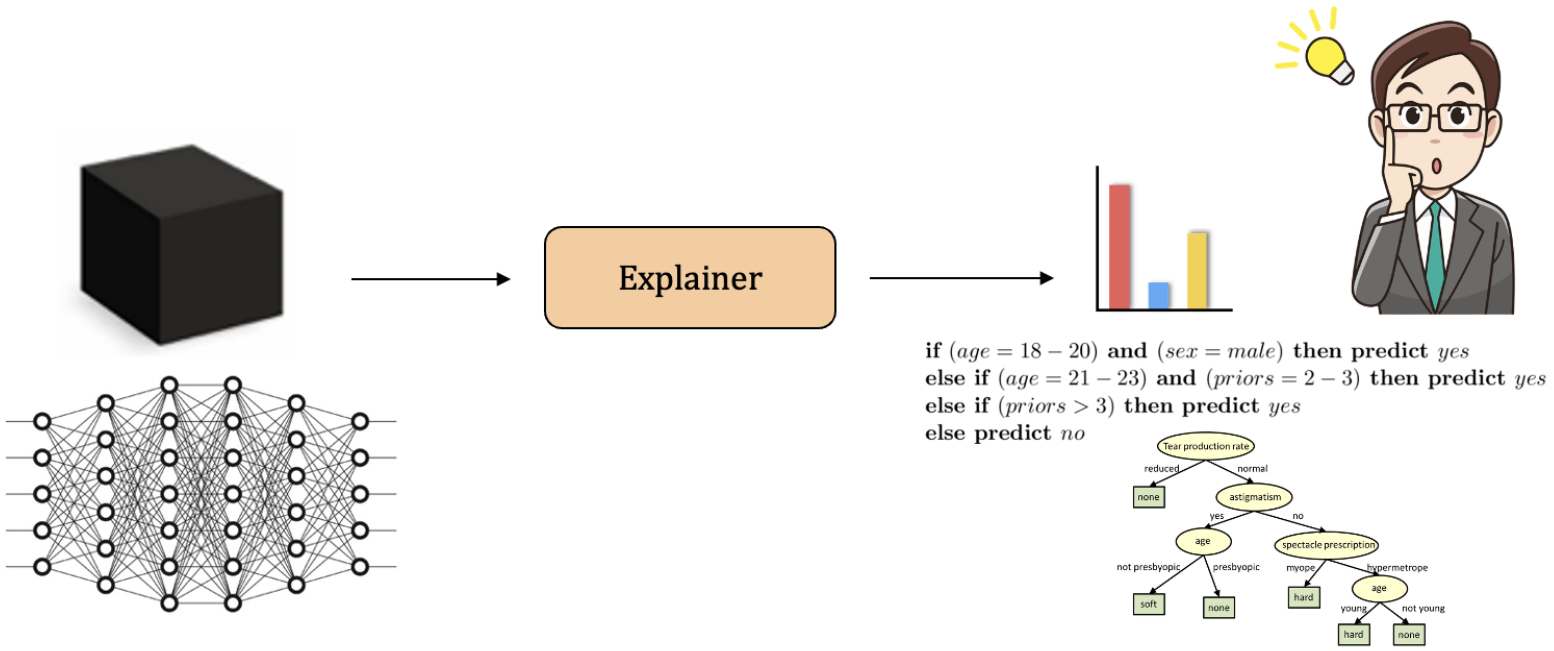
Achieving Model Explainability

- **Approach #1:** Build inherently interpretable ML models

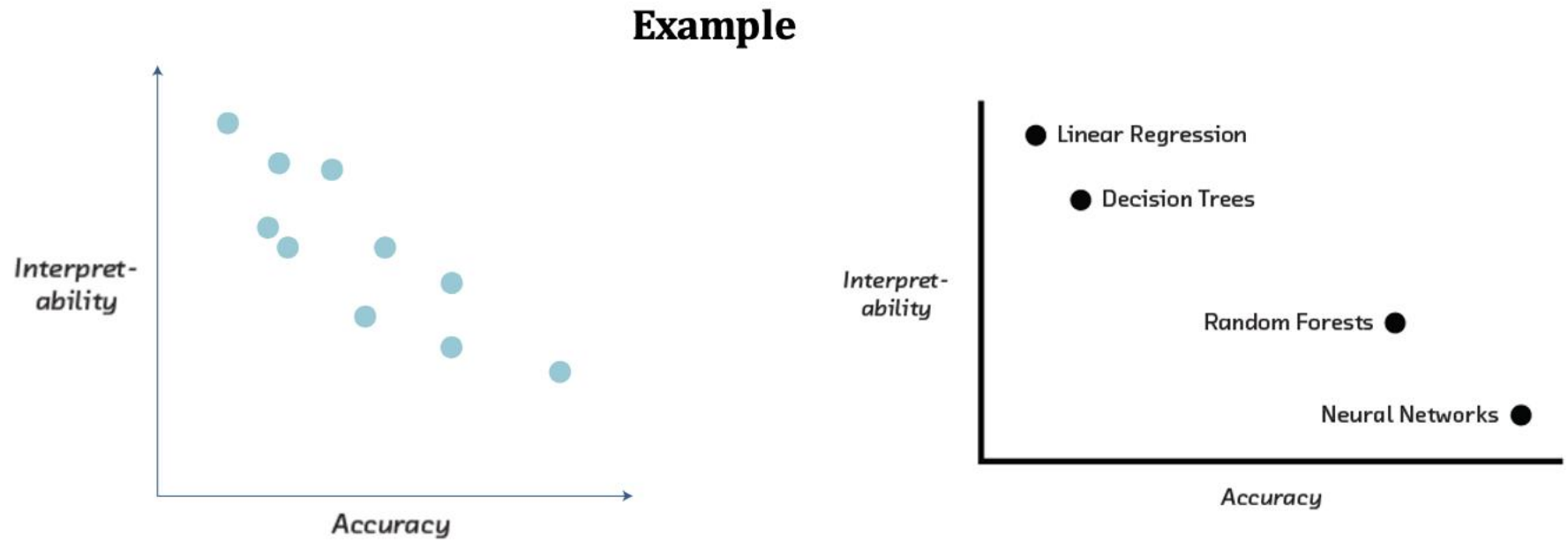


Achieving Model Explainability

- **Approach #1:** Build inherently interpretable ML models
- **Approach #2:** Explain pre-built models in a posthoc manner



Inherently Explainable vs Posthoc Explainability



In ***certain*** settings, *accuracy-interpretability trade offs* may exist.

Explainability

- Global explainability
  - Given a model, what is the impact of a feature on average model accuracy
- Local
  - Given a specific instance, what features contributed to predictions
  - Examples, LIME, SHAP

Local Explanations: LIME

- Explain decisions of any model in a local region around a particular point
- Key idea: Approximate the ML model in the small region by learning a sparse linear model

LIME (Ribeiro et. al.)

LIME intuition

$\mathbb{R}^n \rightarrow \mathbb{R}$

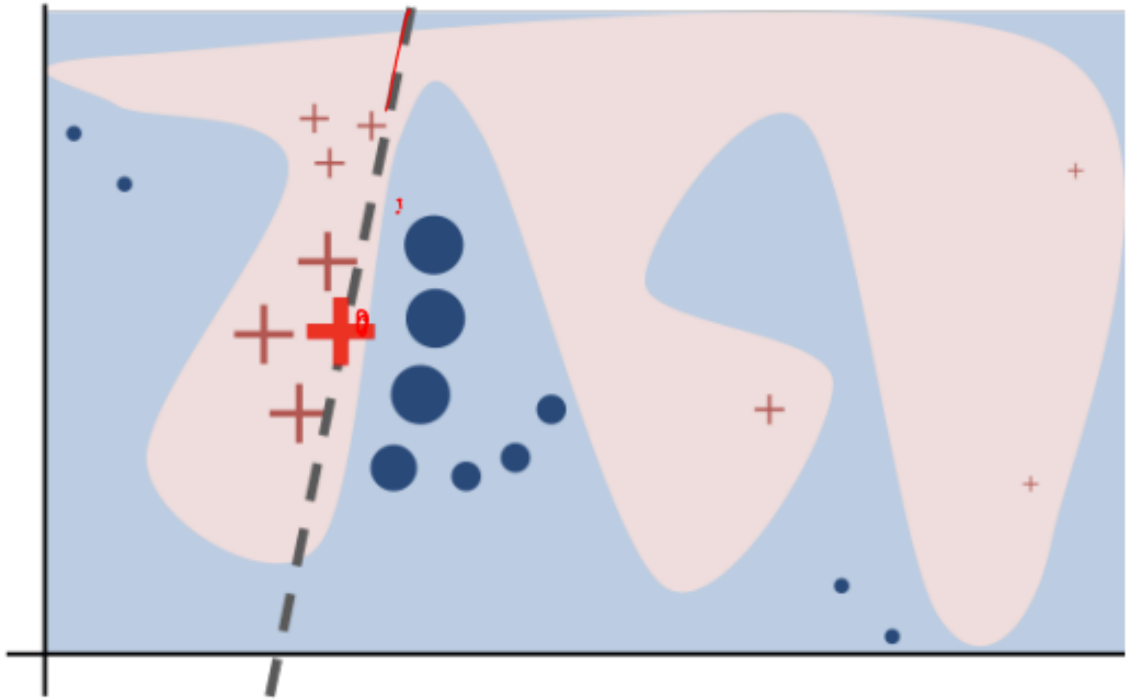
$f: \quad n+1$

$y = \beta^T x + \beta_0$

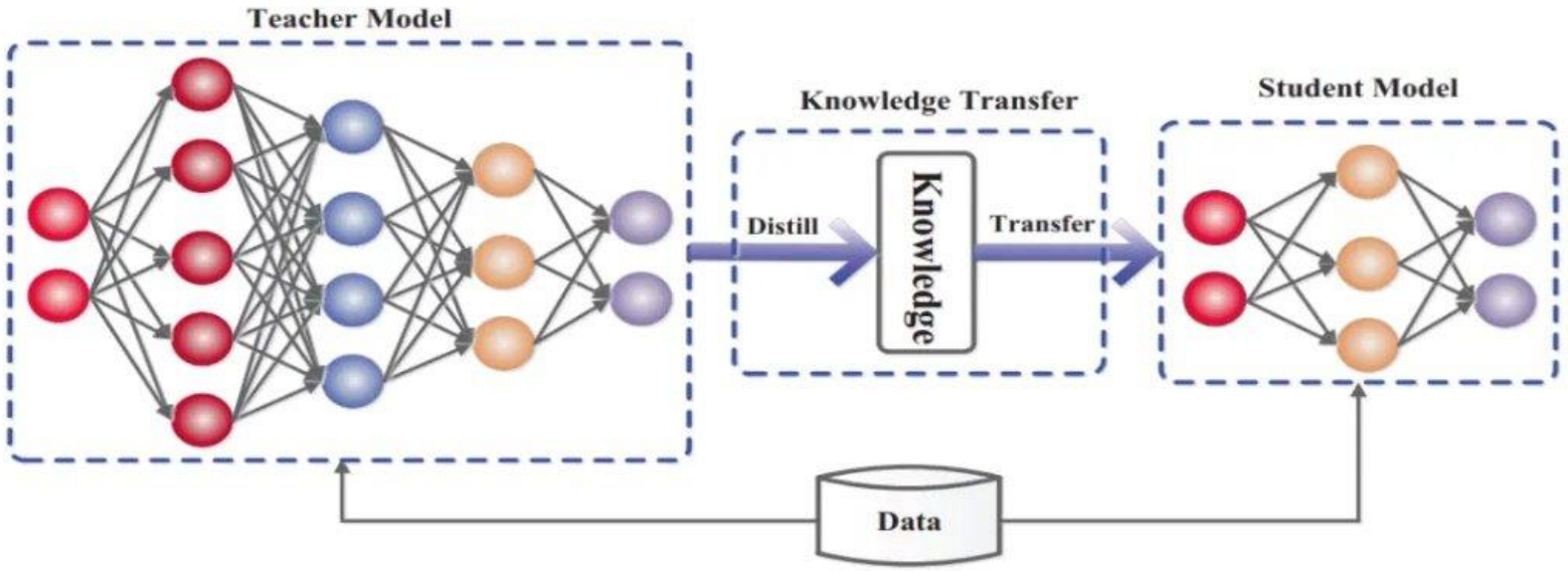
- Find tangent at the precise point

$f($

- What is the tangent at a point?



How to build global explanation?



Teacher-student training

Trustee

Goals

- Posthoc Model-Agnostic Explanation
- Determine underspecification problems in ML models

AI/ML for Network Security: The Emperor has no Clothes

https://trusteeml.github.io/

Arthur S. Jacobs
UFRGS, Brazil
asjacobs@inf.ufrgs.br

Roman Beltiukov
UCSB, USA
rbeltiukov@ucsb.edu

Walter Willinger
NIKSUN Inc., USA
wwillinger@niksun.com

Ronaldo A. Ferreira
UFMS, Brazil
raf@facom.ufms.br

Arpit Gupta
UCSB, USA
arpitgupta@ucsb.edu

Lisandro Z. Granville
UFRGS, Brazil
granville@inf.ufrgs.br

Trustee: High-level Idea

- Approximate a DL classifier with a decision tree
- Draw the important features from the approximate decision tree
- Challenge: Designing proper decision tree to minimize approximation error

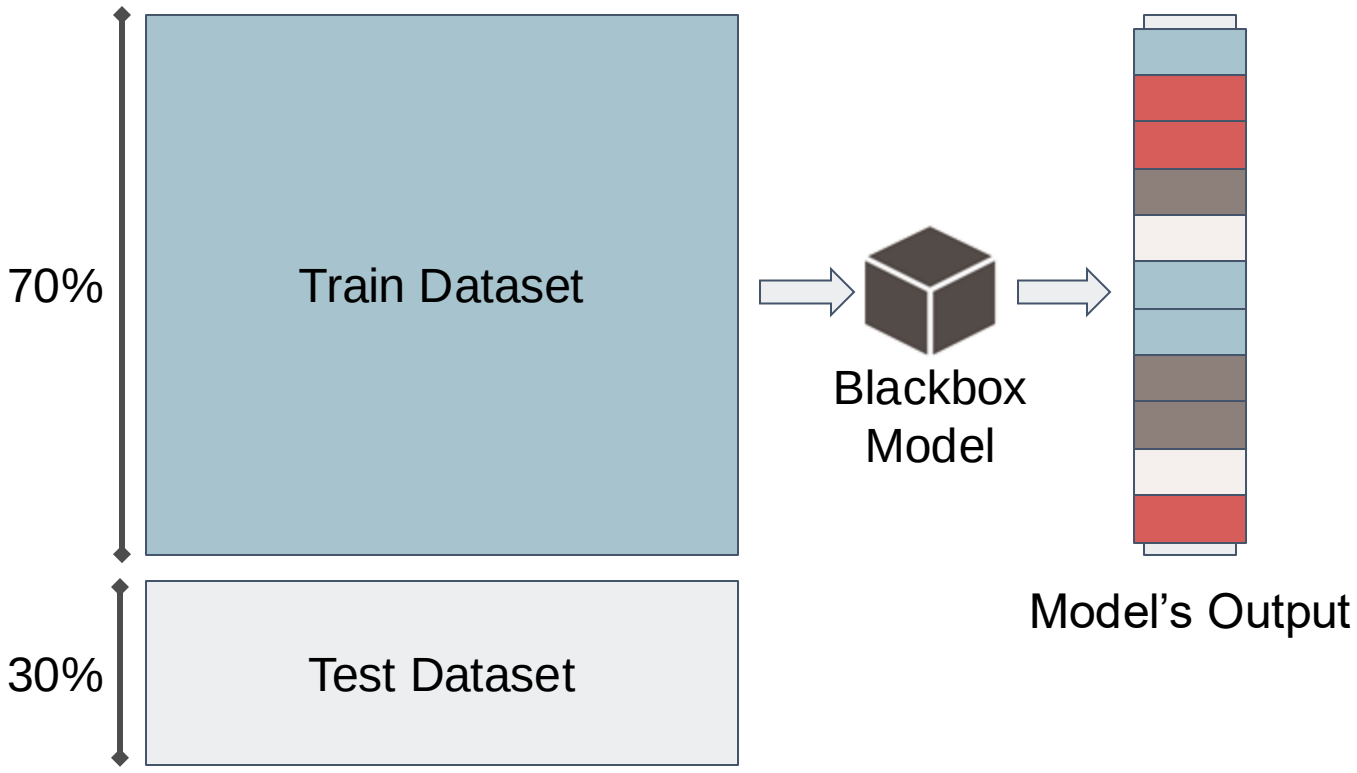
Trustee Challenges

- **High Fidelity**
  Stay faithful to blackbox model's decision-making

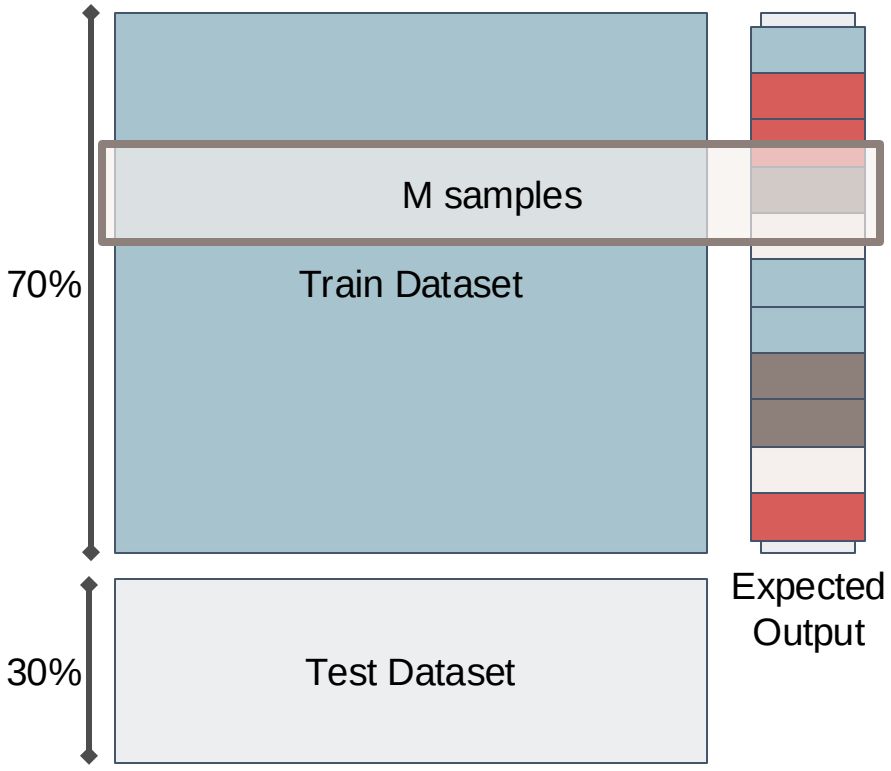
- **Low Complexity**
  Keep tree size small to simplify comprehending its decision-making

- **Stable**
  Output only one representative DT for a given model and dataset

Trustee’s Input

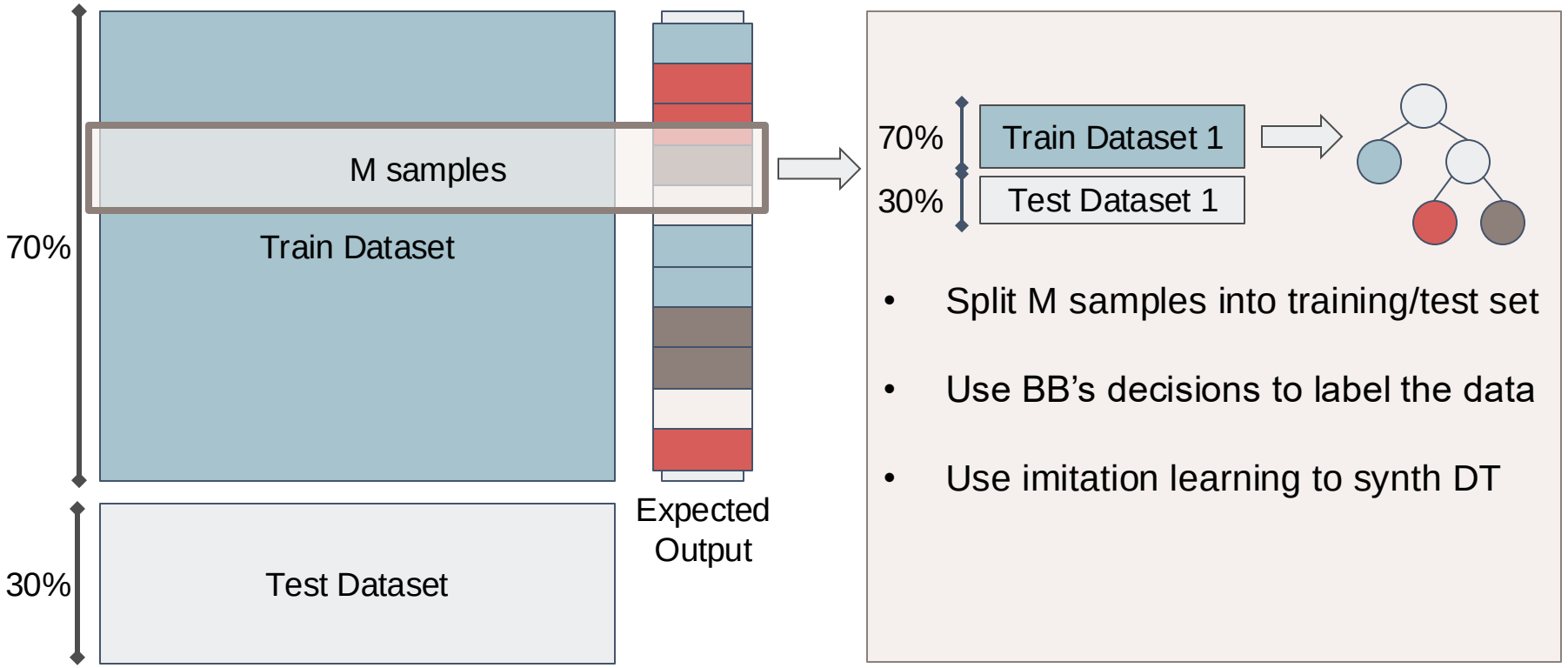


How to synthesize Decision Tree?



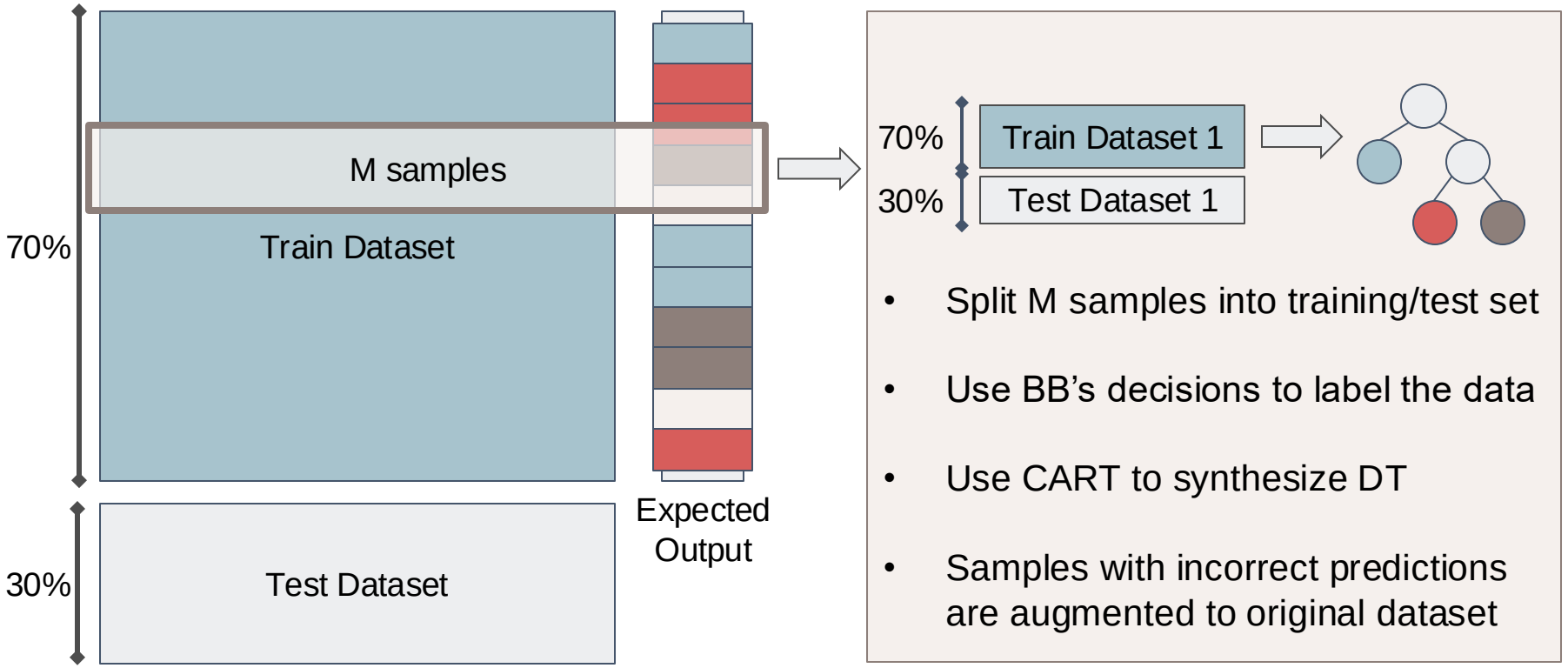
How to synthesize Decision Tree?

Imitation Learning or Model Distillation



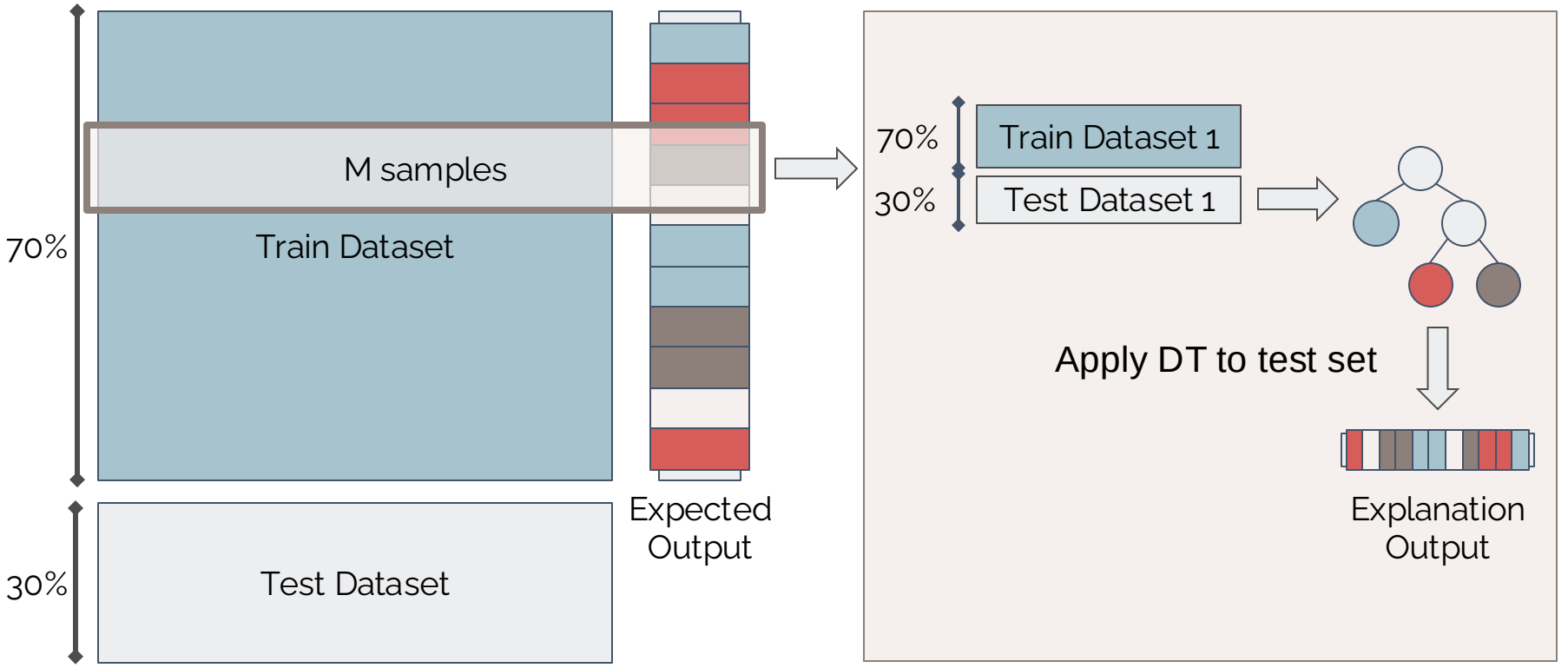
How to synthesize Decision Tree?

Imitation Learning or Model Distillation

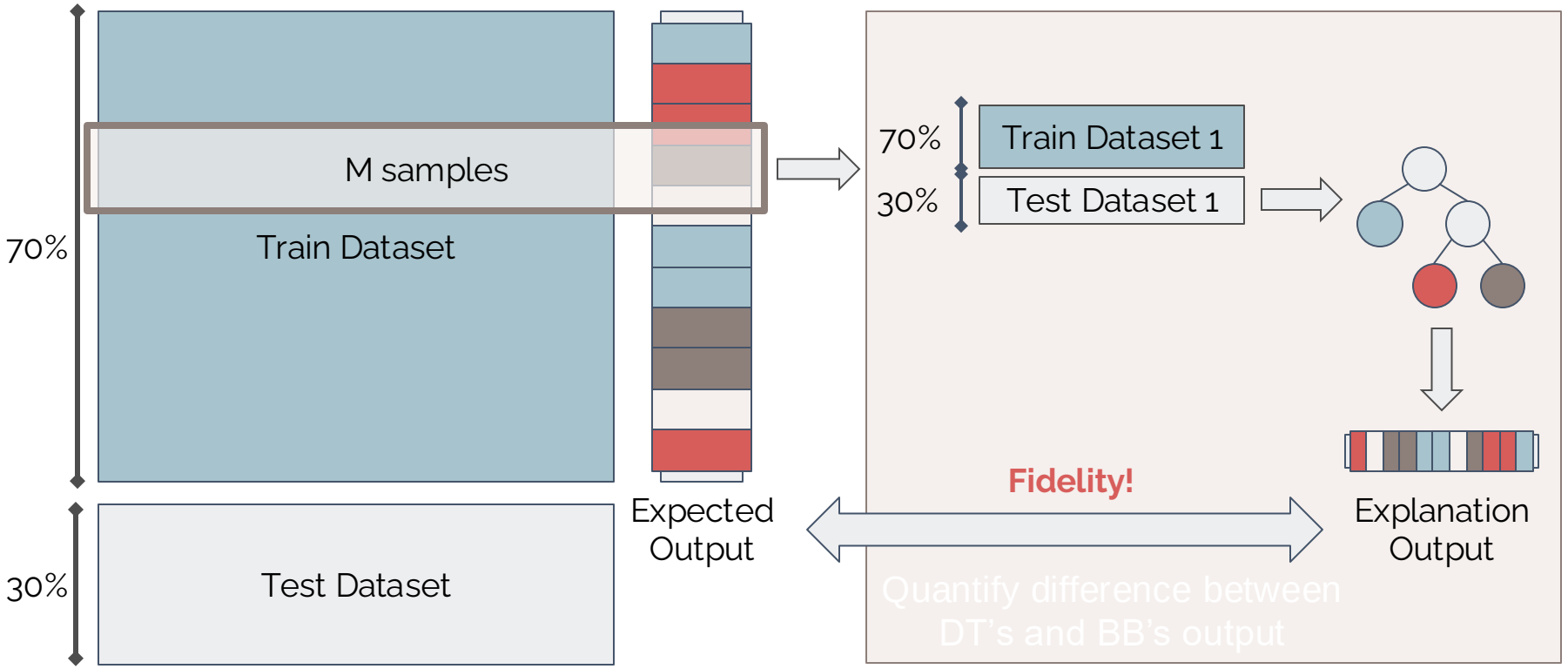


- Split M samples into training/test set
- Use BB's decisions to label the data
- Use CART to synthesize DT
- Samples with incorrect predictions are augmented to original dataset

What is Decision Tree's Fidelity?

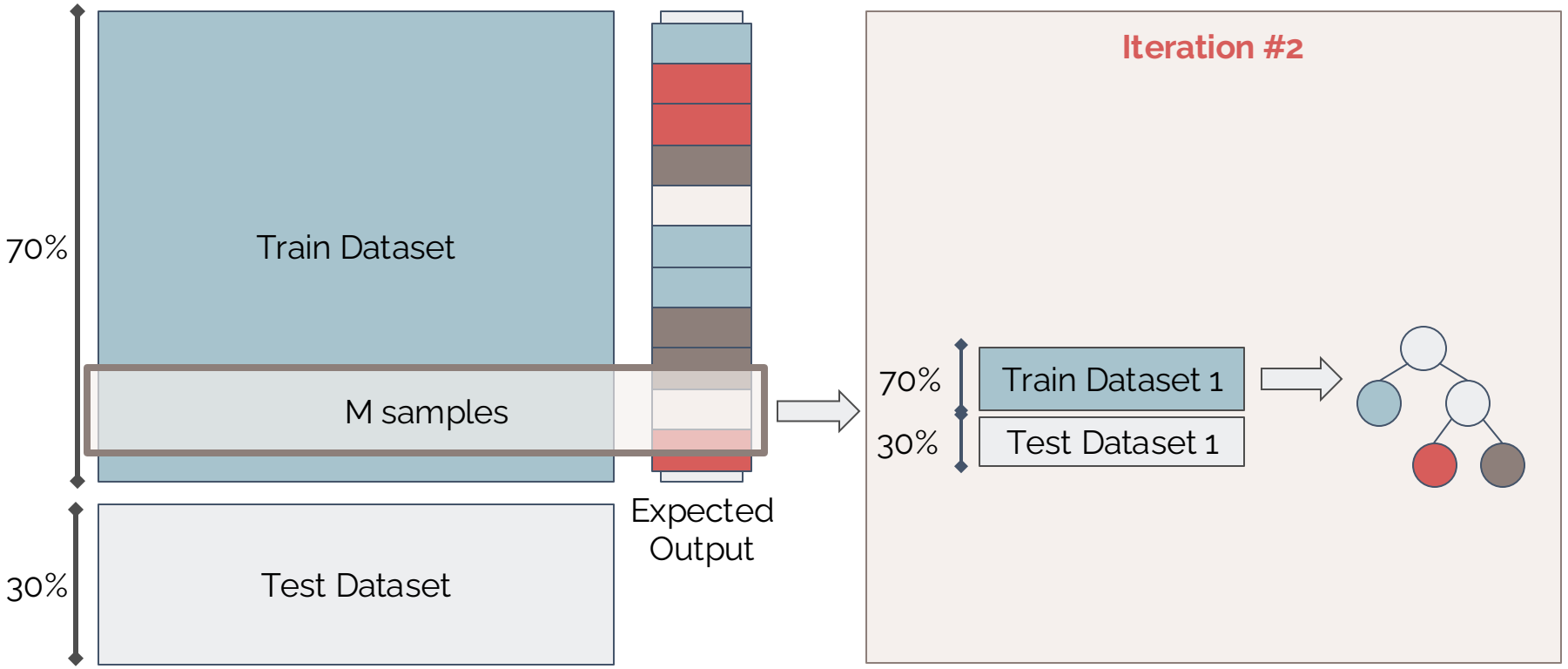


What is Decision Tree's Fidelity?

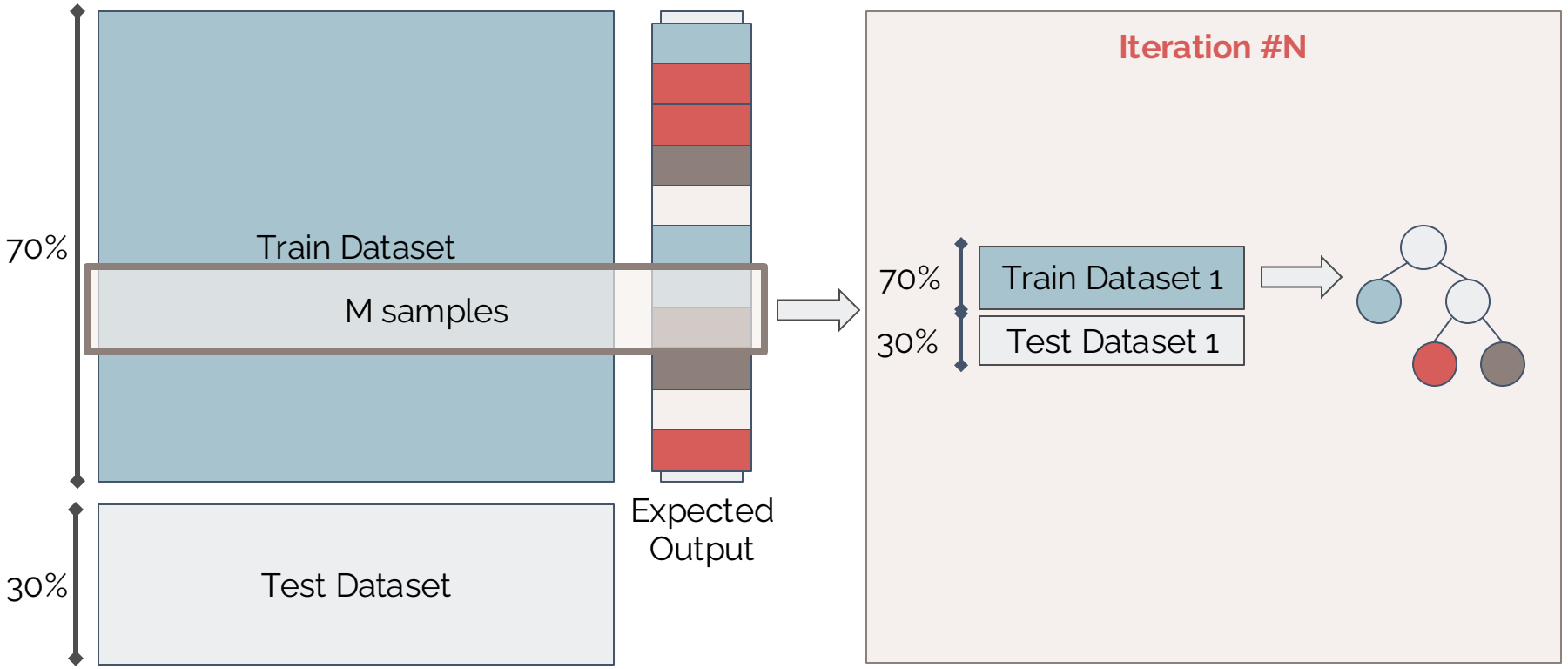


Is this the best Decision Tree?

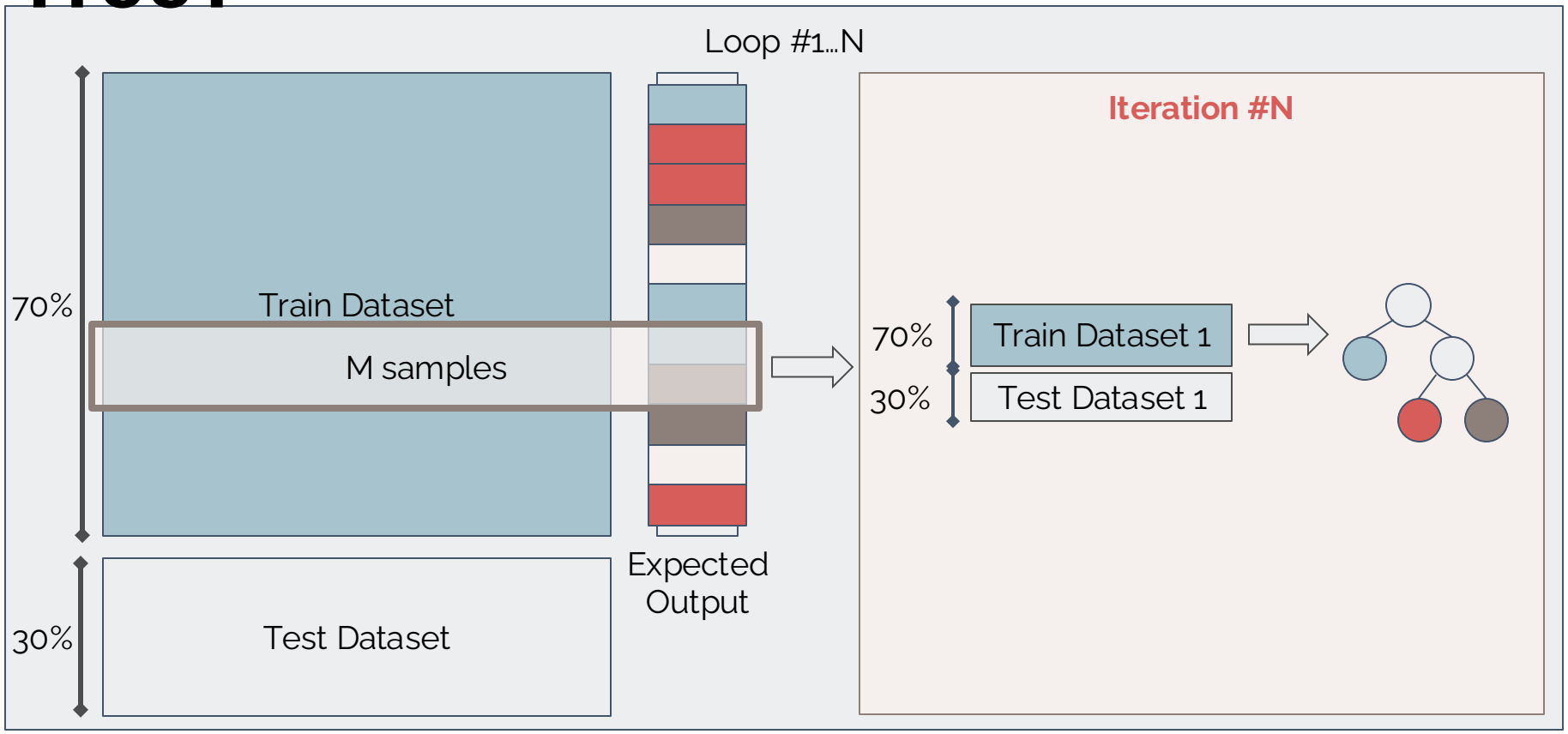
How to find Highest Fidelity Decision Tree?



How to find Highest Fidelity Decision Tree?

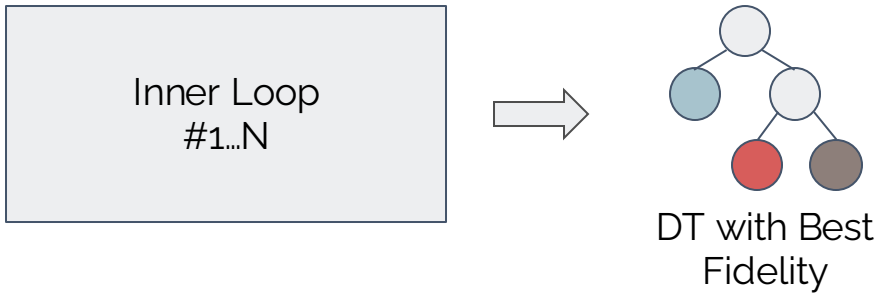


How to find Highest Fidelity Decision Tree?



Generate N Decision Trees

How to find Highest Fidelity Decision Tree?



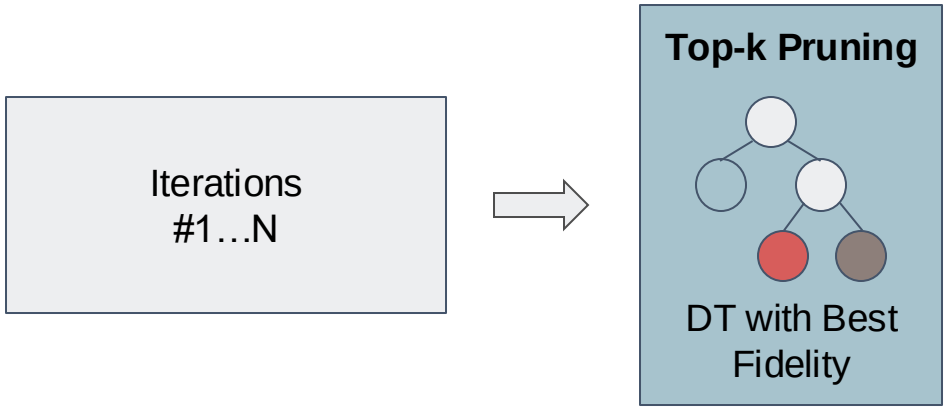
Pick one that exhibits highest fidelity

How to Reduce Decision Tree's Complexity?

- Pre-pruning techniques:
  - Maximum depth, minimum samples per leaf, minimum samples per split

- Post-pruning techniques
  - Cost-complexity pruning
  - Top-k pruning: Keep top-k branches ranked by the number of input samples a branch classifies,

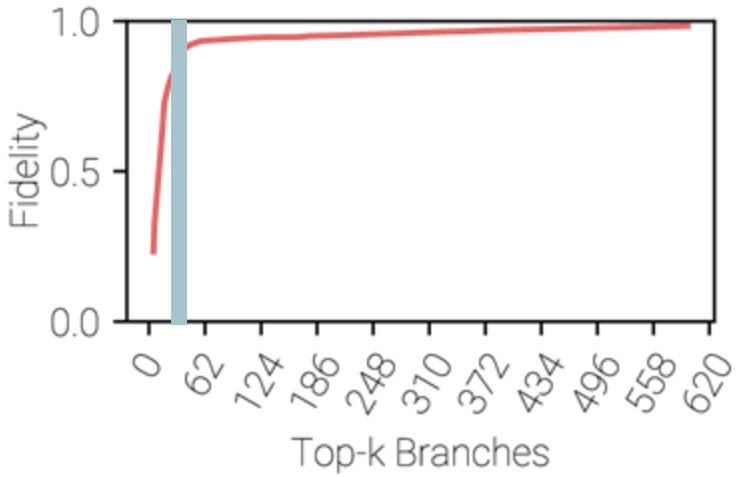
How to Reduce Decision Tree's Complexity?



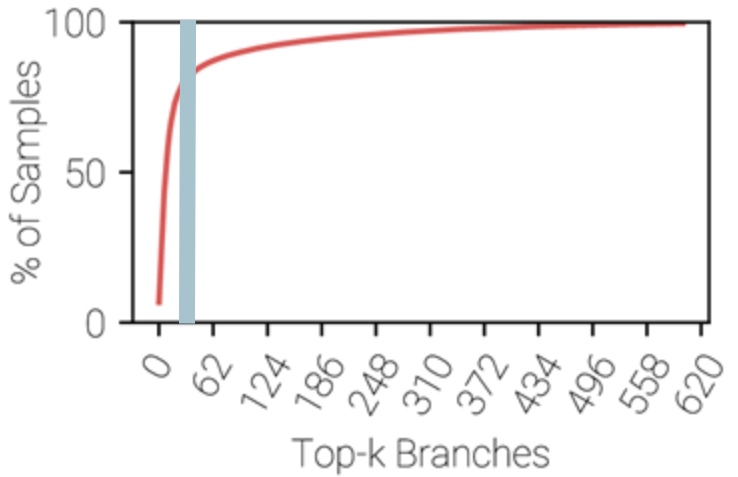
Only consider top-k branches… why?

Leverage Inherent Diminishing Returns

Fidelity

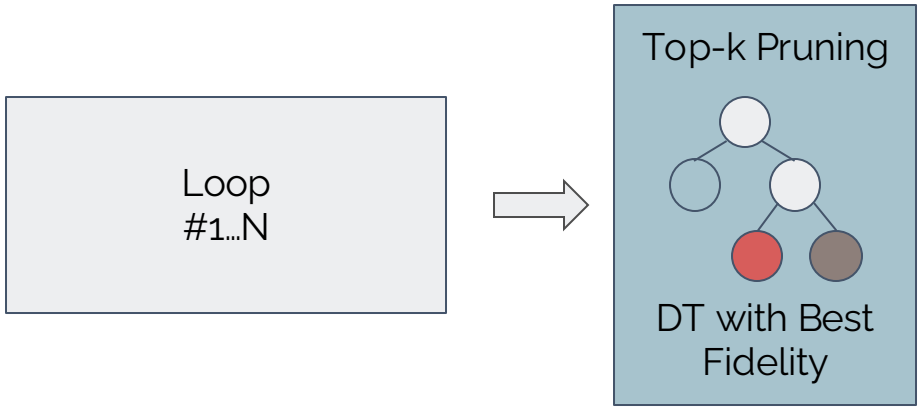


Samples



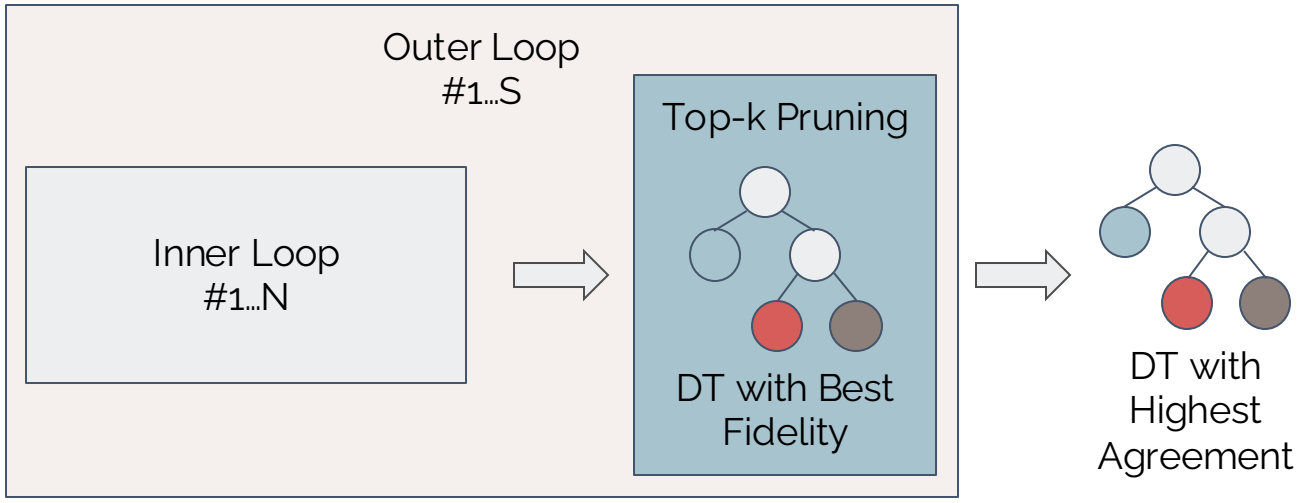
Focus on explaining **most** (not all) decisions

Are we done now?



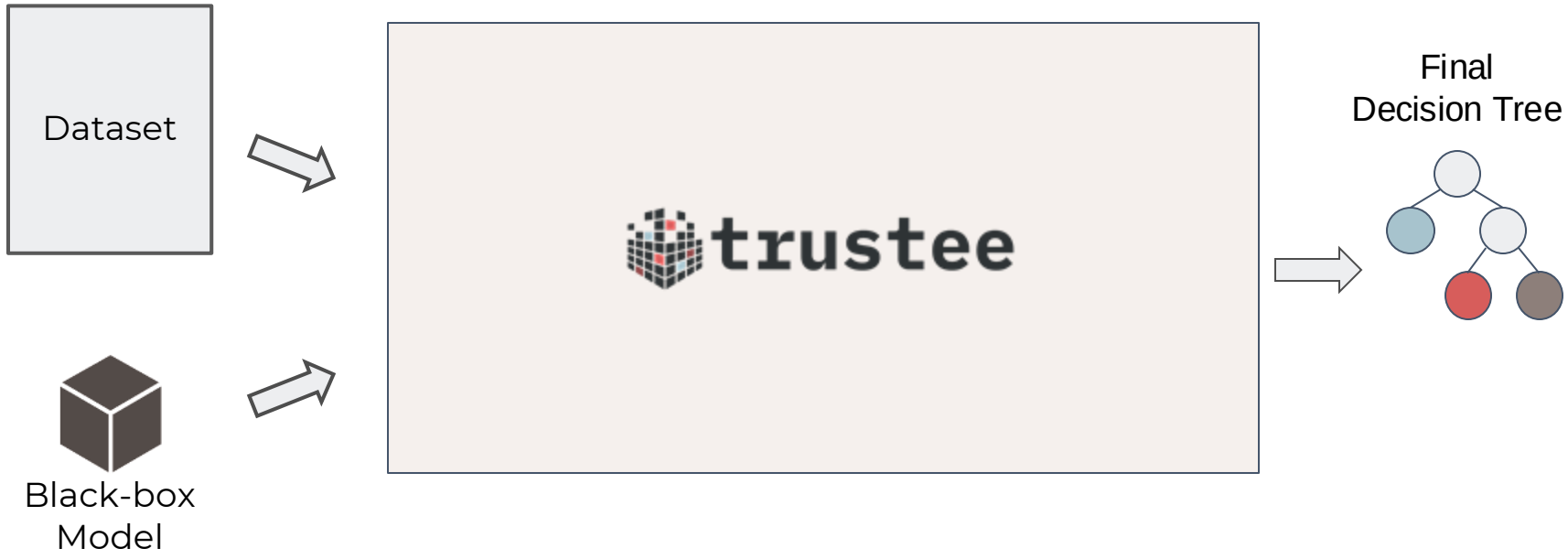
Is this stable?

How to Synthesize Stable Decision Trees?



Agreement → quantifies similarity in decision-making between pair of DTs

Final Outcome



High fidelity, low complexity, and stable decision tree

Step 2: Generate Trust Report

Classification Trust Report

Summary

Blackbox		Whitebox		Top-k Whitebox	
Model:	RandomForestClassifier	Explanation method:	Trustee	Explanation method:	Trustee
Dataset size:	947072	Model:	DecisionTreeClassifier	Model:	DecisionTreeClassifier
Train/Test Split:	70.00% / 30.00%	Iterations:	1	Iterations:	1
		Sample size:	50.00%	Sample size:	50.00%
		Decision Tree Info		Decision Tree Info	
		Size:	2437	Size:	9
		Depth:	31	Depth:	4
		Leaves:	1219	Leaves:	5
# Input features:	61	# Input features:	18 (29.51%)	Top-k:	1
# Output classes:	5	# Output classes:	5 (100.00%)	# Input features:	-
				# Output classes:	5 (100.00%)

Performance

	precision	recall	f1-score	support
0	1.000	0.912	0.954	24408
1	0.752	0.910	0.824	1872
2	0.929	0.827	0.875	10994
3	0.997	0.929	0.962	65188

Fidelity

	precision	recall	f1-score	support
0	1.000	1.000	1.000	22254
1	1.000	1.000	1.000	2265
2	0.969	0.965	0.967	9781
3	0.998	0.998	0.998	60768

Fidelity

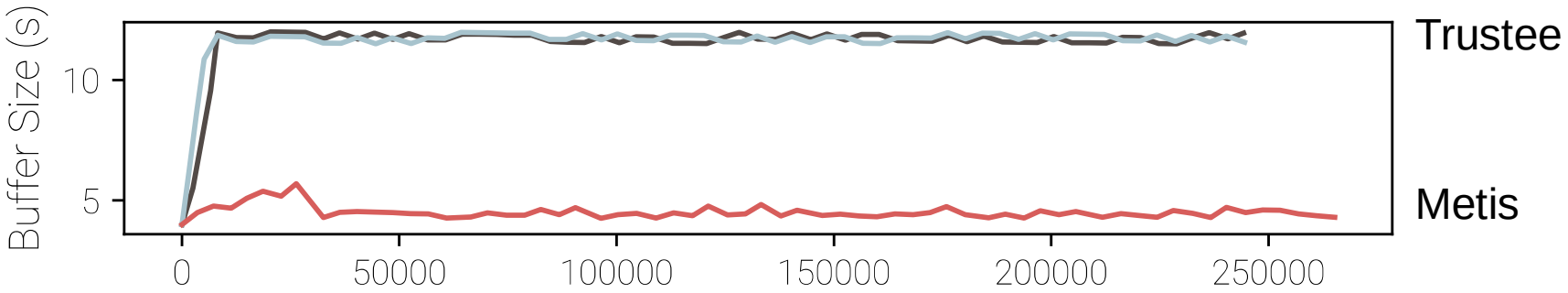
	precision	recall	f1-score	support
0	0.000	0.000	0.000	22254
1	0.000	0.000	0.000	2265
2	0.000	0.000	0.000	9781
3	0.544	0.957	0.694	60768

Helps identify **underspecification** issues

Top 1 Features

Feature	# of Nodes (%)	Data Split % - ↓
TCP Src Port	700 (57.47%)	782029 (37.11%)

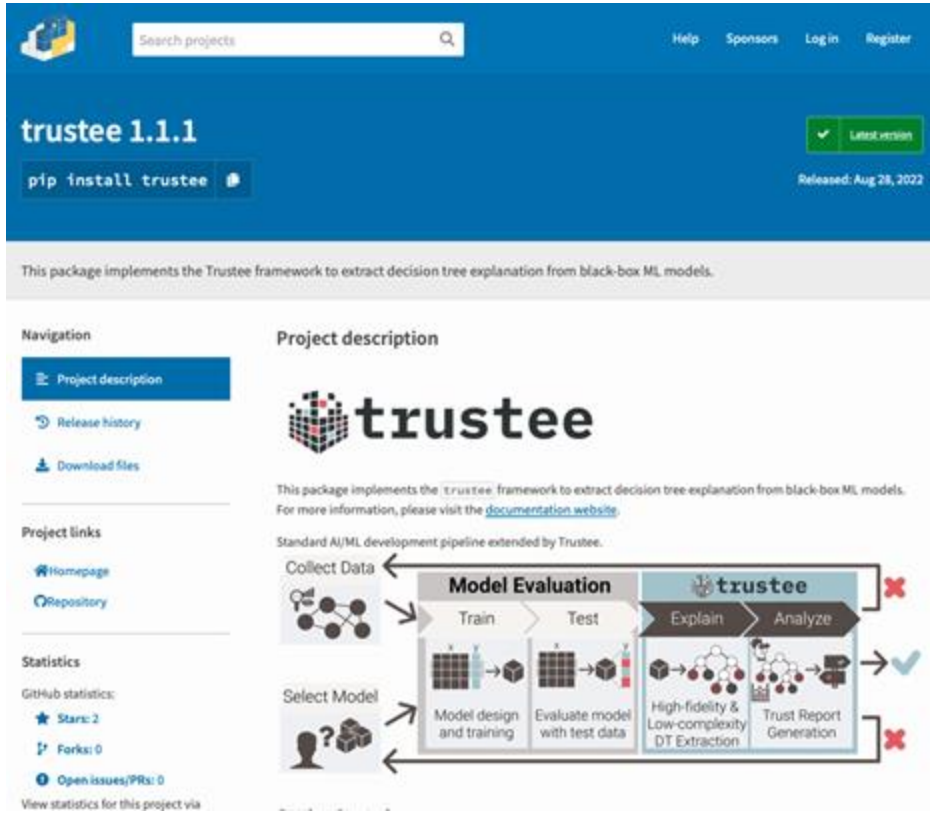
RL-Based QoE Optimization --- Pensieve

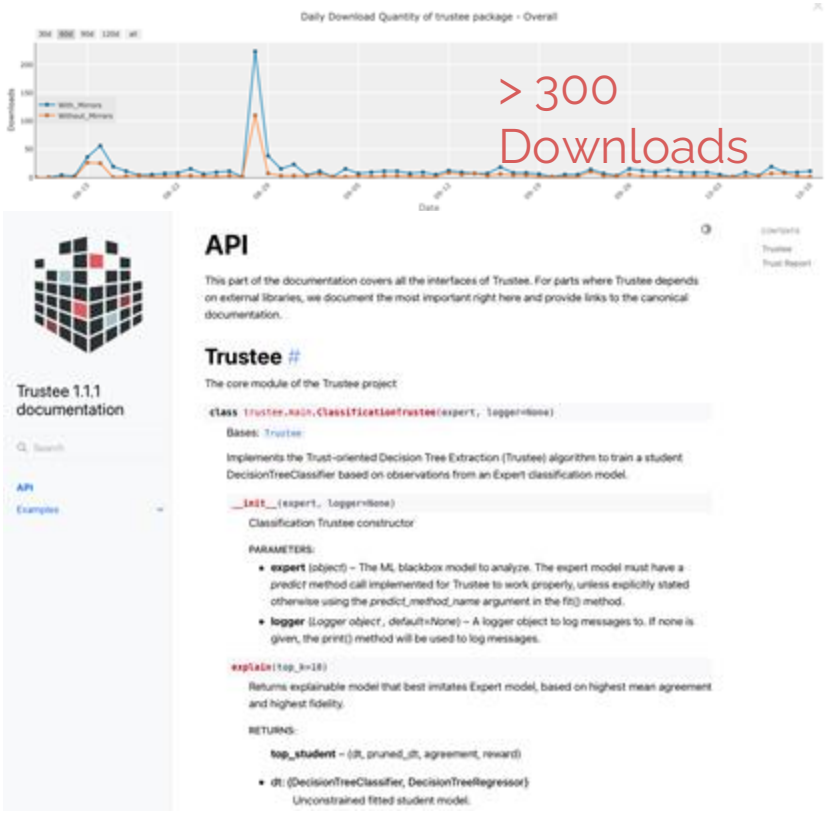


- **Decision Tree**
  - Higher fidelity (0.9) than state-of-the-art solution, Metis (0.6)
- **Observations**
  - Pensieve relies heavily on previous bit rate and buffer size for bit rate selection
  - Pensieve learns to not select 1.2 Mbps bit rate
- **Takeaway**
  - Vulnerable to OOD issues

Trustee Python Package

trusteeml.github.io





Takeaway

- Establishing **trust** requires looking beyond F1-scores

- **Trustee**
  - Synthesizes **high fidelity, low complexity, stable** DTs
  - Generates trust report to aid detecting **under specification** issues

- **Trustee for you!**

  Python package, compatible with popular ML libraries

Trustee Python package

- https://pypi.org/project/trustee/

Trustee Repository

- https://github.com/TrusteeML/trustee

Use Cases Repository

- https://github.com/TrusteeML/emperor

Special Topics: Machine Learning (ML) for Networking

COL867

Holi, 2025

Explainability

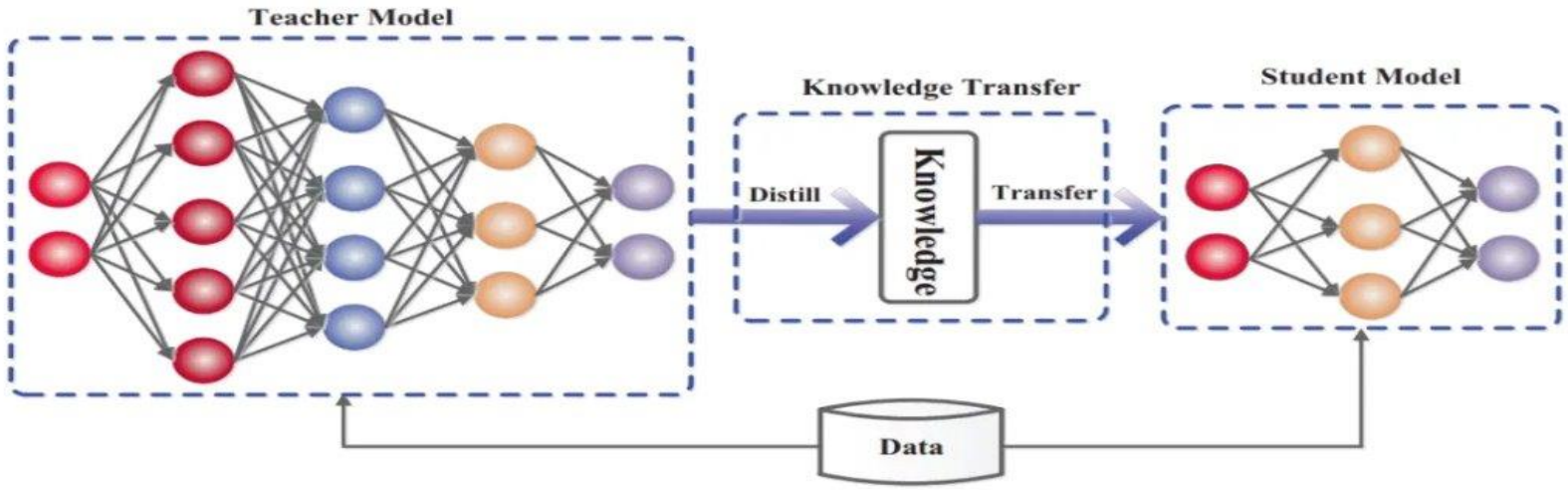
Tarun Mangla

Recap

- Model explainability is useful:
  - Diagnose and patch errors
  - Build trust
  - Learn new knowledge

- Model explainability taxonomy
  - Local: explain model output for a given instance x
  - Global: explain model output on average

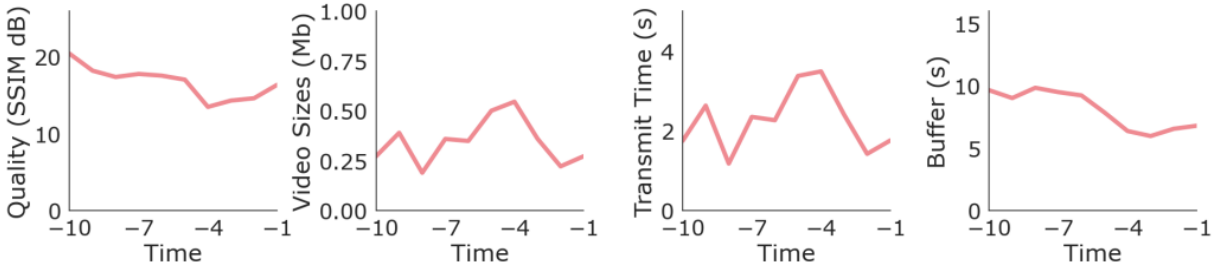
Trustee: Explainability Framework using Imitation Learning

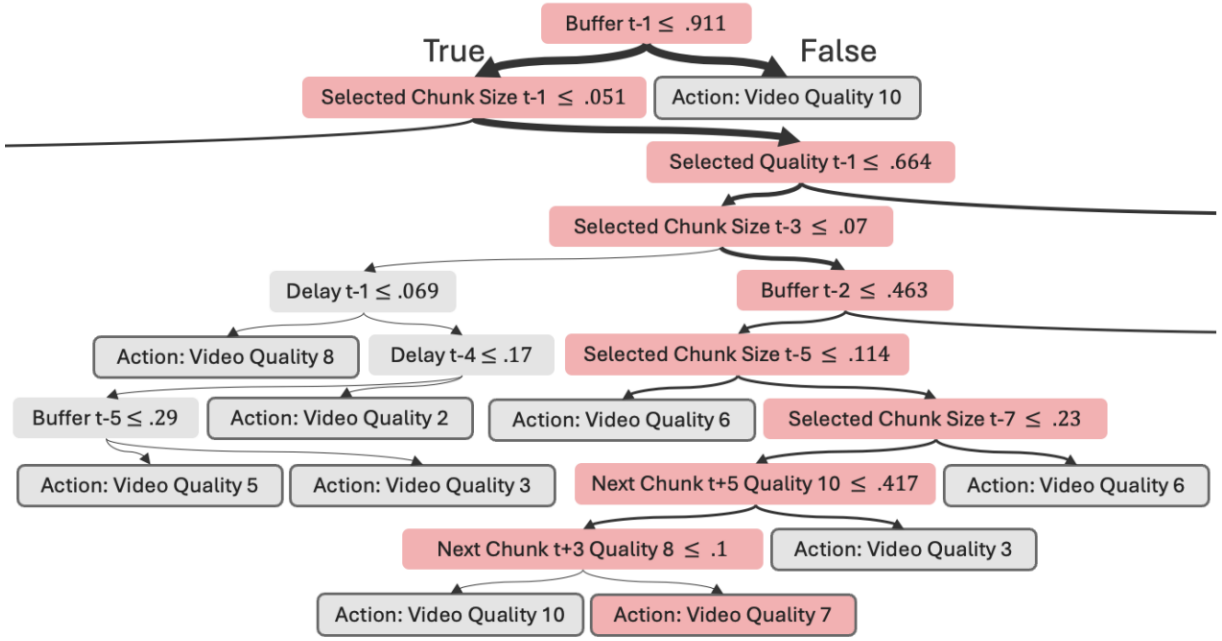


- Addressed challenges related to generating high-fidelity, low-complexity, and stable decision trees

Limitations of Trustee

- Explanation in terms of the raw features: one still needs to make sense of it
- Example, in RL-based bitrate adaptation





Provides interpretability, not explainability

Concept-based Explanability

- Provide explanation in terms of human-understandable concepts (instead of raw features)
- For instance, in case of bitrate adaptation
- **Question:** how do you build a model that provides concept-based explanations?

1. Volatile Network Throughput
2. Rapidly Depleting Buffer
3. Low Content Complexity
4. Recent Network Improvement
5. Extreme Network Degradation
6. Moderate Network Throughput
7. Anticipation of Network Congestion
8. Content requiring High Quality
9. Stable Buffer
10. Nearly Full Buffer
11. Startup of video
12. High Content Complexity
13. Network Volatility needing Switching
14. Avoiding Large Quality Fluctuations
15. Switch to higher quality after startup
16. High Network Throughput

Setup

Open Questions

- How to generate concepts for a learning task?

- How to generate instances with raw features mapped to concepts?

Generating Concepts

- Use LLMs to generate base concepts
  - E.g., generate base concepts for traffic classification problem using network data

- Proposed Improvement:
  - First create a meaningful context by asking an LLM to summarize a survey paper

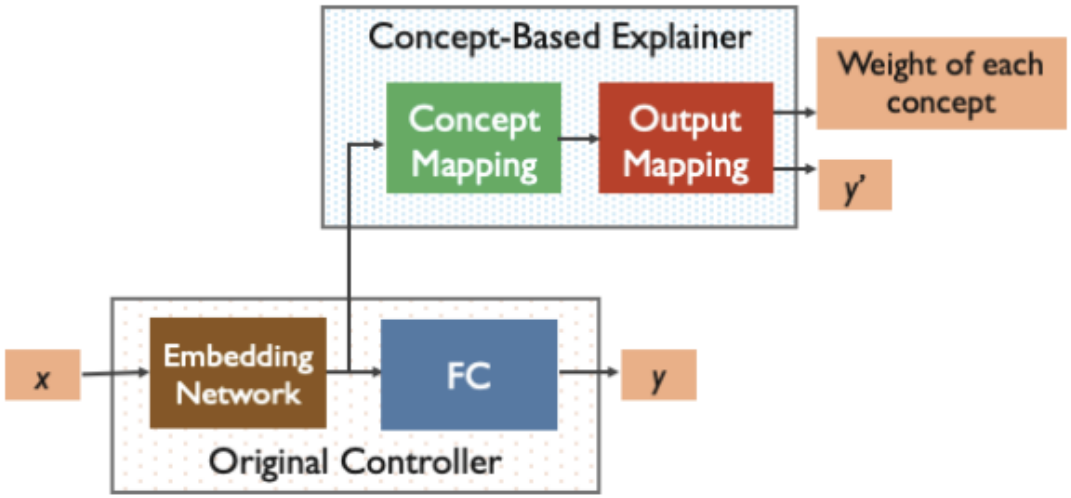
Generating Features to Concept Mapping

- Use LLMs again

Explaining the Model

- Training concept mapping
  - Multi-label classification problem

- Training output mapping

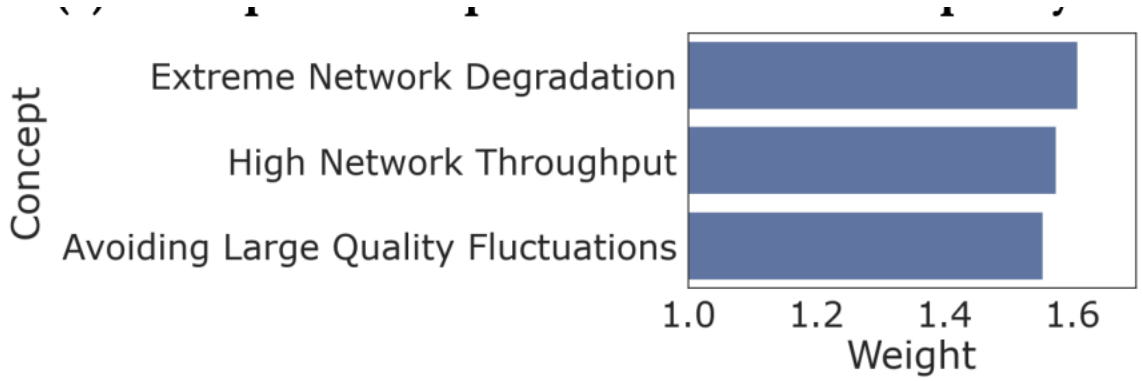


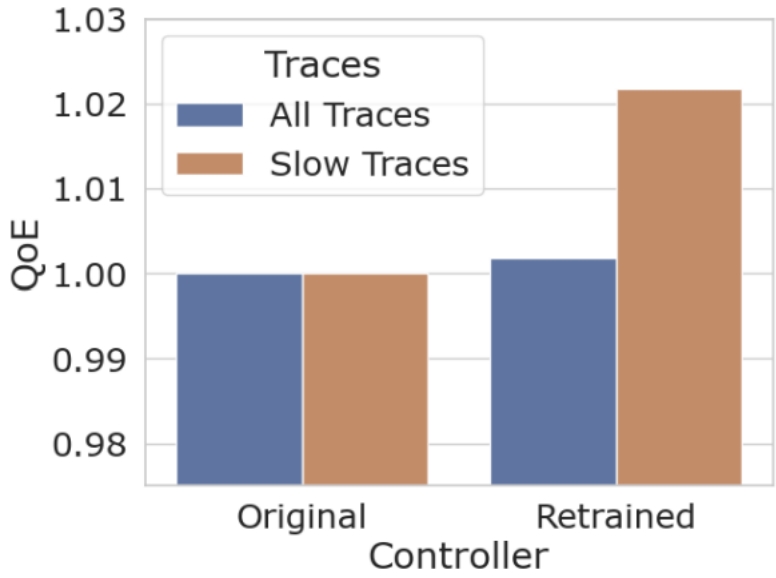
Inference API

- Factual single state: generates an explanation based on the top concepts in the given state
- Counterfactual states: how state A differs from state B by comparing top concepts in both states
- Counterfactual actions: why action A was chosen over action B

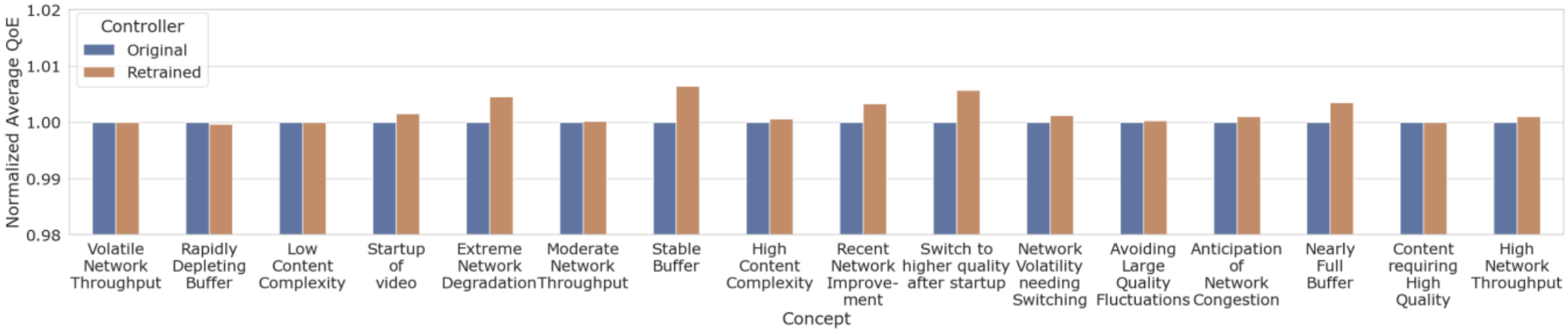
Evaluation

- Improves controller by retraining for specific concepts





Improving Controller by Retraining in Specific Scnarios



Possible Use Cases

- Interactive intent-driven controller design
- Steering dataset selection during design
- Architectural modifications during design
- Concept coverage metrics for testing
- Online safety assurance in deployment

Limitations?

Logistics

- No class on April 11 (coming Friday)
- Assignment 2 is out
- Reach out if you want to discuss project
- **Next class:** ML and formal verification

Trustee Hands-on

https://trusteeml.github.io/auto_examples/index.html

So Far

Case studies around robustness in networking

- **How to train a robust network ML?**
- **How to handle drift?**
- **How do you explain ML models?**
- Systems to benchmark ML models.