Optimizing Continuous integration using Artificial Intelligence

Maher Dissem

maher.dissem@gmail.com

May 3rd, 2022

Overview

- Introduction
  - Context
  - Problem Statement
  - Objective
- Literature Review
  - Dataset
  - Machine Learning Approaches
  - Genetic Algorithms Approaches
- The Project So Far
- Approaching the Problem with Reinforcement Learning
- Conclusion

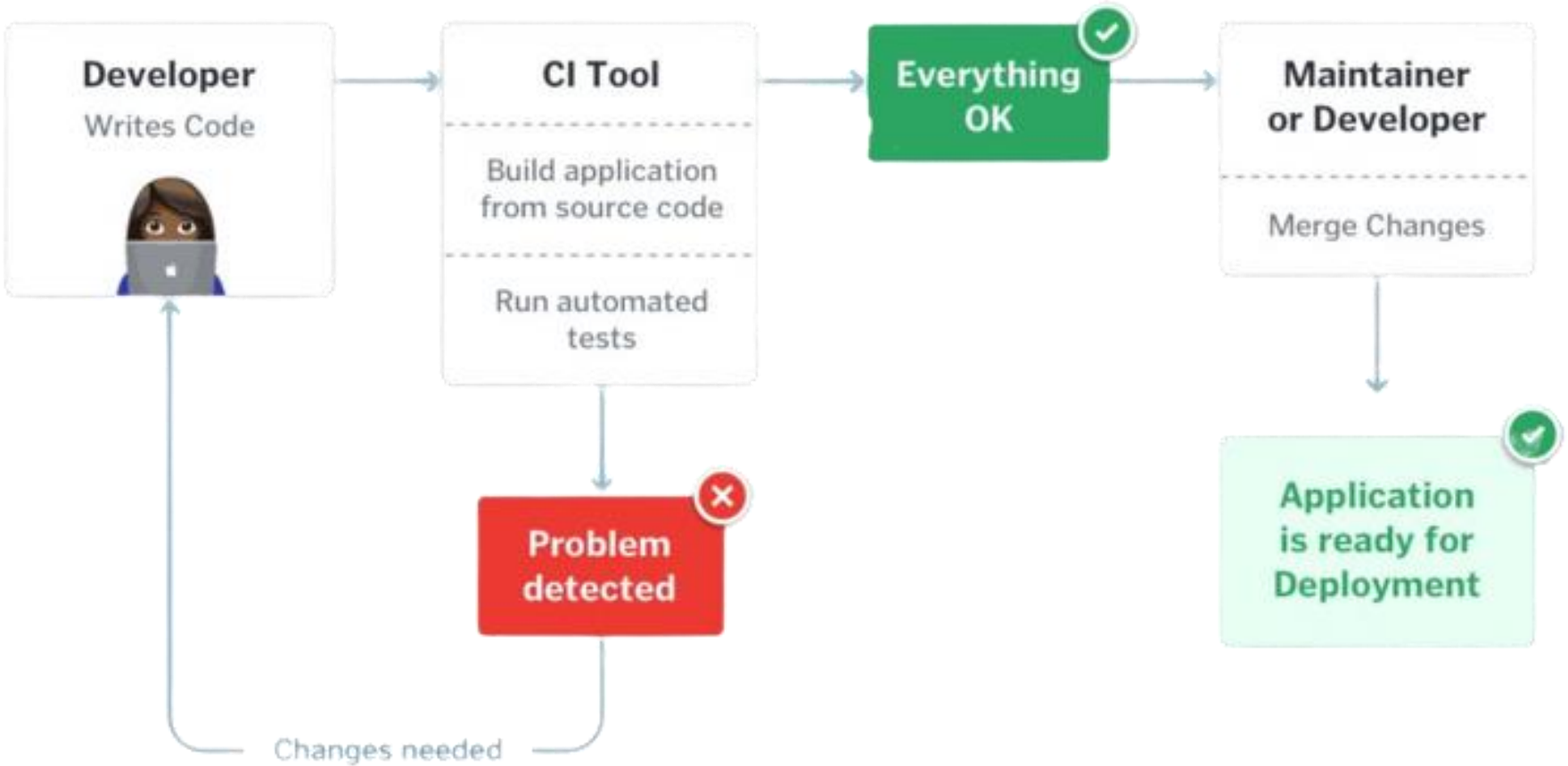
Introduction

Continuous Integration

Continuous integration is a modern, widely used practice.

It urges developers to build and test their software whenever a new commit is submitted.

General Workflow



Benefits

- Faster failure detection
- Faster release cycles
- Lower risk of delivering defective changes
- Improved code quality

Problem

Constant use of computational resources to run and test changes.

Google estimates the cost of running its CI system in **millions of dollars**, and Mozilla estimates theirs as **$201,000/month.**

For smaller-budget companies that have not yet adopted CI, its high cost can pose a strong barrier.

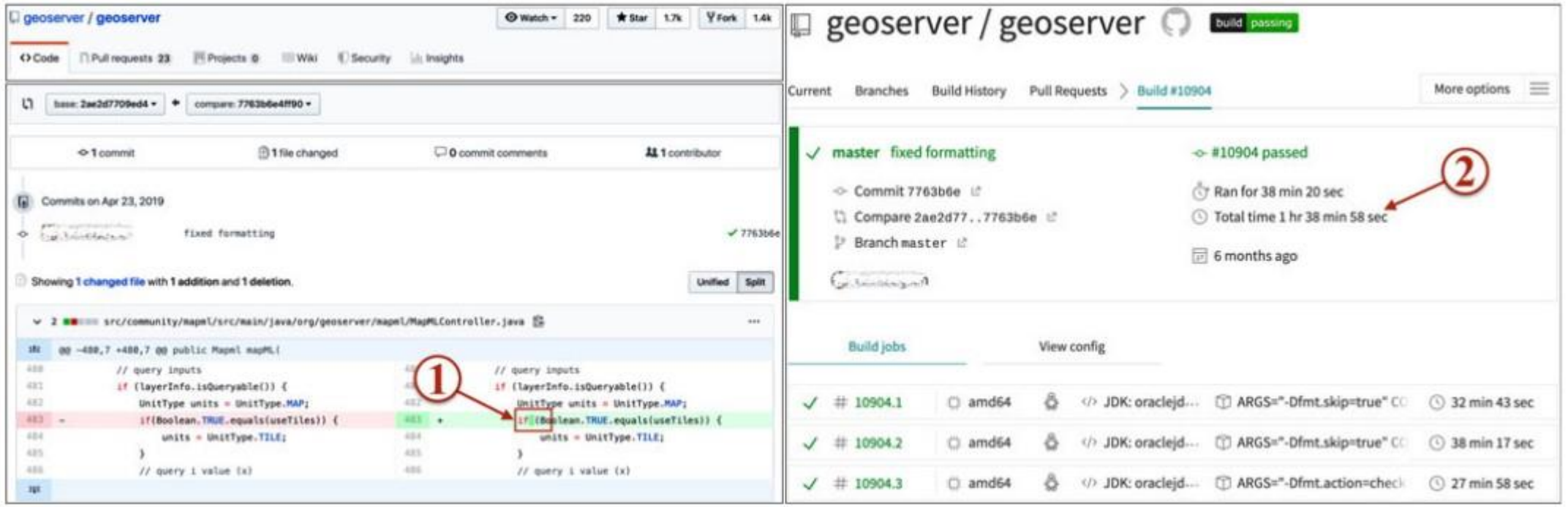
Source: A Cost-efficient Approach to Building in continuous integration, Xianhao Jin and Francisco Servant

Optimization Strategies

- Test selection:
  Running only subset of tests that are more relevant to the changes made.
- CI skip:
  Skipping the execution of unnecessary tests.

Optimization Strategies

Some commits unnecessarily trigger the CI process.



Optimization Strategies

The Manually Extracted Reasons for CI Skipped Commits.

Reason	Description	Number (%)	Information Source
Non-Source code files	Developers add or modify non source code file (e.g,. documentation files)	943 (52.01%)	Repository
Version preparation	Developers change the version of the project	274 (15.11%)	Repository
Source code comment	Adding , removing or editing source code comments	109 (6.01%)	Repository
Meta files	Developers modify meta files in the projects (e.g,. git ignore file)	68 (3.75%)	Repository
Formatting source code	Formatting source code without changing the semantic of the code	21 (1.16%)	Repository
Source code	Change that is made to source code of the project, but developers skip the build commit.	191 (10.54%)	Developer
Build change	Developer made change to build system they use.	112 (6.18%)	Developer
Tests	Change that is related to test cases of the project.	18 (1.00%)	Developer
Other		190 (10.48%)	Various

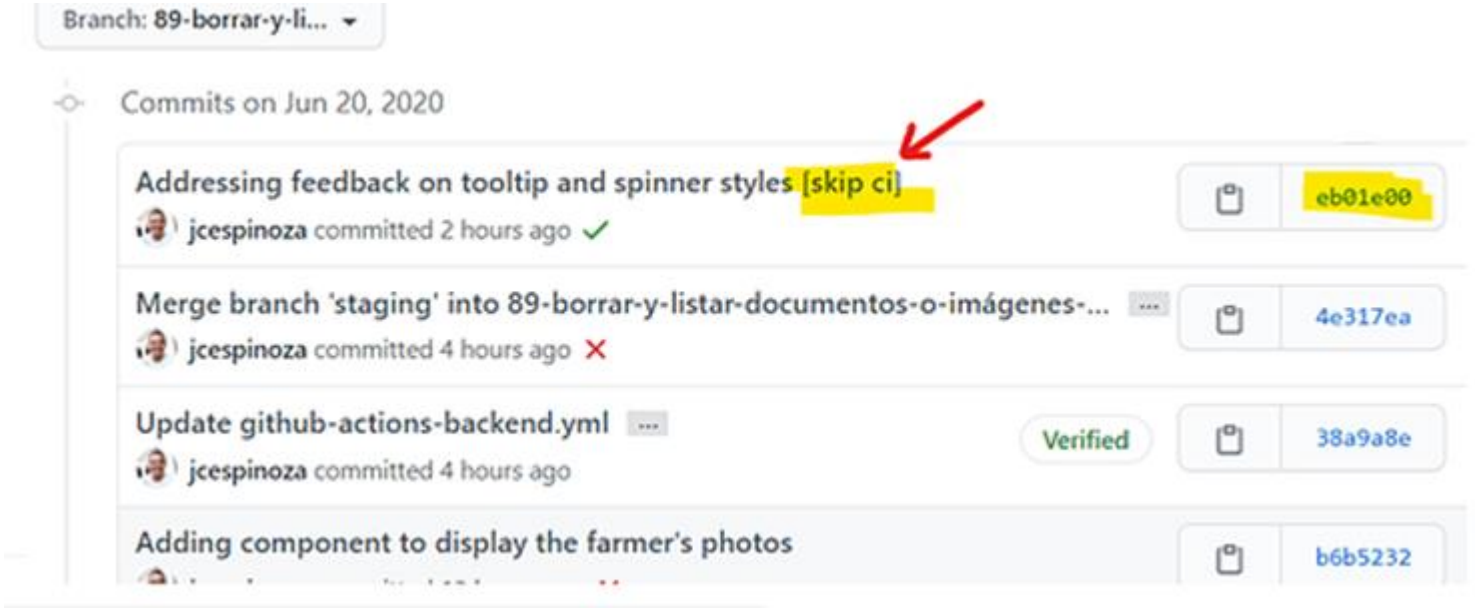
Data from an analysis of 58 open source Java projects show that 80% of commits can be skipped.

Which Commits Can Be CI Skipped? Rabe Abdalkareem et al

Goals

Commits that won't cause a build failure or change the build result vainly use computation resources.

We aim to reduce the number of executed builds by skipping unnecessary ones.



Literature Review

The Travis Torrent dataset

Each row represents a build job executed on Travis CI.

It synthesizes information from three different sources:

- The project's git repository
- Data extracted from GitHub through GHTorrent
- Data from Travis CI's API and an analysis of the build log

Dataset Features

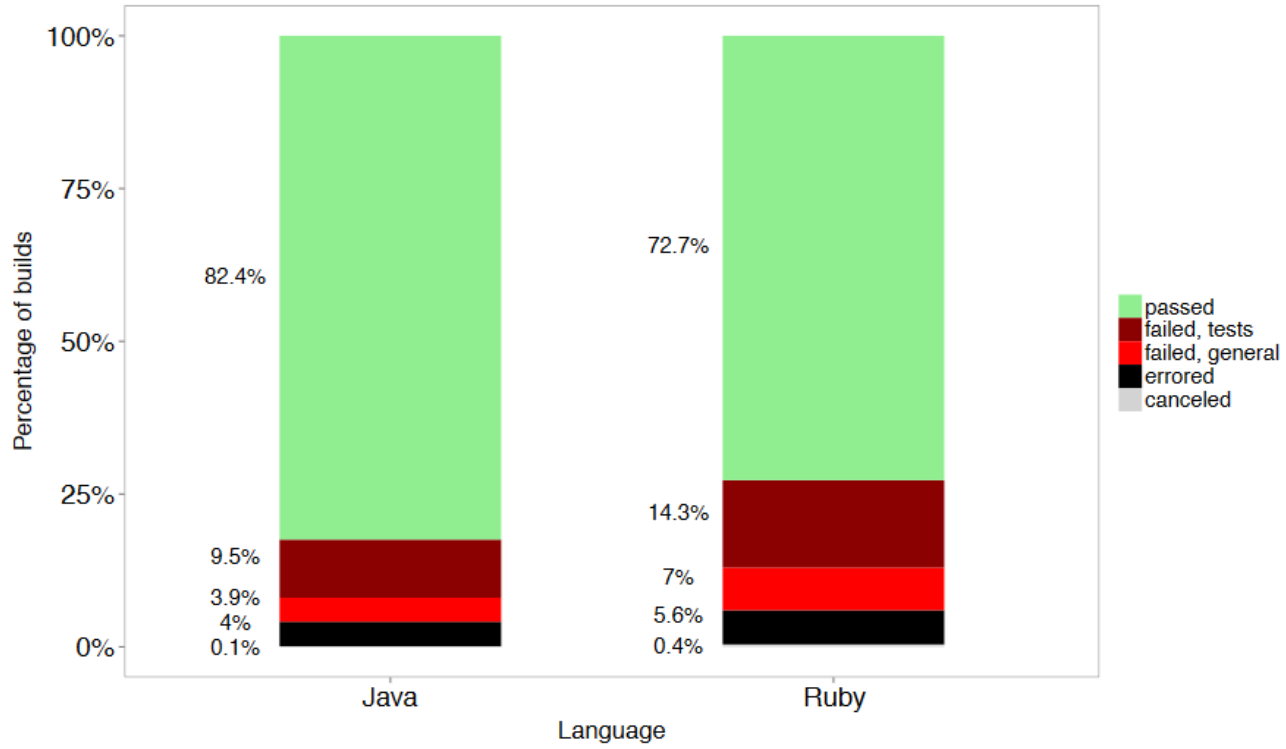
- 66 total features
- Extracted features fall under these categories:
  - Diffusion
  - Size
  - Purpose
  - History

Target is 'build status' (passed/failed)

→ a binary classification problem

Dataset

- Data imbalance



Machine Learning Approaches

- Several research projects make use of Machine Learning to tackle this problem.
  - Within project validation → existing project.
  - Cross project validation → new project with no history.
- Use of common classification metrics
- RF usually yields the best results.

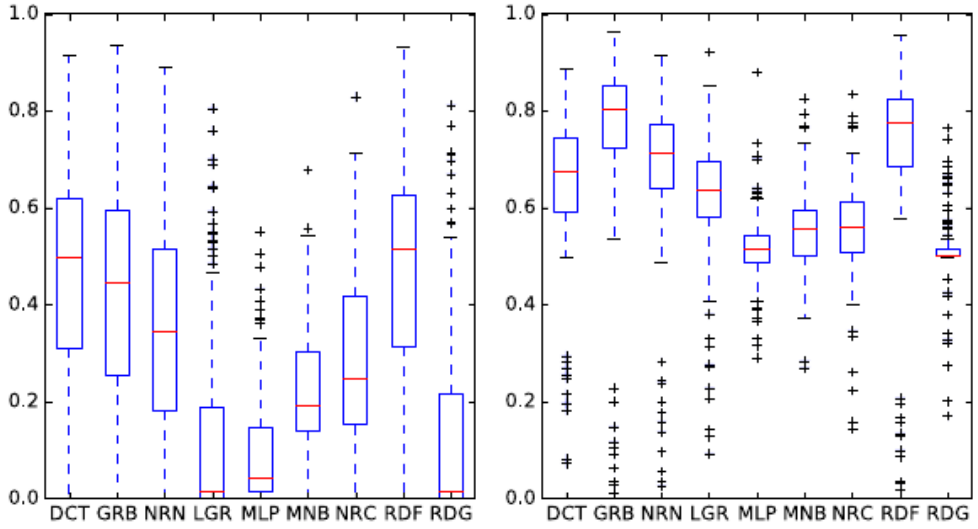
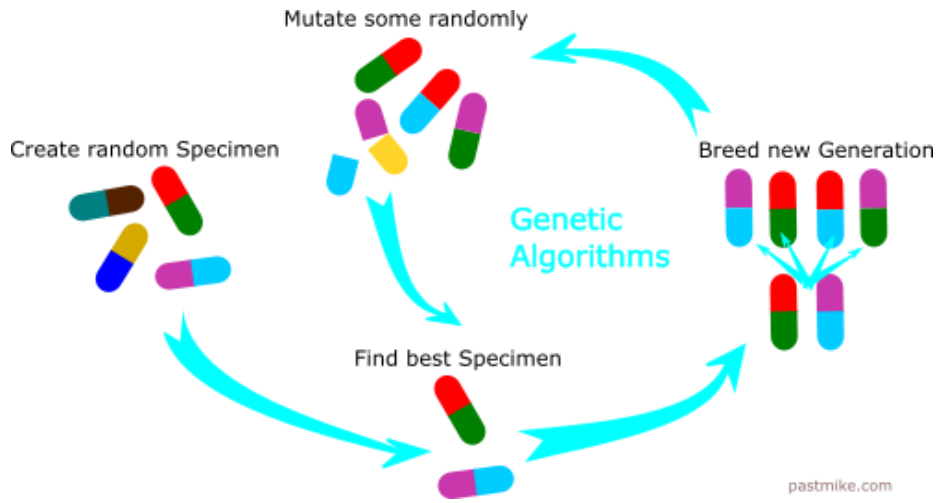


Fig. 2. F1-Score (left) and AUC (right) of Cross-validation Predictions

Genetic Algorithms



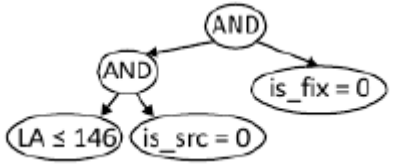
To adapt a genetic algorithm, we need to define the

- Candidate solutions
- Fitness function
- Crossover operation
- Mutation operation

Genetic Algorithms Approaches

- Finding the optimal detection rule using binary tree representations of IF -THEN rules.

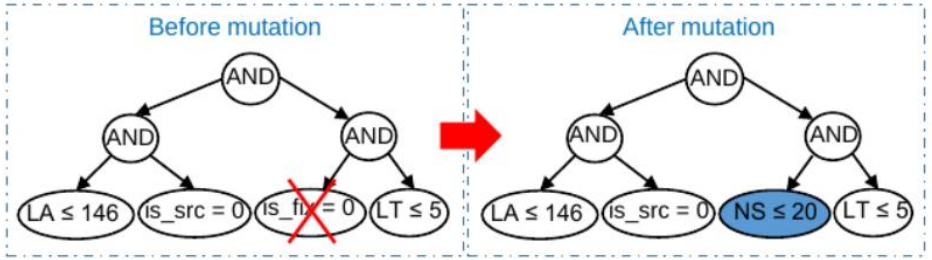
IF LA ≤146 AND IS_SRC =0 AND IS_FIX =0
THEN Skip commit.



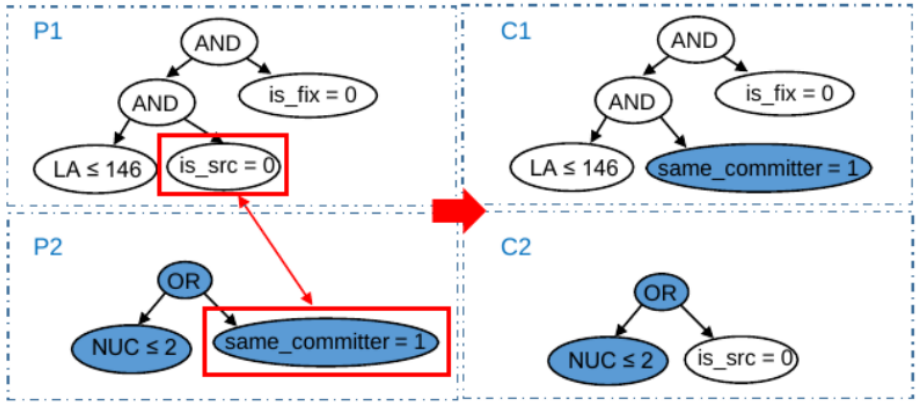
A candidate solution

- Motivation
  - Efficiency of manually defined rule based classification.
  - Promising results of GA in unbalanced classification.

Genetic Algorithms Approaches



Mutation



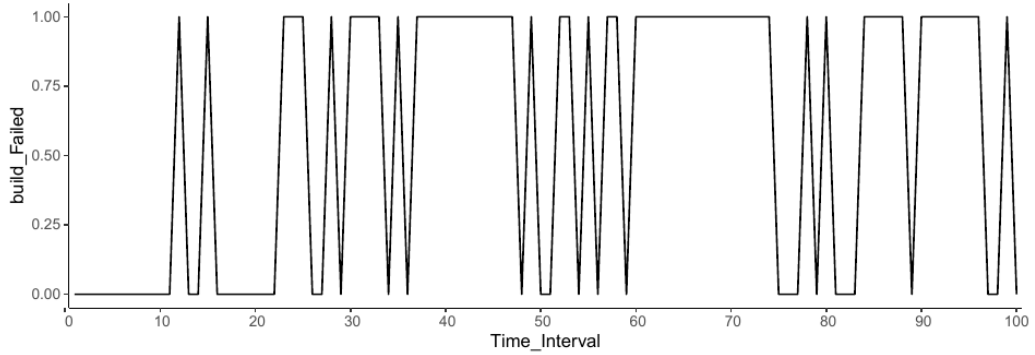
Crossover

Fitness function: $min_{i=1}^{n}\sqrt{(1-TPR_i)^2+{FPR_i}^2}$

- 92% AUC for within-project validation
- 84% AUC for cross-project validation

Genetic Algorithms Approaches

- Failing builds prediction using fail history



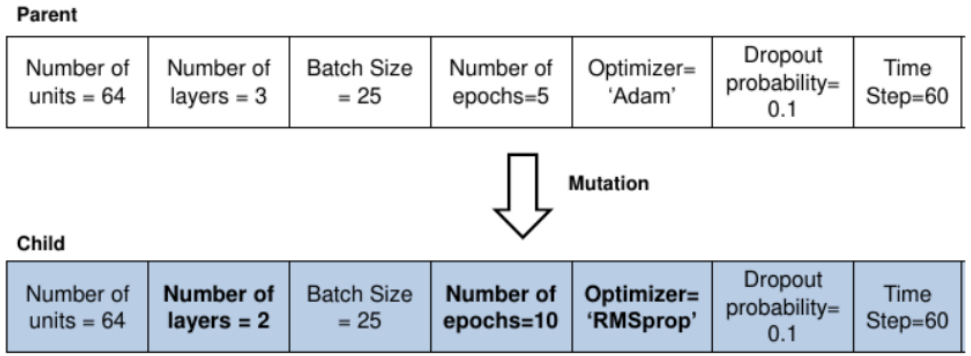
- Hyper-parameters optimization of LSTM using GA

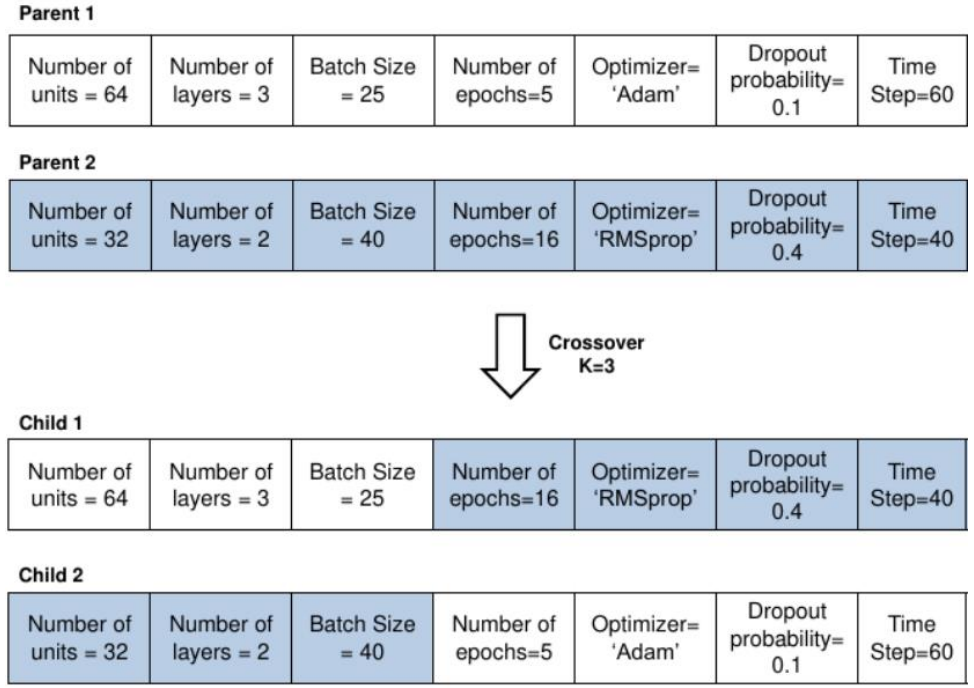
Number of units = 64	Number of layers = 3	Batch Size =25	Number of epochs=5	Optimizer= 'Adam'	Dropout probability= 0.1	Time Step=60

A candidate solution

Genetic Algorithms Approaches

- Fitness: validation loss





The Project So Far

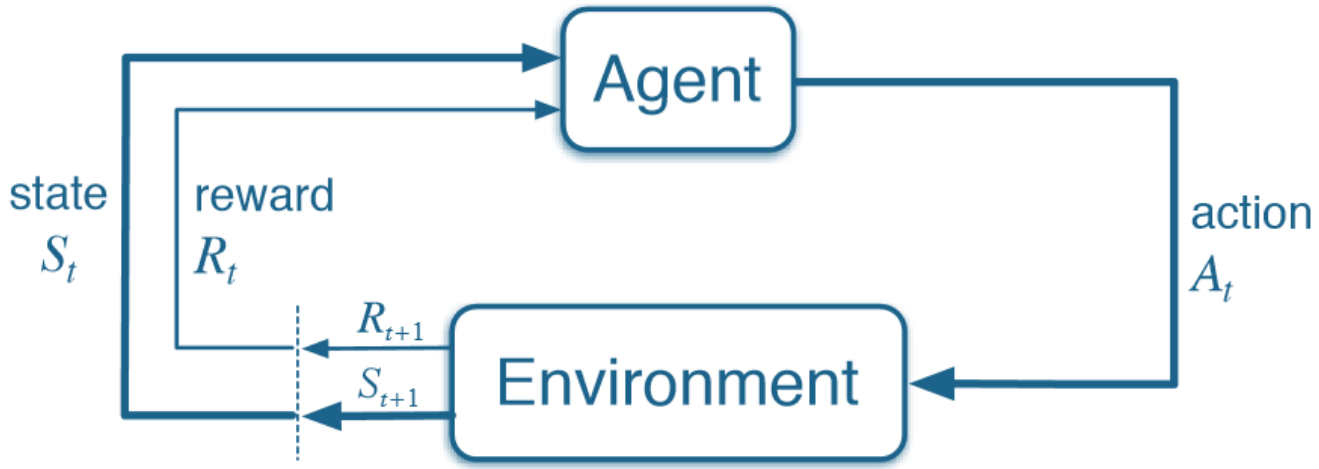
- Reimplementation of the optimal decision rule generation method.
- Extension of the hyper-parameters optimization GA to other ML models.
- Benchmark of several ML models on the TravisTorrent dataset using within-project and cross-project validation.
- Working on a RL approach.

Towards a Novel Reinforcement Learning Approach

RL Approach

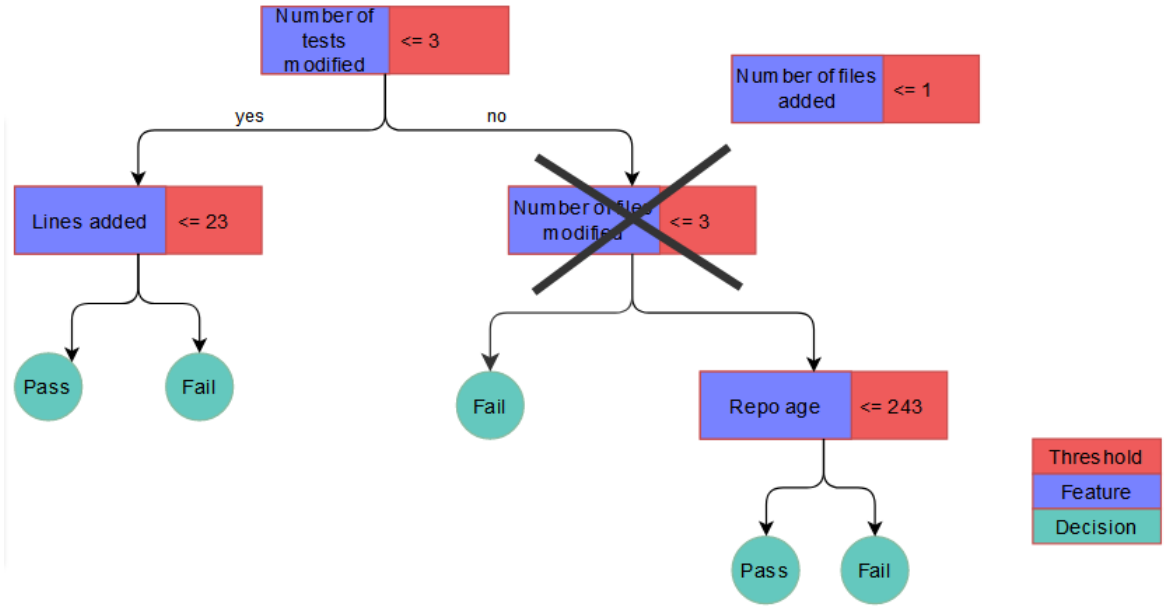
- A Deep Reinforcement Learning method to build decision trees for imbalanced classification.
- “Building Decision Tree for Imbalanced Classification via Deep Reinforcement Learning”, Guixuan Wen, Kaigui Wu (2021)
- Experiments on 15 imbalanced datasets indicate that this approach outperforms baseline DT building methods.

Reinforcement Learning



Game Principle

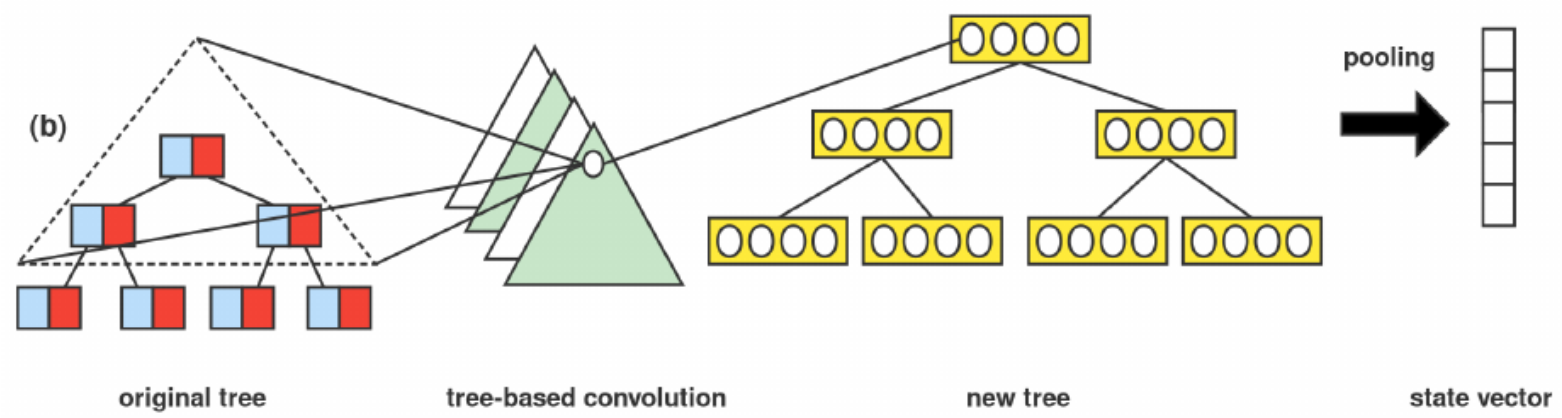
- A decision tree's node is composed of two parts: an attribute and a threshold value.



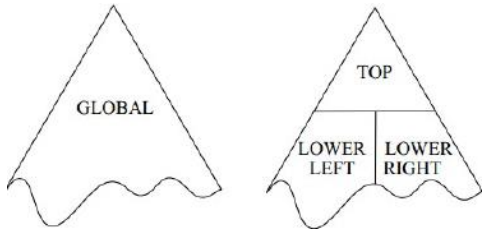
- Update tree nodes to improve the classification metrics.

Environment State

The state s is extracted by a tree-based convolution layer embedding in the actor-network.



- How to aggregate information?
- How to choose kernels?

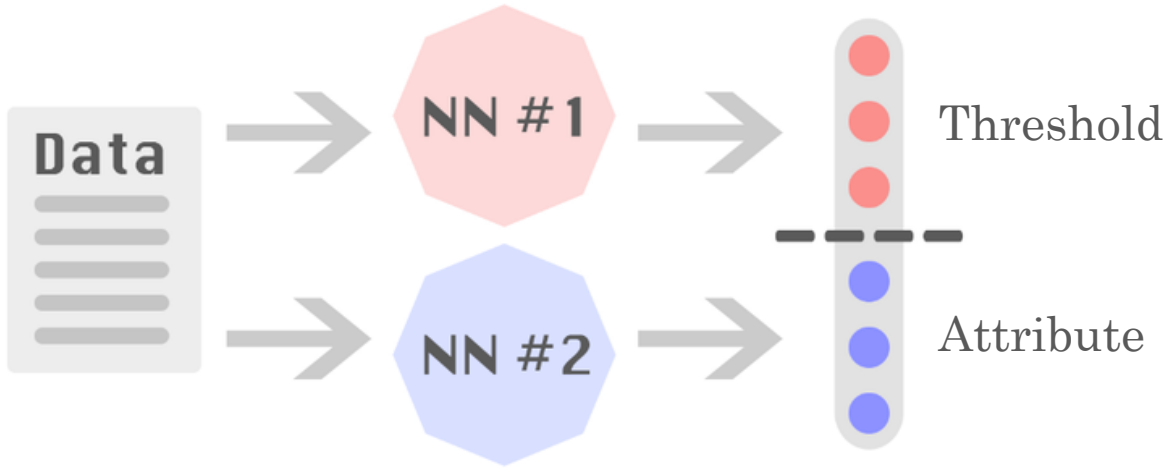


Agent's Actions

At each episode m, a set A_d of actions is taken to generate a new tree.

At each step t, we choose an action a_k for a single node.

- $A_d = \bigcup_{k \in [K]} \{a_k = (k, x_k) \mid x_k \in X_k \subseteq \mathbb{R} \}$
  - k is a feature
  - x_k is the threshold value
  - X_k is the value space of attribute k



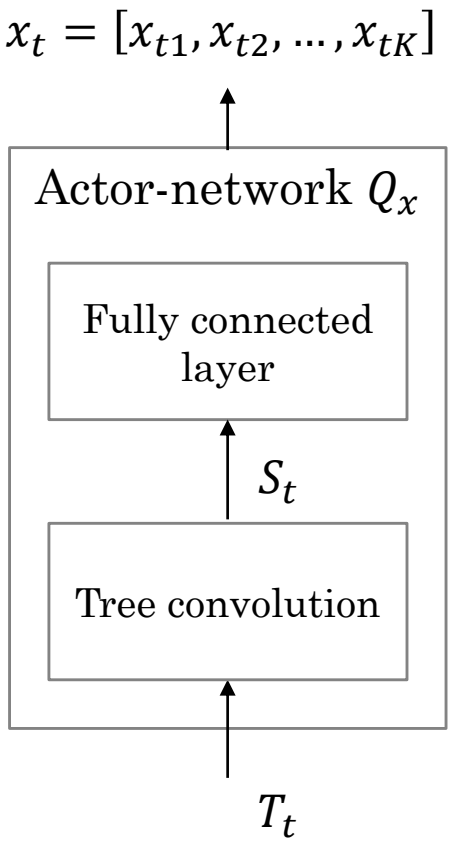
Reward Function

- After each node modification t, classify the data using the new tree.
- The predicted results $\hat{Y}_t$ and the truth Y_t are used to calculate a score.
- $r_t = sc_t - sc_{t-1}$

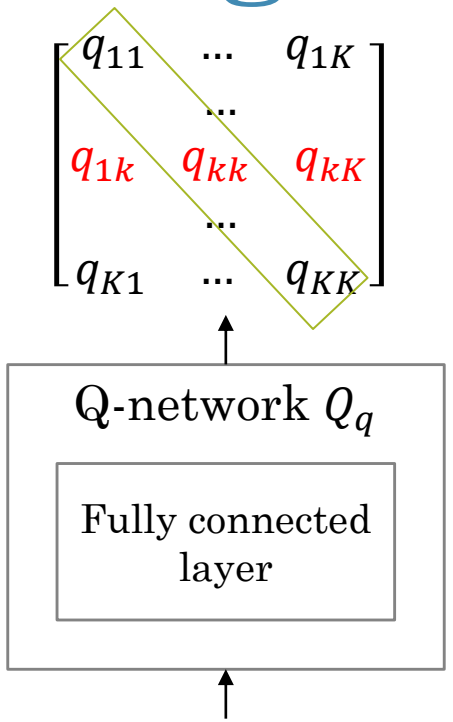
Training Process

For each node t:

- The actor network computes
  - The state S_t
  - The threshold values for all attributes X_t



Training Process



$$Q_t = [q_{11}, q_{22}, \dots, q_{kk}, \dots, q_{KK}]$$

Chose attribute K_t

$$k_t = argmax\, Q_t$$

$$= \underset{k \in [K]}{argmax}\, Q_q\,(S_t, xe_{tk}; \theta_q)$$

Choose action

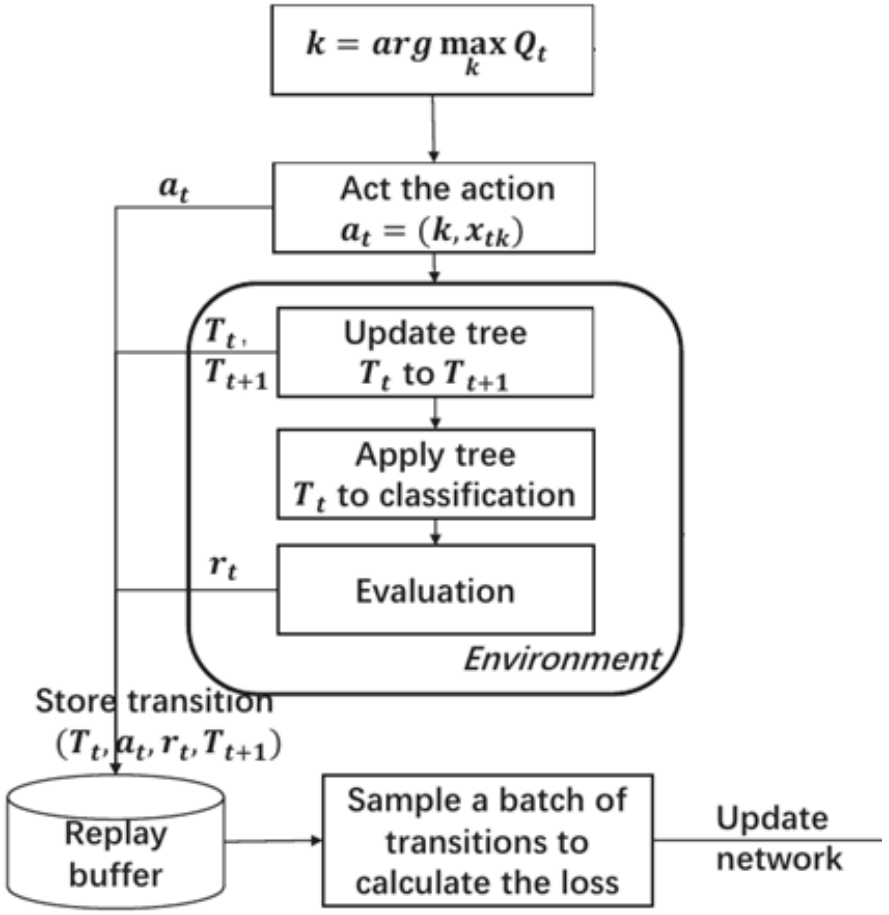
$$a_t = \begin{cases} (k_t, x_{tk}) \text{ with probability } 1 - \varepsilon \\ \text{random sample with probability } \varepsilon \end{cases}$$

attribute k<K for node t

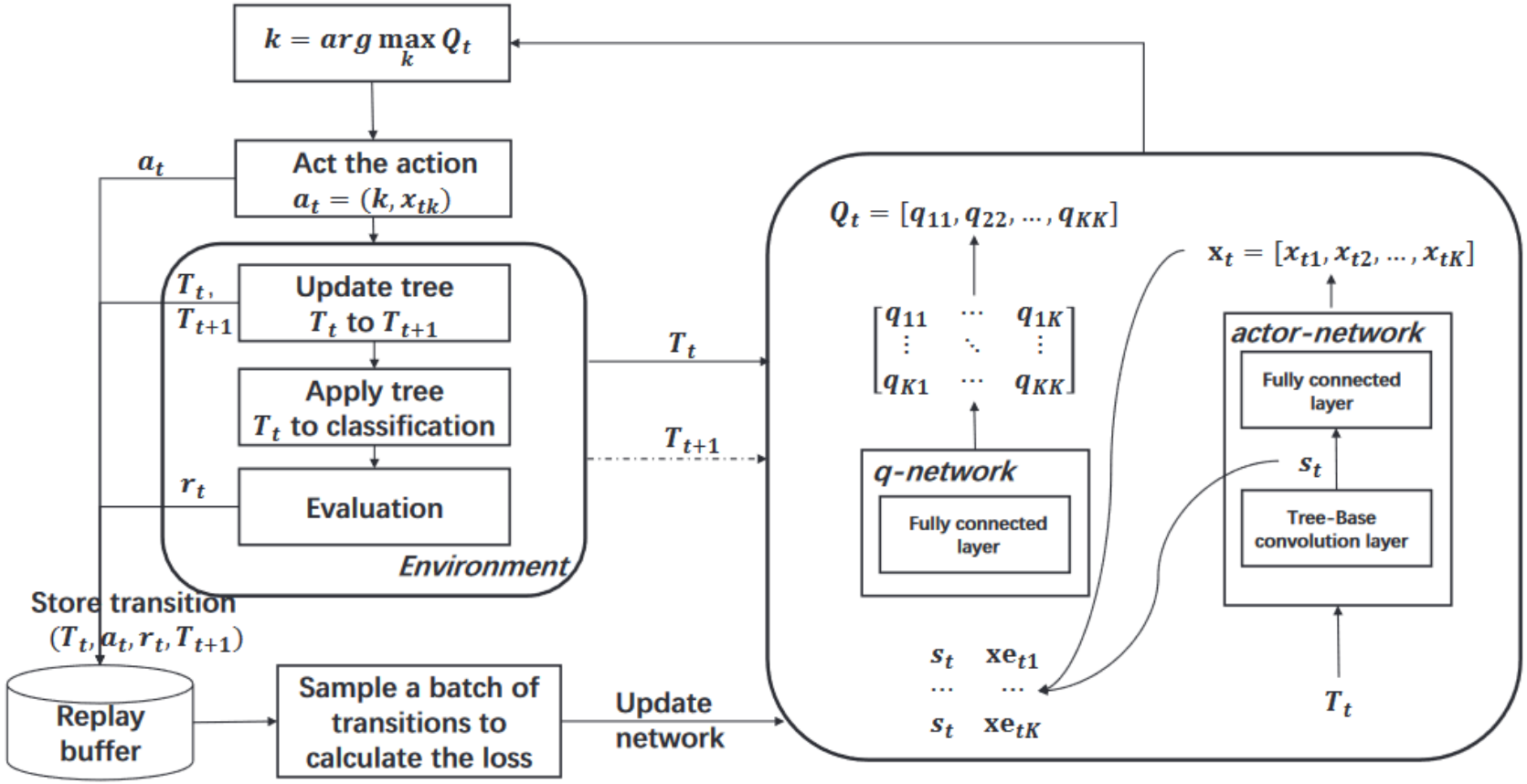
$$\begin{bmatrix} S_t\ ;\ xe_{t1} \\ \dots \\ S_t\ ;\ xe_{tk} \\ \dots \\ S_t\ ;\ xe_{tK} \end{bmatrix}$$

$$xe_{tk} = (0, \dots, x_{tk}, \dots, 0)$$

Training Process



General Architecture



Conclusion

- We aim to propose a novel Deep Q-Learning approach, and compare its classification metrics to other approaches from literature.