Optimizing Continuous integration using Artificial Intelligence

Maher Dissem

maher.dissem@gmail.com

May 13th, 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

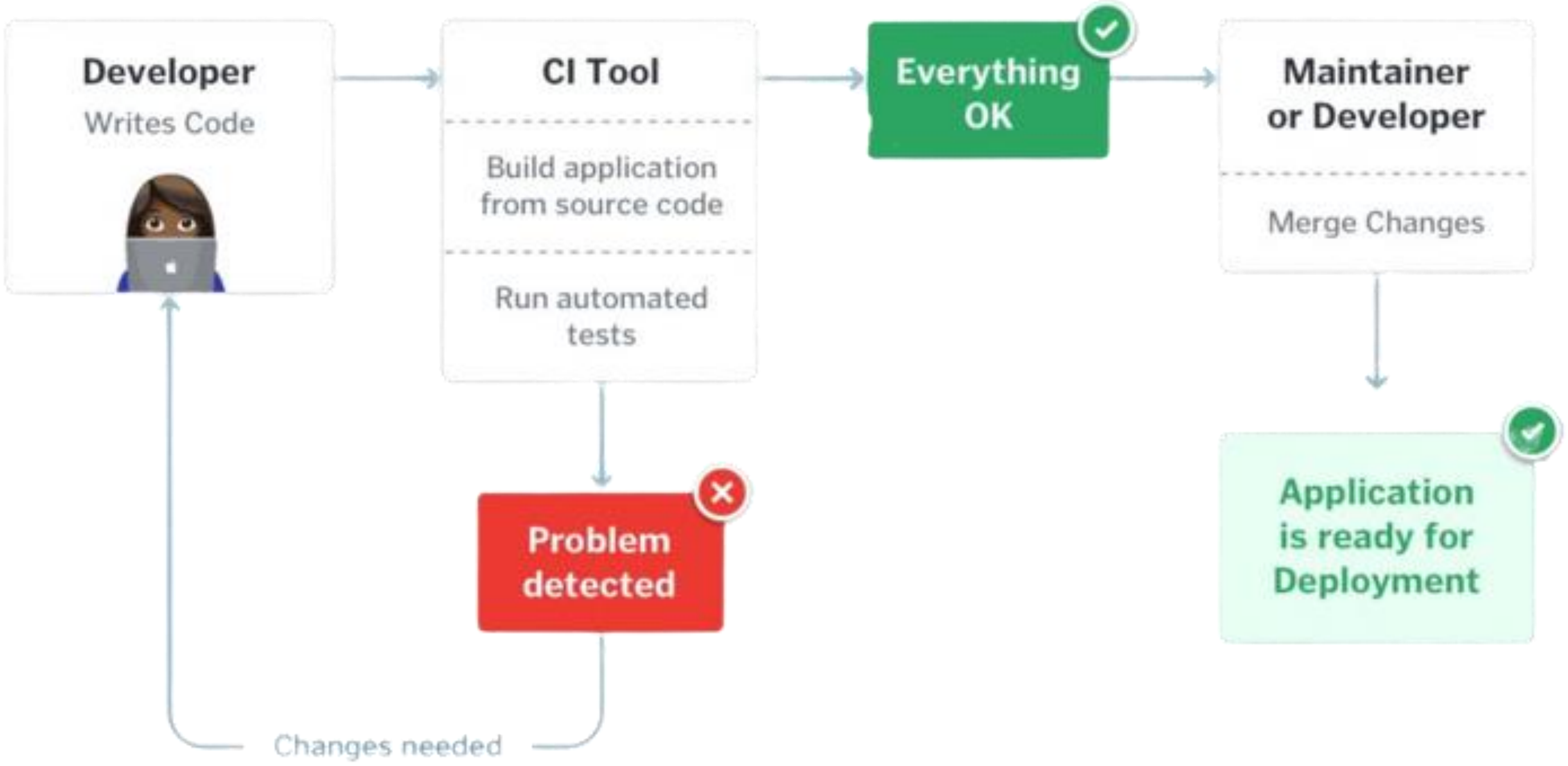
Introduction

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:

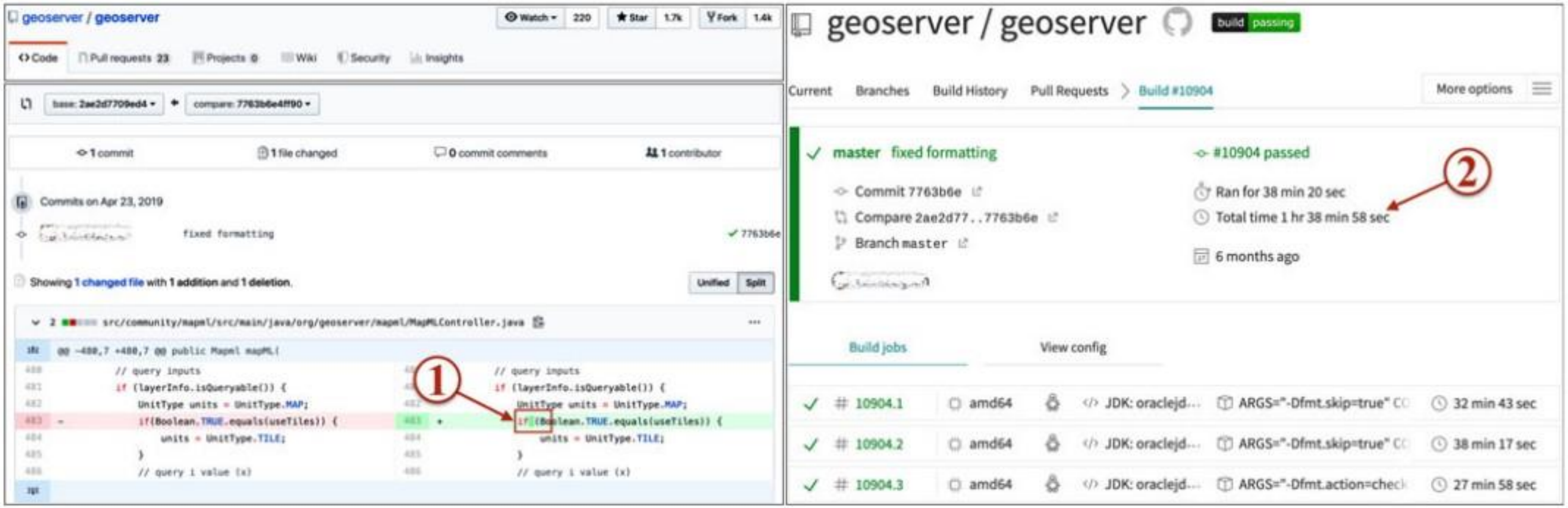
  Running only subset of tests that are more relevant to the changes made.

- CI skip:

  Skipping the execution of unnecessary commits.

Example

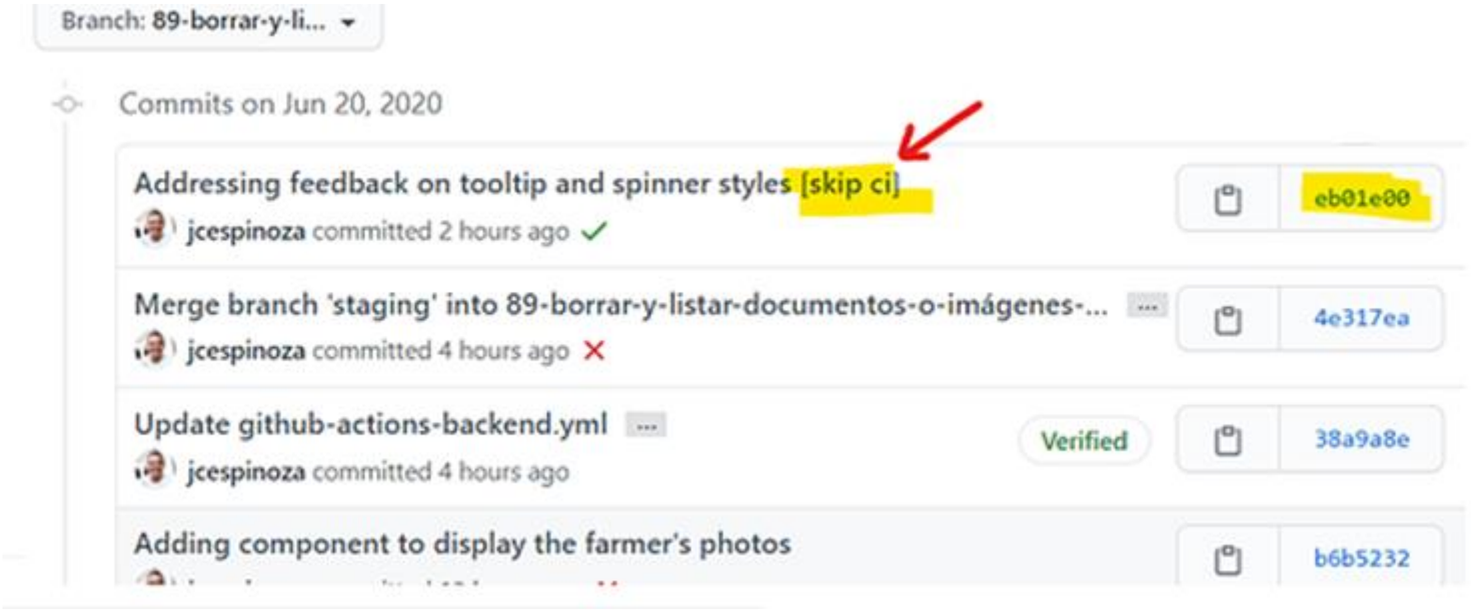
Some commits unnecessarily trigger the CI process.



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 and build logs from Travis CI's API

TravisTorrent: Synthesizing Travis CI and GitHub for Full-Stack Research on Continuous Integration Moritz Beller, Georgios Gousios, Andy Zaidman

Dataset Features

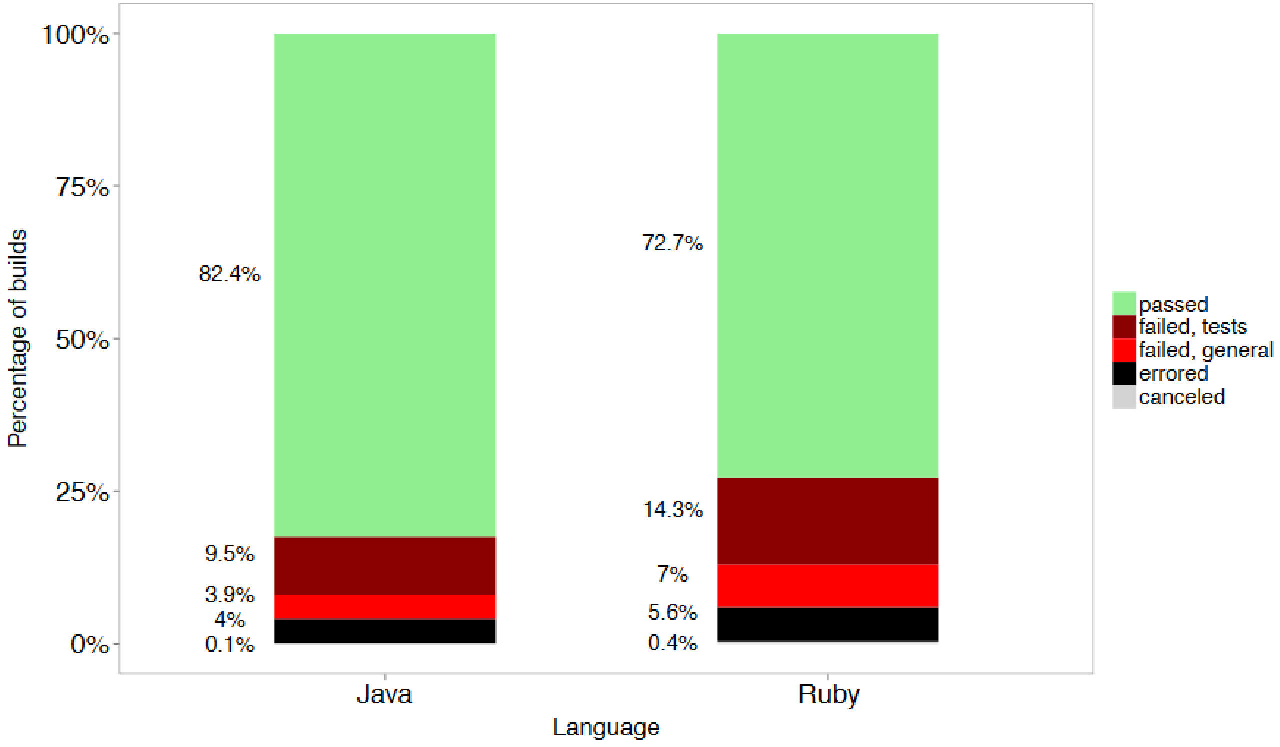
- 66 total features
- Extracted features fall under these categories:
  - Diffusion
  - Size
  - Purpose
  - History

Target is ’build status’ (passed/failed)

→ a binary classification problem

Build Outcome

- Build status of projects in TravisTorrent
  → Data is heavily imbalanced



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
  - F1 score
  - Area under the ROC curve
- RF usually yields the best results.

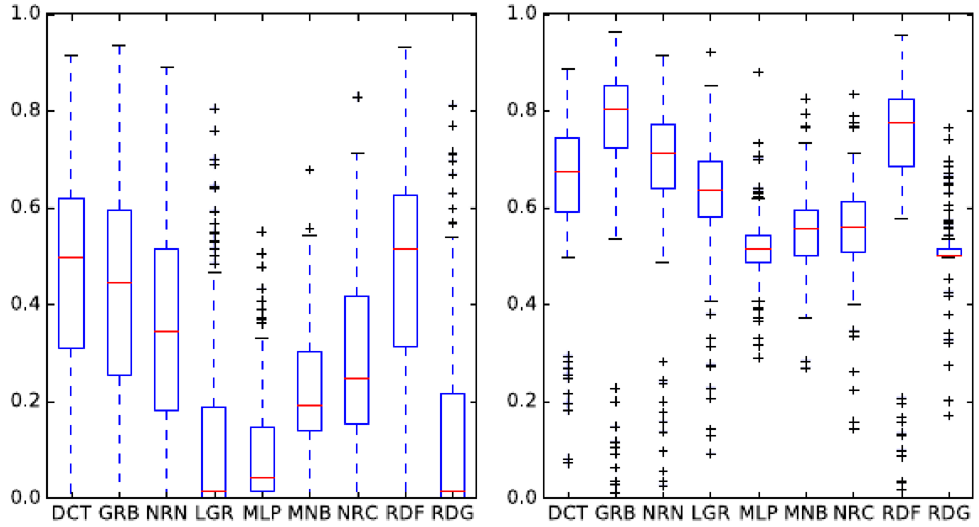
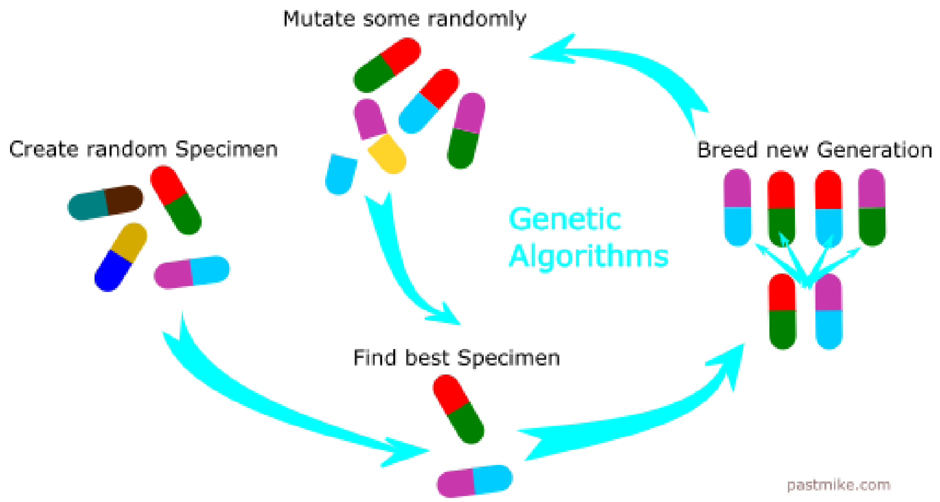


Fig. 2. F1-Score (left) and AUC (right) of Cross-validation Predictions

Genetic Algorithms



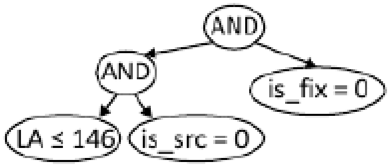
To adopt a genetic algorithm, we need to define the

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

Genetic Algorithms Approaches

- Finding the optimal IF -THEN rule using its binary tree representations.

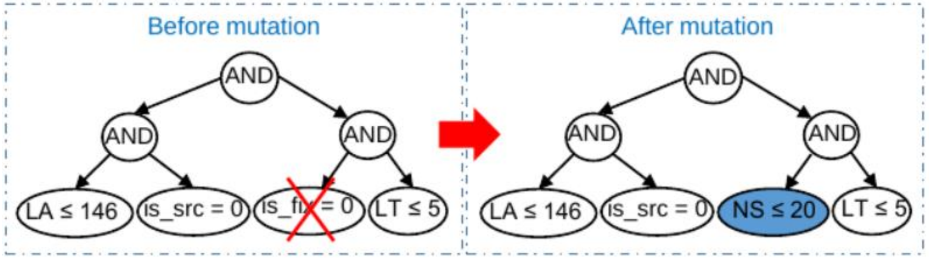
IF LA $\leq$146 AND IS_SRC =0 AND IS_FIX =0
THEN Skip commit.



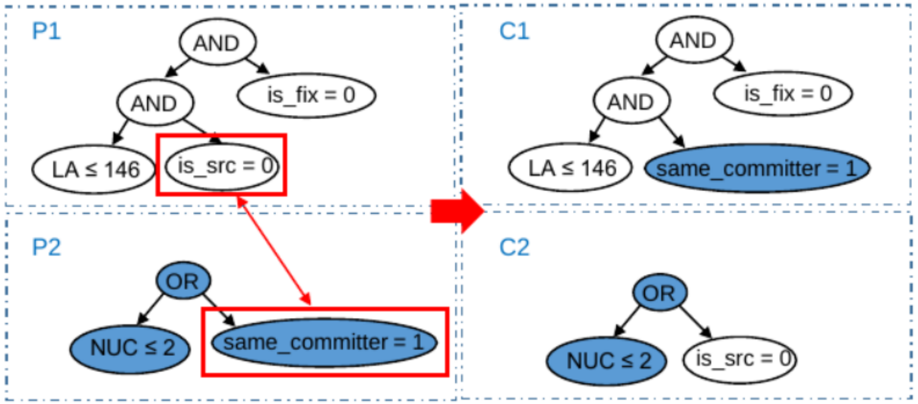
A candidate solution

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

Genetic Algorithms Approaches



Mutation



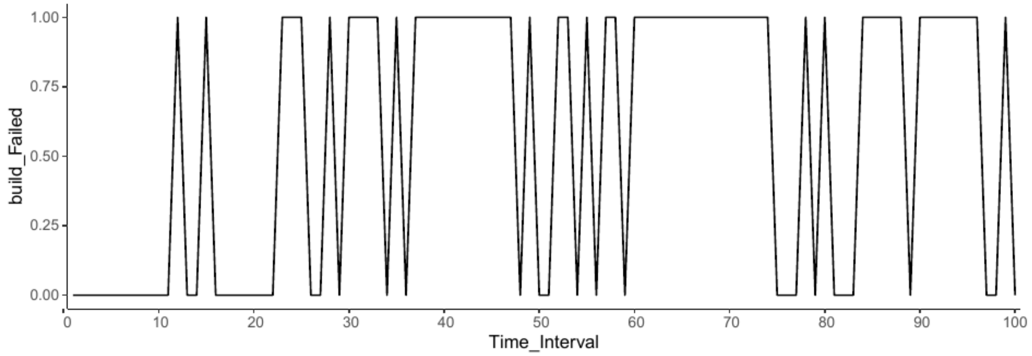
Crossover

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 LSTM



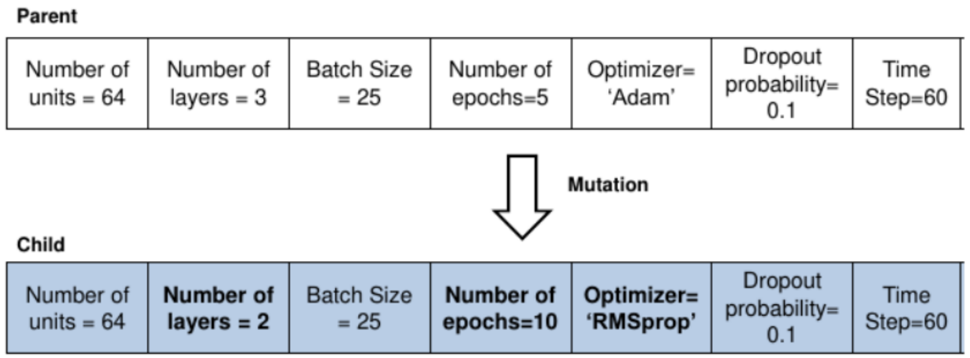
- Hyper-parameters optimization of LSTM using GA

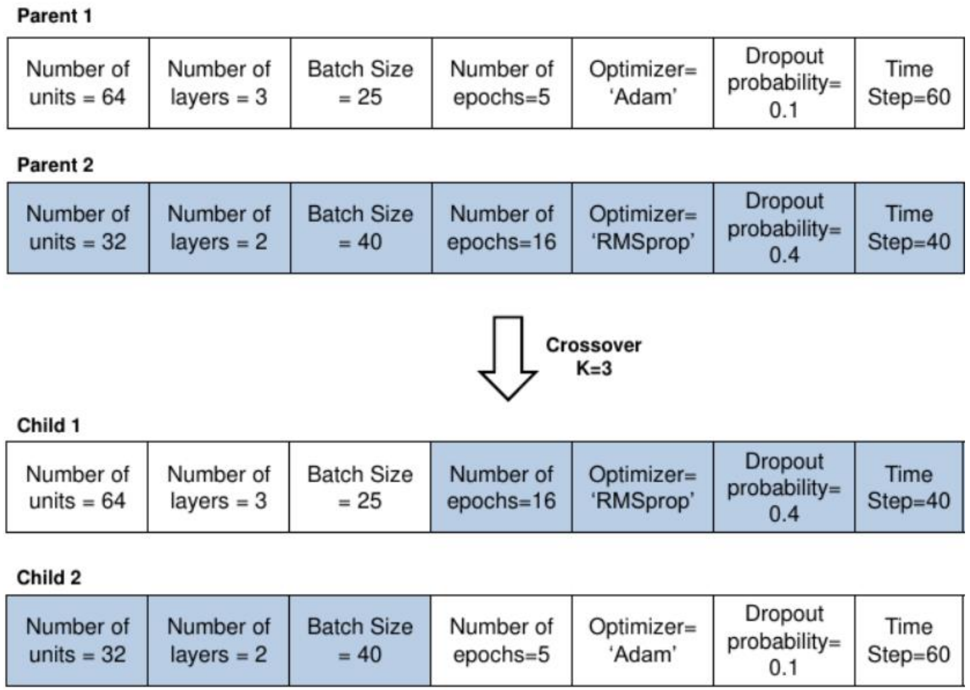
Number of units = 64	Number of layers = 3	Batch Size =25	Number of epochs=5	Optimizer= 'Adam'	Dropout probability= 0.1	Time Step=60

A candidate solution

Genetic Algorithms Approaches

- Fitness: validation loss





The Project So Far

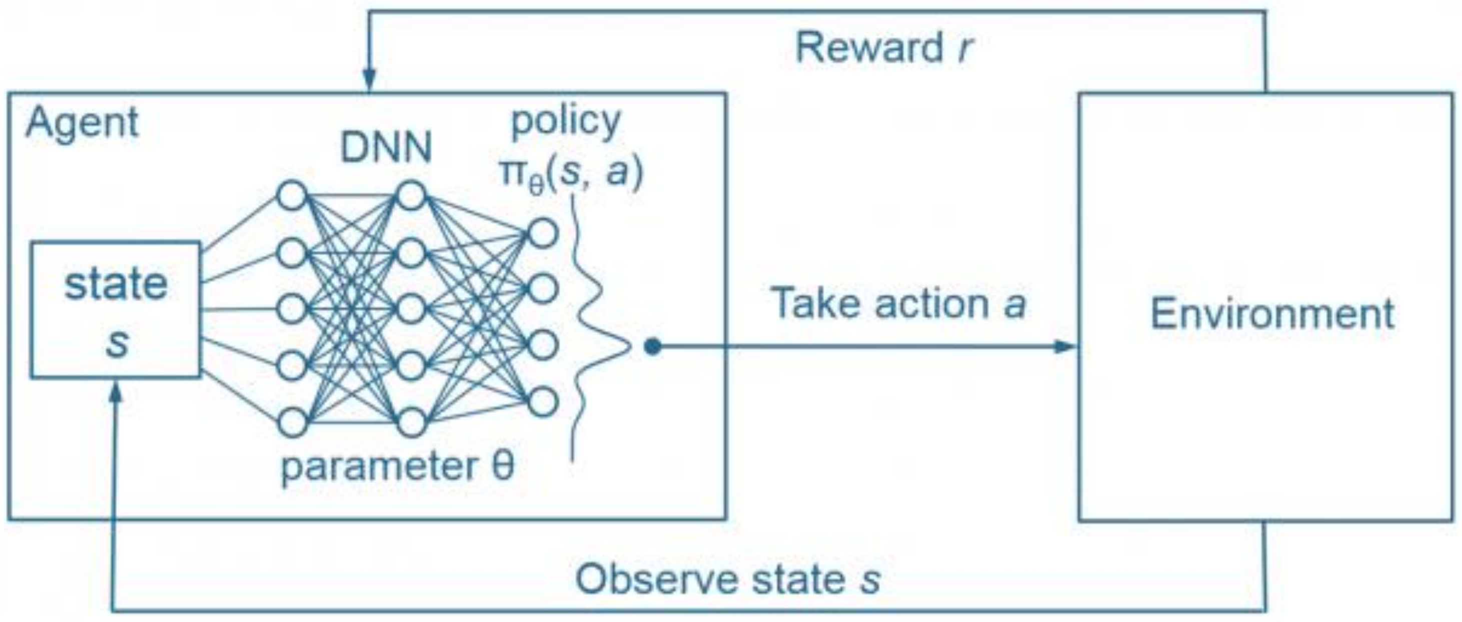
- Reimplementation of the previously mentioned approaches.
- Benchmark of several ML models on the TravisTorrent dataset using both within-project and cross-project validation.
- Working on a novel RL approach.

Towards a Novel Reinforcement Learning Approach

RL Approach

- A Deep Reinforcement Learning method to build decision trees.
- Based on "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.

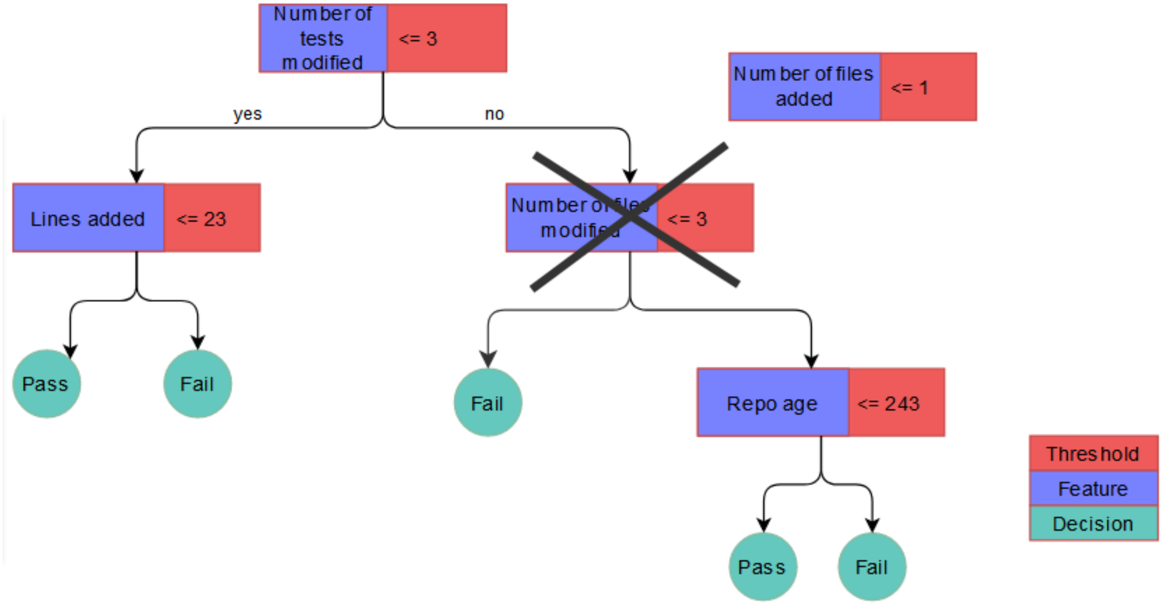
Deep RL



Reward r
Agent
DNN
policy
$\pi_\theta(s, a)$
state
s
parameter θ
Take action a
Environment
Observe state s

Game Principle

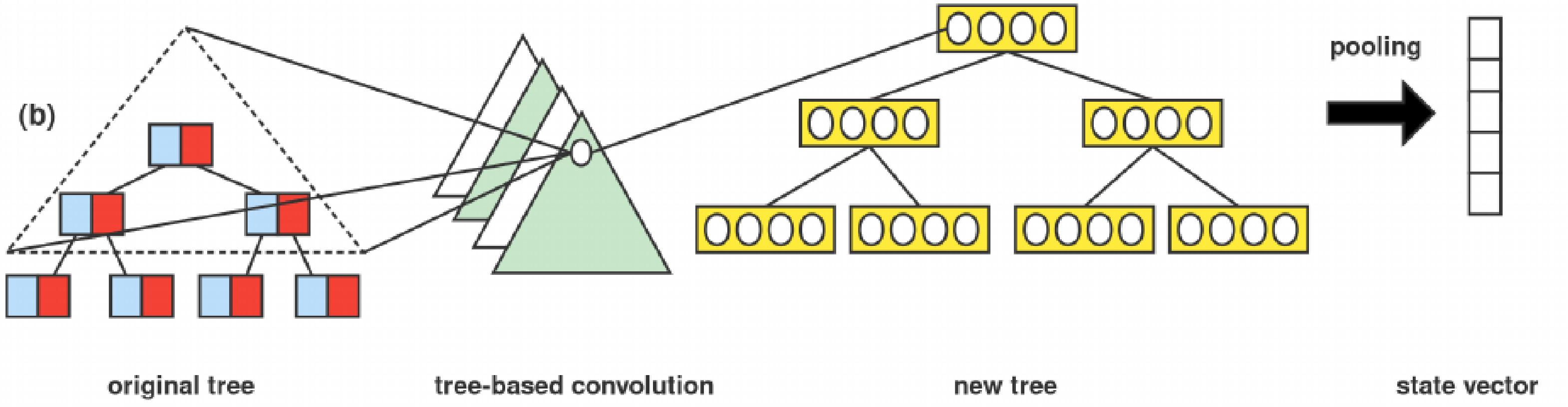
- A decision tree's node is composed of two parts: an attribute and a threshold value.



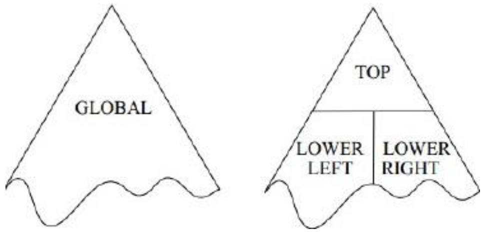
- Update tree nodes to improve the classification metrics.

Environment State

The state s is extracted by tree-based convolution.



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

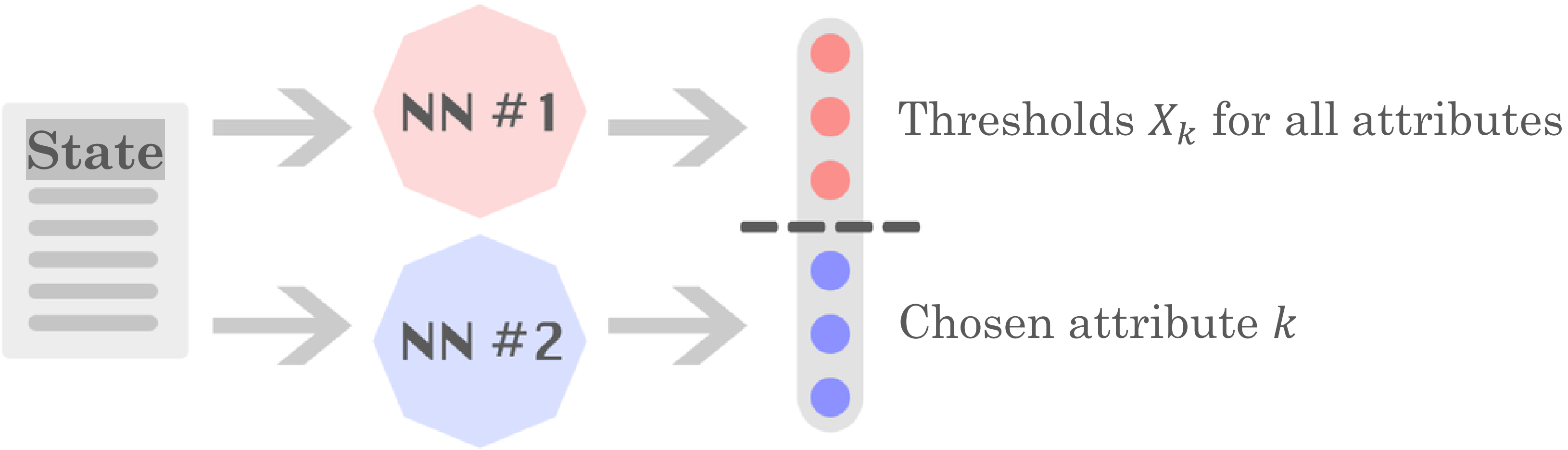


Agent's Actions

At each episode m, a set A_d of actions is taken to generate a new decision 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 the feature/attribute
  - x_k is the threshold value



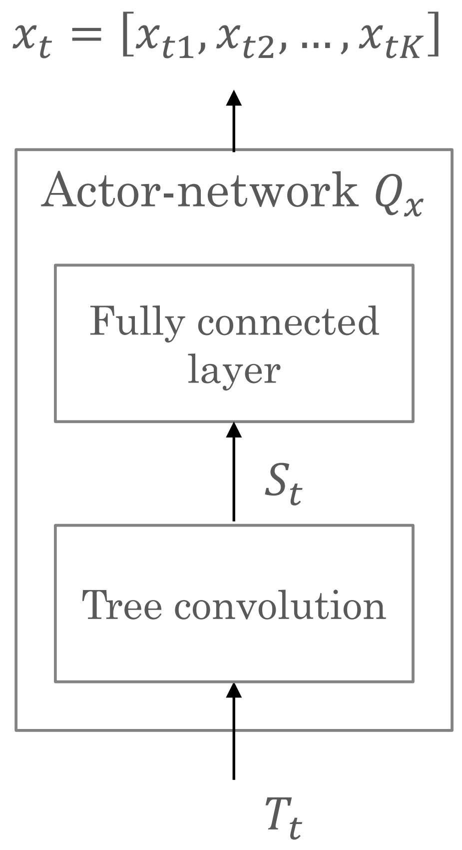
Reward Function

- 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 classification metric m_t.
- $r_t = m_t - m_{t-1}$

Training Process

For each node t:

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

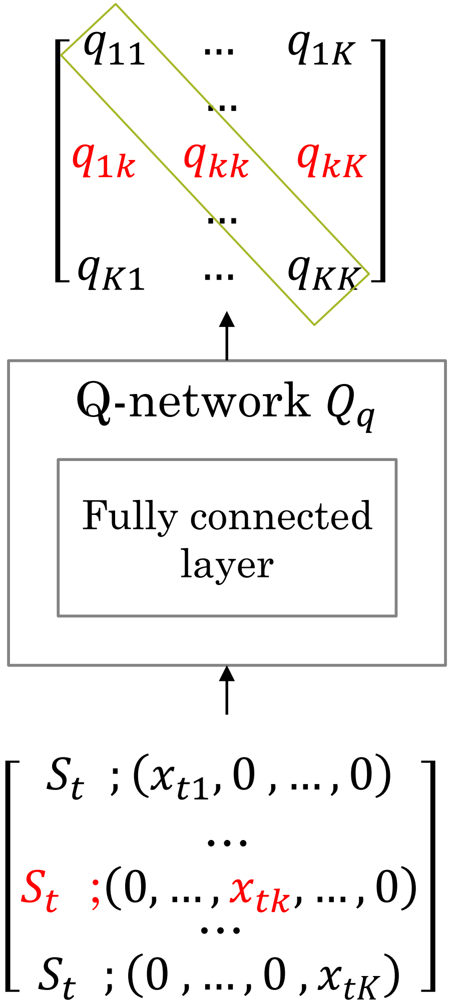


Output x_t :

26	5	6	…	
gh_diff_ src_files	gh_team_size	gh_num_of _comments	…	

→ choose a pair

Training Process



x_{tk}: threshold for attribute k, for node t

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

→ Expected reward for choosing k

Chose attribute K_t

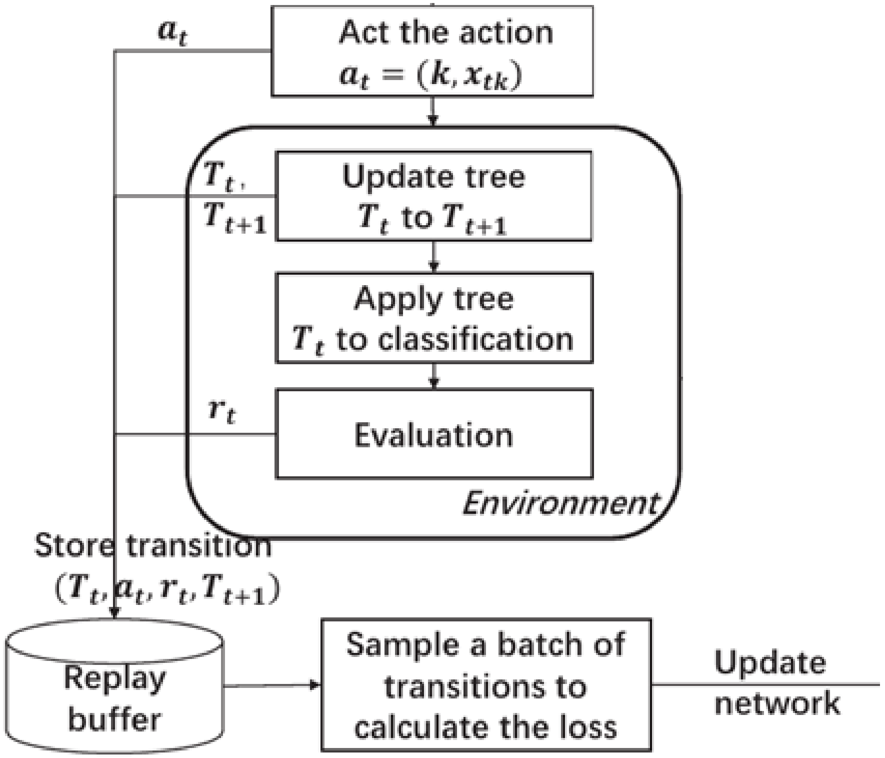
$$k_t = argmax\, Q_t$$

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

Choose action

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

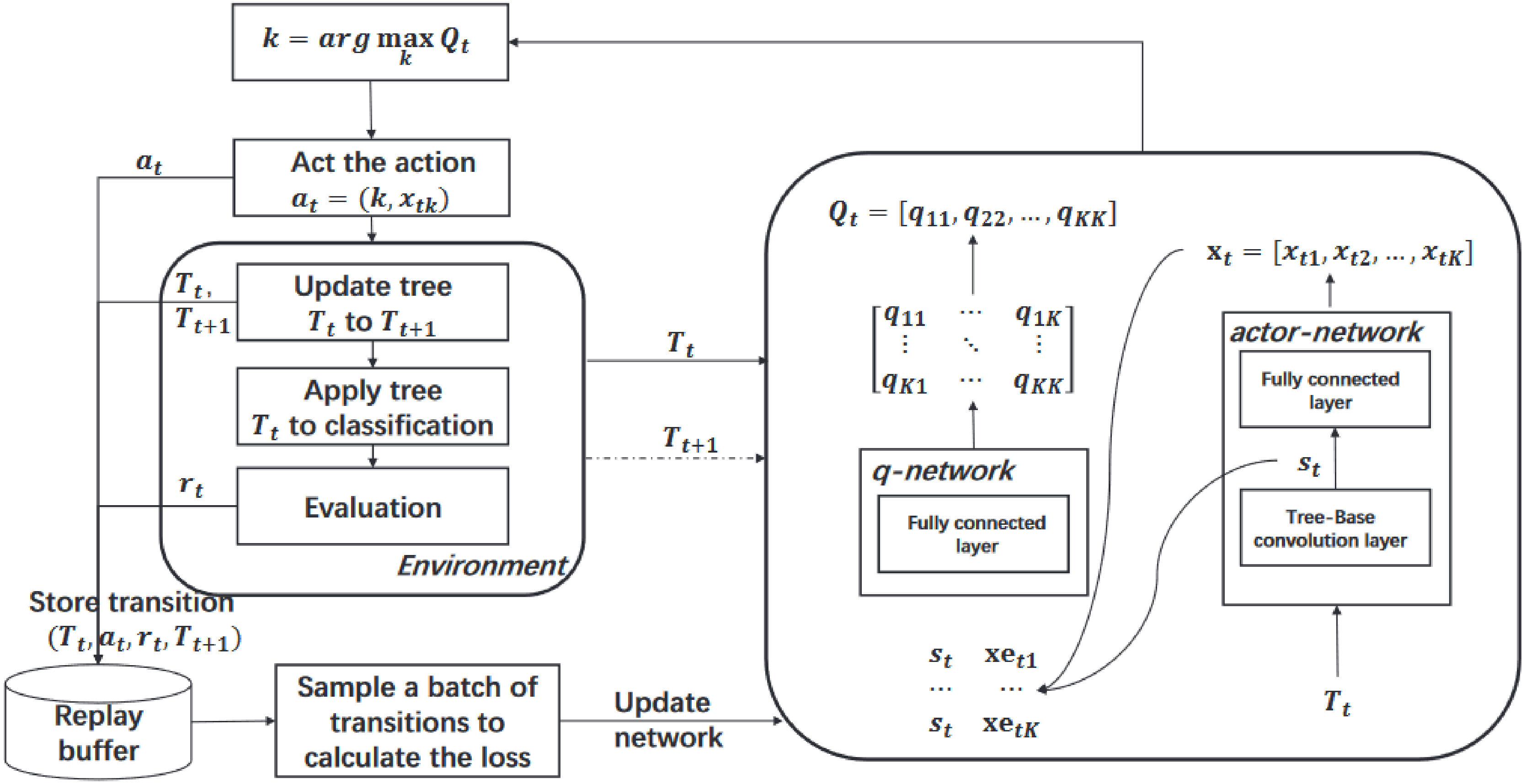
Training Process



Experience Replay

- prevent network weights from diverging due to temporal correlation.
- learn more from individual transitions multiple times.
- recall rare occurrences.

General Architecture



Difficulties

- State size 2 $(2^{depth} - 1)$
- Choice of the convolution operation
- Choice of hyper-parameters (tree depth, learning rate, etc.)

Conclusion

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