

UNIVERSITY OF BAHRAIN

**College of Information
Technology**

**Department of
Computer Science**



Development of Intelligent Tutoring Systems Using Hybrid Machine Learning Approach

A Thesis Submitted in Partial Fulfilment of the Requirements for the
Doctor of Philosophy Degree in Computing and Information Sciences

Submitted By

Fatema Alshaikh

20005999

Supervised By

Prof. Nabil Hewahi

(Professor)

University of Bahrain

December 2024



Dissertation Deposit Form

Author Name:	Fatema Abdullatif Ahmed Alshaikh		
Email:	20005999@stu.uob.edu.bh	Contact No:	39955377
College:	Information Technology		
Department:	Computer Science		
Thesis Title:	Development of Intelligent Tutoring Systems Using Hybrid Machine Learning Approach		
Year:	2024		

Author's Declaration: I agree to the following conditions:

The above thesis will be available (in the University of Bahrain library and externally) and reproduced as necessary at the discretion of the University of Bahrain. It may also be digitized by the University of Bahrain and made available via the university's repository or via other parties on the Internet.

This condition shall apply to ALL copies including electronic copies.

The above thesis has been provided on the understanding that it is copyright protected material and that no quotation from this thesis may be published without proper acknowledgement.

Signature of the Author:		Date:	15-Dec-24
--------------------------	---	-------	-----------

DECLARATION

I declare that this work is the result of my own investigation and that it has not already been accepted in substance for any degree, nor is it currently submitted for any degree.

Signed: -----

Date: 15/12/2024

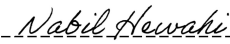
Fatema Abdullatif Ahmed Alshaikh (Candidate)

The Thesis defense committee considers the thesis titled:

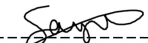
Development of Intelligent Tutoring Systems Using Hybrid Machine Learning Approach

Submitted by **Fatema Abdullatif Ahmed Alshaikh** to the College of Information Technology is satisfactory and acceptable for the Doctor of Philosophy in Computation and Information Sciences.

The Defense Committee:


Prof. Nabil Hewahi
(Supervisor)

Department of Computer Science
University of Bahrain


Prof. El-Sayed M. El-Alfy
(Member)

King Fahad University
Saudi Arabia


Dr. Fawzi Albalooshi
(Member)

Department of Computer Science
University of Bahrain

Abstract

The primary goals of research in the fields of education and Artificial Intelligence (AI) are to develop an autonomous agent that can perform tasks or make decisions without human intervention. An Intelligent Tutoring System (ITS) is essential to education because it gives students individualized, personal assistance as they learn problem-solving techniques through instruction and immediate feedback. This thesis presents three experiments with different architectures to investigate the effectiveness of AI and machine learning algorithms in the domain of intelligent tutoring system.

The first experiment builds three models based on the structure of Convolutional Neural Networks (CNN) to investigate the efficacy of Deep Learning (DL) algorithms in predicting student academic achievement. Two public datasets are used (PTdataset) and (xAPI-dataset). According to the experimental findings, the proposed CNN model's success in predicting student performance at early stages reached an accuracy of 94.36% using the PTdataset and 84.83% using the xAPIdataset.

In the second experiment, we used the Learning Classifier System (LCS) of Genetic Algorithms (GAs) and Reinforcement Learning (RL) to develop an adaptive educational model and construct a learning environment that is personalized for learners. We developed three models. The first model utilized LCS, and the second model utilized RL. The third model integrated a learning classifier system using genetic algorithms with reinforcement learning to create a hybrid approach, which we named it Adaptive Pedagogical Model Based on Learning Classifier System and Reinforcement Learning (PMCR). The study's findings indicate that PMCR more ideal than the other proposed models. It achieved an accuracy of 82% and an F1 score of 0.9 when tested across various problem cases.

The third experiment is based on Deep Reinforcement Learning (DRL). In ITS, DRL provides students with individualized learning experiences based on their own learning styles and preferences. We presented four different models. The first model is based on the structure of the Deep Q-Network (DQN). The second and third models (Double DQN and Dueling DQN) are extensions of DQN algorithm to address some limitations of the DQN approach. The last proposed model is a hybrid model that combines the standard DQN with a learning classifier system. The findings of the study indicate that both Double DQN and Dueling DQN showed complete data efficiency with 100% accuracy, 1.0 F1 score and 1.0 precision at highest training level of 1000 episodes and 10 steps whereas DQN performed fairly but was outperformed by advanced techniques.

Publications

This thesis derives from previously published works with my supervisor, Prof. Nabil Hewahi:

JOURNAL PAPERS

1. (Published) Alshaikh, F., & Hewahi, N. (2024). Convolutional Neural Network for Predicting Student Academic Performance in Intelligent Tutoring System. *International Journal of Computing and Digital Systems*, 15(1), pages 239–258.
URL: <https://doi.org/10.12785/ijcds/150119>
2. (Accepted with Revision) Alshaikh, F., & Hewahi, N. (2024). Adaptive Pedagogical Model Based the Classifier System and Reinforcement Learning. *The Computer Journal*
3. (Accepted with Revision) Alshaikh, F., & Hewahi, N. (2024). Intelligent Tutoring System: A Pedagogical Model Approach Based on Deep Reinforcement Learning. *IAENG International Journal of Computer Science*

CONFERENCE PAPERS

1. (Published) Alshaikh, F., & Hewahi, N. (2021). AI and Machine Learning Techniques in the Development of Intelligent Tutoring System: A Review. *International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, IEEE, pages 403–410.
URL: <https://ieeexplore.ieee.org/adocument/9582029>

Dedication

To my husband Mohamed, who has been a source of strength, support, patience, and motivation for me throughout this entire journey. I am truly blessed to have you as my life partner.

To my daughter, Rayan. I love you beyond measure. Thank you for your patience and support.

Acknowledgments

First and foremost, I want to express my gratitude and appreciation to God, the Almighty, for his great gifts that helped me complete my research. I would like to express my sincere gratitude to Professor Nabil Hewahi, my supervisor, for his continued encouragement and perceptive criticism during my research journey. His intense dedication to academic achievement and meticulous attention to detail have greatly influenced this dissertation. I couldn't have asked for a better supervisor than Prof. Nabil, whose knowledge and experience guided me every step of the way. His dynamism, vision, authenticity, and motivation deeply influenced me. He has instructed me on the methods for doing research and presenting study findings with the greatest degree of clarity. It was a significant privilege and honor to work and study under his mentorship. I am extremely appreciative of his efforts.

I would like to thank Prof. Mustafa Hammad for enlightening me with the first glance at research through his guidance and support. My gratitude also extends to the University of Bahrain, department of computer science professors and staff. In addition, I want to thank my colleagues for their friendship and the thought-provoking conversations that motivated me over my academic career.

I am highly grateful to my parents for their love, prayers, care, and sacrifices in educating and preparing me for my future. I am deeply thankful to my husband and my daughter for their love, understanding, prayers, and continuous encouragement in completing this research effort. Also, I extend my thanks to my sisters and brother for their support and motivation.

Finally, I would like to convey my appreciation to everyone who helped me, either directly or indirectly, to complete my research.

Contents

Defense committee's Approval	A
Declaration	B
Abstract	C
Publications	D
Dedication	E
Acknowledgments	F
Table of Contents	G
List of Tables	K
List of Figures	N
List of Abbreviations	P
1 Introduction	2
1.1 Overview	2
1.1.1 Intelligent Tutoring System (ITS)	2
1.1.2 Convolutional Neural Networks in ITS	4
1.1.3 Learning Classifier System in ITS	5
1.1.4 Reinforcement Learning in ITS	6
1.2 Research Problem	7
1.2.1 Convolutional Neural Network for Student Model . .	7
1.2.2 Learning Classifier System and Reinforcement Learning for Pedagogical Model	8
1.2.3 Deep Reinforcement Learning for Pedagogical Model	9

1.3	Research Objectives	9
1.4	Thesis Questions	10
1.5	Thesis Structure	11
1.6	Summary	11
2	Literature Review	14
2.1	Machine Learning Based Intelligent Tutoring Systems . . .	14
2.1.1	ITS and Artificial Neural Networks	16
2.1.2	ITS and Clustering	16
2.1.3	ITS and Bayesian Networks	17
2.1.4	ITS and Fuzzy Logic	17
2.1.5	Hybrid ITSs	18
2.2	Machine Learning for the Student Model	18
2.3	Genetic Algorithms and Reinforcement Learning for Peda- gogical Model	26
2.3.1	Genetic Algorithms	26
2.3.2	Reinforcement Learning	27
2.3.3	Integrated Approaches Using Genetic Algorithm and Reinforcement Learning	30
2.4	Deep Reinforcement Learning for Pedagogical Model	31
3	Convolutional Neural Network for Predicting Student Academic Performance in Intelligent Tutoring System	37
3.1	Introduction	37
3.2	Overview of CNN	39
3.3	The proposed Model	40
3.3.1	Environment	40
3.3.2	The Architecture of the Proposed Model	41
3.3.3	Model Creation	42
3.4	Dataset Description	50

3.4.1	PTdataset Description	50
3.4.2	XAPIdataset Description	50
3.4.3	Data preprocessing	52
3.4.4	Feature selection	55
3.5	Model Evaluation	56
3.6	Experimental Results	57
3.6.1	PTdataset Results	57
3.6.2	Comparison with previous studies using PTdataset .	64
3.6.3	xAPIdataset Results	68
3.6.4	Comparison with previous studies using xAPIdataset	70
3.7	Discussion	71
3.8	Summary	73

4 Adaptive Pedagogical Model Based on Classifier System and Reinforcement Learning 75

4.1	Introduction	75
4.2	Background	76
4.2.1	Components of the Learning Classifier System (LCS)	77
4.2.2	Overview of Reinforcement Learning	82
4.2.3	Combination of Genetic Algorithm and Reinforce- ment learning	84
4.3	Proposed Models Architecture	85
4.4	Model Evaluation	99
4.5	Experimental Results	100
4.5.1	Results of Model 1 (Learning Classifier System) . .	103
4.5.2	Results of Model 2 (Reinforcement Learning) . . .	104
4.5.3	Results of Model 3 - PMCR	106
4.6	Discussion	107
4.7	Summary	109

5	Deep Reinforcement Learning in Intelligent Tutoring Systems	111
5.1	Introduction	111
5.2	Background	112
5.2.1	Deep Q-network (DQN)	112
5.2.2	DQN main components	113
5.2.3	Double DQN	118
5.2.4	Dueling DQN	119
5.3	Proposed Models Architecture	121
5.4	Model Evaluation	132
5.5	Experimental Results	132
5.5.1	Set the hyper parameters of the models	133
5.5.2	Results of Model 1 DQN	133
5.5.3	Results of Model 2 : Double DQN	137
5.5.4	Results of Model 3 Dueling DQN	138
5.5.5	Results of Model 4 (GA - DQN)	139
5.6	Discussion	143
5.7	Summary	145
6	Conclusion and Recommendations	148
	References	152
	Abstract (in Arabic)	

List of Tables

2.1	Some previous AI Techniques in Intelligent Tutoring Systems	15
2.2	Summary of the previous related work for the student model	25
3.1	Model 1 layer's description (PTdataset)	45
3.2	Model 2 layer's description (PTdataset	47
3.3	Model 3 layer's description (PTdataset)	48
3.4	PTdataset Description	50
3.5	Attributes information of PTdataset	51
3.6	xAPIdataset Description	52
3.7	Attributes information of xAPIdataset	53
3.8	Evaluation metrics used in the proposed Model	57
3.9	Different cases for the PTdataset	58
3.10	Performance evaluation of PTdataset (1044) imbalanced with- out any feature selection	59
3.11	Comparison of accuracy of PTdataset (1044) imbalanced with DT as feature selection and without any feature selection	60
3.12	Accuracy of PTdataset (1044) imbalanced with PCA as fea- ture selection	61
3.13	Comparison of accuracy of PTdataset (1044) with DT, PCA as feature selection and without feature selection	62
3.14	Comparison of accuracy of PTdataset (645) with feature se- lection and without feature selection	63

3.15	Comparison of accuracy of PTdataset with feature selection and without feature selection	66
3.16	Using PTdataset, evaluate the accuracy of earlier research and the proposed model	67
3.17	Different Cases for xAPIdataset	69
3.18	Comparison of accuracy of xAPIdataset with a feature se- lection and without any feature selection	69
3.19	Using xAPIdataset, Compare the Accuracy of Earlier Re- search and the Proposed Model.	70
4.1	Problem description	89
4.2	Example of the initial population	101
4.3	Example of the population of 15 students and their answers to problems	102
4.4	Parameters of the GA Algorithm	103
4.5	Results Obtained from the First Model, the Classifier System	104
4.6	Parameters of the RL	105
4.7	Results Obtained from the Second Model	105
4.8	Results obtained from the third model using the same prob- lem case set	106
4.9	Results obtained from the third model using different prob- lem cases	107
5.1	Parameters of DQN	133
5.2	DQN - The result obtained using the same set of test cases .	134
5.3	DQN - The result obtained using different test cases	135
5.4	Double DQN-The result obtained using the same set of test cases	137
5.5	Double DQN-The result obtained using different test case . .	138

5.6	Dueling DQN-The result obtained using the same set of test case	139
5.7	Dueling DQN-The result obtained using different test cases . .	139
5.8	GA+DQN - The result obtained using the same set of test cases	140
5.9	GA+DQN -The result obtained using different test cases . .	140
5.10	Comparison of the accuracy of the four proposed models using different set	141
5.11	Comparison of the precision and F1 score for the four proposed models	142
5.12	Comparison of the evaluation metrics for several proposed machine learning models	143

List of Figures

1.1	The architecture of ITS.	4
3.1	Basic CNN structure	40
3.2	The structure of the proposed model	43
3.3	Model 1 structure	44
3.4	Model 2 structure	46
3.5	Model 3 structure	49
3.6	Visualization graph for balancing PTdataset	55
3.7	Comparison of accuracy of the proposed model and previ- ous studies for all instances of PTdataset	67
3.8	Comparison of proposed and previous models' accuracy for a portion of PTdataset	68
3.9	Comparison of accuracy of the proposed model and previ- ous studies for xAPIdataset	71
4.1	Learning classifier system main process	77
4.2	Bucket Brigade Algorithm procedure	79
4.3	The three adopted models	86
4.4	Model 1: The architecture of the learning classifier system .	87
4.5	Single Point Crossover	93
4.6	Swap Mutation	93
4.7	Model 2 : Reinforcement Learning	95

4.8	Model 3: Combination of learning classifier system and reinforcement learning (PMCR)	98
4.9	The procedures to evaluate the proposed models	100
4.10	Compares the accuracy of the proposed models	108
4.11	Comparison of accuracies obtained using the same test for problem cases vs. a different set for Model 3 (PMCR) . . .	108
5.1	Q-Learning vs Deep Q-Learning.	114
5.2	Reinforcement learning vs Deep reinforcement learning. . .	114
5.3	DQN flow structure.	118
5.4	Neural network architecture vs Dueling neural network architecture.	120
5.5	Model 1: DQN	126
5.6	Model 4: GA with DQN	130
5.7	Rewards vs Episodes for the standard DQN algorithm. . . .	136
5.8	Comparing the evaluation metrics for several machine learning models	144

List of Abbreviations

A	Advantage Stream
Adam	Adaptive Moment Estimation
AHRec	Agent to Human Recommender System
AI	Artificial Intelligence
AIES	Adaptive and Intelligent Educational Systems
ANN	Artificial Neural Network
BAG	Bagging
BiLSTM	Bidirectional Long Short-Term Memory
BKT	Bayesian Knowledge Tracing
BN	Bayesian Networks
CITS	Conversational Intelligent Tutoring Device
CNN	Convolutional Neural Network
DBD	DataBase Design
DDQN	Double Deep Q-Network
DF	Discount Factor
DL	Deep Learning
DNN	Deep Neural Network
DQN	Deep Q-Network

DRL	Deep Reinforcement Learning
DT	Decision Tree
Dueling DQN	Dueling Deep Q-network
FB-ITS	BN and fuzzy logic
GA	Genetic Algorithm
GBML	Genetic-based reinforcement learning
GP	Genetic Programming
ICAI	Intelligent Computer-Assisted Learning
IIT	Immersive Intelligent Tutoring
ITS	Intelligent Tutoring Systems
KNN	k-Nearest Neighbor
LCS	Learning Classifier System
LMS	Learning Management System
LR	Learning Rate
MAE	Mean Absolute Error
MB	MultiBoost
ML	Machine Learning
MLP	Multilayer Perceptron
NB	Naïve Bayes

PCA	Principal Component Analysis
PMCR	Pedagogical Model Based on the Classifier System and Reinforcement Learning
POMDPs	Markov Decision Processes
QL	Q-learning
ReLU	Rectified Linear Unit
RF	Random Forest
RFC	Random Forest Classifier
RL	Reinforcement Learning
RMSE	Root Mean Square Error
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
V	Value Stream
VT	Voting

Chapter 1

Introduction

1.1 Overview

1.2 Research Problem

1.3 Research Objectives

1.4 Thesis Questions

1.5 Thesis Structure

1.6 Summary

Chapter 1

Introduction

1.1 Overview

This chapter provides a concise introduction to the essential background related to various topics addressed in this thesis. This will be followed with the problem statement, along with the objectives and research questions of the thesis. Finally, it will outline the structure of the rest of the thesis.

1.1.1 Intelligent Tutoring System (ITS)

An Intelligent Tutoring System (ITS) is important in education because it gives adaptive personalized, one-on-one, computer-based support to educators to handle problems by providing instruction and immediate feedback. ITS can be one application of Artificial Intelligence (AI) in education. It provides a smart learning environment for students without intervention from the teacher. ITS's primary goal is to support and help learners obtain domain-specific intellectual knowledge in a practical and productive manner through the use of different computing. ITS has been developed in different subjects, e.g. mathematics, science, linguistics, physics, and the medical sector. The idea of ITS is to obtain knowledge about the learners, then provide instructions, feedback, and hints to each student individually

according to her/his responses.

The development of Intelligent Computer-Assisted Learning (ICAI) in the 1960s and 1970s marked the beginning of ITS. The phrase "Intelligent Tutoring Systems" was first introduced in 1982. Six years later, in 1988, the idea was consolidated widely in the first ITS conference (Nkambou, Bourdeau, & Mizoguchi, 2010). Since then, there has been an improvement in research and studies related to development of such system.

The traditional architecture of ITS is composed of four main components: student model, tutor model (pedagogical model) , domain model and user interface (Alkhatlan & Kalita, 2018). More details about each component as shown in Figure 1.1 and as follows:

1. A student or learner model that contains the states and characteristics of each learner. Consider the knowledge, actions, and learning methods of a student. The main goal of the learner model is to ensure that the system has all the necessary information about each learner so it can respond properly.
2. A tutor or pedagogical model consists of the teaching strategies that will most effectively assist the learner's progress, e.g., how to teach, the steps of learning, and questions a learner might ask. It is responsible for managing the reaction of the student with the systems.
3. A domain or knowledge or expert model contains topics and all materials that students intended to learn. It represents the content, the skills, and procedures that students are required to learn. The model also includes information about student mistakes or misunderstandings so that ITS can provide qualified assistance.
4. A user interface is a form of communication between users and a machine. It can be graphic communications, emotional and social interactions, natural language communication, and component interface.

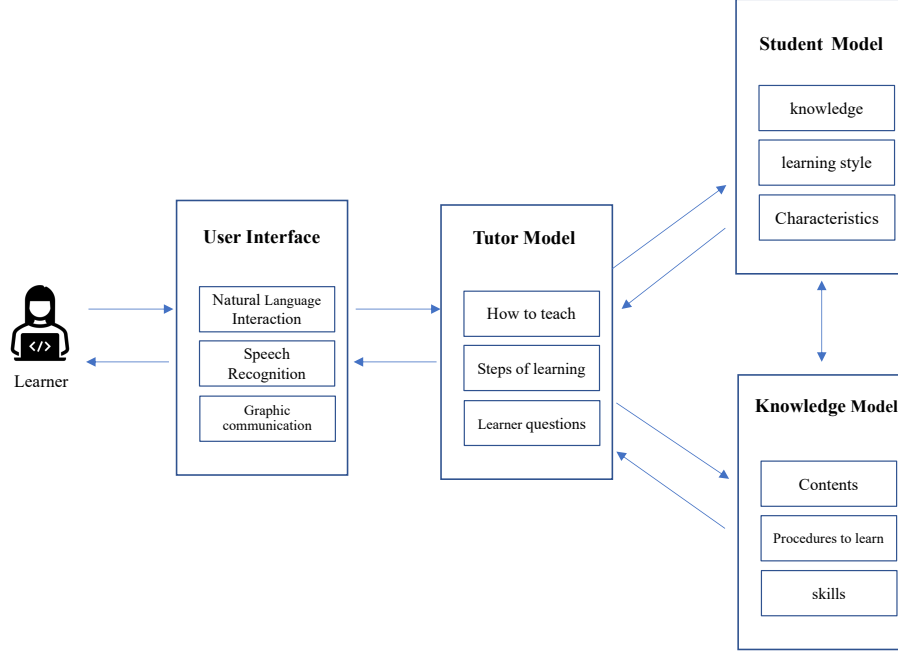


Figure 1.1: The architecture of ITS.

1.1.2 Convolutional Neural Networks in ITS

Convolutional Neural Network (CNN) is a deep learning network structure that obtains knowledge directly from data. CNN consists of an input layer, convolution and pooling layers, one or more fully connected layers, activation function and a final output layer (Zhao et al., 2024). These layers perform operations that extract unique characteristics of the data. The convolution layer utilizes a series of convolutional filters to process the input, in which each filter selectively generates particular features from that input. The pooling layer is simplifying the output through the nonlinear down-sampling which reduces the number of parameters that the network must learn and thus improves generalization. These processes are iterated throughout tens or hundreds of layers, where each layer has the ability to recognize different features. Fully connected layers, together with activation functions, perform tasks such as classification or regression on the extracted features. To optimize CNN performance, various regulatory mechanisms

such as batch normalization and dropout are incorporated alongside diverse mapping functions.

CNNs have demonstrated remarkable success in addressing a wide range of challenging tasks, including image classification, face recognition, natural language processing, language translation and automatic speech understanding. Furthermore, a growing number of applications in the field of education are using CNNs to improve learning experiences and achievements. CNNs are significantly influencing the development of intelligent tutoring systems (Akour, Alsghaier, & Al Qasem, 2020). They can be used to create student modeling based on patterns in student behaviors. They can customize learning paths by analyzing visual data gathered from students' interactions with educational materials and extracting new features. In addition, the CNN architecture can be used in educational institutions to anticipate students academic performance by analyzing and predicting, whether will pass or fail and take appropriate action to assist them.

1.1.3 Learning Classifier System in ITS

The first classifier system was proposed by John Holland in (1978). Holland defines the classifiers as a conditional rule based on if-then statements. The term "classifier" originates from the capacity of rules to categorize messages into various message sets (Urbanowicz & Browne, 2017).

Learning Classifier System (LCS) consists of three main components: classifiers (rules), which map input conditions into output actions or predictions; the Bucket Brigade Algorithm, which is used for credit assignments and to calculate the specific impact of each classifier in obtaining good outcomes; the Genetic Algorithm (GA), which is an optimization technique that employs natural selection and reproduction to discover optimal computational problem solutions.

GA evolves problem solutions using selection, crossover, and mutation

to resemble natural evolution (Vie, Kleinnijenhuis, & Farmer, 2020). Utilizing LCS which is based on the structure of genetic algorithm into the development of pedagogical model of intelligent tutoring systems can improve the learning environment. It can optimize the learning experiences and help to create more personalized and effective learning tailored to individual needs.

1.1.4 Reinforcement Learning in ITS

Reinforcement learning (RL) is a machine learning technique that focuses on how autonomous agents make decisions. An autonomous agent is a system capable of making decisions and taking action in response to its environment without requiring direct guidance from a human user (Ding, Huang, Yuan, & Dong, 2020). The agent learns to complete a task via trial and error without any assistance from a human user. The main components of RL include the agent, environment, state, action, and reward. An agent receives a state from the environment, performs an action, and returns a reward. The agent's goal is to maximize cumulative rewards. This iterative learning process is particularly useful for algorithms to solve complex decision-making problems in dynamic environments.

These features of reinforcement learning can enhance the pedagogical model in intelligent tutoring systems. It has the ability to provide efficient and adaptable learning environments customized to meet the specific needs of the learner. In addition, it helps instructors by generating instruction strategies for teaching topics, and techniques without relying on pre-established rules. It can select appropriate learning materials for student according to his profile and level. On the other hand, the integration of neural networks with reinforcement learning which is known as Deep Reinforcement Learning (DRL) enhances the robustness of RL applications and create new opportunities for improving intelligent tutoring systems by enabling

them to learn optimum strategies for directing students through customized learning paths (Woolf, 2007; Y. Li, 2017). Thus, the potential of deep reinforcement learning algorithms to acquire knowledge from interactions with the environment advances the capability of the pedagogical model in ITS to represent student behavior to be able to adjust to their specific requirements and enhance their tutoring sessions (Memarian & Doleck, 2024).

1.2 Research Problem

1.2.1 Convolutional Neural Network for Student Model

Student modeling is an essential component of ITSs and belongs to a widespread research domain. It includes the development of a comprehensive student profile that involves previous knowledge, educational preferences, behaviors, and skills. Data from a variety of sources, including social media, internet platforms, and academic records, can be used to create this model. Student profiles serve as the basis for individualization in intelligent tutoring systems and adaptive educational software.

Many academics have utilized traditional machine learning algorithms to predict student academic achievement. Still, relatively limited studies have used convolutional neural networks' structure in the intelligent tutoring discipline. According to the literature review, more research is required on using DL algorithms, specifically convolutional neural networks, to predict students' academic performance. Previous studies have focused on using traditional machine learning algorithms, for instance, Naive Bayes (NB), Decision Tree (DT), Support Vector Machine (SVM), and Random Forest (RF), to predict performance. Despite some existing Deep Neural Network (DNN) to build the models, their experiments' results needed more consistent accuracy. We have identified a research gap: the absence of stud-

ies examining and assessing the effectiveness of different CNN structures within the student model of ITS. In this thesis, we explore the performance of various proposed architectures based on CNN to predict the student's performance.

1.2.2 Learning Classifier System and Reinforcement Learning for Pedagogical Model

The main goal is to increase the overall quality of the ITS teaching mechanism and improve the learning experience by providing students with the most suitable feedback and enhance their competence in problem-solving. Genetic algorithms and reinforcement learning have been effectively utilized in various sectors to optimize complex problems (Chen, Huang, Xu, Wang, & Wang, 2020; Albuquerque, Tarrataca, Brandão, & Coutinho, 2022). GAs can be used to evolve classifiers through the learning classifier system, while reinforcement learning guides the learning process by updating the Q-table values, allowing the RL agent to refine further and optimize the behavior of the classifiers.

Although many studies focus on integrating hybrid GA and RL, the research on using them to improve intelligent tutoring systems remains limited. Some previous studies solved the optimization problems by a combination of (GA-RL) in shop scheduling (Zhang & Cai, 2023), production environments (Chen et al., 2020) and satellite scheduling problems (Song et al., 2023), which are unrelated to the field of education and ITS. While other researchers applied genetic algorithms to the student model in ITS, and reinforcement learning was deployed to the instructor model without integrating them as a hybrid in the pedagogical model. This thesis fills this literature gap by identifying how the combination of GA and RL can enhance the pedagogical model in an intelligent tutoring system by building a

hybrid approach uses the strengths of both algorithms.

1.2.3 Deep Reinforcement Learning for Pedagogical Model

The use of deep reinforcement learning has enhanced ITS by acquiring optimum strategies to direct students along customized learning pathways. The pedagogical model in ITS can be tailored to meet the requirements of each student and improve the teaching experience. While the applications of DRL algorithms have been extensively studied and applied in various domains like game playing (Mnih, 2013), robotics (Ausin, 2021), and resource management (Mao, Alizadeh, Menache, & Kandula, 2016), their uses in intelligent tutoring systems are still mostly unexplored. Understanding the relative strengths and weaknesses of deep reinforcement algorithms, as well as their suitability for ITS, is crucial to unlocking their full potential in enhancing personalized and adaptive learning experiences. This research tries to fill the gap in research by exploring the effectiveness of DRL to enhance the pedagogical model in ITS and comparing the performance of these algorithms within an intelligent tutoring system context in detail, aiming to provide valuable insights that guide the development and deployment of DRL-powered ITS.

1.3 Research Objectives

The aim of this thesis is to improve the quality of ITSs by proposing new models using different machine learning algorithms. These models are adapted to the student behavior and designed to communicate directly with different students with various understanding and practical capabilities. The main objective of this work can be summarized as follows:

- To predict student academic performance by proposing a novel ap-

proach depending on deep learning algorithms, including the CNN.

- To compare between the proposed models based on CNN and other previous studies in the same field.
- To predict the best tactic\feedback from teachers\ITS to students by proposing a creative and distinctive hybrid approach that combines two techniques, a learning classifier system, and reinforcement learning. This integration leverages the strengths of both approaches and builds the pedagogical model in an ITS.
- To compare the performance of the proposed models (LCS, RL, and the hybrid model (PMCR), which is the integration between LCS and RL).
- To enhance the decision-making skills of the ITS by proposing different architectures based on deep reinforcement learning algorithms. This approach aims to create a pedagogical model within the ITS and identify the most efficient model for student learning.

1.4 Thesis Questions

We conducted three different types of experiments to answer five research questions in this thesis. The research questions are listed below:

- RQ1: What is the effect of CNN on student's performance evaluation?
- RQ2: What is the effect of the classifier system based on GA on pedagogical model's performance?
- RQ3: What is the effect of RL on pedagogical model's performance?
- RQ4: Does a hybrid approach of a classifier system based on GA and RL increase the performance of the pedagogical model?

- RQ5: Are different architectures of deep reinforcement learning have similar performance on pedagogical model, and whether their performances are better than those of the other previously stated approaches (classifier system, RL, hybrid of classifier system and RL)?

1.5 Thesis Structure

In addition to the first chapter, the research comprises five other chapters. The second chapter presents the most critical research papers on machine learning based on ITSs, ML and DL for student model, LCS, RL and DRL for pedagogical model. Chapter three explains the first experiment in this work, which focuses on predicting student academic performance using CNNs. Experiment 1 answers research question 1 (RQ1). Chapter four presents an adaptive pedagogical model based on the classifier system and reinforcement learning (PMCR) experiment, which addresses research questions 2, 3, and 4 (RQ2, RQ3, RQ4). Chapter five examines the third experiment, based on DRL in ITS, and provides an answer to research question 5 (RQ5). Chapter six presents the conclusion and recommendations.

1.6 Summary

An intelligent tutoring system has become crucial in education as it gives individualized instructional support to learners, facilitating problem-solving using suggestions and immediate feedback. ITS is one application of AI in the field of education. It provides an intelligent learning environment for learners independent of instructor interaction. This thesis presents various techniques of AI and machine learning to develop a new generation of ITS that assist students in gaining domain-specific intellectual information in a beneficial and effective way. We propose a novel approach for

student model depending on deep learning algorithms, including the CNN, to predict student performance. Moreover, we use LCS of GAs and RL to develop instructional policies for ITS. Also, we use a DRL architecture to build a pedagogical model that provides students with individualized learning experiences based on their own learning styles and preferences.

Chapter 2

Literature Review

2.1 Machine Learning Based Intelligent Tutoring Systems

2.2 Machine Learning for the Student Model

2.3 Genetic Algorithms and Reinforcement Learning for Pedagogical Model

2.4 Deep Reinforcement Learning for Pedagogical Model

Chapter 2

Literature Review

2.1 Machine Learning Based Intelligent Tutoring Systems

All of ITS's modules have been researched from the standpoint of artificial intelligence. The objective is to reduce the amount of knowledge engineering done at the development and to make the system more adaptive to the learner. Applied artificial intelligence, specifically machine learning techniques for different ITS modules, has been widely studied. Examples of ML techniques used in ITS are Artificial Neural Network (Venkatesh, Naganathan, & Maheswari, 2010), Fuzzy logic (Crockett, Latham, & Whitton, 2017), Bayesian Network (Santhi, Priya, & Nandhini, 2013) and clustering (Bouchet, Harley, Trevors, & Azevedo, 2013). More details on each technique are provided in the following sections and in Table 2.1.

Table 2.1: Some previous AI Techniques in Intelligent Tutoring Systems

Techniques	Author, year	System	Contributions	Result
Artificial Neural Network (ANN)	(Venkatesh et al., 2010)	ITDS	A framework incorporating a knowledge base student model with pedagogical perspectives	Reduces development times and the technical expertise needed for employees participating in the development of an auto-regulated intelligent tutoring conversation system.
	(Abu-Naser, 2012)	LP-ITS	Determine the learners' academic achievement degree	The accuracy of forecasting learner achievement is extremely good.
	(Bernard, Chang, Popescu, & Graf, 2015)	LSID-ANN	Identify students' learning styles.	Enhance student performance and shorten learning time.
Clustering	(Zakrzewska, 2008)	-	Gathering erroneous user input and analyzing it to identify common misunderstandings.	An ITS prototype for educating multi-column basic arithmetic to schoolchildren aged 5 to 10.
	(Šarić-Grgić, Grubišić, Šerić, & Robinson, 2020)	-	Students were clustered based on their online learning habits.	More adaptable scaffolding can be provided by an intelligent tutoring system or a human tutor.
Bayesian Networks	(Santhi et al., 2013)	-	Review of ITS using Bayesian Networks and how it can be used for efficient decision-making.	Assist learners in making progress in the learning process.
	(Alday, 2018)	-	Developed a model of ITS using BN to learn programming languages.	Improve student performance when using such systems; help instructors evaluate students.
	(Gamboa & Fred, 2002)	-	Proposed ITS using Bayesian Networks.	Implement an architecture of ITS to use it as an e-learning tool.
Fuzzy Logic	(Crockett et al., 2017)	Oscar CITS	Create a fuzzy prediction model based on the variables gathered during natural language conversation.	Improved the Oscar CITS's prediction accuracy.
Hybrid (BN + Fuzzy Logic)	(Eryılmaz & Adabashi, 2020)	(FB-ITS)	- To assist students in learning environments, a student model based on BN and fuzzy logic is being developed.	improved academic achievement, quickness in completing final exams, and academic success.
ANN with RL	(Fenza, Orciuoli, & Sampson, 2017)	-	Tutoring rules are taught by first analyzing the behaviors of human tutors and then monitoring how each student interacts inside a learning environment at various stages of the learning process at run-time.	Combining human tutors' knowledge with adaptation done by the same tutors while monitoring and evaluating student behavior
ANN + Fuzzy	(Cabada, Estrada, Hernández, & Bustillos, 2014)	-	Determine the following exercise to be completed by the learner, allowing for individual learning.	Assist kids in the 2 nd grade of primary school in improving their math skills.

2.1.1 ITS and Artificial Neural Networks

There have been numerous studies that applied ANNs to ITS architecture to improve the teaching process. For example, Venkatesh et al. (2010) proposed a framework incorporating a knowledge base student model with pedagogical perspectives. Their thesis was used to apply ITS through the use of supervised learning neural networks to assist students in effectively constructing skills. A study by Abu-Naser (2012) presented a technique in the field of learner models that apply ANN and expert systems to gain skills for the learner model (LP-ITS). Moreover, Bernard et al. (2015) conducted related research to identify students learning styles. The evaluation was conducted on 75 students, and it was discovered that using ANN improved the quality of learning and provided more reliable instruction to students, resulting in higher levels of learning success and less time needed to learn (Bernard et al., 2015). In addition, de Carvalho, de Melo, Flôres, Pires, and Loja (2020) applied a neural network to enhance the educator's guidelines and determine a learning methodology that is more adaptable and responsive to the student's profile.

2.1.2 ITS and Clustering

On the other hand, several studies have been conducted on data-driven student cluster based on online learning behavior. Gross, Mokbel, Hammer, and Pinkwart (2012) proposed strategies for providing feedback that are automatically grouped using machine learning approaches applied to groups of student results. Another study (Dogan & Camurcu, 2010) employed the k-means and fuzzy c-means algorithms to cluster the data available from an ITS, and the multidimensional data analysis findings were displayed to offer the instructor with pedagogically reliable feedback. Moreover, Bouchet et al. (2013) utilized a clustering technique based on data gathered from

106 undergraduate learners studying the circulatory mechanism to encourage self-regulated learning with MetaTutor. An ITS prototype was created to teach multi-column basic mathematics to children aged 5 to 10 years (Zakrzewska, 2008) . Their system was designed to collect incorrect user input and analyze it using an experimental clustering method to identify common misconceptions. Another study (Šarić-Grgić et al., 2020) clustered students based on their online learning behavior so, more adaptable scaffolding can be provided by an intelligent tutoring system or a human tutor.

2.1.3 ITS and Bayesian Networks

Several studies applying Bayesian Networks (BN) in building ITS. A study by Gamboa and Fred (2002), proposed a model for ITS using BN. A learner model, a knowledge base, an adaptation module, a pedagogical module, and a presentation module comprise the model's architecture. In 2013, Santhi et al. reviewed different intelligent tutoring systems called cognitive tutors using BN, and they presented how it could be used for effective decision making (Santhi et al., 2013). Furthermore, Alday (2018) developed a model for ITS to learn programming language using the concept of BN.

2.1.4 ITS and Fuzzy Logic

Crockett et al. in 2017 developed a novel technique using a fuzzy classification tree to predict and adapt to a student's learning style during tutoring interactions. The Conversational Intelligent Tutoring Device (CITS) attempts to imitate a human instructor to enhance the educational process by delivering lessons in natural language and adapting information to suit individual learning styles. The initial findings suggest that the model significantly enhanced the predictive accuracy of the CITS and identified strong correlations among these factors.

2.1.5 Hybrid ITSs

An interesting study by Dogan and Camurcu (2010) proposed an approach based on integrating an artificial neural network with a reinforcement learning algorithm. These strategies provide the learner with an intelligent assistant that can determine the appropriate level of challenge for the next task. Moreover, a study by Cabada et al. (2014) employed Intelligent tutoring systems for second-grade math lessons, utilizing a feed-forward neural network to determine the student's affective state and a fuzzy expert system that integrates cognitive student data with affective data to evaluate the student's emotional condition. This method allows the system to determine the difficulty of the following exercise, which would be solved by the student. In 2020, an experimental study by Eryılmaz and Adabashi (2020) to build ITS using combination of BN and fuzzy logic and called FB-ITS to enhance student learning has been presented. The research included 120 university level students. The study revealed that students who experimented with FB-ITS performed better academically on overall than students who studied with other methods.

2.2 Machine Learning for the Student Model

Many studies aim to improve student academic performance, analyze obstacles faced throughout the learning process, and provide ways to improve competencies. In other studies, finding failure/dropout criteria was challenging. They analyzed contexts to predict academic results and provide educational solutions. Another common purpose is adaptive learning, which enhances the quality of the educational environment and, ultimately, the learners' achievements. The personalization that occurs in adaptive educational software and Intelligent Tutoring Systems is built on the foundation of student profiles. The results of specific exercises relating to a topic are

used to evaluate a student's level of understanding. To do this, Singh and Vikram (2020) have developed a variety of model extensions throughout the years that include certain cognitive traits in the student profile. The suggested approach considers characteristics like the quality of questions and mistakes caused when practicing (Singh & Vikram, 2020).

A good survey by Hamim, Benabbou, and Sael (2021) presented the current state of the art in student profile modelling by employing machine learning strategies throughout the past four years. For various objectives, including failure, dropout, orientation, academic performance, etc., the study examined popular and effective machine learning approaches in traditional and online classrooms. The findings (Hamim et al., 2021) indicated that most research investigations employ decision trees because they are the most effective and widely used. Moreover, the essential traits utilized to create a student profile are academic, personal identification, and internet behavior. An experiment based on ML algorithms was conducted to reinforce the survey findings. The decision tree performed best, which supports the survey results. A recent study by Khan, Ahmad, Jabeur, and Mahdi (2021) evaluated the efficiency of machine learning algorithms for measuring students' educational progress and alerting tutors to students' challenges, which might improve learning outcomes. This research hypothesis's result includes supportive strategies to control students' progress from the course's beginning effectively and a preventative approach to providing struggling students with great attention. Four machine learning algorithms were selected to predict the performance of students (KNN, DT, NB, and ANN). The study proposed choosing a decision tree model as the best model for evaluation.

Furthermore, Ng, bin Mohd Azha, Yap, and Goh (2022) have proposed a data mining approach for identifying essential factors that affect student performance based on data from two secondary schools in Portugal. Several machine learning algorithms are used for classification: SVM, NB, and

Multilayer Perceptron (MLP). The highest performance was obtained from SVM, scoring 91%. In addition, research by Liu and Koedinger (2017) used data mining techniques to improve an Intelligent Tutoring System. The study results demonstrated that student learning outcomes are increased using the proposed system compared with the prior version.

A previous study by Imran, Latif, Mehmood, and Shah (2019) has built a prediction model using the ensemble method, which combines several models to increase the accuracy of student performance prediction rather than using a single model. In addition, Lincke, Jansen, Milrad, and Berge (2021) have recognized using ML algorithms to predict learning outcomes from student quiz answers and reading records from a Web based learning system. According to the study's findings, the difficulty of the question and the number of times it has been answered incorrectly are valuable parameters for determining whether a student's response is correct. Moreover, Ghorbani and Ghousi (2020) evaluated several resampling methods for dealing with imbalanced data while predicting student performance. They used different ML techniques, including SVM, ANN, Random Forest (RF), DT, and NB. The result indicates that the performance of classifiers was increased when using balanced datasets compared to imbalanced ones.

Many existing studies in the broader literature have investigated the ability of DL algorithms to enhance the performance of student prediction. DL has recently allowed researchers to have the ability to extract high-level features from raw data automatically, affecting student performance prediction. A study by Yousafzai et al. (2021) proposed a model to predict student performance from historical records using a Deep Neural Network (DNN) called Bidirectional Long Short-Term Memory (BiLSTM). The prediction accuracy of the developed model was 90.16%. Another exciting study (Smith, Min, Mott, & Lester, 2015) investigated the role of student drawing in learning. The authors introduced a diagrammatic student model

based on neural network architecture. Compared to competitive baseline techniques, it can predict student performance more precisely.

Seminal contributions have been made by Siddique et al. (2021). The study aimed to identify the essential aspects influencing secondary school student's performance and to develop an effective classification model for academic performance prediction by combining single and ensemble-based classifiers. Three single classifiers were examined individually, comprising a Multilayer Perceptron (MLP), J48 (DT), and PART, as well as three ensemble techniques, including Bagging, MultiBoost, and Voting. In addition, nine new models were created by combining single and ensemble classifiers to improve previous algorithm performance. MultiBoost with MLP outscored the others in the study, scoring 98.7% accuracy. According to the study (Ghorbani & Ghousi, 2020), the suggested model would effectively assess secondary school students' academic performance at an early stage to enhance academic achievement. Moreover, a study by S. Li and Liu (2021) proved that predicting student performance helps in course selection and developing suitable future study plans for learners. Furthermore, teachers and supervisors monitor students, give support, and implement training plans to achieve the best results. The proposed technique used a DNN to extract valuable data as a feature with appropriate weights. Neural networks use multiple hidden layers regulated by feed forwarding and backpropagation data from prior cases. The suggested model achieving the best prediction in MAE (0.593) and RMSE (0.785).

A study by Alhazmi and Sheneamer (2023) used clustering and classification techniques to analyze student performance at early stages. For the clustering, they applied a dimensionality reduction mechanism using the T-SNE algorithm with various factors, e.g., admission scores, first-level courses, and academic achievement tests, to investigate the relationship between these factors and students GPAs. For classification, they presented

experiments on different ML models (logistic regression, SVM, KNN, and RF). The study's findings indicate that educational systems have the potential to reduce the likelihood of student failures in their early stages. Moreover, X. Wang, Zhao, Li, and Ren (2023) proposed a novel student academic performance prediction framework (ProbSAP) that can efficiently extract more student characteristics from imbalanced datasets. ProbSAP consists of three components termed as collaborative data processing, scalable metadata clustering, and XGBoost-enhanced SAP prediction. The experimental findings showed that the promising accuracy and efficiency of ProbSAP on imbalanced datasets, in comparison to other state-of-the-art algorithms.

Many academics have utilized traditional machine learning algorithms to predict student academic achievement. Still, relatively limited studies have used convolutional neural networks' structure in the intelligent tutoring discipline. A survey by Poudyal, Mohammadi-Aragh, and Ball (2022) found a novel model that combines two 2D CNNs, outperforming standard baseline models (NB, DT, and logistic regression).

Many studies in the literature have examined the datasets utilized in this study. Two datasets were collected; the first was gathered and investigated by Cortez and Silva (2008b) of Portugal's University of Minho . It is available in two subjects from two Portuguese secondary schools: Mathematics and Portuguese language. The second dataset was provided by the Kalboard 360 learning management system. It is a three-class dataset where students are sorted into three grade levels: Low, Middle, and High, according to their total grade.

Several researchers applied various ML techniques to predict student performance using both datasets. For instance, a study by Imran et al. (2019) used supervised machine learning algorithms and ensemble methods and compared the results using the first dataset. The study presented in (Imran et al., 2019) highlights the importance of data preparation and

algorithm fine-tuning in addressing issues related to data quality. In this experiment, three supervised learning algorithms, namely J48, NNge, and MLP were used. The results indicate that J48 outperformed its competitors with a 95.78% accuracy rate. Furthermore, an ensemble approach is used to improve the precision of weak classifiers, and when compared to predictions from a single model, ensemble predictions are often more precise

Another study (Ng et al., 2022) used the same dataset with three classification models: SVM, NB, and MLP. The performance indicators were F1-Score, recall, accuracy, and precision. SVM achieved the highest accuracy either for the binary or five class classifications. In addition to the previous studies, a work by Hamoud (2016) used and evaluated the first dataset with three DT algorithms (J48, RepTree, and Hoeffding Trees). The results demonstrated that the J48 algorithm accurately classified and predicted students' intent to finish higher education and course progress.

On the other hand, several publications have been released documenting the use of ML and DL algorithms with the second dataset used in the current study. An improved model of student performance was created using a different set of behavioral characteristics (E. A. Amrieh, Hamtini, & Aljarah, 2015). This sort of feature is associated with learner interaction with an e-learning system. The study (E. A. Amrieh et al., 2015) examined data mining techniques such as ANN, NB, and DT classifiers to determine the effect of such features on student academic achievement. The findings showed a substantial link between learner behaviors and educational outcomes. Compared to the same dataset, results with various classification algorithms employing behavioral characteristics improved classification accuracy by up to 29%. One year later, in 2016, the same researchers (E. A. Amrieh, Hamtini, & Aljarah, 2016) used ensemble approaches, such as Bagging, Boosting, and RF, to increase the model's performance. The model's accuracy improved by up to 25.8% when using ensemble approaches. The model's

accuracy was greater than 80% when tested on new learners. This result validates the suggested model's dependability.

A proposed framework by Saleem, Ullah, Fakieh, and Kateb (2021) includes five machine-learning algorithms and four ensemble techniques: bagging, boosting, stacking, and voting. Using ensemble techniques, the model performance has improved significantly. Among alternative ensemble approaches, the stacking model excelled and achieved the highest F1 score (0.819) by integrating all five classifiers. The ML model integration enhanced the prediction accuracy and outperformed all other ensemble techniques. The suggested methodology can effectively evaluate student performance and assist instructors in making informed decisions.

Another study Vijayalakshmi and Venkatachalapathy (2019) developed a Deep Neural Network based student performance prediction system. After training and evaluating the model with Kaggle datasets using several techniques in R Programming such as Decision Tree (C5.0), NB, RF, SVM, KNN, and DNN, and compared the accuracy of all other algorithms. The result indicated that DNN exceeded the six different algorithms with an accuracy of 84%.

A work by Akour et al. (2020) explored the efficiency of using CNN to estimate students' performance and predict whether a student could complete their degree. The experimental findings showed that the suggested model outperformed the current methodologies regarding prediction accuracy. Limited research has been performed to determine the effectiveness of CNN in the student model, and this opens the avenues for us to explore the impact of various CNN architectures on the student model. Table 2.2 summarizes some previous studies in this field.

Table 2.2: Summary of the previous related work for the student model

Reference (Author, Year)	Techniques	Best Technique	Dataset Size	Evaluation Metrics
(E. A. Amrieh et al., 2015)	ANN, NB, DT	ANN	480	Accuracy, Precision, Recall, F-Measure
(E. A. Amrieh et al., 2016)	Ensemble approaches: Bagging, Boosting, RF	Ensemble	480	Accuracy, Precision, Recall, F-Measure
(Hamoud, 2016)	DT (J48, RepTree, Hoeffding Tree)	J48	1044	Accuracy
(Imran et al., 2019)	Ensemble (J48, Realada-boost)	Ensemble	1044	Accuracy
(Vijayalakshmi & Venkat- achalapathy, 2019)	Decision Tree (C5.0), NB, RF, SVM, KNN, DNN	DNN	500	Accuracy, Precision, Recall, F-score, ROC, RMSE
(Ghorbani & Ghousi, 2020)	RF	RF	1044	Accuracy
(Akour et al., 2020)	CNN	CNN	480	Accuracy, Precision, Recall, F-Measure
(Saleem et al., 2021)	ML with ensemble: BAG, boosting, stacking, VT	Stacking	480	F-Measure
(Hamim et al., 2021)	DT, KNN, SVM, NN, NB	DT	395	Accuracy
(Khan et al., 2021)	KNN, DT, MLP	DT	151	Accuracy, Precision, Recall, F-Measure, MCC
(Ng et al., 2022)	NB, MLP, SVM	SVM	1044	Accuracy, Precision, Recall, F-Measure
(Yousafzai et al., 2021)	CNN, Attention-based BiLSTM	BiLSTM	1044	Accuracy
(S. Li & Liu, 2021)	DNN	DNN	3,828,8	MAE, RMSE
(Poudyal et al., 2022)	CNN, KNN, NB, DT, Logistic regression	CNN	32,593	Accuracy

2.3 Genetic Algorithms and Reinforcement Learning for Pedagogical Model

There exists a considerable body of literature on the development, optimization, and personalization of ITS (Bueff, Papantonis, Simkute, & Belle, 2022), including the use of biometric data (Durães, Toala, Gonçalves, & Novais, 2019), incorporating traditional teaching and learning pedagogical theories (Abu-Dawwas, 2012), and using data-driven techniques to enhance and optimize the functions of intelligent tutoring systems (Koedinger, Brunskill, Baker, McLaughlin, & Stamper, 2013). Furthermore, other studies contribute to a better understanding of how ML can be applied to educational contexts and dealing with the pedagogical aspects, whether it is through modelling interactions (Katz, Aronis, & Creitz, 2000), personalizing interventions (Akyuz, 2020) or ensuring the explainability of ML-driven decisions in pedagogical settings (Kochmar et al., 2022).

2.3.1 Genetic Algorithms

A literature review found the benefit of utilizing genetic algorithms to enhance various aspects of the higher education process. These studies (Bellafkih, Azough, Bellafkih, & Bouyakhf, 2010) and (Hssina & Erritali, 2019) highlight the potential of these techniques to optimize learning experiences and help create more personalized and effective learning tailored to individual needs. A well-written review (Pillay, 2020) highlights the underutilization of GAs in the educational sector. It explored the impact of GAs on various educational aspects, such as intelligent tutoring systems, prediction of student performance, and the design of learning environments. The findings show that GAs can efficiently produce solutions within intelligent tutoring systems and assist in the design of learning environments. However, despite its advantages, the study highlights drawbacks to GAs, including high com-

putational costs, code complexity, and the need for specialized knowledge to execute it.

In addition, the study by Rasim, Rosmansyah, Langi, and Munir (2021) addressed the shortfall of immersive experiences in Intelligent Tutoring Systems (ITS) and proposed an Immersive Intelligent Tutoring (IIT) model using Bayesian Knowledge Tracing (BKT) and genetic algorithms (GA) within a virtual environment to enhance remedial learning. It used BKT for tailored learning paths and GA for dynamic question generation. The IIT model significantly outperformed traditional methods in learning outcomes, indicating its potential to make e-learning more effective and engaging.

However, these studies show that GA can effectively improve learning environments. Despite that, several important issues still need to be resolved, such as complexity, computational costs, generalizability, ethical issues, potential overfitting, and the requirement for additional study on long-term effects and broad user engagement (Vie et al., 2020).

2.3.2 Reinforcement Learning

On the other hand, several studies demonstrate the potential of reinforcement learning to produce effective and flexible learning environments customized to meet the learner's needs. Many studies use RL to enhance the pedagogical model in intelligent tutoring systems. A study by Korovesi and Ktona (2021b) used RL and the Deep Q-Network (DQN) algorithm to choose the right learning materials for the student intelligently. They created a new set of conditions, actions, and rewards to symbolize a student's learning journey inside the ITS.

In addition, a work by F. Wang (2018) introduced a new RL technique for partially observable Markov decision processes (POMDPs), that better accommodates for student learning pattern and behavior uncertainty than previous RL models. The outcomes showed that ITS, using POMDP and

RL, improves its instructional strategies in real time, signifying a major step toward more adaptable and tailored educational technology. Another study (Jasti, 2019) utilized RL to explore how different hint types (textual, diagrammatic, or video) affect the learning efficiency of students. On the other hand, a study (Koedinger et al., 2013) has explored the effectiveness of RL in ITS, including the education of students with autism—a condition that affects communication and social skills. ITS provides personalized education and evaluates student solutions. They update the student’s model through ongoing interactions.

Other studies extend the ITS enhancement theme by concentrating on advanced reinforcement learning (RL) techniques, such as hierarchical reinforcement learning (HRL) (G. Zhou, Azizsoltani, Ausin, Barnes, & Chi, 2019). Adaptive and Intelligent Educational Systems (AIES) with RL have been studied by several authors in the literature, with a focus on improving the learning process and customizing teaching approaches. Using the RL model, the system can learn how to give personalized instructions to each student based on their past interactions with students with similar profiles (A. Iglesias, Martinez, & Fernández, 2003). This method enables the system to create strategies for teaching content, timing, and methods without predefined rules. The study investigates the effectiveness of the RL approach in a DataBase Design (DBD) educational system by evaluating how many students the system needs to interact with before it can establish efficient teaching strategies. With the same idea, another study (A. Iglesias, Martínez, Aler, & Fernández, 2009) explored three crucial dimensions of the learning effectiveness of the educational system: the process of improving policies for teaching, the effect of striking a balance between exploitation and exploration when using RL with AIES, and a plan to accelerate AIES’s initial training. The results of the experiments showed that the system with RL enhancements can gradually learn to provide instruction properly, which

implies a possible path toward improving the flexibility and customization of web-based learning programs. Iglesias et al. presented empirical findings showing that AIES can receive an exact initial pedagogical policy from a value function learned with simulated students. Based on the interaction of over 70 undergraduate computer science students, the evaluation shows that an effective and practical guide through the course materials is obtained.

Another application of reinforcement learning is a tutoring robot, which can teach children according to their learning needs (Ramachandran & Scassellati, 2014). Using RL, the robot can adjust the sequence of educational content based on the child's performance, aiming to optimize learning outcomes and maintain the child's focus. The study proposed an organized assessment process that tracks children's progress through initial assessments and follow-up tests. Additionally, the study (Ramachandran & Scassellati, 2014) aims to monitor the children's level of engagement by utilizing sensors that record data during interactions. Furthermore, a study by Bassen et al. (2020) tested the effectiveness of the RL-driven approach on an online course with over 1,000 students. The findings indicated that the adaptive scheduling system improved learning outcomes, reduced dropout rates, and decreased the number of activities students needed to complete, pointing to a streamlined and more effective educational experience. In 2021, a study introduced RL Tutor (Kubotani, Fukuhara, & Morishima, 2021), which used a virtual student model to optimize teaching strategies with minimal interactions. An experiment showed that it achieves similar or better educational outcomes than traditional methods with fewer interactions. This system could bridge the gap between theoretical optimization and practical application in digital learning.

2.3.3 Integrated Approaches Using Genetic Algorithm and Reinforcement Learning

Several research utilized genetic algorithms with reinforcement learning to solve complex optimization problems. Good work by H. T. Lin, Lee, and Hsiao (2015) tackled the challenge of optimizing tutorial tactics for ITS to enhance student learning gains without depending on extensive pre-existing datasets. A genetic-based reinforcement learning (GBML) approach was proposed to combine a genetic algorithm with reinforcement learning. RL is applied to help the ITS learn from interactions with students by receiving feedback through rewards, which guide the system towards strategies that enhance student learning outcomes. The GA identifies and evolves rules that control the ITS's tutorial tactics. It generates new rules through genetic operations such as mutation, crossover, and selection based on their performance, which is evaluated through the RL reward mechanism. As more data is gathered, the research validates that online tactic optimization is feasible and promising for ITS, improving the system's ability to adapt and customize learning experiences.

Many authors have recognized the use of genetic algorithms to aid reinforcement learning in learning behaviors and tuning parameters for better performance. A paper by R. Iglesias, Rodríguez, Regueiro, Correa, and Barro (2006) proposed a technique combining GA with RL to focus and enhance exploration, which will speed up learning without requiring outside information or guidance. The system's exploration is guided by defining a suitable reinforcement function and using a modified RL algorithm with genetic algorithms, which shortens learning time and increases stability. The methodology is applied to a mobile robotics context, showing its effectiveness through the rapid and stable convergence to an optimal policy. Furthermore, a work by Kryvenchuk, Petrenko, Cichon, Malynovskyy, and Helzhynska (2022) addressed the problem of selecting the best hyperparam-

eters for neural networks in DRL, which is usually a laborious procedure that greatly depends on experts. Their methodology (Kryvenchuk et al., 2022) involves using a genetic algorithm to evaluate different network models against a set of parameters representing potential environmental states and actions. The results demonstrate that the GA can successfully start with a low efficiency model template and improve iteratively to reach good overall performance.

Although many studies focus on integrating hybrid GA and RL, the research on using them to improve intelligent tutoring systems remains limited. Some previous studies solved the optimization problems by a combination of (GA-RL) in shop scheduling (Zhang & Cai, 2023), production environments (Chen et al., 2020), satellite scheduling problems (Song et al., 2023), and Image Processing (Boukthir, Haddad, Hamdani, & Alimi, 2022) which are unrelated to the field of education and ITS. While other researchers applied genetic algorithms to the student model in ITS, and reinforcement learning was deployed to the instructor model without integrating them as a hybrid in the pedagogical model. This paper fills this literature gap by identifying how the combination of GA and RL can enhance the pedagogical model in an intelligent tutoring system.

2.4 Deep Reinforcement Learning for Pedagogical Model

Artificial intelligence has achieved significant advancements in reinforcement learning throughout the past century. The 1990s saw the introduction of deep learning and the subsequent breakthrough of convolutional networks in computer vision. Early in the 2000s, researchers combined deep learning neural networks with reinforcement learning to create a comprehensive framework known as deep reinforcement learning. This framework has sig-

nificantly advanced the capabilities of human-level agents and autonomous systems. A large and growing body of literature has investigated the effect of deep reinforcement learning in intelligent systems, games, robotics, and natural language processing.

There have been several comprehensive reviews published on deep reinforcement learning. For instance, (Mousavi, Schukat, & Howley, 2018) provided a summary of various early RL algorithms with an overview of fundamental and contemporary issues in reinforcement learning, covering core elements, important mechanisms, and applications. A survey by X. Wang et al. (2022) classified the existing deep RL algorithms into three categories: model-based methods, model-free methods, and advanced RL methods. They examined in detail some recent developments, such as exploration, inverse RL, and transfer RL. In addition, a study by Fenjiro and Benbrahim (2018) delved into the development timeline of both reinforcement learning and deep learning technologies, highlighting significant advancements in these fields. Additionally, it discussed DRL's current challenges and real-world applications, as well as the hardware and frameworks used. In 2017, Arulkumaran, Deisenroth, Brundage, and Bharath (2017) provided a comprehensive overview of the field, highlighting the potential of deep reinforcement learning in visual understanding and its application in robotics.

A considerable amount of literature has explored the potential of deep reinforcement learning in education. Reddy, Levine, and Dragan (2017) and X. Li, Xu, Zhang, and Chang (2023) demonstrated the effectiveness of DRL in adaptive learning systems. It assists in creating individualized learning plans and optimizing student learning material selection. Recently, Ruan et al. (2024) supported these findings, showing that DRL could give adaptive instructional assistance, particularly for lower-performing students.

A range of studies have explored the use of deep reinforcement learning in intelligent tutoring systems (Ausin, 2019; Subramanian & Mostow,

2021; Abdelshiheed, Hostetter, Barnes, & Chi, 2023). Ausin (2019) and Abdelshiheed et al. (2023) highlight the potential of DRL in inducing pedagogical policies and provided metacognitive interventions, respectively. In Ausin's work, a credit assignment problem was addressed using DRL with gaussian processes, and adaptive pedagogical tactics were induced using the DQN and Double DQN algorithms. The main findings showed that when combining the DQN policy with inferred rewards, it outperformed the random policy, especially for educators with high pre-test scores. While Abdelshiheed et al. compared the effectiveness of metacognitive interventions for student learning using a Random Forest Classifier (RFC) for static interventions and deep reinforcement learning for adaptive interventions on ITS. The result shows that, despite what the RFC said, the DRL method offered adaptive interventions by using Double DQN to avoid overestimation issues and keep the focus on metacognitive interventions. In 2021, Subramanian and Mostow used DRL to train pedagogical policies in ITS, achieving higher learning gains than traditional methods. Furthermore, Paduraru, Paduraru, and Iordache (2022) developed a framework, the Agent to Human Recommender System (AHRec), to construct ITS using DRL techniques. Using texts, images, and arrows, this framework can provide students with appropriate online suggestions from their teacher agents.

Other studies focus on exploration strategies and the use of a tutor-student network to improve system performance. Koroveshi and Ktona (2021a) focus on the ITS's pedagogical module by investigating different exploration strategies in RL and comparing their performance during training and testing phases. Their study (Koroveshi & Ktona, 2021a) explored four methods: random, greedy, epsilon-greedy, and Boltzmann. According to the findings, the Boltzmann policy outperformed all other strategies during the testing phase, showing efficient learning and decision-making. One issue with using RL is that it requires a large number of iterations and data

for effective training. On the other hand, Zeng, Wang, and Ge (2020), introduced a tutor-guided policy that consists of tutor and student modules to accelerate deep reinforcement learning. Their paper focuses on the following algorithms: deep Q-learning, deep deterministic policy gradient, asynchronous advantage actor-critic, trust region policy optimization, and proximal policy optimization. This approach (Zeng et al., 2020) improves the ability to navigate by offering more information, resulting in accelerated learning and higher rewards.

Another study by Ausin (2019) focuses on assessing the impact of offline DRL on promoting pedagogical policies in intelligent tutoring systems. Its goal is to improve DRL algorithms so that they work better in educational environments. It tackled challenges such as delayed and noisy rewards, enhanced communication between tutors and students, and improved policy generalizability and transferability.

A number of authors have explored the integration of genetic algorithms into deep reinforcement learning. In 2019, research by Sehgal, La, Louis, and Nguyen utilized a GA to optimize parameters in the deep deterministic policy gradient algorithm, leading to improved performance and faster learning in robotic manipulation tasks. In 2022, Kryvenchuk et al. focus on the selection of network hyperparameters and model design, showcasing the algorithm's ability to enhance model performance and decrease development time. These works collectively highlighted the potential of genetic algorithms in improving the effectiveness of deep reinforcement learning systems.

In short, prior research has shown the effectiveness of deep reinforcement learning algorithms in various domains, like education. Moreover, most researchers confirmed the ability of DRL to build an intelligent tutoring system. However, the major contribution of deploying DRL in the field of ITS has not yet been fully discovered, and there has been a lack

of research on developing an effective pedagogical strategy that enhances students' learning efficiency. This study fills this literature gap by exploring the effectiveness of DRL algorithms in producing personalized and adaptive instruction from teachers/ITS for their students.

Chapter 3

Convolutional Neural Network for Predicting Student Academic Performance in Intelligent Tutoring System

3.1 Introduction

3.2 Overview of CNN

3.3 The proposed Model

3.4 Dataset Description

3.5 Model Evaluation

3.6 Experimental Results

3.7 Discussion

3.8 Summary

Chapter 3

Convolutional Neural Network for Predicting Student Academic Performance in Intelligent Tutoring System

3.1 Introduction

Education plays a crucial role in the growth of any country, and it is a key factor in achieving success in life. Academic institutions strive to provide their learners with a high quality education to enhance learning (Ramaphosa, Zuva, & Kwuimi, 2018). Students' academic achievement is a vital part that determines the success of any educational institution. Currently, these organizations have significant difficulty providing quality learning for their learners while improving their success rate. By integrating AI technology in education, these organizations aim to personalize educational experiences,

making learning more efficient and effective for each student.

Researchers from many disciplines propose various techniques and approaches for predicting student performance. They also improve current approaches to increase prediction outcomes. Among all disciplines, machine learning plays a critical role in building models used for prediction purposes. It is a helpful tool for taking early steps to improve student performance and minimize failures at the end of the course. ITS, based on AI and ML, are examples of tools that improve teaching abilities, assist students in learning, and may engage students in dialogues and provide feedback (Asthana & Hazela, 2020).

Student modeling is the central component of an ITS and a broad research field. Student profile modeling (Hamim et al., 2021) is based on a learner profile description that combines the crucial features and provides the student's most coherent, complete, and operational picture. Background knowledge, learning preferences, behaviors, talents, objectives, and so on are all student characteristics. The student profile model may be built by analyzing data from many places (e.g., online learning platforms, social media, and school records). Indeed, some studies have utilized the student profile to suggest adaptive learning, advise them on academic choices (Abyaa, Idrissi, & Bennani, 2016). ML techniques are employed to derive meaningful insights from data for the purposes of informed decision-making in student profile modeling. Student prediction models are very useful to ITS, as knowing the level and performance of the student will help the ITS to know how to select the topics in terms of level of difficulty, knowledge grasping or even how to present the topics to the student.

3.2 Overview of CNN

The structure of CNN consists of three main layers (Yamashita, Nishio, Do, & Togashi, 2018): convolution, pooling, and fully connected layers. The feature extraction is carried out by the first two layers (convolution and pooling). The final output is mapped onto the extracted features by the third layer (fully connected layer). More descriptions about these layers in the following:

1. **Convolution layer:** A convolution layer is a central part of the CNN structure. The main purpose is to extract features from data. The essential idea is that a kernel, a small array of numbers, generates a feature map by taking the element-wise product between the kernel and input tensor and summing up the results to generate the output value.
2. **Pooling layer:** A pooling layer handles a conventional down sampling process and reduces the dimensionality of feature maps. Max pooling is the most commonly used type of pooling, which selects patches and outputs the patch's maximum value.
3. **Fully connected layers:** Fully connected layers are also called dense layers, where each input neuron is associated with each output neuron in the prior layer. In most cases, the output of the last convolution or pooling layer is converted into one flatten dimension array. A subset of fully connected layers maps the features to the network's final outputs. The number of output nodes in the final dense layer typically equals the number of classes. A nonlinear function, for example, rectified linear unit (ReLU), is followed by each fully connected layer.

Figure 3.1 shows the basic structure of CNN layers.

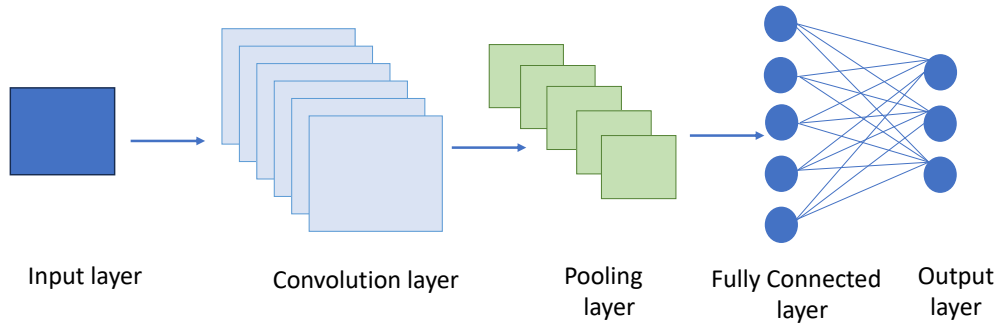


Figure 3.1: Basic CNN structure

3.3 The proposed Model

The literature review indicates a need for additional research on the use of DL algorithms, specifically CNN, to predict students' academic performance. Previous studies have focused on using traditional machine learning algorithms, for instance, NB, DT, SVM, RF, and KNN, to predict performance. We have identified a research gap: the absence of studies examining and assessing the effectiveness of different CNN structures within the student model of ITS. To overcome these research gaps, we proposed three different models based on CNN architecture using two educational datasets. We believe that the work we conducted in this thesis will significantly contribute to the field of deep learning and ITS.

3.3.1 Environment

Three CNN models were built to predict student's performance. We used an interactive computing data science platform, Jupyter Notebook. It is a flexible tool to create and configure machine learning workflows. Furthermore, we created the models using Python 3 and TensorFlow 2.0. Python is a high-level programming language used for scientific research and computation. Python is home to multiple open-source and general-purpose ML libraries used to train DL models. TensorFlow is an open-source machine learning

library for DNN training and inference. It employs data flow graphs for numerical computation. A developer must first create the input layer and link each input layer to the hidden layer, followed by a similar process from the hidden layer to the output layer. The graphs comprise tensors and operations, which define all the neural networks and mathematical calculations. TensorFlow has a Graphical Processing Unit package where all matrix calculations may be performed (Yousafzai et al., 2021).

3.3.2 The Architecture of the Proposed Model

As depicted in Figure 3.2 the proposed model structure is composed of eight distinct stages. The initial stage involves the acquisition of the datasets gathered for the study. The use of robust, high-quality datasets is essential to ensure the accuracy and reliability of the model. More explanation about the datasets and their attributes is shown in Sections 3.4.1 and 3.4.2. The second stage is to preprocess the dataset and improve its quality which is discussed in Section 3.4.3. The next stage is to create a training set consisting of (70%) of the data and a testing set consisting of (30%) of the data. The attributes and classes in the training dataset are separated and saved in a TensorFlow. The process of extracting suitable features from the data to improve the performance of the models arises in the fourth stage. Two types of feature selection were used Decision Trees (DT) and Principal Component Analysis (PCA) to figure out which one might give better results, this is demonstrated clearly in section 3.4.4. The fifth stage is building a convolutional neural network. Three different models were developed to explore the best architecture that gives the highest accuracy. The first model (Model 1) used CNN as input, the second model (Model 2) used fully connected layers, followed by CNN, and the third model (Model 3) used CNN, followed by fully connected layers, followed by CNN. Section 3.3.3 provides additional details about these layers. The evaluation of the models using some evalua-

tion metrics defined in section 3.5. Section 3.6 presents experimental results obtained from predicting student performance. A comparison between the proposed models and other previous studies is shown in sections 3.6.2 and 3.6.4.

3.3.3 Model Creation

The planned architecture is divided into three different models. We used different epochs for all the built models: 10, 50, 100, and 200. The details regarding the proposed model can be found in the following subsections. The structure of each model includes details about the layers, the name and type of all layers in the model, their placement within the model, the output shapes of each layer, the number of parameters (weights) in each layer, and the overall number of trainable and non-trainable parameters of the model. Figures 3.3, 3.4, 3.5 summaries the layers of each model.

1) Model 1: CNN with one dimension as the input

Model 1 architecture comprises five layers: an input layer (Conv1D), a pooling layer, a flatten, and 2 dense layers, as shown in Figure 3.3. The first layer is a convolutional layer with input shape contains the batch size, the number of features and the number of channels. The second layer is a pooling layer called MaxPooling1D, using a length and stride of 2, which splits the size of the convolutional layer's feature maps in half. The next layer is called Flatten. It enables the processing of the output by traditional, fully connected layers. The following layer used a fully connected layer with 64 neurons and a rectifier activation function. The last layer is the output layer, which has three neurons representing the three categories' classes and a Soft-Max activation function. The optimization technique Adaptive Moment Estimation (Adam) is applied to calculate the adaptive momen-

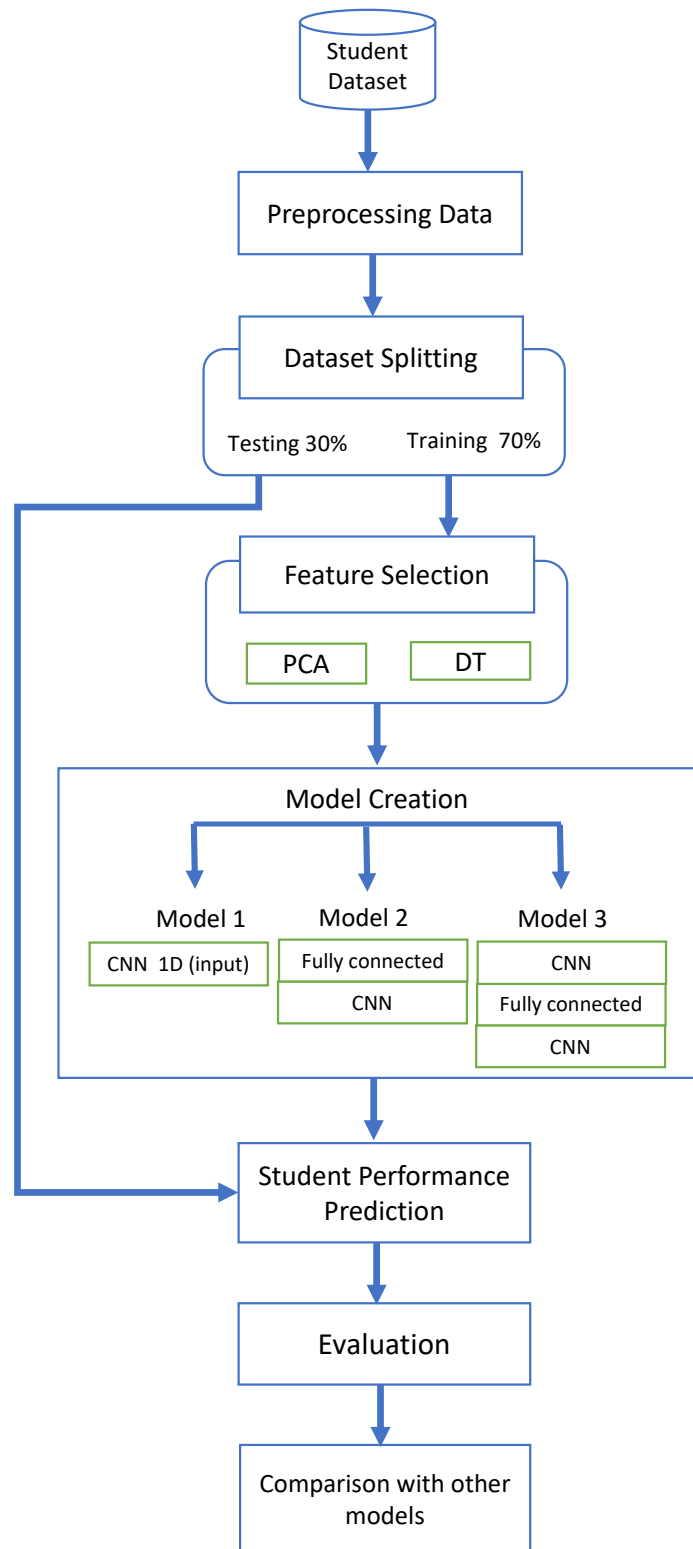


Figure 3.2: The structure of the proposed model

tum value. The error with sparse categorical cross entropy is used as a loss function. We used a batch size of 32 and a different number of epochs 10,50,100, and 200 to evaluate the performance of the models either decreasing or increasing with changes in the number of epochs. Table 3.1 shows the description layers of Model 1 for the first

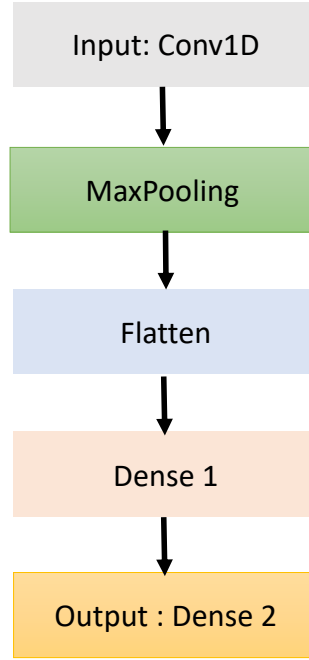


Figure 3.3: Model 1 structure

dataset (PTdataset) used in this study. The input and output shapes of the models depend on the datasets used. Note that the (None) values in the table indicate that the model can accommodate a variable batch size. The input shape of the Conv1D is (32, 58, 1). 32 is the batch size, 58 is the number of features after a preprocessed PTdataset, and 1 is the number of channels. The output shape is (None, 58, 32), which will be the input to the MaxPooling1D layer. The pooling layer produces an output with a shape of (None, 29, 32). Then the flatten layer results in 928 features. The next layer is a fully connected layer with 64 neurons. The final output layer has three neurons representing a multi-class classification with 3 classes.

Table 3.1: Model 1 layer's description (PTdataset)

Layer (type)	Output Shape	Param#
conv1d (Conv1D)	(None, 58, 32)	128
maxpooling1d (MaxPooling1D)	(None, 29, 32)	0
flatten (Flatten)	(None, 928)	0
dense (Dense)	(None, 64)	59456
dense1 (Dense)	(None, 3)	195
Total params: 59,779		
Trainable params: 59,779		
Non-trainable params: 0		

2) Model 2: Fully connected, followed by CNN

The second model starts with fully connected layers, followed by CNN. This approach will allow us to analyze how dimensionality reduction techniques like PCA and DT impact the model's performance and accuracy when the model starts with a fully connected layer instead of CNN. By comparing these results to those obtained from the CNN-based model, we hope to gain deeper insights into the advantages and limitations of each architecture.

The first layer is the input layer with input shape (batch size, the number of features and number of channels). Followed by three fully connected layers, each with 64 neurons with ReLU as the activation layer. The next layer is called Conv1D, which is a convolutional layer with 16 feature maps of size 3×3 . The following layer is a max pooling layer, which takes the max over 2×2 patches. The next layer is a second convolutional layer with 16 feature maps of size 3×3 . After the convolutional layer, a max pooling layer was added with a pool size of 2×2 . The last hidden layer is a the flatten layer. The output

layer has 3 neurons corresponding to the 3 output classes. For the hidden layers, we used the ReLU function, and for the output layer, the SoftMax function. Figure 3.4, illustrates the structure of Model 2, and Table 3.2, summaries the model layers and number of parameters in each layer for PTdataset.

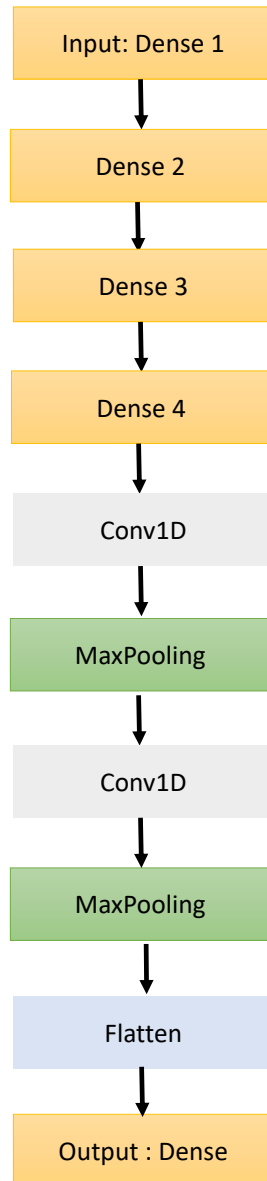


Figure 3.4: Model 2 structure

Table 3.2: Model 2 layer's description (PTdataset)

Layer (type)	Output Shape	Param#
dense1 (Dense)	(None, 58, 32)	64
dense2 (Dense)	(None, 58, 64)	2112
dense3 (Dense)	(None, 58, 64)	4160
dense4 (Dense)	(None, 58, 64)	4160
conv1d1 (Conv1D)	(None, 56, 16)	3088
maxpooling1d1 (MaxPooling)	(None, 28, 16)	0
conv1d2 (Conv1D)	(None, 26, 16)	784
maxpooling1d2 (MaxPooling)	(None, 13, 16)	0
flatten1 (Flatten)	(None, 208)	0
dense5 (Dense)	(None, 3)	627
Total params: 14,995		
Trainable params: 14,995		
Non-trainable params: 0		

3) Model 3: CNN, followed by Fully connected, followed by CNN

Model 3 consists of 14 layers: an input layer, two convolutional layers, three fully connected layers, two more convolutional layers, a flatten layer, and the last, the output layer. The Model starts with the input layer with shape (32, 58, 1). The output shape from the input layer is (58,32) with 64 parameters. The following layer is a one-dimension convolutional layer with 16 filters of size 3×3 and ReLU as an activation function. Then, a pooling layer of pool size of 2×2 . Next, a second convolutional layer with 16 filters of size 3×3 followed by a max pooling layer size of 2×2 . The output shape from the convolutional layers is (15,16). After that, three fully connected layers with different numbers of neurons 32, 16, and 16, respectively, with ReLU activation function, are added to the model. The next layer is a one-dimension convolutional layer with 16 neurons of size 3×3 followed by a max pooling layer. In addition, another one-dimension

Table 3.3: Model 3 layer's description (PTdataset)

Layer (type)	Output Shape	Param#
dense1 (Dense)	(None, 58, 32)	64
conv1d1 (Conv1D)	(None, 58, 16)	1552
maxpooling1d1 (MaxPooling)	(None, 29, 16)	0
conv1d2 (Conv1D)	(None, 29, 16)	784
maxpooling1d2 (MaxPooling)	(None, 15, 16)	0
Dense2 (Dense)	(None, 15, 32)	544
dense3 (Dense)	(None, 15, 16)	528
dense4 (Dense)	(None, 15, 16)	272
conv1d3 (Conv1D)	(None, 15, 16)	784
maxpooling1d3 (MaxPooling)	(None, 8, 16)	0
conv1d4 (Conv1D)	(None, 8, 16)	784
maxpooling1d4 (MaxPooling)	(None, 4, 16)	0
flatten (Flatten)	(None, 64)	0
dense5 (Dense)	(None, 3)	195
Total params: 5,507 Trainable params: 5,507 Non-trainable params: 0		

convolutional layer followed by a pooling layer are added. The final layer is the output layer which, has 3 neurons and SoftMax as the activation function. For all the hidden layers in the model, we used the ReLU function. Figure 3.5 summarize the structure of Model 3, and Table 3.3 show the description of layers using PTdataset.

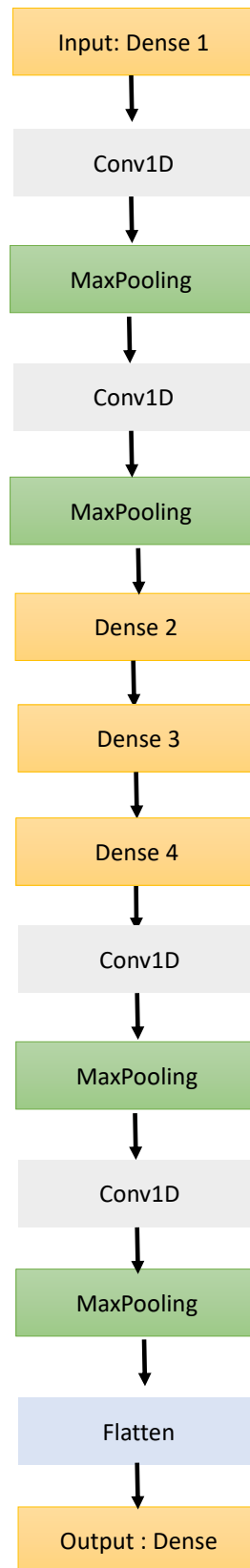


Figure 3.5: Model 3 structure

3.4 Dataset Description

3.4.1 PTdataset Description

This dataset approaches student performance in two Portuguese secondary schools. Two datasets are available in two subjects: Mathematics and Portuguese language. It was retrieved from the website of the UCI machine learning repository (Cortez & Silva, 2008a). The data was gathered via school reports and surveys, including information about a student's grade, demographics, and social background. It is a three-class dataset (Good, Fair, and Bad) based on the final grade. The dataset consists of 1044 observations (649 instances for Portuguese subject and 395 instances for Math subject) with 33 attributes. More details about the dataset used, and its attributes are presented in Table 3.4 and Table 3.5.

Table 3.4: PTdataset Description

Dataset Features	Multivariate
Attribute Types	Integer
Related Tasks	Classification
No. of Instances	1044 (649 and 395)
No. of Attributes	33
Missing Values?	N/A
Domain	Social
Date Donated	27-11-2014

3.4.2 XAPIdataset Description

This educational dataset was collected from the Kalboard 360 learning management system (LMS) and obtained from kaggle (E. Amrieh, Hamtini, & Aljarah, 2016). The dataset contains 480 instances and 16 attributes. The attributes are divided into three primary categories: (a) Demographic char-

Table 3.5: Attributes information of PTdataset

No	Attribute	Attribute Description	Type	Range of Values
1	school	School name	binary	GP or MS
2	gender	Students' gender	binary	F or M
3	age	Students' age	numeric	15, 16, 17, 18, 19, 20, 21, 22
4	address	Home address type	binary	Urban (U) or Rural (R)
5	famsize	Size of family	binary	LE3 if < 3 or GT3 > 3
6	famsup	Family educat. assistance	binary	yes or no
7	M_educ	Education of mother	numeric	No school (0) Elementary schooling up to the fourth year (1) Grades 5 through 9 (2) Secondary schooling (3) University level education (4)
8	F_educ	Education of father	numeric	No school (0) Elementary schooling up to the fourth year (1) Grades 5 through 9 (2) Secondary schooling (3) University level education (4)
9	M_job	Mother's work	nominal	schoolteacher, medical, public service, in the house, other
10	F_job	Father's work	nominal	schoolteacher, medical, public service, in the house, other
11	schoolsup	Further educt. assistance	binary	yes or no
12	net	Home Net access	binary	yes or no
13	traveltime	Time (home-school)	numeric	<15m (m for minute) 15m to 30m 30m to 1h (h for hours), >1 h
14	nursery	Went to nursery school	binary	yes or no
15	failures	Previous course failures	numeric	n if $1 \leq n \leq 3$, else 4
16	reason	Chosen school due to	nominal	near to home, school status, course preferences, or other
17	Pstatus	Cohabitation status of parents	binary	(T) live with parents or (A) not living together
18	health	Present state of health	numeric	1 (extremely poor) to 5 (extremely high)
19	activities	Activities outside of the classroom	binary	yes or no
20	studytime	Weekly study time	numeric	<2 h (h for hours), 2 - 5 h, 5 - 10 h, >10 h
21	higher	Intends to pursue further education	binary	yes or no
22	stguard	Guardian of the learner	nominal	mother, father, other
23	romantic	Involving a relationship	binary	yes or no
24	famrel	Family relationship quality	numeric	1 (extremely poor) to 5 (extremely high)
25	freetime	After-school free hours	numeric	1 (extremely poor) to 5 (extremely high)
26	goout	Interacting out with friends	numeric	1 (extremely poor) to 5 (extremely high)
27	Dalc	Alcohol usage at workday	numeric	1 (extremely poor) to 5 (extremely high)
28	Walc	Alcohol drinking throughout the weekend	numeric	1 (extremely poor) to 5 (extremely high)
29	paid	Additional paid courses within the course	binary	yes or no
30	absences	Amount of absences from school	numeric	0 to 93
31	G1	First semester score	numeric	0 to 20
32	G2	Second semester score	numeric	0 to 20
33	G3	Final score	numeric	0 to 20, output target

acteristics such as nationality and gender. (b) Educational background characteristics include educational stage, grades, and section. (c) Behavioral characteristics include raising hands in class, opening resources, responding to parent surveys, and school satisfaction. It is a three-class dataset in which learners are assigned to one of three numeric intervals based on their grades (Low-Level, Middle- Level, High-Level). The description and the attributes information of xAPIdataset are shown in Table 3.6 and Table 3.7.

Table 3.6: xAPIdataset Description

Dataset Features	Multivariate
Attribute Types	Integer/Categorical
Related Tasks	Classification
No. of Instances	480
No. of Attributes	16
Missing Values?	N/A
Domain	Education and Data Mining
Date Donated	8-11-2016

3.4.3 Data preprocessing

Data preprocessing is necessary after data collection to enhance dataset quality. Data preparation includes data attribute selection, cleansing, transformation, and reduction. It contributes to the process of knowledge discovery. Data transformation is described as the technical process of transforming data from one format, standard, or structure to another without affecting the content of the data (Fan, Chen, Wang, Wang, & Huang, 2021). It can be evaluated to help decision-making processes and improve data quality. Both datasets were transformed into a numeric format. The attributes of PTdataset student’s school, sex, address type, family size, parents’ status, school support, family support, activities, nursery school, higher education, home internet, and romantic relationship were converted to binary ‘0’ and

Table 3.7: Attributes information of xAPIdataset

No	Attribute Name	Description	Type	Range of Values
1	Gender	Student sex	nominal	M or F
2	Placeofbirth	Place of Birth	nominal	LB, EG, KSA, USA, JOR, VEN, IRAN, TUN, SYR, LYB, KUW, MOR, IRQ, PAL
3	Nationality	Student's nationality	nominal	LB, EG, KSA, USA, JOR, VEN, IRAN, TUN, SYR, LYB, KUW, MOR, IRQ, PAL
4	Topic	Course subject	nominal	Eng, Span, Fren, Ar, IT, Math, Chem, Bio, Sci, His, Quran, Geo
5	Semester	School year semester	nominal	1st or 2nd
6	StageID	Educational level	nominal	L (lower), M (middle), H (high)
7	Relation	Responsibility of the student	nominal	Mother or Father
8	Raisedhands	how often does a student raise their hand in class	numeric	0 - 100
9	SectionID	Student class	nominal	A, B, C
10	GradeID	Student grades	nominal	From G1 to G12
11	VisITedResources	how frequently a student accesses course material	numeric	0 - 100
12	AnnouncView	how often the student looks at the most recent announcements	numeric	0 - 100
13	Discgroups	how frequently a student takes part in discussion groups	numeric	0 - 100
14	PAnsweringSurvey	if the parent responded to the questionnaires supplied by the school	nominal	Yes or No
15	PschoolSatisfi	how satisfied parents are with the school	nominal	Yes or No
16	StudentAbsenDays	Students' absences	nominal	more than 7, less than 7
17	Class	Student level	nominal	L, M, H

'1'. Other nominal attributes include the mother and father's jobs, the reason for choosing this school, and the student's guardians' transition into a numerical data type. Additionally, integer numbers were created from the output feature. Good is equal to '2', Fair to '1', and Poor to '0'. The second dataset's characteristics, such as the gender of the student, the responsible parents, school year semester, parent responding to questioners, and parent satisfaction, are also transformed into binary data, '0' and '1'. Country of origin, birthplace, level of education, grade of the student, student classroom, and course subject are additional nominal data type features that have

been converted to numerical data types.

We noticed a highly imbalanced dataset when we performed the discretization procedure to PTdataset, and the distribution of the students' class labels was inconsistent. The class label contains much more instances for the class "Fair" but fewer samples for classes "Good" and "Poor". Figure 3.6 illustrates the issue with a significantly imbalanced dataset. The class labels (Good, Fair, Poor) had corresponding distributions of 20%, 65%, and 15%, respectively. This serious difficulty manifests in categorization issues and decreases the model's performance. One of the most crucial aspects of enhancing the models' performance is addressing the issue of an imbalanced dataset. Due to this problem, the dominant class dominates the minority class. As a result, the classifier's performance is unreliable, and they frequently fall into the majority class (Ghorbani & Ghousi, 2020). Therefore, we must find a solution to this issue as it might provide incorrect outcomes (Kotsiantis, Kanellopoulos, Pintelas, et al., 2006).

To address the issue of an imbalanced dataset, we applied resampling techniques. Resampling falls into three categories: oversampling, under-sampling (Bach, Werner, & Palt, 2019), and hybrid sampling (Gazzah, Heckel, & Amara, 2015). As the datasets used in this experiment are small, oversampling techniques are ideal for handling imbalanced data. It creates additional instances to enhance the number of minority classes in the dataset (Jishan, Rashu, Haque, & Rahman, 2015). Oversampling helps ensure that the model has enough examples to learn from, potentially improving its ability to generalize and can reduce bias towards the majority class. The Synthetic Minority Oversampling Technique (SMOTE) method was employed in this experimentation (Chawla, Bowyer, Hall, & Kegelmeyer, 2002).

To investigate the effectiveness of employing balanced data on model performance, we also evaluated how well our models performed on imbalanced data versus other balanced datasets. On the other hand, the class label

for the second dataset is balanced (Low, Middle, High) and has corresponding distributions of 44%, 26%, and 30%, respectively.

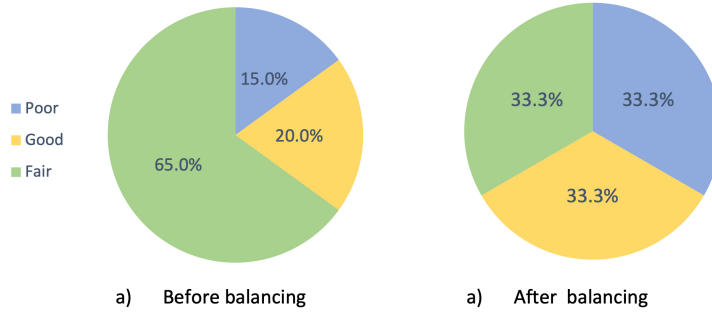


Figure 3.6: Visualization graph for balancing PTdataset

3.4.4 Feature selection

Feature selection is vital for increasing classifier accuracy, saving data-collecting effort, improving model interpretability, and shortening prediction time (Guyon, Gunn, Nikraves, & Zadeh, 2008). Feature importance scores are crucial in a predictive modeling task because they give an understanding of the data and the models. The basis for dimensionality reduction and feature selection may enhance a predictive model's effectiveness and performance on the problem. Although CNNs perform feature extraction through their convolutional layers when dealing with images dataset, combining them with PCA or DT when working with linear dataset (non-images) can lead to better generalization and accuracy in predictions.

This experiment applied two feature selection techniques to both datasets, principal component analysis and decision trees. These two approaches are used independently from each other to determine which one of the approaches incorporates improving the accuracy of the classification. The principal component analysis is an approach to minimize the dimensionality of such data, improving interpretability while preventing information

loss (Jolliffe & Cadima, 2016). It helps identify the most critical attributes in a dataset and simplifies data visualization in 2D and 3D. PCA helps to identify a sequence of linear variable combinations (Kurita, 2021).

The second feature selection technique is the decision tree. DT is a commonly used technique in machine learning and data mining (Sun & Hu, 2017). CART, ID3, and C4.5 are the traditional decision tree construction methods (Singh & Gupta, 2014). C4.5 improves ID3 by avoiding biased attributes. CART can handle features with more values (Roy, Mondal, Ekbal, & Desarkar, 2016)(H. Zhou, Zhang, Zhou, Guo, & Ma, 2021). The feature selection process is the procedure for creating a decision tree. Each feature is computed by particular standards (Quinlan, 1986). The decision tree algorithm has several key advantages, including adequate classification accuracy and high reliability. We employed the decision tree classifier (CART method) in selecting features, and it was implemented in the Scikit Learn library. The model offers a feature importance property that can be used to get the relative importance scores for each input feature once it has been fitted.

3.5 Model Evaluation

In this study, we evaluated the performance of the models using four evaluation metrics: accuracy, precision, recall, and F1-score, as shown in Table 3.8. Accuracy is commonly used to assess classifier generalization capacity. The trained classifier's accuracy is determined using total correctness, which refers to the total number of occurrences correctly predicted by the trained classifier. In general, the accuracy metric computes the ratio of accurate predictions to the total number of instances investigated. Precision is the second evaluation metric used, which is specified as the proportion of correctly predicted positive patterns in a class to all predicted positive pat-

terns. The third metric used is Recall, which determines the proportion of positive patterns that are identified correctly. F-Measure is also used as an evaluation metric, which shows the harmonic mean of recall and precision levels (Hossin & Sulaiman, 2015).

Table 3.8: Evaluation metrics used in the proposed Model

Metrics	Formula
Accuracy (Acc)	$\frac{TP + TN}{TS} \quad (3.1)$
Precision (PR)	$\frac{TP}{TP + FP} \quad (3.2)$
Recall (RE)	$\frac{TP}{TP + FN} \quad (3.3)$
F1 score(FS)	$\frac{2 \cdot PR \cdot RE}{PR + RE} \quad (3.4)$
TP: predicted Yes, and actual output was Yes. TN: predicted NO, and actual output was NO. FP: predicted YES, and actual output was NO. FN: predicted NO, and actual output was YES. TS: Total number of samples	

3.6 Experimental Results

This section describes the outcomes of the three proposed models using PTdataset and xAPIdataset. For both datasets, the sample data were split into training data (70%) and testing data (30%).

3.6.1 PTdataset Results

The experiment was repeated in different cases for the first dataset, as shown in Table 3.9. Cases 1, 2, and 3 with all data, 1044 instances for both subjects (Math and Portuguese). 730 instances for training and 314 instances

for testing. However, cases 4, 5, 6, 7, 8 and 9 with only one subject (Portuguese) with 649 instances and 33 attributes (454 instances for training and 195 instances for testing). One reason for choosing Portuguese instead of Mathematics is the sufficient data size and the availability of existing research, which enabled for meaningful comparisons with our findings using the same dataset.

Table 3.9: Different cases for the PTdataset

Cases	No. Instances	With DT	With PCA	Imbalanced	Accuracy %	Best Proposed Model	No. of Epochs
Case 1	1044			✓	89.81	Model 2	100
Case 2	1044	✓		✓	89.81	Model 1	100
Case 3	1044		✓	✓	88.22	Model 2	200
Case 4	649			✓	94.36	Model 2	100
Case 5	649	✓		✓	87.69	Model 1	50
Case 6	649				89.74	Model 2	100
Case 7	649	✓			86.67	Model 2, 3	100
Case 8	649		✓	✓	85.13	Model 1	100
Case 9	649		✓		88.21	Model 1, 3	50, 100

1) Case 1: All instances (1044) imbalanced without any feature selection

Table 3.10 shows the results of the experiment using the entire dataset for each model. Comparing the results of the three models, Model 2 achieved the highest accuracy of 89.81% when the number of epochs reached 100. Precision, Recall, and F1 score were high with results of 0.90, 0.94, and 0.92, respectively. Furthermore, the highest accuracy of Model 3 when the number of epochs is 50 was 88.85% which is higher than that of Model 1 when the number of epochs is 100, 87.68%.

Table 3.10: Performance evaluation of PTdataset (1044) imbalanced without any feature selection

Models	No. Epochs	Accuracy	Precision	Recall	F Score
Model 1	10	0.8567	0.9061	0.8542	0.8794
	50	0.8758	0.8923	0.9062	0.8992
CNN as the input	100	0.8768	0.8732	0.9323	0.9018
	200	0.8694	0.8802	0.9037	0.8918
Model 2	10	0.8662	0.8989	0.8802	0.8895
	50	0.8885	0.8792	0.9479	0.9123
Fully Connected - CNN	100	0.8981	0.8922	0.9479	0.9192
	200	0.8503	0.8156	0.8750	0.8398
Model 3	10	0.8694	0.8646	0.9171	0.8901
	50	0.8885	0.9062	0.9110	0.9086
CNN - Fully Connected - CNN	100	0.8662	0.8854	0.8947	0.8901
	200	0.8599	0.8442	0.8554	0.8427

2) Case 2: All instances (1044) imbalanced with DT as a feature selection

The experiment was repeated after applying a decision tree (DT) for feature selection. Table 3.11 illustrates the accuracy of each model with and without DT. We can observe from the results that Model 1 shows a significant improvement in accuracy with DT (89.81%) compared to accuracy without any feature selection (87.68%). However, Model 2 has its best standalone accuracy (89.81%) without DT, but its accuracy with DT is lower (86.94%). Model 3 has the same accuracy (88.85%) whether using DT or not, indicating no advantage either way. Comparing the outcome of the proposed models with DT as feature selection and without using any features, Model 2 achieves the highest accuracy obtained without any feature selection, while the impact of DT improves the performance of Model 1 but decrease the performance of Model 2. Model 3 shows no difference in performance with or without DT as feature selection.

we can see that almost all accuracies were obtained from models with feature selection more than the ones obtained without using any fea-

ture. However, the highest accuracy from Model 2 was 89.81%, which is equal to the highest accuracy when using Model 1 when CNN is used as the model's input.

Table 3.11: Comparison of accuracy of PTdataset (1044) imbalanced with DT as feature selection and without any feature selection

Models	No. Epochs	Accuracy%	Accuracy% with DT
Model 1 CNN as the input	10	85.67	84.39
	50	87.58	89.81
	100	87.68	89.49
	200	86.94	88.22
Model 2 Fully Connected - CNN	10	86.62	81.53
	50	88.85	86.94
	100	89.81	86.94
	200	85.03	85.99
Model 3 CNN - Fully Connected - CNN	10	86.94	85.67
	50	88.85	87.58
	100	86.62	88.85
	200	85.99	85.67

3) Case 3: All instances (1044) imbalanced with PCA as a feature extraction

The second feature extraction used in this examination is PCA. We tried to extract 40 features from PTdataset. Table 3.12 presents the experiment's findings using PCA as feature extraction. All models acquired polite accuracy results, and Model 2 reached the highest of 88.22% with 200 epochs. Furthermore, Table 3.13 compares the accuracies of PTdataset when applied feature selection and without any features. We can observe from the results that the accuracies obtained from the models with DT as feature selection was better than PCA. The highest accuracy was 89.81% with DT and 88.22% with PCA. Overall, the accuracies when applied DT with Model 1 are higher than the accuracies obtained from the same model without any feature selection.

Table 3.12: Accuracy of PTdataset (1044) imbalanced with PCA as feature selection

Models	No. Epochs	Accuracy% with PCA
Model 1 CNN as the input	10	85.03
	50	85.35
	100	87.26
	200	84.39
Model 2 Fully Connected - CNN	10	82.09
	50	86.31
	100	86.62
	200	88.22
Model 3 CNN - Fully Connected - CNN	10	81.85
	50	85.03
	100	83.44
	200	80.89

4) Case 4: Part of the dataset (649 instances) imbalanced without any feature selection

We employed the experiment using part of the imbalanced dataset, selecting only one subject with 649 instances. From Table 3.14, we can observe that the highest accuracy of 94.36% when the model was built using fully connected layers, followed by CNN with 100 epochs. Furthermore, the results of the accuracies obtained from all models were high, between 89%- 94%.

Table 3.13: Comparison of accuracy of PTdataset (1044) with DT, PCA as feature selection and without feature selection

Models	No. Epochs	Accuracy%	Accuracy% with DT	Accuracy% with PCA
Model 1 CNN as the input	10	85.67	84.39	85.03
	50	87.58	89.81	85.35
	100	87.68	89.49	87.26
	200	86.94	88.22	84.39
Model 2 Fully Connected - CNN	10	86.62	81.53	82.09
	50	88.85	86.94	86.31
	100	89.81	86.94	86.62
	200	85.03	85.99	88.22
Model 3 CNN - Fully Connected - CNN	10	86.94	85.67	81.85
	50	88.85	87.58	85.03
	100	86.62	88.85	83.44
	200	85.99	85.67	80.89

5) Case 5: Part of the dataset (649 instances) imbalanced with DT as a feature selection

The investigation was replicated using only 649 instances from all datasets after selecting features using decision trees. Table 3.14 indicates the results of the accuracy of the three models with DT as feature selection and without them. The highest score obtained reached 87.69% when using CNN as the input layer to Model 1 after selecting features from the dataset.

6) Case 6: Part of the dataset (649 instances) balanced without DT

In addition, we tried to oversample the dataset, as shown in Table 3.14. The highest accuracy for the balanced dataset without any feature selection reached 89.74% when we applied Model 2 with 100 epochs. Moreover, the accuracies from Model 3 are near the highest score, which reached 88.21% and 88.72% with epochs 50 and 100, respectively.

Table 3.14: Comparison of accuracy of PTdataset (645) with feature selection and without feature selection

Model	No. Epochs	Imbalanced			Balanced		
		Accuracy%	Accuracy% with DT	Accuracy% with PCA	Accuracy%	Accuracy% with DT	Accuracy% with PCA
Model 1 CNN as the input	10	90.77	84.62	66.67	85.13	84.62	70.26
	50	89.74	87.69	81.03	88.72	84.62	88.21
	100	91.28	87.18	80.00	83.08	85.64	87.18
	200	92.31	87.69	85.13	86.67	86.15	87.18
Model 2 Fully Connected - CNN	10	89.74	86.15	66.15	82.56	83.08	81.54
	50	90.77	86.15	70.26	87.18	86.67	84.10
	100	94.36	86.59	73.33	89.74	85.13	85.13
	200	90.26	85.64	73.85	86.15	82.05	85.15
Model 3 CNN - Fully Connected - CNN	10	93.85	85.13	69.74	86.61	82.56	83.59
	50	91.79	86.15	70.26	88.21	81.03	86.67
	100	92.31	86.67	71.79	88.72	86.67	88.21
	200	90.77	87.18	72.82	85.13	85.64	86.15

CNN

7) Case 7: Part of the dataset (649 instances) balanced with DT

In case 7, we used 649 instances from the balanced dataset with DT as a feature selection as exhibited in Table 3.14. From the outcomes, we can observe that all the accuracies obtained decreased compared to the same dataset without feature selection. The highest accuracy was 86.67% using Model 2 and Model 3 with 100 epochs.

8) Case 8: Part of the dataset (649 instances) imbalanced with PCA

For the feature extraction part, we applied PCA to the part of the imbalanced dataset. The result indicates that the highest accuracy obtained was 85.13% from Model 1, when using CNN as an input layer. From Table 3.14, Model 1 achieves the highest accuracy; either we applied DT or PCA as feature selection.

9) Case 9: Part of the dataset (649 instances) balanced with PCA

In the last case in our experiment, we tried to extract features from 649 instances after resampling the dataset. Table 3.14 showed that the highest accuracy was 88.21% when using CNN as an input layer

with 50 epochs or using CNN followed by fully connected followed by CNN with 100 epochs.

Table 3.15 summarizes the results of the accuracies of PTdataset with feature selections and without any feature selection. Comparing the outcomes when we used all the datasets and a part of them, the accuracy from 649 instances was higher than the accuracy obtained when we used 1044 instances from the dataset. Moreover, the performance of models when we used an imbalanced dataset was greater than the performance of balanced dataset. The highest accuracy for the imbalanced dataset was 94.36% and 89.74% for the balanced. This indicates that, in the imbalanced dataset, the majority class has a significant number of instances compared to the minority class. The model may achieve a higher accuracy by predicting the majority class correctly most of the time. This indicates that the model is not effectively addressing the minority class. However, a balanced dataset, where both classes have equal representation, forces the model to learn to predict both classes fairly, which can often lead to lower overall accuracy if the model struggles with the minority class. The lower accuracy in a balanced dataset reflects a more realistic performance measure across both classes. A model that performs well on a balanced dataset may be more desirable, even if its accuracy is lower. In addition, the accuracies with DT as feature selection were lower than that without any feature selection. However, using PCA as a feature selection for a balanced dataset gives higher accuracies than using DT as feature selection.

3.6.2 Comparison with previous studies using PTdataset

This study produced results that corroborate the findings of many of the earlier studies in this field. Table 3.16 compares our model's performance to previous works. All these studies, which are shown in Table 3.16 used PT-

dataset. In 2017, researchers used Decision Stump to predict the academic performance of students with 649 instances of the dataset (Akhai, Karia, Mahadik, Shah, & Gidwani, 2017). The accuracy obtained was 90.81%, which is good. Furthermore, a study in 2019, used ensemble techniques (J48 and Realada-boost) to predict student achievement with an accuracy of 95.78% using all instances of the dataset (Imran et al., 2019).

A research by Yousafzai et al. (2021) utilized CNNs, to get an accuracy of 85.56%. Several previous researchers performed a similar series of experiments using different machine learning algorithms to obtain good accuracy results. A work by Ghorbani and Ghousi (2020) used RF, and another study (Ng et al., 2022) predicted academic performance using a SVM with 88.22% and 77% accuracy, respectively.

Our model performed well compared with all the models mentioned in this research, with an accuracy rate of 89.81% when the entire dataset with 1044 instances was used, 89.74% when the entire dataset with 649 instances and applying the second model (fully connected followed by CNN) after oversampling, and 94.36% if the dataset was imbalanced. Our model got good findings compared to the baseline model and models from other researchers, as shown in Figure 3.7 and Figure 3.8. This difference depends on how the other studies dealt with the datasets, how they pre-processed the data, what algorithms were used, and how they evaluated their models and got their results. In addition, we compared the results of the proposed model accuracy with previous studies applying the CNN model; it performed better than them. As a result, according to our study topic, the proposed CNN models succeed in predicting student academic performance with educational datasets.

Table 3.15: Comparison of accuracy of PTdataset with feature selection and without feature selection

Models	No. Epochs	Imbalanced (1044 instances)			Imbalanced (649 instances)			Balanced (649 instances)		
		Accuracy %	Accuracy % with DT	Accuracy % with PCA	Accuracy %	Accuracy % with DT	Accuracy % with PCA	Accuracy %	Accuracy % with DT	Accuracy % with PCA
Model 1 CNN as the input	10	85.67	84.39	85.03	90.77	84.62	66.67	85.13	84.62	70.26
	50	87.58	89.81	85.35	89.74	87.69	81.03	88.72	84.62	88.21
	100	87.68	89.49	87.26	91.28	87.18	80.00	83.08	85.64	87.18
	200	86.94	88.22	84.39	92.31	87.69	85.13	86.67	86.15	87.18
Model 2 Fully Connected - CNN	10	86.62	81.53	82.09	89.74	86.15	66.15	82.56	83.08	81.54
	50	88.85	86.94	86.31	90.77	86.15	70.26	87.18	86.67	84.10
	100	89.81	86.94	86.62	94.36	86.59	73.33	89.74	85.13	85.13
	200	85.03	85.99	88.22	90.26	85.64	73.85	86.15	82.05	85.15
Model 3 CNN - Fully Connected - CNN	10	86.94	85.67	81.85	93.85	85.13	69.74	86.61	82.56	83.59
	50	88.85	87.58	85.03	91.79	86.15	70.26	88.21	81.03	86.67
	100	86.62	88.85	83.44	92.31	86.67	71.79	88.72	86.67	88.21
	200	85.99	85.67	80.89	90.77	87.18	72.82	85.13	85.64	86.15

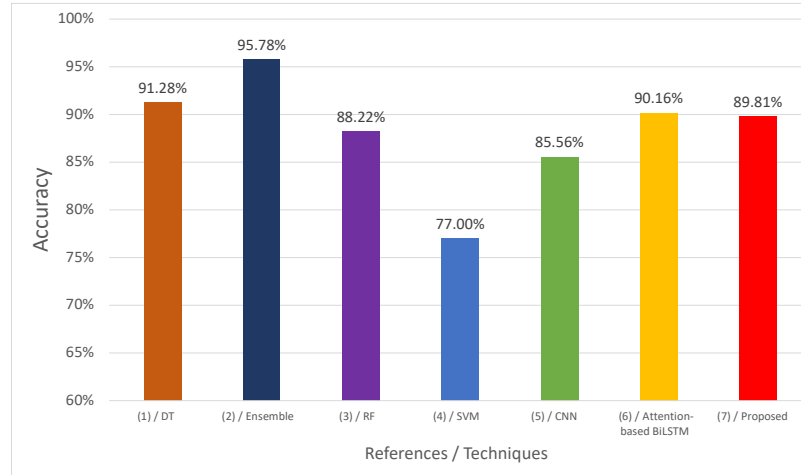


Figure 3.7: Comparison of accuracy of the proposed model and previous studies for all instances of PTdataset

Table 3.16: Using PTdataset, evaluate the accuracy of earlier research and the proposed model

No	Ref	Year	Techniques	No. of All Instances	Testing Accuracy
1	(Hamoud, 2016)	2016	DT	1044	91.28%
2	(Imran et al., 2019)	2019	Ensemble (J48, Realadaboost)	1044	95.78 %
3	(Ghorbani & Ghousi, 2020)	2020	RF	1044	88.22%
4	(Ng et al., 2022)	2021	SVM	1044	77.00%
5	(Yousafzai et al., 2021)	2021	CNN	1044	85.56 %
6	(Yousafzai et al., 2021)	2021	Attention-based BiLSTM	1044	90.16 %
7	proposed	2024	fully connected – CNN / CNN as the input	1044	89.81%
8	(Cortez & Silva, 2008b)	2008	DT	649	93.00%
9	(Akhai et al., 2017)	2017	Decision Stump	649	90.81 %
10	proposed	2024	fully connected – CNN	649	94.36%

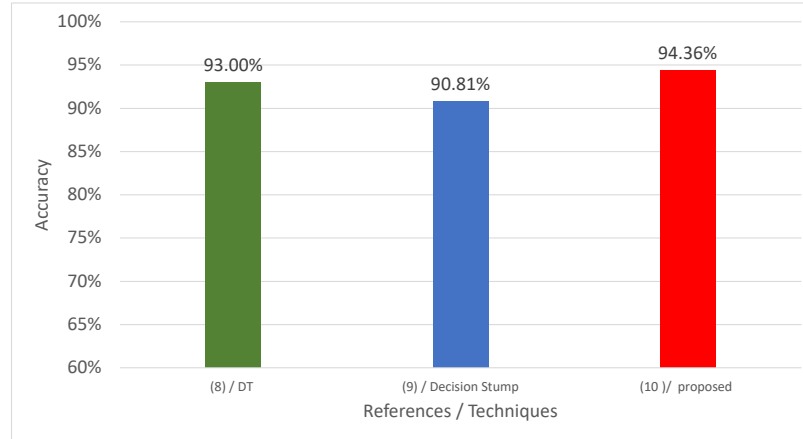


Figure 3.8: Comparison of proposed and previous models' accuracy for a portion of PT-dataset

3.6.3 xAPIdataset Results

For the second dataset as presents in Table 3.17, we have three cases as the follows:

1) Case 1: Without feature selection

The highest accuracy obtained was 83.45% when we used CNN as the model's input, reaching 200 epochs. As shown in Table 3.17, the accuracies of Model 3 are higher than that of Model 2. For instance, the accuracy of Model 3 was 80.13%, with 100 epochs, while the highest accuracy of Model 2 was 72.52% with 200 epochs.

2) Case 2: With DT as a feature selection

We repeated the experiment with DT as a feature selection. The highest score was 84.83% when using CNN as the model's input. Furthermore, we can observe that the accuracies achieved from Model 1 with different epochs are higher than those from Model 2 and Model 3. However, Model 3 accuracies are higher than those of Model 2. The highest accuracy was 80.21% for Model 3 and 70.83% for Model 2.

3) Case 3: With PCA as feature extraction

In the third case we applied feature extraction using PCA. The highest accuracy obtained was 82.29% when CNN was used as the model input. For the other models, the performance of Model 3 was higher than that of Model 2. In addition, when the number of epochs is 100, the accuracy of the Model 3 is 81.25%, and 77.08% for the Model 2.

When we compare the accuracy of the three models as indicated in Table 3.18, it is evident that the maximum accuracy is achieved when using CNN as the input for Model 1. It reaches 84.83% when using DT as a feature selection with 50 epochs and 83.45% without feature selection.

Table 3.17: Different Cases for xAPIIdataset

Cases	With DT	With PCA	Accuracy %	Best Proposed Model	No. of Epochs
Case 1			83.45	Model 1	200
Case 2	✓		84.83	Model 1	50
Case 3		✓	82.29	Model 1	200

Table 3.18: Comparison of accuracy of xAPIIdataset with a feature selection and without any feature selection

Models	No. Epochs	Accuracy%	Accuracy% with DT	Accuracy% with PCA
Model 1 CNN as the input	10	76.16	81.13	81.25
	50	78.25	84.83	80.21
	100	80.52	83.33	81.25
	200	83.45	82.29	82.29
Model 2 fully connected - CNN	10	69.16	65.14	70.83
	50	67.14	67.71	72.92
	100	71.25	70.83	77.08
	200	72.52	69.79	71.88
Model 3 CNN - Fully connected - CNN	10	76.13	75.00	72.92
	50	78.45	76.04	76.04
	100	80.13	78.12	81.25
	200	77.12	80.21	79.17

3.6.4 Comparison with previous studies using xAPIIdataset

Comparative analysis was done with the previous studies that used xAPIIdataset, as shown in Table 3.19. In 2015, a paper by E. A. Amrieh et al. (2015) utilized the same dataset and three distinct machine learning algorithms in their study: DT, NB, and ANNs. Their results indicated that the accuracies were 61.30% for DT, 73.8% for ANN, and 72.5% for NB. One year later, the same authors (E. A. Amrieh et al., 2016) used ensemble techniques to predict the performance and got an accuracy of 79.1%. Another work by Pujianto, Prasetyo, and Taufani (2020) analyzed the effectiveness of two classifiers, C4.5 and KNN, using the SMOTE preprocessing approach. The C4.5 decision tree technique produced improved prediction performance in experiments with accuracy, recall, and precision scores of 71.09%, 71.63%, and 71.54%, respectively.

Table 3.19: Using xAPIIdataset, Compare the Accuracy of Earlier Research and the Proposed Model.

No	Ref	Year	Techniques	No. of instances	Accuracy
1	(E. A. Amrieh et al., 2015)	2015	ANN, NB, DT (best ANN)	480	73.8 %
2	(E. A. Amrieh et al., 2016)	2016	Ensemble methods	480	79.1 %
3	(Pujianto et al., 2020)	2021	DT and KNN	480	71.09%
4	(M. Kumar, Mehta, Nayar, & Sharma, 2021)	2021	Ensemble Meta-Based Tree	480	80.33%
5	proposed	2024	CNN with one dimension as the input (DT as feature extraction)	480	84.83%

Moreover, in 2021, a publication (M. Kumar et al., 2021) constructed a prediction model using various machine learning techniques on the entire dataset. In addition, they implemented several ensemble meta-based models that were integrated with ML algorithms for classifying data. For instance: Bagging, AdaBoostM1, and RandomSubSpace. The results showed that the Multilayer Perceptron Machine Learning approach had up to 80.33% accuracy performance using the ensemble meta-based technique (AdaBoostM1). Compared to all the models mentioned in previous studies, our model per-

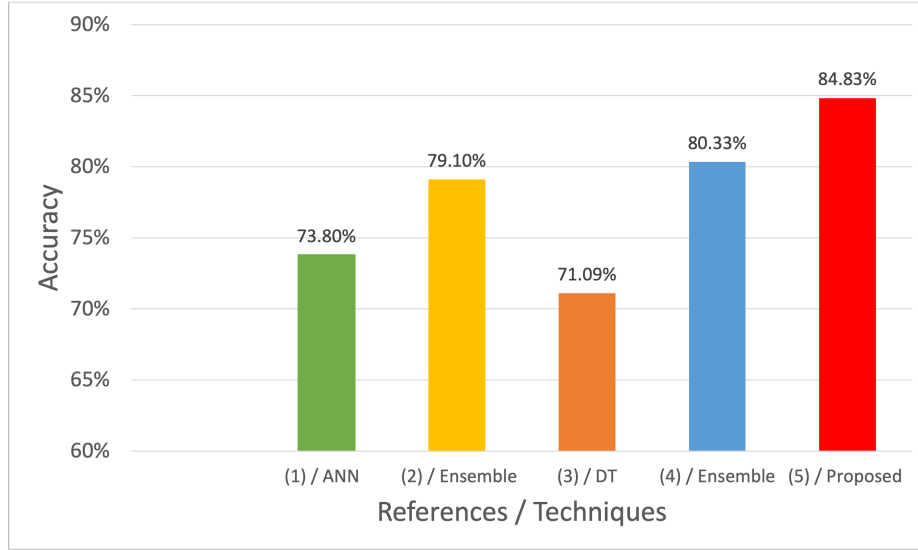


Figure 3.9: Comparison of accuracy of the proposed model and previous studies for xAPI-dataset

formed better, with an accuracy rate of 84.83% as presented in Figure 3.9. As a result, Model 1 using CNN with one dimension as the input and DT as feature extraction can be used to predict student performance successfully.

3.7 Discussion

From the experiments, the best algorithms for the first dataset compared to other classifiers is Model 2, using fully connected layers followed by CNN. The accuracy obtained using all instances was 89.81% for imbalanced without feature selection, 86.94% with DT, and 88.22% with PCA as feature selection. Moreover, using part of PTdataset with imbalanced reached accuracy of 94.36% without feature selection and 86.59% with DT as feature selection and 85.13% with PCA as feature selection. However, after re-sampling the dataset, the accuracy was 89.74% without feature selection and 86.67% with DT as feature selection, and 88.21% with PCA. In addition, the other evaluation metrics results for Model 2 were higher than other models, the obtained precision, recall, and F1 score were 0.892, 0.948, and

0.919, respectively. Using part of a dataset may yield to higher performance compared to the whole dataset. The subset may contain quality or more relevant data, while the whole set has irrelevant or noisy instances that could affect the model's performance. Furthermore, a part of the dataset can reduce the complexity of the learning problem, enabling the model to learn faster. The final results from PTdataset indicate that the best model compared with other proposed models is Model 2, which uses fully connected layers followed by CNN. The second was Model 1 and last was Model 3. Despite that Model 2 does not get the best accuracy in all cases, but it still reaches greater accuracy compared to all other accuracies obtained in the previous studies. This makes Model 2 is a strong architecture for student performance prediction.

For the second dataset (xAPIdataset) used in this experiment, the best model among the other models is Model 1, using CNN as model input. The accuracy results of Model 1 are 83.45% without any feature selection, 84.83% with DT, and 82.29% with PCA as feature selection. The results obtained indicate that Model 1 has better performance than other previous studies, with an accuracy of 84.83%.

The findings are consistent with other studies showing that machine learning and deep learning techniques are ideal for predicting student performance with more than 84% accuracy. Furthermore, different datasets and machine learning techniques result in various performances. When comparing the performance of the selected datasets, PTdataset performed better than xAPIdataset. When comparing the performance of the chosen models, Model 2 performed better than Model 1 in PTdataset but was lower in xAPIdataset.

3.8 Summary

Identifying students' academic progress and improving their grades are major benefits of applying machine learning and deep learning. It can be used for student profile modeling, aiming to produce knowledge from data automatically. This study proposes three models using convolutional neural networks and fully connected layers to predict student success in the year. Model 1 used CNN as the model input, Model 2 used fully connected layers followed by CNN, and Model 3 used CNN followed by fully connected layers followed by CNN. In addition, two publicly available datasets were used (PTdataset and xAPIdataset). The first dataset is imbalanced, and the second dataset is balanced. We applied two feature selection techniques: decision tree and principal component analysis. The models were evaluated with feature selection and without any feature selection. Moreover, different evaluation measures were used for evaluating each model's performance. Among the models obtained for the PTdataset, Model 2 has the best results when using part of the dataset, with an accuracy of 94.36% without any feature selection when the data was imbalanced and 89.74% when the data was balanced. However, for the xAPIdataset, Model 1 obtained the best results either with or without feature selection. It was 84.83% accuracy with DT, 82.29% with PCA, and 83.45% without feature selection. The proposed model's results were compared with previous models, and based on the outcomes, it can be concluded that the suggested models are highly effective in predicting student performance. It can also be valuable in assisting teachers and improving their teaching skills. Based on the promising findings presented in this study, further research in the field of intelligent tutoring systems are strongly encouraged, including utilizing more machine learning models with deep learning, investigating new approaches to conduct experiments, and selecting the appropriate features influencing student performance.

Chapter 4

Adaptive Pedagogical Model Based on Classifier System and Reinforcement Learning

4.1 Introduction

4.2 Background

4.3 Proposed Models Architecture

4.4 Model Evaluation

4.5 Experimental Results

4.6 Discussion

4.7 Summary

Chapter 4

Adaptive Pedagogical Model Based on Classifier System and Reinforcement Learning

4.1 Introduction

In recent years, the area of education has changed rapidly due to the improvements in technology and the growing need for learning experiences that are individualized and adaptable. Traditional education methods struggle to meet all learners' needs because they do not adapt to their unique experiences, preferences, and learning styles. Consequently, there is a growing need for innovative and flexible pedagogical approaches that may address the requirements of individual students and improve their educational journey (Melo, Mascarenhas, & Paiva, 2018; Korkmaz & Correia, 2019; Asthana & Hazela, 2020).

The primary purpose of this study is to design and develop an adaptive

pedagogical model that uses the strength of genetic algorithms and reinforcement Learning to build a learning environment that is both individualized and flexible for learners. The proposed model integrates the idea of classifier systems with reinforcement learning to develop a reliable pedagogical system within the ITS (Alshaikh & Hewahi, 2021). This model uses these techniques (GA and RL) to find the best policies (tactics) to decide the next action when multiple actions are available. The term tactic means here the policy that the ITS should follow according to the student's answer and performance. This will enhance the learning experience and improve the overall quality of the ITS teaching mechanism. This would enable ITS to provide students with optimal feedback, allowing them to effectively address their mistakes and improve their problem-solving skills. Therefore, employing various strategies would have an impact on students' academic progress. GAs can be used to evolve classifiers through the learning classifier system. The input to the LCS is the initial population, and the output is the trained classifier. The reinforcement learning guides the learning process by updating the Q-table values, allowing the RL agent to refine further and optimize the behavior of the classifiers. The input to the RL is the last population obtained by GAs, and the output is the Q-table values, which can be used to predict the best tactic.

4.2 Background

This section presents an overview of the Learning Classifier System (LCS) and its main components, the reinforcement learning, and the integration of LCS with RL.

4.2.1 Components of the Learning Classifier System (LCS)

Learning classifier systems (Holland & Reitman, 1978) uses an if-then rule structure to adaptively train and develop classifiers that may predict or categorize data instances (Urbanowicz & Browne, 2017). Holland defines the classifiers as a conditional rule based on if-then statements. The term "classifier" originates from the capacity of rules to categorize messages into various message sets.

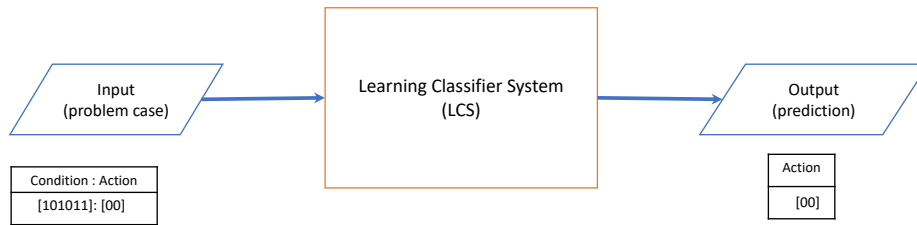


Figure 4.1: Learning classifier system main process

The process of any learning classifier system is shown in Figure 4.1. The input to the learning classifier system is a string using binary representation 0s and 1s. This classifier is consisting of two parts: condition and action. For example, [101011]:[00] is a classifier that has [101011] as its condition and [00] as its action. The values of 0's and 1's are codes for certain inputs and outputs. After processing the input to the LCS, the output will be the predicted action. Learning classifier systems include these essential components:

1. **Classifiers/Rules:** Classifiers and rules are the building blocks of an LCS. They map input (conditions) into output (actions or predictions) (Booker, Goldberg, & Holland, 1989). When the criteria are met, action can be taken. These rules classify fundamental objects by their features.
2. **Bucket Brigade Algorithm (Credit Assignment):** The Bucket Brigade Algorithm (BBA) is a technique used to address the problem of credit

assignment in learning classifier systems. Its purpose is to efficiently allocate credit. This method calculates the specific impact of each classifier in obtaining good outcomes. Figure 4.2 illustrates the fundamental procedure of the BBA approach, and the following material is a comprehensive explanation of the operational principles of this algorithm (Urbanowicz & Browne, 2017):

- Find the matching classifiers for a problem case, including conditions and actions.
- Determine the bid values for each classifier, considering its specificity and strength.

$$\text{Bid} = C \cdot \text{Strength} \cdot \text{Specificity} \quad (4.1)$$

Where specificity is the ratio of non-don't care bits to condition portion length, and C is a bid coefficient, generally constant, less than 1.

- The matched classifier with the highest bid wins the auction. The message of the winner is then added to the list, and the bid amount is subtracted from their strengths.
- Next, verify the accuracy of the output by comparing the predicted action from the winning classifier to the actual action within the specified ranges.
- The winner of the classifier should be rewarded and added to its strength if the output is the correct possible option.

3. **Genetic Algorithm (Rule Discovery Component):** GA utilizes evolutionary processes to generate new rules. The system selects, combines, and mutates the classifier population in order to generate new rules that are more adapted to the environment. The objective is to

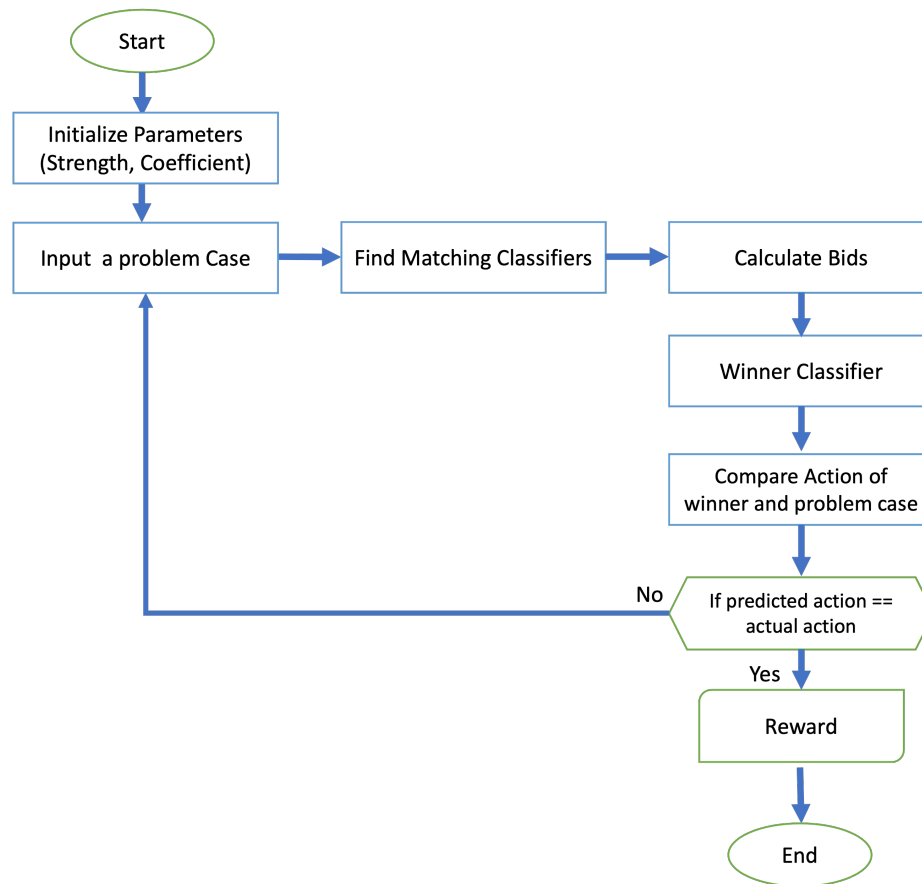


Figure 4.2: Bucket Brigade Algorithm procedure

enhance the overall performance of the system by generating novel classifiers through the combination and modification of current ones.

GA is an optimization technique that employs natural selection and reproduction to discover the optimal computational problem solution (Carr, 2014). They require no extra information about the problem and provide user-set randomization and control. Genetic algorithms are more efficient and powerful than other search algorithms and can solve issues that other optimization approaches cannot (Beasley, Bull, & Martin, 1993). GA evolves problem solutions using selection, crossover, and mutation to resemble natural evolution. Several processes make up the fundamental operations of the genetic algorithm, as shown in the following:

- 1) **Initial Population:** The population is the initial collection of individuals generated randomly when the algorithm is called into play. A chromosome is formed by integrating a series of parameters known as genes, and each individual is a potential solution to the problem (Thede, 2004). An individual is a possible solution to the problem.
- 2) **Fitness Function:** It is an instrument used to determine the distance between a particular solution and the optimal value. Within the population, this function assigns a fitness score to each individual; higher scores indicate more effective solutions.
- 3) **Selection:** Individuals are chosen to pass on their genes to the subsequent generation depending on the fitness score they have achieved. Selection methods include rank, roulette wheel (Buseti, 2007), and tournament (Mitchell, 1998). To achieve their goal of selecting stronger individuals, they prioritize diversity while maintaining some degree of variety.
- 4) **Crossover:** This process, also known as recombination, occurs when the genetic information of selected parents is combined to generate novel offspring. The strategy is to integrate components from two reasonable alternatives to come up with a more effective solution. Multi-point, single-point, and uniform crossover methods can be utilized for crossover (Buseti, 2007).
- 5) **Mutation:** Random gene mutations create variance in offspring. The population's genetic diversity is ensured by doing this, and the algorithm can examine a wider area, which may allow it to avoid finding a local optimal solution. The new population formed from the offspring will entirely or partially replace the existing population.

- 6) **Replacement:** The new population formed from the offspring will either completely or partially replace the old population through the replacement process. Some of these strategies, such as elitism, enable the most effective solutions to be retained by the subsequent generation, while others entirely replace the population. It is necessary for the algorithm to repeat step 2, which involves the replacement of the previous generation of chromosomes with the current generation.
- 7) **Termination:** The algorithm is deemed terminated when a stopping requirement is satisfied. This terminating criterion could be a high enough fitness level, a maximum number of generations, or a drop in fitness, which indicates that additional iterations are not likely to provide better answers.

Each operation helps the evolutionary algorithm seek the solution space and converge to the ideal solution. When it comes to the success of the algorithm, the design and adjusting of those steps are pretty important. The evolutionary algorithm has many benefits, one of which is its robustness in dealing with problem spaces that are complicated, multi-modal, and noisy. It is an excellent fit for scenarios involving parallel processing because its parallelism feature allows it to evaluate many solutions simultaneously. Furthermore, the genetic algorithm is flexible and can be used for a wide variety of issues, eliminating the requirement for considerable expertise tailored to the particular problem. In addition, it has an effective global search capability, which reduces the chances of being caught in local optima, an issue that frequently arises in various optimization strategy approaches (Bodenhofer, 2003).

Despite its benefits, genetic algorithms have drawbacks. It might be

computationally intensive, slowing performance and requiring many resources, especially for complex cases. Early convergence can occur if population variation is not maintained. This causes the algorithm to concentrate on solutions that are not optimum (Z. Wang & Sobey, 2020). The performance of the genetic algorithm is highly influenced by the size of the population as well as the design of its parameters, which include crossover and mutation rates. To achieve the greatest results, fine-tuning these values may need a lot of trial and error.

The field of genetic algorithms is considerable breakthroughs as a result of research into hybrid methodologies, adaptive parameter tuning, and sophisticated selection processes. One of the objectives of the development of this discipline is to make GAs even more efficient, faster, and adaptable to a wider variety of issues.

4.2.2 Overview of Reinforcement Learning

Reinforcement Learning is a decision-making area where an agent interacts with its environment and earns rewards to complete an assigned task (AlMahamid & Grolinger, 2021). It maximizes reward through progressive refining. It is like a young child learning to walk; the child earns positive reinforcement for activities that bring them closer to the goal, while they receive punishment when they make a mistake.

Key components of the RL mechanism are as follows (Nian, Liu, & Huang, 2020):

1. Agent: The learner or independent decision-maker is referred to as the agent.
2. Environment: a real-world setting in which an agent may move about, be taught, and make choices.

3. State: The present position that the agent is situated in with regard to the environment
4. Action: The capabilities of the agent and the tasks that an agent is able to carry out.
5. Reward: Ecological feedback
6. Policy: The agent's approach for determining what to do next (next action) is contingent on the circumstance that is now taking place (current action), as stated in the policy.
7. Value Function: This function is responsible for determining the expected reward of a state.
8. Model of the Environment: When seen from the perspective of the agent, the probability distributions of state-action pairings are laid out over the states.

An agent is required to receive a state from the environment, carry out an action, and then return with a reward and the next state in order to complete the process of RL. This loop continues to run forever or until a condition that is considered to be terminal is achieved. The purpose of the agent is to maximize cumulative rewards, which are frequently discounted, while maintaining a balance between now-available rewards and those that will be in the future. Performing activities like as playing video games is an example of how the agent learns from the results of its actions in an iterative fashion.

RL advantages would be fascinating to study. That makes it a good choice for teaching algorithms to complicated decision-making. In dynamic environments, RL agents can adapt to their surroundings, which is a significant asset. Unlike previous machine learning paradigms, reinforcement

learning does not require a pre-labeled dataset, so agents may learn by performing actions like humans. This method handles delayed rewards well since it is built for long-term decision-making. RL agents may directly translate unprocessed input data into actions and integrate learning components using end-to-end learning. They scale effectively to address complicated issues with multiple states and actions. Continuously interacting with their surroundings helps them enhance their policies.

Sample inefficiency, expense, and impracticality are its downsides, especially with physical systems. Exploration and exploitation are challenging to balance. The agent struggles to recognize advantageous behaviors due to delayed rewards and sparse cues. Stability, convergence, incentive structures, and computing loads are further issues. As RL agents investigate dangerous activities, safety is an issue. Partial state observation complicates learning. Credit assignment remains a major RL application challenge.

4.2.3 Combination of Genetic Algorithm and Reinforcement learning

Reinforcement learning and genetic algorithms are two effective methods that may be utilized to solve complicated optimization and decision-making challenges.

Using the production of populations, evaluation of fitness, selection, crossover, mutation, and termination, genetic algorithms aim to replicate natural evolution and work by involving these processes. The ability to encode answers and construct fitness functions makes them an excellent tool for problem-solving in situations when these capabilities are available. However, reinforcement learning employs ML models to teach them to make a series of decisions depending on their interactions with an environment. In a setting that is both complicated and unpredictable, the model acquires the

ability to accomplish its objectives through the receipt of rewards or penalties, which ultimately results in the improvement of policy (strategy) over the course of time (Drugan, 2019).

The purpose of integrating genetic algorithms and reinforcement learning is to capitalize on the advantages that are offered by both methods. In complicated issues, GAs has the ability to incorporate variety, develop populations of policies, improve RL parameters for faster convergence, and minimize the search space for RL. It is possible to create a more robust learning mechanism that can learn both inside and beyond the lifespan of agents by combining GAs with RL tools (Song et al., 2023). An improvement in learning processes in complicated simulations may be achieved by the integration of RL with GAs. It can provide a powerful tool for solving complex problems where neither GA nor RL alone would be as effective or efficient (Song et al., 2023).

4.3 Proposed Models Architecture

This section describes the main strategies to build a novel model that integrates the learning classifier system and reinforcement learning. It explains the structure of the proposed models in detail. In this study, we propose three distinct methodologies to construct the pedagogical model of an intelligent tutoring system. We introduce three models with different architectures, as illustrated in Figure 4.3. The First model is to build a learning classifier system. Second, we use only reinforcement learning. Third, we develop a hybrid learning classifier system with reinforcement learning (PMCR). The main objective of deploying these techniques is to determine the most effective strategies for making decisions by finding the best policies (tactics) to choose the next action when several actions are available. For example, the instructor/ITS may use these tactics to guide learners based on their

answers to specific problems. This approach helps teachers/ITS provide students with the best feedback, enabling them to correct their mistakes and solve tasks correctly. Thus, different tactics can significantly affect student's learning progress.

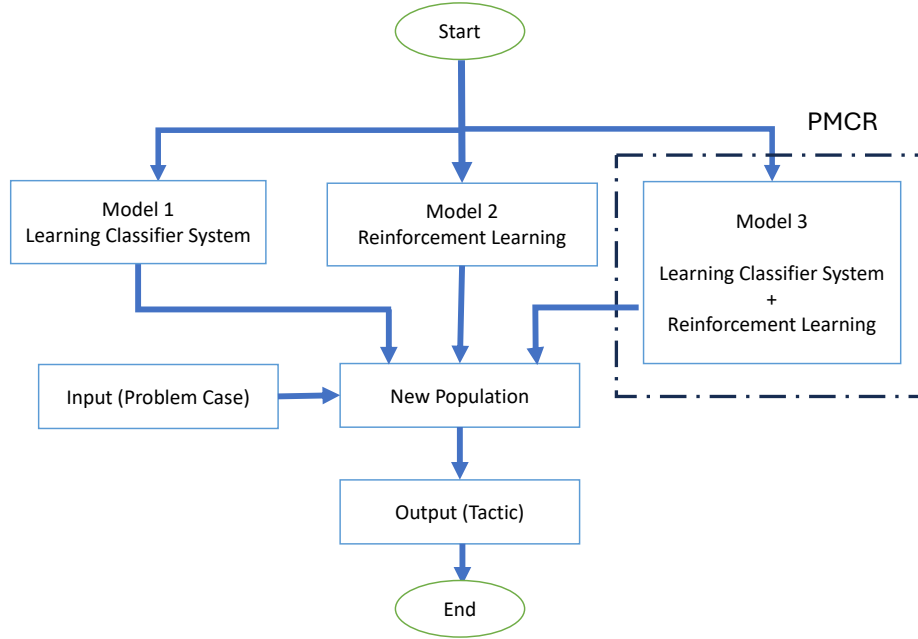


Figure 4.3: The three adopted models

1. Model 1: Learning Classifier System.

Building the architecture of the first model depends on the components of the learning classifier system and is based on GA. Figure 4.4 presents the proposed model architecture, which begins with individual/chromosome representation and then generates a random population. After that, it goes to the processing step to produce a new population. The output will be the prediction of a suitable tactic for a specific student. The pseudocode algorithm for model 1 is given in Algorithm 1.

The following are the main procedures that build this learning classifier system

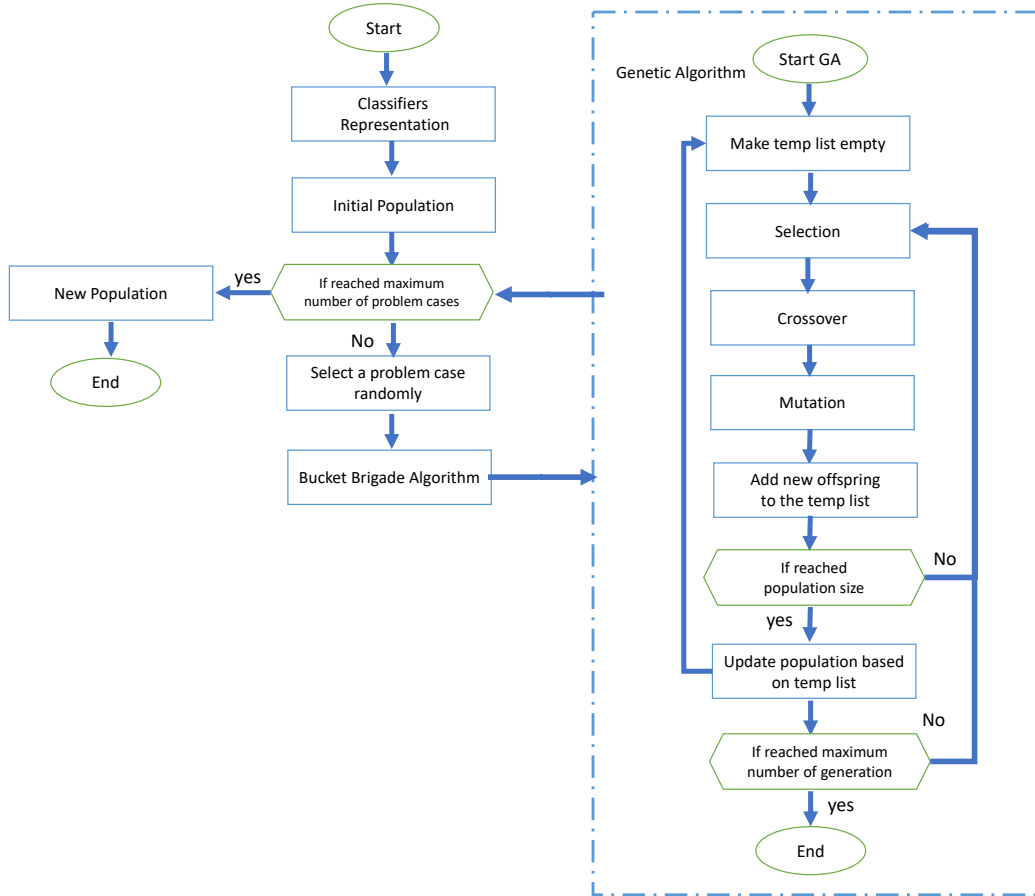


Figure 4.4: Model 1: The architecture of the learning classifier system

(a) **Classifier Representation** The classifier is a sort of a rule that contains condition and action and written in the form “condition: action”.

- **Conditions:** In educational environments, conditions related to challenges or circumstances that are presented by teachers to learners. These circumstances may be part of an activity, task, or problem to solve. These conditions attempt to test students understanding, increase critical thinking, or instruct them to problem-solving. We define a set of problems $(P) = [p1, p2, p3, \dots, pn]$, where pi is one of the n problems given to the learner. Moreover, S is a set of solutions $= [s1, s2, \dots, sn]$, where each problem pi has a

Algorithm 1. Learning Classifier System (Model 1)
Input: max_num_gen, maturity_function, pop_size, mutation_rate, crossover_rate, max_num_problems.
Output: new population with best individual.
1. Represent the classifier. 2. Generate initial population. 3. $num_problem \leftarrow 0$ 4. While $num_problem < max_num_problems$ do: 5. Select a problem case randomly. 6. Apply bucket brigade algorithm. 7. Initialize temp list to empty list 8. $n_gen \leftarrow 1$ 9. While $n_gen < max_num_gen$ do: 10. While $pop_size < max_pop_size$ do: 11. Select parents from population. 12. Produce offspring (crossover) from chosen parents 13. Mutate the resulting offspring. 14. Add new offspring to the temp list 15. $pop_size \leftarrow pop_size + 1$ 16. Update population based on temporary list 17. Make temp list empty. 18. End while. 19. $n_gen \leftarrow n_gen + 1$ 20. End while. 21. $num_problem \leftarrow num_problem + 1$ 22. End while. 23. return new population with best individual

corresponding value si , which is one of three possible values, 0 (not solved the problem), 1 (partially solved the problem), and 2 (solved the problem). Furthermore, we give a weight to each problem based on its type, objective and level of complexity. We start from lower-level skills to higher-level ones and assign weights $W=[w_1, w_2, \dots, w_n]$. If we assume we have six problems that students must solve $[p1, p2, p3, p4, p5, p6]$, for each problem, we assign a corresponding weight, as shown in Table 4.1. The weights in the

table corresponding to problem's nature are just assumption and can be changed or updated based on requirements. Furthermore, for classifier representation we add to the condition a wildcard # that can accept any value (*si*), 0, 1, or 2.

Table 4.1: Problem description

Problems	Descriptions	Weights (w)
p1	Remembering	0.4
p2	Understanding	0.5
p3	Applying	0.6
p4	Analyzing	0.7
p5	Evaluating	0.8
p6	Creating	0.9

- **Actions:** The actions or tactics are strategies that the teachers use to enhance students learning and improve their performance. In the context of a learning system, actions relate to the agent's decisions or interventions to affect the learning environment. Teacher's feedback or comments are essential teaching tools to guide learners toward better understanding. They clarify misunderstandings, encourage good work, and promote progress. They guide learners with constructive criticism and good motivation. Our proposed model suggests eight actions/tactics ($T1, T2, \dots, T8$) that are going to be explained in the next sections.

(b) **Generate an initial population.**

The second step is to generate random classifiers. Every classifier has a fitness value, and initially all fitness values for all the classifiers are equal. From the generation of random classifiers, we get the initial population. The population has a fixed number

of classifiers.

(c) **Generate random problem cases**

The next step is to generate random problem cases (P), for each problem in P , a solution that can be '0', '1' or '2' is given. It represents the input for which the classifiers will be evaluated. For each problem case, we must find the best match/similar classifiers by applying the bucket brigade algorithm, explained in the next step, and then check if it is a potentially correct solution. The potential correct choice can be known if its action (the tactic) is within the range of maturity, as will be explained later.

(d) **Bucket Brigade Algorithm (BBA)**

After generating the initial population randomly, we must discover the problem-case-matches classifiers and assign credit using the bucket brigade algorithm. The quality of the classifier depends on strength and specificity integrated into one metric (bid). For each problem case generated randomly and over a specific number of generations, we apply the BBA to get the matched classifiers and the winner classifier. The fitness function is utilized to assess the individual's fitness level within the population. It signifies an individual's capacity to outperform others. Each individual is assessed according to their fitness function in every iteration. The fitness values of each classifier will be the same at the time of the population's initial generation. Depending on how well the classifier performs, its fitness values are increased or decreased. If the predicted action from the winner classifier matches the actual action for the problem case, then the classifier is going to be rewarded, and its strength is increased.

(e) **Maturity function**

The maturity function is the function used to determine the proper tactic for the given student's performance. The result of the maturity function identifies the tactic which is supposed to be used. The tactic obtained through the maturity function is compared with the predicted tactic by the model to identify whether the model produces a correct tactic or not. The range of the maturity function is from 0 to 7.8 as explained below. Each problem is given a weight based on its importance and objective as given before in Table 4.1. In addition, let (E) be a set of problem evaluations where $E=[e_1, e_2, \dots, e_n]$. The maturity equation can be written as follows:

$$\text{Maturity function} = \sum_{i=1}^n e_i \quad (4.2)$$

Where e_i is calculated by multiplying s_i by its corresponding weight w_i . For example, e_1 is calculated by multiplying s_1 by the corresponding weight w_1 . Similarly, e_2 is calculated by multiplying s_2 with w_2 , and so on as the following:

$$\begin{aligned} \text{Maturity function} = & (s_1 \cdot w_1) + (s_2 \cdot w_2) + (s_3 \cdot w_3) \\ & + (s_4 \cdot w_4) + (s_5 \cdot w_5) + (s_6 \cdot w_6) + (s_n \cdot w_n) \end{aligned} \quad (4.3)$$

where e_i is the evaluation result of p_i , and p_i has a corresponding value s_i , which is one of three possible values, 0 (not solved the problem), 1 (partially solved the problem), and 2 (solved the problem). The result of the maturity function can be matched against a specific tactic that could be suitable for an individual based on maturity ranges. The ranges used here are only for scientific research, and do not reflect real tactics. The used maturity ranges are as below:

- If the maturity range is between 0 and 1, assign tactic T1.
- If the maturity range is greater than 1 and up to 2, assign tactic T2.
- If the maturity range is greater than 2 and up to 3, assign tactic T3.
- If the maturity range is greater than 3 and up to 4, assign tactic T4.
- If the maturity range is greater than 4 and up to 5, assign tactic T5.
- If the maturity range is greater than 5 and up to 6, assign tactic T6.
- If the maturity range is greater than 6 and up to 7, assign tactic T7.
- If the maturity range is greater than 7 and up to 7.8, assign tactic T8.

(f) **Genetic operators (Selection, Crossover, and Mutation)**

To evolve the population of classifiers, a process of selection, crossover, and mutation, is applied. The generation of new classifiers was handled by these genetic operators by replacing the individuals with the weakest fitting by the strongest ones. Furthermore, the classifiers conditions are randomly swapped, altering their condition, and creating offspring classifiers by merging the features of two-parent classifiers and taking the average of their strength. This process can be applied using a crossover operator. Also, for some classifiers, we do some crossover to its actions. In addition, we apply mutation to their genes, where we select one or more random gen and flip them or select two positions on the chromosome at random and interchange their

values. Here is an example showing how the genetic operators (crossover, and mutation) are used. Let’s assume we select randomly two chromosomes, as shown in Figure 4.5. We select a crossover point on each chromosome and swap the data beyond that point between the two chromosomes. The output will be two new offspring (new chromosomes).

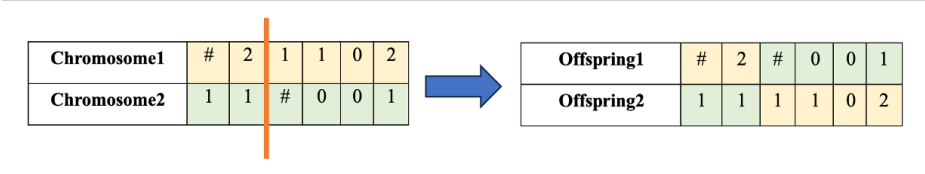


Figure 4.5: Single Point Crossover



Figure 4.6: Swap Mutation

Then, we can apply mutation to these chromosomes by selecting two positions (gens) on the chromosome at random and exchanging their values as presented in Figure 4.6.

(g) **New population**

The algorithm keeps running, tracks the correct prediction, and terminates once the desired accuracy or the maximum number of training problem cases is achieved. The output from the GA algorithm is a population set of trained classifiers.

2. Model 2: Reinforcement Learning

The second model utilized in this study, as shown in Figure 4.7, is to train the initial population of classifiers by reinforcement learning.

The pseudocode for reinforcement learning (Q-learning) is given in Algorithm 2 (Y. H. Wang, Li, & Lin, 2013). The following procedures show how the model was built:

- (a) Define the Q-table and initialize the values with zeros.
- (b) Select a random problem case.
- (c) The Q-table values will be updated during the training process based on the rewards received when actions are taken.
- (d) For each classifier, the reward is calculated, and the Q-values are updated for the corresponding state-action pair using the Q-learning update rule:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot \left(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \quad (4.4)$$

where $Q(s_t, a_t)$ is the current estimate of the action-value function for state s_t and action a_t , α is the learning rate (controls the weight given to the new information), r_{t+1} is the immediate reward received after taking action a_t in state s_t , γ is the discount factor (controls the importance of future rewards), and $\max_a Q(s_{t+1}, a)$ is the maximum estimated action-value for the next state s_{t+1} over all possible actions a .

- (e) Repeat the training iterations for a certain number of problem cases. In each iteration, update the Q-table based on the rewards and save it to be loaded and used for further training.
- (f) Evaluate the performance on a set of problem cases using the saved Q-table. We use the final result Q-table to choose the action with the maximum Q-value for the current state.

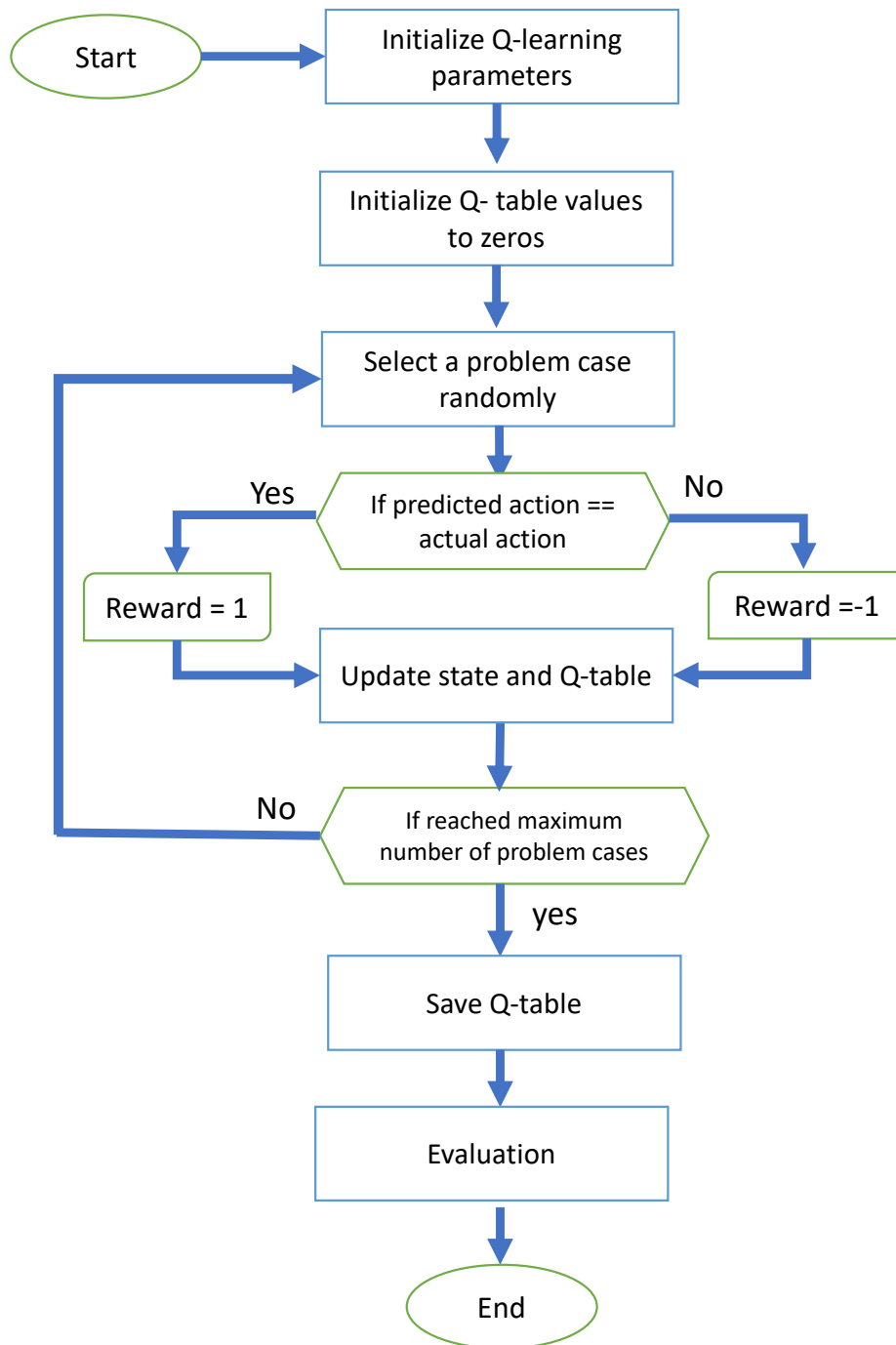


Figure 4.7: Model 2 : Reinforcement Learning

Algorithm 2. Q-Learning Algorithm (QL)
Input: pop_size, maturity_function, learning_rate, discount_factor, Q-table, max_num_problem_case.
Output: Q-table.
1. Set RL parameters. 2. Generate initial population randomly. 3. Initialize Q-table with zeros. 4. Select a random problem case. 5. $num_problem \leftarrow 1$ 6. While $num_problem < max_num_problem_case$ do: 7. Get current state index and action. 8. If predicted_action == actual_action then: 9. reward $\leftarrow 1$. 10. Else: 11. reward $\leftarrow -1$. 12. Get next state. 13. Get max Q-value for next state 14. Update Q-value for current state-action pair $Q(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot (r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$ 15. $num_problem \leftarrow num_problem + 1$. 16. End while. 17. Save Q-table. 18. return updated Q-table.

3. Combination of Learning Classifier System and Reinforcement Learning (PMCR)

Learning classifier system is combined with reinforcement learning to increase the performance of the models. A population of classifiers has evolved using the learning classifier system through the GA, and then they are trained using the Q-learning update rule to increase their decision-making capabilities over time. Figure 4.8 presents the third model procedure, and the pseudocode is given in Algorithm 3.

Algorithm 3. Learning Classifier System and Reinforcement Learning
Input: pop_size, max_num_gen, maturity function, mutation_rate, crossover_rate, max_num_iteration, learning_rate, discount_factor, Q-table, max_num_problem_case.
Output: updated Q-table.
<pre> 1. Set GA and RL parameters. 2. Represent the classifier. 3. Generate initial population. 4. num_problem ← 0 5. While num_problem < max_num_problems do: 6. Select a problem case randomly. 7. Apply bucket brigade algorithm. 8. Initialize temp list to empty list 9. n_gen ← 1 10. While n_gen < max_num_gen do: 11. While pop_size < max_pop_size do: 12. Select parents from population. 13. Produce offspring (crossover) from chosen parents 14. Mutate the resulting offspring. 15. Add new offspring to the temp list 16. pop_size ← pop_size + 1 17. Update population based on temporary list 18. Make temp list empty. 19. End while. 20. n_gen ← n_gen + 1 21. End while. 22. num_problem ← num_problem + 1 23. End while. 24. Return trained classifier from GA. 25. Input the trained classifier from GA to RL. 26. Initialize Q-table with zeros. 27. Select a random problem case. 28. num_problem ← 1 29. While num_problem < max_num_problem_case do: 30. Get current state index and action. 31. If predicted_action == actual_action then: 32. reward ← 1. 33. Else: 34. reward ← -1. 35. Get next state. 36. Get max Q-value for next state. 37. Update Q-value for current state-action pair. 38. $Q(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot (r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$ 39. num_problem ← num_problem + 1. 40. End while. 41. Save Q-table. 42. return updated Q-table. </pre>

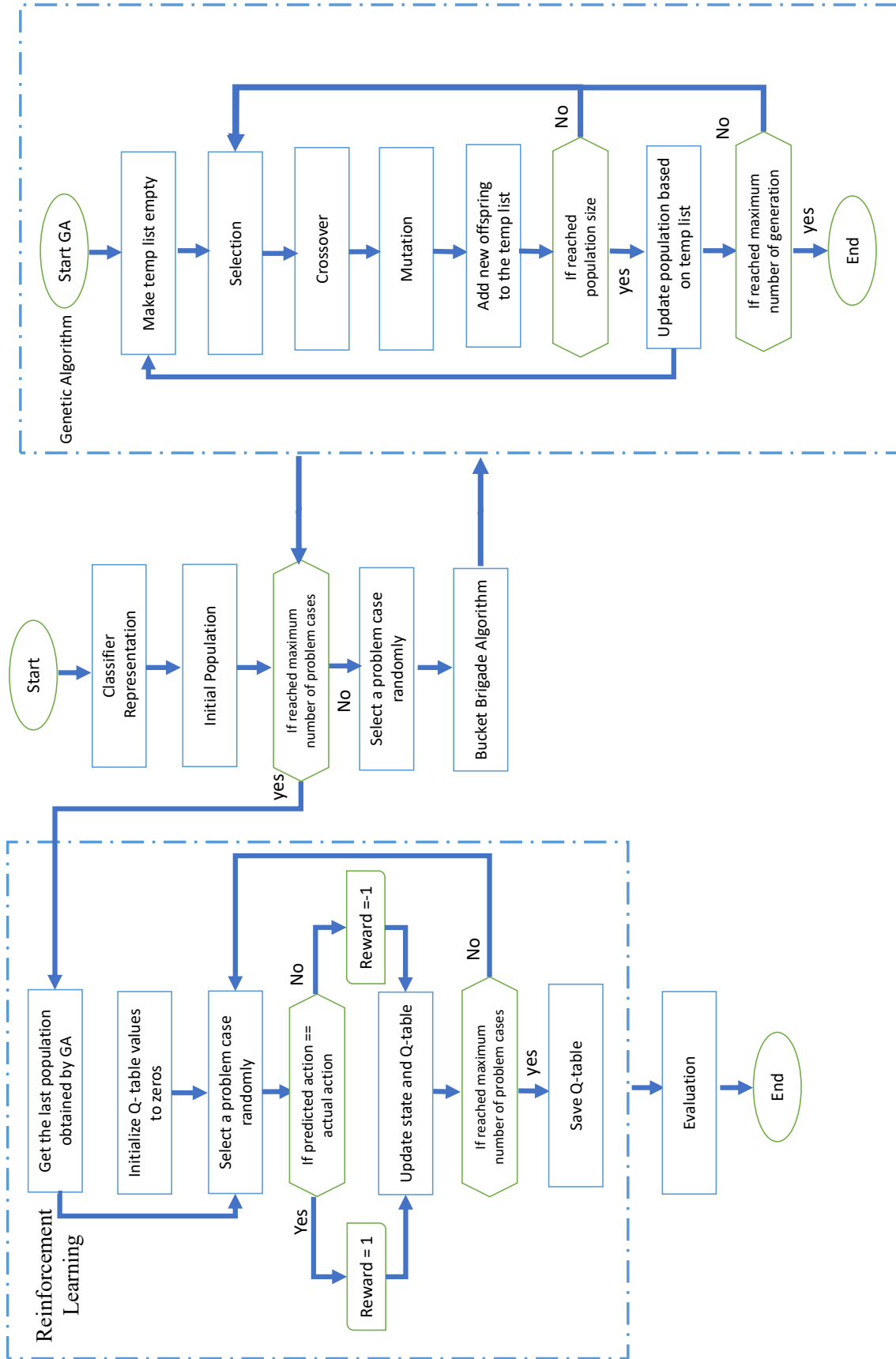


Figure 4.8: Model 3: Combination of learning classifier system and reinforcement learning (PMCR)

4.4 Model Evaluation

In the study, we use different evolution metrics to measure the performance and the effectiveness of the proposed models. These measurements include F1 score, recall, accuracy, and precision. We calculate the accuracy of each proposed model by comparing the predicted actions with the actual actions of each problem case. So, if they are equal, then the winner classifier is rewarded, and the number of corrected prediction problem cases will be increased. A low accuracy score suggests that the model is producing a large percentage of inaccurate predictions, whereas a high accuracy score shows that the model is making a significant proportion of accurate predictions. In addition, we measure the precision of each model which is the ratio of correctly predicted positive actions to the total predicted actions. The third evaluation metric is recall which is the percentage of true positive predictions among all positive cases. The last metric is F1 score, defined as the harmonic mean of precision and recall. The equations of each evaluation metrics were previously presented in Table 3.8.

Figure 4.9 presents the steps we follow to evaluate the performance of the proposed models. The first step we select a random problem case which can be the input to the trained model. Then we compared the actions of the problem case (actual action) with the predicted action. If they are equal, the number of problem cases with correct prediction will increase. But if the predicted action is not equal to the actual action, then we have to continue the iteration and input another problem case. At the end, we will get the total number of problem cases with correct prediction and the total number of all problem cases with incorrect prediction. From the results obtained, we can calculate the accuracy, F1 score, recall, and precision.

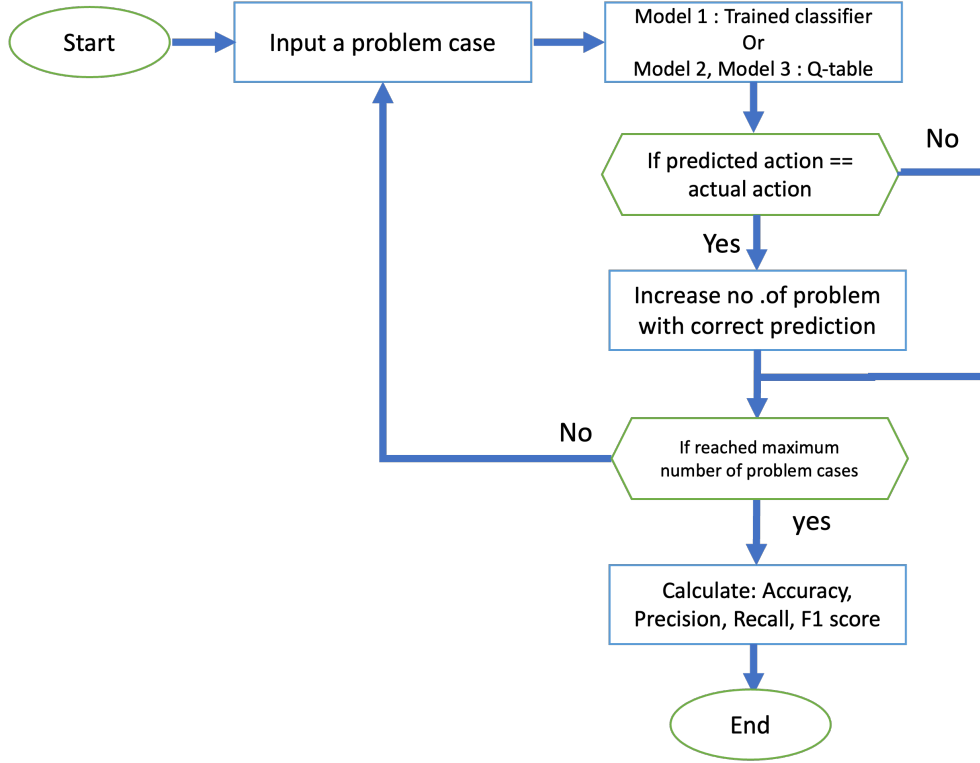


Figure 4.9: The procedures to evaluate the proposed models

4.5 Experimental Results

This section illustrates the experimentation procedures, and the results obtained from the three proposed models. The overall steps to get the results are divided into four prominent cases. First, we get the accuracy of model 1 after applying the classifier system. Second, we find the accuracy obtained after applying reinforcement learning for model 2. Third, we get the classifier system trained population as input to RL and get the results for the third model. Finally, we compare the accuracy results obtained by the three models. In each experiment, we use different parameters such as diverse population size, number of generations, and maximum training cases. Before going into the details of experimentation results, we present two examples to explain the mechanism of the system and the indication of the obtained trained classifiers. Table 4.2 shows an example of the output after

randomly generating an initial population for five conditions. For instance, if the first condition is [2 # 1 1 2 #], that means the learner is completed problem 1 (p1) and problem 5 (p5) correctly and has partially solved problem 3 (p3) and problem 4 (p4). However, if the condition is [1 # 2 # 0 1], that means the learner partially solved problems 1 (p1) and problem 6 (p6) and correctly answered problem 3 (p3) and incorrectly (not solved) problem 4 (p4).

Table 4.3 shows an example of a population of 15 students, their answers for problems, the calculation of maturity function for each problem, and the tactic assigned according to maturity ranges. For instance, student 1 answers the problem as [1 2 1 0 0 2], and the maturity function calculation equals 3.8, which is assigned to tactic 4.

Table 4.2: Example of the initial population

no	conditions	p1	p2	p3	p4	p5	p6	Tactics
	p1 p2 p3 p4 p5 p6							
1	[2 # 1 1 2 #]	CS	DC	PS	PS	CS	DC	T3
2	[# 1 2 # 2 2]	DC	PS	CS	DC	CS	CS	T4
3	[0 # 1 # 1 #]	NS	DC	PS	DC	PS	DC	T7
4	[1 # 2 # 0 1]	PS	DC	CS	DC	NS	PS	T5
5	[2 # 1 0 0 #]	CS	DC	PS	NS	NS	DC	T5

0 (NS) Not solved

1 (PS) Partially solved

2 (CS) Completely solved

(DC) Don't care

Table 4.3: Example of the population of 15 students and their answers to problems

Population	Problems P1 P2 P3 P4 P5 P6	Maturity Function	Tactics
Student 1	[1 2 1 0 0 2]	3.8	T4
Student 2	[1 0 2 0 0 2]	3.4	T4
Student 3	[1 0 1 2 2 1]	4.9	T5
Student 4	[1 2 0 0 1 1]	3.1	T4
Student 5	[1 1 2 0 2 0]	3.7	T4
Student 6	[0 1 2 0 2 1]	4.2	T5
Student 7	[2 0 0 0 0 2]	2.6	T3
Student 8	[0 2 1 2 0 2]	4.8	T5
Student 9	[2 0 1 2 0 0]	2.8	T3
Student 10	[1 1 0 2 0 0]	2.3	T3
Student 11	[2 1 1 2 2 1]	5.8	T6
Student 12	[2 1 1 0 0 1]	2.8	T3
Student 13	[1 1 2 0 2 1]	4.6	T5
Student 14	[1 1 2 2 2 2]	6.9	T7
Student 15	[1 0 1 1 2 1]	4.2	T5
p1: Remembering p2: Understanding p3: Applying p4: Analyzing p5: Evaluating p6: Creating	w1: 0.4 w2: 0.5 w3: 0.6 w4: 0.7 w5: 0.8 w6: 0.9	NS = 0: Not solved, PS = 1: Partially solved, CS = 2: Correctly Solved	
Maturity function = $\sum_{i=1}^n e_i$ Maturity function = $(s_1 \cdot w_1) + (s_2 \cdot w_2) + (s_3 \cdot w_3) + (s_4 \cdot w_4) + (s_5 \cdot w_5) + (s_6 \cdot w_6)$ The range between [0 – 7.80] Higher maturity is 7.80, and lower is 0.			

4.5.1 Results of Model 1 (Learning Classifier System)

For the first experiment to build the first model which depends on the learning classifier system, we set the initial parameters of the classifier system and the GA as shown in Table 4.4. We also used different population sizes and different number of problem cases and generations. In addition, we present True Positives (TP) and False Positives (FP) values. TP indicates the correctly identified positive cases, while FP denotes the instances that were incorrectly classified as positive. The outcomes of the first model are shown in Table 4.5.

Table 4.4: Parameters of the GA Algorithm

Parameter	Values
Population size	200, 500
Number of conditions	6 (possible values: 0, 1, 2, #)
Number of actions	8 (T1 to T8)
Problem cases	400, 500, 1000, 2000, 3000
Mutation rate	0.1
Crossover rate	0.3
Generations	100, 200, 300, 500

We started the experiment with a population size of 200, a number of generations 100, and 500 for the maximum number of training cases. The number of problem cases generated randomly for testing was 40 problem cases. Table 4.5 shows the growth in testing accuracy as we increased the number of generations and maximum number of trainings cases. With a population size of 200 and 400 training cases as input, the testing accuracy only reached 33% after a certain number of generations. However, when we increased the number of generations to 300 and the training cases to 800, the accuracy reached 47%. The best testing accuracy obtained is 68% by correctly predicting 27 out of 40 problems. From Table 4.5, it can be observed that at a certain point when the maximum training cases reached 2000 and

number of generations was 500, the testing accuracy dropped to 50%. Then when the number of generations increased to 600 and the number of training cases increased to 3000, the accuracy went up to 68%. Moreover, the precision and F1 score increased when the number of generations and the maximum number of training cases increased. The precision was 0.20 and increased to 0.68 and the F1 score from 0.33 to 0.81. The result of recall was 1.0 in all cases because the model produced no false negatives or true negatives, which is particularly good.

Table 4.5: Results Obtained from the First Model, the Classifier System

Population Size	No. Gen.	Max Train Cases	Train Acc.	No. Test Cases	TP	FP	Acc.	Prec.	F1
200	100	500	0.22	40	8	32	0.20	0.20	0.33
200	200	400	0.35	40	13	27	0.33	0.33	0.49
500	100	800	0.58	40	13	27	0.33	0.33	0.49
500	200	800	0.86	40	16	24	0.40	0.40	0.57
500	300	800	0.53	40	18	22	0.47	0.47	0.62
500	500	1000	0.57	40	25	15	0.63	0.63	0.77
500	500	1500	0.64	40	24	16	0.60	0.60	0.75
500	500	2000	0.57	40	20	20	0.50	0.50	0.67
500	600	3000	0.60	40	27	13	0.68	0.68	0.81

4.5.2 Results of Model 2 (Reinforcement Learning)

We set the initial parameters of RL as shown in 4.6. The algorithm of reinforcement learning continues to iterate through cycles until the maximum number of iterations are reached which set initially as 500 iterations. In each iteration, a random problem case is selected until it reaches 1000 problem cases.

The results of the second experiment after applying reinforcement learning to build the second model are presented in Table 4.7. The population size used was 500, 0.4 for the learning rate, 0.9 for the discount factor, the

Table 4.6: Parameters of the RL

Parameter	Value
Population size	500
Number of conditions	6 (possible values: 0, 1, 2, #)
Number of actions	8 (T1 to T8)
Problem cases	100, 500, 1000, 2000, 3000
Learning rate	0.4
Discount factor	0.9

number of training problem cases was 1000, and the number of test cases was 40. In addition, we started with a population size of 100 and then 200, but the accuracy percentage was low. Table 4.7 demonstrates that the highest accuracy obtained was 33%, with 13 correct prediction cases out of 40 when we trained the model several times and got the results from the updated Q-table. In addition, the best precision and F1 score were 0.33 and 0.49 respectively. However, the recall was 1 because the false negative and the true negative were zero.

Table 4.7: Results Obtained from the Second Model

	Q-Table	Number of Test Cases	TP	FP	Accuracy	Precision	F1 Score
1	Q-table 1	40	6	34	0.15	0.15	0.26
2	Q-table 2	40	7	33	0.18	0.18	0.30
3	Q-table 3	40	8	32	0.20	0.20	0.33
4	Q-table 4	40	10	30	0.25	0.25	0.40
5	Q-table 5	40	13	27	0.33	0.33	0.49

Table 4.8: Results obtained from the third model using the same problem case set

	Q-Table	Number of Test Cases	TP	FP	Accuracy	Precision	F1 Score
1	Q-table 1	40	7	33	0.18	0.18	0.30
2	Q-table 2	40	8	32	0.20	0.20	0.33
3	Q-table 3	40	13	27	0.33	0.33	0.49
4	Q-table 4	40	15	25	0.34	0.34	0.55
5	Q-table 5	40	17	23	0.43	0.43	0.60
6	Q-table 6	40	20	20	0.50	0.50	0.67
7	Q-table 7	40	24	16	0.60	0.60	0.75

4.5.3 Results of Model 3 - PMCR

For the third model, we tested the correct prediction of problem cases in two ways. In the first method, we examined the same set of problem cases with different Q-tables values each time. In the second method, we tested different set of problem cases with different Q-tables values. Such cases are depicted in Table 4.8 and Table 4.9. As shown in Table 4.8, the model was trained seven times, resulting in different Q-tables. For each Q-table, we tested 40 problem cases. We can observe a significant increase in the number of test cases with correct prediction. The accuracy percentage changed from 18% for the first training to 33% for the fourth training, then to the highest accuracy of 60% in the last training. So, 24 problem cases out of 40 had correct predictions (TP), and 33 problem cases had incorrect predictions (FP). The best precision was 0.60, and the best F1 score was 0.75, with the true positive at 24 and the false positive at 16. However, the recall was 1.0 in all cases because the false and true negative were zero.

For the second method, as illustrated in Table 4.9, the results obtained were interesting when we tested different problem cases each time we trained the model. The number of correct predicted problem cases increased. For the first Q-table, the number of problem cases with correct prediction was

Table 4.9: Results obtained from the third model using different problem cases

	Q-Table	Number of Test Cases	TP	FP	Accuracy	Precision	F1 Score
1	Q-table 1	50	32	18	0.64	0.64	0.78
2	Q-table 2	50	35	15	0.70	0.70	0.82
3	Q-table 3	50	36	14	0.72	0.72	0.84
4	Q-table 4	50	39	11	0.78	0.78	0.88
5	Q-table 5	50	33	17	0.66	0.66	0.80
6	Q-table 6	50	38	12	0.76	0.76	0.86
7	Q-table 7	50	40	10	0.80	0.80	0.89
8	Q-table 8	50	41	9	0.82	0.82	0.90

32 out of 50, reaching 64% accuracy, 0.64 precision, and 0.78 F1 score. However, after several trainings, the accuracy reached 78%, 0.78 precision, and 0.84 F1 score. The highest accuracy obtained was 82% after training the model eight times; the precision also reached 0.82, and the F1 score increased to 0.90. In each training, we saved the Q-table values to be used in the following training.

4.6 Discussion

From the results obtained in the current study and as shown in Figure 4.10, the best model compared to other proposed models is PMCR, the third model that combines the classifier system and reinforcement learning in one hybrid model. The accuracy obtained from this model reached 82% using different input problem cases and 60% using the same set of problem cases as input to the model. Moreover, Figure 4.11 shows clearly the difference between the accuracy obtained from PMCR using different input of problem cases and using the same set of problem cases. This higher evaluation metrics were acquired after the model was trained several times with an updated Q-table in each training phase. In addition, the precision and the F1

score for Model 3 were high reached 0.82 and 0.90 respectively.

It's clear from Figure 4.10 that the third model, PMCR, is a robust architecture for predicting the best tactic from teacher to student and is ideal for use in building pedagogical models in ITS. The findings in this study are consistent with earlier research that has examined the impact of genetic algorithms and reinforcement learning on the development of intelligent systems. The experiment's outcome and results demonstrate that integrating genetic algorithms with reinforcement learning may successfully construct an efficient ITS. This integration can be utilized to design a pedagogical model for intelligent tutoring systems with an accuracy of over 80%.

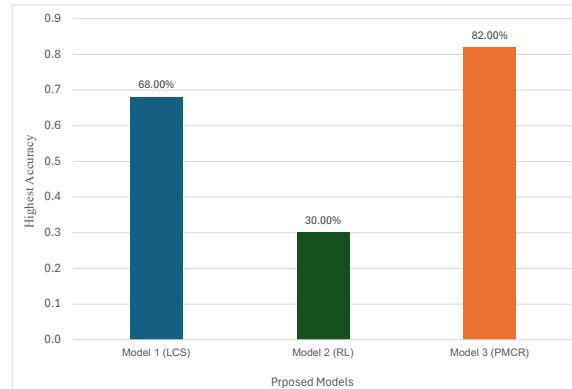


Figure 4.10: Compares the accuracy of the proposed models

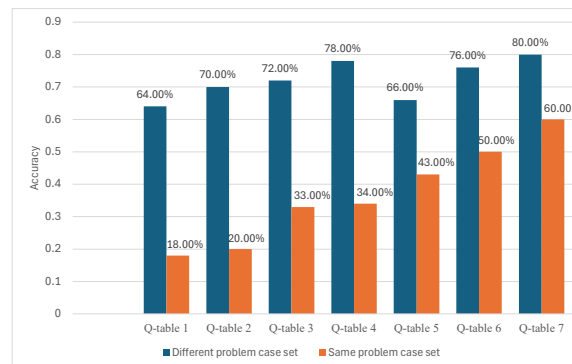


Figure 4.11: Comparison of accuracies obtained using the same test for problem cases vs. a different set for Model 3 (PMCR)

4.7 Summary

This research proposed three models with different architectures. The first model is a learning classifier system that depends on one of the essential evolutionary algorithms, the genetic algorithm. The second model was developed by reinforcement learning and choosing a Q-learning algorithm. The third model, PMCR, integrates the learning classifier system with reinforcement learning to build a hybrid model. We used four evaluation metrics for these models: accuracy, precision, recall, and F1 score. PMCR was the best model compared with the other two models, with accuracy reaching 82%, and F1 score of 0.90.

Furthermore, the first model is better than the second model, with an accuracy of 60%, while the second model reached only 30% accuracy. The findings from this study make several contributions to the current literature. It suggests that PMCR could be suitable approach for developing the pedagogical model in intelligent tutoring systems. PMCR can successfully predict the best tactic from the teacher to his learner when multiple tactics are available.

Chapter 5

Deep Reinforcement Learning in Intelligent Tutoring Systems

5.1 Introduction

5.2 Background

5.3 Proposed Models Architecture

5.4 Model Evaluation

5.5 Experimental results

5.6 Discussion

5.7 Summary

Chapter 5

Deep Reinforcement Learning in Intelligent Tutoring Systems

5.1 Introduction

Currently, the development of reinforcement learning is moving into an advanced chapter toward integration with Deep learning (DL). Traditional reinforcement learning techniques are restricted to simple decision-making problems and have constraints on representing solutions. The development of deep learning overcomes this restriction, and when combined with reinforcement learning, it gives more strength to RL applications (Cao & ZhiMin, 2019).

Deep Q-Network (DQN), Double DQN, and Dueling DQN are three DRL algorithms that have gained significant attention due to their performance in several applications. The DQN algorithm is a milestone in the utilization of deep neural networks for the estimation of the Q-function,

allowing DRL agents to efficiently operate complex and high-dimensional environments (Mnih et al., 2015). Double DQN and Dueling DQN are extensions to the DQN algorithm that solve many problems and optimize the performance of the standard algorithm DQN. Double DQN solves the issue of overestimation of the Q-values (Hasselt, Guez, & Silver, 2016), while the Dueling DQN aims at improving the representation of the value function (Z. Wang et al., 2016). These developments have shown enhanced performance and increased data efficiency in several fields, making them suitable for investigating their potential in the field of intelligent tutoring systems (Fahad Mon, Wasfi, Hayajneh, Slim, & Abu Ali, 2023).

In this chapter, we introduce three approaches based on DQN, Double DQN, and Dueling DQN, which serve as pedagogical models for ITS to predict the proper tactics to be used for enhancing the student's understanding and knowledge grasping.

5.2 Background

This section provides a brief overview of deep reinforcement learning algorithms. Our main focus is on the Deep Q-Network, which is a recent major breakthrough, along with its extensions (Double DQN and Dueling DQN).

5.2.1 Deep Q-network (DQN)

Deep Q-network (DQN) is an extension of the Q-learning algorithm. DQN is a combination of two parts: a category of Artificial Neural Networks (ANNs) known as Deep Neural Networks (DNNs) (Montavon, Samek, & Müller, 2018) which are capable of processing complicated environmental data, and a Reinforcement Learning (RL) part that analyzes these data and determines what should be done next.

DQN was developed in 2015 by Mnih et al.. The main idea of deep Q-network is to use DNN rather than Q-tables to estimate the Q-values, which indicates the future expected reward for performing an action in a specific state. The difference between Q-learning and deep Q-network is clearly shown in Figure 5.1. In Q-learning algorithm, both the current state and action are taken as input to calculate the Q-values for that state-action pair. While in the DQN algorithm, the inputs are the states, and the output are the Q-values associated with all possible actions. The neural network in DQN learns to estimate the Q-value function directly by receiving only the current state as input. This means that DQN is more computationally efficient than Q-learning. It prevents the need of computing the Q-value for every expected action in each condition. Furthermore, in complex problems in a real world scenario, the number of states could be large, which would make the creation of a Q-table very difficult.

Figure 5.2 illustrates the difference between Q-Learning and DQN. An agent at every time step in a state s , takes an action a , receives a reward r and transition to the next state s' from the environment. The agent attempts to learn a policy or map the states to actions in a form of Q-table as shown in Figure 5.2a, in order to maximize the rewards. However, in many decision-making problems with high-dimensional state space, instead of using Q-table, we can utilize artificial neural networks to predict values for actions in a given state as shown in Figure 5.2b. Once the AI agent has accumulated experience, it should be capable of optimizing goals in the form of rewards.

5.2.2 DQN main components

The following are the deep Q-network major components (Phan, Lai, & Lin, 2020):

- Environment: The DQN algorithm involves with an environment which

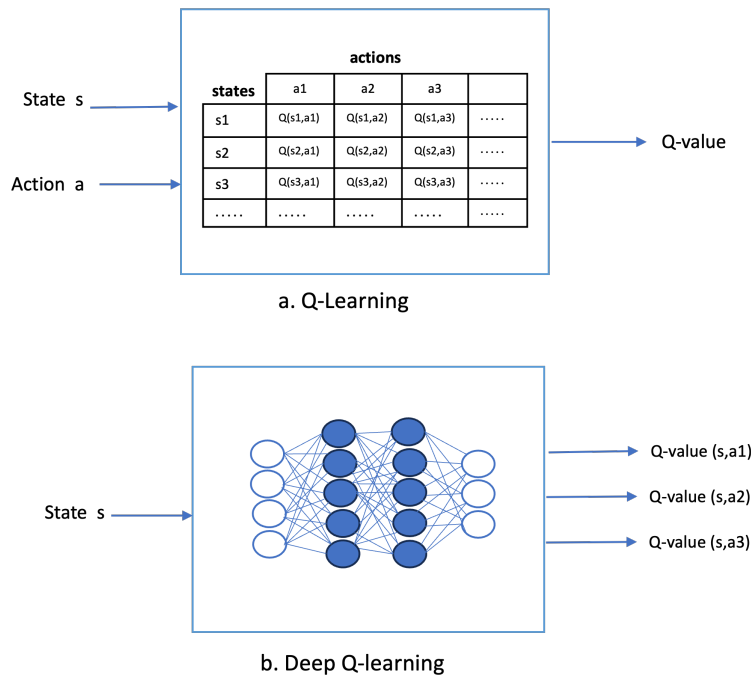


Figure 5.1: Q-Learning vs Deep Q-Learning.

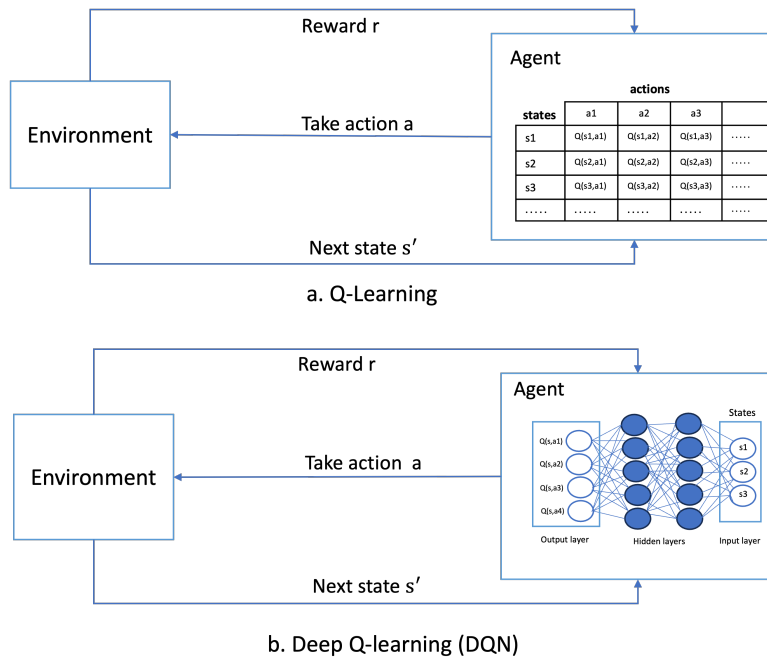


Figure 5.2: Reinforcement learning vs Deep reinforcement learning.

can be a task or problem that the RL agent must solve. The environment provides the agent with observations, rewards for taking suitable actions.

- **Replay Buffer:** The DQN algorithm involves a memory called replay buffer to preserve past experiences. Every experience is a tuple consisting of four elements: state, action, reward, and next state. This tuple represents a single transition from one state to another. Experiences are stored in replay memory for random sampling to be used as input data.
- **Q Neural Network:** The Deep Q-Network approach utilizes a deep neural network (Q-network) to estimate the Q-values associated to each (state, action) combination. The Q network is the agent that learns to generate the optimal state-action value. The architecture of the neural network consists of the following layers (Montavon et al., 2018):
 1. **Input layer:** This layer takes the state representation as input. The dimensionality of the state space determines the size of the input layer.
 2. **Hidden layers:** These layers represent the links between the input state and the Q-values for actions. The number of layers depends on the nature and complexity of the problem. One common activation function used in the hidden layers is ReLU (Rectified Linear Unit).
 3. **Output layer:** The output layer generates Q-values for all possible actions in the specified state. The total number of actions available in the environment sets the dimension of the output layer. The output layer commonly employs a linear activation function because it directly reflects the Q-values

The framework of the neural network used in DQN can vary depending on the specific problem and the complexity of the state space. For example, employing convolutional layers for image-based inputs (Simonyan & Zisserman, 2014) or the integration of recurrent layers for sequential data (Hausknecht & Stone, 2015). Moreover, advanced approaches such as dueling networks or prioritized experience replay (S. Kumar, 2020) can enhance the effectiveness and consistency of the standard DQN in certain situations. However, the main idea of the neural network is to receive the state as input and produce the Q-value for each action as output.

- **Target Neural Network:** The target network (Q-target) is a copy of the primary neural network with fixed parameters. It estimates the target Q-values in DQN. The Q-network parameters are transferred to the target network during the training phase (S. Kumar, 2020).

The DQN approach employs a deep neural network to estimate the Q-values for the current state, a replay memory buffer to store previous experiences, and a target network that is identical to the main network to estimate the Q-values for the next state. During the training phase, the agent employs an exploration process known as epsilon-greedy (Korovesi & Ktona, 2021a) to choose an action. The epsilon value represents the probability of selecting a random action versus selecting the action with the highest predicted Q-value. The agent chooses an action randomly with a probability of epsilon value and selects the action with the greatest Q-value with a chance of $(1 - \epsilon)$. Both networks use random batches from the experience replay to compute Q-values, following performing backpropagation (Kim et al., 2021).

The Q-network trains by computing the loss function using the predicted Q-values, the target Q-values, and the observed reward from the data sample. After several rounds of iterations, the Q-network directly synchronizes

its parameters with the Q-target, which is not involved in the training process. To update the network, the experience replay method (L.-J. Lin, 1992) stores observed transitions for a certain period and then takes a uniform sample from this memory bank. Both the target network and the experience replay significantly enhance the algorithm's performance. Figure 5.3 illustrates the flow structure of DQN algorithm.

Mathematically, a deep Q-network is represented as a neural network that for a given a state s , produces a vector of action values $Q(s, a; \theta)$, where θ represents the Q-network parameters. The Q-target network is identical to the Q-network with parameters denoted by θ^- . However, its parameters are updated by copying the parameters from the Q-network every t step, resulting in $\theta_t^- = \theta_t$. Equation 5.1 defines the target value Y that DQN uses, while equation 5.2 displays the updated Q-values. The loss function calculation in equation 5.3 uses the squared difference between the target Q-value and the predicted Q-value.

$$Y = r + \gamma \max_{a'} Q(s', a'; \theta^-) \quad (5.1)$$

where r represents the reward, γ indicates the discount factor, and $Q(s', a'; \theta^-)$ denotes the Q-value for the next state s' and the optimal action a' with parameter θ^- .

$$Q(s, a) = Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right] \quad (5.2)$$

where r is the reward, γ is the discount factor, α is the learning rate, $Q(s, a)$ is the predicted Q-value for state s and action a , $Q(s', a')$ is the Q-value for the next state s' and the optimal action a' as computed by the target network.

$$\text{Loss} = \left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right)^2 \quad (5.3)$$

where $r + \gamma \max_{a'} Q(s', a')$ is the target Q-value and $Q(s, a)$ is the predicted Q-value.

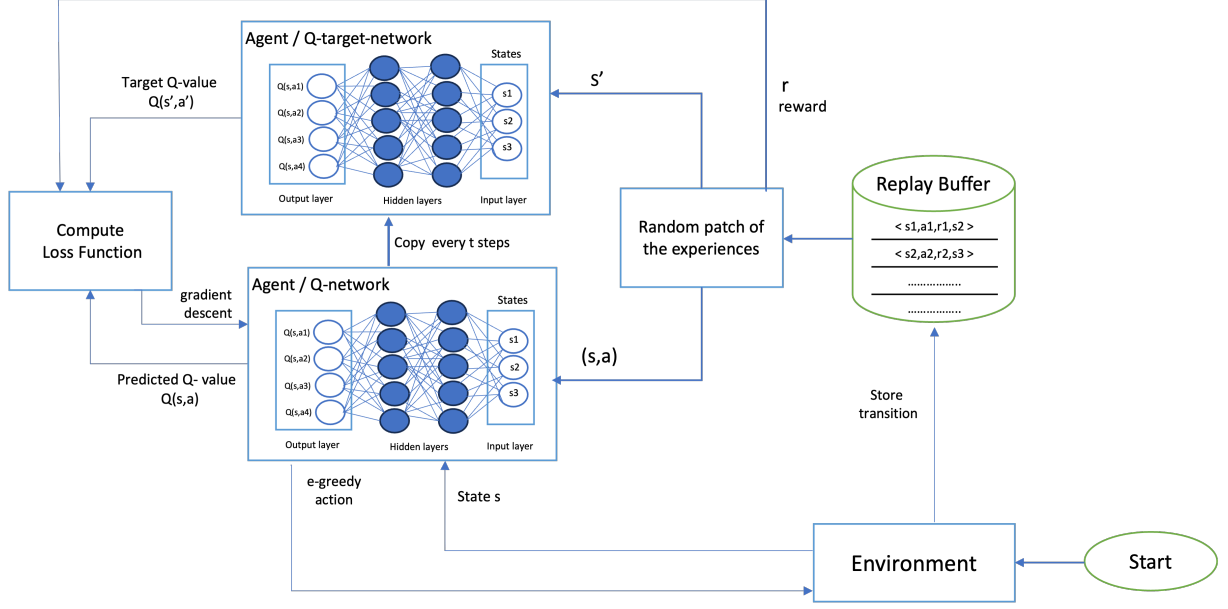


Figure 5.3: DQN flow structure.

To improve the performance and stability of the deep Q-network, two extensions were proposed: the Double DQN (DDQN) (Hasselt et al., 2016) and the Dueling DQN (Z. Wang et al., 2016). The following subsections show the main concept and purpose of these algorithms.

5.2.3 Double DQN

One issue with the DQN algorithm is the overestimation of the Q-values. This particular problem refers to the tendency of the Q-learning algorithm to overestimate the true future rewards, particularly in the beginning stages of learning, when the agent has no knowledge about the environment. Selecting the action with the highest Q-value can lead to overestimating future rewards (Ly, Dazeley, Vamplew, Cruz, & Aryal, 2024). The maximum operator with environmental noise and uncertainty might induce the algorithm to choose the most optimistic Q-value estimate, resulting in an upward bias in Q-value estimations.

Hasselt et al. (2016) introduced Double DQN, a new version of the DQN algorithm, to address the problem of Q-value overestimation by splitting the max term in DQN into action selection and action evaluation. The DQN algorithm employs a single network for both action selection and evaluation, whereas the Double DQN algorithm utilizes two distinct networks: the main neural network for action selection and the target network for action evaluation. So, only the target is changed in Double DQN, as shown in equation 5.4.

$$Y = r + \gamma Q(s', \arg \max_{a'} Q(s', a'; \theta); \theta^-) \quad (5.4)$$

where r is the reward, γ represents the discount factor, $Q(s', a'; \theta^-)$ is the Q-value for the next state s' and the optimal action a' with parameter θ^- .

Initially, the primary neural network (Q-network) selects the optimal next action a' from a range of potential actions. After that, the target neural network (Q-target) evaluates this action to determine its Q-value. Therefore, the evaluation policy replaces the Q-network's weights with the Q-target's, and it regularly updates the Q-target by copying parameters from the Q-network. Using two separate networks reduces the Q-value overestimation problem and improves the final policy (Abdelshiheed et al., 2023).

5.2.4 Dueling DQN

Dueling Deep Q-network architectures were proposed by (Z. Wang et al., 2016). The main idea is to split the network into two streams: the value stream (V), which evaluates the value of a state regardless of actions taken, and the advantage stream (A), which calculates the additional value of a given action from a state compared to other actions. Then, by combining the value and the advantage streams we can get the final Q-values. This change is beneficial, as there are situations where knowing the exact value of each action is unnecessary. In such cases, simply learning about the state-

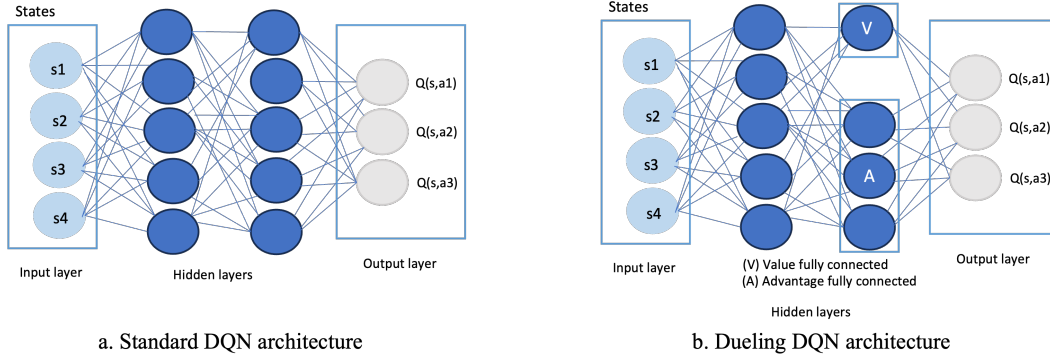


Figure 5.4: Neural network architecture vs Dueling neural network architecture.

value function will be enough. Figure 5.4 presents the differences between the neural networks of standard DQN and dueling DQN.

How Dueling DQN integrates the value and advantage streams is the key innovation. The aggregation is performed in a manner that keeps the relative advantage of each action, while also ensuring that the Q-values are based on the actual value of being present in that state. One common way to do this is to take advantage of each action and deduct its mean or maximum from the advantage stream. In this way, the Q-value is still determined by the value function.

The Q-value estimate may be obtained by using the below two techniques of aggregation: average advantage aggregation, as shown in equation 5.6 and maximum advantage aggregation, as shown in equation 5.7.

$$Q(s, a) = V(s) + \left(A(s, a) - \frac{1}{|A|} \sum_{a'} A(s, a') \right) \quad (5.5)$$

$$Q(s, a) = V(s) + \left(A(s, a) - \max_{a'} A(s, a') \right) \quad (5.6)$$

where $Q(s, a)$ is the Q-value for state s and action a , $V(s)$ is the value function for state s , $A(s, a)$ is the advantage function, $\max_{a'} A(s, a')$ is the maximum advantage value for all possible actions a' , $\sum_{a'} A(s, a')$ is the sum of the advantage values for all possible actions, $|A|$ is the number of possible

actions.

Dueling DQN enhances learning efficiency by separating state value estimation from action advantages, especially when the action does not significantly affect the result. This distinction allows the agent to focus on gaining a deeper understanding of the significance of each state, resulting in effective learning and improved policy-making.

5.3 Proposed Models Architecture

This section explains the main procedure for building a pedagogical model for an intelligent tutoring system. It describes the structure of four different proposed models in detail. The first model is based on a DQN structure. The second and third models are an advanced version of the standard DQN, which solves some problems in DQN (Double DQN and Dueling DQN). Fourth, we present a hybrid approach integrating GA with DQN.

The primary goal of implementing those models is to determine the most efficient strategies for decision-making by identifying the best policies (tactics) to choose from various options. For instance, the instructor, or ITS, may employ these strategies to direct learners according to their responses to certain challenges. This technique assists teachers /ITS in providing students with optimal feedback, thus supporting them in fixing their errors and successfully solving problems. Therefore, various tactics might have a substantial impact on the academic advancement of students.

The following outline is the main procedures for building such models.

1. Determine the models dimensions

Initialize the model dimensions by determining the input and output sizes. The input size is equal to the number of states, and the output size is equal to the number of actions.

- **States:** We defined states as the problems that each student must solve. We assigned six problems to each student. Each answer to the problem can either be 0, 1, or 2. A value of 0 indicates no solution or totally incorrect solution, a value of 1 denotes partial solution, and a value of 2 indicates complete solution.
- **Actions:** We introduced 8 different actions, or we can call them tactics (*T1, T2, T3, T4, T5, T6, T7, and T8*), which are the manners and behaviour for the system that should follow with the student based on his answer.

2. Define the model parameters

After determining the model's input and output sizes, we should define some crucial parameters for each algorithm's training.

- **Number of episodes** is the number of iterations where each episode corresponds to one problem case.
- **Number of steps:** each episode includes a maximum number of steps where the agent interacts with the environment during the training.
- **Epsilon parameters:** the variables that control the balance between exploration and exploitation (epsilon start, epsilon end, and epsilon decay)

3. Define the neural network architecture

The neural network's primary architecture is a feedforward neural network with three fully connected layers:

- The input layer is a fully connected layer that receives the state representation as input. The size of the input layer is determined

by the dimensionality of the state space, which in our situation is 6.

- Hidden layers: Two fully connected layers, each including 64 neurons, are followed by a ReLU activation function.
- The output layer is a fully connected layer that produces the Q-values for each action in the state. The size of the output layer is determined by the total number of actions, which is 8 (tactics from $T1$ to $T8$).

4. **Generate random problem cases**

Generate a random problem case, which is defined as a list of six characters with values '0', '1', or '2'. The problem case presents the input (state) to the neural networks.

5. **Maturity function**

A mathematical equation determines the appropriate tactic for a student's performance. It determines the tactic to be used and compares it with the predicted tactic after training to determine if the model produces the correct tactic or not. (Refer to chapter 4 for more details.)

6. **Model 1: Deep Q-Network (DQN)**

Deep Q-network architecture based on two neural networks (Q-network and Q-target). The Q-network calculates the Q-value in state s , where Q-target calculates the Q-value in the next state s' .

A large number of episodes are used to train the DQN over multiple steps. At each time step, it performs a series of operations as presented in Algorithm 4 and Figure 5.5 More detail about the training procedures of the model in the following:

- For each episode:

o Generate a random problem case (state) and provide it to the environment.

o For each step:

- The agent selects an action a using an epsilon-greedy strategy. The epsilon value starts at 1.0 and gradually decreases to 0.01 over the training process.
- Execute the action in the environment.
- Compute the actual action based on the current state. If the actual action matches the agent selected action (predicted action), then the agent receives a positive reward of 1, otherwise, it receives a negative reward of -1.
- Generate a new problem case (next state)
- The agent stores the experience tuple (current state, action, reward, next state) in the replay buffer.
- Sample random batches from the experience replay memory.
- The neural network model is updated using the stored transition which helps in making the right decisions following these steps:
 - Compute the target Q-values using the target network (Q-target) for the sampled next states as presented previously in equation 5.1.
 - Compute the loss between the predicted Q-values and the target Q-values and the observed rewards as shown in equation 5.3.
- The neural network weights are modified by using the method of backpropagation and stochastic gradient descent.
- The agent's target Q-network is updated every specific number

of steps to match the main Q-network and stabilize the learning process.

- o The total reward for each episode is recorded and the process repeats for the next episode.
- o DQN tests the learnt policy by making environmental decisions after training. The agent chooses the action with the greatest Q-value for a specific condition.

Algorithm 4. Deep Q-Network (DQN)
Input: Input size, output size, maturity function, learning rate, discount factor, number of episodes, number of steps, epsilon start, epsilon end, epsilon decay
Output: Q-values.
1. Initialize Q-network weights θ. 2. Initialize Q-target network with weights $\theta^- = \theta$. 3. Initialize replay memory size 4. For each episode from 1 to number of episodes: 5. Select a random problem case (state) 6. For each step from 1 to number of steps: 7. Select action based on the state using a greedy strategy. 8. Get current state and predicted action 9. Calculate the actual action using the maturity function 10. if predicted action = actual action: 11. reward = 1 12. else: 13. reward = -1 14. Select a random problem case (get the next state) 15. Store the transition in replay memory 16. Sample a batch of transitions from replay memory 17. Calculate the target Q-value: 18. target Q-value = $r + \gamma \max_{a'} Q(s', a'; \theta^-)$ 19. Compute loss: 20. Loss = $(\text{target Q-value} - Q(s, a; \theta))^2$. 21. Copy Q-network weights to Q-target network weights every C steps 22. End for (steps) 23. End for (episodes)

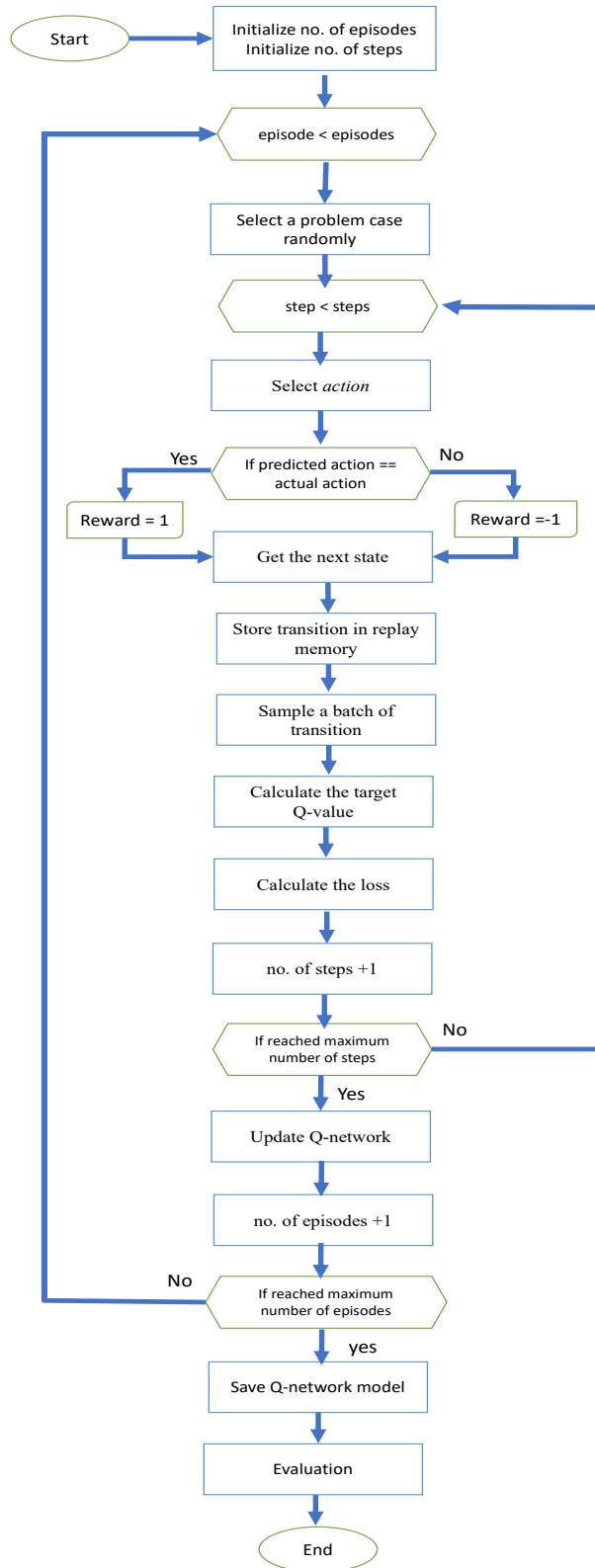


Figure 5.5: Model 1: DQN

7. Model 2: Double DQN

The second proposed model in this study is based on DQN's structure, with some modifications in calculating the target Q-values. Using the maximum Q value as the target value for training the DQN might lead to a bias towards maximizing learning, resulting in overestimation over time. The problem may be resolved using Double Deep Q-Network (Double DQN) algorithm. We selected Hasselt's version (2016), which uses two networks that share weights at regular intervals. The Q-network is used for action selection, whereas the Q-target is used for action evaluation. Algorithm 5 presents the pseudocode of the model.

Algorithm 5. Double DQN
Input: Input size, output size, maturity function, learning rate, discount factor, number of episodes, number of steps, epsilon start, epsilon end, epsilon decay
Output: predicted Q-values
1. Initialize Q-network weights θ. 2. Initialize Q-target network with weights $\theta^- = \theta$ 3. Initialize replay memory size 4. For each episode from 1 to number of episodes do 5. Select a random problem case (state) 6. For each step from 1 to number of steps do 7. action (a) based on main Q-network (Q) using an epsilon-greedy strategy 8. Get current state and predicted action 9. Calculate the actual action using the maturity function 10. if predicted action == actual action: 11. reward = 1 12. else: 13. reward = -1 14. Select a random problem case (get the next state) 15. Store the transition in replay memory 16. Sample a batch of transitions from replay memory 17. Calculate the target Q-values using the target Q-network $Y = r + \gamma Q(s', \arg \max_{a'} Q(s', a'; \theta^-); \theta^-).$ 18. Compute loss (Q-target values – Q-network values) $\text{Loss} = (\text{target Q-value} - Q(s, a; \theta))^2.$ 19. Update Q-network 20. Update target Q-network 21. End for 22. End for

8. Model 3: Dueling DQN

The third model, Dueling DQN, modifies the standard DQN to calculate the Q-value. The Q-value may be decomposed as the combination of $V(s)$, which represents the value of being in a certain state, and $A(s, a)$, which represents the advantage of executing a specific action at that state compared to all other possible actions. The system utilizes two distinct estimators for these two components, merging them via a specific aggregation layer to generate an estimate of $Q(s, a)$. Algorithm 6 shows the pseudocode of the Dueling DQN.

9. Model 4: Combination of GA and DQN

Under this model, we try to measure the effect of the classifier system based on genetic algorithm on deep Q-network by first training the classifier system to generate an initial population of classifiers that represent our rules of conditions and actions, where the conditions are the student performance values for each of the six problems and the action is the tactic that should be adopted. According to Alshaikh and Hewahi (2024), classifier system could improve very well the performance of Q-learning, and the main question here is that, does DQN needs the help of classifier system or not to get high accuracy results. Algorithm 7 provides the pseudocode, while Figure 5.6 depicts the fourth proposed model procedure.

Algorithm 6. Dueling DQN

Input: input size, output size, maturity function, learning rate, discount factor, number of episodes, number of steps, epsilon start, epsilon end, epsilon decay.

Output: predicted Q-values.

1. Initialize Q-network weights θ .
2. Initialize Q-target network with weights $\theta^- = \theta$
3. Initialize replay memory size
4. For episode from 1 to episodes do:
 5. Select a random problem case (state)
 6. For step from 1 to steps do:
 7. Select action based on the state using a greedy strategy
 8. Get current state and predicted action
 9. Calculate the actual action using the maturity function
 10. **if** predicted action == actual action:
 11. reward = 1
 12. **else:**
 13. reward = -1
 14. Select a random problem case (get the next state)
 15. Store the transition in replay memory
 16. Sample a batch of transitions from replay memory
 17. Calculate the predicted Q-values using the Q-network:
$$Q(s, a) = V(s) + \left(A(s, a) - \frac{1}{|A|} \sum_{a'} A(s, a') \right)$$
 18. Calculate the target Q-value:
$$\text{target Q-value} = r + \gamma \max_{a' \in A} Q(s', a'; \theta^-)$$
 19. Compute loss:
$$\text{Loss} = (\text{target Q-value} - Q(s, a; \theta))^2$$
 20. Copy Q-network weights to Q-target network weights every C steps
 21. End for
 22. End for

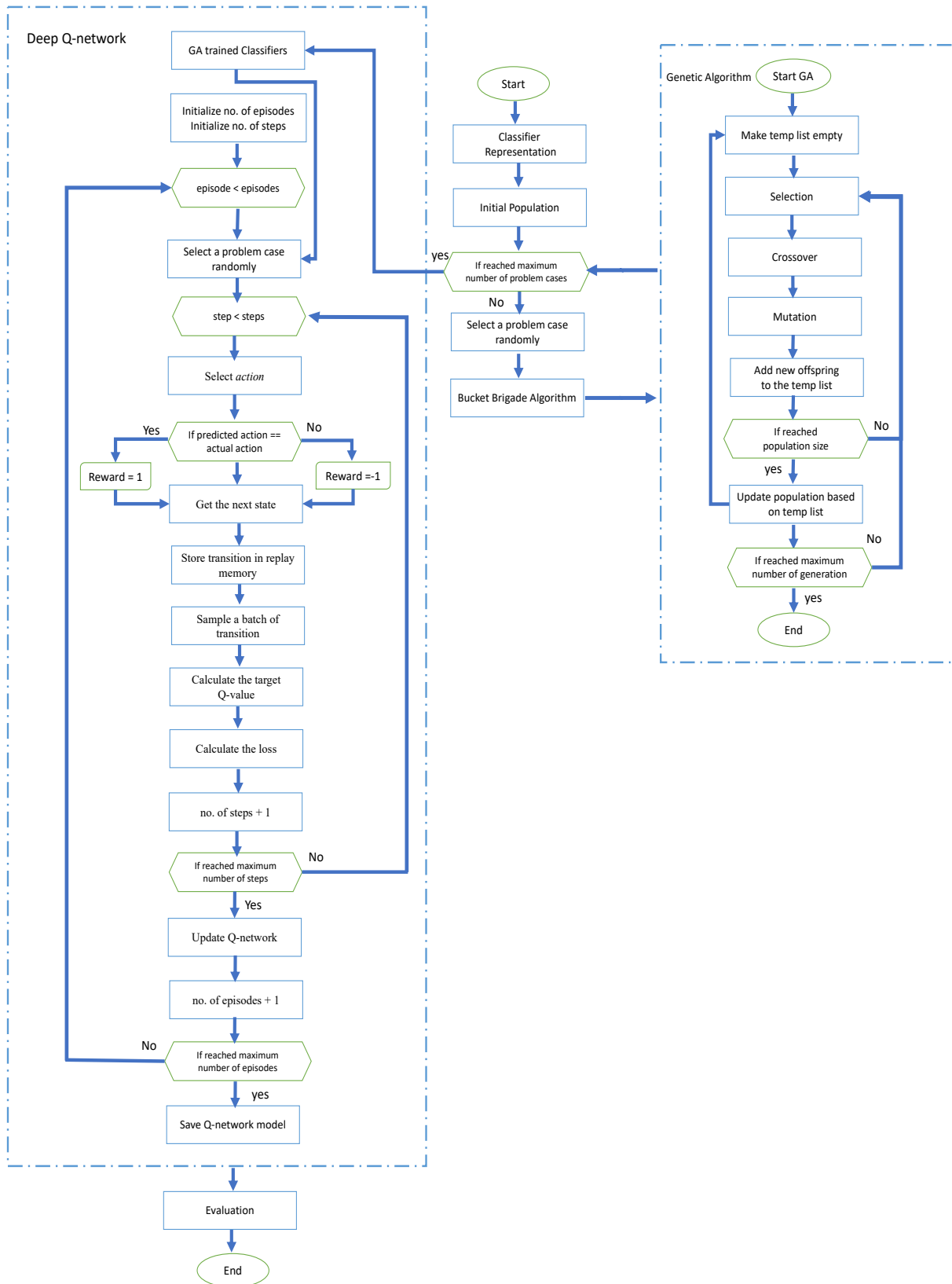


Figure 5.6: Model 4: GA with DQN

Algorithm 7. GA with DQN

Input: population size, max_num_gen, mutation_rate, crossover_rate, max_num_iteration, max_num_problem_case, input size, output size, learning rate, discount factor, number of episodes, number of steps, epsilon start, epsilon end, epsilon decay.

Output: Q-values.

1. Set GA and DQN parameters
2. Represent the classifier
3. Generate initial population
4. num_problem = 0
5. While num_problem < max_num_problem_case do:
6. Select a random problem case
7. Apply bucket brigade algorithm
8. num_gen = 1
9. While num_gen < max_num_gen do:
10. Select parents from population
11. Produce offspring (crossover) from chosen parents
12. Mutate the resulting offspring
13. Add new offspring to the temp list
14. num_gen++
15. End while
16. Update population based on temp list
17. Make temp list empty
18. num_problem++
19. End while
20. Return trained classifier from GA
21. Input the trained classifier from GA to DQN
22. Initialize Q-network weights θ
23. Initialize Q-target network with weights $\theta^- = \theta$
24. Initialize replay memory size
25. For episode from 1 to episodes do:
26. Select a random problem case (state)
27. For step from 1 to steps do:
28. Select action based on the state using a greedy strategy
29. Get current state and predicted action
30. Calculate the actual action using the maturity function
31. **if** predicted action == actual action:
32. reward = 1
33. **else:**
34. reward = -1
35. Select a random problem case (get the next state)
36. Store the transition in replay memory
37. Sample a batch of transitions from replay memory
38. Calculate the target Q-value: $y = r + \gamma \max_{a'} Q(s', a'; \theta^-)$.
39. Compute loss: Loss = (target Q-value - $Q(s, a; \theta)$)²
40. Copy Q-network weights to Q-target network weights every C steps
41. End for
42. End for

5.4 Model Evaluation

This work uses some common evaluation metrics to evaluate the performance of the deep reinforcement learning models. Accuracy, precision, recall and F1 score. These performance metrics add more dimensions to the evaluation of deep reinforcement learning models and hence provide nuanced, informative comparisons. Precision, recall, and F1-score convey the degree of accuracy of a model in identifying and classifying the correct actions relevant in real-world applications. Such metrics, put together with the accuracy, will help in building an idea of strengths, weaknesses, and general characteristics of all models.

5.5 Experimental Results

This section presents the experimental approaches and outcomes of the four proposed models. In each experiment, we used a different number of episodes and steps. Overall, five main scenarios break down the steps to get the final results. Initially, we obtained the results from the standard DQN model. Subsequently, we determined the outcomes from the double DQN model. Following this, we observed the results of the dueling model. Next, we obtained the results from GA based on our previous work (Alshaikh & Hewahi, 2024) and applied them to the DQN model. Finally, we compared the results of these approaches with our prior research (Alshaikh & Hewahi, 2024).

For each experiment, we test the correct prediction of the problems in two different situations. The first situation uses the same collection of problem cases, while the second situation consistently uses new problem cases in each test. Furthermore, we compute the true positive, false positive, recall, precision, F1 score, and accuracy. Nevertheless, the recall outcome for all experiments is 1 due to the absence of negative data.

5.5.1 Set the hyper parameters of the models

In this experiment we used different number of episodes range from 100 to 1000. In addition, we used different number of steps range from 1 step until 30 steps as shown in Table 5.1. At the beginning of the training, the initial epsilon value is typically set to a high value (1.0), so that the agent can explore more by selecting a random action. Over the course of training, the value of epsilon is gradually decreased towards a lower value (0.01) by choosing the action with the greatest predicted Q-value. The decay rate (0.99) determines how quickly the epsilon value is reduced over the training process. A lower decay rate allows the agent to have more exploration while a higher decay rate leads to more exploitation.

Table 5.1: Parameters of DQN

Parameter	Value
Number of Episodes	100 to 1000
Number of Steps	1, 5, 10, 30
Learning Rate (α)	0.001
Discount Factor (γ)	0.9
Epsilon Start	1.0
Epsilon End	0.01
Epsilon Decay	0.99

5.5.2 Results of Model 1 DQN

Table 5.2 and Table 5.3 provide the results obtained from Model 1 using the same set of problem cases and different sets of problem cases, respectively. Initially, we set the parameters of the DQN model, as shown in Table 5.1. The data in Table 5.2 shows that when there are 100 episodes and 5 steps, the number of correct predictions equals 18 out of 40, with an accuracy of 45%, using the same set of problem cases. However, this accuracy drops

to only 35% when using a new set of problem cases, as presented in Table 5.3. However, when there are more than 300 iterations and more than 30 steps, the number of true positives is 32, in contrast to the number of false positives, which is only 6 out of the total 40 problem cases. The precision, F1 score, and accuracy are 0.80, 0.89, and 80%, respectively, when testing the same set of problem cases. Table 5.3 indicates a consistent growth in the performance metrics as the number of episodes and steps improves, while Table 5.3 presents some variations in the performance across different test cases. The findings indicate that as the number of iterations increases, the performance of the model increases with higher precision, F1 score, and accuracy for both tables (Table 5.2 and Table 5.3). In addition, the best performance is achieved at 500 episodes with 30 steps and 1000 episodes with 10 steps.

Table 5.2: DQN - The result obtained using the same set of test cases

No	Episodes	Steps	TP	FP	Precision	F1 Score	Accuracy
1	100	1	6	34	0.15	0.26	0.15
2	100	5	18	22	0.45	0.62	0.45
3	100	10	20	20	0.50	0.67	0.50
4	200	10	34	6	0.85	0.92	0.85
5	300	10	32	8	0.80	0.89	0.80
6	500	10	38	2	0.95	0.97	0.95
7	500	30	40	0	1.00	1.00	1.00
8	1000	10	40	0	1.00	1.00	1.00

Refer to Figure 5.7, which illustrates the relationship between rewards and episodes for the standard DQN algorithm in different cases. The horizontal axis (x-axis) corresponds to episodes, while the vertical axis (y-axis) corresponds to the cumulative awards obtained in each episode. We tested the result in 8 cases with different episodes and steps. In the first case, we trained the model over 100 episodes and 1 step, as shown in Figure 5.7-A. The rewards fluctuate significantly, with large positive and negative spikes,

Table 5.3: DQN - The result obtained using different test cases

No	Episodes	Steps	TP	FP	Precision	F1 Score	Accuracy
1	100	1	8	32	0.20	0.33	0.20
2	100	5	14	26	0.35	0.52	0.35
3	100	10	22	18	0.55	0.71	0.55
4	200	10	32	8	0.80	0.89	0.80
5	300	10	38	2	0.95	0.97	0.95
6	500	10	37	3	0.925	0.96	0.925
7	500	30	40	0	1.00	1.00	1.00
8	1000	10	39	1	0.975	0.987	0.975

indicating a volatile learning process. Overall, it shows the gradual increase of the rewards as the agent learns the task. In cases 2, 3 and 4, as shown in Figure 5.7 (B, C, and D), the fluctuations in rewards are more apparent, with even larger positive and negative spikes compared to case 1. The agent's considerable exploration of the environment led to these dramatic reward changes during the early stages of learning, resulting in unstable reward patterns. Case 5 covers a much larger episode range, up to 300 episodes, so the rewards still exhibit significant fluctuations, but the magnitude of the spikes is reduced compared to the earlier cases 2, 3, and 4. In the following cases, from case 5 to case 8, the range of episodes extends to 1000 iterations that show rewards of volatility with large spikes both in the positive and negative directions. As the agent gains more experience, cases 5 through 8, the patterns of rewards become more stable. The wider episode range in the later cases provides a more comprehensive view of the agent's learning dynamics, revealing that the initial fluctuation gives way to more consistent performance improvements over time.

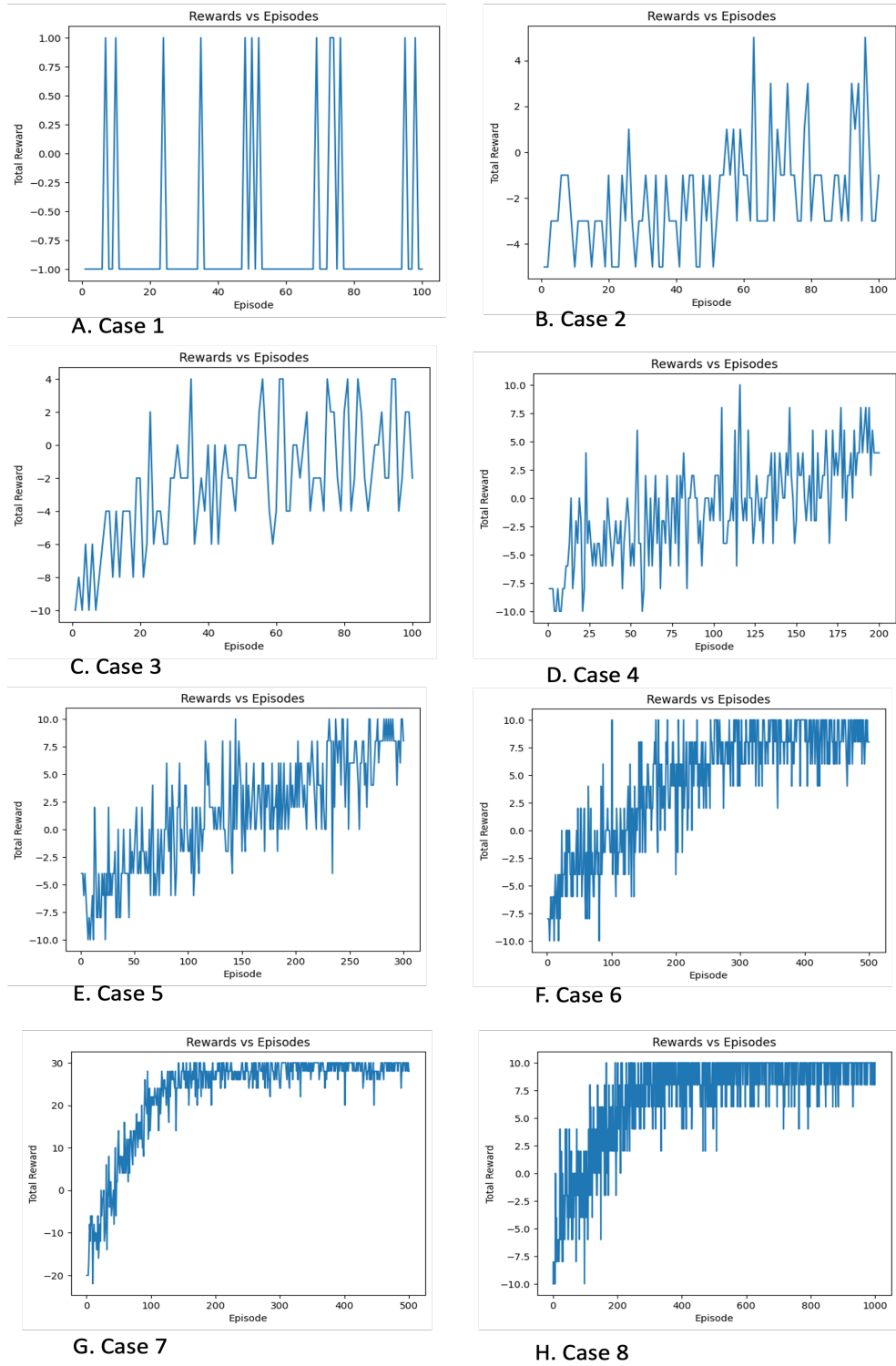


Figure 5.7: Rewards vs Episodes for the standard DQN algorithm.

5.5.3 Results of Model 2 : Double DQN

Tables 5.4 and 5.5 present the results obtained from the double DQN model. The testing began with 100 episodes and one step, then progressed until it reached 1000 episodes and 10 steps. We used the same set of problem cases as observed in Table 5.4, as well a different set of problem cases as presented in Table 5.5. The total number of problem cases tested is 40. As shown in Tables 5.4 and 5.5, there is a clear trend toward improving the performance of the Double DQN as the number of episodes and steps increases. For example, as illustrated in Table 5.4, with 100 episodes and 5 steps, the model achieves a precision of 0.45, a test accuracy of 45%, and an F1 score of 0.62. The number of correct predictions is 18 out of 40. Despite this, after running 200 episodes and 10 steps, the model attains a precision of 0.75, an accuracy of 75%, and an F1 score of 0.86.

Table 5.4: Double DQN-The result obtained using the same set of test cases

No	Episodes	Steps	TP	FP	Precision	F1 Score	Accuracy
1	100	1	8	32	0.20	0.33	0.20
2	100	5	18	22	0.45	0.62	0.45
3	100	10	28	12	0.70	0.82	0.70
4	200	10	30	10	0.75	0.86	0.75
5	300	10	28	12	0.70	0.82	0.70
6	500	10	38	2	0.95	0.97	0.95
7	500	30	39	1	0.98	0.99	0.98
8	1000	10	40	0	1.00	1.00	1.00

Table 5.5: Double DQN-The result obtained using different test case

No	Episodes	Steps	TP	FP	Precision	F1 Score	Accuracy
1	100	1	7	33	0.16	0.30	0.16
2	100	5	19	21	0.48	0.64	0.48
3	100	10	21	19	0.53	0.69	0.53
4	200	10	28	12	0.70	0.82	0.70
5	300	10	31	9	0.78	0.87	0.78
6	500	10	39	1	0.98	0.99	0.98
7	500	30	39	1	0.98	0.99	0.98
8	1000	10	40	0	1.00	1.00	1.00

5.5.4 Results of Model 3 Dueling DQN

For model 3, we follow the same procedure as the previous models. Tables 5.6 and 5.7 present the results. For the first case with 100 episodes and 1 step, the dueling DQN model correctly identified 8 correct predictions out of a total of 40 (8 is TP, and 32 is FP) with a precision of 0.2, an F1 score of 0.33, and an accuracy of 20%, as shown in Table 5.7. However, in the second case, when we increase the number of steps to 5, the model's performance improves gradually, resulting in a precision of 0.4, an accuracy of 40%, and an F1 score of 0.57. Furthermore, Table 5.6 indicates that precision, F1 score, and accuracy increase as the number of episodes increases. For instance, with 200 episodes and 10 steps, the model achieves a precision of 0.63, an F1 score of 0.77, and an accuracy of 63%. Nevertheless, with 500 episodes and 10 steps, the model reaches a precision of 0.98, an accuracy of 98%, and an F1 score of 0.97, indicating excellent performance.

In addition to the results obtained from the Dueling DQN using the same set of problem cases, we employed different sets of problem cases for each test, as presented in Table 5.7. Despite the minor variations in the individual metrics, the overall trend remains consistent as the number of episodes and steps increases, the dueling DQN model's performance improves across all

Table 5.6: Dueling DQN-The result obtained using the same set of test case

No	Episodes	Steps	TP	FP	Precision	F1 Score	Accuracy
1	100	1	8	32	0.20	0.33	0.20
2	100	5	16	24	0.40	0.57	0.40
3	100	10	21	19	0.53	0.69	0.53
4	200	10	25	15	0.63	0.77	0.63
5	300	10	32	8	0.80	0.89	0.80
6	500	10	39	1	0.98	0.99	0.98
7	500	30	38	2	0.95	0.97	0.95
8	1000	10	40	0	1.00	1.00	1.00

Table 5.7: Dueling DQN-The result obtained using different test cases

No	Episodes	Steps	TP	FP	Precision	F1 Score	Accuracy
1	100	1	9	31	0.23	0.37	0.23
2	100	5	16	24	0.40	0.57	0.40
3	100	10	22	18	0.55	0.71	0.55
4	200	10	24	16	0.60	0.75	0.60
5	300	10	33	7	0.83	0.90	0.83
6	500	10	37	3	0.93	0.96	0.93
7	500	30	38	2	0.95	0.97	0.95
8	1000	10	40	0	1.00	1.00	1.00

the metrics, including precision, F1 score, and test accuracy.

5.5.5 Results of Model 4 (GA - DQN)

The results obtained from Model 4, which is based on the combination of GA and DQN, are shown in Tables 5.8 and 5.9, using the same set of problem cases and different sets of problem cases, respectively. The proposed model was trained with various numbers of episodes, ranging from 100 to 1000 episodes, and steps, ranging from 1 to 30 steps. However, the number of test problem cases with correct predictions ranges from 4 to 9 out of the

total 40 test cases. This means that the model is not able to correctly predict the outcomes of test cases. Furthermore, the precision values range from 0.13 to 0.30, as shown in Table 5.8, showing that the model has a low ratio of true positives to the total of true positives and false positives. The F1 scores vary between 0.22 and 0.46. In addition, the accuracy values range from 13% to 30%. The low levels of precision, F1 score, and accuracy indicate that the GA-DQN model is performing poorly and is ineffective in predicting outcomes.

Table 5.8: GA+DQN - The result obtained using the same set of test cases

No	Episodes	Steps	TP	FP	Precision	F1 Score	Accuracy
1	100	1	9	31	0.23	0.37	0.23
2	100	5	8	32	0.20	0.33	0.20
3	100	10	9	31	0.23	0.37	0.23
4	200	10	8	32	0.20	0.33	0.20
5	300	10	4	36	0.10	0.18	0.10
6	500	10	7	33	0.18	0.30	0.18
7	500	30	6	34	0.15	0.26	0.15
8	1000	10	8	32	0.20	0.33	0.20

Table 5.9: GA+DQN -The result obtained using different test cases

No	Episodes	Steps	TP	FP	Precision	F1 Score	Accuracy
1	100	1	12	28	0.30	0.46	0.30
2	100	5	12	28	0.30	0.46	0.30
3	100	10	12	28	0.30	0.46	0.30
4	200	10	8	32	0.20	0.33	0.20
5	300	10	7	33	0.18	0.30	0.18
6	500	10	7	33	0.18	0.30	0.18
7	500	30	5	35	0.13	0.22	0.13
8	1000	10	10	30	0.25	0.40	0.25

Table 5.10 presents the difference of the accuracy results across the four proposed models. The comparison is based on a different set of problem cases with varying numbers of training episodes and steps. The DQN, Double DQN, and Dueling DQN models generally show an improvement in accuracy values as the number of episodes and steps increases, indicating their ability to learn and enhance performance with more training. Nevertheless, the fourth model (GA with DQN) has a low level of accuracy, fluctuating between 18% and 30%, suggesting that the addition of genetic algorithm components does not significantly increase the performance of the DQN model to predict the correct tactics. The DQN model reaches a near perfect performance (100%) of accuracy at 500 episodes and 30 steps. While the Dueling DQN and Double DQN models reach 100% accuracy at 1000 episodes and 10 steps, In contrast, Model 4 reaches only 30% accuracy at 100 episodes.

Table 5.10: Comparison of the accuracy of the four proposed models using different set

No	Episodes	Steps	Model 1 DQN	Model 2 Double DQN	Model 3 Dueling DQN	Model 4 GA + DQN
1	100	1	0.20	0.16	0.23	0.30
2	100	5	0.35	0.48	0.40	0.30
3	100	10	0.55	0.53	0.55	0.30
4	200	10	0.80	0.70	0.60	0.20
5	300	10	0.95	0.78	0.83	0.18
6	500	10	0.925	0.98	0.93	0.18
7	500	30	1.00	0.98	0.95	0.13
8	1000	10	0.975	1.00	1.00	0.25

Regarding precision and F1 score, Table 5.11 compares the results between all models. At 100 episodes and 1 step, the best model is Model 4, with 0.3 precision and 0.46 F1 score. However, when we increase the number of steps to 5 steps, the Double DQN shows the highest precision and F1 score, which are 0.48 and 0.64, respectively. As we increase the number

Table 5.11: Comparison of the precision and F1 score for the four proposed models

No	Episodes	Steps	DQN		Double DQN		Dueling DQN		GA + DQN	
			Precision	F1 Score	Precision	F1 Score	Precision	F1 Score	Precision	F1 Score
1	100	1	0.20	0.33	0.16	0.30	0.23	0.37	0.30	0.46
2	100	5	0.35	0.52	0.48	0.64	0.40	0.57	0.30	0.46
3	100	10	0.55	0.71	0.53	0.69	0.55	0.71	0.30	0.46
4	200	10	0.80	0.89	0.70	0.82	0.60	0.75	0.20	0.33
5	300	10	0.95	0.97	0.78	0.87	0.83	0.90	0.18	0.30
6	500	10	0.93	0.96	0.98	0.99	0.93	0.96	0.18	0.30
7	500	30	1.00	1.00	0.98	0.99	0.95	0.97	0.13	0.22
8	1000	10	0.975	0.987	1.00	1.00	1.00	1.00	0.25	0.40

of iterations and steps, the DQN, Dueling DQN, and Double DQN models show a higher level of performance compared to the fourth model. In addition, the Double DQN generally outperforms the other models, especially in higher training complexity settings with 500 episodes and above. At 500 episodes and 10 steps, Double DQN has a 5% higher precision, and a 3% higher F1-score compared to the standard DQN Model 1. However, when we increased the steps to 30, the DQN model achieved the highest precision with an F1 score of 1.0, outperforming the Dueling DQN and Double DQN. At the highest training level of 1000 episodes and 10 steps, both the Dueling DQN and Double DQN models achieved perfect precision with an F1-score of 1.0, outperforming the standard DQN. This demonstrates the ability of the more advanced architectures to fully leverage the additional training data and steps to achieve optimal performance.

Table 5.12 presents the evaluation metrics for several proposed machine learning models. Learning Classifier System (LCS), Reinforcement Learning (RL) and PMCR (LCS with RL) models were proposed previously by Alshaikh and Hewahi (2024). It is apparent from Table 5.12 that the deep reinforcement learning algorithms, including DQN, Dueling DQN, and Double DQN models, reach the highest level of performance with perfect ac-

Table 5.12: Comparison of the evaluation metrics for several proposed machine learning models

Models	Accuracy %	Precision	F1 Score
Learning Classifier System (LCS)	68%	0.68	0.81
Reinforcement Learning (RL)	33%	0.33	0.49
PMCR (LCS with RL)	82%	0.82	0.90
Deep Q-Networks (DQN)	100%	1.00	1.00
Dueling DQN	100%	1.00	1.00
Double DQN	100%	1.00	1.00
Genetic Algorithm (GA) with DQN	30%	0.30	0.46

curacy, precision, and an F1 score of 1.0 compared to other models. The PMCR model (Alshaikh & Hewahi, 2024), which is based on the combination of the learning classifier system and reinforcement learning using the Q-learning algorithm, performs well, with an accuracy of 82%, a precision of 0.82, and an F1 score of 0.90. The learning classifier system (Alshaikh & Hewahi, 2024) has a moderate level of performance, with an accuracy of 68%, a precision of 0.68, and an F1 score of 0.81. On the contrary, reinforcement learning has the lowest performance among the proposed models, with an accuracy of 33%, a precision of 0.33, and an F1 score of 0.49. On the other hand, the integration of genetic algorithms with DQN performs worst, with 30% accuracy, 0.30 precision, and a 0.46 F1 score. Figure 5.8 shows clearly the different evolution metrics used between various proposed machine learning models.

5.6 Discussion

The core significance of this research is to examine the effectiveness of deep reinforcement learning techniques in predicting the appropriate feedback from teachers to students in intelligent tutoring systems. The results of the study demonstrate four different proposed models. Each of them had their

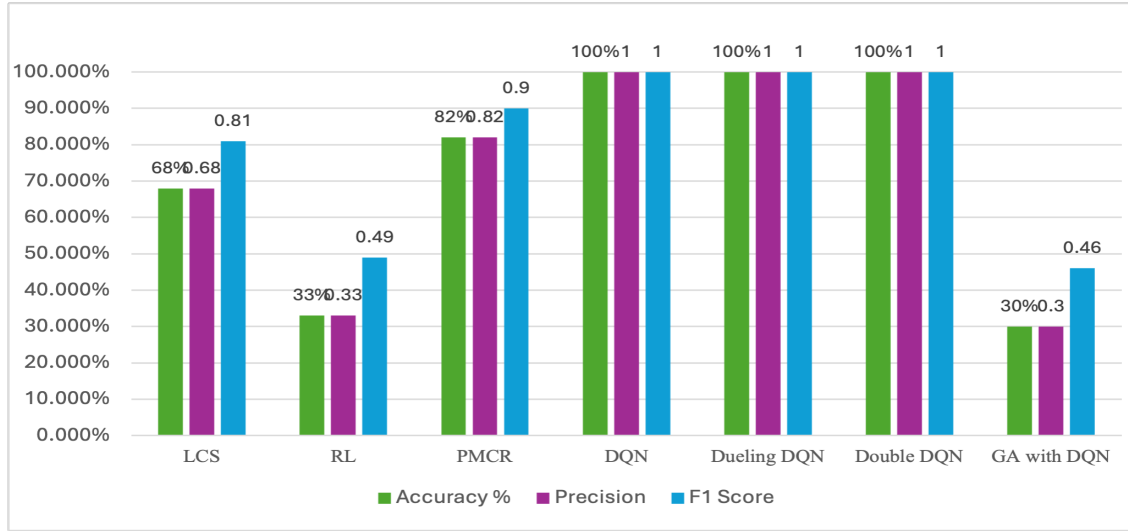


Figure 5.8: Comparing the evaluation metrics for several machine learning models

own strengths and trade-offs, depending on the specific training configuration.

In general, the DQN, Double DQN, and Dueling DQN, as the number of episodes and steps goes up, they show a trend toward improvement in accuracy, precision, and F1 score. The optimal configuration for DQN is 500 episodes with 30 steps, achieving 100% accuracy, while the Double and Dueling DQN reach the highest performance at 1000 episodes with 10 steps. Although the Double DQN model consistently outperforms the DQN and Dueling DQN models at higher training complexities. This suggests that the second proposed model (Double DQN) architecture is better suited to leverage the additional training data and complexity to continue improving its performance at higher training levels. The key advantage of Double DQN seems to its ability to more effectively minimize the overestimation bias inherent in standard DQN, allowing it to maintain high accuracy even as the training becomes more extensive.

On the contrary, the fourth model, which is based on the integration of genetic algorithms and DQN, shows a different pattern of results compared to other algorithms. The results were not encouraging, the accuracy is gen-

erally low, with a maximum of 30% even with more episodes and steps. However, the GA-DQN model has lower precision and F1 score values, and they do not improve significantly with training complexity. Moreover, the proposed deep reinforcement learning models have higher performance outcomes compared with standard reinforcement learning. This suggests that utilizing deep learning through the neural networks empowers the model to optimize the most effective strategy from teachers to the learners.

The findings from these experiments, highlight the superior performance of DQN, Dueling DQN, and Double DQN models compared to the GA-DQN model. The poor performance and the lack of convergence of the GA-DQN model indicate that the genetic algorithm approach is not an effective modification approach for improving the DQN model's efficiency. However, the integration of Q-learning with genetic algorithm had a promising result as proven by Alshaikh and Hewahi (2024). Due to the complexity of the of neural network structure comparing to the simplicity of Q-learning, the GA can easily leverage QL to more effectively explore the search space and find optimal solutions. In contrast, the non-linear optimization landscape of DQN could cause difficulties for the GA to navigate successfully, leading to lower performance. From all these, we can conclude that deep reinforcement learning techniques are successful in building a pedagogical model of an intelligent tutoring system and helping teachers give their students the appropriate feedback according to their answers to problems.

5.7 Summary

In this work, we performed a comparison and analysis of the performance of four different deep reinforcement learning architectures, including DQN, Double DQN, Dueling DQN, and GA-DQN, with different levels of training complexity. The main goal of building these models is to help ITS selects

the appropriate feedback to the students.

During the initial training phase, with 100 episodes, the Dueling DQN model outperforms the other models. However, when the training progressed to a moderate level at 200 to 300 episodes, the standard DQN model learned robust representations and policies, reaching high accuracy. After 500-1000 episodes, Double DQN and Dueling DQN achieved 100% data efficiency, whereas DQN performed well but was outperformed by advanced approaches. In contrast, the hybrid approach, combining GA with DQN, consistently underperformed the other three standalone models across all training configurations.

These results are consistent with the literature on the comparative advantages of these deep RL algorithms. Previous studies indicated the enhanced capacities of Dueling DQN in exploration and generalization (Z. Wang et al., 2016), and the improved efficiency and stability of Double DQN have also been documented (Hasselt et al., 2016).

Chapter 6

Conclusion and Recommendations

Chapter 6

Conclusion and Recommendations

The field of education is expanding, and with advanced technologies being developed, it is expected to change within the following years. Machine learning for automated learning tasks is a dynamic study domain requiring further advancement in the education sector. This thesis presented different architectures based on machine learning and deep learning algorithms to build intelligent tutoring systems.

In chapter 3, we concentrated on student modeling. We proposed three models using CNNs and fully connected layers to predict student success in the year. Model 1 used CNN as the model input; Model 2 used fully connected layers, followed by CNN; and Model 3 used CNN, followed by fully connected layers, followed by CNN. We used two different datasets, and we applied two feature selection techniques: decision tree and principal component analysis. This experiment answers research question 1 (RQ1) by demonstrating the impact of integrating CNN architecture in ITS student modeling. It proved that the proposed models based on CNN are highly ef-

fective in predicting student performance at early stages of the year, which is crucial for preventing failure and enabling personalized teaching and learning strategies.

Chapter four of the thesis focused on ITS pedagogical model. We proposed three models with different structures. The first model is a learning classifier system that depends on one of the essential evolutionary algorithms, the genetic algorithm. The second model was developed by reinforcement learning and choosing a Q-learning algorithm. The third model, PMCR, integrates the learning classifier system with reinforcement learning to build a hybrid model. PMCR outperformed the other two models, with an accuracy of 82%, precision of 0.82, and F1 score of 0.90. The second experiment answers research questions (RQ2, RQ3, RQ4). It proved that the classifier system based on GA has a good performance on the pedagogical model in ITS and better performance than RL. Furthermore, implementing a hybrid approach to a classifier system based on GA and RL improves the performance of the pedagogical model.

In chapter five, we performed a comparative evaluation of the effectiveness of four distinct DRL architectures, namely DQN, Double DQN, Dueling DQN, and GA-DQN, at different levels of training complexity. The primary goal of building these models is to assist the ITS in selecting the appropriate feedback for the learners. In the first phases of training, including 100 episodes, the Dueling DQN model demonstrates superior performance compared to the other models. Nevertheless, when the training advanced to a reasonable level with 200 to 300 episodes, the standard DQN model achieved a high degree of accuracy. After 500-1000 episodes, both Double DQN and Dueling DQN achieved a data efficiency of 100%, while DQN performed well but was outperformed by advanced methods. The third experiment addresses research question 5 (RQ5) by demonstrating that constructing diverse deep reinforcement learning architectures influences the

pedagogical model's performance differently and yields superior outcomes compared to experiment 2, which relied on a classifier system, RL, and a hybrid of a classifier system and RL (PMCR).

The findings of all experiments done in this thesis indicate the effectiveness of ML and DL algorithms in the development of an ITS that enhance the learning outcomes and help student in their education journey.

Further research might examine different types of machine learning algorithms or combinations with other AI techniques, such as natural language processing or computer vision, which may give personalized help and instruction to learners. The development of adaptive pedagogical models in the future can be of assistance in refining and strengthening existing models. Implementing this will guarantee that they are efficient, ethical, and able to adapt to various educational environments.

Additional studies should further investigate the performance of these techniques in more varied environments and task domains, despite this work providing valuable insights into the comparative advantages of various DRL architectures. Moreover, it would be very important to explore the combination of the dueling DQN and Double DQN architectures for the purpose of leveraging the strengths of both approaches.

Some areas that require further development and research include formative automation assessment, which give students more personalized feedback by matching student data to teacher-specified assessment. This enables creation of interactive reports and graphs (Oliveira Neto & Nascimento, 2012). Additionally, automatically generated rules to answer questions from students and help other students in the same course. This method often relieves the instructor to add rules manually into the system. Also, more research should be done to identify why students make a mistake so that they can be avoided rather than only corrected them. Automatic conversational systems are one of the best technologies used machine learning in education,

and more attention moves to it. These virtual assistants help students and teachers by making information more accessible. It simulates human and human interaction. Colleges can use it to address student inquiries about work deadlines, class places, and test dates. Not only for students to get prompt responses, but college employees are often relieved of answering redundant questions. Cognii Virtual Learning Assistant is an example of a virtual assistant that is specifically developed and designed for educational conversations (Pedro, Subosa, Rivas, & Valverde, 2019).

References

- Abdelshiheed, M., Hostetter, J. W., Barnes, T., & Chi, M. (2023). Leveraging deep reinforcement learning for metacognitive interventions across intelligent tutoring systems. In *International Conference on Artificial Intelligence in Education* (pp. 291–303). Springer Nature Switzerland.
- Abu-Dawwas, W. A. (2012). Enhancing the tutor model of intelligent tutoring systems. *American Academic & Scholarly Research Journal*, 4(4), 28–34.
- Abu-Naser, S. S. (2012). Predicting learners performance using artificial neural networks in linear programming intelligent tutoring system. *International Journal of Artificial Intelligence and Applications (IJAIA)*, 3(2).
- Abyaa, A., Idrissi, M. K., & Bennani, S. (2016). Towards an adult learner model in an online learning environment. In *2016 15th International Conference on Information Technology Based Higher Education and Training (ITHET)* (pp. 1–5). IEEE.
- Akhai, S., Karia, R., Mahadik, A., Shah, A., & Gidwani, M. (2017). *Automated performance evaluation system*. IJARIT.
- Akour, M., Alsghaier, H., & Al Qasem, O. (2020). The effectiveness of using deep learning algorithms in predicting students achievements. *Indonesian Journal of Electrical Engineering and Computer Science*, 19(1), 387–393.

- Akyuz, Y. (2020). Effects of intelligent tutoring systems (its) on personalized learning (pl). *Creative Education*, 11, 953-978. doi: 10.4236/ce.2020.116069
- Albuquerque, D., Tarrataca, L., Brandão, D., & Coutinho, R. (2022). A genetic algorithm with flexible fitness function for feature selection in educational data: Comparative evaluation. *Journal of Information and Data Management*, 13(3).
- Alday, R. B. (2018). Bayesian networks in intelligent tutoring systems as an assessment of student performance using student modeling. In *Proceedings of the 2018 2nd international conference on algorithms, computing and systems* (pp. 119–122).
- Alhazmi, E., & Sheneamer, A. (2023). Early predicting of students performance in higher education. *IEEE Access*, 11, 27579–27589.
- Alkhatlan, A., & Kalita, J. (2018). Intelligent tutoring systems: A comprehensive historical survey with recent developments. *arXiv preprint arXiv:1812.09628*.
- AlMahamid, F., & Grolinger, K. (2021). Reinforcement learning algorithms: An overview and classification. In *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)* (pp. 1–7). IEEE.
- Alshaikh, F., & Hewahi, N. (2021). Ai and machine learning techniques in the development of intelligent tutoring system: A review. In *2021 International Conference on innovation and Intelligence for informatics, computing, and technologies (3ICT)* (pp. 403–410). IEEE.
- Alshaikh, F., & Hewahi, N. (2024). Adaptive pedagogical model based on classifier system and reinforcement learning. *The Computer Journal (Accepted with Revision)*.
- Amrieh, E., Hamtini, T., & Aljarah, I. (2016). *Student performance prediction dataset*. <https://www.kaggle.com/aljarah/xAPI-Edu>

- Data?select=xAPI-Edu-Data.csv. (Accessed: 07-2022)
- Amrieh, E. A., Hamtini, T., & Aljarah. (2016). Mining educational data to predict student's academic performance using ensemble methods. *International journal of database theory and application*, 9(8), 119–136.
- Amrieh, E. A., Hamtini, T., & Aljarah, I. (2015). Preprocessing and analyzing educational data set using x-api for improving student's performance. In *2015 IEEE Jordan conference on applied electrical engineering and computing technologies (AEECT)* (pp. 1–5). IEEE.
- Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017, August). A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*. doi: 10.1109/MSP.2017.2743240
- Asthana, P., & Hazela, B. (2020). Applications of machine learning in improving learning environment. *Multimedia Big Data Computing for IoT Applications: Concepts, Paradigms and Solutions*, 417–433.
- Ausin, M. S. (2019). Leveraging deep reinforcement learning for pedagogical policy induction in an intelligent tutoring system. In *Proceedings of the 12th International Conference on Educational Data Mining (EDM 2019)*.
- Ausin, M. S. (2021). *A transfer learning framework for human-centric deep reinforcement learning with reward engineering*. North Carolina State University.
- Bach, M., Werner, A., & Palt, M. (2019). The proposal of undersampling method for learning from imbalanced datasets. *Procedia Computer Science*, 159, 125–134.
- Bassen, J., Balaji, B., Schaarschmidt, M., Thille, C., Painter, J., Zimmaro, D., ... Mitchell, J. C. (2020). Reinforcement learning for the adaptive scheduling of educational activities. In *Proceedings of the 2020 CHI conference on human factors in computing systems* (pp. 1–12).

- Beasley, D., Bull, D. R., & Martin, R. R. (1993). An overview of genetic algorithms: Part 1, Fundamentals. *University computing*, 15(2), 56–69.
- Bellafkih, M., Azough, S., Bellafkih, M., & Bouyakhf, E. H. (2010). Adaptive e-learning using genetic algorithms. *IJCSNS International Journal of Computer Science and Network Security*, 10, 237. Retrieved from <https://www.researchgate.net/publication/264347091>
- Bernard, J., Chang, T.-W., Popescu, E., & Graf, S. (2015). Using artificial neural networks to identify learning styles. In *International conference on artificial intelligence in education* (pp. 541–544). Springer.
- Bodenhofer, U. (2003). Genetic algorithms: theory and applications. *Lecture notes, Fuzzy Logic Laboratorium Linz-Hagenberg, Winter, 2004*.
- Booker, L. B., Goldberg, D. E., & Holland, J. H. (1989). Classifier systems and genetic algorithms. *Artificial intelligence*, 40(1-3), 235–282.
- Bouchet, F., Harley, J. M., Trevors, G. J., & Azevedo, R. (2013). Clustering and profiling students according to their interactions with an intelligent tutoring system fostering self-regulated learning.
- Boukthir, K., Haddad, L., Hamdani, T. M., & Alimi, A. M. (2022). A reinforcement learning framework based on genetic algorithm for annotating arabic text in natural scene images. Retrieved from <https://doi.org/10.21203/rs.3.rs-2237131/v1>
- Bueff, A., Papantonis, I., Simkute, A., & Belle, V. (2022). Explainability in machine learning: a pedagogical perspective. *arXiv preprint arXiv:2202.10335*.
- Buseti, F. (2007). Genetic algorithms overview. *Retrieved on December, 1*.
- Cabada, R. Z., Estrada, M. L. B., Hernández, F. G., & Bustillos, R. O. (2014). Intelligent tutoring system with affective learning for mathe-

- matics. In *Mexican international conference on artificial intelligence* (pp. 483–493). Springer.
- Cao, L., & ZhiMin. (2019). An overview of deep reinforcement learning. In *Proceedings of the 2019 4th international conference on automation, control and robotics engineering* (pp. 1–9).
- Carr, J. (2014). An introduction to genetic algorithms. *Senior Project*, 1(40), 7.
- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16, 321–357.
- Chen, Q., Huang, M., Xu, Q., Wang, H., & Wang, J. (2020). Reinforcement learning-based genetic algorithm in optimizing multidimensional data discretization scheme. *Mathematical Problems in Engineering*, 2020. doi: 10.1155/2020/1698323
- Cortez, P., & Silva, A. M. G. (2008a). *Student performance prediction dataset*. <https://archive.ics.uci.edu/dataset/320/student+performance>. UCI Machine Learning Repository. (Accessed: 10-07-2022)
- Cortez, P., & Silva, A. M. G. (2008b). Using data mining to predict secondary school student performance..
- Crockett, K., Latham, A., & Whitton, N. (2017). On predicting learning styles in conversational intelligent tutoring systems using fuzzy decision trees. *International Journal of Human-Computer Studies*, 97, 98–115.
- de Carvalho, S. D., de Melo, F. R., Flôres, E. L., Pires, S. R., & Loja, L. F. B. (2020). Intelligent tutoring system using expert knowledge and kohonen maps with automated training. *Neural Computing & Applications*, 32(17).
- Ding, Z., Huang, Y., Yuan, H., & Dong, H. (2020). Introduction to re-

- inforcement learning. *Deep reinforcement learning: fundamentals, research and applications*, 47–123.
- Dogan, B., & Camurcu, A. Y. (2010). Visual clustering of multidimensional educational data from an intelligent tutoring system. *Computer Applications in Engineering Education*, 18(2), 375–382.
- Drugan, M. M. (2019). Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms. *Swarm and evolutionary computation*, 44, 228–246.
- Durães, D., Toala, R., Gonçalves, F., & Novais, P. (2019). Intelligent tutoring system to improve learning outcomes. *Ai Communications*, 32(3), 161–174.
- Eryılmaz, M., & Adabashi, A. (2020). Development of an intelligent tutoring system using bayesian networks and fuzzy logic for a higher student academic performance. *Applied Sciences*, 10(19), 6638.
- Fahad Mon, B., Wasfi, A., Hayajneh, M., Slim, A., & Abu Ali, N. (2023). Reinforcement learning in education: A literature review. In *Informatics* (Vol. 10, p. 74). MDPI.
- Fan, C., Chen, M., Wang, X., Wang, J., & Huang, B. (2021). A review on data preprocessing techniques toward efficient and reliable knowledge discovery from building operational data. *Frontiers in energy research*, 9, 652801.
- Fenjiro, Y., & Benbrahim, H. (2018). Deep reinforcement learning overview of the state of the art. *Journal of Automation, Mobile Robotics and Intelligent Systems*, 12, 20-39. doi: 10.14313/JAMRIS_3-2018/15
- Fenza, G., Orciuoli, F., & Sampson, D. G. (2017). Building adaptive tutoring model using artificial neural networks and reinforcement learning. In *2017 IEEE 17th international conference on advanced learning technologies (ICALT)* (pp. 460–462). IEEE.
- Gamboa, H., & Fred, A. (2002). Designing intelligent tutoring systems: a

- bayesian approach. *Enterprise Information Systems III*. Edited by J. Filipe, B. Sharp, and P. Miranda. Springer Verlag: New York, 146–152.
- Gazzah, S., Heckel, A., & Amara, N. E. B. (2015). A hybrid sampling method for imbalanced data. In *2015 IEEE 12th International Multi-Conference on Systems, Signals & Devices (SSD15)* (pp. 1–6). IEEE.
- Ghorbani, R., & Ghousi, R. (2020). Comparing different resampling methods in predicting students' performance using machine learning techniques. *IEEE Access*, 8, 67899–67911.
- Gross, S., Mokbel, B., Hammer, B., & Pinkwart, N. (2012). Feedback provision strategies in intelligent tutoring systems based on clustered solution spaces. *DeLFI 2012: Die 10. e-Learning Fachtagung Informatik der Gesellschaft für Informatik eV*.
- Guyon, I., Gunn, S., Nikravesh, M., & Zadeh, L. A. (2008). *Feature extraction: foundations and applications* (Vol. 207). Springer.
- Hamim, T., Benabbou, F., & Sael, N. (2021). Survey of machine learning techniques for student profile modeling. *International Journal of Emerging Technologies in Learning (IJET)*, 16(4), 136–151.
- Hamoud, A. (2016). Selection of best decision tree algorithm for prediction and classification of students' action. *American International Journal of Research in Science, Technology, Engineering & Mathematics*, 16(1), 26–32.
- Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 30).
- Hausknecht, M., & Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*.
- Holland, J. H., & Reitman, J. S. (1978). Cognitive systems based on adaptive algorithms. *Acm Sigart Bulletin*(63), 49–49.

- Hossin, M., & Sulaiman, M. N. (2015). A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2), 1.
- Hssina, B., & Erritali, M. (2019). A personalized pedagogical objectives based on a genetic algorithm in an adaptive learning system. In (Vol. 151, p. 1152-1157). Elsevier. doi: 10.1016/j.procs.2019.04.164
- Iglesias, A., Martinez, P., & Fernández, F. (2003). An experience applying reinforcement learning in a web-based adaptive and intelligent educational system. *Informatics in Education*, 2(2), 223–240.
- Iglesias, A., Martínez, P., Aler, R., & Fernández, F. (2009, August). Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning. *Applied Intelligence*, 31, 89-106. doi: 10.1007/s10489-008-0115-1
- Iglesias, R., Rodríguez, M. A., Regueiro, C. V., Correa, J., & Barro, S. (2006). Combining reinforcement learning and genetic algorithms to learn behaviours in mobile robotics. In *Icinco-ra* (pp. 188–195).
- Imran, M., Latif, S., Mehmood, D., & Shah, M. S. (2019). Student academic performance prediction using supervised learning techniques. *International Journal of Emerging Technologies in Learning*, 14(14).
- Jasti, M. S. (2019). Using reinforcement learning in a simulated intelligent tutoring system. *Northern Illinois University*.
- Jishan, S. T., Rashu, R. I., Haque, N., & Rahman, R. M. (2015). Improving accuracy of students' final grade prediction model using optimal equal width binning and synthetic minority over-sampling technique. *Decision Analytics*, 2, 1–25.
- Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374(2065), 20150202.

- Katz, S., Aronis, J., & Creitz, C. (2000). Modeling pedagogical interactions with machine learning. *Kognitionswissenschaft*, 9(1), 45–49.
- Khan, I., Ahmad, A. R., Jabeur, N., & Mahdi, M. N. (2021). An artificial intelligence approach to monitor student performance and devise preventive measures. *Smart Learning Environments*, 8(1), 1–18.
- Kim, J., Kwon, D., Woo, S. Y., Kang, W. M., Lee, S., Oh, S., ... Lee, J. H. (2021, August). On-chip trainable hardware-based deep q-networks approximating a backpropagation algorithm. *Neural Computing and Applications*, 33, 9391-9402. doi: 10.1007/s00521-021-05699-z
- Kochmar, E., Vu, D. D., Belfer, R., Gupta, V., Serban, I. V., & Pineau, J. (2022, June). Automated data-driven generation of personalized pedagogical interventions in intelligent tutoring systems. *International Journal of Artificial Intelligence in Education*, 32, 323-349. doi: 10.1007/s40593-021-00267-x
- Koedinger, K. R., Brunskill, E., Baker, R. S., McLaughlin, E. A., & Stamper, J. (2013). New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine*, 34(3), 27–41.
- Korkmaz, C., & Correia, A.-P. (2019). A review of research on machine learning in educational technology. *Educational Media International*, 56(3), 250–267.
- Koroveshi, J., & Ktona, A. (2021a). A comparison of exploration strategies used in reinforcement learning for building an intelligent tutoring system. In *RTA-CSIT* (pp. 11–17).
- Koroveshi, J., & Ktona, A. (2021b). Training an intelligent tutoring system using reinforcement learning. *International Journal of Computer Science and Information Security (IJCSIS)*, 19(3).
- Kotsiantis, S., Kanellopoulos, D., Pintelas, P., et al. (2006). Handling imbalanced datasets: A review. *GESTS international transactions on computer science and engineering*, 30(1), 25–36.

- Kryvenchuk, Y., Petrenko, D., Cichon, D., Malynovskyy, Y., & Helzhynska, T. (2022). Selection of deep reinforcement learning using a genetic algorithm. In *COLINS* (pp. 1129–1138).
- Kubotani, Y., Fukuhara, Y., & Morishima, S. (2021). Rltutor: Reinforcement learning based adaptive tutoring system by modeling virtual student with fewer interactions. *arXiv preprint arXiv:2108.00268*.
- Kumar, M., Mehta, G., Nayar, N., & Sharma, A. (2021). Emt: Ensemble meta-based tree model for predicting student performance in academics. In *IOP Conference Series: Materials Science and Engineering* (Vol. 1022, p. 012062). IOP Publishing.
- Kumar, S. (2020). Balancing a cartpole system with reinforcement learning—a tutorial. *arXiv preprint arXiv:2006.04938*.
- Kurita, T. (2021). Principal component analysis (pca). In *Computer vision: a reference guide* (pp. 1013–1016). Springer.
- Li, S., & Liu, T. (2021). Performance prediction for higher education students using deep learning. *Complexity*, 2021, 1–10.
- Li, X., Xu, H., Zhang, J., & Chang, H.-h. (2023). Deep reinforcement learning for adaptive learning systems. *Journal of Educational and Behavioral Statistics*, 48(2), 220–243.
- Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.
- Lin, H. T., Lee, P. M., & Hsiao, T. C. (2015). Online pedagogical tutorial tactics optimization using genetic-based reinforcement learning. *Scientific World Journal*, 2015, 352895. doi: 10.1155/2015/352895
- Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8, 293–321.
- Lincke, A., Jansen, M., Milrad, M., & Berge, E. (2021). The performance of some machine learning approaches and a rich context model in student answer prediction. *Research and Practice in Technology En-*

- hanced Learning*, 16(1), 1–16.
- Liu, R., & Koedinger, K. R. (2017). Closing the loop: Automated data-driven cognitive model discoveries lead to improved instruction and learning gains. *Journal of Educational Data Mining*, 9(1), 25–41.
- Ly, A., Dazeley, R., Vamplew, P., Cruz, F., & Aryal, S. (2024). Elastic step dqn: A novel multi-step algorithm to alleviate overestimation in deep q-networks. *Neurocomputing*, 576. doi: 10.1016/j.neucom.2023.127170
- Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource management with deep reinforcement learning. In *Proceedings of the 15th acm workshop on hot topics in networks* (pp. 50–56).
- Melo, F. S., Mascarenhas, S., & Paiva, A. (2018, September). A tutorial on machine learning for interactive pedagogical systems. *International Journal of Serious Games*, 5, 79-112. doi: 10.17083/ijsg.v5i3.256
- Memarian, B., & Doleck, T. (2024, June). A scoping review of reinforcement learning in education. *Computers and Education Open*, 6, 100175. doi: 10.1016/j.caeo.2024.100175
- Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.
- Mnih, V. (2013). Playing atari with deep reinforcement learning. *arXiv preprint*.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Hassabis, D. (2015, February). Human-level control through deep reinforcement learning. *Nature*, 518, 529-533. doi: 10.1038/nature14236
- Montavon, G., Samek, W., & Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital signal processing*, 73, 1–15.
- Mousavi, S. S., Schukat, M., & Howley, E. (2018). Deep reinforcement learning: an overview. In *Proceedings of SAI Intelligent Systems Con-*

- ference (IntelliSys) 2016: Volume 2* (pp. 426–440). Springer.
- Ng, H., bin Mohd Azha, A. A., Yap, T. T. V., & Goh, V. T. (2022). A machine learning approach to predictive modelling of student performance. *F1000Research*, *10*, 1144.
- Nian, R., Liu, J., & Huang, B. (2020). A review on reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, *139*, 106886.
- Nkambou, R., Bourdeau, J., & Mizoguchi, R. (2010). Introduction: what are intelligent tutoring systems, and why this book? In *Advances in intelligent tutoring systems* (pp. 1–12). Springer.
- Oliveira Neto, J. D. d., & Nascimento, E. V. (2012). Intelligent tutoring system for distance education. *JISTEM-Journal of Information Systems and Technology Management*, *9*, 109–122.
- Paduraru, C., Paduraru, M., & Iordache, S. (2022). Using deep reinforcement learning to build intelligent tutoring systems. In *Icsoft* (pp. 288–298).
- Pedro, F., Subosa, M., Rivas, A., & Valverde, P. (2019). Artificial intelligence in education: Challenges and opportunities for sustainable development.
- Phan, B. C., Lai, Y. C., & Lin, C. E. (2020, June). A deep reinforcement learning-based mppt control for pv systems under partial shading condition. *Sensors (Switzerland)*, *20*. doi: 10.3390/s20113039
- Pillay, N. (2020). The impact of genetic programming in education. *Genetic Programming and Evolvable Machines*, *21*, 87–97.
- Poudyal, S., Mohammadi-Aragh, M. J., & Ball, J. E. (2022). Prediction of student academic performance using a hybrid 2d cnn model. *Electronics*, *11*(7), 1005.
- Pujianto, U., Prasetyo, W. A., & Taufani, A. R. (2020). Students Academic Performance Prediction with k-Nearest Neighbor and C4. 5

- on SMOTE-balanced data. In *2020 3rd international seminar on re-search of information technology and intelligent systems (ISRITI)* (pp. 348–353). IEEE.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1, 81–106.
- Ramachandran, A., & Scassellati, B. (2014). Adapting difficulty levels in personalized robot-child tutoring interactions. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*.
- Ramaphosa, K. I. M., Zuva, T., & Kwuimi, R. (2018). Educational data mining to improve learner performance in gauteng primary schools. In *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)* (pp. 1–6). IEEE.
- Rasim, R., Rosmansyah, Y., Langi, A. Z., & Munir, M. (2021, August). Immersive intelligent tutoring system for remedial learning using virtual learning environment. *Indonesian Journal of Science and Technology*, 6, 507-522. doi: 10.17509/ijost.v6i3.38954
- Reddy, S., Levine, S., & Dragan, A. (2017). Accelerating human learning with deep reinforcement learning. In *Nips workshop: teaching machines, robots, and humans*.
- Roy, S., Mondal, S., Ekbal, A., & Desarkar, M. S. (2016). Crdt: correlation ratio based decision tree model for healthcare data mining. In *2016 IEEE 16th International Conference on Bioinformatics and Bioengineering (BIBE)* (pp. 36–43). IEEE.
- Ruan, S., Nie, A., Steenbergen, W., He, J., Zhang, J. Q., Guo, M., . . . Brunskill, E. (2024, May). Reinforcement learning tutor better supported lower performers in a math task. *Machine Learning*, 113, 3023-3048. doi: 10.1007/s10994-023-06423-9
- Saleem, F., Ullah, Z., Fakieh, B., & Kateb, F. (2021). Intelligent decision support system for predicting student’s e-learning performance using

- ensemble machine learning. *Mathematics*, 9(17), 2078.
- Santhi, R., Priya, B., & Nandhini, J. (2013). Review of intelligent tutoring systems using bayesian approach. *arXiv preprint arXiv:1302.7081*.
- Šarić-Grgić, I., Grubišić, A., Šerić, L., & Robinson, T. J. (2020). Student clustering based on learning behavior data in the intelligent tutoring system. *International Journal of Distance Education Technologies (IJDET)*, 18(2), 73–89.
- Sehgal, A., La, H., Louis, S., & Nguyen, H. (2019). Deep reinforcement learning using genetic algorithm for parameter optimization. In *2019 Third IEEE International Conference on Robotic Computing (IRC)* (pp. 596–601). IEEE.
- Siddique, A., Jan, A., Majeed, F., Qahmash, A. I., Quadri, N. N., & Wahab, M. O. A. (2021). Predicting academic performance using an efficient model based on fusion of classifiers. *Applied Sciences*, 11(24), 11845.
- Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Singh, S., & Gupta, P. (2014). Comparative study id3, cart and c4. 5 decision tree algorithm: a survey. *International Journal of Advanced Information Science and Technology (IJAIST)*, 27(27), 97–103.
- Singh, S., & Vikram, S. (2020). A novel approach to student profiling in intelligent tutoring systems. *Wesleyan Journal of Research*, 13(27).
- Smith, A., Min, W., Mott, B. W., & Lester, J. C. (2015). Diagrammatic student models: Modeling student drawing performance with deep learning. In *User modeling, adaptation and personalization: 23rd international conference, umap 2015, dublin, ireland, june 29–july 3, 2015. proceedings 23* (pp. 216–227). Springer.
- Song, Y., Wei, L., Yang, Q., Wu, J., Xing, L., & Chen, Y. (2023). RLga: A reinforcement learning-based genetic algorithm for electromag-

- netic detection satellite scheduling problem. *Swarm and Evolutionary Computation*, 77, 101236.
- Subramanian, J., & Mostow, J. (2021). Deep reinforcement learning to simulate, train, and evaluate instructional sequencing policies. In *Spotlight presentation at Reinforcement Learning for Education workshop at Educational Data Mining 2021 conference*.
- Sun, H., & Hu, X. (2017). Attribute selection for decision tree learning with class constraint. *Chemometrics and Intelligent Laboratory Systems*, 163, 16–23.
- Thede, S. M. (2004). An introduction to genetic algorithms. *Journal of Computing Sciences in Colleges*, 20(1), 115–123.
- Urbanowicz, R. J., & Browne, W. N. (2017). *Introduction to learning classifier systems*. Springer.
- Venkatesh, R., Naganathan, E., & Maheswari, N. U. (2010). Intelligent tutoring system using hybrid expert system with speech model in neural networks. *International Journal of Computer Theory and Engineering*, 2(1), 12.
- Vie, A., Kleinnijenhuis, A. M., & Farmer, D. J. (2020). Qualities, challenges and future of genetic algorithms: a literature review. *arXiv preprint arXiv:2011.05277*.
- Vijayalakshmi, V., & Venkatachalapathy, K. (2019). Comparison of predicting student's performance using machine learning algorithms. *International Journal of Intelligent Systems and Applications*, 11(12), 34.
- Wang, F. (2018). Reinforcement learning in a pomdp based intelligent tutoring system for optimizing teaching strategies. *International Journal of Information and Education Technology*, 8, 553-558.
- Wang, X., Wang, S., Liang, X., Zhao, D., Huang, J., Xu, X., . . . Miao, Q. (2022). Deep reinforcement learning: A survey. *IEEE Transactions*

- on Neural Networks and Learning Systems*, 35(4), 5064–5078.
- Wang, X., Zhao, Y., Li, C., & Ren, P. (2023). Probsap: A comprehensive and high-performance system for student academic performance prediction. *Pattern Recognition*, 137, 109309.
- Wang, Y. H., Li, T. H. S., & Lin, C. J. (2013). Backward q-learning: The combination of sarsa algorithm and q-learning. *Engineering Applications of Artificial Intelligence*, 26, 2184-2193. doi: 10.1016/j.engappai.2013.06.016
- Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning* (pp. 1995–2003). PMLR.
- Wang, Z., & Sobey, A. (2020). A comparative review between genetic algorithm use in composite optimisation and the state-of-the-art in evolutionary computation. *Composite Structures*, 233, 111739.
- Woolf, B. P. (2007). *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann.
- Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9, 611–629.
- Yousafzai, B. K., Khan, S. A., Rahman, T., Khan, I., Ullah, I., Ur Rehman, A., ... Cheikhrouhou, O. (2021). Student-performulator: Student academic performance using hybrid deep neural network. *Sustainability*, 13(17), 9775.
- Zakrzewska, D. (2008). Using clustering technique for students' grouping in intelligent e-learning systems. In *Symposium of the austrian hci and usability engineering group* (pp. 403–410). Springer.
- Zeng, F., Wang, C., & Ge, S. S. (2020). Tutor-guided interior navigation with deep reinforcement learning. *IEEE Transactions on Cognitive*

and Developmental Systems, 13(4), 934–944.

- Zhang, J., & Cai, J. (2023, April). A dual-population genetic algorithm with q-learning for multi-objective distributed hybrid flow shop scheduling problem. *Symmetry*, 15. doi: 10.3390/sym15040836
- Zhao, X., Wang, L., Zhang, Y., Han, X., Deveci, M., & Parmar, M. (2024). A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*, 57(4), 99.
- Zhou, G., Azizsoltani, H., Ausin, M. S., Barnes, T., & Chi, M. (2019). Hierarchical reinforcement learning for pedagogical policy induction. In *Artificial intelligence in education: 20th international conference, aied 2019, chicago, il, usa, june 25-29, 2019, proceedings, part i 20* (pp. 544–556). Springer.
- Zhou, H., Zhang, J., Zhou, Y., Guo, X., & Ma, Y. (2021). A feature selection algorithm of decision tree based on feature weight. *Expert Systems with Applications*, 164, 113842.

ملخص الدراسة

الأهداف الأساسية للبحث في مجالات التعليم والذكاء الاصطناعي هي تطوير وكسب مستقل يمكنه أداء المهام أو اتخاذ القرارات دون تدخل بشري. يعد نظام للتدريس الذاتي (ITS) أمراً بالغ الأهمية في التعليم لأنه يوفر مساعدة تعليمية شخصية فردية للمتعلمين أثناء تعليمهم كيفية حل المشكلات من خلال التوجيه والملاحظات السريعة. إنه يوفر بيئة تعليمية ذكية للطلاب دون تدخل من المعلم. الهدف الأساسي للنظام للتدريس الذاتي هو دعم ومساعدة المتعلمين على الحصول على المعرفة الفكرية الخاصة بالمجال بطريقة عملية ومنتجة من خلال استخدام تقنيات الحوسبة المختلفة. تقدم هذه الأطروحة ثلاث تجارب بديهيكل معمارية مختلفة للتحقيق في فعالية خوارزميات الذكاء الاصطناعي والتعلم الآلي في مجال نظام التدريس الذاتي.

تبنى التجربة الأولى ثلاثة نماذج بناء على بنى الشبكات العصبية التلافيفية (CNN) لاستكشاف فعالية خوارزميات التعلم العميق (DL) للتنبؤ بالأداء الأكاديمي للطلاب. يتم استخدام مجموعتين من البيانات العامة (PTdataset) و (xAPIdataset) علوة على ذلك، تم استخدام تقنيتين لاختبار العوزات في هذه التجربة: تحليل المكونات الرئيسية (PCA) وأشجار القرار (DTs). وفقاً للنتائج التجريبية، وصل نجاح نموذج CNN المقترح في التنبؤ بأداء الطلاب في المراحل المبكرة إلى دقة 94.36٪ باستخدام مجموعة بيانات PT و 84.83٪ باستخدام مجموعة بيانات xAPI. عند مقارنة النهج المقترح بالدراسات السابقة، تفوق النهج المقترح على جميع الدراسات السابقة عند استخدام مجموعة بيانات xAPI وجزء من مجموعة بيانات PT. بالنسبة لمجموعة بيانات PT الكاملة، كان أداء النموذج المقترح جيداً للغاية.

في التجربة الثانية، استخدمنا نظام تصنيف التعلم (LCS) للخوارزميات الجينية (GAs) والتعلم التعزيزي (RL) لتطوير مبادئ تعليمية لأنظمة تكنولوجيا المعلومات. قسماً ثلاثة نماذج. في النموذج الأول، استخدمنا نظام تصنيف التعلم (LCS)، استخدمنا التعلم التعزيزي وفي النموذج الثالث، قمنا بدمج نظام تصنيف التعلم للخوارزميات الوراثية مع التعلم التعزيزي لإنشاء نموذج هجين يسمى النموذج التبروي التكيفي القائم على نظام تصنيف التعلم والتعلم التعزيزي (PMCR). الهدف الأساسي من هذه التجربة هو إنشاء وتنفيذ نموذج تعليمي تكيفي يستخدم مزايا الخوارزميات الوراثية والتعلم التعزيزي لإنشاء بيئة تعليمية مخصصة وقابلة للتكيف مع المتعلمين. يبدأ النموذج المقترح بتوليد مجموعة عشوائية من الشروط والإجراءات، حيث يمثل الشرط التقديرات للمشاكل المقدمة للمعلم والإجراء هو مياصة للتدريس (التكيف) التي يجب مراعاتها لمثل هذا النوع من الطلاب. في النهاية، يجب أن يتأق النظام بمجموعة جديدة يمكنها في الغالب إنتاج مياصة للتدريس الصحيحة بناء على استجابات الطلاب للمشاكل المقدمة. تشير نتائج الدراسة إلى أن النموذج الثالث (PMCR)، الذي يجمع بين LCS و RL، هو أفضل نموذج مقارنة بالنماذج المقترحة الأخرى. بلغت دقة 82% PMCR ووصلت درجة F1 إلى 0.9 عندما تم اختبار النموذج من خلال حالات مشكلة مختلفة.

تعتمد التجربة الثالثة على التعلم التعزيزي العميق (DRL). DRL هو نمج تقنيات التعلم التعزيزي والتعلم العميق. في ITS، يوفر DRL للطلاب تجارب تعليمية فردية بناء على أنماط التعلم وتفضيلاتهم الخاصة. في هذا العمل، نستخدم بنى التعلم التعزيزي العميق لإنشاء نموذج تربوي لـ ITS. نقدم أربعة نماذج مختلفة. يعتمد النموذج الأول على بنى Deep Q-Network (DQN)، والنموذج الثاني الثلاثي (Double DQN) و (Dueling DQN) عبارة عن امتدادات لخوارزمية DQN وقد طورها باحثون سابقون لمعالجة بعض القيود المفروضة على نهج DQN. النموذج المقترح الأخير هو نموذج هجين يجمع بين DQN القياسي ونظام تصنيف التعلم القائم على الخوارزميات الجينية. تشير نتائج الدراسة إلى أن كلا من Double DQN و Dueling DQN أظهرتا كفاءة بيانات كاملة بدقة 100% ودرجة F1 1.0 بدقة 1.0 عند أعلى مستوى تدريبي مكون من 1000 حلقة و 10 خطوات بينما كان أداء DQN عادلاً ولكن تفوقت عليه التقنيات المتقدمة. على نفس مستوى التكوين. يحقق DQN القياسي دقة 0.975 ودرجة F1 0.987 ودقة 98% من ناحية أخرى، كان أداء النهج الهجين، الذي يجمع بين GA و DQN، سبب مقارنة بالنماذج الفردية الثلاثة الأخرى .

جامعة البحرين

كلية تقنية المعلومات

قسم علوم الحاسوب



تطوير نظام التعلم الذكي باستخدام نهج التعلم الآلي الهجين

أطروحة مقدمة كجزء من متطلبات الحصول على
درجة الدكتوراة في علوم الحوسبة والمعلومات

إعداد

فاطمة عبداللطيف الشيخ

٢٠٠٠٥٩٩٩

إشراف

ا.د. نبيل الحويحي

(أستاذ)

جامعة البحرين

مملكة البحرين

ديسمبر ٢٠٢٤