

**Deep Learning Principles and Practices &**

**Reinforcement Learning**

**Assignment**

**Implementation of Deep Learning and Reinforcement Learning in *Crop Recommendation System***

*Submitted by*

1. Harsh Santwani 211310142015
2. Shubham Prajapati 211310142018

**Branch :** CSE(AI-ML)

**Batch :** A2

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Content** | **Page** |
| 1 | Introduction | 3 |
| 2 | Aim and Scope | 3 |
| 3 | Dataset and Experimental Setup | 4 |
| 4 | Methodology | 6 |
| 5 | Results | 13 |

**1. Introduction**

**Overview**

Agriculture is crucial for economies like India, where over 60% of the population depends on farming. However, farmers often struggle to choose the right crops due to various factors such as soil quality and weather conditions, leading to reduced yields and financial losses. Leveraging technology, especially machine learning, can help farmers make informed decisions about crop selection. This project focuses on developing a Crop Recommendation System using Deep Learning (DL) and Reinforcement Learning (RL) methodologies.

**Deep Learning and Reinforcement Learning in Crop Recommendation System**

In this project, we will implement an Artificial Neural Network (ANN) for the DL component. ANNs can effectively analyze multiple input parameters—such as Nitrogen, Phosphorous, Potassium, pH, humidity, temperature, and rainfall—to predict the most suitable crop. By training the ANN on historical crop data, it can learn complex relationships and provide accurate recommendations.

For the RL aspect, we will use **Q-learning**. Q-learning is a reinforcement learning algorithm that helps an agent learn optimal actions based on environmental interactions. This approach will enable the system to recommend the best crop based on expected rewards, such as yield potential and profitability.

**2. Aim and Scope**

**2.1 Research Paper Analysis and Objectives**

The selected research paper focuses on developing a **Crop Recommendation System** that utilizes machine learning algorithms to enhance agricultural productivity. The study emphasizes the significance of various environmental factors, such as soil health and climatic conditions, in determining the best crop to cultivate.

**Objectives of the Research Paper:**

* To assess how soil nutrients, temperature, humidity, and pH levels influence crop yield.
* To implement machine learning models, including Decision Trees and Support Vector Machines, for predicting suitable crops based on environmental data.
* To provide farmers with actionable insights that can lead to improved crop yields and economic benefits.

This analysis highlights the necessity for reliable crop recommendations in optimizing agricultural practices, especially in regions facing environmental variability.

**2.2 Project Aim: Using Deep Learning (DL) and Reinforcement Learning (RL)**

The goal of this project is to enhance the Crop Recommendation System by incorporating both **Deep Learning (DL)** and **Reinforcement Learning (RL)** methodologies.

**Key Components:**

* **Deep Learning (DL)**:
  + We will implement an **Artificial Neural Network (ANN)** to process input data such as nitrogen (N), phosphorous (P), potassium (K), temperature, humidity, pH, and rainfall. The ANN will learn complex patterns in the data to accurately predict the most suitable crops for given conditions.
  + The DL approach allows for deeper insights and more accurate predictions due to its capability to model intricate relationships within the dataset.
* **Reinforcement Learning (RL)**:
  + We will utilize Q-learning to create an agent that dynamically interacts with the environment. The Q-learning algorithm will learn optimal crop selection strategies by exploring different combinations of inputs and assessing the resulting outcomes (rewards) over time.
  + This RL approach enables the system to adapt to changing conditions, maximizing crop yield and profitability through intelligent decision-making.

**3. Dataset and Experimental Setup**

**3.1 Dataset Description**

The dataset used in this project is the Crop Recommendation Dataset, which is available on Kaggle. It contains various features that influence crop growth and yield, making it suitable for developing a predictive crop recommendation system.

Key Features of the Dataset:

* N: Nitrogen content (in kg/ha)
* P: Phosphorous content (in kg/ha)
* K: Potassium content (in kg/ha)
* Temperature: Average temperature (in °C)
* Humidity: Average humidity (in %)
* pH: Soil pH level
* Rainfall: Average rainfall (in mm)
* Label: Crop type (e.g., rice, wheat, maize, etc.)

| * **Crop Type** | **Count** |
| --- | --- |

|  |  |
| --- | --- |
| Rice | 1165 |

|  |  |
| --- | --- |
| Maize | 557 |

|  |  |
| --- | --- |
| Wheat | 422 |

|  |  |
| --- | --- |
| Chickpea | 148 |

|  |  |
| --- | --- |
| Kidney Beans | 145 |

|  |  |
| --- | --- |
| Pigeon Peas | 138 |

|  |  |
| --- | --- |
| Lentils | 144 |

|  |  |
| --- | --- |
| Barley | 103 |

|  |  |
| --- | --- |
| Oilseed | 85 |

|  |  |
| --- | --- |
| Cotton | 103 |

|  |  |
| --- | --- |
| Groundnut | 80 |
|  |  |

**3.2 IDE Setup**

To implement the Crop Recommendation System using Deep Learning (DL) and Reinforcement Learning (RL), follow these steps to set up your Integrated Development Environment (IDE):

1. Choose an IDE:
   * Recommended: Jupyter Notebook, PyCharm, or Visual Studio Code.
2. Install Required Libraries:
   * Ensure you have Python installed (preferably Python 3.6 or higher).
   * Use pip to install the following libraries:

pip install numpy pandas scikit-learn tensorflow keras matplotlib seaborn gym

1. Download the Dataset:
   * Download the Crop Recommendation Dataset from Kaggle using the provided link and save it in your project directory.
2. Create a New Project:
   * Start a new project in your chosen IDE and create a new notebook or Python file for your implementation.
3. Load Libraries:
   * Import the necessary libraries at the beginning of your script or notebook

**4. Methodology**

**4.1 Deep Learning: Artificial Neural Networks (ANN)**

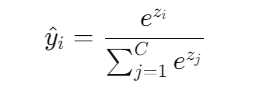
Overview:  
The ANN model will be trained to predict the optimal crop based on soil and climatic features. The input features will be processed through multiple layers to produce the output label (crop type).

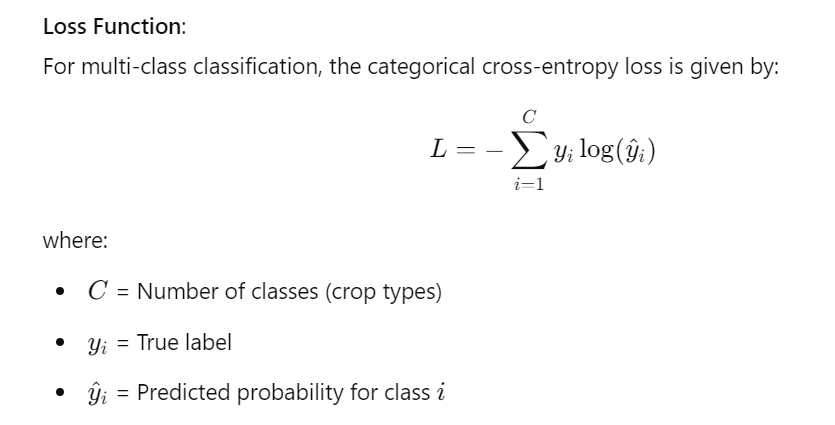
Model Architecture:

1. Input Layer: Accepts the input features (N, P, K, Temperature, Humidity, pH, Rainfall).
2. Hidden Layers: One or more layers with activation functions.
   * Activation Function: ReLU (Rectified Linear Unit) is commonly used in hidden layers for its simplicity and efficiency.



1. Output Layer: Uses the Softmax activation function to produce probabilities for each

****

****

**4.1.1 Pseudo code:**

*BEGIN*

*// Import necessary libraries*

*IMPORT pandas as pd*

*IMPORT train\_test\_split from sklearn.model\_selection*

*IMPORT LabelEncoder, StandardScaler from sklearn.preprocessing*

*IMPORT Sequential from tensorflow.keras.models*

*IMPORT Dense from tensorflow.keras.layers*

*IMPORT numpy as np*

*// Load the dataset*

*SET file\_path = 'Crop\_recommendation.csv'*

*LOAD crop\_data from CSV file at file\_path*

*// Encode the target variable*

*INITIALIZE label\_encoder as LabelEncoder*

*ENCODE crop\_data['label'] using label\_encoder*

*// Separate features and target variable*

*SET X to crop\_data without 'label'*

*SET y to crop\_data['label']*

*// Scale the feature data*

*INITIALIZE scaler as StandardScaler*

*SCALE X using scaler to get X\_scaled*

*// Split the dataset into training and testing sets*

*SPLIT (X\_scaled, y) INTO X\_train, X\_test, y\_train, y\_test WITH test\_size 0.2 AND random\_state 42*

*// Build the neural network model*

*INITIALIZE model as Sequential*

*ADD Dense layer with 128 neurons, 'relu' activation to model*

*ADD Dense layer with 64 neurons, 'relu' activation to model*

*ADD Dense layer with 32 neurons, 'relu' activation to model*

*ADD Dense layer with number of unique labels neurons, 'softmax' activation to model*

*// Compile the model*

*COMPILE model with optimizer 'adam', loss 'sparse\_categorical\_crossentropy', and metric 'accuracy'*

*// Train the model*

*SET history to model.fit(X\_train, y\_train, epochs 30, validation\_data (X\_test, y\_test), batch\_size 32)*

*// Evaluate the model on test data*

*SET test\_loss, test\_accuracy to model.evaluate(X\_test, y\_test)*

*PRINT "Test Accuracy: " + test\_accuracy*

*// Prepare for making predictions*

*SET test\_data to array with specific test data values*

*// Scale the test data*

*SET test\_data\_scaled to scaler.transform(test\_data)*

*// Make predictions using the model*

*SET predictions to model.predict(test\_data\_scaled)*

*// Decode the predicted label*

*SET predicted\_crop to label\_encoder.inverse\_transform(index of max value in predictions)*

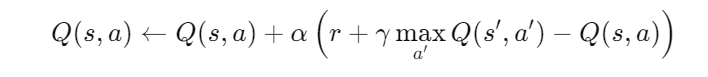
*PRINT "Predicted Crop: " + predicted\_crop[0]*

*END*

**4.2 Reinforcement Learning: Deep Q-Network (DQN)**

**Overview:** The Q-learning algorithm will be implemented to optimize crop recommendations based on rewards received from the environment. The agent learns to select actions (crop types) based on the state (environment features).

**Q-Value Formula:** The Q-value is updated using the Bellman equation:



**Experience Replay:** To improve stability, Q-learning uses experience replay where past experiences are stored and sampled randomly for training:

* Experiences are stored in a memory buffer.
* During training, a random sample of past experiences is used to update the Q-values.
* This helps break the correlation between consecutive experiences, leading to more stable training.

**4.2.1 Pseudo code:**

*BEGIN*

*// Imports*

*IMPORT necessary libraries: numpy, pandas, random, torch, nn, optim, deque, tqdm, train\_test\_split, accuracy\_score, precision\_score, confusion\_matrix, StandardScaler*

*// Load dataset*

*LOAD crop data from 'Crop\_recommendation.csv'*

*// Normalize features*

*SET feature\_columns to ['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']*

*INITIALIZE scaler as StandardScaler*

*SCALE crop\_data[feature\_columns] using scaler*

*// Define action mapping*

*CREATE crop\_action\_mapping from unique labels in crop\_data*

*CREATE action\_crop\_mapping from crop\_action\_mapping*

*// Stratified Train-Test Split*

*SPLIT crop\_data into train\_data and test\_data with test\_size 0.2 and stratify by 'label'*

*PRINT number of training samples*

*PRINT number of testing samples*

*// Set seed for reproducibility*

*SET random seeds for torch, numpy, and random*

*// Hyperparameters*

*DEFINE state\_size as 7 // Number of features*

*DEFINE action\_size as number of unique crops*

*DEFINE other hyperparameters: batch\_size, gamma, epsilon, epsilon\_min, epsilon\_decay, learning\_rate, target\_update, memory\_size, episodes*

*// Neural network for Q-learning*

*DEFINE QLearningModel class with \_\_init\_\_ and forward methods:*

*- \_\_init\_\_: initialize three fully connected layers*

*- forward: define the forward pass to return Q-values for each action*

*// Initialize the Q-learning agent*

*DEFINE QLearningAgent class with methods:*

*- \_\_init\_\_: initialize state size, action size, memory, model, optimizer, loss function, and epsilon*

*- remember: store experiences in memory*

*- act: choose action based on exploration/exploitation*

*- replay: sample experiences and train the model*

*// Convert state (row from crop\_data) to feature vector*

*DEFINE get\_state function to return feature vector from row*

*// Initialize the Q-learning agent*

*INITIALIZE agent as QLearningAgent with state\_size and action\_size*

*// Training loop*

*FOR each episode in range(episodes):*

*FOR each sample in train\_data:*

*SET state using get\_state*

*// Choose action (predict crop)*

*SET action using agent.act(state)*

*// Get actual crop for the state*

*SET actual\_crop using crop\_action\_mapping*

*// Define reward*

*SET reward to +1 for correct prediction, -1 for incorrect*

*// In a static environment, next state is the same as current state*

*SET next\_state to state*

*SET done to True*

*// Store experience*

*CALL agent.remember(state, action, reward, next\_state, done)*

*// Train using experience replay*

*CALL agent.replay(batch\_size)*

*// Update target network every 'target\_update' episodes (if applicable)*

*IF (episode + 1) % target\_update == 0 THEN*

*CALL agent.update\_target\_model() // This line may be omitted if not applicable*

*// Optional progress logging*

*IF (episode + 1) % 50 == 0 THEN*

*PRINT progress*

*// Evaluate the agent's performance*

*DEFINE evaluate function with parameters agent, test\_data, crop\_action\_mapping:*

*SET test\_states from test\_data*

*SET test\_labels using crop\_action\_mapping*

*INITIALIZE test\_predictions list*

*SET model to evaluation mode*

*WITH no gradient tracking:*

*FOR each state in test\_states:*

*SET state\_tensor*

*GET q\_values using agent's model*

*APPEND predicted action to test\_predictions*

*// Calculate evaluation metrics*

*SET accuracy using accuracy\_score*

*SET precision using precision\_score*

*SET confusion matrix using confusion\_matrix*

*PRINT evaluation metrics*

*// Run evaluation*

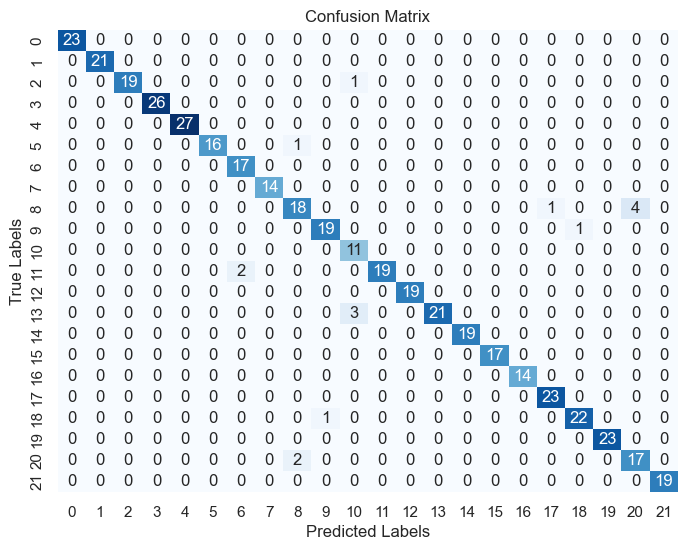
*CALL evaluate with agent, test\_data, crop\_action\_mapping*

*END*

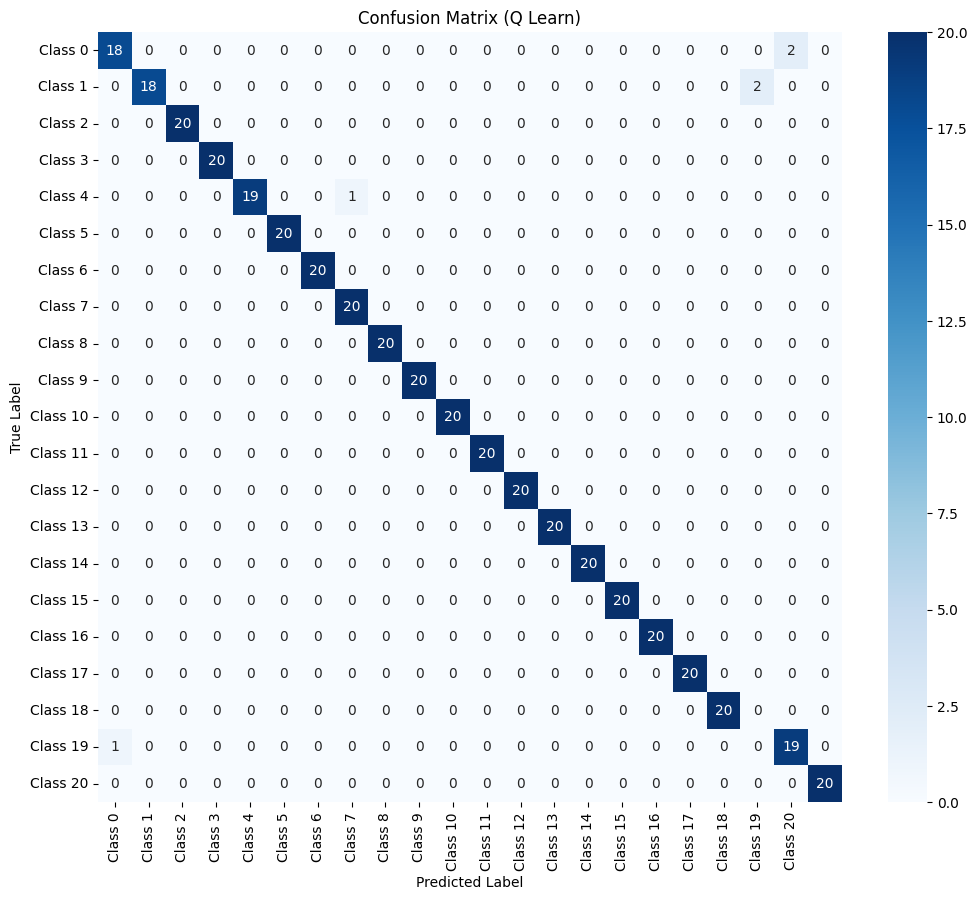
**5. Results**

|  |  |  |
| --- | --- | --- |
| **Metric** | **DL (ANN)** | **RL (DQN)** |
| Precision | 0.961800 | 0.55 |
| Recall | 0.966128 | 0.68 |
| F1 Score | 0.962337 | 0.60 |
| Accuracy | 0.963636 | 0.71 |

**Confusion Matrix (DL-CNN):**

****

**Confusion Matrix (RL-Q Learning):**

****