The Journal of Supercomputing
https://doi.org/10.1007/s11227-023-05568-7

Active learning-based hyperspectral image classification: a reinforcement learning approach

Usha Patel[1] · **Vibha Patel**[2]

Accepted: 3 August 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

In the last few years, deep neural networks have been successful in classifying hyperspectral images (HSIs). However, training deep neural networks needs a large number of labeled datasets. In HSIs, acquiring a large amount of labeled data is costly and time-consuming. Active learning (AL) is a technique for selecting a small subset of data for annotation so that the classifier can learn from the data with high accuracy. Most of the AL methods are designed based on some statistical approach. The efficacy of the statistical methods is limited, and their performance varies depending on the scenario. So, a reinforced pool-based deep active learning (RPDAL) approach is proposed to overcome limitations of statistical selection approaches. The reinforcement learning (RL)-based agent is designed and trained to select informative samples for annotation. The learned RL-based agent can transfer and choose samples for annotation on any other HSI dataset after being trained on one. Indian Pines (IP), Pavia University (PV), and Salinas Valley (SL) are three publicly available datasets used in the experiment. The proposed approach achieves 92.78%, 97.85%, and 97.94% accuracy using 400 labeled samples with IP, PV, and SL datasets, respectively. The labeled samples selected using the proposed approach achieve better classification performance than other AL techniques.

Keywords Hyperspectral image classification · Active learning · Deep Q Network · Reinforcement learning

✉ Usha Patel
ushapatel@nirmauni.ac.in

Vibha Patel
vibhadp@gmail.com

[1] CSE Department, Institute of Technology, Nirma University, Ahmedabad, India

[2] IT Department, Vishwakarma Government Engineering College, Gujarat Technological University, Ahmedabad, India

Published online: 14 August 2023

1 Introduction

Hyperspectral imagery plays a prominent role in various applications in the field of remote sensing. In remote sensing, land-cover classification through HSIs is widely used in applications like object segmentation, land cover mapping, security, defense, etc. HSIs carry a wide range of spectral reflectance of an object. Contents of images captured by making use of advanced high-resolution hyperspectral sensors can be classified precisely. Such high resolution and spate of HSIs demand innovative methods for analyzing the images. Hyperspectral image classification aims to assign class labels to each pixel of an image. Spectral and spatial information obtained from neighboring pixels helps improve classification results. Supervised machine learning (ML) algorithms are frequently used for spectral and spatial land cover classification. Deep learning (DL), a subset of ML, has complex structures that automatically pull low-level information from images. Many DL methods like autoencoders [1], recurrent neural networks [2], and convolutional neural networks (CNN) [3, 4] are widely applied to classify HSIs. DL algorithms perform efficiently when a large amount of labeled training data is available. The quality and quantity of the labeled samples determine the performance of a DL algorithm. However, labeling individual pixels in high-resolution HSIs is costly and time-consuming. To handle the issue of limited numbers of labeled samples, researchers used semi-supervised learning [5], transfer learning [6] and a few-shot learning [7] approaches for HSIs classification. Semi-supervised learning can extract information from the labeled and unlabeled samples. AL [8] is one of the categories of semi-supervised learning.

AL is a cost-effective iterative process for selecting the most informative samples from a pool of unlabeled samples. The chosen samples need to be annotated by a human. The objective of the AL task is to determine the minimum number of samples that must be labeled to achieve a better classification performance. Deep active learning (DeepAL) [9] is an emerging field that retains the learning capabilities of DL and lowers the cost of annotation through AL. DeepAL is a hybrid of DL and AL that improves image classification, text classification, and object detection. This article employs the DeepAL approach, which integrates a DL model with AL techniques for HSIs classification. Once the samples have been chosen by AL, these samples which have been annotated by humans are included in the training dataset. The selection of samples for annotation of HSIs can be viewed as an optimization problem, aiming to identify the minimum collection of samples that offer the most detailed information.

Several heuristic-based algorithms are used in the literature to solve this optimization problem. Such AL acquisition functions select samples for annotation based on statistics like uncertainty [10], classifier confidence of class prediction [11], diversity [12], etc. These AL functions are predetermined and known before the classification. Whether the choice of a specific AL function is appropriate or not is unknown until the budget is exhausted. A majority of these algorithms are

developed with the help of domain experts through trial and error. As a result, we are curious to discover if this heuristic design strategy for sample selection tasks can be automated using RL. RL systems, in principle, are trained by their own experience, allowing them to work in situations where human knowledge is limited and, thus, helpful in discovering novel selection strategies without the need for hand-engineered reasoning. Deep RL, which incorporates DL with RL, has recently demonstrated breakthrough results in various disciplines [13–17].

In this paper, a novel technique, namely, RPDAL, is suggested for clarification of HSIs with limited labeled samples. This article presents an innovative framework utilizing deep RL through Deep Q Network (DQN) [18] to address the challenge of selecting a minimum set of informative samples for annotation. The proposed work is divided into two phases. In phase I, the DQN agent is trained for informative sample selection. The agent undergoes training over multiple episodes to acquire the necessary knowledge. The designed framework defines DQN architecture components, including state, action, and reward, with the AL objective to minimize the number of annotations required to achieve higher classification accuracy for HSIs. The trained agent is applied to other HSI classification tasks in Phase II. The designed DQN agent can learn and transfer strategies across different datasets. The representation of states and actions is designed to accommodate large action spaces and multiclass classifications. In this phase, the trained DQN agent selects samples from an unlabeled dataset for annotation. The classifier is trained using the selected annotated samples, and the classification result is compared with other AL methods. Overall, this proposed technique introduces a novel framework combining DeepAL with DQN for selecting informative samples for HSIs classification.

The major contributions of the article are as follows:

- Designs a deep neural network with an AL technique framework for HSI classification.
- Proposes a novel Reinforcement-based AL for HSI classification to maximize classification accuracy with the least number of labeled samples.
- Introduces a policy transfer algorithm that allows the learned policy for selecting samples for annotation to be transferred to other HSIs, lowering the cost of annotating.
- Uses three publicly available HSI datasets to evaluate the learned DQN agent. The outcome is compared to the results of various statistical AL approaches.

The rest of the paper is organized as follows. Section 2 outlines AL and RL and how RL uses AL in other domains. Section 3 describes the proposed method for formulating AL with RL, AL agent, and classifier for HSIs. Section 4 describes implementation details like preprocessing, the dataset used, and other AL methods for comparison, experiment parameters, and results. Section 5 contains the conclusion and future potential of the proposed method.

2 Background

2.1 Active learning (AL)

AL is one of the research domains to identify the most informative samples which are to be annotated by human oracles. Pool-based and stream-based are the two types of AL methods. Pool-based AL takes samples for annotation from the total dataset, whereas stream-based AL selects samples for annotation in each data stream independently. A DQN agent selects samples for annotation from a pool of unlabeled datasets in the experiment, which is pool-based AL.

Traditional AL approaches select one sample at a time for annotation, and the model gets trained using the updated training set. In the DL model, the traditional AL technique is insufficient and leads to overfitting. DeepAL lowers the labeling cost by combining AL and DL and provides robust learning capabilities. Ren et al. [9] outline significant challenges and their solutions. Batch-based query approaches are employed in the DeepAL model to overcome the overfitting problem of the DL model. This method selects a batch of samples from the unlabeled data pool for annotation. Then, to train the DL model, this batch of annotated samples is added to the training set. This implementation uses pool-based batch mode AL for training the DL model.

Figure 1 illustrates the overall iterative procedure. The original dataset is separated into D_{train} and D_{pool}, as shown in the diagram. Initially, a few samples of each class are in D_{train}. D_{train} is used to train the DL model, and sample selection is from D_{pool} using AL criteria. Annotated samples are added to the training set and removed from the unlabeled data. This iterative process continues until the target criteria are met.

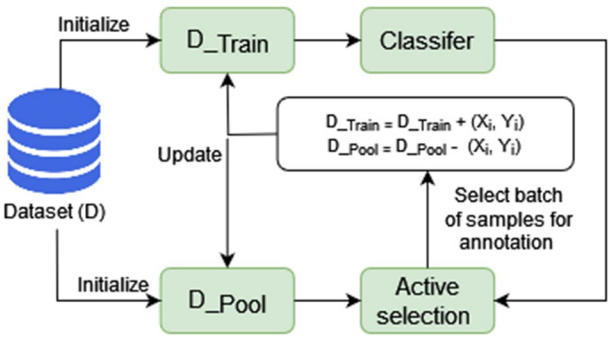


Fig. 1 Active learning framework

2.2 Reinforcement learning

RL is learning how to behave to achieve a desired goal. Over a series of discrete-time steps, the RL agent and its environment interact. Significant components of RL are State, Action, and Reward. The actions are the choices made by the agent, the environment for making the choices defines the states, and the rewards are the basis for evaluating the choices. A policy is a stochastic rule that the agent uses to choose actions based on states. The agent's goal is to maximize the total rewards in the long run. One-step transition probabilities and expected rewards for all states and their actions are included in such a model. A complete model of the environment is challenging to build in some domains. Even if the agent has a complete and accurate environment model, computational complexity and memory availability make it difficult to process every possible state and action. To overcome this problem, approximations of value functions, rules, and model building are necessary.

Dynamic programming, Monte Carlo methods, and Temporal difference learning are three fundamental methods for handling the RL problem. Temporal-difference methods are widely accepted as they do not require a model and are fully incremental. An RL agent solution may not be optimum; however, it can be approximate. Due to the advancement in DNNs, an artificial agent such as DQN can learn a policy directly using end-to-end RL. Atari 2600 [18] could successfully let the RL agent receive input from pixels and obtain scores comparable to human-level intelligence with the same network and hyperparameters. This human-level control using DQN will provide the scope to come up with many real-time application solutions with a DQN agent.

2.3 Active learning with reinforcement learning

The RL-based AL approach has been introduced to address the challenge of acquiring annotated data in various domains. This section aims to review articles that delve into this approach, providing insights into its application and effectiveness.

Fang et al. [19] proposed AL using an RL agent for the sample selection in the Natural Language processing domain. The authors also transferred the learned agent to another dataset. Sun et al. [20] and SSmann et al. [21] defined RL-based AL for image classification. Vu et al. [22] formulated AL as Markov decision process (MDP) for text classification and evaluated cross-domain and cross-lingual text classification. Liu et al. [23] selected n nearest neighbor as the most informative sample; then, the RL agent selected the most crucial among them. Rudovic et al. [24] proposed a multi-model training network for human-computer interaction. A large amount of labeled data was required for multi-model network training. The author used RL-based AL to select informative samples for the same. Rudovic et al. [25] used RL to identify video streams for manual labeling to apply in personalized Estimation of Engagement from videos. Pixel-based image segmentation promises accuracy, but labeling individual pixels is tedious and costly. Casanova et al. [26] used the pool-based RL technique for image segmentation. Taguchi et al. [27] used a

pre-trained RL agent to select a fixed budget label for the annotation. Konyushkova et al. [28] proposed a generalized AL technique using the DQN agent. In this work, the authors give state and action as input to DQN. The authors used –1 as a reward for every request for annotation. Hsu and Lin [29] defined learning-based AL using an m-arm bandit approach. Different statistics-based AL techniques perform differently on various datasets with varying iterations of AL. Each iteration selects an AL technique from an m-arm bandit in the proposed approach. The selection criteria are then updated using the generated reward.

One-shot learning is one of the attractive research fields in the domain of computer vision. "One-shot AL" aims to identify the object with either one or only a few labeled samples of an object. Woodward and Finn [30] proposed active one-shot learning using an RL agent. The author used the long short-term memory network to predict a class as a classifier or request a label. Huang et al. [31] used active one-shot learning for aircraft type recognition.

Mou et al. [32] used the DQN agent for band selection of HSIs. Correlation between the selected band and entropy was used as the reward of the DQN agent. Feng et al. [33] also used RL for band selection with semi-supervised learning.

It is essential to define state, action, and reward in the RL framework. Table 1 presents a literature review containing the use of state, action, and reward to solve the defined problem. The table shows that RL-based AL is limited to natural language applications, with some image and video processing usage. Only a few publications in HSI processing use RL to choose influential bands for HSIs. This article is the first to propose RL-based AL for HSI classification.

3 Methodology

This section presents the proposed RL-based AL method for HSIs classification.

Problem formulation D is a Hyperspectral image dataset, where x_i is the spectral reflectance of ith pixel and y_i is its corresponding label. $D_{train(t)}$ is the labeled training dataset and $D_{pool(t)}$ is the unlabeled pool dataset at time t. The neighboring pixel in any image is highly correlated. To take advantage of the spatial correlation, each pixel is considered with its neighboring pixels to extract the most important spatial and spectral information. $(x_m, y_m)_{m=1}^{n} \in D_{train(t)}$ where x_m is a 3D patch to represent the input of the HSIs. y_m is the associated label of the central pixel of the patch. $(x_m)_{m=1}^{n} \in D_{pool(t)}$ where x_m an unlabeled 3D patch. The labeled test dataset D_{test} is used to assess the classifier's performance. Classifier(f) is a customized CNN model having 2D and 3D convolution layers. Equation 1 defines the objective function for the given problem.

$$\max_{acc(f)} \text{ subject to } \min_{n}(x_m, y_m)_{m=1}^{n} \tag{1}$$

Table 1 Literature on RL-based AL, HSIs and RL

	Objective	State	Action	Reward
[19]	AL using RL to select data for annotation for Natural Language processing task, transfer the learning of policy net on another dataset	Vector representation of xi input and the output of the trained model of xi	1—for annotation, 0—not	Improvement in classification accuracy
[20]	RL approach for selection of image for labeling for image classification (CIFAR)	Classifier extracted features	ai = 0 for annotation, ai = 1 for not	Entropy loss function
[21]	Selection of labeled sample acquisition function as a learning predictor, use for the image classification (MNIST and CIFAR)	The posterior probability of k selected samples based on entropy	Data points sampled for annotation	Improvement in data fit and diversity of selected sample
[22]	Formulated AL as MDP for text classification, evaluated on cross-domain and cross-lingual text classification	Labeled and unlabeled datasets paired	Selection of a query data point	Improvement in the learning
[23]	RL policy agents select samples for annotation from n nearest neighbors to minimize human efforts	Observe n nearest neighbors as the unlabeled pool	Request kth instance from the unlabeled pool for the annotation	Hard triplet loss is used to measure the uncertainty of data
[24]	RL-based AL approach to train multi-model for human–computer interaction	Classifier output is used as the state of DQN	Action is [1 0] for label request or [0 1] otherwise	For label request $r = -0.5$, Incorrect classification = −1 and correct classification = 1
[25]	RL is used to decide which video stream is selected for manual labeling for personalized Estimation of Engagement from Videos	Input video deep feature extracted by the classifier	1—for annotation, 0—not	The reward for request, correct and incorrect prediction
[26]	Pool-based RL AL for image segmentation, Select a small portion of the image for labeling	Ensembled feature representation of region Ds is used as the state of the policy agent	Query network selects action as a sample for the labeling from the k randomly selected samples	The reward is calculated from the labeled samples D_R

Table 1 (continued)

	Objective	State	Action	Reward
[27]	Train DQN for AL on one dataset and use for the selection of sample on another dataset	Performance parameter of the classifier	Select a sample from the pool for the annotation	Accuracy improvement after adding an actively selected sample
[28]	Generalized AL as RL for binary class classification where state and action were given as input to DQN	Randomly selected samples class prediction probability of single class vector as state	Action vector of xi sample is class prediction probability, distance from D_{train}, and D_{pool}	−1 for every label request
[29]	M—arm bandit approach for selection of AL strategies	For each arm weight vector maintain	Select AL technique	Weighted accuracy
[30]	Combine RL with One-shot learning with the LSTM network	Instance xt was given as input to the policy net	Predict class or request label	The reward for a correct prediction, incorrect prediction, or request
[31]	Amsterdam Library of Object Images (ALOI) dataset to show the general performance for target recognition	Instance xt was given as input to the policy net	Predict class or request label	The reward for a correct prediction, incorrect prediction, or request
[32]	DQN is used for the band selection of Hyperspectral image	30 bands selected out of 200 bands for the Indian pine dataset select action as one band	30 bands selected out of 200 bands for the Indian pine dataset select action as one band	Entropy and the correlation values of the selected band as a reward
[33]	RL is used for the band selection of HSIs. Semisupervised CNN is used to evaluate band selection	The subset of available band	Continue for the selection or terminate	Penalty for same band selection, where +ve reward for a new band

Where (x_m, y_m) are labeled training samples. The proposed work aims to maximize the classifier accuracy $acc(f)$ with minimum labeled training data. The proposed algorithm trains an RL agent to select samples for the annotation to achieve the goal. It is a generalized RL agent which determines samples for annotation in the HSIs domain.

3.1 AL as MDP

MDP formally represents the environment for RL agents. So MDP has been defined in the proposed AL algorithm for HSI classification. MDP defines how the environment and agent interact with each other. The MDP environment specifies the current state, based on which the agent will select the appropriate action. Executing this action in an environment will lead to a state change, and a reward is assigned. Based on the reward received, the agent will change its policy to maximize the reward. This approach is repeated for the agent to learn and achieve the desired goal.

In Fig. 2 (left), the agent and environment are depicted as entities that interact with each other. The agent takes actions based on the current state of the environment, and in response, the environment transitions to a new state and provides feedback or observations to the agent. This interaction follows the principles of an MDP, where the agent's actions affect the environment's state transitions and subsequent observations. Figure 2 (right) represents the formulation of the AL problem as an MDP. In this context, the AL problem is framed as an MDP, allowing for the application of RL techniques. The agent's actions in the AL problem involve selecting informative samples for annotation, and the environment's states and transitions capture the evolving knowledge and uncertainty of the classifier. By modeling the AL problem as an MDP, the agent can learn a policy that maximizes the acquisition of valuable information while minimizing the annotation effort or cost.

Researchers have proposed various RL algorithms like DQN, Double DQN, Actor critics algorithm (A2C, A3C), etc. This experiment use DQN, a simple and fundamental RL algorithm, in its implementation. DQN is a DNN that learns a policy to select an action using end-to-end RL. In standard DQN, the Q-function takes a state representation as input and outputs values corresponding to discrete actions.

In contrast to traditional DQN scenarios where repeatedly selecting the same action can maximize rewards, such an approach is unsuitable for AL. Choosing the

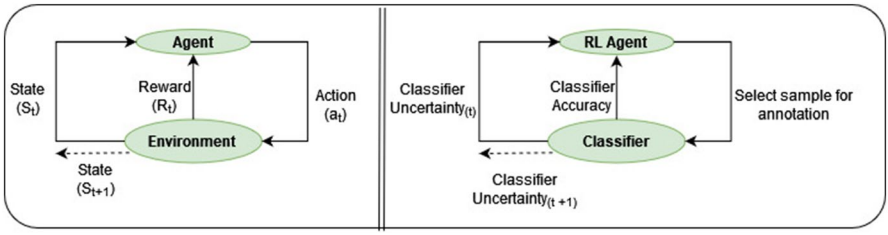


Fig. 2 (left) The interaction between the agent and the environment represented as MDP, and (right) the formulation of the AL problem as an MDP

same sample for annotation in AL does not yield optimal results. To address this issue, Konyushkova et al. [28] proposed a modified architecture for DQN. In their approach, both, the state and possible actions are provided as inputs to the DQN network. The modified DQN architecture facilitates informed decision-making for sample annotation by incorporating state information and selecting the action with the highest Q-value. The DQN architecture proposed by the authors is limited to binary classification [28]. The state and action vector representation is not suitable for multi-class classification. Our proposed solution, which includes carefully designed state, action, and reward formation, is well-suited for addressing the multi-class classification problem. DQN has the following defined state, action, and reward:

- *State* The DQN state represents the classifier's current status. Uncertainty is a crucial parameter to represent the model during training. So, the DQN state is defined to specify the classifier's uncertainty. The best measure of classifier uncertainty is the calculated entropy of class prediction probability for specific input. Classifier output entropy is considered as the state of the DQN agent. Usually, the size of the unlabeled dataset D_{pool} is enormous. So randomly, D_{select} samples are selected from D_{pool} to calculate the entropy. The calculated entropy of D_{select} is considered the state of the DQN agent.

$$E(x_k) = -\Sigma_{i=1..c} P(\hat{y}_i \mid x_k) log P(\hat{y}_i \mid x_k) \tag{2}$$

  Calculating entropy is as given in Eq. 2 for $x_k \in D_{select}$. $E(x_k)$ is the entropy of the selected sample x_k. $\hat{y}_i$ is i^{th} class prediction probability of x_k. The calculated entropy of D_{select} forms a state vector for DQN.
- *Action* Each sample $x_k \in D_{select}$ selected from the unlabeled dataset is specified as an action vector with the following element: Entropy of x_k, Mean distance of x_k and D_{train}, and Mean distance of x_k and D_{pool}. Here, distance is computed with the standard cosine formula. Equation 3 specifies the action vector of sample x_k. Along with the state vector, this action vector is fed into the DQN network. The DQN agent computes the Q value for every action. The agent will choose an action, specifically a sample from the set of available samples called D_{select}, based on the maximum Q-value.

$$A(x_k) = [E(x_k), d(x_k, D_{train}), d(x_k, D_{pool})] \tag{3}$$

- *Reward* Rewards serve as indicators of the quality or effectiveness of the selected action. It measures the chosen action in the context of the problem. By iteratively adjusting its policy based on received rewards, the agent can learn and improve its decision-making abilities over time. Improvement in the overall accuracy (OA) of classifier (f) is considered a reward for the DQN network.

3.2 Proposed algorithm

This section gives the detailed algorithm of a proposed RPDAL. The algorithm is defined in two subsections. The first subsection outlines the training process of the

DQN agent for informative sample selection. The second subsection describes the classifier's training with the actively selected samples by the trained DQN agent.

3.2.1 Learning DQN agent

The goal is to train the DQN agent to choose informative samples from the HSIs Dataset. The learning of the DQN agent is an episodic task. Two procedures, classifier training for target class prediction and DQN learning for active data selection, are iteratively executed in each episode. The DQN network gets trained using one dataset for multiple episodes.

Figure 3 illustrates the process of the DQN agent learning for a single episode. In every episode, initially the classifier uses only a few labeled samples named D_{train}. Here, the initial training dataset D_{train} must contain the samples of each class. To select a sample from the unlabeled pool for annotation, the D_{select} samples are identified randomly to ensure a random and unbiased selection from the pool.

State and action vectors are defined from the D_{select} samples. The DQN agent gets the state and action vectors as input to choose a sample for the annotation. A batch of samples is then selected for annotation. After adding annotated samples to the training dataset, retraining of the classifier is done with an updated D_{train} sample. Retraining the classifier will change the state and action vectors to the next state and the next action. $State(s_i)$, $action(a_i)$, $reward(r_i)$, $nextstate(s_{i+1})$, and $nextaction(a_{i+1})$ are stored in the reply buffer to train the DQN agent.

$$minibatch(s_i, a_i, r_i, s_{i+1}, a_{i+1}, done) \tag{4}$$

Minibatch samples, as given in Eq. 4, are fetched randomly from the reply buffer to break the correlation. The process is repeated until the episode is completed, which occurs when the annotation budget has been exhausted. At the end of each episode,

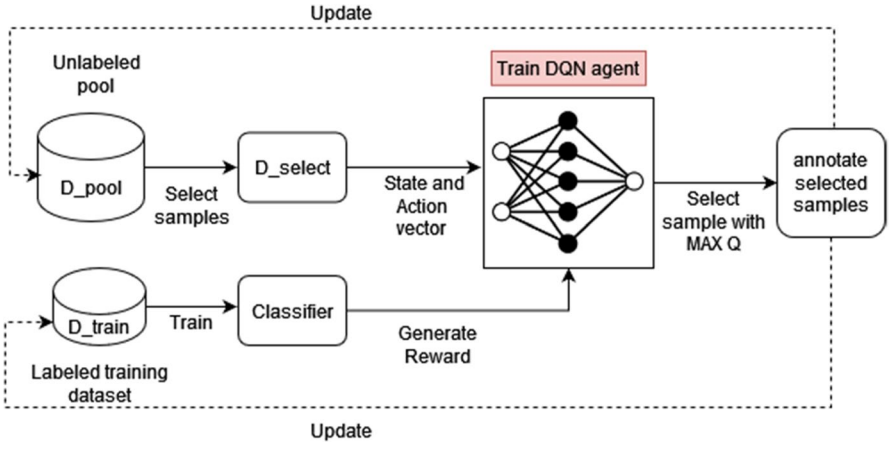


Fig. 3 RPDAL architectural diagram: training DQN agent

done = 1 and $y_i = r_i$; otherwise, y_i is calculated using the current policy given by Eq. 5.

$$y_i = \begin{cases} r_i \\ r_i + \gamma max_{a'} \hat{Q}(s_{i+1}, a_{i+1}, a'; \pi) \end{cases} \tag{5}$$

The DQN agent is trained [18] as given in Eq. 6 using a minibatch.

$$L(\pi) = (y_i - Q(s_i, a_i; \pi))^2 \tag{6}$$

The next episode starts with the classifier, training dataset, and pool dataset re-initialization. The DQN agent will gradually learn which samples to select for the annotation, given that the higher reward significantly improves classification accuracy. Algorithm 1 outlines the detailed steps of the proposed framework.

Algorithm 1 Learn DQN agent(π)

Require: Initial D_{pool}, D_{test}, k (Batch size)
  for all Episode: 1, 2 ..... n **do**
    Select initial D_{train} from D_{pool}
    Train classifier(f) with D_{train}
    repeat
      for all Sample = 1, 2 ..... k **do**
        Randomly choose D_{select} Samples from D_{pool}
        Define (State vector; f) and (Action vector; f)
        Compute Q value with current policy (π)
        $D_{req} = \max_Q D_{select}$
        Acquire label of D_{req}
        $D_{pool} = D_{pool}$ / D_{req}
        $D_{train} = D_{train} \cup D_{req}$
      end for
      if Budget Exhausted **then**:
        done = 1
      else
        done = 0
      end if
      Retrain classifier(f) with updated D_{train}
      Reward = Improve Accuracy (D_{test}; f)
      for all Sample = 1, 2 ..... k **do**
        Define (Next_State vector; f) and (Next_Action vector; f)
        Store $(s_i, a_i, r_i, s_{i+1}, a_{i+1}, done)$ in replay buffer
      end for
      Train DQN(π) agent from eq. (5),(6)
      $y_i = r_i$ or $r_i + \gamma max_a(s_{i+1}, a_{i+1})$
    until Budget Exhausted
  end for

The set of unlabeled dataset D_{pool}, test labeled dataset D_{test} and the batch size (k) is given as input to algorithm 1. For every episode, initial D_{train} samples are selected from D_{pool}. The classifier f is trained with the D_{train} dataset. To prevent overfitting in the DL model, the proposed framework incorporates a batch-mode AL approach. Instead of selecting a single informative sample at a time, a batch of samples k is chosen and included in the training dataset during each AL iteration. By adopting this batch-based strategy, the framework effectively addresses the issue of overfitting and improves the overall performance. For every sample in the batch, define D_{select} as randomly selected samples from D_{pool}. State and action vectors derived using D_{select} are given as input to the DQN. D_{req} samples having the highest Q value are selected. These batches of samples are selected and, after annotation, added in D_{train} and removed from D_{pool}. The classifier now gets trained with the updated D_{train} and the reward gets calculated as an improvement in the test accuracy. With the updated classifier, the state vector and action vector get changed. State, action, reward, next state, and next action are stored in the replay buffer. DQN is trained with the randomly selected minibatch.

The algorithm aims to train the DQN agent for informative sample selection. This is achieved through multiple episodes of experimentation using an exhaustive labeled dataset. The learning of the agent is then evaluated. The trained agent selects samples, and based on these selections, the classifier is trained using the same dataset used for training the agent and other HSI datasets.

3.2.2 Classifier training with DQN agent

The trained DQN agent selects informative samples from the dataset. These selected samples are then used to train the classifier. Such a learned agent can be applied to other environments, which are not seen during the training. The process of transferring learned policies to new environments is known as policy transfer [34]. Here, a state, action, and reward selection is independent of the dataset. The learned agent is used to choose samples for annotation for the same dataset or another dataset with policy transfer. Algorithm 2 provides a concise description of the steps involved in selecting samples for annotation using the trained agent.

Algorithm 2 AL with DQN agent (RPDAL)

Require: Initial D_{pool}, Policy agent π
  Initialize D_{train} and train classifier(f) with D_{train}
  repeat
    for all Sample = 1, 2 ..... k **do**
      Randomly choose D_{select} Samples from D_{pool}
      Define (State vector; f) and (Action vector; f)
      $D_{req} = \max_Q V$
      Acquire label of D_{req}
      $D_{pool} = D_{pool} \ / \ D_{req}$
      $D_{train} = D_{train} \cup D_{req}$
    end for
    Train/Retrain Classifier f with D_{train}
    Reward = Accuracy (D_{train}; f)
    for all Sample = 1, 2 ..... k **do**
      Define (Next_State vector; f) and (Next_Action vector; f)
      Store (State, Action, Reward, Next_state, Next_Action, Done)
    end for
    Update policy π
  until Budget Exhausted

The set of unlabeled dataset D_{pool} and the trained agent are given as input to the algorithm. A few samples are selected from every class to initialize the D_{train} dataset. In every AL iteration, a batch of samples is selected and after annotation, included in the training dataset. The agent is fine tuned in every iteration. In this approach, the reward computation is modified, utilizing the improvement in the training accuracy instead of relying on the test accuracy. This approach streamlines the active learning process and reduces the dependency on additional test labeled datasets.

3.3 Classifier and DQN agent architecture

In this section, the architectures of both the classifier and the DQN agent are outlined. The description provides an overview of the design and structure of these components within the proposed framework.

3.3.1 Classifier

A DNN like CNN can extract the low level of features compared to Machine learning. Here, a customized CNN model, namely Hybrid Spectral Net [35], is used for the classification of HSIs. This model can extract spectral and spatial features using a 3D-convolutional layer followed by 2D convolutional layers. The 2D convolutional operation cannot extract some essential features from spectral dimensions, whereas the 3D convolutional layers are computationally intensive and sometimes perform poorly in HSI classification. So, a hybrid classification model is proposed by the author. HybridSN has three 3D convolutional layers followed by two 2D

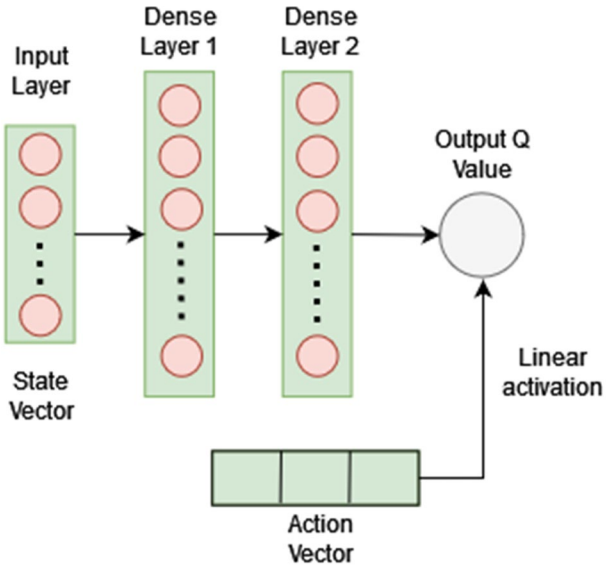


Fig. 4 DQN agent architecture

convolutional layers. The output of convolutional layers is connected to fully connected layers, followed by the output layer.

3.3.2 DQN agent

DNN is used to implement the DQN policy agent. Figure 4 shows the detailed architecture of the network.

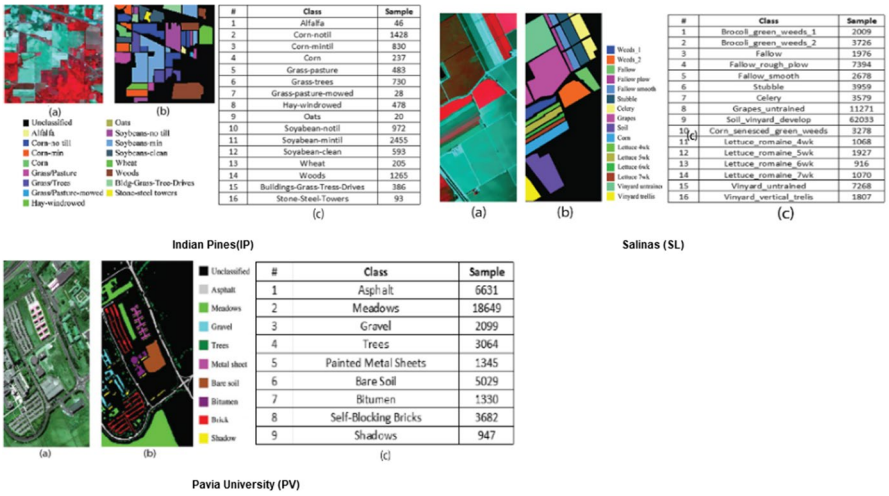
As shown in the figure, DNN consists of two fully connected layers. Both dense layers have *128* units and a *'sigmoid'* activation function. The output of the dense layer is concatenated with the action vector, and with *'linear'* activation, it will generate the output Q value.

4 Experiment and analysis

4.1 Dataset

Following are the publicly available HSI dataset details with ground truth mapping for the experiment.

Indian Pines This dataset is from the northwestern Indiana agriculture site. It was gathered by Airborne visible/infrared imaging spectrometer (AVIRIS) sensors in June 1992. It contains 220 band information from 400 to 2500 nm. This dataset contains 140×140 pixels, each pixel of 20 m spatial resolution. It includes 16 different classes of different types of crops.



#	Class	Sample
1	Alfalfa	46
2	Corn-notil	1428
3	Corn-mintil	830
4	Corn	237
5	Grass-pasture	483
6	Grass-trees	730
7	Grass-pasture-mowed	28
8	Hay-windrowed	478
9	Oats	20
10	Soyabean-notil	972
11	Soyabean-mintil	2455
12	Soyabean-clean	593
13	Wheat	205
14	Woods	1265
15	Buildings-Grass-Tress-Drives	386
16	Stone-Steel-Towers	93

#	Class	Sample
1	Brocoli_green_weeds_1	2009
2	Brocoli_green_weeds_2	3726
3	Fallow	1976
4	Fallow_rough_plow	7394
5	Fallow_smooth	2678
6	Stubble	3959
7	Celery	3579
8	Grapes_untrained	11271
9	Soil_vinyard_develop	62033
10	Corn_senesced_green_weeds	3278
11	Lettuce_romaine_4wk	1068
12	Lettuce_romaine_5wk	1927
13	Lettuce_romaine_6wk	916
14	Lettuce_romaine_7wk	1070
15	Vinyard_untrained	7268
16	Vinyard_vertical_trells	1807

#	Class	Sample
1	Asphalt	6631
2	Meadows	18649
3	Gravel	2099
4	Trees	3064
5	Painted Metal Sheets	1345
6	Bare Soil	5029
7	Bitumen	1330
8	Self-Blocking Bricks	3682
9	Shadows	947

Fig. 5 False color, ground truth image and class details of HSIs dataset

Pavia University scene It is an image of the Engineering school at Pavia University. Data have been captured by a reflective optics system imaging spectrometer (ROSIS). The data contain 115 band information from 430nm to 860 nm. The number of pixels in the image is 610×340 with 1.3 m spatial resolution. The dataset contains nine different classes.

Salinas valley The image is of Salinas Valley, California, captured by AVIRIS sensors. The images have 224 bands with a 3.7 m spatial resolution. The ground truth of the image contains 16 classes. The water absorption bands present in the IP and SL datasets are removed during preprocessing. Figure 5 shows the false-color, ground truth, and class details of the HSI dataset.

4.2 Implementation details

The RPDAL algorithm is executed in two steps. In the first step, the DQN agent is trained using the labeled dataset exhaustively. This training phase aims to optimize the agent's decision-making capabilities and enhance its ability to select informative samples. In the second step, the trained DQN agent is employed to select samples for annotation. The IP dataset is used to serve the goal of training the DQN agent for AL. The IP dataset is more complex than the PV and SL datasets. The preprocessing step is applied to all the datasets according to the following steps.

Preprocessing HSIs contain the spectral reflectance of hundreds of bands. So, there is scope for redundancy among the data. Principle component analysis (PCA) is frequently used to reduce spectral redundancy. PCA is one of the traditional and efficient methods to extract essential features from the data. The initial few PCA components carry most of the energy of the data. The IP dataset contains 200 spectral bands where the first 30 PCA components of IP hold 98% of energy. So, in our experiment, 30 PCA components are used for the IP dataset, whereas 15 PCA

components are used for the PV and SL datasets. In HSIs, neighbor pixels are highly correlated. HSIs are defined in the form of cubes that can take advantage of spatial correlation for efficiently extracting spectral and spatial features. The cube's size also influences computational requirements and the area considered for spatial correlation. A larger cube size will increase the computation, and smaller sizes cannot efficiently extract spatial details. So here, a moderate cube size of $5 \times 5 \times p$ is used. Where p represents the PCA components.

Parameters and training

This section covers the setting of experimental parameters. A batch size of five and an initial training dataset comprising two samples per class are used for all the experiments. D_{select} is made up of 30 randomly selected samples from D_{pool}. The remaining parameters are set empirically.

(1) Classifier parameters A customized 2D–3D CNN model is used as a classifier. The 'relu' activation function is used for the convolutional layers and dense layers, whereas the output layer uses the 'softmax' activation function. With a learning rate of 0.0001, the 'adam' optimizer is used. In each AL iteration, the CNN model gets trained with a batch size of 32 for five epochs.

(2) DQN agent The DQN agent is a DNN with two fully connected layers. The 'Nadam' optimizer is used with a learning rate of 0.0001. In each AL iteration, the DQN model gets trained with a batch size of 32 for three epochs. The future reward γ (discounted) is 0.99.

The DQN agent learning algorithm is executed for 40 episodes. As mentioned in Algorithm 1, the RPDAL algorithm utilizes 800 labeled samples as the budget for each episode. With this specified budget, a test accuracy of approximately 98% is achieved. Afterward, the trained DQN agent is employed to choose samples, and once labeled, the classifier undergoes training. The steps for classifier training are outlined in algorithm 2. As depicted in algorithm 2, the DQN agent is refined through fine-tuning during the classification training process using AL. The experiment is carried out on IP, PV, and SL datasets.

4.3 Comparison with other AL methods

To evaluate the performance of the proposed approach, we implemented other AL techniques using the same classifier and hyperparameters. The comparisons are made with the following techniques.

Without AL This implementation uses 3D-CNN and 2D-CNN without any AL methods. All the preprocessing steps and hyperparameters are the same.

Random sampling (RS) In this method, N samples are selected randomly, and based on that the classifier model is fine-tuned.

Best versus second-best (BvSB) sampling The BvSB technique uses class prediction probabilities of the classifier to select the most informative sample. In this technique, the difference between the highest and second-highest prediction probability aids in selecting samples with a minimum difference, as shown in Eq. 7. Where $\hat{y_1}$ is the highest class prediction probability and $\hat{y_2}$ is the second

highest class prediction probability. A minor difference indicates the model's uncertainty for the prediction of the sample.

$$x = \arg\min_{x} P_\theta(\hat{y_1} \mid x) - P_\theta(\hat{y_2} \mid x) \tag{7}$$

Entropy sampling (ES) In many ML applications, entropy represents a measure of uncertainty. Higher entropy samples represent more uncertainty in prediction. Entropy sampling selects those samples which have higher entropy. Entropy is calculated as given in Eq. 8.

$$x = \arg\max_{x} -\Sigma_{i=1\ldots c} P_\theta(\hat{y_i} \mid x) log P_\theta(\hat{y_i} \mid x) \tag{8}$$

where y_i is the class prediction probability of x, and c is the number of classes.

4.4 Evaluation parameters

Evaluation parameters used to evaluate the performance of the proposed algorithm with different AL methods for HSIs classification are as follows:

- Overall accuracy (OA): OA is the total number of pixels correctly identified and divided by the number of test pixels given by Eq. 9.

$$OA = \frac{NPC}{NTP} \tag{9}$$

  where *NPC* is the number of pixels correctly identified, and *NTP* is the number of test pixels.
- Average accuracy (AA): AA plays a vital role in the multi-class classification of HSIs. AA is the average of all class accuracies given by Eq. 10.

$$AA = \frac{\sum_{i=1}^{i=n} Acc(C_i)}{n} \tag{10}$$

  where n is the total number of classes, and $Acc(C_i)$ is the accuracy of class C_i
- Kappa Coefficient: Cohen's Kappa is a statistical measure of agreement between the final classification and the ground truth. It is the correct agreement percentage by the expected level of agreement given by Eq. 11 [36]. This statistical measure is a robust measure for multi-class classification.

$$K = \frac{c * s - \Sigma_k^K p_k * t_k}{s^2 - \Sigma_k^K p_k * t_k} \tag{11}$$

  where c is the correctly predicted element, s is the total element, p_k is the class k prediction, and t_k are class k occurrences.

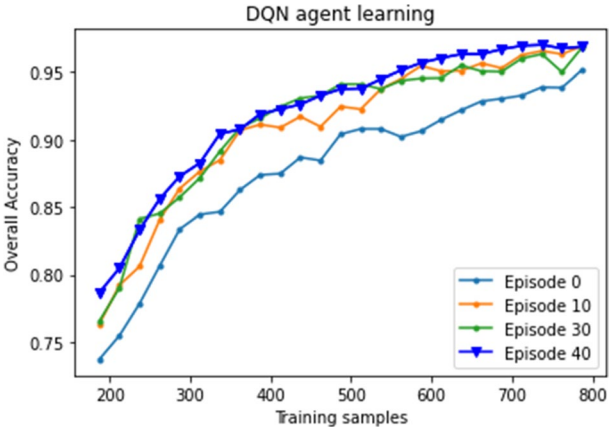


Fig. 6 Result with multiple episode learning of the DQN agent

4.5 Result discussion

This section covers experimental results derived through several executions of the experiments, and the average effect of the experimentation is then presented here.

4.5.1 DQN agent learning

The IP dataset is used to serve the goal of training the DQN agent for AL. The DQN agent learning algorithm is executed for 40 episodes with 800 labeled samples as the budget for an episode. During the course of the episode, the classifier achieved a test accuracy of approximately 98% using 800 labeled samples.

Figure 6 depicts agent learning as improving test accuracy over the episode. There are approximately 10,000 samples in the IP dataset. As shown in the figure,

Table 2 Overall test accuracy of different AL methods on IP dataset

Indian Pines					
Label samples	Without AL	RS	BvSB	ES	RPDAL
100	60.17	51.38	54.86	57.4	**67.48**
150	68.78	69.24	67.77	68.16	**77.43**
200	73.56	78.86	76.1	74.41	**82.41**
250	80.26	78.5	79.9	80.46	**84.72**
300	81.98	82.83	82.86	84.23	**89.37**
350	83.48	86.05	84.42	89.07	**90.73**
400	83.58	85.5	86.28	91.45	**92.78**

The bold text highlights the best outcome attained in the various algorithmic comparisons

the classification result achieved about 95% test accuracy with only 500 labeled samples, which is less than 1% of the total.

4.5.2 Classifier training with DQN agent

This section provides a detailed analysis of the DQN agent's learning performance in the context of AL. The experiment is conducted with the IP dataset, and the classifier's performance with DQN agent selected samples is discussed further.

The overall test accuracy is shown in Table 2 along with the number of labeled samples. Here, the performance of the proposed approach RPDAL is compared with other AL methods, namely RS, BvSB, and ES, with the same number of labeled samples. The performance of CNN classification, along with the different AL techniques, is compared with the classification result without applying AL using the same number of labeled samples. Initially, the classification results without AL outperform the RS, BvSB, and ES methods with very few labeled samples. However, as the size of the training dataset increases, the results of all AL methods improve compared to those without AL. ES produces good results among the RS, BvSB, and ES methods. But ES can only perform well when the classification model is close to convergence. In all scenarios, the proposed RPDAL method consistently outperforms the other methods.

Also, the application of the learned AL policy used with other hyperspectral datasets. An extensive training dataset with multiple-episode learning only makes sense when employed on different datasets. To test the learning agent's performance on a different dataset, the PV and SL datasets are used, as shown in algorithm 2.

For every iteration, the agent selects a batch of samples as specified in algorithm 2. Here, the batch size is five, i.e., the agent selects five samples and updates D_{train} in each iteration. The proposed algorithm is compared with other AL methods that use datasets of the same size as the training dataset. Table 3 shows the overall test accuracy with the number of labeled samples used for the PV and SL datasets.

The preprocessing of the PV and SL datasets is done as explained in the previous section. The model hyperparameters also remain the same for all the AL methods for a fair comparison. The entire unlabeled dataset is given as input to algorithm 2. The algorithm is executed until the budget is exhausted, which is 400 samples in this experiment. In the case of the PV dataset, the RPDAL approach achieves an OA of over 95%, surpassing all other methods which attain an OA of approximately 92% with 200 samples. When using 400 labeled samples, the OA rises to 97.85%, demonstrating a substantial improvement over other AL methods.

Initially, none of the AL methods exhibited significant improvements when using 100 and 150 labeled samples from the SL dataset, as the CNN model tended to overfit. However, noticeable enhancements in classification accuracy were observed after that. Using 400 samples, the classification accuracy reached nearly 98%. These observations help us conclude that the RPDAL technique yielded the most favorable overall results.

Table 3 Overall test accuracy of different AL methods on PV and SL dataset

Label samples	Pavia University					Salinas				
	Without AL	RS	BvSB	ES	RPDAL	Without AL	RS	BvSB	ES	RPDAL
100	82.13	83.46	81.94	83.86	**88.49**	**89.46**	84.2	87.22	77.39	85.37
150	88.75	89.75	90.81	85.6	**92.32**	**91.45**	89.78	91.18	89.52	90.74
200	91.7	92.77	92.04	87.84	**95.32**	91.21	91.34	92.01	90.72	**94.38**
250	92.22	93.7	93.66	88.03	**96.81**	91.24	92.98	93	94.66	**95.59**
300	92.27	93.71	95.39	92.53	**97.76**	93.76	93.34	93.73	95.55	**96.87**
350	93.17	95.01	95.27	94.26	**97.7**	92.84	93.79	93.89	96.74	**97.17**
400	94.42	95.51	95.65	95.89	**97.85**	94.8	93.79	93.79	96.45	**97.94**

The bold text highlights the best outcome attained in the various algorithmic comparisons

Table 4 Class-wise accuracy of Indian Pines Dataset

Method/class	Without AL	RS	BvSB	ES	RPDAL
“Alfalfa"	14.29	85.71	78.57	**100.00**	78.57
“Corn-notill"	79.44	83.41	77.34	89.25	**90.65**
“Corn-mintill"	81.53	79.52	84.34	**89.56**	86.75
“Corn"	59.15	**97.18**	91.55	85.92	94.37
“Grass-pasture"	88.28	85.52	93.79	**97.93**	94.48
“Grass-trees"	98.17	88.58	99.09	93.15	**99.54**
“Grass-pasture-mowed"	100.00	100.00	100.00	100.00	**100.00**
“Hay-windrowed"	100.00	100.00	99.30	97.20	**100.00**
“Oats"	83.33	83.33	83.33	**100.00**	66.67
“Soybean-notill"	77.74	75.68	76.03	80.48	**88.01**
“Soybean-mintill"	88.20	83.72	87.25	**91.72**	91.59
“Soybean-clean"	74.72	87.64	75.84	**93.82**	92.13
“Wheat"	96.72	96.72	96.72	98.36	**100.00**
“Woods"	94.47	92.11	92.11	94.47	**98.68**
“Buildings-Grass-Trees-Drives"	71.55	75.86	79.31	**93.97**	87.93
“Stone-Steel-Towers"	89.29	100.00	96.43	96.43	**96.43**
OA	85.27	85.50	86.28	91.45	92.78
AA	81.05	88.44	88.19	93.89	91.61
Kappa	0.83	0.84	0.84	0.90	0.92

The bold text highlights the best outcome attained in the various algorithmic comparisons

Table 5 Class-wise accuracy of Pavia University Dataset

Method/class	Without AL	RS	BvSB	ES	RPDAL
“Asphalt"	98.19	96.08	95.17	**99.10**	98.94
“Meadows"	97.80	99.43	99.41	94.35	**99.75**
“Gravel"	91.27	92.70	91.59	89.84	**95.56**
“Trees"	89.66	90.64	91.19	**95.97**	90.86
“Painted metal sheets"	100.00	100.00	100.00	100.00	**100.00**
“Bare Soil"	95.96	93.31	91.05	95.56	**99.47**
“Bitumen"	90.48	96.49	97.49	96.49	**99.50**
“Self-Blocking Bricks"	81.54	80.90	86.97	**99.00**	89.23
“Shadows"	97.54	97.18	97.89	99.65	**100.00**
OA	95.18	95.51	95.65	95.89	97.85
AA	93.60	94.08	94.53	96.66	97.03
Kappa	0.94	0.94	0.94	94.59	0.97

The bold text highlights the best outcome attained in the various algorithmic comparisons

The analysis indicates that the DQN agent outperforms other methods in terms of selecting uncertain samples and minimizing the need for extensive labeling. It is also worth noting that ES performs the worst in the PV classification results initially.

Table 6 Class-wise accuracy of Salinas Dataset

Method/class	Without AL	RS	BvSB	ES	RPDAL
“Brocoli_green_weeds_1"	100.00	100.00	100.00	100.00	**100.00**
“Brocoli_green_weeds_1"	**100.00**	99.55	99.91	100.00	99.91
“Fallow"	97.47	90.05	99.49	**100.00**	99.83
“Fallow_rough_plow"	**99.52**	99.28	98.33	99.28	99.04
“Fallow_smooth"	95.64	97.88	97.26	98.13	**98.63**
“Stubble"	99.33	99.92	99.49	100.00	**100.00**
“Celery"	100.00	100.00	100.00	96.93	**100.00**
“Graps_untrained"	92.46	84.06	89.94	88.26	**95.68**
“Soil_vinyard_develop"	100.00	99.14	**100.00**	98.66	99.57
“Corn_senesced_green_weeds"	97.66	91.56	92.17	**100.00**	98.98
“Lettuce_romaine_4wk"	85.63	99.38	87.81	100.00	**100.00**
“Lettuce_romaine_5wk"	99.83	100.00	100.00	**100.00**	99.83
“Lettuce_romaine_6wk"	100.00	86.91	95.27	100.00	**100.00**
“Lettuce_romaine_7wk"	94.70	96.26	98.44	**100.00**	99.38
“Vinyard_untrained"	78.04	89.64	77.99	**95.23**	93.12
“Vinyard_vertical_trellis"	92.99	98.34	97.05	100.00	**100.00**
OA	94.34	93.79	93.79	96.45	97.94
AA	95.83	95.75	95.82	98.53	99.00
Kappa	0.94	0.93	0.93	0.96	0.98

The bold text highlights the best outcome attained in the various algorithmic comparisons

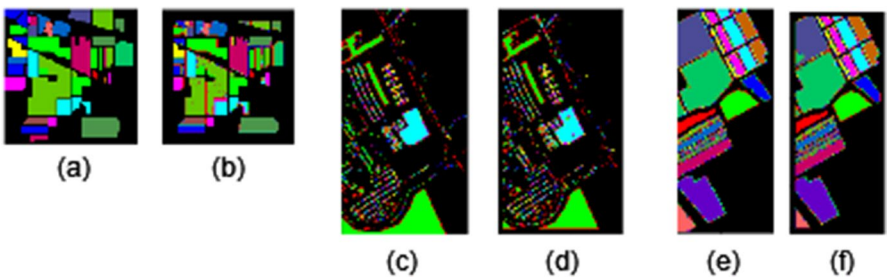


Fig. 7 Ground truth and classification map with RPDAL: **a**, **c**, **e** ground truth of IP, PV, and SL dataset, **b**, **d**, **f** classification result

HSI classification is a multiclass classification problem. So, the average accuracy and Cohen’s Kappa are essential performance metrics for the multiclass classification. Table 4 shows the class-wise accuracy, OA, AA, and Kappa coefficient with 400 labeled samples for the IP dataset.

The IP dataset has 16 classes, including Oats and Alfalfa classes having 20 and 46 labeled samples in total, respectively. Due to the imbalanced dataset, the agent could not correctly classify these class samples, reducing the average accuracy. This class imbalance problem can be addressed in future by designing a more robust method with RL-based AL for HSI classifications.

Table 5 gives the class-wise accuracy, OA, AA, and kappa for the PV dataset with 400 labeled samples. The PV dataset has a total of nine classes. From that, the proposed method achieved the highest class accuracy for six classes. ES achieved good class-wise accuracy for the rest of the classes.

Table 6 shows the class-wise accuracy, OA, AA, and kappa coefficient with 400 labeled samples for the SL dataset. From the 16 classes of the SL dataset, eight classes have the highest classification accuracy with the RPDAL method. The classification accuracy is nearer to the maximum value for the rest of the classes. The average accuracy and kappa value are also good compared to the other AL methods.

The results presented above conclusively prove that the RPDAL approach works best with fewer labeled samples required to train the CNN model for the HSI classification. According to the investigation, the DQN agent is more adaptive in picking informative samples to reduce the cost of labeling.

Figure 7 shows the classification map where the classifier is trained with the RPDAL selected samples. In the image, some of the pixels near the boundary are misclassified. To demonstrate the real-time capabilities of the proposed method, the trained DQN agent was deployed on a computer equipped with an 11th Gen Intel Core i5-1135G7 2.40GHz processor. On average, it takes 10.25 ms to select a single batch of samples during each AL iteration which is considered feasible for practical solution.

5 Conclusion and future work

The research presents deep RL-based AL for HSI classification. A deep reinforcement learning agent, namely DQN, is trained with exhaustive labeled samples from a single dataset with multiple episodes. The trained agent selects the candidate sample for annotation with the current state of the classifier as a function. The trained agent can choose the most important samples for other HSI datasets. The designed DQN agent is generalized for any dataset/classifier. The effectiveness of the proposed approach is evaluated on three publicly available datasets.

The designed DQN agent learns the query selection strategies with exhaustive use of IP dataset samples over multiple episodes. The experiment was conducted to select the set of samples for annotation. First, the IP dataset was used in the experimentation and achieved an OA accuracy of 92.78% with 400 labeled samples. Significantly, the learning of the DQN agent is not restricted to a specific dataset but can be transferred as a policy to other datasets, allowing for verification of its performance. Through policy transfer, the PV dataset achieved an OA of 97.85% with 400 labeled samples, while the SL dataset achieved an OA of 97.94%. Our solution outperforms other AL methods based on the experimental results.

The proposed approach provides a new paradigm for HSI classification. In future, more remote sensing applications like change detection, segmentation, and spectral unmixing can be solved using deep reinforcement learning.

Author Contributions UP has done literature search, data analysis, and drafted; VP critically revised the work.

Funding No funding was obtained for this study.

Data availability statement Manuscript has no associated data.

Declarations

Conflict of interest None declare.

Ethical approval Not applicable.

References

1. Ghasrodashti EK, Sharma N (2021) Hyperspectral image classification using an extended auto-encoder method. Signal Process Image Commun 92:116111
2. Mou L, Ghamisi P, Zhu XX (2017) Deep recurrent neural networks for hyperspectral image classification. IEEE Trans Geosci Remote Sens 55(7):3639–3655
3. Hu W, Huang Y, Wei L, Zhang F, Li H (2015) Deep convolutional neural networks for hyperspectral image classification. J Sens. https://doi.org/10.1155/2015/258619
4. Chen Y, Jiang H, Li C, Jia X, Ghamisi P (2016) Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. IEEE Trans Geosci Remote Sens 54(10):6232–6251
5. Patel U, Dave H, Patel V (2020) Hyperspectral image classification using semi-supervised learning with label propagation. In: 2020 IEEE India Geoscience and Remote Sensing Symposium (InGARSS). IEEE, pp 205–208
6. He X, Chen Y, Ghamisi P (2019) Heterogeneous transfer learning for hyperspectral image classification based on convolutional neural network. IEEE Trans Geosci Remote Sens 58(5):3246–3263
7. Qu Y, Baghbaderani RK, Qi H (2019) Few-shot hyperspectral image classification through multitask transfer learning. In: 2019 10th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS). IEEE, pp 1–5
8. Jia S, Jiang S, Lin Z, Li N, Xu M, Yu S (2021) A survey: deep learning for hyperspectral image classification with few labeled samples. Neurocomputing 448:179–204
9. Ren P, Xiao Y, Chang X, Huang P-Y, Li Z, Gupta BB, Chen X, Wang X (2021) A survey of deep active learning. ACM Comput Surv (CSUR) 54(9):1–40
10. Cao X, Yao J, Xu Z, Meng D (2020) Hyperspectral image classification with convolutional neural network and active learning. IEEE Trans Geosci Remote Sens 58(7):4604–4616
11. Datta D, Mallick PK, Bhoi AK, Ijaz MF, Shafi J, Choi J (2022) Hyperspectral image classification: potentials, challenges, and future directions. Comput Intell Neurosci. https://doi.org/10.1155/2022/3854635
12. Patel U, Dave H, Patel V (2021) Hyperspectral image classification using uncertainty and diversity based active learning. Scalable Comput Pract Exp 22(3):283–293
13. Ahn KU, Park CS (2020) Application of deep Q-networks for model-free optimal control balancing between different HVAC systems. Sci Technol Built Environ 26(1):61–74
14. Song L, Fan W (2021) Traffic signal control under mixed traffic with connected and automated vehicles: a transfer-based deep reinforcement learning approach. IEEE Access 9:145228–145237
15. Altuner AB, Kilimci ZH (2021) A novel deep reinforcement learning based stock direction prediction using knowledge graph and community aware sentiments. arXiv preprint arXiv:2107.00931
16. Kumari A, Tanwar S (2021) A reinforcement learning-based secure demand response scheme for smart grid system. IEEE Internet Things J 9:2180–2191
17. Liang W, Huang W, Long J, Zhang K, Li K-C, Zhang D (2020) Deep reinforcement learning for resource protection and real-time detection in IoT environment. IEEE Internet Things J 7(7):6392–6401

18. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
19. Fang M, Li Y, Cohn T (2017) Learning how to active learn: a deep reinforcement learning approach. arXiv preprint arXiv:1708.02383
20. Sun L, Gong Y (2019) Active learning for image classification: a deep reinforcement learning approach. In: 2019 2nd China Symposium on Cognitive Computing and Hybrid Intelligence (CCHI). IEEE, pp 71–76
21. Haußmann M, Hamprecht FA, Kandemir M (2019) Deep active learning with adaptive acquisition. arXiv preprint arXiv:1906.11471
22. Vu T, Liu M, Phung D, Haffari G (2019) Learning how to active learn by dreaming. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp 4091–4101
23. Liu Z, Wang J, Gong S, Lu H, Tao D (2019) Deep reinforcement active learning for human-in-the-loop person re-identification. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 6122–6131
24. Rudovic O, Zhang M, Schuller B, Picard R (2019) Multi-modal active learning from human data: a deep reinforcement learning approach. In: 2019 International Conference on Multimodal Interaction, pp 6–15
25. Rudovic O, Park HW, Busche J, Schuller B, Breazeal C, Picard RW (2019) Personalized estimation of engagement from videos using active learning with deep reinforcement learning. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE, pp 217–226
26. Casanova A, Pinheiro PO, Rostamzadeh N, Pal CJ (2020) Reinforced active learning for image segmentation. arXiv preprint arXiv:2002.06583
27. Taguchi Y, Hino H, Kameyama K (2021) Pre-training acquisition functions by deep reinforcement learning for fixed budget active learning. Neural Process Lett 53(3):1945–1962
28. Konyushkova K, Sznitman R, Fua P (2018) Discovering general-purpose active learning strategies. arXiv preprint arXiv:1810.04114
29. Hsu W-N, Lin H-T (2015) Active learning by learning. In: Twenty-Ninth AAAI Conference on Artificial Intelligence
30. Woodward M, Finn C (2017) Active one-shot learning. arXiv preprint arXiv:1702.06559
31. Huang H, Feng Y, Huang J, Zhang J, Chen L (2019) A reinforcement one-shot active learning approach for aircraft type recognition. IEEE Access 7:147204–147214
32. Mou L, Saha S, Hua Y, Bovolo F, Bruzzone L, Zhu XX (2021) Deep reinforcement learning for band selection in hyperspectral image classification. IEEE Trans Geosci Remote Sens 60:1–14
33. Feng J, Li D, Gu J, Cao X, Shang R, Zhang X, Jiao L (2021) Deep reinforcement learning for semi-supervised hyperspectral band selection. IEEE Trans Geosci Remote Sens 60:1–19
34. Wang Z, Zhang K, Zhang J, Chen G, Ma X, Xin G, Kang J, Zhao H, Yang Y (2022) Deep reinforcement learning and adaptive policy transfer for generalizable well control optimization. J Petrol Sci Eng 217:110868
35. Roy SK, Krishna G, Dubey SR, Chaudhuri BB (2020) Hybridsn: exploring 3D–2D CNN feature hierarchy for hyperspectral image classification. IEEE Geosci Remote Sens Lett 17:277–281
36. Grandini M, Bagli E, Visani G (2020) Metrics for multi-class classification: an overview. arXiv preprint arXiv:2008.05756

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.