The Astrophysical Journal Supplement Series, 265:34 (8pp), 2023 April https://doi.org/10.3847/1538-4365/acb76d

Application of Deep Reinforcement Learning to Major Solar Flare Forecasting

Kangwoo Yi[1], Yong-Jae Moon[1,2], and Hyun-Jin Jeong[2]
[1] Department of Astronomy and Space Science, College of Applied Science, Kyung Hee University, 1732, Deogyeongdae-ro, Giheung-gu, Yongin-si Gyunggi-do, 17104, Republic of Korea; moonyj@khu.ac.kr
[2] School of Space Research, Kyung Hee University, 1732, Deogyeongdae-ro, Giheung-gu, Yongin-si Gyunggi-do, 17104, Republic of Korea
Received 2022 September 8; revised 2023 January 6; accepted 2023 January 29; published 2023 March 15

Abstract

In this study, we present the application of deep reinforcement learning to the forecasting of major solar flares. For this, we consider full-disk magnetograms at 00:00 UT from the Solar and Heliospheric Observatory/Michelson Doppler Imager (1996–2010) and the Solar Dynamics Observatory/Helioseismic and Magnetic Imager (2011–2019), as well as Geostationary Operational Environmental Satellite X-ray flare data. We apply Deep Q-Network (DQN) and Double DQN, which are popular deep reinforcement learning methods, to predict "Yes or No" for daily M- and X-class flare occurrence. The reward functions, consisting of four rewards for true positive, false positive, false negative, and true negative, are used for our models. The major results of this study are as follows. First, our deep-learning models successfully predict major solar flares with good skill scores, such as HSS, F1, TSS, and ApSS. Second, the performance of our models depends on the reward function, learning method, and target agent update time. Third, the performance of our deep-learning models is noticeably better than that of a convolutional neural network (CNN) model with the same structure: 0.38 (CNN) to 0.44 (ours) for HSS, 0.47 to 0.52 for F1, 0.53 to 0.59 for TSS, and 0.09 to 0.12 for ApSS.

Unified Astronomy Thesaurus concepts: Solar flares (1496); Neural networks (1933); The Sun (1693)

1. Introduction

A solar flare is a sudden flash on the Sun, releasing a huge amount of energy in a broad spectrum of emissions and accelerating particles to interplanetary space. The flare is categorized into five classes (A, B, C, M, and X, from smallest to biggest) according to its strength in the soft X-ray flux (0.1–0.8 nm). Each class represents a 10-fold energy increase. Intense X-ray fluxes and accelerated particles from strong flares result in enormous economic losses such as satellite drag, GPS disruption, and radio fade-outs.

One of the major challenges in flare forecasting is handling class imbalance issues. The solar flare occurrence rate depends on its energy. Strong flares ($\geqslant$M class) rarely occur while weak flares ($\leqslant$C class) frequently. The flare data reported by the National Oceanic and Atmospheric Administration (NOAA) during solar cycle 23 (1996–2008) shows that the occurrence ratio between X-, M-, and C-class flares is around 1:10:100. Prediction methods take common events more frequently than rare ones and overfit to common events. To alleviate the class imbalance problem, previous studies use oversampling, undersampling, and weighted-loss methods (Bobra & Couvidat 2015; Liu et al. 2017, 2019; Nishizuka et al. 2018; Zheng et al. 2019; Cinto et al. 2020; Jiao et al. 2020; Li et al. 2020; Deng et al. 2021).

The class imbalance issue is essential in not only solar flare forecasting but also many classification problems. In deep learning, Lin et al. (2020) suggested using deep reinforcement learning to handle the class imbalance problem. Reinforcement learning is a machine-learning technique used to learn how to choose actions in order to maximize a reward. They assigned different rewards for true positive (TP; predicting positive when positive), false negative (FN; predicting negative when positive), false positive (FP; predicting positive when negative), and true negative (TN; predicting negative when negative) for the Deep Q-Network (DQN; Mnih et al. 2015) classification model. Their result suggests that deep reinforcement learning could be a more effective approach to imbalanced classification than other methods.

In this study, for the first time, we apply deep reinforcement learning to major solar flare forecasts. For this, we develop DQN and Double DQN (van Hasselt et al. 2016) flare-forecasting models, which predict "Yes or No" for the daily occurrence of $\geqslant$M-class flares. The solar full-disk line-of-sight magnetograms at 00:00 UT from the Solar and Heliospheric Observatory (SOHO; Domingo et al. 1995)/Michelson Doppler Imager (MDI; Scherrer et al. 1995) and Solar Dynamics Observatory (SDO; Pesnell et al. 2012)/Helioseismic and Magnetic Imager (HMI; Schou et al. 2012) are used for model training and testing. We find that deep reinforcement learning with proper reward functions achieves high performance in view of the evaluation scores.

This paper is organized as follows. The data are described in Section 2. The structure of our model and reinforcement learning methods used for this research are described in Section 3. Details of the evaluation methods are described in Section 4. The results of the models are given in Section 5. Conclusions and discussions are presented in Section 6.

2. Data

Solar full-disk line-of-sight magnetograms at 00:00 UT from SOHO/MDI and SDO/HMI are used for model training and testing. The SOHO mission, a cooperative effort between the European Space Agency and NASA, was launched in 1995 December. One of the scientific instruments on SOHO, MDI, provides 1024 × 1024 full-disk solar magnetograms with 1.″98 per pixel at a cadence of 96 minutes. The SDO mission, launched in 2010 February by NASA, is designed to provide

Table 1
Data Set Information

	X	M	C	Non (<C)	Total
Training	93	647	1478	1696	3914
	740		3174		
Test	40	360	1240	1852	3492
	400		3092		

data to predict solar activity. One of the instruments on SDO, HMI, takes over the role of MDI. HMI produces the 4096×4096 full-disk magnetograms with $0.''5$ per pixel at 720 s cadence.

In order to merge the MDI data and HMI data, the spatial resolution of MDI and HMI are resized into 512×512 size by block average. In addition, we apply an equation of the relationship between MDI and HMI, proposed by Liu et al. (2012), to convert the MDI data to the HMI proxy data. The data values are expressed in a byte scale of ± 100 G, which shows well the magnetic features of the solar active area. The magnetograms are labeled separately as major flaring event days ($\geqslant$M1.0 Class) or non-major flare event days ($<$M1.0 Class) using Geostationary Operational Environmental Satellite (GOES) X-ray flare data.

Finally, we build the data set including solar cycle 23 data (3914 data; MDI from 1996 May to 2008 December) for training, and solar cycle 24 data (3492 data; 405 MDI from 2009 January to 2010 December; 3087 HMI from 2011 January to 2019 December) for the test. By using all periods of solar cycle 23 and solar cycle 24, we fully consider the solar cycle effect on the solar activity in both the training and test periods. The data set is fully described in Table 1.

3. Methods

3.1. Deep Reinforcement Learning

3.1.1. Q-learning and Deep Q-learning Network

In this study, we apply DQN to solar flare forecasting for magnetograms that are environments of solar flares. DQN and its variants are widely used to select an optimal action from the input environment. It optimizes itself to get the maximum reward that depends on the reward functions. If we consider that strong flares ($\geqslant$M class) are much more valuable than weak flares ($\leqslant$C class) and assign higher rewards and stronger penalties to the strong flares than weak flares, the deep reinforcement learning flare model could be optimized by considering strong flares to be more important than weak ones.

DQN is a deep-learning application of Q-learning (Watkins & Dayan 1992), which is a reinforcement learning algorithm that is used to learn the action value. In Q-learning, the learned action-value function Q is approximated to the optimal action-value function (Sutton & Barto 2018). Q-learning is an off-policy method (Baird 1995) that makes it possible to learn from a different policy, not a current policy (Francois-Lavet et al. 2018). Q-learning is also a model-free algorithm that uses a trial-and-error approach (Sutton & Barto 2018). A function of the Q-learning is expressed as

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_t + \gamma Q(S_{t+1}, A'_{t+1}) - Q(S_t, A_t)), \quad (1)$$

where S_t, A_t, and R_t represent the state, the action, and the reward from the action, at a given time t, respectively. α is the learning rate. γ is the discount factor for the estimated future value. Q represents the action-value function of the state S and the action A, which is calculated by the agent that decides what action to take. A is the action selected by the ϵ-greedy policy, and A' is the action selected by the greedy policy. The greedy policy is to select the most valuable action, and the ϵ-greedy policy is to select a random action with a probability ϵ and to select the most valuable action with a probability $1 - \epsilon$.

In order to apply Q-learning to deep learning, Mnih et al. (2015) developed DQN. Its flow chart is given in Figure 1. Mnih et al. (2015) constructed convolutional neural network (CNN; Lecun et al. 1998) agents to analyze the frames of the Atari game screen and to determine an optimal action. In addition, they used two key ideas for DQN. First, they separated the online agent Q and the target agent Q' for the Q-learning update to improve the stability of the method. The online agent, which uses the ϵ-greedy policy, calculates the Q value of the action in the current state. The target agent, which uses the greedy policy, calculates the Q value of the action in the next state. The online agent is updated at every training step, while the target agent is updated as a clone of the online agent after N iterations. The agent to be used for the operation is the target agent. Second, they used the experience replay technique (Lin 1993). Game playing and agent updating are asynchronous processes. The agent plays the Atari game without updating. It stores its experiences $e_t = (S_t, A_t, R_t, S_{t+1})$ at each time step t into the replay memory. The replay memory stores only the last N experience. The agent is updated using randomly selected experience data from the replay memory. By learning previous experiences stored in the replay memory, the agent can be trained more effectively and avoid issues with instability or divergence in the parameters (Mnih et al. 2013). With these ideas, the DQN model played Atari games at a human level.

3.1.2. Double Q-learning and Double DQN

Furthermore, we apply Double DQN to flare forecasting. Double DQN, which is a variant of DQN, is a deep-learning application of Double Q-learning (Hasselt 2010). One of Q-learning's problems is that it overestimates the action values because Q-learning uses the action with the maximum value to determine the value of the next state. To avoid overestimation, Hasselt (2010) developed Double Q-learning. Double Q-learning is similar to Q-learning but uses two agents, Q^A and Q^B. A function of Double Q-learning is given by

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_t + \gamma Q'(S_{t+1}, A^*_{t+1}) - Q(S_t, A_t)), \quad (2)$$

where Q is Q^A and Q' is Q^B when updating Q^A, and vice versa when updating Q^B. The update agent is selected randomly at every update. A^*_{t+1} is the maximum value action selected by the agent Q. Taking an action from a different agent reduces overestimation and performs well in some settings in which Q-learning performs poorly. van Hasselt et al. (2016) presented Double DQN to reduce the overestimation of DQN. Double Q-learning uses two agents to avoid overestimation, and DQN also uses two agents, the online agent and the target agent. A

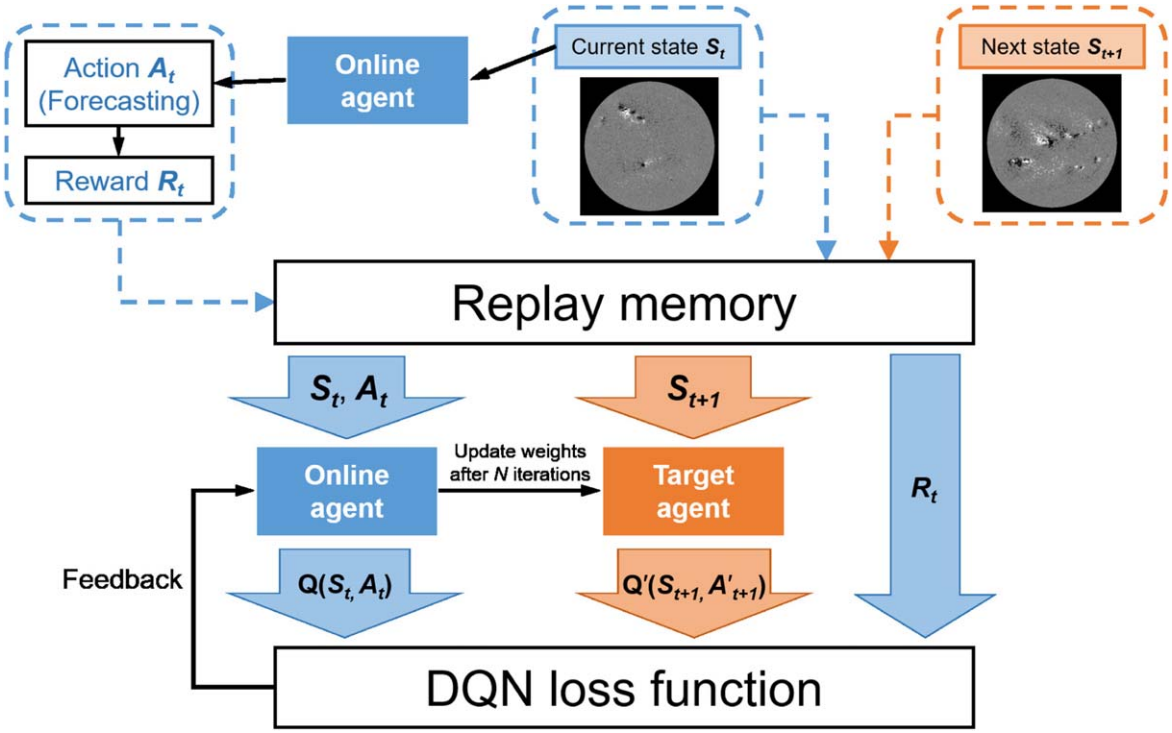

Figure 1. A flow chart of the DQN algorithm for flare forecasting. The model's experiences are stored in the replay memory. Randomly selected experiences in the replay memory are used for the DQN model training.

function of Double DQN is expressed as

$$Q^O(S_t, A_t) \leftarrow Q^O(S_t, A_t^O) + \alpha(R_t + \gamma Q^T(S_{t+1}, A_{t+1}^{O'}) - Q^O(S_t, A_t^O)), \quad (3)$$

where Q^O and Q^T are the online agent and the target agent, respectively. A^O and $A^{O'}$ indicate actions selected by the online agent with the ϵ-greedy policy and greedy policy, respectively. The target agent in a Double DQN estimates the value of the next state using the action $A_{t+1}^{O'}$ selected by the online agent. The main difference between Double Q-learning and Double DQN is that in Double DQN, the online agent and target agent do not replace each other. van Hasselt et al. (2016) showed that Double DQN better performs than DQN in many environments.

3.1.3. Reinforcement Learning for Imbalanced Classification

Lin et al. (2020) proposed a DQN model for imbalanced classification. Their model was evaluated with various data sets: IMDB (Maas et al. 2011), Cifar-10 (Krizhevsky & Hinton 2009), Mnist (Lecun et al. 1998), and Fashion-Mnist (Xiao et al. 2017). They showed that their DQN model performed better than other models that are based on imbalanced classification methods such as oversampling and undersampling.

3.2. Our Model

CNNs are widely used in 2D image processing for classification. CNNs use many convolutional filters that have different weights and are complexly connected with each other to analyze values and structures of input data. Many deep-learning solar flare models used CNNs to analyze magnetograms (Nishizuka et al. 2018; Park et al. 2018; Zheng et al. 2019; Tang et al. 2021a; Yi et al. 2021). By visual explanation using Grad-CAM (Selvaraju et al. 2017) and guided backpropagation (Springenberg et al. 2015), Yi et al. (2021) showed that CNNs can recognize an active region in a full-disk magnetogram and consider it and the polarity inversion line for flare prediction. They tested the deep-learning model, adjusting the number and size of convolutional filters, the number of dense blocks, and the pooling mechanism, and found that the proposed hyperparameters are good for magnetogram analysis for solar flare forecasting.

In this study, to examine the effect of deep reinforcement learning on flare forecasting, we follow Yi et al.'s (2021) model structure and hyperparameters. Figure 2 shows the adopted CNN model architecture. K is kernel size and D is the number of feature dimensions. The model consists of an initial block, five dense blocks using a dense connection (Huang et al. 2017), and a last block. The initial block consists of the 3×3 convolutional kernel and 2×2 max pooling layer. Dense blocks contain a batch normalization (BN; Ioffe & Szegedy 2015), a rectified linear unit (ReLU; Nair & Hinton 2010), a 1×1 convolutional kernel, a BN, a ReLU, a 3×3 convolutional kernel, a concatenation layer, and a 2×2 average pooling layer, in this order. The last block consists of a BN, a 2×2 average pooling layer, and a fully connected layer. Two outputs of the last block are scores for the flare and nonflare. A higher score is used as a result of the model prediction.

The CNN architecture (Figure 2) that we employ is used as an agent for our deep reinforcement learning models. The deep reinforcement learning models use RMSprop optimizer, proposed by Geoff Hinton in Lecture 6e of his Coursera

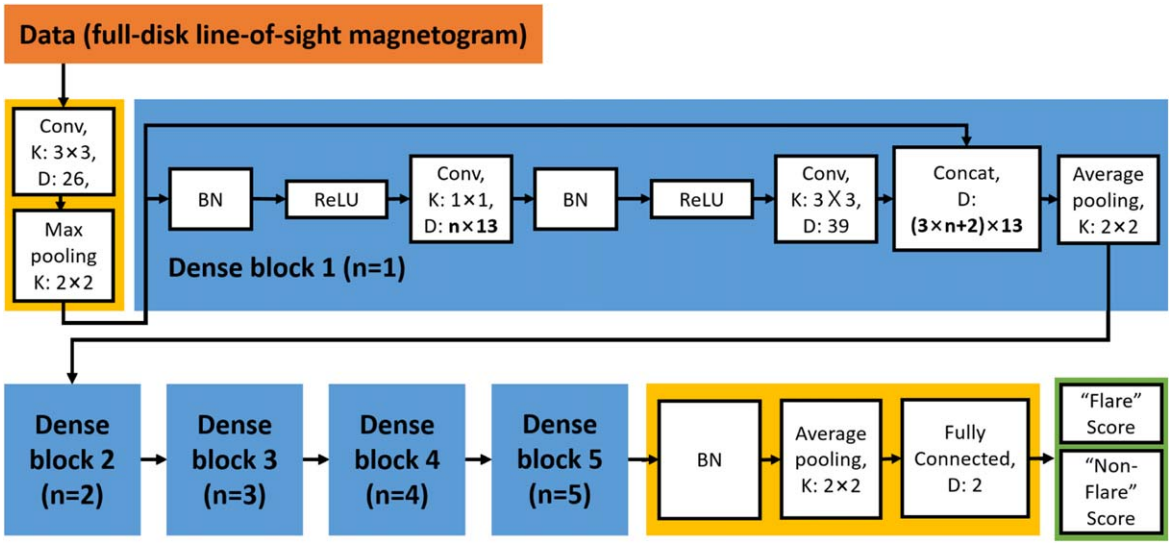

Figure 2. CNN architecture from Yi et al. (2021). K is kernel size. D is the number of feature dimensions and is displayed in the case that it is changed in the layer.

class,[3] and CNN models use the Adam optimizer (Kingma & Ba 2014).

Our models use mean square error loss functions. For a CNN, its loss function is

$$L_{\mathrm{CNN}} = (x_i - y_i)^2, \tag{4}$$

where x_i denotes observation, and y_i denotes model result. In DQN, its loss function is

$$L_{\mathrm{DQN}} = (R_t + \gamma Q'(S_{t+1}, A'_{t+1}) - Q(S_t, A_t))^2. \tag{5}$$

In Double DQN, its loss function is

$$L_{\mathrm{DoubleDQN}} = (R_t + \gamma Q^T(S_{t+1}, A^{O'}_{t+1}) - Q^O(S_t, A_t{}^O))^2. \tag{6}$$

4. Evaluation Methods

4.1. Evaluation Metrics

Different evaluation metrics indicate different aspects of model performance (Barnes et al. 2016; Leka et al. 2019). To evaluate the model fairly from various points of view, we consider four skill scores.

The Heidke skill score (HSS; Heidke 1926),

$$\mathrm{HSS} = \frac{2[(\mathrm{TP} \times \mathrm{TN}) - (\mathrm{FP} \times \mathrm{FN})]}{(\mathrm{TP} + \mathrm{FN}) \times (\mathrm{FN} + \mathrm{TN}) + (\mathrm{TP} + \mathrm{FP}) \times (\mathrm{FP} + \mathrm{TN})}, \tag{7}$$

informs the ratio of improvement over random forecasting. It is frequently used in flare-forecasting evaluation; however, it has to be handled carefully because its range depends on the event rate. In an imbalanced data set, its range is from $-2(P \times N)/(P^2 + N^2)$ to 1, where P is the number of positive data while N is the negative data. A negative value means that random forecasting is better, 0 is no skill prediction, and 1 indicates perfect forecasting.

[3] https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

The F1 score, which is a harmonic mean between the precision $\left(\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}\right)$ and recall $\left(\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}\right)$, is given by

$$\mathrm{F1} = \frac{\mathrm{TP}}{\mathrm{TP} + 0.5 \times (\mathrm{FP} + \mathrm{FN})}. \tag{8}$$

It is a measure of model performance that ranges from 0 to 1, assuming that precision and recall have the same gradient for the evaluation score when both are equal. The F1 score is not biased toward either precision or recall, making it a good evaluation index to analyze the performance of a prediction model. It is used for the evaluation of sunspot analysis for classification (Tang et al. 2021b).

The true skill statistic (TSS; Allouche et al. 2006) evaluates the model in the range -1 to 1. It is described as

$$\mathrm{TSS} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} - \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}. \tag{9}$$

TSS has mainly been used to compare flare models since Bloomfield et al. (2012). They suggested that TSS be used as a standard metric for comparing flare models, as it is not impacted by changes in the ratio of positive to negative values in the test set (Woodcock 1976). However, flare models that overforecast can achieve a high score in TSS easily, while those that underforecast are less likely to (Leka et al. 2019). This overestimation problem becomes more serious as the data set is biased because the ratio of TP to positive data (TP+FN) becomes relatively large while the ratio of FP to negative data (FP+TN) becomes relatively small. For example, Nishizuka et al. (2018) achieved TSS = 0.80 while P:N was 0.03:0.97 and the false-alarm ratio $\left(\frac{\mathrm{FP}}{\mathrm{TP}+\mathrm{FP}}\right)$ was 0.82. In addition, Lim et al. (2019) achieved 0.91 TSS while P:N was 0.01:0.99 and the FAR was 0.9.

Appleman's skill score (ApSS; Appleman 1960), which is similar to HSS except that it informs the improvement relative to unskilled forecasting, is given by

$$\mathrm{ApSS} = \left\{ \begin{array}{l} \frac{\mathrm{TP}-\mathrm{FP}}{\mathrm{TP}+\mathrm{FN}}, \text{ if event rate } < 0.5 \\ \frac{\mathrm{TN}-\mathrm{FN}}{\mathrm{FP}+\mathrm{TN}}, \text{ if event rate } \geqslant 0.5 \end{array} \right\}, \tag{10}$$

where the event rate is the ratio of the number of flare events to the total number of data. The ApSS considers both FN and FP to be equal, providing a good overview of the performance of the methods. Its scale depends on the data distribution, from $-\frac{N}{P}$ to 1. ApSS is a challenging evaluation metric because it becomes negative when FP is greater than TP, which is a common result of imbalanced data classification. It is common for models of major solar flares to achieve a high TSS while having a negative ApSS. For example, Nishizuka et al. (2018) achieved TSS = 0.80 while ApSS was −3.36. In addition, Wang et al. (2020) achieved TSS = 0.68 while ApSS was −1.18.

4.2. Strategy for Comparison

The focus of this study is to determine the effectiveness of reinforcement learning for major flare forecasting. For this, we analyze the performance of the models using four skill scores: HSS, F1, TSS, and ApSS. As we mention in Section 4.1, different skill scores indicate different aspects of model performance. To measure the effects of reinforcement learning on model performance, we present an "optimum skill score" for each model. The "optimum skill score" is a skill score measured from the optimized model for each skill score. To optimize models, we apply two methods to the models. First, to avoid the overfitting problem and find the best performance, we train deep-learning models for enough epochs and select the model with the best score in epochs. Second, we determine the prediction threshold to measure the optimum skill score. This strategy suggests proper model development options for a specific task.

For comparison, we make CNN models whose structures are described in Figure 2. The CNN models are optimized within 250 epochs. However, reinforcement learning models are trained for 100 epochs because of training time. The training time for 250 epoch CNN models is 4 hr while the training time for 100 epoch reinforcement learning models is 28 hr on the Nvidia RTX 2080 GPU. We compare the performance of the CNN models optimized within 250 epochs and 100 epochs and find that the former outperforms the latter. In addition, we find that our CNN models are overfitted after 250 epochs. In order to compare the results more rigorously, we train the CNN models for 250 epochs while the reinforcement learning models are trained for 100 epochs.

5. Results

We make 64 deep reinforcement learning models, based on 16 reward functions, 2 reinforcement learning methods (DQN and Double DQN), and 2 target agent update times (one epoch and two epochs). The 16 reward functions used in this study are listed in Table 2. The reward of TN is fixed at 1, and other rewards indicate the ratio against TN. We assign different reward values for TP, FP, FN, and TN to find optimized reward functions. In order to increase the cost of rare events, many of the reward functions have a higher penalty for FN than FP.

Table 3 shows the results of the deep reinforcement learning flare models optimized for HSS, F1, TSS, and ApSS, respectively. Out of the 64 models, the highest and second-highest performances are 0.44 and 0.43 for HSS, 0.52 and 0.51 for F1, 0.59 and 0.58 for TSS, and 0.12 and 0.11 for ApSS (shown in bold).

Table 2
Reward Functions Used in This Study

TP	FP	FN	TN
1	−1	−1	1
4	−2	−8	1
4	−4	−16	1
4	−16	−32	1
4	−16	−64	1
8	−4	−16	1
8	−4	−32	1
8	−8	−16	1
8	−8	−64	1
8	−16	−8	1
8	−16	−32	1
8	−16	−64	1
8	−32	−16	1
4	−64	−4	1
4	−64	−8	1
8	−96	−8	1

The models that achieve the highest or second-highest score in HSS, F1, or TSS are listed in Table 2. However, in ApSS, we present the models that achieved the highest score because many models were tied for the highest ApSS. The number of listed models for each evaluation metric has no influence on the model analysis. It is hard to evaluate which model is better than others considering reward functions, methods, and the update time, we present all models achieving the highest and the second-highest score for HSS, F1, and TSS, and achieving the highest score for ApSS.

Model 0 is the CNN flare model that has the same structure as the other deep reinforcement learning models. CNN model results are presented for comparison with the deep reinforcement learning models. The reinforcement learning models show better scores than the CNN model, with an increase from 0.38 to 0.44 in HSS, from 0.47 to 0.52 for F1, from 0.50 to 0.58 in TSS, and from 0.09 to 0.12 in ApSS. Models 1, 2, 3, and 4 show good performance in HSS, F1, and/or TSS. In HSS and TSS, the models achieve optimized performance in all four model settings. The optimized HSS models show good performance with different five reward functions, while the optimized TSS models achieve good performance with two reward functions only. All models that showed good performance in HSS, F1, and TSS include a reward of −64 or lower in FP or FN in their reward functions. It indicates that in order to achieve high HSS, F1, or TSS, the model requires a strong penalty for missed forecasting. In ApSS, many models that use Double DQN and have an update time of two epochs achieve the best results. The optimized ApSS models show similar or a little better optimum scores in the HSS than the CNN model, while many of them have lower TSS than the CNN model.

Table 3
Results of the Optimized Deep Reinforcement Learning Flare Models

Model Number	Reward Function				Method	Update Time	Optimum Scores			
	TP	FP	FN	TN			HSS	F1	TSS	ApSS
0					CNN		0.38	0.47	0.53	0.09
HSS optimization models										
1	8	−64	−8	1	DQN	2 epochs	**0.44**	0.51	0.59	0.10
2	8	−16	−64	1	Double DQN	1 epoch	**0.44**	0.52	0.55	0.08
3	4	−16	−64	1	Double DQN	1 epoch	**0.43**	0.50	0.58	0.11
4	8	−8	−64	1	DQN	1 epoch	**0.43**	0.51	0.57	0.09
5	8	−96	−8	1	Double DQN	2 epochs	**0.43**	0.50	0.51	0.10
F1 optimization models										
2	8	−16	−64	1	Double DQN	1 epoch	0.44	**0.52**	0.55	0.08
1	8	−64	−8	1	DQN	2 epochs	0.44	**0.51**	0.59	0.10
4	8	−8	−64	1	DQN	1 epoch	0.43	**0.51**	0.57	0.09
TSS optimization models										
1	8	−64	−8	1	DQN	2 epochs	0.44	0.51	**0.59**	0.10
3	4	−16	−64	1	Double DQN	1 epoch	0.43	0.50	**0.58**	0.11
6	8	−64	−8	1	Double DQN	2 epochs	0.43	0.49	**0.58**	0.10
7	4	−16	−64	1	DQN	1 epoch	0.39	0.47	**0.58**	0.09
ApSS optimization models										
8	4	−4	−16	1	Double DQN	1 epoch	0.41	0.49	0.56	**0.12**
9	8	−96	−8	1	Double DQN	1 epoch	0.39	0.47	0.48	**0.12**
10	8	−4	−16	1	DQN	2 epochs	0.40	0.47	0.54	**0.12**
11	4	−2	−8	1	Double DQN	2 epochs	0.39	0.47	0.49	**0.12**
12	4	−4	−16	1	Double DQN	2 epochs	0.40	0.46	0.47	**0.12**
13	8	−4	−16	1	Double DQN	2 epochs	0.38	0.44	0.45	**0.12**
14	8	−4	−32	1	Double DQN	2 epochs	0.41	0.49	0.53	**0.12**
15	8	−8	−64	1	Double DQN	2 epochs	0.40	0.47	0.50	**0.12**

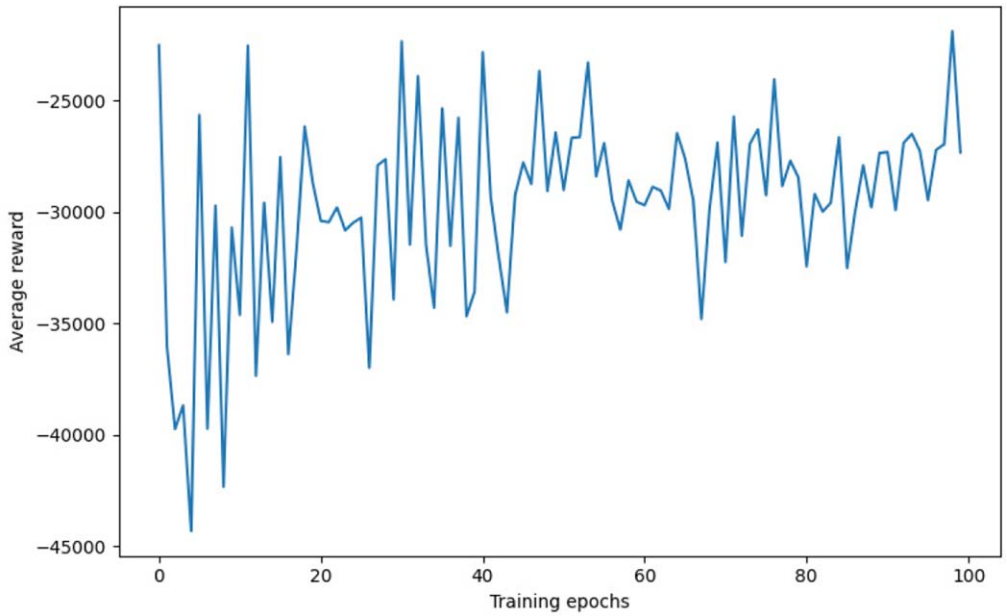

Figure 3. Training curve tracking the average rewards of the 10 models based on Model 3, using the test data set. Each value is the average reward of the models per epoch.

To estimate the uncertainty of the reinforcement learning flare model, we train Model 3 10 times with different random seeds and calculate averages and standard deviations of four skill scores. The results are 0.42 ± 0.02 for HSS, 0.49 ± 0.01 for F1, 0.58 ± 0.02 for TSS, and 0.10 ± 0.01 for ApSS.

In reinforcement learning research, fairly evaluating the progress of training is challenging. We follow the suggestion of

Bellemare et al. (2013), which calculates total rewards in an episode averaged over the number of games. The metric of the average total reward could be highly distributed because a small adjustment to the weights of the agent results in significant changes to the output of the agent (Mnih et al. 2013). Figure 3 provides the average training performance of the 10 models used to estimate the uncertainty. The plot is noisy, but as training progresses, the distribution of the plot decreases and converges to a high value.

6. Conclusion and Discussion

In this study, for the first time, we have presented the application of deep reinforcement learning to the flare forecast of a daily major solar flare. For this, we use full-disk magnetograms at 00:00 UT from SOHO/MDI (1996–2010) and SDO/HMI (2011–2019) and GOES X-ray flare data. We apply deep reinforcement learning to predict "Yes or No" for daily M- and X-class flare occurrences. The deep reinforcement learning flare models are trained using various reward functions, DQN and Double DQN methods, and one-epoch and two-epoch target agent update times. The performance of the model is presented using HSS, F1, TSS, and ApSS. The models are compared with other deep reinforcement learning models and the basic CNN models.

The major results of this study are as follows. First, our deep-learning models successfully predict major solar flares with good skill scores such as HSS, F1, TSS, and ApSS. Second, the performance of our models depends on the reward function, learning method, and target agent update time. Third, the performance of our deep-learning models is noticeably better than that of a CNN model with the same structure: 0.38 (CNN) to 0.44(ours) for HSS, 0.47 to 0.52 for F1, 0.53 to 0.59 for TSS, and 0.09 to 0.12 for ApSS.

We make other major solar flare-forecasting models considering oversampling, undersampling, and a weighted-loss function. When sampling a data set, we keep the ratio between X-, M-, C- and nonflare events (including $\leqslant$B flare) to preserve the effects of flare climatology in the data set. Sampling methods are applied to the training set only, not the test set. The best model uses the weighted-loss function based on the weight formula proposed by Ahmadzadeh et al. (2021). Our models that show good results, such as model 1 or 3 in Table 3, are a little better than or similar to the best weighted-loss function model. In addition, we make a deep reinforcement learning flare model using the reward function suggested by Lin et al. (2020). It shows a similar performance to model 3 in Table 3, but with a lower ApSS of 0.08.

The model structure and hyperparameters taken from Yi et al. (2021) are optimized for CNN-based flare forecasting. Forecasting performance could be improved if the model uses different structures and parameters optimized for deep reinforcement learning.

The performance of the flare-forecasting model depends on the solar cycle phase of the test data set (Wang et al. 2020). Many flare-forecasting researchers used one solar cycle data set (2010–2015 or 2010–2018) and divided it into the training data set and the test data set (Nishizuka et al. 2017; Liu et al. 2019; Zheng et al. 2019; Jiao et al. 2020; Li et al. 2020). To evaluate model performance considering solar cycle phase dependency, they used 10-fold cross-validation. For this, they randomly separated their data set into 10 folds, trained the model 10 times with a different set of 9 folds each training, and tested the model with the remaining fold. However, a randomly separated training data set and test data set could contain similar data observed at similar times, and the model performance could be overly evaluated. In this work, we do not use 10-fold cross-validation because our research fully considers solar cycle dependency (1996–2008 for training and 2009–2019 for testing). Instead, to evaluate uncertainty and generalize the performance of the model, we train the model 10 times with different random seeds and calculate the mean and standard deviation of the skill scores.

In this study, we show that many deep reinforcement learning flare models easily achieve better ApSS than the CNN model even when the other scores are low. This result suggests that deep reinforcement learning is useful for forecasting rare events precisely while reducing the false-alarm ratio. For example, deep reinforcement learning models could be useful in managing the schedule of flight attendants to minimize their annual cosmic radiation exposure.

We thank the numerous team members who have contributed to the success of the SDO mission, as well as the SOHO mission. This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2021R1A6A3A01088835) and the Korea Astronomy and Space Science Institute under the R&D program (project No. 2022-1-850-05) supervised by the Ministry of Science and ICT.

Software: PyTorch (Paszke et al. 2019), NumPy (Harris et al. 2020), Matplotlib (Hunter 2007), SciPy (Virtanen et al. 2020), Astropy (Robitaille et al. 2013; Price-Whelan et al. 2018), SunPy (The SunPy Community et al. 2020).

ORCID iDs

Kangwoo Yi https://orcid.org/0000-0003-4342-9483
Yong-Jae Moon https://orcid.org/0000-0001-6216-6944
Hyun-Jin Jeong https://orcid.org/0000-0003-4616-947X

References

Ahmadzadeh, A., Aydin, B., Georgoulis, M. K., et al. 2021, ApJS, 254, 23
Allouche, O., Tsoar, A., & Kadmon, R. 2006, J. Appl. Ecol., 43, 1223
Appleman, H. S. 1960, BAMS, 41, 64
Baird, L. C. 1995, in Proc. Twelfth Int. Conf. on Machine Learning, ICML'95, ed. A. Prieditis & S. Russell (San Francisco, CA: Morgan Kaufmann), 30
Barnes, G., Leka, K. D., Schrijver, C. J., et al. 2016, ApJ, 829, 89
Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. 2013, J. Artif. Intell. Res., 47, 253
Bloomfield, D. S., Higgins, P. A., McAteer, R. T. J., & Gallagher, P. T. 2012, ApJL, 747, L41
Bobra, M. G., & Couvidat, S. 2015, ApJ, 798, 135
Cinto, T., Gradvohl, A. L. S., Coelho, G. P., & da'Silva, A. E. A. 2020, MNRAS, 495, 3332
Deng, Z., Wang, F., Deng, H., et al. 2021, ApJ, 922, 232
Domingo, V., Fleck, B., & Poland, A. I. 1995, SoPh, 162, 1
Franćois-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., & Pineau, J. 2018, Found. Trends Mach. Learn., 11, 219
Harris, C. R., Millman, K. J., Van Der Walt, S. J., et al. 2020, Natur, 585, 357
Hasselt, H. 2010, in Advances in Neural Information Processing Systems 23, ed. J. Lafferty et al. (Red Hook, NY: Curran Associates), 2613, https://papers.nips.cc/paper/2010/hash/091d584fced301b442654dd8c23b3fc9-Abstract.html
Heidke, P. 1926, Geografiska Annaler, 8, 301
Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. 2017, in 2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (Los Alamitos, CA: IEEE Computer Society), 2261

Hunter, J. D. 2007, CSE, 9, 90
Ioffe, S., & Szegedy, C. 2015, in Proc. Machine Learning Research 37, Proc. 32nd Int. Conf. on Machine Learning, ed. F. Bach & D. Blei, 448, http://proceedings.mlr.press/v37/ioffe15.html
Jiao, Z., Sun, H., Wang, X., et al. 2020, SpWea, 18, e2020SW002440
Kingma, D. P., & Ba, J. 2014, arXiv:1412.6980
Krizhevsky, A., & Hinton, G. 2009, Learning Multiple Layers of Features from Tiny Images, Tech. Rep., Univ. of Toronto
Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. 1998, IEEEP, 86, 2278
Leka, K. D., Park, S.-H., Kusano, K., et al. 2019, ApJS, 243, 36
Li, X., Zheng, Y., Wang, X., & Wang, L. 2020, ApJ, 891, 10
Lim, D., Moon, Y.-J., Park, J., et al. 2019, JKAS, 52, 133
Lin, E., Chen, Q., & Qi, X. 2020, Appl. Intell., 50, 2488
Lin, L.-J. 1993, Reinforcement Learning for Robots Using Neural Networks., Tech. Rep., Carnegie Mellon Univ.
Liu, C., Deng, N., Wang, J. T. L., & Wang, H. 2017, ApJ, 843, 104
Liu, H., Liu, C., Wang, J. T. L., & Wang, H. 2019, ApJ, 877, 121
Liu, Y., Hoeksema, J. T., Scherrer, P. H., et al. 2012, SoPh, 279, 295
Maas, A. L., Daly, R. E., Pham, P. T., et al. 2011, in Proc. 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, ed. D. Lin et al. (Portland, OR: Association for Computational Linguistics), 142, http://www.aclweb.org/anthology/P11-1015
Mnih, V., Kavukcuoglu, K., Silver, D., et al. 2013, arXiv:1312.5602
Mnih, V., Kavukcuoglu, K., Silver, D., et al. 2015, Natur, 518, 529
Nair, V., & Hinton, G. E. 2010, in Proc. 27th Int. Conf. on Machine Learning, ed. J. Fürnkranz & T. Joachims (Madison, WI: Omnipress), 807, https://icml.cc/Conferences/2010/papers/432.pdf
Nishizuka, N., Sugiura, K., Kubo, Y., Den, M., & Ishii, M. 2018, ApJ, 858, 113
Nishizuka, N., Sugiura, K., Kubo, Y., et al. 2017, ApJ, 835, 156
Park, E., Moon, Y.-J., Shin, S., et al. 2018, ApJ, 869, 91
Paszke, A., Gross, S., Massa, F., et al. 2019, in Advances in Neural Information Processing Systems 32, ed. H. Wallach et al., 8026, https://papers.nips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html
Pesnell, W. D., Thompson, B. J., & Chamberlin, P. C. 2012, SoPh, 275, 3
Price-Whelan, A. M., Sipőcz, B., Günther, H., et al. 2018, AJ, 156, 123
Robitaille, T. P., Tollerud, E. J., Greenfield, P., et al. 2013, A&A, 558, A33
Scherrer, P. H., Bogart, R. S., Bush, R. I., et al. 1995, SoPh, 162, 129
Schou, J., Scherrer, P. H., Bush, R. I., et al. 2012, SoPh, 275, 229
Selvaraju, R. R., Cogswell, M., Das, A., et al. 2017, in 2017 IEEE Int. Conf. on Computer Vision (ICCV) (Los Alamitos, CA: IEEE Computer Society), 618
Springenberg, J., Dosovitskiy, A., Brox, T., & Riedmiller, M. 2015, arXiv:1412.6806
Sutton, R. S., & Barto, A. G. 2018, Reinforcement Learning: An Introduction (2nd ed.; Cambridge, MA: MIT Press)
Tang, R., Liao, W., Chen, Z., et al. 2021a, ApJS, 257, 50
Tang, R., Zeng, X., Chen, Z., et al. 2021b, ApJS, 257, 38
The SunPy Community, Barnes, W. T., Bobra, M. G., et al. 2020, ApJ, 890, 68
van Hasselt, H., Guez, A., & Silver, D. 2016, in Proc. AAAI Conf. on Artificial Intelligence 2016 (Palo Alto, CA: AAAI Press), 2094
Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, NatMe, 17, 261
Wang, X., Chen, Y., Toth, G., et al. 2020, ApJ, 895, 3
Watkins, C. J. C. H., & Dayan, P. 1992, Mach. Learn., 8, 279
Woodcock, F. 1976, MWRv, 104, 1209
Xiao, H., Rasul, K., & Vollgraf, R. 2017, arXiv:1708.07747
Yi, K., Moon, Y.-J., Lim, D., Park, E., & Lee, H. 2021, ApJ, 910, 8
Zheng, Y., Li, X., & Wang, X. 2019, ApJ, 885, 73