Model-Free Reinforcement Learning Mining the Optimal Policy in DICE Model

Jiaxun Lyu

Master in Master of Economics

Credits: 30

University of Oslo

Department of Economics

Faculty of Social Sciences

Submitted: May 2022

Acknowledgements

I would like to thank my supervisor Christian Traeger for his help and support I received during these troublesome times. Furthermore, I would like to thank my family and friends for their support and encouragement.

Contents

Abstract 1

1 Introduction 1

2 Methodology and Process 5

2.1 Formation of the Model 5

2.2 An Brief introduction to Reforement Learning 8

2.3 Deep Reinforcement Learning Environment for DICE 11

2.4 Initialization 15

3 Results 16

3.1 System Results 16

3.2 Base Results 20

3.3 Comparison of RL results with DICE2013 data 22

4 Discussion 26

5 Conclusion 28

Reference 30

Abstract

Economic dynamic models of climate change usually involve many variables, complex dynamics and uncertainties. However, the limited computing tools do not have abilities to include these factors, so the complexity of the models is often reduced. In this paper, I investigated if the model-free deep reinforcement learning (RL) approach can provide a viable alternative in finding optimal strategies in IAM models with multi-objectives. As the first step to promote RL in large-scale IAM models, I adapted an IAM based on DICE baseline into an OpenAI Gym environment. I use stable_baseline 3 as the RL framework, and apply Soft Actor Critic (SAC), which is one of the most advanced model-free RL training algorithms nowadays. Policy was learned through interactions with the environment without knowledge of model dynamics. After 88,000 timesteps, I got a learning strategy in my model, which consists of annual abatement rate and consumption. The results are compared with the baseline policy data in DICE 2013, showing a high consistency and accuracy. It demonstrates the potential of RL framework for economic dynamics.

Keywords: IAM, DICE, Reinforcement Learning, Model-Free

1 Introduction

Climate Economy has emerged as climate change is one of the greatest challenges that mankind facing and is becoming a hot topic of academic research.

Nordhaus (1975) pioneered the integration of the economic and climate systems within a modeling framework, using economic principles to assess and compare the benefits and costs faced by different climate scenarios to analyze the effects of climate policies. Since then, academic research has been conducted to study climate change issues using modern methods of economic analysis. Stern (2006) published the classic work on the economics of climate change, which laid the foundation for the economics of climate change. Since then, economists have constructed several Integrated Assessment Models (IAM), which have become the mainstream tool for analyzing the effects of climate change policies in the Intergovernmental Panel on Climate Change (IPCC) and in some countries (Clarke et al., 2009).

The Integrated Assessment Model (IAM) describe the continuous interactions between climate and economics. IAM is based on a cost-benefit analysis. It uses the assumptions and analysis methods of traditional economics to explore the impact of specific topics such as

emission reduction, abatement control, and assessing on optimal economic policies in climate change issues. IPCC classifies models into two main categories: Policy Optimization Models (POMs) and Policy Evaluation Models (PEMs). One of the most popular POMS is the Dynamic Integrated Model of Climate and the Economy (DICE) (Nordhaus,1994, 2007, 2010). Being a deterministic model, it can be directly solved with the power of nonlinear programming, such as the Generalized Reduced Gradient (Brooke et al. 2005).

However, it is a more difficult task to apply these methods to such complex and uncertain problems as climate change: firstly, climate change and many economic processes are unpredictable because of their inherent nature. Numerous variables affecting the climate change phenomenon and economics, among which, technological progress, human consumption patterns, energy use patterns, people's subjective judgments on certain value scales etc., can easily overturn the results of an evaluation. When a large degree of uncertainties must be considered to analyze the impact of climate policies, the complexity of obtaining a solution from these models will be increased exponentially.

The issue has been addressed in several works. A growing literature has been attempting to apply probabilistic uncertainty to IAMs. One approach is Monte Carlo simulation, e.g., Manne and Richels (1994), Reilly et al. (1987), Scott et al. (1999), Webster et al. (2008), Webster et al. (2009). However, Monte Carlo simulation can be far from accurate, as Crost and Traeger (2013) pointed out that and get the sign of uncertainty wrong from Monte Carlo simulation. Another notable approach is stochastic dynamic programming. Baker and Solak (2011) introduce a stochastic dynamic programming version of an IAM. However, stochastic dynamic programming algorithms based on value iteration or policy iteration (Bertsekas 2007) suffer greatly from the curse of dimensionality, where the complexity of the problem grows exponentially with the number of states (Bellman, 1957). Therefore, to address the complexity problems we need new methods from other fields.

Recently, machine learning methods have been used to excel traditional algorithms in various fields. Traditionally, machine learning algorithms can be classified into supervised learning, unsupervised learning, and reinforcement learning. Methods of supervised learning and unsupervised learning have been applied in economic dynamics, which are mainly used to reduce dimension and approximate function. For example, Duffy and McNelis (2001) used neural networks to express expected functions. Renner and Scheidegger (2018) and Scheidegger and Billions (2019) use Gaussian-process machine learning and active subspace methods for dimensionality reduction. Fernández-Villaverde et al. (2019) for applying neural networks to approximate the Russell and Smith (1998) model.

Reinforcement Learning is a powerful mathematical framework for experience-driven autonomous learning solving sequential decision problem (Sutton, 1996). It was combined with deep learning, in which neural network were constructed to represent strategies or value functions, such as TD-Gammon (Tesauro, 1995), which is now called "Deep RL". Generally speaking, "Deep" means that deep neural networks is involved as an approximator of reinforcement learning structures such as the values functions. When we talk about reinforcement learning, we usually mean deep RL. In this document, we also pay attention to a specific type of reinforcement learning, that is, model-free learning, as up to now, most deep reinforcement learning algorithms are developed for model-free learning.

Although RL algorithms have been successful in many fields, there have been fewer studies that have explored the power of this category of machine learning in Economics compared to other machine leaning methods. One early adaption is Jirniy and Lepetyuk (2012) incorporated “Temporal Difference” (TD) (Sutton, 1988) in solving Krusell and Smith (1998) model. More recent work can be found in Lilia Maliar et al. (2021), presenting a more comprehensive review of reinforcement learning, and brought up a lifetime reward algorithm based on such an approach.

Impressive applications in computing, including the pioneering AlphaGo (Silver, 2016), inspired me to try to integrate such a framework into IAMs.

I consider deep RL approaches to be especially well-suited to IAMs as:

1. Economic dynamics can generally be described as a state transition process (time series), involving optimization, such as minimum cost or maximum utility. This is exactly the RL problem.

2. The models in economics can be represented as simulators and can be interacted repetitively by the agent and therefore an abundant data can always be obtained.

3. The deep network model makes RL more flexible for online updating and adapts to more varied environments.

4. RL have proven successful solving large-scale systems and decision-making policies with deep neural networks.

Therefore, as the first step to evaluate the potential RL approach in large scale IAM, I first brough RL into an IAM based on DICE2013 (Nordhaus and P. Sztorc, 2013).

DICE2013 draws on the earlier DICE model created by Nordhaus (1994, 2007, 2010), which is a global aggregation model that aggregates output, capital stock, technology and emission levels of different countries, as well as a simplified analytical and empirical model. The model approaches the economics of climate change from neoclassical economic growth

theory. Its main purpose is to run as a policy optimization model or a simple forecasting model. Its aim is to maximize the objective function (i.e., social welfare). It prevents the hazards of climate change by investing in capital, technology, and education in each economy, forgoing or reducing current consumption, and increasing future consumption by working to reduce emissions. It integrates the relevant factors affecting economic growth, emissions, the carbon cycle, climate change, and policies to improve climate change. The equations in the model follow different laws of economics, ecology, and earth sciences, and then, mathematical optimization software is used to plan for economic and environmental outcomes.

The model assumes that economic and climate policies are based on maximizing total consumption, and the results of the DICE2013 runs include projections of world gross product, per capita consumption, industrial CO2 emissions, CO2 concentration, temperature change, emission control rate, and carbon price for six emission scenarios.

I implemented the IAM based on DICE2013(Nordhaus and P. Sztorc, 2013) as a simulator (environment) with Python. In designing the environment, I took advantage of the OpenAI Gym (Brockman et al., 2016) environment instance, which is the standard API for building machine learning environment with Python. I also use stable_baseline3 (Raffin et al., 2021) as the RL framework. I applied Soft Actor Critic (SAC) (Haarnoja et al., 2018a, 2018b, 208c), which is one of the current state-of-the-art model-free RL training algorithms. I obtained the training results and compared the optimal policy with DICE2013 under baseline scenario.

Because of the limited time in this project, the scope of the study is limited to the economics part of the DICE model (which does not include the atmospheric physics part). By tapping the emission control rate as a solution target for RL, the prediction of world gross product, per capita consumption, industrial CO2 emissions, emission control rate, and carbon price is achieved, and the results are compared with the baseline DICE2013, which proves that the platform is valid, and the results are satisfactory for the continued evolution of the RL environment as a solution for the DICE model.

The rest of the paper is organized as follows:

In Section 2, we will explain the methods and procedures used. Section 2.1 focuses on the formulation of economic relations in the DICE2013 model. Section 2.2 is a brief introduction to reinforcement learning. In Section 2.3, we will focus on how to translate our proposed DICE model into a reinforcement learning problem. Including how to design reward function and action space, how to choose reinforcement learning algorithm. In Sections 2.4 and 2.5, we

describe the experimental procedure and how to tune the RL parameters. In Section 3, we present the findings. In Section 3.1, the system operation of the RL framework will be presented. In Sections 3.2 and 3.3, we present data obtained from RL system predictions to demonstrate the effectiveness of the system. In Section 4, there is some discussion on the practice of converting from DICE problems to RL problems. Finally, Section 5 is the conclusion.

2 Methodology and Process

In this section I present my study of the economics dynamics of DICE2013, which aims to address the abatement rates while maximizing the social welfare.

The research work is modeled based on DICE2013 under the baseline scenario (or without controls) to develop the analysis. According to DICE2013 (Nordhaus 2013), the baseline scenario represents the results of market and policy factors that currently exist. In other words, the baseline model attempts to predict the level and growth of key economic and environmental variables from a positive perspective, as is the case with current climate change policies. the baseline scenario is an indefinite extension of current policies as of 2010.

Due to timing reasons, this study is limited to the study of economic growth and carbon emissions and does not involve the geophysical component of the study, using only impairment data. The simplification of the model does not affect the significance or the methodology of the study.

As the time step of DICE2013 has been changed to five years. To maintain synchronization with the DICE2013 cycle, the software also uses 5 years as a period. Of course, this software cycle can be changed to one year without affecting the results of the study.

2.1 Formation of the Model

This section focuses on the economic relations and formulations of the research question.

The model is based on DICE2013. I also adapted the notions from DICE2013 (Nordhaus and P. Sztorc, 2013).

The model solves total wealfare,

$$W = \max \sum_{t=1}^{T} U\,[c(t), L(t)]R(t) \tag{1}$$

The utility is

$$U[c(t), L(t)] = L\,(t)c(t)^{(1-\alpha)}/(1-\alpha) \tag{2}$$

c(t) is per capita consumption, L(t) is population as well as labor inputs, and R(t) is the discount factor. The pure rate of social time preference, ρ, is the discount rate which provides the welfare weights on the utilities of different generations.

$$R(t) = (1+\rho)^{(-t)} \tag{3}$$

Net GDP Y:

$$Y_t = F(A_t, L_t, K_t, E_t) = A_t K_t^{\gamma} L_t^{1-\gamma}\big(1 - \Lambda_t(\mu_t)\big) \tag{4}$$

The function $\Lambda_t(\mu_t)$ is Nordhaus' abatement cost function. DICE2013 assumes that abatement costs to be in the form of

$$\Lambda(t) = \theta_1(t)\mu(t)^{\theta_2} \tag{5}$$

where μ(t) is emissions control rate.

Capital stock comes from the accumulation of capital and new investments it depreciates over time.

$$K(t+1) = F\big(K(t), I(t)\big) = \delta_K K(t) + I(t) \tag{6}$$

$$I(t) = Q(t)\zeta(t) \tag{7}$$

Temperatures above pre-industrial level cause a damage,

$$Q(t) = Y_t(1 + \Omega_t) \tag{8}$$

where Ω_t is the total damage represented by the fraction of gross output. This study is limited to the study of economic growth and carbon emissions and does not involve temperature dynamics. The damage data used are directly from DICE 2013.

Total factor productivity (TFP) and the level of labor force/population increase with time.

$$A(t) = A(t-1)^{(1+g_A(t))} \tag{9}$$

$$g_A(t) = g_A(t-1)/(1+\delta_A) \tag{10}$$

$$L(t) = L(t-1) * \left(1 + g_L(t)\right) \tag{11}$$

$$g(t) = g_L(t-1)/(1+\ \delta_L) \tag{12}$$

Production causes emissions,

$$E_{\text{Ind}}\,(t) = \sigma(t)[1-\mu(t)]A(t)K(t)^{\gamma}L(t)^{1-\gamma} \tag{13}$$

This also means that the raw production value Y(t) includes 1) the production achieved with zero emissions 2) the production achieved with emissions. The ratio is μ: $(1-\mu)$. The corresponding output values are μY(t) and $(1-\mu)$Y(t). Corresponding to μY(t), the backstop technology is used to achieve emission reduction, and the backstop technology has a cost, which corresponds to the backstop price.

$$\mathrm{PB(t+1)} = \mathrm{PB(t)(1-g_b)} \tag{14}$$

As the backstop price decreases, the abatement rate gradually increases.

$$\theta 1(t) = \sigma(t)BP(t)/\theta 2/1000 \tag{15}$$

$$\sigma(t) = \sigma(t-1)[1+g_\sigma(t)] \tag{16}$$

$$g_\sigma(t) = g_\sigma(t-1)/(1+\delta_\sigma) \tag{17}$$

$$\mathrm{CP}(t+1) = \mathrm{CP(t)}(1+g_c)^5 \tag{18}$$

, where $\sigma(t)$ is the emissions intensity.

Note, this study focuses on the carbon emission control rate and the specific baseline of the DICE 2013 model.

2.2 A Brief Introduction to Reinforcement Learning

In this section, reinforcement learning is briefly introduced to help the reader understand the research methods in advance, even though he knows little about reinforcement learning. We suggest that reader refer to detailed explanations, such as the amazing introduction done by Sutton (1998).

Generally, RL Problem Solving markov decision processes (MDP). A MDP can be formalized as $\langle S, A, R, P, \gamma \rangle$. S is the set of states, A is the set of actions, R is the reward function, and P describes the probability distribution of moving to the next state s', $p(s' \mid s, a)$. At each time step t, the agent:

5. observes the state of $s_t \in S$;
6. takes and action $a_t \in A$;
7. and receive an instant reward $r: S \times A \to R$.

A policy π is to generate $a \sim \pi(\cdot \mid s)$ which action to take according to s . The goal of RL is to find the optimal policy π^* that maximize the lifetime discontinued return

$$V^{\pi}(s) = E\left[\sum_{t} \gamma^t r_t\right] \tag{19}$$

this is referred as the (state) value function, where $\gamma \in [0,1]$ is the discount factor. The optimal state-value function is

$$V^*(s) = \max_{\pi} V^{\pi}(s) \tag{20}$$

Please note that in practice of the model-free RL application, R and P are not observed. Agent only knows where the states is, what actions agents can take (a), and a reward (r) indicating whether our actions are good or bad, even though it being built in the environment in the code.

RL algorithms can be divided into 3 categories: value-based, policy-based and actor-critic. We only introduce the simplest RL algorithm here because it defines some important symbols.

First, we studied the value-based approach. Value-based methods draws lessons from the concept of dynamic programming, that is, value function iteration and strategy function iteration. Value-based reinforcement learning algorithm calculates the value of each action according to the current state, and then selects the actions according to the value according to a certain strategy (e.g., greedy).

We first introduce the bellman equation,

$$V(s) = \max_a \big(R(s,a) + \gamma V(s')\big) \tag{21}$$

This formula states that for a given state s and its value function $V(s)$, is determined by the action a that has the highest value among all the actions a that the agent can choose in this state s. It consists of the direct reward $R(s,a)$, and the value $V(s')$ of the new state s′ in the next step. $R(s,a)$ is the reward that is determined, and $V(s')$ is the value of the state after the next step,

With stochasticity, it can be written as

$$V(s) = \max_a \left(R(s,a) + \gamma \sum P(s,a,s')V(s')\right) \tag{22}$$

We then define $Q(s,a)$, to represent the values of the value where:

$$Q(s,a) = R(s,a) + \gamma \sum P(s,a,s')V(s') \tag{23}$$

This Q function is one of the functions to be learned inside the reinforcement learning algorithm. Because when we get this Q function and enter a certain state, its optimal behavior can be obtained by this Q function.

We also have,

$$V(s) = \max_a Q(s,a) \tag{24}$$

substitute into the previous formula we get,

$$Q(s,a) = \left(R(s,a) + \gamma \sum P(s,a,s') \max_{a'} Q(s',a')\right) \tag{25}$$

If we introduce TD method into this formula, we can get

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha \left(R(s,a) + \gamma \sum P(s,a,s') \max_{a'} Q(s',a')\right) - Q_{t-1}(s,a)\Big) \tag{26}$$

And this is the famous Q-learning algorithm (Watkins, 1989).

Policy-based methods are based directly on the current state, using the outputs of the action or the probability of the action. The most representative is the policy gradient (Sutton et al. 2000). The idea is to increase the probabilities of actions that have higher return and decrease the ones with a lower return.

More recent reinforcement learning is implemented through an Actor-Critic neural network. Actor-critic method is a time difference method (TD method), which combines the method based on the value function with the method based on the policy function. In this case, the Policy function is an "Actor" that gives actions. The value function acts as a "critic", evaluating the quality of the actions given by the actors, and generating TD difference signals to guide the updating of the value function and policy functions.

The Agent obtains the state of the environment (both actor and critic get the same state), and the actor selects the action from the action space according to the policy function and passes it to the environment. After receiving the action, the environment will give feedback (received by the critics) to the agent, and the feedback will give the environment a good or bad result. After receiving the feedback, the critic deduces the TD error according to the value function, and the TD error is passed on to the actor, who adjusts his strategy according to the TD error. At the same time, critics also adjust their Q value. The result of this interaction is that the actor's policy will choose the action with higher probability of scoring higher in an identical state, and the critic's Q value will become more and more stable. At last, the two sides reached an approximate optimal policy.

- The following are some commonly used Reinforcement Learning algorithms:
- DQN (Minh et al. 2013): An algorithm combining deep learning and Q-learning. Designed for discrete action spaces.
- DDPG (Silver et al. 2014): Off-policy method making neural network updates are more efficient, can only be used with continuous action spaces.
- TD3 (Fujimoto et al, 2018): It introduced 3 tricks to improve the performance of DDPG: Clipped Double-Q Learning, “Delayed” Policy Updates, Target Policy Smoothing. It can work with continuous action spaces.
- SAC (Haarnoja et al, 2018a, 2018b, 2018c): Introduce maximum entropy to actor-critic model. So that the agent can explore more uniformly and easier to adjust when faced with interference.

2.3 Deep Reinforcement Learning Environment for DICE

In this section, we will focus on the integration of DICE models with Reinforcement Learning to determine the reward function, action space and observation space employed in Reinforcement Learning for our research topic. The related research of the DICE model has been introduced in Section 2.1, and the relevant knowledge of Reinforcement Learning is briefly introduced in Section 2.2.

Now, we need to translate the research problem in Section 2.1 into the DRL problem described in Section 2.2.

The first step is to determine the technical framework for DRL. Through research on the current popular DRL technologies, we decided to use OpenAI gym and Stable Baselines 3 (SB 3) as the foundation for building the DRL environment quickly.

OpenAI gym is a powerful contains relevant and useful wrappers, utilities and tests that can help users quickly create new RL environments. In this study, the environment is created by inheriting from gym.Env, which is specifically implemented as class inheritance in the python language.

RL environment is a term in the field of artificial intelligence, which is used to indicate the MDP formulation of the certain problem. It is simply called "world" by many articles.

Such an environment can be well built in the OpenAI gym, which links the definition of MDP with the Python coding. According to the structure of OpenAi gym class, I define a continuous state set (which is essentially the observation set in OpenAI gym) and an action set A, where S is constrained by $\left[\underline{S}, \overline{S}\right]$, where $\underline{S}$ is the inf for S and where is $\overline{S}$ the sup for S, Action set $\left[\underline{A}, \overline{A}\right]$ takes a similar fashion. I normalize our state space and action space according to how DICE model is usually constructed, but the action sets and state sets don't necessarily need to be normalized at initialization, because the activation functions in our networks will map the value space to the normalized value space.

We use a build-in method step() to define the $S \to A \to R \to S'$ process. The classic diagram is as follows:

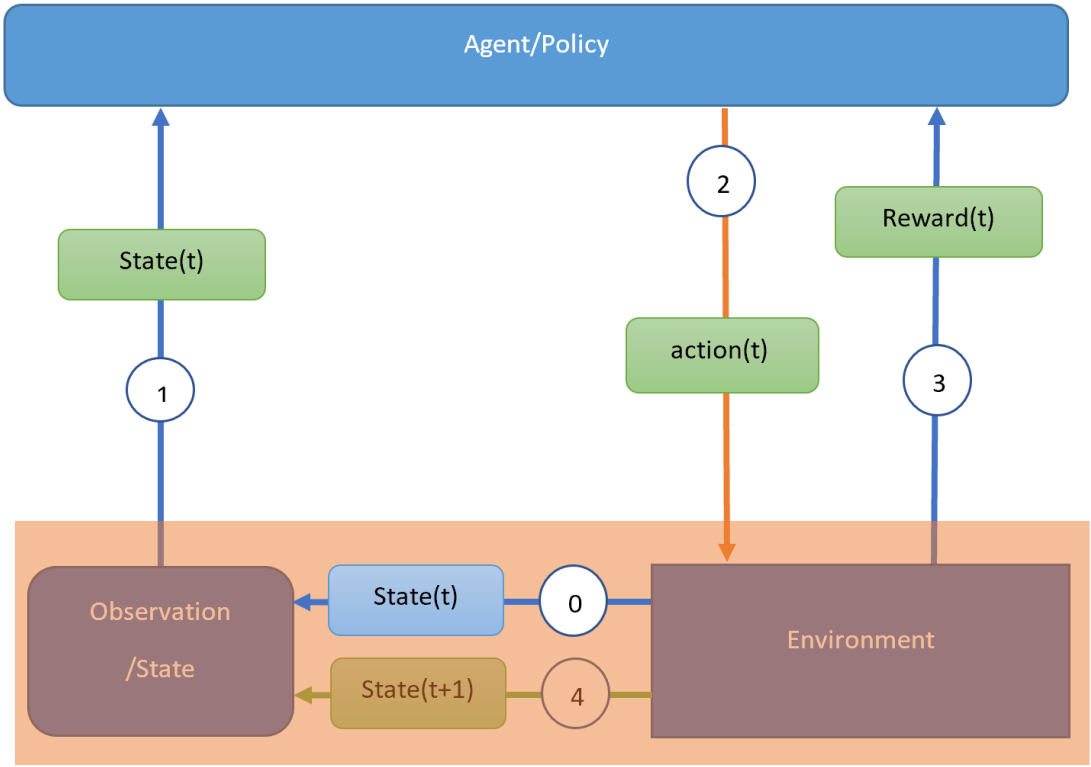


Diagram 1: The Classical "Agent Environment Loop": the agent performs some actions in the environment (the growth rate and saving rate of μ are given in this paper) and observes how the state of the environment changes.

At each step, the agent observes states (observations) and selects the actions $a \sim \pi(\cdot \mid s)$, collects current round of rewards, and finally calculates the value of the motion law of the next state, and passes it to the next step. The equation of motion is defined in section 2.1.

According to the research scope described in Section 2.1, we have identified data and formula resources, including:

1. 18 basic equations
2. Variables (Table 1)

Table 1 Variables

No.	name	notes
1	A	total factor productivity
2	gA	Rate of growth of productivity (per 5-year, log)
3	K	Capital ($trill, 2005$)
4	L	population
5	BP	backstop price
6	σ	emissions intensity industrial

7	gσ	Growth rate of sigma (per year, log)
8	CP	co2 price
9	Y	Output gross of abatement cost
10	θ1	θ1 (t) in Lambda function
11	Λ	Abatement cost function
12	Q	Net output
13	μ	Emissions control rate
14	I	Gross investment ($trill per year, 2005$)
15	C	Consumption
16	cpc	per capita consumption
17	u	Utility of p. c. consumption
18	Eind	Industrial emissions (GTCO2 per year)
19	CCA	Cumulative Emissions to date
20	left_cc	remaining quota of carbon resources (GtC)

Therefore, I can design the environment as:

Reward The reward at each time step consists of two parts. The first part is a positive reward U(t), which is the population-weighted utility of per capita consumption for each period. This is because maximizing the population-weighted utility of per capita consumption is the target in the DICE model.

The second part is the punitive measures to prevent total carbon consumption from exceeding the GtC. If the agent selects small abatement at every step, it will cause the quota remaining for carbon resources (left _ cc) to be insufficient to support the total period. To prevent the quota of carbon resources from running out. When this happens, the environment will give the agent a penalty, that is, it will give the agent a negative reward.

Specifically, we define:

$reward = -10\ln(N-t)$ if（left_cc(t)<0）

and $reward = L(t)c(t)^{(1-\alpha)}/(1-\alpha)/500$

Note I use reward/500. This is to prevent the reward being too large and thus affecting convergence.

Action and Action Space Selecting abatement growth rate and saving rate as action space allows the agent to control the appropriate amount of carbon reduction. Control rate and savings rate ensure the utility maximization under the DICE model. A reasonable saving rate

ensures a reasonable investment and consumption ratio, considering the present and the future interests. As a variable, reasonable investment promotes the change in K (t).

The appropriate μ to ensure the minimization of carbon emission reduction cost. To simplify the realization of the model, we indirectly control μ by the growth rate of μ.

The savings rate is used to choose the ratio of investment and consumption, considering present and the future income. Choosing a reasonable investment rate to drive the capital change.

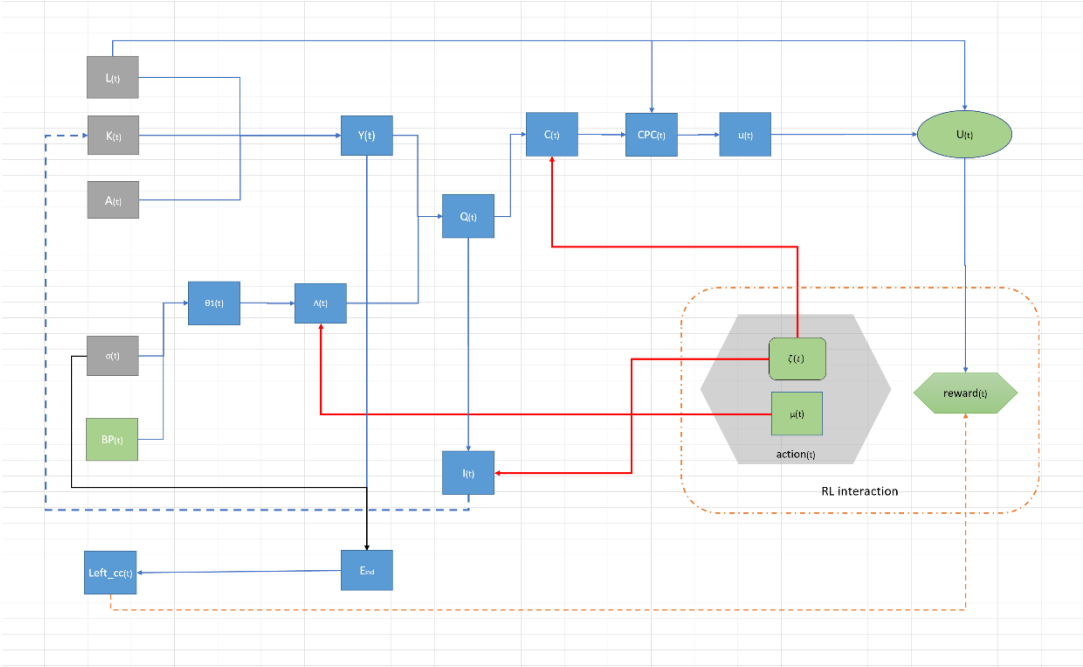
Recall that

$$Y_t = F(A_t, L_t, K_t, E_t) = A_t K_t^{\gamma} L_t^{1-\gamma}\left(1 - \Lambda_t(\mu_t)\right)$$

Thereby driving the change of original output and the changes of net output, CPC (t) and U(t).

States and State space We analyze the 18 formulas and 20 variables in Table 1 to find the dependencies between the variables, as shown in Figure 1.

Figure 1



With Figure 1, we analysis the lineage between variables. The following 8 variables (table 2) describes the motion; thus, we use those to construct our observation space in the RL issue has at most these 8 variables.

Table 2 Observation Space

No.	Name	Notes
1	A(t)	total factor productivity at t
2	K(t)	Capital at t
3	L(t)	Population at t
4	I(t)	Investment at t
5	BP(t)	backstop price at t
6	σ (t)	emissions intensity industrial at t
7	left_cc(t)	remaining quota of carbon resources (GtC) at t
8	μ(t)	emissions control rate at t, As the growth rate of μ is used as the action of the world, μ becomes an important variable

In this way, the observations are passed to the agent. Then, the agent chooses the action based on the policy function by observing the state accordingly.

According to the analysis in section 2.2, we choose SAC as the algorithm for our purpose.

The realization of this algorithm is provided by the stable Baselines 3. Stable Baseline 3 naturally blends in with gym.

2.4 Initialization

I initialized the environment with the parameters in Table 3. Dataset is borrowed from DICE2013.

Table 3 Initial Data

index	name	value	Notes
0	YEARS_PER_PERIOD	5	fixed
1	gL	0.13449	growth rate of world population
2	L0	6838	Initial world population (millions)
3	popasym	10500	infinity world population (millions)
4	A0	3.7976214	Initial level of total factor productivity
5	gA	0.079	Initial growth rate for TFP per 5 years
6	δA	-0.006	annual rate of decline of production tech growth rate
7	θ2	2.8	Exponent of control cost function
8	BP0	344	Backstop price (1000$ per ton CO2)
9	gb	0.025	Decline rate of backstop price (per half-decade)

10	init_sigma	0.489	Initial sigma (MtC per $1000 GDP US $)
11	gσ	-0.01	Growth rate of sigma (per year log)
12	δσ	-0.001	Decline rate of sigma growth rate (per year)
13	μ0	0.039	Initial emissions control rate for base case 2010
14	α	1.45	Elasticity of MU of consumption
15	ρ	0.015	Initial rate of social time preference per year
16	k0	135	Initial capital value (trill 2005 USD)
17	I0	16.361	Initial investment value (trill 2005 USD)
18	γ	0.3	Capital elasticity in production function
19	δ K	0.1	Depreciation rate on capital (per year)
20	cca0	90	init cca
21	inf_cca	680	Maximum cumulative extraction fossil fuels (GtC)
22	inf_cca_year	2060	Year to inf_cca

For the hyperparameters for the RL framework, I use 1000 hidden units, ReLU as the activation function. Learning rate is 0.002. Batch size is set to 128, and the loss function is KL-Divergence and MSE as in the SAC implemented in the stable_baseline3. The hyperparameter of temperature in SAC are automatically turned in stable_baseline3. I also use the multi-vector environment in training to speed up sampling. Note that in the absence of parallel environment, the deep learning framework such as PyTorch or Tensorflow requires adding a dimension to meet the the default tensor format data structure, this encourages one to use multiple vector environments releasing the power of SIMD.

3 Results

Run the model using the settings I described in Section 2. This section presents the results obtained.

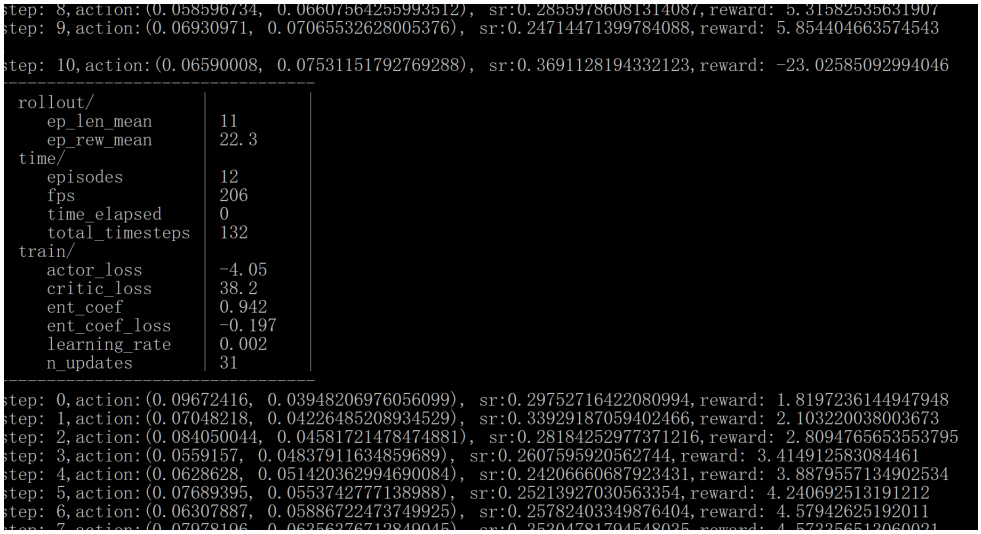
Section 3.1 describes the performance of the RL framework that I implemented.

Section 3.2 and 3.3 show the predicted data obtained from the training. According to our research, predictions of world GDP, per capita consumption, industrial carbon dioxide emissions, emission control rate and carbon price is put forward. This result is also compared with the result of DICE 2013.

3.1 Performance

We used a PC with i7-7700K CPU @ 4.20 GHz, 32.0 GB RAM. Usable results are obtained at after 88,000 steps.

At the beginning of training, the agent will choose as little abatement as possible, because it yields better current reward, resulting in a negative reward by running out of carbon quota (GtC) in the first few steps, as follows



As can be seen, in all 10th steps agent is receiving a negative reward.

This is because at the beginning the Reinforcement Learning agent does not know what will happen if it takes a certain action, it can only explore through trial and error. Therefore, exploration is a trial-and-error method, to find out whether the actions taken have a good return. Exploitation means that we directly take an action which is known to have a good return. Therefore, there is a trade-off on how to gain an understanding of behavior by sacrificing some short-term rewards, to learn better strategies.

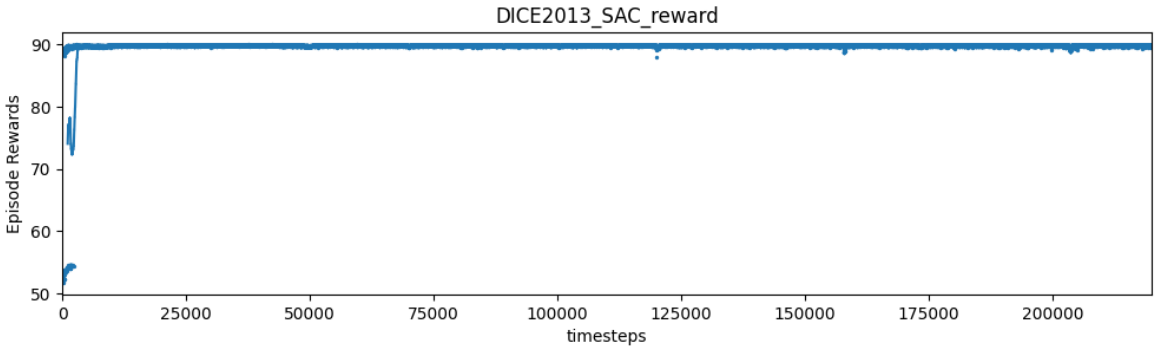
For this training, the learning rate is 0.002, and when it runs to the 152nd episodes, the reward no longer has a negative value, which means that the agent has learned to abate the region. The following is shown.

```
step: 3, action: (0.05834287,  0.04725576476321165),  sr:0.27441368600845337, reward: 3.2808002741038957
step: 4, action: (0.065506235,  0.0503513119837927),  sr:0.2609323561191559, reward: 3.7184712900749233
step: 5, action: (0.0805172,  0.05440545877275997),  sr:0.25098806619644165, reward: 4.2044320679809015
step: 6, action: (0.06834127,  0.05812359692507325),  sr:0.25034528970718384, reward: 4.601369048727755
step: 7, action: (0.08309825,  0.06295356597650398),  sr:0.24717889726161957, reward: 4.9853341028572835
step: 8, action: (0.082045585,  0.0681186281196615),  sr:0.2484985738992691, reward: 5.339545682844185
step: 9, action: (0.084166616,  0.073851942548775),  sr:0.2502848207950592, reward: 5.66991087117326
step: 10, action: (0.0876572,  0.0803255969223442),  sr:0.24759148061275482, reward: 6.000831083738798
-----------------------------
| rollout/
|    ep_len_mean        | 11
|    ep_rew_mean        | 28.1
| time/
|    episodes           | 152
|    fps                | 54
|    time_elapsed       | 30
|    total_timesteps    | 1672
| train/
|    actor_loss         | -10.9
|    critic_loss        | 12.5
|    ent_coef           | 0.0668
|    ent_coef_loss      | -4.56
|    learning_rate      | 0.002
|    n_updates          | 1571
-----------------------------
step: 0, action: (0.053206332,  0.03791542795300484),  sr:0.2438584417104721, reward: 2.032774493056249
step: 1, action: (0.06836562,  0.040507539638985415),  sr:0.2876007556915283, reward: 2.328211022294143
step: 2, action: (0.08364409,  0.04389575601460356),  sr:0.2840173542499542, reward: 2.7172758733815865
step: 3, action: (0.06972596,  0.04695642976429203),  sr:0.2585605978965759, reward: 3.3102442156036105
step: 4, action: (0.06879085,  0.05018660262271711),  sr:0.2453195154666906, reward: 3.809961535833976
step: 5, action: (0.060775515,  0.053236719227759036),  sr:0.27322089672088623, reward: 4.106183413911175
step: 6, action: (0.07211048,  0.05707564468795951),  sr:0.24232542514801025, reward: 4.603932659168453
step: 7, action: (0.082490325,  0.06178383316637327),  sr:0.24495486915111542, reward: 5.011724661786687
step: 8, action: (0.086350724,  0.0671189118980291),  sr:0.25027525424957275, reward: 5.327099710248857
step: 9, action: (0.09321228,  0.07337521848548041),  sr:0.2526470422744751, reward: 5.657732312305767
step: 10, action: (0.09161557,  0.08009753113960529),  sr:0.2488234043121338, reward: 5.9963978614977895
step: 0, action: (0.052771706,  0.037899781420826915),  sr:0.3639054000377655, reward: 1.521082209920853
```

The later training is to reinforce this result and to explore the learning gradually to approach the optimal solution.

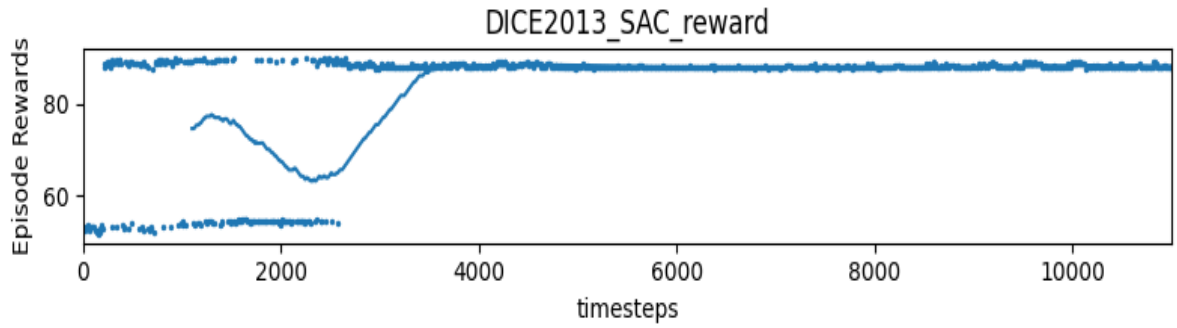
In 88000 training steps, we can get stable results, and in 220000 steps, we can get a usable training model. The reward function during the training steps can is shown in figure 2

Figure 2



In Figure 3, the blue scattered points are the settlement reward obtained within an episode, and at the beginning, the settlement reward value changes a lot, and at the beginning, there are still data below 60.

Figure 3



The reward data for the first 2000 steps can be shown more clearly, and the change curve is more obvious. This is due to the reason that the agent tries a low abatement rate resulting in a negative reward value when there are no more carbon resources available in the last few steps. As the number of learning increases, the following scatter points become less and less, and have converged by 180,000 steps, indicating that the agent has initially learned the carbon reduction control.

Figures 4 and 5 show the change curves of actor loss and critic loss with learning. Over time, actor loss fluctuates in a range, while critic loss tends to zero.

Figure 4

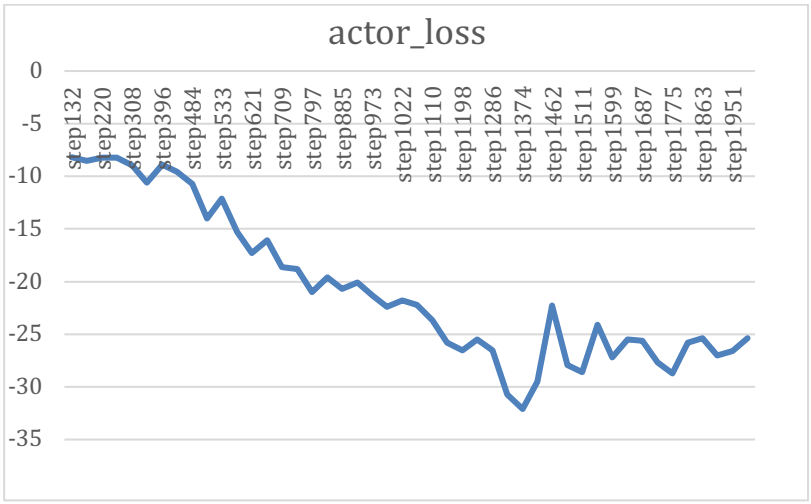
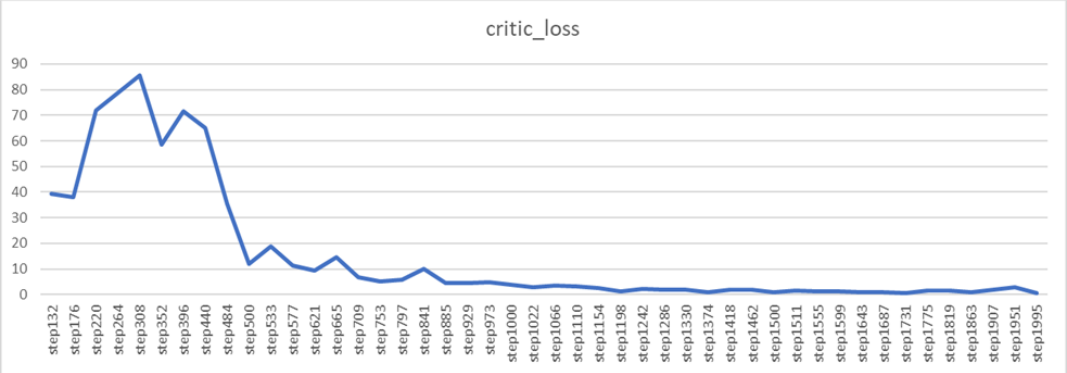


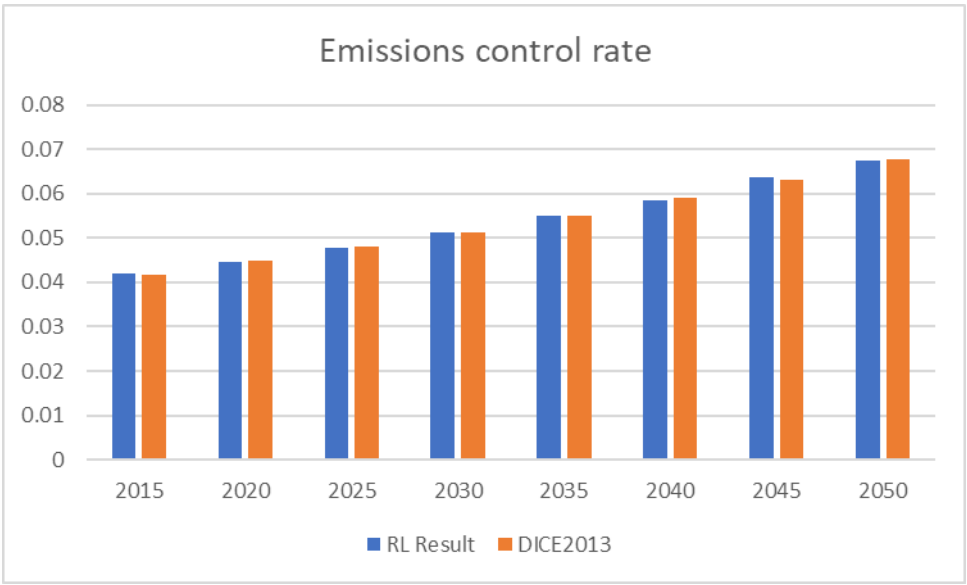
Figure 5



3.2 Base Results

The actions obtained from the training are the basic results, including the carbon reduction rate μ and saving rate. These two are the two key indicators as they are the results learned by the agent.

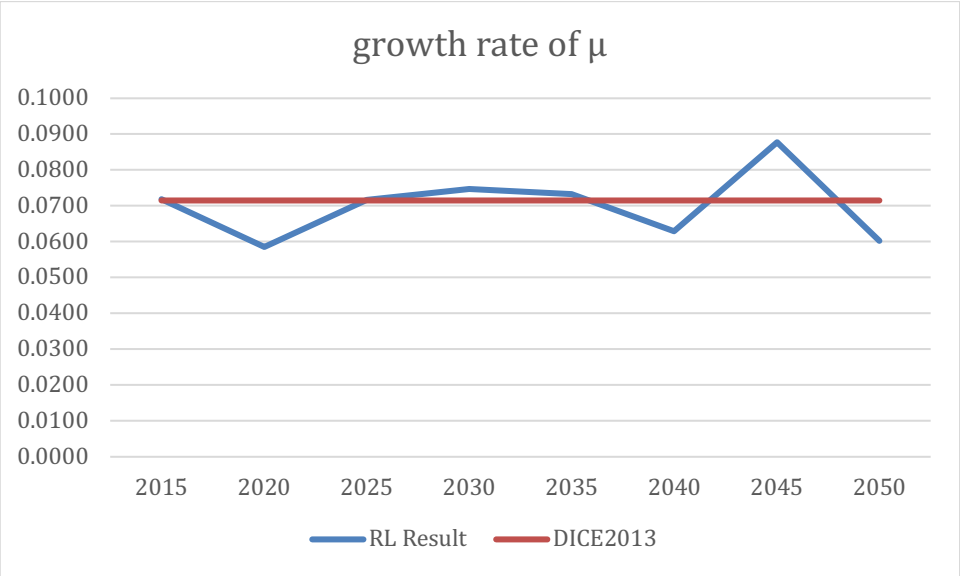
Figure 6



Comparing the data, from 2015 to 2050, with DICE 2013 data, the relative errors are 0.68%, -0.54%, -0.52%, -0.23%, -0.06%, -0.85%, 0.65%, and -0.41%, respectively.

Since we choose the growth rate of μ instead of using μ directly, we show this data, see Figure 7.

Figure 7



In the DICE 2013 document, the growth rate of μ is a constant value equal to 0.0715. The data obtained in this study fluctuate around this constant value, with a mean of 0.0701 and a standard deviation of 0.00948. The result shows that the carbon emission control rate obtained by RL training is acceptable.

The second data is the saving rate, see Figure 8 for comparison,

Figure 8

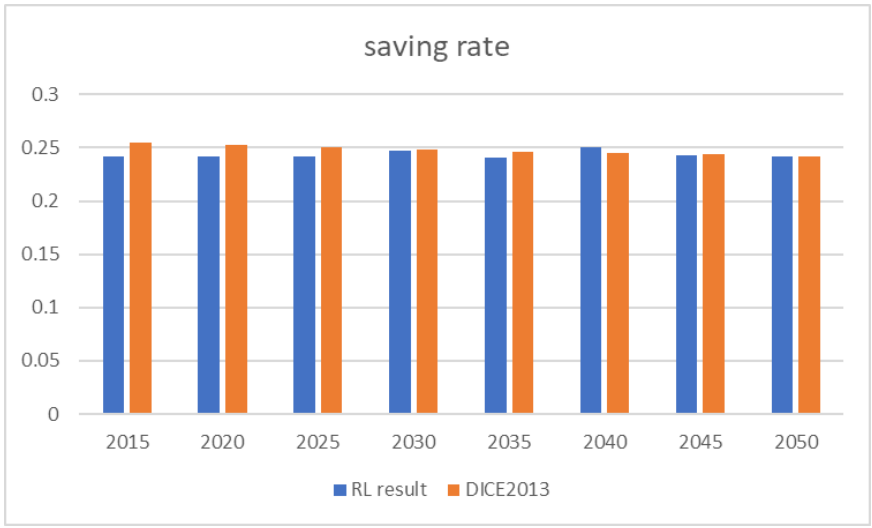


Table 4 Saving Rate

year	2015	2020	2025	2030	2035	2040	2045	2050
RL result	0.241715	0.241484	0.24126	0.246682	0.241081	0.250159	0.242857	0.241291
DICE2013	0.255117	0.252528	0.25021	0.248155	0.246348	0.244767	0.243389	0.242192
R_d %	-5.25	-4.37	-3.58	-0.59	-2.14	2.20	-0.22	-0.37

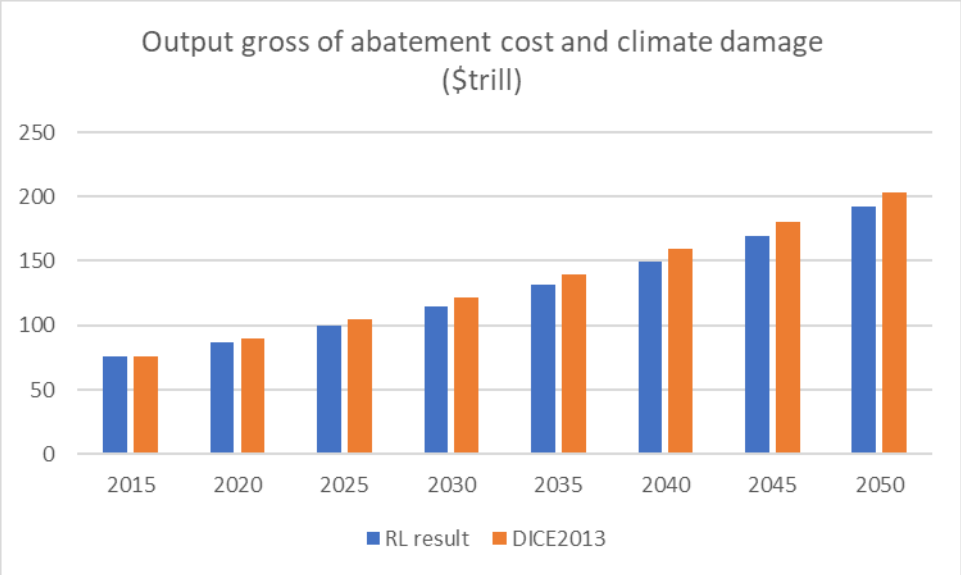
From 2015 to 2050, comparing DICE 2013saving rate data, the savings rates are generally lower, except for only one cycle (2040) being higher. This deviation can be within the confidence interval. It is worth noticing that, if we use saving rate from the RL training, we can get a higher u(t).

Based on the learned policy of the above two actions, other research data can be obtained. Those data are given in Section 3.3.

3.3 Comparison of RL results with DICE2013 data

This section shows that we have obtained relevant forecast data, including the forecast results of world gross product, per capita consumption, industrial CO2 emissions, and carbon price as shown below.

Figure 9



Comparing the data from RL and DICE2013 for output gross of abatement cost and climate damage and abatement cost, the total output is lower than the data of DICE2013 by 3%~6%. The main reason is that the saving rate is lower in my result.

Figure 10

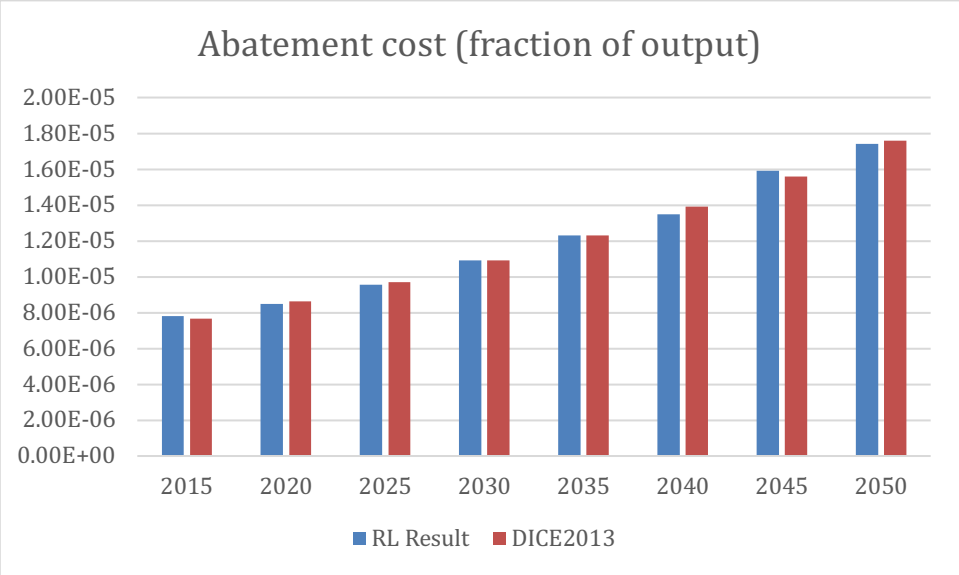
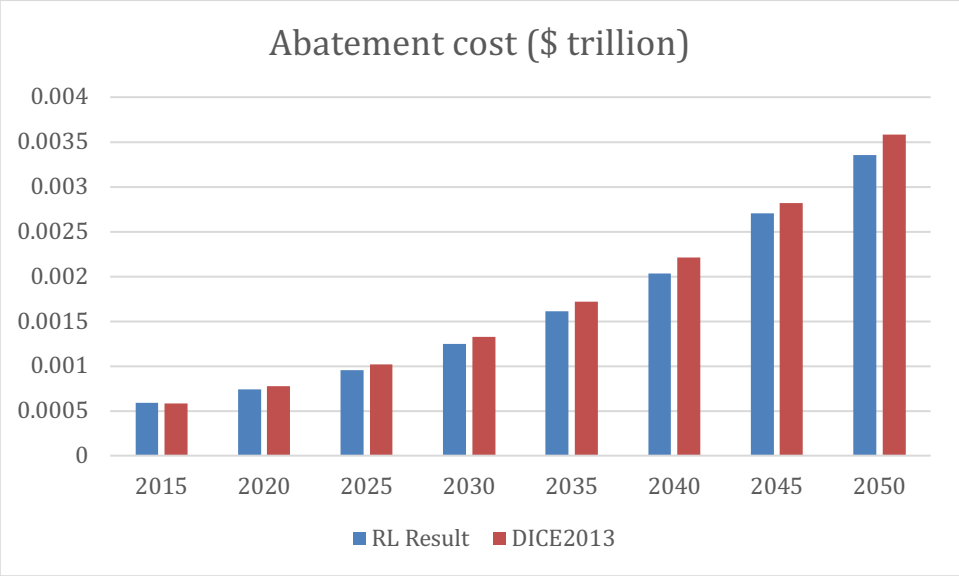


Figure 11



Comparing the RL result and abatement cost faction, the consistency of the predicted data between the two is very good, thanks to the consistency of the prediction of μ-values, and the low abatement cost is related to the low Output gross of abatement cost and climate damage, which is rooted in the saving rate.

Figure 12

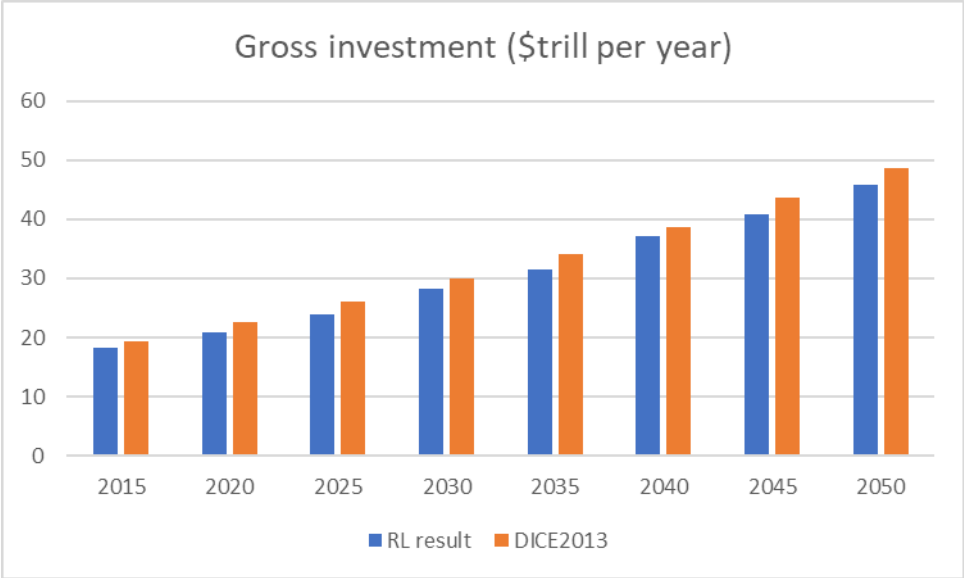


Figure 13

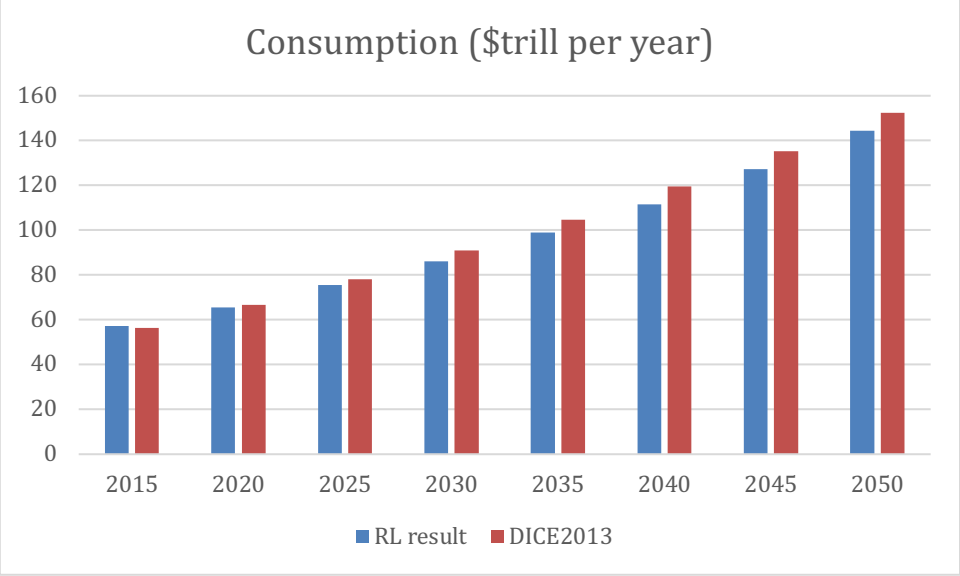
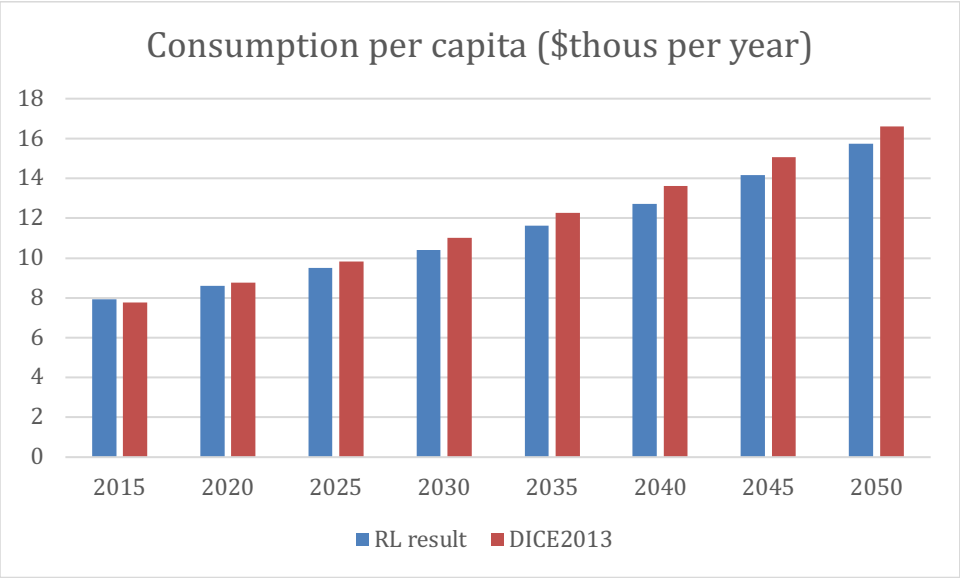


Figure 14



Comparing RL results with the investment and consumption data of DICE 2013, we can see that the forecast data of both are very consistent. The reason for the lower values is the same: our savings rate is lower.

Figure 15

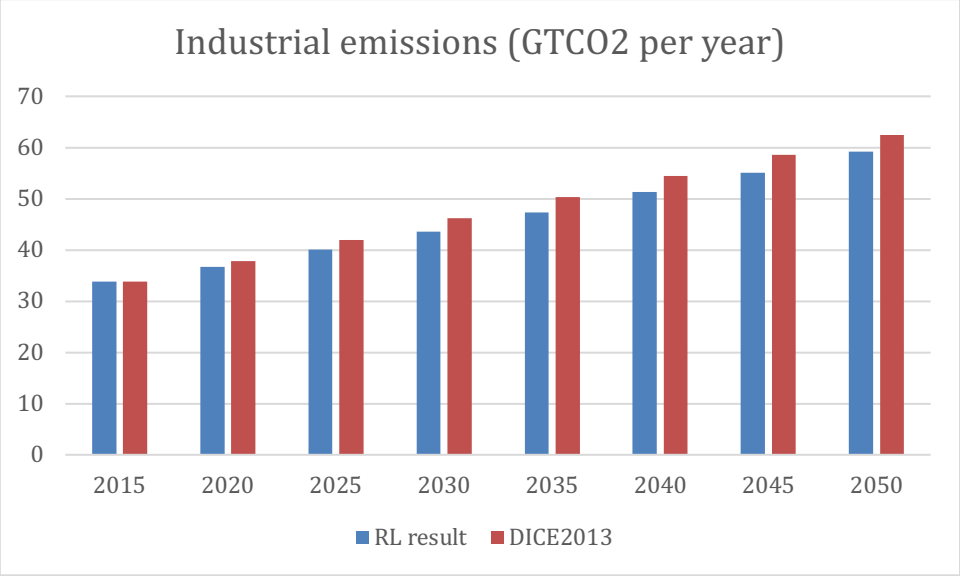
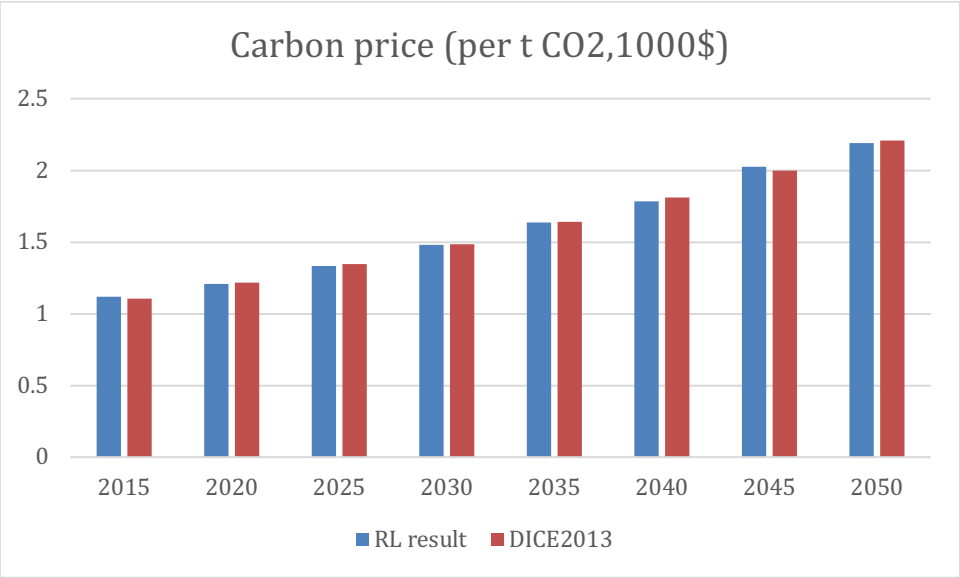


Figure 16



The predictions of industrial carbon emissions and carbon prices is directly learned by RL without the help of model dynamics. They are very consistent, and the savings rate being the cause of low carbon emissions.

4 Discussion

Overall, this study uses reinforcement learning to solve the optimal carbon emission rate and the consumption in a DICE-like model, which is a beneficial exploration and has achieved satisfactory results.

Here we address some topics associated to the RL.

First is the adjustment of parameters. Tuning the parameters is one of the main tasks during the training of Reinforcement Learning. Here I describe how I tuned the parameters of the model. Recall the discount factor means how many next steps of reward one wants to consider for each step you want your agent to perform at minimum. If I want to consider the next t step, then I let the reward of step t be a 0.1 fraction of the Q value of the current step, which is the

$$0{,}1 \approx \gamma^t$$

For the reward, there is total rewards in an episode, generally called the settlement rewards, and the rewards of each step, called the daily rewards. The percentage of rewards in

each step of the settlement reward is determined by the gamma value. In general, if an episode is trained at the end of step t, from the Bellman expansion formula

$$Q_1 = \gamma^0 r_1 + \gamma^1 r_2 + \cdots + \gamma^{t-1} r_t$$

that can be seen more clearly, which is the theoretical basis for adjusting the gamma parameter. It is recommended to try between 0.95 and 0.98.

The learning rate can be tried from a relatively large value, such as 0.02, and then 0.02, 0.002, and 0.0002 can be chosen to follow the principle of large to small, which can avoid choosing a very small value (such as 0.0002) to enter the local optimum and not get the correct result.

The batch size represents the size of the data used for one calculation. Generally, we try from 64, 128, 256 (Nth power of 2). It is recommended to use a larger batch size, as it is easier to obtain a monotonically increasing learning curve with a larger batch size, but leads to more training sessions, which is slower but more stable, but allows a larger learning rate.

Usually, the combination of a large batch size and a larger learning rate is used to save training time and get better results. batch size is chosen to match the memory of the computer used and the storage size of the GPU.

It might be possible to use RL method can as a method for economics because the RL method enables model-free learning. Economic dynamics are designed as state transition processes in time series, and involve optimization, such as cost minimization or utility maximization. For any economic problem, if the state process is described as an MDP, and the definition of the value function associated with the optimal strategy satisfies Bellman's principle, this problem is exactly consistent with the Reinforcement learning principle, and Reinforcement learning can be attempted to solve this problem. This is also the theoretical support for transforming an economic problem into a RL problem as described in Section 2.3.

The code implementation also shows the applicability of RL. I have shown that the RL environment is OpenAI Gym is a good description of the RL environments. And many toolkits like stable_baseline 3 can be a reliable framework of reinforcement learning algorithms. These algorithms will allow the research groups and industry to easily set up projects based on implementation, either to copy, improve the current algorithms or put new ideas into RL framework without getting caught up in the implementation details of the algorithms.

However, there also a trade-off between generic. One I immediately identify is the low sample efficiency. This is partly due to the current model-free algorithm. Intuitively a long convergence time. It is worth noticing that a model-free RL model can be much more arduous to train compared to conventional methods (Biemann et al., 2021). Even when it was compared to model-based RL models, model-free RL turns to have less sample efficiency, which thus results in more computational time. On the other hand, let's keep in mind that, as stated by (Marlir et al, 2019), the advancement of computational power over the years will soon keep up and overcome the long-time training issue.

I also suggest a way to partially address the issue of sample inefficiency. One can apply a prior knowledge to increase the speed of learning. Here I used a parametric form μ to address such issue. It is assumed in DICE2013 that μ increases over time. For a variable in a time series, it can always be written as $x(t+1) = x\big(1 + r(t)\big)$. As in this project, we define $\mu(t+1) = \mu(t)\left(1 + g_\mu(t)\right)$ where $g_\mu(t)$ is the growth rate of $\mu(t)$. Following this equation, let $g_\mu(t)$ be the action of the agent instead of μ. When the agent learns an appropriate $g_\mu(t)$ strategy from the environment, we can also easily obtain the value of μ. I was able to observe a faster convergence speed after applied such a form of μ. As there are many assumptions in the economic problems, I encourage applying a priori knowledge into the design of the RL environment and try with different parameters.

There is also a disadvantage that one should pay attention to. The convergence theory has not been fully developed for this kind of RL. Despite the convergence of discrete action spaces though SAC is proved, the author mentioned that the convergence neural network approximated value functions only work in practice (Haarnoja et al., 2018a).

Yet the main goal of this project is to demonstrate the potential such a framework might work and pave a way for researchers to apply economic dynamics were predicting ahead is not feasible. For example, in IAMs where complex dynamics and uncertainties are involved. The ability of learning without fully observing the MDP can come to real use.

5 Conclusion

In this project I have made a demonstration of a model-free deep Reinforcement Learning framework and applied it to a DICE model with 8 variables constituting states in observation

space, 2 actions in action space. The reward is U(t), the population-weighted utility of per capita consumption for each step. The agent was able to interact with the environment and obtain optimal policies of actions that maximize the lifetime reward without making any prediction of the utility or dynamics in the model. Instead, the agent is self-learning to find the optimal policy through repeated training from the observed states, the actions chosen, and the rewards obtained. And after around 90K timesteps, the results we get are the optimal policies that demonstrate to be a good match with the DICE2013 reference results. This shows to what extent of potential such a framework possesses. As a matter of fact, it could be extended to more complex and unpredictable environments, where further works should and could be done. It's also worth mentioning that deep Reinforcement Learning can be computationally expensive with the context in this project.

Reference

Baker, E., & Solak, S. (2011). Climate change and optimal energy technology R&D policy. *European Journal of Operational Research*, *213*(2), 442-454.

Bellman, R., & Kalaba, R. (1957). Dynamic programming and statistical communication theory. *Proceedings of the National Academy of Sciences of the United States of America*, *43*(8), 749.

Bertsekas, D. (2012). *Dynamic programming and optimal control: Volume I* (Vol. 1). Athena scientific.

Biemann, M., Scheller, F., Liu, X., & Huang, L. (2021). Experimental evaluation of model-free reinforcement learning algorithms for continuous HVAC control. *Applied Energy*, *298*, 117164. https://doi.org/10.1016/j.apenergy.2021.117164

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. *ArXiv Preprint ArXiv:1606.01540*.

Brooke, Anthony, David Kendrick, Alexander Meeraus, and Ramesh Raman. (2005). *GAMS: A Users Guide, GAMS Development Corporation*, Washington, D.C.

Clarke, L., J. Edmonds and V. Krey, et al. (2009) . International Climate Policy Architectures: Overview of the EMF 22 International Scenarios, *Energy Economics, 31(2), S64-S81.*

Crost, B., & Traeger, C. P. (2013). Optimal climate policy: uncertainty versus Monte Carlo. *Economics Letters*, *120*(3), 552-558.

Duffy, J., & McNelis, P. D. (2001). Approximating and simulating the stochastic growth model: Parameterized expectations, neural networks, and the genetic algorithm. *Journal of Economic Dynamics and Control*, *25*(9), 1273-1303.

Fernández-Villaverde, J. (2010). Fiscal policy in a model with financial frictions. *American Economic Review*, *100*(2), 35-40.

Fujimoto, van Hoof, H., & Meger, D. (2018). *Addressing Function Approximation Error in Actor-Critic Methods*.

Haarnoja, Zhou, A., Abbeel, P., & Levine, S. (2018a). *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor.*

Haarnoja, Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., & Levine, S. (2018b). *Soft Actor-Critic Algorithms and Applications.*

Haarnoja, Ha, S., Zhou, A., Tan, J., Tucker, G., & Levine, S. (2018c). *Learning to Walk via Deep Reinforcement Learning*.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., & Kavukcuoglu, K. (2016). Reinforcement Learning with unsupervised auxiliary tasks.

Jirniy, A. and V. Lepetyuk, (2012). A Reinforcement Learning approach to solving incomplete market models with aggregate uncertainty.

Krusell, P., & Smith, Jr, A. A. (1998). Income and wealth heterogeneity in the macroeconomy. *Journal of political Economy*, *106*(5), 867-896.

Lemoine, D., & Traeger, C. (2014). Watch your step: optimal policy in a tipping climate. *American Economic Journal: Economic Policy*, *6*(1), 137-66.

Maliar Lilia, Serguei Maliar and pablo Winant (2021). "Deep Learning for Solving Dynamic Economic Models". *Journal of Monetary Economics 122*. – Version 2019 CEPR working paper 14024. – Version 2018.

Manne, A. S., & Richels, R. G. (1994). The costs of stabilizing global CO2 emissions: a probabilistic analysis based on expert judgments. *The Energy Journal*, *15*(1).

Melillo J, Prinn R, Jacoby H (2009) Analysis of climate policy targets under uncertainty. Tech. rep, MIT JPSPGC, Report No 180.

Mnih, Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning*.

Nordhaus, W. D. (1994). *Managing the global commons: the economics of climate change* (Vol. 31). Cambridge, MA: MIT press.

Nordhaus, W. D. (1975). The Political Business Cycle. *The Review of Economic Studies*, *42*(2), 169–190. https://doi.org/10.2307/2296528

Nordhaus, W. D. (2007). A review of the Stern review on the economics of climate change. *Journal of economic literature*, *45*(3), 686-702.

Nordhaus, W. D. (2010). Economic aspects of global warming in a post-Copenhagen environment. *Proceedings of the National Academy of Sciences*, *107*(26), 11721-11726.

Nordhaus, W., & Sztorc, P. (2013). *DICE 2013R: Introduction and User's Manual*.

Raffin, Hi, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research, 22,* 1–8.

Reilly, J. M., Edmonds, J. A., Gardner, R. H., & Brenkerf, A. L. (1987). Uncertainty analysis of the IEA/ORAU CO2 emissions model. *The Energy Journal*, *8*(3).

Renner, P., & Scheidegger, S. (2018). Machine learning for dynamic incentive problems.

Scheidegger, S., & Bilionis, I. (2019). Machine learning for high-dimensional dynamic stochastic economies. *Journal of Computational Science*, *33*, 68-82.

Scott, H. L., Jakobsson, E., Subramaniam, S., Gould, H., & Tobochnik, J. (1998). Simulations of lipid membranes with atomic resolution. *Computers in Physics*, *12*(4), 328-334.

Stern. (2007). *The Economics of Climate Change*. Cambridge University Press. https://doi.org/10.1017/CBO9780511817434

Stern, N. (2006) .Stern Review: The Economics of Climate Change. *World Economics*，98 (2) , 1-10.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic Policy Gradient Algorithms. *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, I–387–I–395. Beijing, China: JMLR.org.

Sutton, L. K. R. (1996). Model-based Reinforcement Learning with an approximate, learned model. In *Proceedings of the ninth Yale workshop on adaptive and learning systems* (pp. 101-105).

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, *3*(1), 9-44.

Sutton, & Barto, A. G. (1998). *Reinforcement Learning*. MIT Press.

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, *38*(3), 58-68.

Webster M, Paltsev S, Parsons J, Reilly J, Jacoby H (2008) Uncertainty in greenhouse emissions and costs of atmospheric stabilization. Tech rep, MIT JPSPGC, Report No. 165123362

Webster, M., Sokolov, A. P., Reilly, J. M., Forest, C. E., Paltsev, S., Schlosser, A., ... & Jacoby, H. D. (2012). Analysis of climate policy targets under uncertainty. *Climatic change*, *112*(3), 569-583.