Scientific Reports

https://doi.org/10.1038/s41598-026-42086-x

Article in Press

Visualising backward information propagation in deep reinforcement learning from a variational data assimilation perspective

Received: 15 December 2025

Accepted: 24 February 2026

Published online: 02 March 2026

Cite this article as: Wang K. Visualising backward information propagation in deep reinforcement learning from a variational data assimilation perspective. *Sci Rep* (2026). https://doi.org/10.1038/s41598-026-42086-x

Kuo-Ying Wang

Visualising backward information propagation in deep reinforcement learning from a variational data assimilation perspective

Kuo-Ying Wang*
Department of Atmospheric Sciences,
National Central University, Chung-Li, Taiwan
*kuoying@mail.atm.ncu.edu.tw

February 24, 2026

Abstract

It has long been recognised that variational data assimilation, including four-dimensional variational methods (4D-Var), is grounded in Bayesian inference and gradient-based optimisation. Deep reinforcement learning (RL) employs related mathematical machinery, iteratively minimising a scalar objective function through backward propagation of information. In this study, we do not propose new algorithms or theoretical connections, but instead provide a transparent and visual illustration of these well-established relationships. Using a compact neural network trained to play the classic Snake game, we track the evolution of all network weights at every training iteration. Short-horizon temporal-difference updates yield frequent local gradient steps on a linearised error signal, closely resembling the inner-loop minimisation of incremental 4D-Var, while experience replay repeatedly recomputes gradients under updated parameters, analogous to outer-loop relinearisation about an evolving reference trajectory. This minimal and fully observable system serves as a controlled laboratory for visualising backward information propagation in optimisation processes familiar to both reinforcement learning and variational data assimilation. The resulting comparison offers an interpretable, pedagogical perspective on reinforcement learning using concepts long established in the data-assimilation literature, without claiming new algorithmic insights or applications.

1 Introduction

Weather research and operational numerical weather prediction have long relied on variational and ensemble-based data assimilation as the core framework for optimally combining observations from diverse platforms with dynamical models. As a result, many atmospheric scientists are deeply familiar with adjoint sensitivity methods and gradient-based optimisation, but less so with the training dynamics of modern neural networks. The mathematical connections between variational data assimilation and gradient-based machine learning have been discussed previously, including interpretations of learning as statistical data assimilation and variational optimisation frameworks for Earth-system models [40, 41]. However, how these connections manifest specifically within reinforcement learning—where optimisation is driven by temporally propagated reward signals rather than direct observational misfits—has received comparatively little attention.

The motivation for this study is rooted in earlier work on adjoint-based variational data assimilation [20], which examined how observational information propagates backward through an adjoint model during four-dimensional variational data assimilation (4D-Var) to shape the optimisation of initial conditions. The present study revisits this same mathematical mechanism from a different perspective: reinforcement learning. By visualising the backward propagation of reward-driven gradients in a minimal neural-network system, we aim to expose the shared optimisation structure underlying reinforcement learning and 4D-Var, while deliberately avoiding the additional complexity of application-specific atmospheric models. In contrast to prior theoretical treatments of the ML–DA connection, the focus here is on making backward information propagation directly observable at the level of individual optimisation steps.

Machine learning has seen rapid adoption across Earth-system science in recent years [31, 39, 34]. Neural networks and other data-driven approaches now contribute to applications ranging from multi-year climate prediction [25] to high-resolution precipitation nowcasting [26]. A growing class of hybrid Earth-system learning approaches—including physics-informed machine learning [33], reinforcement-learning formulations for data-assimilation decisions [37], and online learning strategies in Earth-system digital twins [35, 32]—demonstrate

Table 1: Structural correspondence between Q-learning and 4D-Var data assimilation.

Concept	**Q-learning**	**4D-Var**
State	$\mathrm{s_t}$	$\mathbf{x}(\mathrm{t})$
Control variable	$\mathrm{a_t}$	Initial condition $\mathbf{x}_0$
State evolution	$\mathrm{s_{t+1}} = \mathcal{T}(\mathrm{s_t}, \mathrm{a_t})$	$\mathbf{x}_{\mathrm{i+1}} = \mathcal{M}(\mathbf{x}_\mathrm{i})$
Feedback signal	$\mathrm{r_{t+1}}$	$-(\mathcal{H}_\mathrm{i}(\mathbf{x}_\mathrm{i}) - \mathbf{y}_\mathrm{i})^{\mathrm{T}}\mathbf{R}^{-1}(\mathcal{H}_\mathrm{i}(\mathbf{x}_\mathrm{i}) - \mathbf{y}_\mathrm{i})$
Value analogue	$\mathrm{Q}(\mathrm{s}, \mathrm{a}; \boldsymbol{\theta})$	$\nabla_{\mathbf{x}_0}\mathrm{J}$
Backward propagation	Bellman backup + backpropagation	Adjoint model integration
Objective	$\min(\mathrm{y} - \mathrm{Q})^2$	$\min \mathrm{J}(\mathbf{x}_0)$

that, across a wide range of contexts, learning is governed by repeated gradient-based minimisation of a scalar objective, with information propagated backward through a model (neural network or adjoint) to update parameters or control variables.

Reinforcement learning (RL) represents a complementary optimisation paradigm and has powered major advances in artificial intelligence, from Atari game-playing agents [1] to strategic decision-making systems such as AlphaGo [2]. Despite its success, RL is often perceived as a "black box," in part because its learning dynamics are rarely visualised or examined at the parameter level. This lack of transparency poses a barrier to broader adoption in the geosciences, where understanding the origin and structure of model updates is often as important as predictive performance itself.

Although today's large-scale AI systems, including large language models, operate at extreme computational scale, their training ultimately relies on the same mathematical backbone as variational data assimilation: gradient-based minimisation of scalar objective functions and backward propagation of information. While this shared structure is well recognised in theory, it has not been demonstrated visually in a manner that is readily accessible to atmospheric scientists.

Here, we provide a transparent, minimal laboratory for illustrating this connection. We train a compact deep Q-network to play the classic Snake game and record the evolution of all 3,584 network parameters at every training iteration. By animating both short-memory (within-episode) updates and long-memory (experience-replay) updates, we demonstrate a clear algorithmic correspondence between reinforcement-learning temporal-difference updates and adjoint-based backward information propagation in four-dimensional variational (4D-Var) data assimilation, with both arising from gradient-based minimisation of a scalar objective function. This visualisation offers an intuitive bridge between two optimisation traditions—artificial intelligence and atmospheric data assimilation—and clarifies how backward propagation of information drives learning and adjustment in both systems.

The purpose of this study is not to propose a new atmospheric modelling framework, nor to demonstrate the application of reinforcement learning within numerical weather prediction systems. Instead, the objective is to make the minimisation process itself directly observable and interpretable. Reinforcement learning and four-dimensional variational data assimilation both operate via steepest-descent optimisation of a scalar cost function, differing primarily in the choice of control variables: network parameters in reinforcement learning and initial conditions (or other control parameters) in 4D-Var. By selecting a minimal and fully transparent system—the Snake game—we isolate and visualise the backward propagation of information that drives learning in both paradigms, allowing the underlying optimisation mechanism to be examined without the confounding complexity of high-dimensional atmospheric models, while remaining fully representative of the same mathematical structure.

2 Methods

2.1 The Shared Mathematical Core

Here we focus on reinforcement learning as an optimisation process, and use four-dimensional variational data assimilation as a mathematical reference framework, emphasising the shared structure of iterative local gradient updates and periodic relinearisation that enable backward propagation of information in both systems [42, 43, 44, 45].

Reinforcement learning (RL) and four-dimensional variational data assimilation (4D-Var) share a common mathematical structure: both seek to minimise a scalar objective function defined over a trajectory by propagating information backward in time. In RL, this backward propagation is achieved through the Bellman backup and neural-network backpropagation; in 4D-Var, it is achieved through the adjoint of the forecast model. Table 1 summarises the correspondence between Q-learning and 4D-Var using notation

consistent with atmospheric data-assimilation conventions. We note that the RL action a_t is an external control applied to the environment and has no direct analogue in strong-constraint 4D-Var, where the model evolution is autonomous. The corresponding optimisation control variable in 4D-Var is the initial condition x_0, as indicated in Table 1.

2.2 Q-learning formulation

The Bellman optimality equation for Q-learning is

$$Q^*(s,a) = \mathbb{E}\Big[r + \gamma \max_{a'} Q^*(s',a')\Big], \tag{1}$$

where s denotes the state, a the action, r the scalar reward, and $\gamma \in (0,1)$ the discount factor.

In Eq. 1, $Q^*(s,a)$ denotes the optimal action–value function, defined as the maximum expected cumulative discounted reward achievable by following an optimal policy from state s after taking action a. The expectation $\mathbb{E}[\cdot]$ is taken with respect to the stochastic transition dynamics of the environment and the reward distribution, conditioned on (s, a), i.e. $p(s', r \mid s, a)$. In practice, Q^* is not known and is approximated by a parameterised function $Q(s,a;\theta)$ learned from sampled transitions.

The temporal-difference (TD) target is

$$y = r + \gamma \max_{a'} Q(s',a'), \tag{2}$$

and training proceeds by minimising the squared TD error

$$J = \frac{1}{2}\left[y - Q(s,a;\boldsymbol{\theta})\right]^2 \tag{3}$$

with respect to the network parameters $\boldsymbol{\theta}$ via gradient descent.

Eq 1 defines the Bellman optimality condition for the unknown optimal action–value function Q^* and involves an expectation over the environment transition and reward distributions. Eq. 2 differs in that it defines a sample-based temporal-difference (TD) target computed from a single realised transition (s, a, r, s') using the current approximation $Q(s,a;\theta)$. In practice, learning proceeds by minimising the discrepancy between this sample-based target and the predicted value, providing a stochastic approximation to the expectation in Eq. 1. This distinction is analogous to replacing an expectation over all possible observation realisations with an instantaneous observation misfit in variational data assimilation.

The gradient $\partial J/\partial \boldsymbol{\theta}$ is computed by backpropagation, corresponding to the reverse application of the chain rule through the network.

From a variational data-assimilation perspective, the Q-learning loss can be interpreted as a local quadratic approximation to a trajectory-dependent cost function, where the network parameters θ play the role of control variables analogous to the initial condition x_0 in 4D-Var. The temporal-difference target y acts as a forcing term derived from future information, analogous to observation misfits driving the adjoint model backward in time.

2.3 4D-Var formulation

The 4D-Var cost function over an assimilation window $[t_0, t_N]$ is

$$J(\mathbf{x}_0) = \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}_b)^T \mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}_b) + \frac{1}{2}\sum_{i=0}^{N}(\mathcal{H}_i(\mathbf{x}_i) - \mathbf{y}_i)^T \mathbf{R}^{-1}(\mathcal{H}_i(\mathbf{x}_i) - \mathbf{y}_i), \tag{4}$$

where $\mathbf{x}_0$ is the initial state, $\mathbf{x}_b$ the background state, $\mathbf{B}$ and $\mathbf{R}$ the background and observation error covariance matrices, $\mathcal{M}$ the forecast model, and $\mathcal{H}_i$ the observation operator. Minimisation of J is performed by integrating the adjoint model backward in time, propagating sensitivities from observation times to the start of the window.

Conversely, the 4D-Var cost function J(x) may be viewed through a reinforcement-learning lens as defining a value function over trajectories, where the adjoint gradient plays a role analogous to a value-gradient signal. Backward integration of the adjoint equations thus corresponds to propagating future error information to earlier control variables, in the same algorithmic sense that temporal-difference learning propagates reward information backward through a network.

2.4 Interpretation of the analogy

In Q-learning, future rewards influence earlier decisions through the Bellman backup and neural-network backpropagation. In 4D-Var, observation misfits influence earlier states through adjoint sensitivity propagation. In both frameworks, optimisation is achieved by iteratively reducing a scalar objective function defined over an entire trajectory rather than at a single instant. Figure 1 illustrates this correspondence.

In incremental 4D-Var, the outer loop updates the nonlinear trajectory over a fixed assimilation window and redefines a locally linear(ised) optimisation problem, while the inner loop approximately minimises that quadratic subproblem to obtain an increment (often interpreted as an approximate Gauss–Newton method [42, 43, 44, 45]. In DQN training, the network parameters define the current 'reference' used to compute temporal-difference targets; repeated gradient steps minimise the squared Bellman error for recent (or replayed) transitions, and replay repeatedly recomputes targets under the evolving parameters. The correspondence we emphasise is therefore the shared iterative pattern of (i) local gradient-driven updates and (ii) periodic recomputation of the linearised error signal about an updated reference.

Among data-assimilation approaches, incremental 4D-Var most closely mirrors the phased learning cycle of reinforcement learning. In incremental 4D-Var, a nonlinear model trajectory is first integrated forward in time, observation misfits are evaluated along the trajectory, and the model is then linearised about this reference state. Sensitivities are propagated backward using the adjoint model to compute gradients of the cost function with respect to the control variables, followed by an incremental update of the initial condition. This outer–inner loop structure directly parallels the forward evaluation, reward-based feedback, and gradient-driven parameter updates that characterise reinforcement learning.

2.5 The Snake Laboratory

To make this correspondence explicit and observable, we adopt the game of *Snake* as a minimal reinforcement-learning laboratory. Figure 2 shows the deep Q-network (DQN) architecture used in this study.

We use the open-source implementation of [24] with minor modifications to record all network parameters at each training iteration. The network contains 3,584 trainable parameters, providing a compact yet fully interpretable system in which the backward propagation of reward information can be directly visualised.

3 Results

3.1 Game setting

In the Snake game, an agent controls a moving snake on a two-dimensional grid with the objective of repeatedly reaching randomly placed food items while avoiding collisions with walls or its own body. At each time step, the agent selects one of three actions (move straight, turn left, or turn right) based on the current state of its local environment. A positive reward is received when food is consumed, while negative rewards terminate the episode when a collision occurs. Learning proceeds by adjusting the network parameters to maximise cumulative reward over successive game episodes.

From a variational data assimilation perspective, this process may be viewed as an optimisation over trajectories: the reinforcement-learning agent explores sequences of actions, while the optimisation updates the control variables (network parameters) to favour trajectories that lead to higher reward. This is conceptually analogous to four-dimensional variational data assimilation, in which different initial conditions are iteratively adjusted to produce model trajectories that best fit available observations over a time window.

3.1.1 Network architecture and control variables

Figure 3 provides a detailed, step-by-step visualisation of the Snake reinforcement-learning experiment, combining the game environment with the internal neural-network state and action-selection process. Because the figure displays multiple layers of information simultaneously, we first describe its components and notation before discussing the results. Figure 3 shows, pedagogically, illustration of the learning process; the main results and quantitative comparisons are presented starting from Figure 4.

Figure 3(a–c) show trained trajectories, while Figure 3(d–f) show untrained trajectories. Panels (a)–(c) show successive time steps from the same game episode at increasing training iterations. Panel (b) follows directly from panel (a): after the action selected in (a), the environment updates deterministically, causing the snake head to move and the local configuration around the head to change. As a result, the binary state vector—and the corresponding upper-left inset—are updated according to the new relative positions of walls, food, and the snake body. The downward movement observed in (b) reflects either an exploratory action or

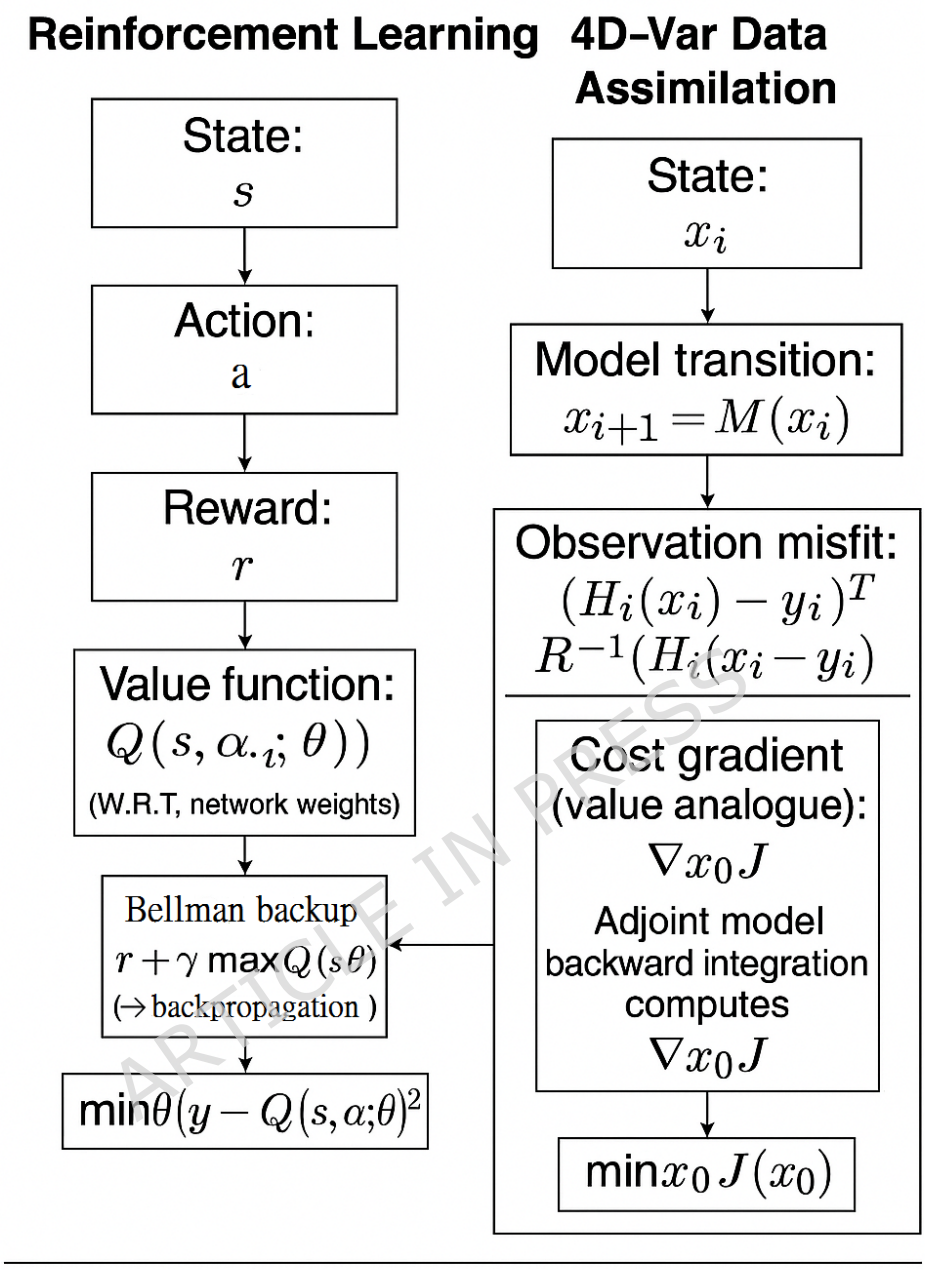

Figure 1: Side-by-side conceptual analogy between Reinforcement Learning (left) and 4D-Var data assimilation (right). In RL, the agent occupies a state s, takes an action a, receives a reward r, and updates the value function $Q(s, a; \theta)$ with respect to the network weights θ. Here and throughout this paper, the term "weight" refers to a collection of trainable parameters. Accordingly, $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ denote weight matrices (or equivalently, flattened parameter vectors) associated with different layers of the neural network, rather than scalar quantities. The Bellman backup, $y = r + \gamma \max_{a'} Q(s', a'; \theta)$, propagates information backward and provides the signal used for neural-network backpropagation. In 4D-Var, the model evolves the atmospheric state $\mathbf{x}_i$ forward in time via $\mathbf{x}_{i+1} = \mathcal{M}(\mathbf{x}_i)$, and the observation misfit $(\mathcal{H}_i(\mathbf{x}_i) - \mathbf{y}_i)^T \mathbf{R}^{-1} (\mathcal{H}_i(\mathbf{x}_i) - \mathbf{y}_i)$ contributes to the overall cost function. Backward propagation occurs through the adjoint model, which computes the cost gradient with respect to the initial state, $\nabla_{\mathbf{x}_0} J$, directly analogous to the value function in RL. Both systems minimise an overall objective: $(y - Q(s, a; \theta))^2$ in RL and $J(\mathbf{x}_0)$ in 4D-Var. Network parameters θ play the role of control variables, analogous to the initial condition x_0 in 4D-Var.

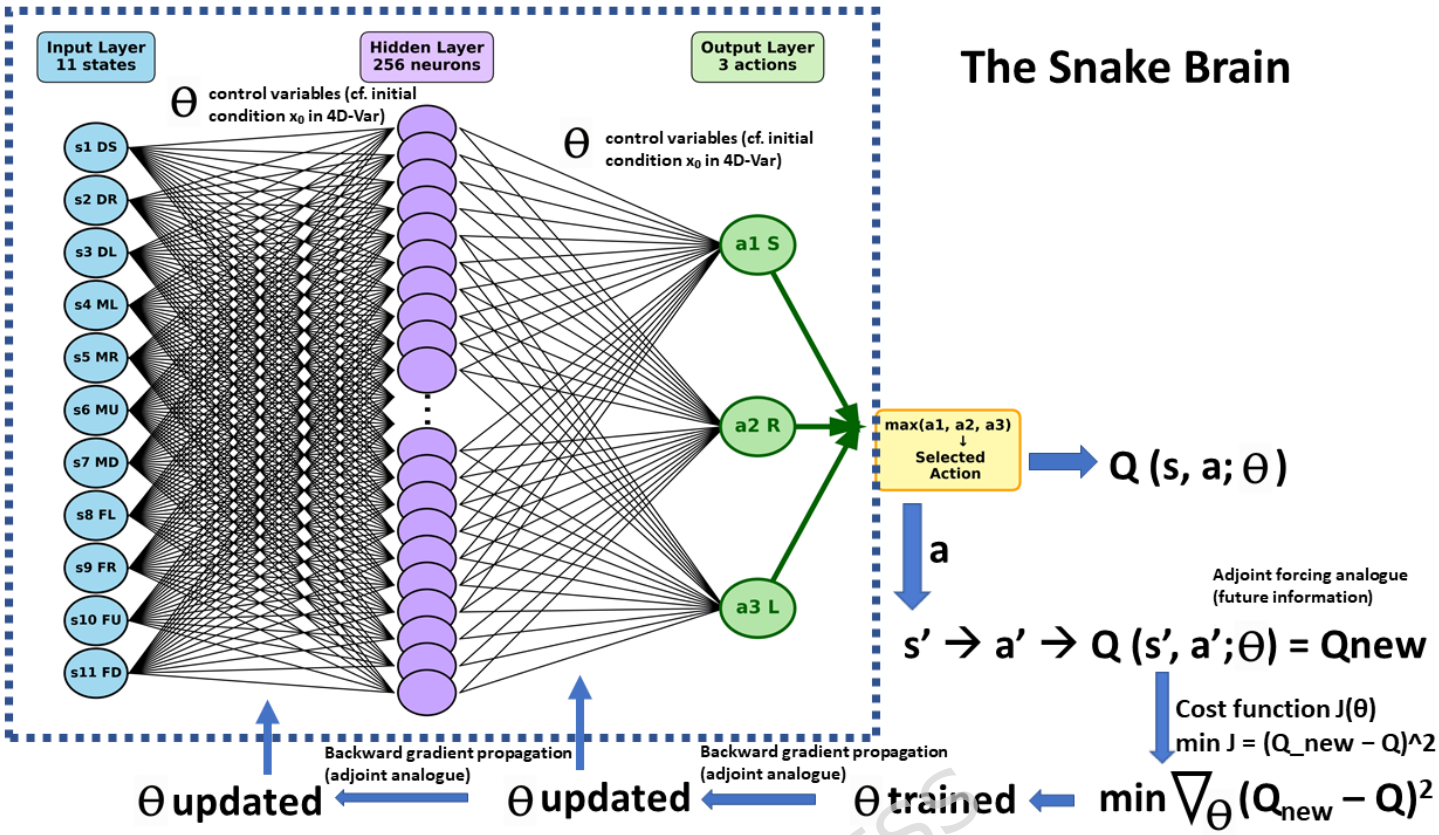

Figure 2: Architecture of the deep Q-network (DQN) used as the "Snake Brain" in an AI-controlled Snake game. The network consists of an input layer with 11 state features (representing danger straight DS, danger left DL, and danger right DR; food directions left FL, up FU, right FR, and down FD; and snake movement up MU, left ML, righ MR, and down MD), a fully-connected hidden layer with 256 neurons, and an output layer producing Q-values for three possible actions (Straight S, Right turn R, Left turn L). During inference, the action with the maximum Q-value is selected. During training, network parameters θ are updated by minimizing the squared temporal-difference error between the current Q-value and the target Q-value derived from the next state s', following standard Q-learning update rules. Annotations highlight the correspondence between network parameters, cost-function minimisation, and backward gradient propagation in reinforcement learning and their counterparts in four-dimensional variational data assimilation.

an early-stage policy choice, depending on the training iteration, and illustrates how state transitions arise naturally from the interaction between the agent's action and the game environment.

Figure 3a shows the layout of the Snake game and the neural network used to control the agent. The network consists of 11 input neurons representing the game state, a hidden layer of 256 neurons, and an output layer of three neurons representing possible actions. The 11 input states are danger straight (DS), danger right (DR), danger left (DL), move left (ML), move right (MR), move up (MU), move down (MD), food left (FL), food right (FR), food up (FU), and food down (FD).

The input layer connects to the hidden layer through 2,816 trainable parameters (11×256), denoted as weight θ_1. The hidden layer connects to the action layer through 768 parameters (256×3), denoted as weight θ_2. Together, $\theta = \theta_1, \theta_2$ defines the control variables of the system (Figure 2), analogous to the control variables (e.g. initial conditions or parameters) optimised in variational data assimilation.

3.1.2 Reward and backward information propagation

Learning is driven by rewards received from interactions with the environment. A positive reward is obtained when the snake eats food, while negative rewards occur when the snake collides with a wall or its own body. These reward signals are propagated backward through the network to update the control variables θ_1 and θ_2 via gradient-based optimisation.

States lead to actions through the network weights, actions lead to rewards through environmental feedback, and rewards in turn modify the weights that produced those actions. Training therefore consists of

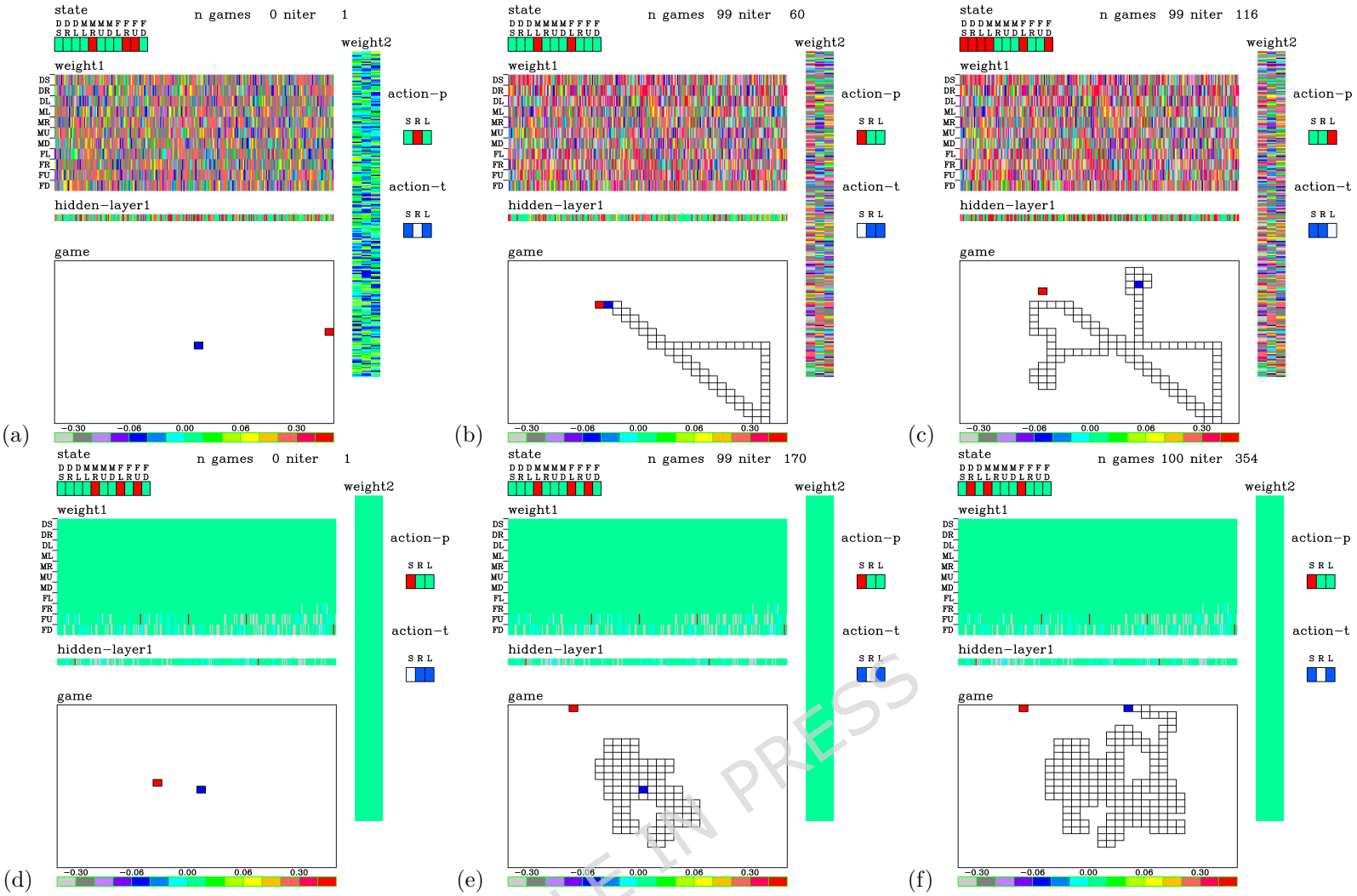

Figure 3: Visualisation of the Snake reinforcement-learning experiment at different stages of training. Each panel shows the internal network state, selected actions, and game evolution at a given game number and training iteration. The term *state* refers to the 11-dimensional binary input vector supplied to the neural network at each time step, encoding local environmental information around the snake head: danger straight (DS), danger right (DR), danger left (DL), move left (ML), move right (MR), move up (MU), move down (MD), food left (FL), food right (FR), food up (FU), and food down (FD). The horizontal axis in the state inset enumerates these 11 state components, while colours indicate binary values (active/inactive). The upper-left inset labelled `DDDMMMMFFFF` provides a compact mnemonic for the state vector, grouping danger (D), movement (M), and food-direction (F) indicators. Colours in this inset correspond directly to the binary state values and are determined by the game environment at each time step. The panels labelled *weight1* and *weight2* show the learned neural-network weight matrices connecting the input layer to the hidden layer (weight1) and the hidden layer to the output layer (weight2), respectively. These are visualised as colour maps to illustrate how the strength and sign of individual connections evolve during training. The structured patterns that emerge in trained cases reflect learned action preferences, whereas untrained cases exhibit random or uniform patterns. The quantities *action-p* and *action-t* denote the predicted action (argmax of the Q-values output by the network) and the action actually taken by the agent, respectively. Different colours are used to distinguish between straight (S), right (R), and left (L) actions. Differences between action-p and action-t arise during exploration, when actions are intentionally selected at random. The *game* panel shows the spatial evolution of the Snake environment, with the snake head shown in blue, food shown in red, and the snake body traced in grey. The colour bar below the game panel encodes the temporal sequence of actions taken during the displayed trajectory, with colours corresponding to different action types. Panels (a)–(c) show progressively trained states of the same agent, while panels (d)–(f) illustrate an untrained and subsequently trained agent for comparison, highlighting the emergence of structured behaviour as training proceeds. Here, n denotes the game (episode) index, while "iteration" refers to the discrete time step within a single game episode.

iteratively strengthening parameter combinations that lead to positive rewards and weakening those that lead to negative outcomes. This backward propagation of information is conceptually analogous to adjoint-based sensitivity propagation in variational data assimilation.

3.1.3 Exploration versus exploitation

In the game space, food is shown as a red square and the head of the snake as a blue square. Food locations are generated randomly. Once food is eaten, new food is generated at a different random location. Action selection follows an ϵ-greedy strategy. At the start of training, ϵ is set to $\epsilon_0 = 1.0$, corresponding to purely exploratory behaviour, and is linearly decayed to $\epsilon_{\min} = 0.01$ over the course of training. As ϵ decreases, action selection gradually transitions from exploration to exploitation, explaining the initially random and clustered trajectories observed in early games and the increasingly organised, food-directed behaviour observed at later stages.

During early games, actions are selected largely at random (exploration). As training progresses and rewards accumulate, action selection increasingly reflects the learned value estimates (exploitation). In the figures, predicted actions are indicated as action-p, while executed actions are indicated as action-t. The gradual transition from exploration to exploitation reflects increasing confidence in the optimised control variables.

In reinforcement learning, exploration plays a crucial role during the early stages of training, when the agent has not yet acquired reliable estimates of action values. A purely exploitative strategy at this stage would lock the agent into arbitrary and potentially suboptimal behaviour. Exploration therefore introduces controlled stochasticity into action selection, allowing different actions to be sampled and rewards to be observed across a diverse range of situations. This process enables the agent to acquire the information necessary for stable estimation of the value function and for the emergence of an effective policy. In this respect, exploration in reinforcement learning is conceptually analogous to the use of perturbations around a reference trajectory in incremental 4D-Var, where variations are introduced to probe the local sensitivity of the cost function and to guide subsequent optimisation. In both frameworks, such perturbations are not ends in themselves but serve to reveal informative gradients that drive learning or state adjustment.

3.1.4 Trained versus untrained trajectories

Figure 3b–c show examples from trained games, where the snake develops organised trajectories that consistently move toward food. Here, n denotes the game (episode) index, and "iteration" refers to the discrete time step within a single game episode. As optimisation proceeds, trajectories become increasingly structured and goal-directed.

In contrast, Figure 3d–f show trajectories produced by the same network architecture without training. In these cases, actions remain effectively random, and snake trajectories cluster around the initial position without systematic movement toward food.

The purpose of Figure 3 is therefore not to interpret individual weight values, but to illustrate the emergence of organised trajectories as a direct consequence of optimising the control variables. This distinction between structured and unstructured trajectories mirrors the role of optimisation in variational data assimilation, where improved control variables produce dynamically consistent trajectories that better satisfy observational constraints.

3.1.5 How training organises behaviour

An essential indicator of learning is the ability of the trained agent to consistently identify the relative location of food and generate actions that move toward it (Figure 3b–c). This requires transforming environmental information into effective actions through optimisation of the network parameters.

To understand how this transformation occurs, we track the evolution of the network parameters and hidden-layer activations across successive iterations. We define

$\Delta\theta_1(\mathrm{t}) = \theta_1(\mathrm{t}) - \theta_1(\mathrm{t}-1), \quad \Delta\theta_2(\mathrm{t}) = \theta_2(\mathrm{t}) - \theta_2(\mathrm{t}-1)$

and similarly for the hidden-layer activations. These incremental updates reflect local gradient steps taken during optimisation.

While individual parameter changes are not interpreted element-wise, their cumulative effect produces the transition from random exploration to organised, goal-directed behaviour. This process closely parallels variational data assimilation, where repeated gradient-based minimisation reshapes control variables to produce trajectories that better satisfy a cost function.

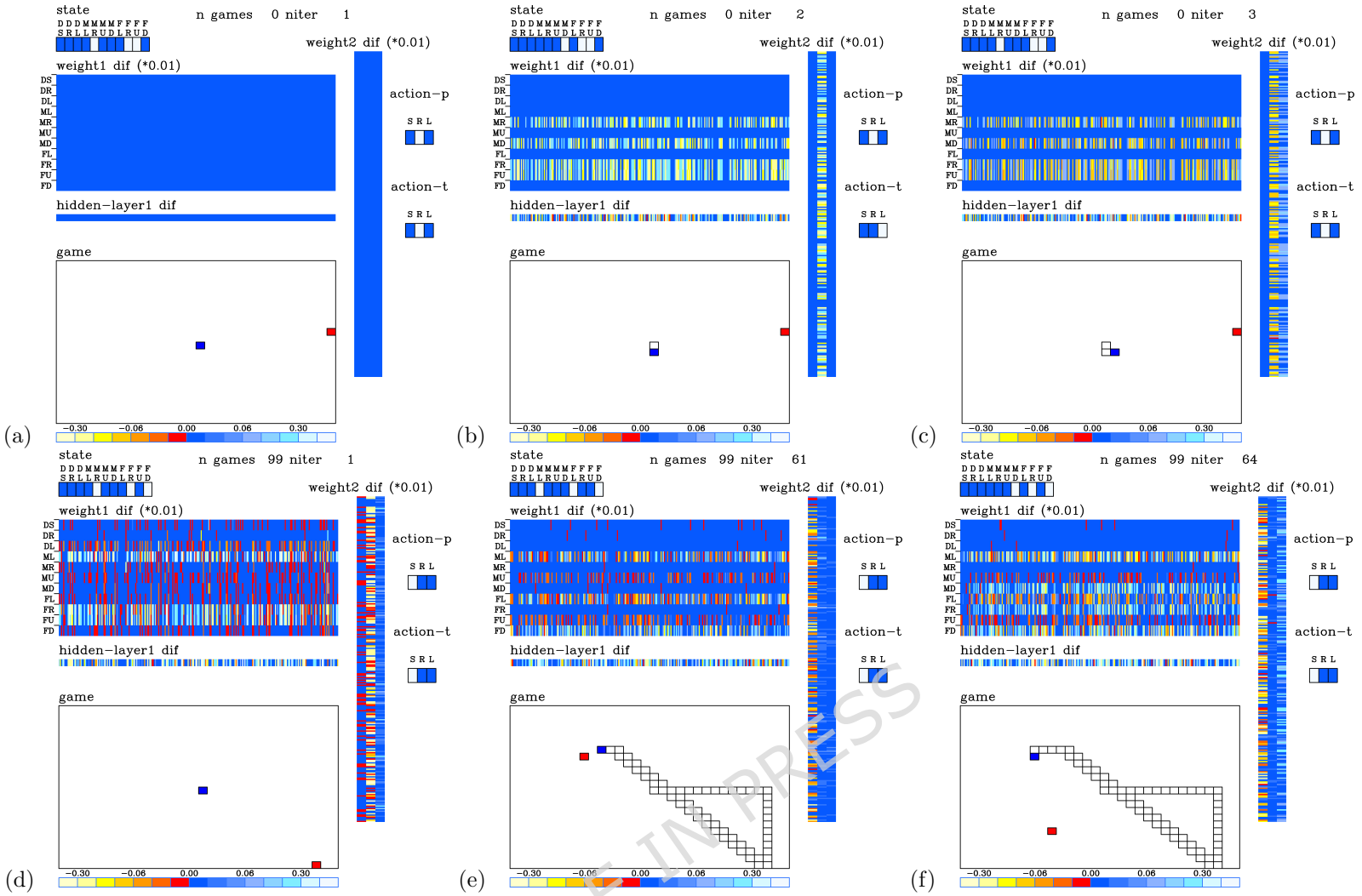

Figure 4: Evolution of the weight $\theta 1$ and weight $\theta 2$ during training. (a) n=0, iter=1. (b) n=0, iter=2. (c) n=0, iter=3. (d) n=99, iter=1. (e) n=99, iter=61. (f) n=99, iter=64. Blue colours correspond to negligible parameter updates, while warmer colours indicate larger updates. The horizontal axis enumerates individual weight components. These panels visualises the *changes* in network parameters between successive training iterations rather than their absolute values. The colour maps labelled $\Delta\theta_1$, $\Delta\theta_2$, and Δhidden layer represent element-wise differences between parameters at iteration t and t − 1. Blue indicates values close to zero (no update), while warmer colours indicate larger positive or negative updates. The horizontal axis enumerates individual components of the corresponding weight vectors (or matrices flattened into vectors), allowing the spatial distribution of parameter updates to be visualised. Displaying parameter increments highlights where learning signals are injected into the network, making visible the backward propagation of reward information that is otherwise hidden when only states or actions are shown.

3.2 Evolution of network parameters and backward information propagation

Figure4 illustrates the evolution of incremental changes in the network parameters θ_1, θ_2, and the hidden-layer activations during training. Rather than interpreting individual weight values, the figure is used to visualise how future reward information propagates backward through the optimisation variables, analogous to adjoint sensitivity propagation in incremental four-dimensional variational data assimilation (4D-Var).

At the start of training (Figure fig.fig2a), the parameters are unadjusted and no learning has occurred. As the game proceeds (Figure 4b–c), parameter updates appear only along pathways associated with recently visited states and actions. This localisation of updates reflects a short-horizon gradient response: only states contributing to the immediate forecast error (reward) influence the optimisation step. In 4D-Var terms, this corresponds to an inner-loop minimisation, where gradients of a linearised cost function are propagated backward over a fixed window to adjust the control variables.

In the Snake environment, the reward signal is sparse and event-based, taking non-zero values only when

the snake eats food or collides with a wall or itself, while intermediate steps yield zero reward but still contribute to learning through the temporal-difference target.

As training progresses across multiple games (Figure 4d–f), parameter updates accumulate from repeated episodes. These updates reflect the reinitialisation of the optimisation around an evolving reference trajectory, conceptually analogous to the outer-loop relinearisation used in nonlinear 4D-Var. Information gained from previous episodes influences subsequent optimisation windows, shaping the parameter field in a manner similar to how prior windows influence subsequent assimilation cycles.

Thus, Figure 4 visualises the same structural optimisation process present in 4D-Var: repeated backward propagation of error information combined with iterative reinitialisation around an updated nonlinear trajectory. At early training stages, the update patterns appear spatially unstructured because actions are largely exploratory and rewards are sparse, resulting in small and diffuse parameter changes. As training progresses, coherent structures emerge in $\Delta\theta_1$ and $\Delta\theta_2$, reflecting consistent credit assignment associated with food-directed actions.

While the game state evolution is shown in Figure 3, Figure 4 focuses on the internal learning dynamics by visualising parameter updates, which provide complementary insight into how reward information is propagated backward through the network.

3.3 Trained versus untrained trajectories

Figure 5 contrasts the behaviour of trained and untrained networks to demonstrate the macroscopic effect of backward information propagation on system trajectories. The figure is not intended to illustrate individual weight magnitudes, but rather to show how optimisation reorganises trajectories in state space.

In trained cases (Figure 5a–c), the agent consistently selects actions that move it toward the target (food), indicating that backward-propagated reward information has reshaped the parameter field to favour dynamically coherent trajectories. In contrast, the untrained cases (Figure 5d–f) exhibit random, spatially clustered motion around the initial location, reflecting the absence of any optimisation constraint.

This distinction mirrors a fundamental outcome of variational data assimilation: an unoptimised forecast evolves freely from its initial condition, while an optimised forecast follows a dynamically consistent trajectory that minimises a cost function measuring misfit to observations. In both cases, trajectory organisation is the emergent result of backward-propagated information acting on control variables—network parameters in reinforcement learning and initial conditions in 4D-Var.

3.4 Verification of the reward mechanism

The key factor for a machine to learn to play the game is the rewards mechanism. Figure 6a shows the evolution of the values of the actions and the rewards with respect to the games. The values of the actions vary between 0 and -4 in the first 50 games. As the games learn from the rewards, the values of the actions gradually increase with the games played. After game 150, the dominant values of the actions gradually converged to between -2 and 8. The value 8 is close to the positive reward of 10 when a food is eaten by the snake. The -2 is close to the negative reward of -10 when a snake hits the wall or runs into itself. After about 50 games, the snake has gradually learned to avoid walls. The negative rewards are resulting from a snake that has grown too long (as more food is eaten), and the snake runs into itself. Figuree 6b shows the evolution of the values of the actions for the games that have been trained for the first 150 games. After game 150, the games were played without the training. The computer has already learned to play games. In other words, the weight $\theta 1$ and the weight $\theta 2$ have been fully trained to play the game all by themselves.

4 Discussion

This work does not propose a new optimisation method, nor does it aim to improve the performance of state-of-the-art reinforcement learning. Instead, its contribution lies in clarifying the shared mathematical structure underlying reinforcement learning and variational data assimilation through a deliberately simple, fully observable system. By visualising weight evolution in a compact neural network, the study provides an accessible analogue to adjoint sensitivity propagation familiar to the atmospheric science community, where incremental updates accumulate to reshape a model trajectory. The Snake laboratory is intended not as an application model, but as a transparent didactic system that allows atmospheric scientists to directly observe backward information propagation—an otherwise hidden process in operational data assimilation systems.

A natural Earth-system context in which both reinforcement learning and variational data assimilation may be applied is atmospheric state estimation for numerical weather prediction. In operational practice, incremental 4D-Var estimates the atmospheric initial condition by iteratively minimising a cost function

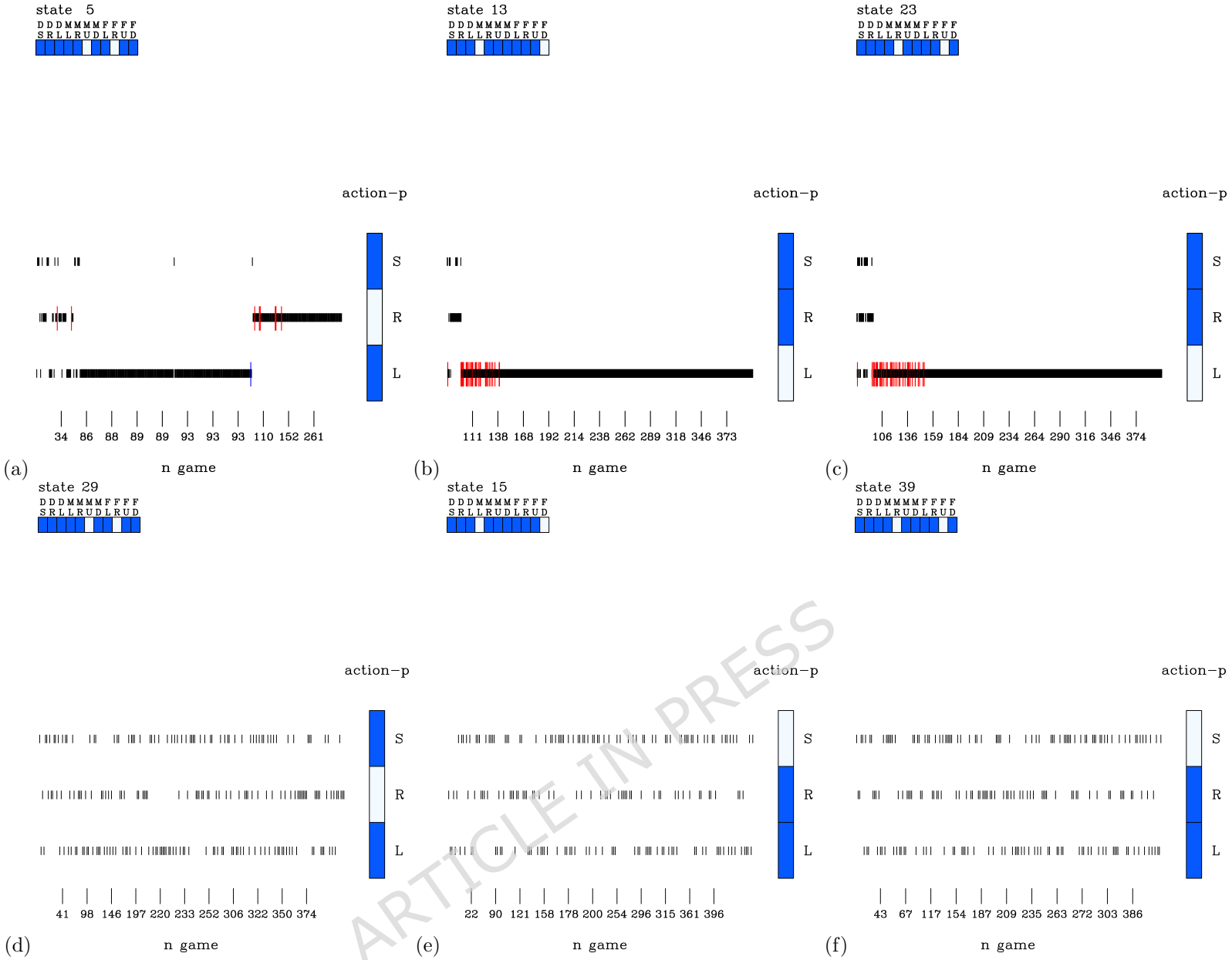

Figure 5: Evolution of actions with respect to specific input states. (a) Trained actions for state MU and FR. (b) Trained actions for state ML and FD. (c) Trained actions for state MR and FU. (d) Untrained actions for same states as in (a). (e) Untrained actions for same states as in (b). (f) Untrained actions for same states as in (c). The short red vertical lines indicate the reward was found during the training process.

that measures misfit to observations distributed over a time window, using forward model integration and adjoint-based gradient propagation. In contrast, reinforcement learning formulations have been explored in which the atmospheric state is treated as the environment, model evolution defines the state transition, and a reward function penalises forecast error or observation misfit. Although the objectives and constraints differ, both approaches involve repeated forward model evaluation, error or reward assessment along a trajectory, and backward propagation of information to update a set of control variables. This conceptual overlap makes atmospheric data assimilation a useful Earth-system example for illustrating the structural similarities discussed in this study, without implying algorithmic equivalence or operational substitution.

Table 2 summarises the algorithmic operation counts and optimisation characteristics of reinforcement learning and incremental 4D-Var, providing a quantitative basis for discussing their respective advantages and limitations.

We emphasise that the analogy developed in this study is not intended to suggest algorithmic equivalence between reinforcement learning and variational data assimilation. In particular, explicit exploration—through stochastic action selection—is a defining feature of reinforcement learning and has no direct

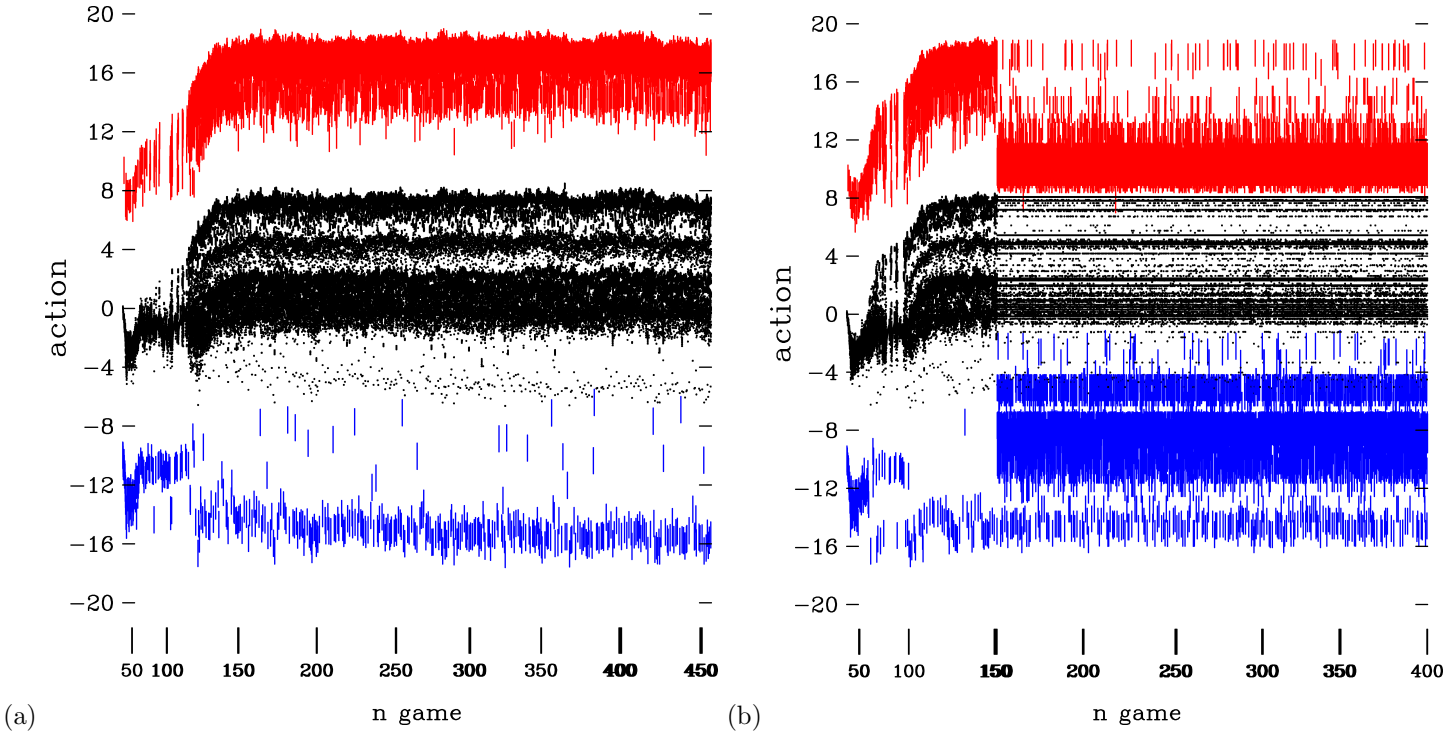

Figure 6: Evolution of action values during training. (a) Action-value evolution across 450 trained games. (b) First 150 games trained; remaining games played without training using the trained weight $\theta 1$ and weight $\theta 2$.

counterpart in deterministic incremental 4D-Var, where optimisation proceeds along a single reference trajectory using adjoint-derived gradients. In this sense, reinforcement learning shares stronger similarities with stochastic data-assimilation approaches, such as Monte Carlo or particle-based methods, which explicitly sample state or parameter space. The present work therefore focuses on a more limited but well-defined correspondence: the shared use of forward model evaluation, backward propagation of sensitivity information, and iterative reinitialisation around evolving trajectories, which can be visualised transparently in a reinforcement-learning setting.

Limitations of the Reinforcement Learning–4D-Var Analogy

While reinforcement learning and four-dimensional variational data assimilation share a common optimisation structure—namely the iterative minimisation of a scalar objective function using forward model evaluation and backward sensitivity propagation—their goals and interpretations differ fundamentally. In classical incremental 4D-Var, optimisation is performed over the system state or a set of physical model parameters within a fixed dynamical model and over a prescribed assimilation window. The objective is state correction: to recover the most probable system trajectory consistent with observations and prior information.

In contrast, reinforcement learning optimises the parameters of a policy or value function, typically represented by a neural network, that governs the agent's behaviour across repeated interactions with an environment. The objective is behavioural learning rather than state estimation, and learning proceeds through trial-and-error interactions mediated by a reward signal. As a result, while the backward propagation of sensitivity information in reinforcement learning is mathematically analogous to adjoint-based optimisation in 4D-Var, the quantities being optimised and their physical interpretations are fundamentally different.

The analogy developed in this study is therefore intended to be structural and pedagogical, highlighting shared optimisation mechanics rather than implying algorithmic equivalence or interchangeability between the two frameworks.

The use of a compact neural network necessarily abstracts away effects associated with overparameterisation, such as redundancy and implicit regularisation. The present analysis therefore does not aim to characterise all learning dynamics observed in large-scale deep reinforcement-learning systems. Instead, it focuses on making the core optimisation structure visible in a controlled setting, with the expectation that these structural elements persist—though in more complex form—in larger models.

While the present study does not introduce a new data-assimilation or DA–ML algorithm, it contributes

Table 2: Algorithmic operation counts and optimisation characteristics of reinforcement learning (Q-learning / DQN) and incremental 4D-Var data assimilation. The comparison focuses on structural and algorithmic aspects rather than implementation-dependent computational cost.

Aspect	**Reinforcement Learning (Q-learning / DQN)**	**Incremental 4D-Var Data Assimilation**
Primary objective	Minimise temporal-difference loss $L_t = [Q_{new/target} - Q(s_t, a_t; \theta)]^2$	Minimise variational cost function $J(x_0)$ over an assimilation window
Control variables	Neural-network parameters θ	Initial state (or control vector) x_0
Forward model evaluation	One environment transition per time step	One nonlinear model integration per outer-loop iteration
Backward information propagation	Gradient backpropagation through the network at each update	Adjoint model integration backward in time per outer loop
Update frequency	Frequent, local updates (per step or mini-batch)	Infrequent, global updates (per assimilation window)
Optimisation method	Stochastic gradient descent or variants	Deterministic gradient-based variational minimisation
Data usage structure	Sequential / online interaction with environment	Batch use of observations distributed in time
Scalability characteristics	Cost dominated by number of updates and network size	Cost dominated by forward and adjoint model integrations
Typical advantages	Flexible, online learning; no adjoint required; adaptable to non-stationary environments	Physically constrained optimisation; globally consistent state estimate
Typical limitations	Large number of updates; stochastic convergence; limited physical interpretability	Adjoint development required; batch-oriented; high per-iteration cost

to the growing literature on machine learning combined with data assimilation and uncertainty quantification by providing an interpretable view of optimisation dynamics. In hybrid DA–ML frameworks, neural networks are increasingly used as surrogate models, error representations, or components of observation operators. The visualisation strategy presented here offers a diagnostic perspective on how gradient information propagates through such systems, complementing existing methodological and application-focused studies. In this sense, the work may aid the interpretation, training diagnostics, and pedagogical understanding of complex DA–ML systems rather than directly improving forecast skill.

Although the visualisation strategy presented here is demonstrated in a minimal reinforcement-learning setting, its conceptual basis may inform diagnostic tools for large-scale data-assimilation systems. In operational 4D-Var, adjoint sensitivities and incremental updates are routinely computed but rarely inspected in a structured, visual manner. Extending similar visual diagnostics to examine the localisation, persistence, and accumulation of sensitivities across assimilation windows could support model debugging and system evaluation, even if direct visualisation of full state spaces remains infeasible. Such extensions are beyond the scope of the present study but represent a potential avenue for future development.

All code, trained models, and per-iteration diagnostics are openly available, enabling readers to reproduce and explore the optimisation dynamics without requiring specialised machine-learning infrastructure.

5 Conclusions

This study demonstrates, in a fully transparent and visually interpretable setting, how backward information propagation operates in deep reinforcement learning and how this process parallels adjoint-based optimisation in four-dimensional variational data assimilation. While the existence of a shared optimisation structure between machine learning and data assimilation has been established in previous theoretical and review studies [48, 47, 40, 41, 46], the present work contributes by making this structure directly observable at the level of individual iterations and parameter updates in a simple reinforcement-learning system. Rather than identifying new mathematical equivalences, the contribution of this work is to render a well-established optimisation structure explicit and observable in a simple, interpretable setting.

By explicitly visualising the evolution of all network parameters during training, we show how behavioural adaptation in reinforcement learning emerges from the accumulation of small, backward-propagated gradient updates, closely analogous to the way observational information is propagated backward in time to refine atmospheric state estimates in 4D-Var. This perspective emphasises optimisation dynamics rather than application performance.

Viewing reinforcement learning through a variational data assimilation lens provides a pedagogical and interpretative bridge between the two communities. Rather than proposing new algorithms or applications, this work aims to clarify how apparently distinct optimisation frameworks are governed by the same underlying principles. By making backward information propagation explicit, the study offers a transparent conceptual framework that may help atmospheric scientists interpret modern learning systems and help machine-learning practitioners better appreciate variational data assimilation methodology.

6 Algorithm

The reinforcement-learning algorithm employed in this study follows the standard deep Q-learning framework. At each iteration t, the current state vector s_t is propagated forward through the network,

$$s_t \;\rightarrow\; \mathbf{W}_1 \;\rightarrow\; \text{hidden layer} \;\rightarrow\; \mathbf{W}_2 \;\rightarrow\; Q(s_t, \cdot),$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ denote the input-to-hidden and hidden-to-output weight matrices, respectively. The network outputs three action values corresponding to the discrete actions (straight, left, right), and the predicted action value is a'

$$a' = \max_a Q(s_t, a) \;\rightarrow\; s'$$

After executing the selected action, the environment advances to the next state s' and returns a scalar reward r_{t+1}. The temporal-difference (TD) target is then computed as

$$Q_{new} = r_{t+1} + \gamma \max_{a'} Q(s', a'),$$

where $\gamma \in (0, 1)$ is the discount factor. The network parameters are updated by minimising the squared TD error,

$$\mathcal{L}_t = \left[Q_{\text{new}} - Q\right]^2,$$

with respect to $\mathbf{W}_1$ and $\mathbf{W}_2$ using gradient descent. Gradients are evaluated via backpropagation.

The network architecture consists of 11 binary input neurons, a fully connected hidden layer of 256 neurons, and an output layer of three action-value neurons. Accordingly, $\mathbf{W}_1$ has dimension 11×256 and $\mathbf{W}_2$ has dimension 256×3, yielding a total of 3,584 trainable parameters.

The algorithm was implemented in Python using the PyTorch framework to perform forward evaluation, gradient computation, and parameter updates. The approach is not specific to PyTorch; any programming environment supporting matrix operations and automatic differentiation can be used to implement the same learning procedure.

Data and Code Availability

All code, trained models, per-iteration weight movies, and figures are permanently archived at
`https://doi.org/10.6084/m9.figshare.30795956`

Acknowledgments

We gratefully acknowledge the adoption of the python codes from Patrick Loeber in this work, and the open sources Python (3.9.5), PyTorch, Pygame, PyCharm (2021.1.1 Community Edition), Linux, Cygwin, Fortran, NCAR (National Center for Atmospheric Research, USA) Graphics, and ffmpeg for this works. All figures produced in this work were produced based on the NCAR graphics and the Fortran codes.

Author Contribution

KYW researched, coded, wrote, and reviewed the manuscript.

Funding Declaration

This work was funded by the National Science and Technology Council under the grant MOST 107-2111-M-008-027-.

References

[1] Mnih, V. et al. (2015). Human-level control through deep reinforcement learning. *Nature* 518, 529–533.

[2] Silver, D. et al. (2016), Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 484–489 (2016).

[3] Abadi, M. et al. (2015), TensorFlow: Large-scale machine learning on heterogeneous systems. Available at tensorflow.org.

[4] Paszke, A. et al. (2019), PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32* 8024–8035.

[5] ThinkAutomation (2021), The AI black box problem. https://www.thinkautomation.com/bots-and-ai/the-ai-black-box-problem/

[6] Bleicher, A. (2017), Demystifying the black box that is AI. *Scientific American*. https://www.scientificamerican.com/article/demystifying-the-black-box-that-is-ai/

[7] Rudin, C. & Radin, J. (2019), Why are we using black box models in AI when we do not need to? *Harvard Data Science Review*.

[8] Castelvecchi, D. (2016), Can we open the black box of AI? *Nature* 538, 20–23.

[9] ODSC (2019). AI black box horror stories – when transparency was needed more than ever. https://medium.com/@ODSC/ai-black-box-horror-stories-when-transparency-was-needed-more-than-ever-3d6ac043

[10] Manica, M. & Cadow, J. (2019), Opening the black box – interpretability in deep learning. https://opendatascience.com/opening-the-black-box-interpretability-in-deep-learning/

[11] Bathaee, S. (2018), The artificial intelligence black box and the failure of intent and causation. *Harvard Journal of Law and Technology* 31(2), 890–938.

[12] Bender, E. (2019), Unpacking the black box in artificial intelligence for medicine. https://undark.org/2019/12/04/black-box-artificial-intelligence/

[13] Big Cloud (2020), The difference between white box and black box AI. https://bigcloud.global/the-difference-between-white-box-and-black-box-ai/

[14] Deloitte (2021), Managing the black box of artificial intelligence (AI). https://www2.deloitte.com/us/en/pages/advisory/articles/black-box-artificial-intelligence.html

[15] Roelenga, B. 2021, Think outside the black box. https://towardsdatascience.com/think-outside-the-black-box-7e6c95bd2234

[16] LeCun, Y., Bottou, L., Orr, G. B. & Müller, K.-R. (1998), Efficient Backprop. In *Neural Networks: Tricks of the Trade* (Springer, 1998).

[17] Haykin, S. (1999), *Neural Networks: A Comprehensive Foundation*. (Prentice Hall, 1999).

[18] Bellman, R. (1952), On the theory of dynamic programming. *Proc. Natl. Acad. Sci. USA* 38, 716–719.

[19] Daley, R. (1991), *Atmospheric Data Analysis*. (Cambridge Univ. Press, 1991).

[20] Wang, K.-Y. et al. (2001), A review on the use of the adjoint method in four-dimensional atmospheric data assimilation. *Q. J. R. Meteorol. Soc.* 127(576), 2181–2204.

[21] Vodopivec, T., Samothrakis, S. & Ster, B. (2017), On Monte Carlo tree search and reinforcement learning. *J. Artif. Intell. Res.* 60, 881–936.

[22] Clear, J. (2018), *Atomic Habits*. (Avery, New York, 2018).

[23] Gerald, C. F. & Wheatley, P. O. (1998), *Applied Numerical Analysis*. 6th Ed. (Addison-Wesley, 1998).

[24] Loeber, P. (2021), Teach AI to play snake! Reinforcement learning with PyTorch and Pygame. `https://github.com/python-engineer/snake-ai-pytorch`

[25] Ham, Y.-G., Kim, J.-H. & Luo, J.-J. (2019), Deep learning for multi-year ENSO forecasts. *Nature* 573, 568–572.

[26] Ravuri, S. et al. (2021), Skillful precipitation nowcasting using deep generative models of radar. *Nature* 597, 672–677.

[27] Mihai, D. (2019), Artificial intelligence for very high resolution earth observation. `https://elib.dlr.de/130905`

[28] Zhou, L. et al. (2017), Machine learning on big data: opportunities and challenges. *Neurocomputing* 237, 350–361.

[29] Kruitwagen, L. (2021), Mapping global solar plants using satellites and machine learning. `https://theconversation.com/we-mapped-every-large-solar-plant-on-the-planet-using-satellites-and-machine-`

[30] Kruitwagen, L. et al. (2021), A global inventory of photovoltaic solar energy generating units. *Nature* 598, 604–610.

[31] Reichstein, M. et al. (2019), Deep learning and process understanding for data-driven Earth system science. *Nature* 566, 195–204.

[32] Baek, A., et al. (2024), Online machine learning for Earth system digital twins. *Nature Machine Intelligence*, **6**, 987–998.

[33] Cheng, S., et al. (2024), Physics-informed machine learning for weather and climate modeling. *Rev. Geophys.*, **62**, e2023RG000812.

[34] Ebrahimi, M., et al. (2024), Artificial intelligence for geoscience: Progress, challenges, and perspectives. *Earth-Sci. Rev.* , **249**, 104650.

[35] Kochkov, D., et al. (2024), Machine learning acceleration of Earth system models. *Nature*, **624**, 782–789.

[36] Le Dimet, F.-X., & Talagrand, O. (1986), Variational algorithms in meteorology. *Q. J. R. Meteorol. Soc.*, **112**, 473–494.

[37] Lee, H., et al. (2024), Reinforcement learning for ensemble data assimilation. *J. Adv. Model. Earth Syst.*, **16**, e2023MS004012.

[38] Talagrand, O., & Courtier, P. (1987), Variational assimilation of meteorological observations with the adjoint vorticity equation. *Q. J. R. Meteorol. Soc.*, **113**, 1311–1328.

[39] Irrgang, C. et al. (2021), Will artificial intelligence supersede Earth system models?, *Nature Machine Intelligence*.

[40] Abarbanel, H. D. I., Rozdeba, P.J., Shirman, S. (2018), Machine Learning: Deepest Learning as Statistical Data Assimilation Problems. *Neural Computation*, **30(8)**, 2025–2055, `https://doi.org/10.1162/neco_a_01094`.

[41] Geer, A.J. (2021), Learning earth system models from observations: machine learning or data assimilation? *Philos. Trans. A Math. Phys. Eng. Sci.*, **5**, 379(2194):20200089, `doi:10.1098/rsta.2020.0089`.

[42] Gratton, S., Lawless, A. S., & Nichols, N. K. (2007), Approximate Gauss–Newton methods for nonlinear least squares problems. *SIAM Journal on Optimization*, **18(1)**, 106–132, `https://doi.org/10.1137/050624935`.

[43] Lawless, A. S., Gratton, S., & Nichols, N. K. (2005), An investigation of incremental 4d-var using non-tangent linear models. *Quarterly Journal of the Royal Meteorological Society*, **131(606)**, 459–476, `https://doi.org/10.1256/qj.04.20`.

[44] Lawless, A. S., & Nichols, N. K. (2006), Inner-loop stopping criteria for incremental four-dimensional variational data assimilation. *Monthly Weather Review*, **134(11)**, 3425–3435, https://doi.org/10.1175/MWR3242.1.

[45] Solvik, K., Penny, S. G., & Hoyer, S. (2025), 4D-Var using hessian approximation and backpropagation applied to automatically differentiable numerical and machine learning models. *Journal of Advances in Modeling Earth Systems*, **17**, e2024MS004608, https://doi.org/10.1029/2024MS004608.

[46] Abarbanel, H.D.I. 2022, The Statistical Physics of Data Assimilation and Machine Learning. Cambridge University Press.

[47] Gardner, M. W. Dorling, S. R. (1998) Artificial Neural Networks (The Multilayer Perceptron) - A Review of Applications in the Atmospheric Sciences. *Atmospheric Environment*, **32 (14/15)**, 2627–2636.

[48] Miller, R. N., Ghil, M. Gauthiez, F. (1994). Advanced Data Assimilation in Strongly Nonlinear Dynamical Systems. *Journal of the Atmospheric Sciences*, **51(8)**, 1037-1056.