frontiers in Applied Mathematics and Statistics

ORIGINAL RESEARCH
published: 02 August 2021
doi: 10.3389/fams.2021.689934

Deep Reinforcement Learning for Constrained Field Development Optimization in Subsurface Two-phase Flow

Yusuf Nasir, Jincong He, Chaoshun Hu, Shusei Tanaka, Kainan Wang and XianHuan Wen*

Chevron Technical Center, Houston, TX, United States

Oil and gas field development optimization, which involves the determination of the optimal number of wells, their drilling sequence and locations while satisfying operational and economic constraints, represents a challenging computational problem. In this work, we present a deep-reinforcement-learning-based artificial intelligence agent that could provide optimized development plans given a basic description of the reservoir and rock/fluid properties with minimal computational cost. This artificial intelligence agent, comprising of a convolutional neural network, provides a mapping from a given state of the reservoir model, constraints, and economic condition to the optimal decision (drill/do not drill and well location) to be taken in the next stage of the defined sequential field development planning process. The state of the reservoir model is defined using parameters that appear in the governing equations of the two-phase flow (such as well index, transmissibility, fluid mobility, and accumulation, etc.,). A feedback loop training process referred to as deep reinforcement learning is used to train an artificial intelligence agent with such a capability. The training entails millions of flow simulations with varying reservoir model descriptions (structural, rock and fluid properties), operational constraints (maximum liquid production, drilling duration, and water-cut limit), and economic conditions. The parameters that define the reservoir model, operational constraints, and economic conditions are randomly sampled from a defined range of applicability. Several algorithmic treatments are introduced to enhance the training of the artificial intelligence agent. After appropriate training, the artificial intelligence agent provides an optimized field development plan instantly for new scenarios within the defined range of applicability. This approach has advantages over traditional optimization algorithms (e.g., particle swarm optimization, genetic algorithm) that are generally used to find a solution for a specific field development scenario and typically not generalizable to different scenarios. The performance of the artificial intelligence agents for two- and three-dimensional subsurface flow are compared to well-pattern agents. Optimization results using the new procedure are shown to significantly outperform those from the well pattern agents.

Keywords: deep reinforcement learning (deep RL), field development optimization, subsurface flow, two-phase flow, neural network, proximal policy optimization (PPO)

OPEN ACCESS

Edited by:
Dongxiao Zhang,
Southern University of Science and Technology, China

Reviewed by:
Mikhail Basarab,
Bauman Moscow State Technical University, Russia
Haibin Chang,
Peking University, China
Nanzhe Wang,
Peking University, China

****Correspondence:***
Jincong He
JincongHe@chevron.com

Specialty section:
This article was submitted to Optimization, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *01 April 2021*
Accepted: *14 July 2021*
Published: *02 August 2021*

Citation:
Nasir Y, He J, Hu C, Tanaka S, Wang K and Wen X (2021) Deep Reinforcement Learning for Constrained Field Development Optimization in Subsurface Two-phase Flow. Front. Appl. Math. Stat. 7:689934. doi: 10.3389/fams.2021.689934

1 INTRODUCTION

Field development decisions such as the number of wells to drill, their location and drilling sequence need to be made optimally to maximize the value realized from a petroleum asset. Optimization algorithms such as evolutionary strategies are, in recent times, widely applied to the field development optimization problem [1,2]. However, the application of these optimization algorithms is challenging due to the very large number of computationally expensive flow simulations required to obtain optimal (or near optimal) solutions. In addition, the field development optimization problem is typically solved separately for each petroleum field due to the variation in geological model, constraints to be considered or even the economic condition. Thus, the large number of computationally expensive flow simulations needs to be run for each field under consideration. This suggests the field development optimization problem will benefit from strategies that would allow for the generalization of the optimization process to several petroleum fields.

In our recent work He et al. [3], we developed a deep reinforcement learning technique for the field development optimization in two-dimensional subsurface single-phase flow settings. The deep reinforcement learning technique allows for the training of an artificial intelligence agent that provides a mapping from the current state of a two-dimensional reservoir model to the optimal decision (drill/do not drill and well location) in the next step of the development plan. Our goal in this work is to extend the procedures in He et al. [3] for the field development optimization, in the presence of operational constraints, in both two- and three-dimensional subsurface two-phase flow. Once properly trained, the artificial intelligence agent should learn the field development logic and provide optimized field development plans instantly for different field development scenarios. Besides the described extensions to more complex simulation models, several algorithmic treatments are also introduced in this work to enhance the training of the artificial intelligence agent under the deep reinforcement learning framework.

In the literature of oil and gas field development and production optimization, different optimization algorithms have been applied to solve different aspects of the optimization problem. The well control or production optimization problem in which the time-varying operational settings of existing wells are optimized has been efficiently solved with gradient-based methods [4–7], ensemble-based methods [8,9] or with efficient proxies like reduced-order models [10,11]. For the well placement optimization, popular algorithms such as the genetic algorithms and the particle swarm optimization algorithms [1,2,12–14], typically entail thousands of simulation runs to get improved result with no guarantee of global optimality. The joint well placement and production optimization problems have also been considered [1,2,15–17], which typically require even larger numbers of simulations. In addition, while the approaches considered in these studies may provide satisfactory results for the field development optimization problem, the solution obtained in each case is tied to a specific field development scenario. If the economic conditions or geological models used in the optimization change, the optimization process needs to be repeated. In other words, the solution from traditional optimization methods lacks the ability to generalize when the underlying scenario changes. It should be noted that the ability to generalize that is discussed here is different from the robust optimization as studied in Chen et al. [18,19], in which the optimization is performed under uncertainty. While the solutions from robust optimization accounts for the uncertainty in the model parameters, they still don't generalize when the ranges of those uncertainties change.

In this work, we consider the reinforcement learning technique in which a general AI (in the form of a deep neural network) can be applied to optimize field development planning for a range of different scenarios (e.g., different reservoir, different economics, different operational constraints) once it is trained. The reinforcement learning technique considered in this work for the field development optimization problem has shown great promise in other fields. Google DeepMind trained an artificial intelligence agent AlphaGo [20] using reinforcement learning with a database of human expert games. AlphaGo beat the human world champion player in the game of Go. AlphaGo Zero [21] which is a variant of AlphaGo was trained through self-play without any human knowledge. AlphaGo Zero defeated AlphaGo in the game of Go. In AlphaGo and AlphaGo Zero, the AI takes in a description of the current state of the Go game and chooses an action to take for the next step. The choice of action is optimized in the sense that it maximizes the overall probability of winning the game. The success of AlphaGo and AlphaGo Zero suggests the possibility of training an AI for field development that takes in a description of the current state of the reservoir and chooses the development options (e.g., drilling actions) for the next step, without prior reservoir engineering knowledge.

In the petroleum engineering literature, deep learning algorithms have seen much success recently for constructing proxies for reservoir simulation models [22–24]. Reinforcement learning algorithms have been applied to solve the production optimization problem for both steam injection in steam-assisted gravity drainage (SAGD) recovery process [25] and waterflooding [26]. Deep reinforcement learning (in which the artificial intelligence agent is represented by a deep neural network) has also been applied to the production optimization problem [27,28]. Ma et al. [27] evaluated the performance of different deep reinforcement learning algorithms on the well control optimization problem with fully connected neural networks (FCNN). The state of the reservoir model defined by two-dimensional maps are flattened and used as input to the FCNN and thus do not retain the spatial information inherent in the data. An FCNN-based artificial intelligence agent was also trained in Miftakhov et al. [28]. It should be noted that in the studies discussed here, the reinforcement learning process is used as a replacement for traditional optimization algorithms (such as particle swarm optimization, genetic algorithm) to optimize a predefined field development scenario.

He et al. [3] applied the deep reinforcement learning technique to the more challenging field development optimization problem in which the decisions to be made include the number of wells to drill, their locations, and drilling sequence. In contrast to the

previous studies discussed, He et al. [3] used DRL to develop artificial intelligence which could provide optimized field development plans given any reservoir description within a predefined range of applicability. In addition, unlike in previous studies [27,28] in which the artificial intelligence agent is represented by fully connected neural networks, He et al. [3] utilized a convolutional neural network to represent the agent which allows for better processing of spatial information. The field development optimization problem in He et al. [3], however, considers a single-phase flow in two-dimensional reservoir models.

In this paper, we build upon He et al. [3] and formulate deep reinforcement learning-based artificial intelligence agents that could provide optimized field development plans instantaneously based on the description of a two- or three-dimensional subsurface two-phase flow in the presence of operational constraints. The sequence of actions to take during the field development process is made by the artificial intelligence agent after processing the state of the reservoir model, the prescribed constraints and economic condition. The state of the reservoir model in this work is defined using parameters that appear in the governing equations of the two-phase flow (such as pressure, saturation, well index, transmissibility, fluid mobility, and accumulation). We also propose a dual-action probability distribution parameterization and an improved convolutional neural network architecture to enhance the training efficiency of the artificial intelligence agents for the field development optimization problem.

Compared to prior work on this topic such as He et al. [3], the novelties in this work include: 1) extended the problem description from single-phase flow to two-phase flow, and thus allowing for handling of reservoirs with strong aquifers and waterflooding problems. 2) redesigned deep network neural network architecture for better performance. 3) extended to handle operational constraints such as maximum liquid production, drilling duration and water-cut limit. 4) extended to three-dimensional reservoir model.

This paper proceeds as follows. In **Section 2**, we present the governing equations for the two-phase flow and discuss the different field development optimization approaches–traditional and reinforcement learning-based field development optimization. In **Section 3**, we present the deep reinforcement learning field development approach which includes the state and action representation, the proximal policy algorithm and the deep neural network used to represent the policy and value functions of the agent. Computational results demonstrating the performance of the deep reinforcement learning agents, for both 2D and 3D problems, are presented in **Section 4**. We conclude in **Section 5** with a summary and suggestions for future work.

2 GOVERNING EQUATIONS AND FIELD DEVELOPMENT OPTIMIZATION APPROACHES

In this section, we first briefly discuss the governing equations for the two-phase flow. We then describe the traditional and reinforcement learning-based field development optimization approaches.

2.1 Governing Equations

In this work, we consider the isothermal immiscible oil-water flow problem with gravitational effects. Combining Darcy's law for multiphase flow and the mass conservation equation while neglecting capillary pressure, the flow of each phase in the reservoir can be described using:

$$\nabla \cdot \left[\mathbf{k}\rho_l\lambda_l\left(\nabla p - \gamma_l \nabla D\right)\right] = \frac{\partial}{\partial t}\left(\phi\rho_l S_l\right) + q_l, \tag{1}$$

where the subscript l represents the phase ($l = o$ for oil and $l = w$ for water), $\mathbf{k}$ is the permeability tensor, ρ_l is the phase density, $\lambda_l = k_{r,l}/\mu_l$ is the phase mobility, with $k_{r,l}$ the phase relative permeability and μ_l the phase viscosity, p is the pressure (with $p = p_o = p_w$ since capillary pressure is neglected), $\gamma_l = \rho_l g$ is the phase specific weight, with g the gravitational acceleration, D is the depth, ϕ is the porosity, S_l is the phase saturation and q_l is the mass sink term. The phase flow rate for a production well w in well-block i is defined by the Peaceman well model [29]:

$$\left(q_l^w\right)_i = WI_i\left(\lambda_l\rho_l\right)_i\left(p_i - p^w\right), \tag{2}$$

where WI_i is the well-block well index which is a function of the well radius, well block geometry and permeability, p_i denotes the well block pressure and p^w denotes the well bottomhole pressure.

The discretized form of **Eq. (1)** given in **Eq. (3)** is solved for each grid block i, in time, using the fully implicit method.

$$\left[\frac{(c_t)_l\phi V S_l}{B_l}\right]_i \frac{p_i^{n+1} - p_i^n}{\Delta t} = \sum_k \Gamma_l^k \Delta\Psi_l^{k,n+1} + (q_l)_i, \tag{3}$$

where p_i the pressure of grid block i, n and $n + 1$ indicate the time levels, V is the volume of the grid block, $(c_t)_l$ denotes the total phase compressibility of the grid block, B_l is the phase formation volume factor of the grid block, k represents an interface connected to grid block i, $\Delta\Psi_l^k$ is the difference in phase potential over the interface k, $\Gamma_l^k = \Gamma^k\lambda_l^k$ is the phase transmissibility over the interface k, with Γ^k the rock transmissibility which is a function of the permeability and geometry of the grid blocks connected by interface k.

2.2 Optimization Problem Formulation and Traditional Approaches

The aim in the field development optimization problem in this work is to determine the number of wells to drill in a greenfield, alongside their locations and drilling sequence. Only the primary depletion mechanism is considered, thus only production wells are drilled. The constraints imposed on the field development include the maximum liquid production rate, drilling duration for each well, the allowable water cut limit, and minimum inter-well spacing.

Mathematically, the field development optimization problem can be written as follows:

$$\max_{\mathbf{x}\in\mathbb{X}} \quad J(\mathbf{x}), \quad \text{subject to } \mathbf{c}(\mathbf{x}) \le \mathbf{0}, \tag{4}$$

where J is the objective function to be optimized, the decision vector $\mathbf{x} \in \mathbb{X}$ defines the number of wells to drill, their locations, and drilling sequence. The space $\mathbb{X}$ defines the feasible region (upper and lower bounds) for the decision variables. The vector $\mathbf{c}$ defines optimization constraints that should be satisfied. Interested readers can refer to Isebor et al. [1,2], for possible ways of parameterizing the decision vector $\mathbf{x}$ for the field development optimization problem.

The net present value (NPV) is typically considered as the objective function to be maximized in the field development optimization. Thus, J in **Eq. (4)** can be specified as the NPV. The NPV for primary depletion (only production wells) can be computed as follows:

$$\mathrm{NPV}(\mathbf{x}) = \sum_{n=1}^{N_t} \gamma^{t_n} \left[\Delta t_n \sum_{i=1}^{N_w} \left(\left(p_o - c_{opex} \right) q_{o,n}^i - c_{pw}\, q_{w,n}^i \right) - I_n\, c_w \right], \tag{5}$$

Here N_t is the number of time steps in the flow simulation, N_w is the number of production wells, t_n and Δt_n are the normalized time and time step size at time step n, and p_o, c_{opex} and c_{pw} represent the oil price, operating cost and the cost of produced water, respectively. The variables c_w and γ represent the well drilling cost and annual discount rate, respectively. The indicator function I_n is one if a well is drilled at time step n and zero otherwise. The rates of oil and water production for well i at time step n are, respectively, $q_{o,n}^i$ and $q_{w,n}^i$.

In the traditional approach of solving the field development optimization problem, the field development scenario of interest is first defined. These include the reservoir model definition (structural, rock, and fluid properties), the specific constraints and the economic condition to be considered. Afterward, the optimization variables that define the number of well, location, and drilling sequence are parameterized to obtain a formulation of $\mathbf{x}$. Finally, an optimizer (e.g. particle swarm optimization or genetic algorithm) iteratively proposes sets of decision variables (that represent field development plans and are evaluated with a reservoir simulator) in order to maximize an economic metric of interest. The optimized set of decision variables is tied to a particular field development scenario predefined before the flow simulation.

In the traditional approach, the evaluation of **Eq. (5)** for each field development plan requires performing the two-phase flow simulation which involves solving **Eq. (3)**. From those simulations, the only information used by the traditional optimization approach is the corresponding NPV. The reservoir states at each time step, which contain a lot of useful information, are discarded.

Different from the traditional optimization approach, Reinforcement learning makes use of the intermediate states generated from the simulator and provides optimized policy applicable to a range of different scenarios.

2.3 Reinforcement Learning-Based Field Development Optimization

Reinforcement learning is a sub-field of machine learning concerned with teaching an artificial intelligence agent how to make decisions so as to maximize the expected cumulative reward. The training and the decision making process of the AI agent follows a feedback paradigm as shown in **Figure 1**.

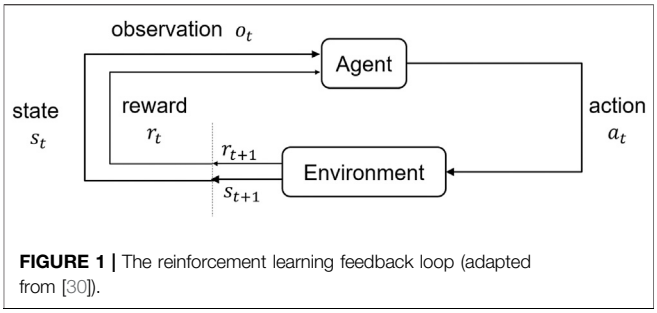

FIGURE 1 | The reinforcement learning feedback loop (adapted from [30]).

The reinforcement learning problem can be expressed as a system consisting of an agent and an environment. The agent and the environment interact and exchange signals. These signals are utilized by the agent to maximize a given objective. The exchanged signals, also referred to as experience, are (s_t, a_t, r_t), which denotes the state, action, and reward respectively. Here, t defines the decision stage in which the experience occurred. At a given stage t of the decision-making process, with the environment (which represents where the action is taken) at state s_t, the agent takes an action or decision a_t. The quality of the action a_t is quantified and signaled to the agent through the reward r_t. The environment then transits to a new state s_{t+1} which indicates the effect of action a_t on the environment. The transition from s_t to s_{t+1} in reinforcement learning is formulated as a Markov decision process (MDP)—which assumes the transition to the next state s_{t+1} only depends on the previous state s_t and the current action a_t. This assumption is referred to as the Markov property [31]. The feedback loop terminates at a terminal state or a maximum time step $t = T$. The time horizon from $t = 0$ to when the environment terminates is referred to as an episode.

The choice of the action to take at stage t by the agent depends on the observation o_t which is a function of s_t. If o_t contains all the information in s_t, then the decision making process is referred to as fully observable MDP. Otherwise, it is referred to a partially observable MDP (POMDP). The agent's action-producing function is referred to as a policy. Given a state s_t (relayed to the agent through o_t), the policy produces the action a_t. This mapping operation is mathematically represented as $\pi(a_t \mid o_t, \theta)$, where θ are the parameters that define the policy function. The reinforcement learning problem is now essentially an optimization problem with the goal of finding the optimal policy parameters (θ^{opt}) that maximizes an expected cumulative reward. This optimization problem is posed as:

$$\theta^{opt} = \arg\max_{\theta} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t \right], \tag{6}$$

where the reward at time t, $r_t = \mathbb{R}(s_t, a_t, s_{t+1})$ depends on the reward function $\mathbb{R}$ and $\gamma \in (0, 1)$ is the discount factor that accounts for the temporal value of rewards. The expectation accounts for the stochasticity that may exist in the action and

environment. If the parameters of the policy function are the weights of a deep neural network, then the optimization problem is referred to as a deep reinforcement learning (described in details later).

In the context of field development optimization, our goal is to determine the policy function that maximizes the expected NPV. At each drilling stage, the agent defines the action to be taken, which includes whether to drill a producer or not and, if drill, the optimal well location. The environment, which is the reservoir simulator, advances the flow simulation to the next drilling stage by solving the governing equation for the immiscible oil-water system given in **Eq. (1)**. The discounted reward function $\gamma^t \mathbb{R}$ for the reinforcement learning-based field development optimization problem is given by the NPV (**Eq. (5)**), where the discounted reward ($\gamma^t r_t$) for a drilling stage t is given by $\text{NPV}_t = \text{NPV}(s_t, a_t, s_{t+1})$. Here, $\text{NPV}(s_t, a_t, s_{t+1})$ defines the NPV obtained if the flow simulation starts from state s_t (including all wells drilled from s_0 to s_t) and ends at state s_{t+1}.

The formulation of the RL-based field development optimization in this work allows the exploitation of the problem states (s_t) generated during the simulation and is thus a more efficient use of the information from the potentially expensive simulation runs. In addition, the output of the RL-based approach is not the optimal actions itself, but the optimal policy, a mapping from the states to the optimal actions $\pi(a_t|o_t, \theta)$, which can be used obtain the optimal actions under various different scenarios. This is distinctively different from the use of RL in Ma et al. [27]; Miftakhov et al. [28], where the goal was the optimal actions rather than the policy.

3 DEEP REINFORCEMENT LEARNING FOR FIELD DEVELOPMENT OPTIMIZATION

In this section, we present the deep reinforcement learning approach for field development optimization where the parameters (θ) of the policy function are defined by the weight of a convolutional neural network. We first describe the action and state representation for the field development optimization problem. Finally, the training procedure of the agent using the proximal policy optimization algorithm [32] and the convolutional neural network architecture are described.

3.1 Action Representation

The action representation is important because it affects the computational complexity of the learning process. At each drilling stage, the agent needs to decide to drill a well at a certain location or not drill at all. We introduce two variables to make these decisions. The drill or do not drill decision at drilling stage t is represented by a binary categorical variable $w_t \in \{0, 1\}$ where 0 represents the decision not to drill a well and 1 the decision to drill. The second variable (active only when the agent decides to drill a well, i.e. $w_t = 1$) defines where the well should be drilled and is represented by the well location variable u_t, which is chosen from the possible drilling locations denoted by $\mathbf{u} \in \mathbb{Z}^{N_x N_y}$ (the 2D grid flattened into 1D), where N_x and N_y represents the number of grid blocks in the areal x and y directions of the reservoir model. Each index in $\mathbf{u}$ maps to a grid block in the reservoir model. Thus, at each drilling stage t, the action a_t is defined by $a_t = [w_t, u_t]$, where u_t is only considered if $w_t = 1$.

The action representation used in this work differs from that used in He et al. [3]. Specifically, in He et al. [3] a single variable of dimension $N_x \times N_y + 1$ is used to define the possible actions at each drilling stage, where the additional index represents the do-not-drill decision. Such an action parameterization means the action a_t at each drilling stage is sampled from a single probability distribution of cardinality $N_x \times N_y + 1$. Thus, the do-not-drill decision depends on the number of possible well locations ($N_x \times N_y$). As the size of the reservoir model increases, it becomes increasingly difficult for the agent to learn when not to drill a well due to the cardinality of the probability distribution dominated by possible drilling locations. This is in contrast to the approach used in this work where the variables w_t and u_t are sampled from two different probability distributions of cardinality 2 and $N_x \times N_y$, respectively. Hence, the drill or do-not-drill action is independent of the size of the reservoir model. The parameterization used in this work can be naturally extended to other field development cases. For example, for waterflooding, the decision on the well type can be incorporated in w_t as $w_t \in \{-1, 0, 1\}$, where -1 represents the decision to drill an injector, 0 to not drill a well, and 1 to drill a producer.

Given the vector $\mathbf{u}$, drilling in some grid blocks may lead to an unacceptable field development plan from an engineering standpoint. These include field development plans where more than one well is to be drilled in the same location, violation of minimum inter-well distance constraint (acceptable spacing between wells) and/or drilling in inactive regions of the reservoir. At any given drilling stage, an action mask of the same dimension with $\mathbf{u}$ defines acceptable drilling locations that the agent could choose from. Specifically, the action mask contain binary values (0 or 1) and the probability of sampling an invalid drilling location (where the values of the action mask are 0) in $\mathbf{u}$ is set to zero. This action masking technique [33,34] have been proven to be an effective strategy to improve the convergence of the policy optimization and also ensures the agent only takes valid action. The action masking technique was used in He et al. [3] for the field development optimization problem to ensure only feasible drilling locations are proposed by the agent.

3.2 State Representation

The definition of the state of the environment for the field development optimization problem depends on the model physics. For example, the state would include pressure for single-phase flow and, additionally, saturation for two-phase flow. The definition of the state also depends on the kind of generalization capability that we want the AI to have. For example, if we want the AI to provide different optimal solutions for different geological models, geological structures and properties such as permeability, porosity should also be included in the state definition. If we want the AI to be able to provide different optimal solutions under different oil prices, the oil price should also be part of the state.

Our goal in this work is to develop an AI for two-phase flow that can provide optimal solutions for different field development optimization scenarios with variable reservoir models,

TABLE 1 | Distribution of parameters for 2D and 3D oil-water system field development optimization. $U(a, b)$ denotes uniform distribution over the range of (a, b). $\{a_1, a_2, \ldots, a_n\}$ indicate uniform probability distribution over the n discrete options.

Number	Variable	Symbol	Distribution
1	Grid size in x-direction (ft)	dx	U (500, 700)
2	Grid size in y-direction (ft)	dy	U (500, 700)
3	Grid thickness (ft)	dz	SGS 1000, 100, 30, 60)
4	Variogram azimuth (^{0})	ang	U (0, 90)
5	Variogram structure	$struct$	(Gaussian, exponential)
6	Porosity	ϕ	SGS [U (0.15, 0.25),U (0.01, 0.07), 2, 8]
7	Permeability (md)	k	Cloud transform from ϕ
8	Vertical to horizontal permeability ratio (3D)	$kvkh$	logUnif (0.001, 0.1)
9	Active cell indicator	$active$	Random elliptical
10	Datum depth (ft)	d_{datum}	U (5,000, 36,000)
11	Depth from datum (ft)	D	SGS (1000, 1000, 30, 60)
12	Pressure gradient (psi/ft)	p_{grad}	U (0.7, 1)
13	Initial water saturation	S_{winit}	Cloud transform from ϕ
14	Oil references formation volume factor	$B_{o,ref}$	U (1, 1.5)
15	Water references formation volume factor	$B_{w,ref}$	U (1, 1.5)
16	Oil compressibility (psi^{-1})	c_o	U (1e-6, 4e-6)
17	Water compressibility (psi^{-1})	c_w	U (2e-6, 5e-6)
18	Oil specific gravity	γ_o	U (0.8, 1)
19	Water density (lbs/ft^3)	ρ_w	U (62, 68)
20	Oil references viscosity (cp)	$\mu_{o,ref}$	U (2, 15)
21	Water references viscosity (cp)	$\mu_{w,ref}$	U (0.5, 1)
22	Residual oil saturation	S_{or}	U (0.05, 0.2)
23	Connate water saturation	S_{wc}	U (0.05, 0.2)
24	Oil relative permeability endpoint	k_{roe}	U (0.75, 0.95)
25	Water relative permeability endpoint	k_{rwe}	U (0.75, 0.95)
26	Corey oil exponent	n_o	U (2, 4)
27	Corey water exponent	n_w	U (2, 4)
28	Producer BHP (% pressure at datum)	p^w	U (0.3, 0.7)
29	Producer skin	s	U (0, 2)
30	Oil price per bbl ($)	p_o	U (40, 60)
31	Water production cost (% oil price)	c_{wp}	U (0, 0.02)
32	Operating cost per bbl ($)	c_{opex}	U (8, 15)
33	Drilling cost per well ($)	c_{capex}	U (1e8, 5e8)
34	Discount rate (%)	γ	U (89, 91)
35	Drilling time per well (days)	d_{time}	(90, 120, 180, 240)
36	Project duration (years)	p_{time}	U (15, 20)
37	Watercut limit (%)	wc_{limit}	U (60, 98)
38	Max. well liquid production rate (bbl/day)	$q_{l,max}$	U (1e4, 2.5e4)

operational constraints, and economic conditions within a predefined range. Thus, this variation in the scenarios should be captured in the state representation.

In this work, the field development optimization scenarios considered are characterized by parameters following the distributions listed in **Table 1**. Each scenario defines a given geological structure, rock and fluid properties, operational constraints, and economic conditions. This includes the grid size, the spatial distribution of the grid thickness obtained through Sequential Gaussian simulation (SGS) with a fixed mean, standard deviation, and variogram ranges. The porosity fields which are also generated using SGS have a variable variogram structure and azimuth. A cloud transform of the porosity field is used to generate the permeability and initial saturation fields. Although the DRL framework can handle temporally-varying economic parameters, they remain fixed for each field development scenario in this work and thus do not vary in time. After appropriate training, the AI agent is expected to provide an optimized field development plan for an optimization scenario randomly sampled from this defined range of parameters. The generalization and applicability of the resulting AI agent will depend on the optimization problem parameters and their distributions. The number of learnable parameters of the neural network and computational complexity of the training process of the AI increases as more parameters and larger distributions are considered.

As pointed out in He et al. [3], the parameters listed in **Table 1** do no affect the model independently. Using them directly as input to the deep neural network increase the complexity and the ill-conditioning of the problem. Therefore, we define the state of the environment using parameter groups that aid in reducing the number of input channels provided to the artificial intelligence agent. These input channels include parameter groups that appear in the discretized form of the oil-water immiscible flow problem (**Eq. 3**) and the reward. Additional input channels are also included to define the operational constraints imposed on the field development. The constraints considered include the drilling duration per well, watercut limit and the maximum well liquid

TABLE 2 | List of the input channels used to define the state for 2D and 3D oil-water system field development optimization.

Number	Input channel	Static/Dynamic
1	Pressure	Dynamic
2	Water saturation	Dynamic
3	x-directional transmissibility	Static
4	y-directional transmissibility	Static
5	z-directional transmissibility (3D)	Static
6	Oil accumulation	Dynamic
7	Water mobility	Dynamic
8	Oil mobility	Dynamic
9	Well index	Static
10	Producer drawdown	Dynamic
11	Well location mask	Dynamic
12	Well cost to net oil price ratio	Static
13	Water production cost	Static
14	Current discount rate	Dynamic
15	Remaining production time	Dynamic
16	Max. liquid production rate	Static
17	Drilling time	Static
18	Water cut constraint	Static

production rate. The list of input channels used to define the state of the environment are given in **Table 2**.

There are 17 input channels considered for the 2D subsurface system while 18 (including the z-directional tramsmissibility) are used for the 3D system. The observation at any given drilling stage (o_t) is represented by $o_t \in \mathbb{R}^{N_x \times N_y \times N_c}$, where N_c defines the number of channels. For the 2D case $N_c = 17$, where each scalar property (such as the drilling time or water cut constraint) is converted to a 2D map of constant value. Out of the 18 input channels for the 3D system, 10 (e.g. pressure, saturation, well index, e. t.c) vary spatially and thus the values for each layer of the reservoir model is included in the input channel. This means in the 3D case, $N_c = 10N_z + 8$, where N_z is the number of layers in the 3D reservoir model.

Once an optimization scenario is defined during training, the static input channels are computed and saved. After advancing the flow simulation to any given drilling stage, the dynamic properties are extracted and stacked with the static properties. This stacked input channels are then provided to the agent as the state of the environment at that given drilling stage. In order to improve the efficiency of the training process, all the input channels (except for the well location mask that is zero everywhere and one only at regions where wells are located) are normalized to be very close to 0 to one range. Due to the highly skewed distribution of the transmissibilities and well index, they are scaled with a nonlinear scaling function as done in He et al. [3]. Other input channels are scaled using the linear min-max scaling function.

3.3 Proximal Policy Optimization

Our goal is to determine the learnable parameters (defined by the weights of a neural network) of the policy function that maximizes the expected cumulative reward as posed in the optimization problem given in **Eq. (6)**. In this work, we use a policy gradient method in which the expected cumulative reward is maximized by performing gradient descent on the parameters of the policy. The vanilla policy gradient method is susceptible to performance deterioration due to the instability that may be introduced by the update step during gradient descent. Several variants of the policy gradient method have been proposed to improve the stability of the optimization. Some variants of the policy gradient method include the trust region policy optimization (TRPO) [35] and proximal policy optimization (PPO) [32].

The PPO algorithm proposes a surrogate objective that improves the policy optimization by guaranteeing monotonic policy improvement. Following He et al. [3], we now briefly describe the implementation of PPO used in this work.

The PPO loss or objective function given in **Eq. (7)** contains four components: the surrogate policy loss (L^{π}), the Kullback–Leibler (KL) divergence penalty (L^{kl}), the value function loss (L^{vf}), and the entropy penalty (L^{ent}).

$$L^{PPO} = L^{\pi} + c_{kl}L^{kl} + c_{vf}L^{vf} + c_{ent}L^{ent} \tag{7}$$

where c_{kl}, c_{vf} and c_{ent} are user-defined weighting factors for the KL divergence, value function and entropy terms, respectively.

The surrogate policy loss (L^{π}) that directly maximizes the expected cumulative reward is given by:

$$L^{\pi} = \mathbb{E}_t\left[\min\left(r_t(\theta)A_t, \text{clip}\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right)A_t\right)\right] \tag{8}$$

where $r_t(\theta)$ quantifies the policy change and it is defined as the ratio of the old policy $\pi(a_t|s_t, \theta_{old})$ to the new policy $\pi(a_t|s_t, \theta)$. The term $clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)$ removes the incentive to change the policy beyond a pre-defined step size limit ε. This prevents large updates of the parameters that could lead to deterioration of the policy. The advantage function A_t defines how good an action is for a specific state relative to a baseline. Following the generalized advantage estimation (GAE) framework proposed by Schulman et al. [36], A_t is defined by:

$$A_t^{\text{GAE}(\gamma,\lambda)}(t) = \sum_{l=t}^{T}(\gamma\lambda)^{l-t}\left(r_l + \gamma V^{\pi}(s_{l+1}) - V^{\pi}(s_l)\right) \tag{9}$$

where γ and λ are hyperparameters that control the bias and variance introduced by the various terms in the summation. The value function $V^{\pi}(s_t)$, given in **Eq. (10)**, represents the expected total reward of being in state s_t and then following the current policy π. The value function is parameterized by the weights of a neural network ψ.

$$V^{\pi}(s_t) = \mathbb{E}\left[\sum_{l=t}^{T} r_l\right] \tag{10}$$

The value function is learned by minimizing the value-function loss L^{vf} which is given by:

$$L^{vf} = \mathbb{E}_t\Big[\max\Big(\left(V_{\psi}(s_t) - V_{target}(s_t)\right)^2, \left(V_{\psi_{old}} + \text{clip}\left(V_{\psi}(s_t)\right.\right.$$
$$\left.\left. - V_{\psi_{old}}(s_t), -\eta, \eta\right) - V_{target}(s_t)\right)^2\Big)\Big], \tag{11}$$

where the $V_{target}(s_t)$ is the computed value function from the training samples, $V_{\psi_{old}}$ is the value function using the current

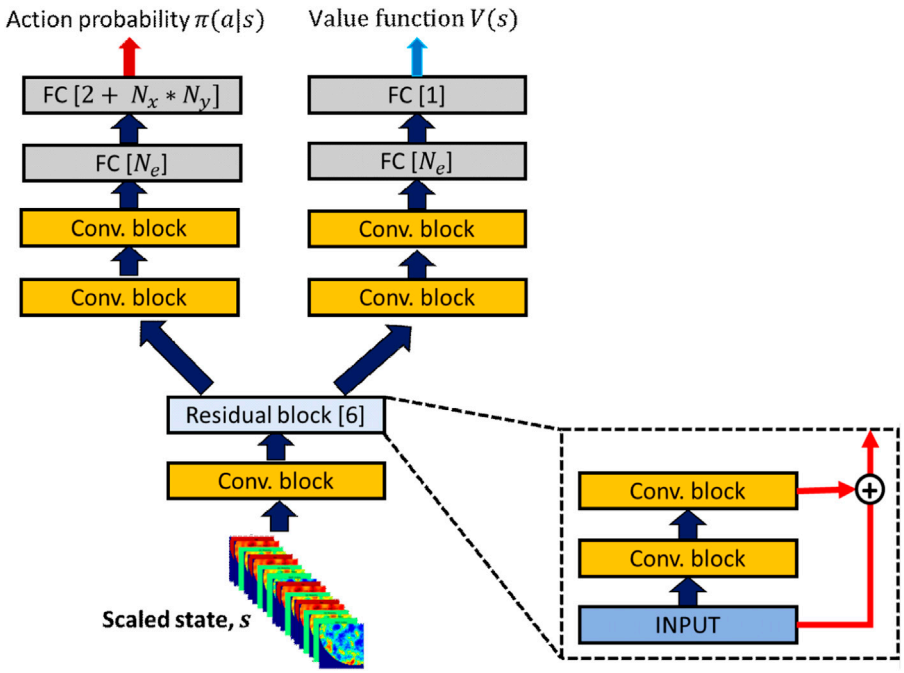

FIGURE 2 | The neural network architecture that defines the policy and value functions. The "conv" block represents a convolutional neural network (CNN) layer followed by rectified linear unit (ReLU). FC (N_e) represents a fully connected layer with N_e neurons.

parameters of the value function ψ_{old}. **Eq. (11)** essentially tries to minimize the mismatch between the computed and predicted value of the sampled states. The hyperparameter η has same effect as ε in **Eq. 8** and is used to ensure large updates of the value parameters ψ are not allowed.

The KL divergence penalty (L^{kl}) serves as an additional term to avoid large policy update and it is given by:

$$L^{KL} = \mathbb{E}_t\left[D_{KL}\left(\pi\left(a_t|s_t,\theta\right)|\pi\left(a_t|s_t,\theta_{old}\right)\right)\right], \tag{12}$$

where $D_{KL}\ [\pi(a_t|s_t,\ \theta)|\pi(a_t|s_t,\ \theta_{old})]$ is the Kullback–Leibler divergence [37] that measures the difference between the old and new policy.

3.4 Policy and Value Function Representation

We now describe the representation of the policy function $\pi\ (a|s,\ \theta)$ and the value function $V^{\pi}\ (s_t)$ with a deep neural network. In our previous work [3], the policy and value networks are represented by different neural networks. Thus the parameters θ and ψ are independent. In this work, we use the approach in AlphaGo Zero [38] where the value and policy networks share some layers. In this case, there is an overlap in the parameters θ and ψ. The neural network architecture used in this work is shown in **Figure 2**. The shared layers in the neural network are used to learn features from the state that are relevant to both the policy and the value network. This reduces the computational cost that will otherwise be expensive during training if the policy and value network are represented by different neural networks.

Given the scaled state at any given drilling stage, the shared layers process the state through a series of convolutional operation. The shared layer comprises a "conv" block and six residual blocks [39]. The "conv" block is essentially a convolutional neural network (CNN) layer [40] followed by a rectified linear unit (ReLU) activation function [41]. Interested readers may refer to He et al. [3] for a brief description of CNN layers and ReLU activation functions. Each residual block contains two "conv" blocks as shown in **Figure 2**. The convolutional operations in the shared layers are performed with 64 kernels of size 3 × 3 for the two-dimensional subsurface system. The agent for the three-dimensional system, however, uses 128 kernels for the "conv" blocks in the shared layers. This is primarily because the size of the input channels in the three-dimensional subsurface system is larger than that of the two-dimensional case.

The learned features from the shared layers serves as input to the policy and value arms of the network. These features are further processed with two "conv" blocks in the individual arms. The first and second conv blocks consist of 128 and two kernels, respectively, of size 1 × 1. The high dimensional output after the convolutional operations are reduced in dimension with an embedding layer [3,42] which consist of a fully connected (FC) layer with N_e units. Here, N_e is set to 50. The learned embeddings from the policy arm of the network serves as input to an additional fully connected layer which predicts the probability of all actions. The embeddings from the value arm, however, predicts the value of the state which is a scalar quantity.

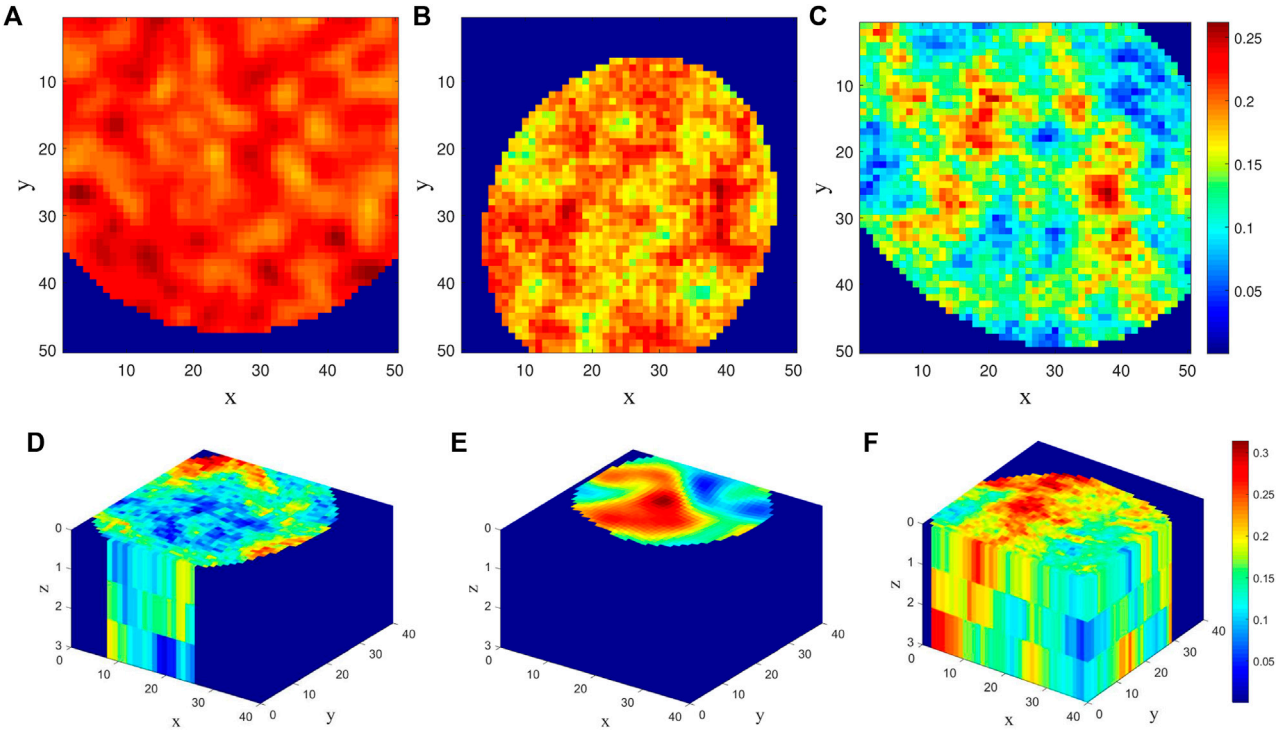

FIGURE 3 | Porosity fields for random training scenarios of the 2D and 3D models sampled from the parameter distributions given in **Table 1**.

4 COMPUTATIONAL RESULTS

In this section, we evaluate the performance of the artificial intelligence agents for field development optimization in two- and three-dimensional subsurface systems. The 2D reservoir model is of dimension 50 × 50, while that of the 3D case is 40 × 40 × 3. A maximum of 20 production wells are considered for the field development with one well drilled per drilling stage. The number of drilling stages for each specific field development scenario depends on the total production period and drilling duration sampled from **Table 1**. Wells operate at the sampled bottom-hole pressures unless a maximum (sampled) water cut is reached, at which point the well is shut in. Porosity fields for three random training scenarios of the 2D and 3D models are shown in **Figures 3A–C** and **Figure 3D–F**, respectively. Note that other variables (such as the constraints, economics and fluid properties) for these training scenarios are, in general, different.

The simulations in both cases are performed using Delft Advanced Research Terra Simulator (DARTS) with operator-based linearization [43,44]. The simulator has a python interface that allows for easy coupling with the deep reinforcement learning framework. The light weight nature of the simulator makes it suitable for running millions of simulations with minimal overhead. The overhead is significantly reduced due to the absence of redundant input/output processing. In our implementation, the simulator was further extended to handle dynamic addition of wells between drilling stages.

The training process utilizes 151 CPU cores for the 2D case and 239 CPU cores for the 3D case. The CPU cores are used to run the simulations in parallel. The training data generated are then passed to four GPU cores used for training the deep neural network. The weighting factors for the terms in the loss function are defined based on those proposed in He et al. [3]. Accordingly, c_{kl}, c_{vf} and c_{ent} in **Eq. (7)** are set to 0.2, 0.1, 0.001. A linear learning rate decay schedule is used for the gradient descent with an initial learning rate of $1e^{-3}$ and a final learning rate of $5e^{-6}$ (at 15 million training samples). It should be noted that each combination of a_t, s_t and r_t generated in each drilling stage represents a single training sample. A mini-batch size of 256 and 5 epochs are used for the gradient descent.

For the 2D case, we first compare the performance of the action parameterization used in our previous work He et al. [3], with the one proposed in this work. We then benchmark the performance of the 2D artificial intelligence agent with well-pattern drilling agents. The 3D artificial intelligence agent is also compared with the well-pattern drilling agents.

4.1 Case 1: Two-Dimensional System

We now present results for the 2D case. **Figure 4** shows the evolution of some performance indicators for the training process in which the dual-action probability distribution is used. At each of the training iteration, each of the 151 CPUs runs a maximum of two simulations. The set of training data generated (used to train the agent in that specific iteration) are used to compute the performance indicators reported. The training of the agent involves approximately two million simulations, which corresponds to approximately 7,100 equivalent simulations or training iterations.

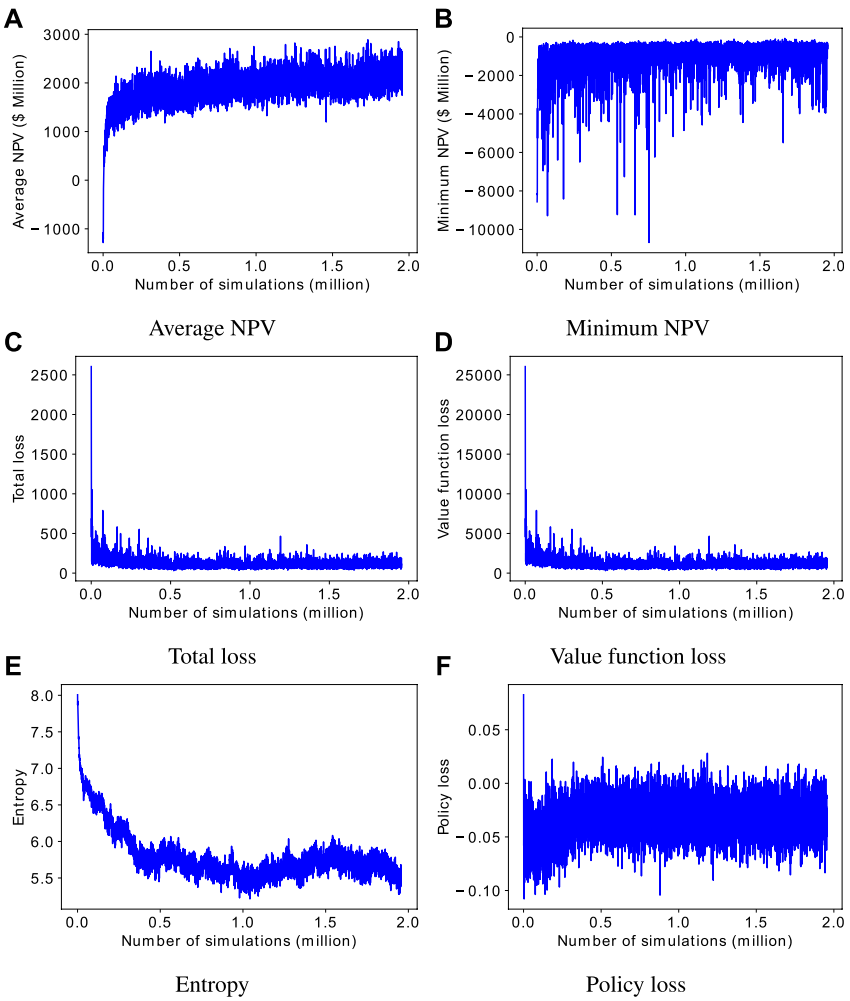

FIGURE 4 | Evolution of training performance metrics for the dual-action probability distributions for the 2D case.

From **Figure 4A**, it is evident that the average NPV of the field development scenarios, in general, increases as the training progresses. Starting from a random policy (randomly initiated weights of the deep neural network) which results in negative average NPV, the average NPV increased to more than $2 billion. The fluctuation in the average NPV is due to the fact that the ease of developing the various field development scenarios, which are randomly generated, varies from iteration to another.

Figure 4B shows the minimum NPV of the field development scenarios generated at each iteration. Theoretically, the agent should have a minimum NPV of zero as it could choose not to drill any well. However, it should be noted that during the generation of the training data, the action to be taken is sampled from the action distribution of the policy. While this aids in exploring the action space (the entropy loss also encourages exploration) and improves the training performance, it also means the policy is not strictly followed during training. This leads to some fluctuation in the minimum NPV in **Figure 4B**, but overall the minimum NPV approaches zero. During the application of the agent (after training), the action with the maximum probability is taken. Results for the case in which the policy is followed strictly are presented later.

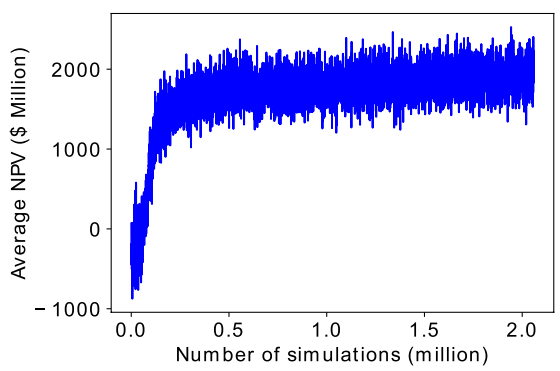

FIGURE 5 | Evolution of the average NPV using the single-action probability distribution for the 2D case.

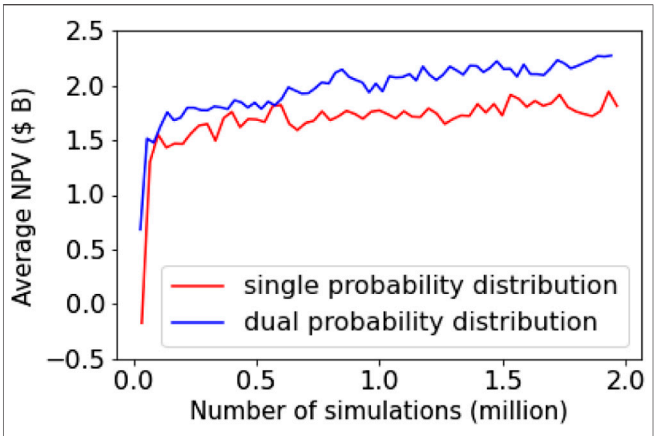

FIGURE 6 | Evolution of the average NPV for the single and dual-action probability distributions.

The PPO total loss, given in **Eq. (7)**, is shown in **Figure 4C**. The loss, in general, decreases as the optimization progresses. The value function loss, entropy and policy loss are shown in **Figure 4D–F**, respectively. From the figures, we can see that the total loss is dominated by the value function loss. From our limited experimentation, there was no noticeable advantage in the reduction of the weighting factor for the value function loss. The entropy loss which indicates the convergence of the policy can be seen to be decreasing. It is observed that the weight for the entropy term needs to be small enough for the optimization to proceed smoothly. The policy loss, on the other hand, oscillates and does not show any clear trend. This is a common behaviour in PPO [32] because the definition of the policy loss varies from one iteration to another (due to variation in the value function which the advantage function depends on).

We now compare the performance of the single-action probability distribution used in He et al. [3] with the dual-action probability distribution. **Figure 5** shows the evolution of the average NPV for the single-action probability distribution. In general, the average NPV increases as the training progresses. However, when compared to **Figure 4A**, the use of a single-action probability distribution leads to a slow learning in the initial training phase. The result shown here is the best found after several trials. Depending on the initial policy, the learning could be significantly slower than the case shown here. This slow learning is mainly because the learning of the decision not to drill a well is very slow since the action probability distribution is dominated by possible drilling locations. This slow learning increases the computational cost of the training because the timing of the simulation generally increases with number of wells. The timing for the first 100,000 simulations when the single-action probability distribution is used is 11.8 h. This is in contrast to 7.6 h for the dual-action probability distribution case, leading to a computational cost saving of approximately 4 h for the initial 100,000 simulations.

As noted earlier, during training, there is an exploratory aspect to the agent's policy and the field development scenarios are randomly generated. For this reason, we generate 150 random field descriptions, fluid properties, economics and constraints

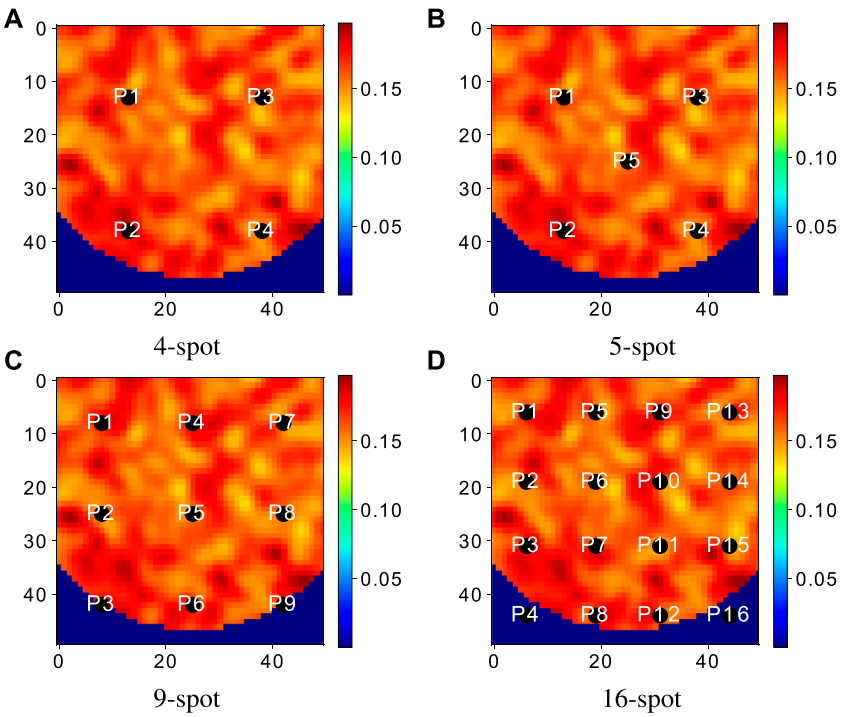

FIGURE 7 | Illustration of the 4-, 5-, 9- and 16-spot well pattern with the background as the porosity field of a specific test scenario.

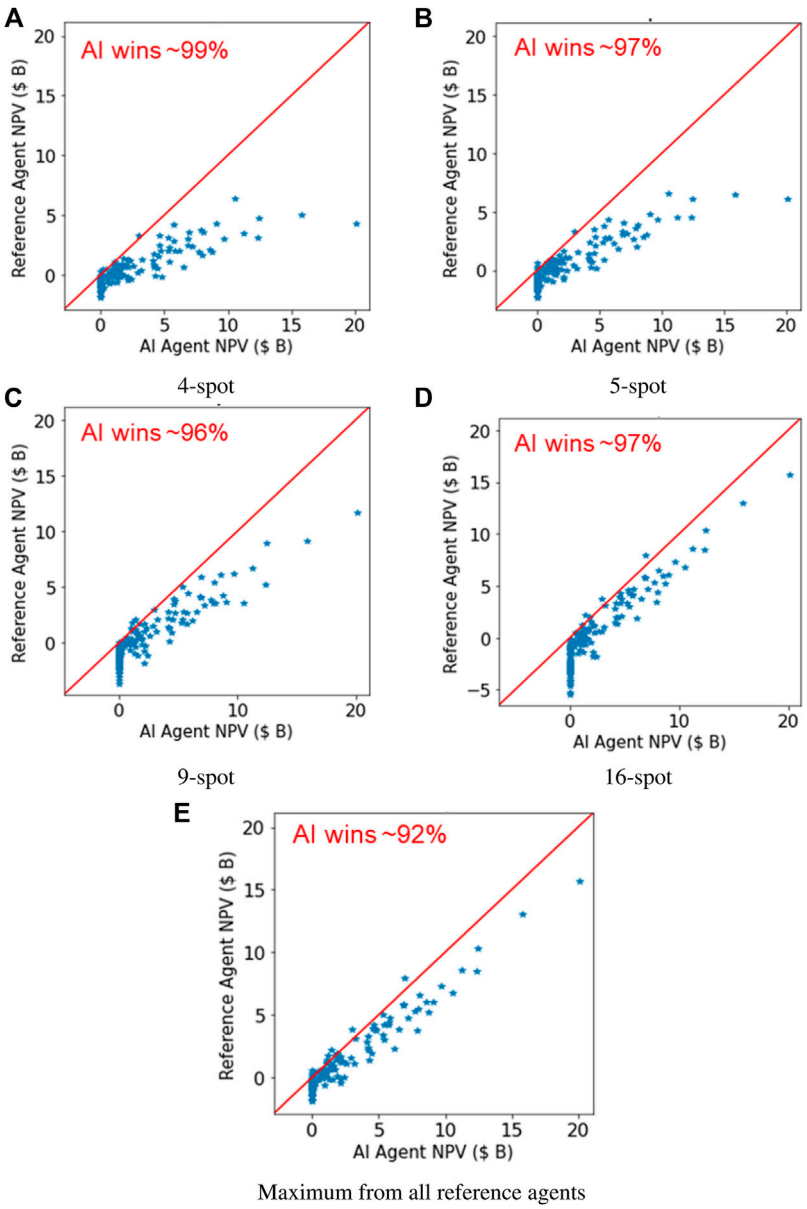

FIGURE 8 | Comparison of the best artificial intelligence (AI) agent trained using the dual-action probability representation with the reference well-pattern agents for the 2D case.

which are used for consistent comparison of the training performance of the single and dual action distributions. **Figure 6** shows the evolution of the average NPV for the 150 field development scenarios. The agent after every 100 training iteration is applied for the development of the 150 scenarios. Clearly, the use of dual-action probability distribution outperforms the single-action probability distribution.

Once the AI agent is trained, it can be used to generate optimized field development plans for any new scenarios within the range of applicability without additional simulations. Its extremely low cost in optimization for new scenarios makes it distinctively different from the traditional optimization methods, which for any new scenario would require hundreds or thousands of simulations.

The performance of the best AI agent found using the dual-action probability distribution is now compared to reference well-pattern agents. The best artificial intelligence agent is taken to be the agent with the maximum average NPV of the 150 field development scenarios previously considered. The reference well patterns considered include the 4-, 5-, 9- and 16-spot patterns which are illustrated in **Figure 7**. Note that these patterns are made up of only production wells and the wells are equally spaced. Wells that fall in the inactive region (dark-blue region) of the reservoir are not considered in the field

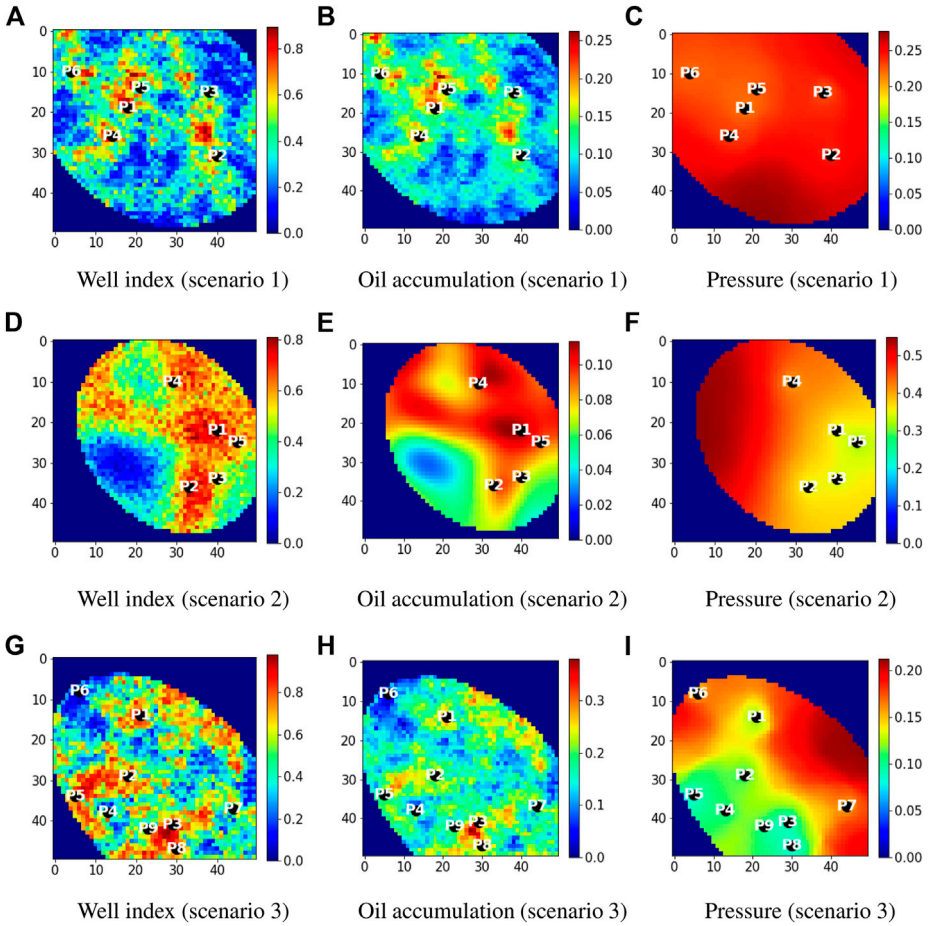

FIGURE 9 | Well configuration proposed by the best artificial intelligence agent for three random 2D field development scenarios. The well numbers indicate the drilling stage in which the wells are drilled.

development. For example, wells P3 and P9 in the 9-spot pattern (**Figure 7C**) and wells P4 and P16 in the 16-spot pattern (**Figure 7D**) are removed from the development plan.

Figures 8A–D shows the performance of the artificial intelligence agent against the 4-, 5-, 9- and 16-spot reference well patterns, respectively. In all four cases, the artificial intelligence agent outperforms the reference well patterns in at least 96% of the 150 field development scenarios considered.

Figure 8E shows the comparison of the NPV obtained from the artificial intelligence agent with the maximum NPV obtained from the four well-pattern agents for each of the 150 field development scenarios. In this case, the artificial intelligent agent outperforms the maximum from all well-pattern agents in approximately 92% of the field development scenarios. The minimum NPV obtained by the artificial intelligence agent in the 150 field development scenarios is zero NPV. For a significant proportion of these cases, the drilling of wells using the well-pattern agents leads to negative NPV, while the AI agent simply recommended not to develop the field. The results demonstrate the ability of the artificial intelligence agent to identify unfavorable field development scenarios.

We note that the solutions from the DRL AI is not guaranteed to be mathematically optimal. It also does not necessarily outperform solution from traditional optimization methods such as GA and PSO (any comparison would likely be setup and scenario dependent). The value of the DRL AI is that, once it is trained, it can be used for different fields and different scenarios, while solutions from traditional methods are tied to a specific field and scenario. In addition, a trained DRL AI generate the solution for a new field in seconds, while GA and PSO may take thousands of runs. We further note that the solution of the DRL agent could be used as an initial guess for traditional optimization algorithm to further refine the solution, which should lead to reduction in computational cost compared to the use of a random initial solution.

The positioning of wells by the artificial intelligence agent for three random cases out of the 150 field development scenarios are shown in **Figure 9**. These cases entails field development plans with six, five, and nine production wells, in scenario 1, 2, and 3, respectively. All wells in the three scenarios are drilled early. For example, in scenario 1, the six wells are drilled in the first six drilling stages with well P1 drilled in the first drilling stage and P6

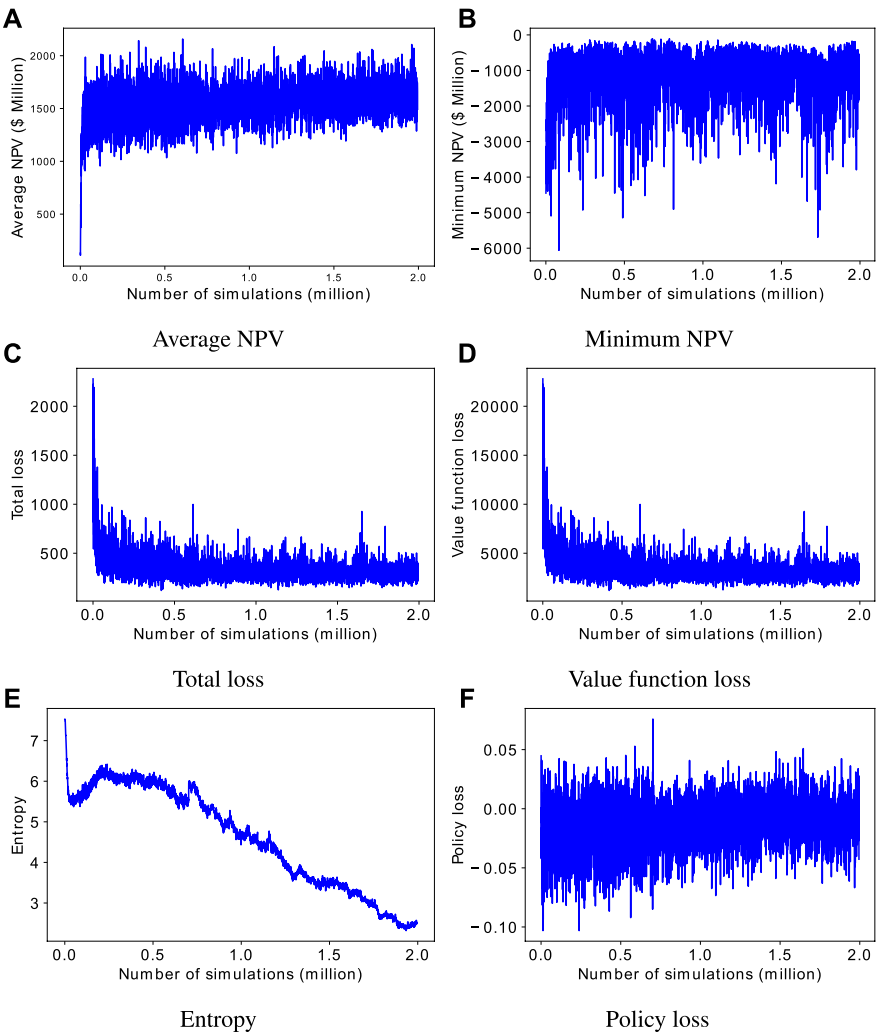

FIGURE 10 | Evolution of training performance metrics for the dual-action probability distributions for the 3D case.

in the sixth drilling stage. The wells are strategically placed in regions of the reservoir with high productivity (**Figure 9A,D,G**) and oil accumulation (**Figure 9B,E,H**). The pressure distributions for the scenarios are shown in **Figure 9C,F,I**. The wells are also properly spaced resulting in a good coverage of the producing region.

4.2 Case 2: Three-Dimensional System

Results from the training of the agent for the 3D subsurface system is now presented. For the 3D case, the wells are assumed to be always vertical and fully penetrate all layers of the reservoir in this work. The procedures discussed in this work can, however, be readily extended to horizontal wells with varying completion strategies. Compared to that of the 2D case, a larger neural network (in terms of number of learnable parameters) is used to represent the policy for the 3D case. This is because the size of the input channels to the agent is larger than that of the 2D case. The vertical heterogeneity in the reservoir models is captured by including the transmissibility in the Z-direction in the state representation.

Figure 10 shows the evolution of the performance indicators for the training process using the dual-action probability distribution for the 3D case. The training entails approximately two million simulations. From **Figure 10A** it can be seen that the average NPV generally increases as the training progresses. This demonstrates the artificial intelligence agent can learn the field development logic for the even more complicated three-dimensional system. Consistent with the behaviour of the agent for the two-dimensional case when the dual-action probability distribution is used, we see a rapid increase in average NPV during the initial phase of the training. As observed in the 2D case, the total loss (**Figure 10C**), value function loss (**Figure 10D**), and entropy loss (**Figure 10E**) decrease as the training progresses.

The performance of the best artificial intelligence agent is also compared with the four reference well-pattern agents for a set of random field development scenarios. The performance of the

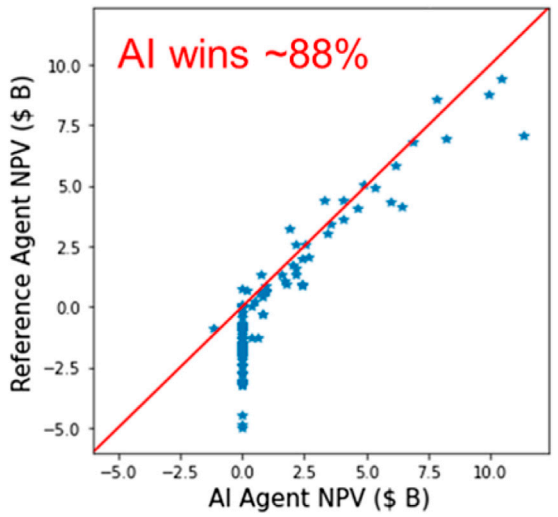

FIGURE 11 | Comparison of the best artificial intelligence (AI) agent trained using the dual-action probability representation with the maximum NPV from the reference well-pattern agents for the 3D case.

artificial intelligence agent against the maximum NPV obtained from the four well-pattern agents (same as those considered in the 2D case) is shown in **Figure 11**. The artificial intelligence agent outperforms the well-pattern agents in approximately 88% of the cases considered.

Figure 12 shows the well configurations proposed by the best AI agent for three random field development scenarios. The field development plans involves eight, four and five production wells.

5 CONCLUDING REMARKS

In this work, we developed an AI agent for constrained field development optimization for two- and three-dimensional subsurface two-phase flow models. The training of the agent utilizes the concept of deep reinforcement learning where a feedback paradigm is used to continuously improve the performance of the agent through experience. The experience includes the action or decision taken by the agent, how this action affects the state of the environment (on which the action is taken), and the corresponding reward which indicates the quality of the action. The training efficiency is enhanced using a dual-action probability distribution parameterization and a convolutional neural network architecture with shared layers for the policy and value functions of the agent. After appropriate training, the agent instantaneously provides optimized field development plan, which includes the number of wells to drill, their location, and drilling sequence for different field development scenarios within a predefined range of applicability.

Example cases involving 2D and 3D subsurface systems are used to assess the performance of the training procedure and the resulting artificial intelligence agent. The use of the dual-action probability distribution shows clear advantage over the single-action probability distribution for the training of the artificial intelligence agent. The trained artificial intelligence agents for the 2D and 3D case are shown to outperform four reference well-pattern agents. For the 2D case, the artificial intelligence agent found a better field development plan than the reference agents in approximately 92% of 150 random field development scenarios. The trained agent for the 3D case outperformed the reference well-pattern agents in approximately 88% of a set of random field development scenarios. The results demonstrated the ability of the trained agents to avoid developing unfavorable field development scenarios and strategically place wells in regions of high productivity.

In future work, the performance of other deep reinforcement learning algorithms, such as soft actor critic (SAC) [45], importance weighted actor-learner architectures (IMPALA) [46] (and PPO variant of IMPALA), that have demonstrated great sample and computational efficiency in other domains should be evaluated for the field development optimization problem. Although the use of distributed computing led to reduction of the computational cost of the training from millions of simulations to an order of thousands of equivalent simulations, the use of surrogate models should be investigated to further reduce the computational cost. The proposed procedure should also be extended to the joint

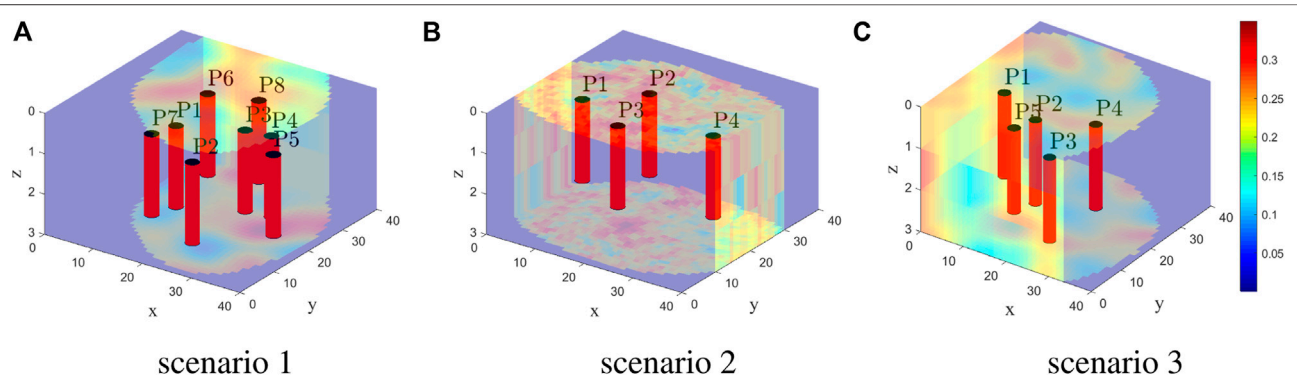

FIGURE 12 | Well configuration proposed by the best artificial intelligence agent for three random 3D field development scenarios.

optimization of well locations and operational settings. This would require an additional variable in the action to represent the operation settings of each well. Finally, the deep reinforcement learning framework should be extended to other field development cases such as waterflooding and optimization under uncertainty where multiple realizations of the geological model are used to represent geological uncertainty.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

YN: Idea, implementation and paper drafting JH: Idea and paper writing CH: High performance computing infrastruture ST: Simulation result KW: Idea and testing XW: Project guidance.

ACKNOWLEDGMENTS

We thank Chevron Technical Center for permission to publish this work. We would also like to thank the Reinforcement Learning Team in Microsoft who provided support during this project. Finally, we thank Denis Voskov and the Delft Advanced Research Terra Simulator (DARTS) team at TU Delft for providing the reservoir simulator used in this work.

REFERENCES

1. Isebor OJ, Ciaurri DE, and Durlofsky LJ. Generalized Field-Development Optimization with Derivative-free Procedures. *SPE J* (2014b) 19(05):891–908. doi:10.2118/163631-pa
2. Isebor OJ, Durlofsky LJ, and Echeverría Ciaurri D. A Derivative-free Methodology with Local and Global Search for the Constrained Joint Optimization of Well Locations and Controls. *Comput Geosciences* (2014a) 18(3-4):463–82. doi:10.1007/s10596-013-9383-x
3. He J, Tang M, Hu C, Tanaka S, Wang K, Wen XH, et al. Deep Reinforcement Learning for Generalized Field Development Optimization. *SPE J* (2021). doi:10.2118/203951-PA
4. Brouwer DR, and Jansen J. Dynamic Optimization of Water Flooding with Smart wells Using Optimal Control Theory. In: European Petroleum Conference. Aberdeen, United Kingdom: OnePetro (2002) doi:10.2118/78278-ms
5. Sarma P, Durlofsky LJ, Aziz K, and Chen WH. Efficient Real-Time Reservoir Management Using Adjoint-Based Optimal Control and Model Updating. *Comput Geosci* (2006) 10(1):3–36. doi:10.1007/s10596-005-9009-z
6. Ciaurri DE, Isebor OJ, and Durlofsky LJ. Application of Derivative-free Methodologies to Generally Constrained Oil Production Optimisation Problems. *Ijmmno* (2011) 2(2):134–61. doi:10.1504/ijmmno.2011.039425
7. Awotunde AA. A Comprehensive Evaluation of Dimension-Reduction Approaches in Optimization of Well Rates. *SPE J* (2019) 24(03):912–50. doi:10.2118/194510-pa
8. Fonseca R, Leeuwenburgh O, Van den Hof P, and Jansen J. Ensemble-based Hierarchical Multi-Objective Production Optimization of Smart wells. *Comput Geosciences* (2014) 18(3-4):449–61. doi:10.1007/s10596-013-9399-2
9. Chen Y, Oliver DS, and Zhang D. Efficient Ensemble-Based Closed-Loop Production Optimization. *SPE J* (2009) 14(04):634–45. doi:10.2118/112873-pa
10. Cardoso MA, and Durlofsky LJ. Linearized Reduced-Order Models for Subsurface Flow Simulation. *J Comput Phys* (2010) 229(3):681–700. doi:10.1016/j.jcp.2009.10.004
11. He J, Sætrom J, and Durlofsky LJ. Enhanced Linearized Reduced-Order Models for Subsurface Flow Simulation. *J Comput Phys* (2011) 230: 8313–41. doi:10.1016/j.jcp.2011.06.007
12. Bangerth W, Klie H, Wheeler MF, Stoffa PL, and Sen MK. On Optimization Algorithms for the Reservoir Oil Well Placement Problem. *Comput Geosci* (2006) 10(3):303–19. doi:10.1007/s10596-006-9025-7
13. Onwunalu JE, and Durlofsky LJ. Application of a Particle Swarm Optimization Algorithm for Determining Optimum Well Location and Type. *Comput Geosci* (2010) 14(1):183–98. doi:10.1007/s10596-009-9142-1
14. Bouzarkouna Z, Ding DY, and Auger A. Well Placement Optimization with the Covariance Matrix Adaptation Evolution Strategy and Meta-Models. *Comput Geosci* (2012) 16(1):75–92. doi:10.1007/s10596-011-9254-2
15. Zandvliet M, Handels M, van Essen G, Brouwer R, and Jansen J-D. Adjoint-based Well-Placement Optimization under Production Constraints. *Spe J* (2008) 13(04):392–9. doi:10.2118/105797-pa
16. Bellout MC, Echeverría Ciaurri D, Durlofsky LJ, Foss B, and Kleppe J. Joint Optimization of Oil Well Placement and Controls. *Comput Geosci* (2012) 16(4):1061–79. doi:10.1007/s10596-012-9303-5
17. Nasir Y, Yu W, and Sepehrnoori K. Hybrid Derivative-free Technique and Effective Machine Learning Surrogate for Nonlinear Constrained Well Placement and Production Optimization. *J Pet Sci Eng* (2020) 186(106): 726. doi:10.1016/j.petrol.2019.106726
18. Chen C, Li G, and Reynolds ACC. Robust Constrained Optimization of Short- and Long-Term Net Present Value for Closed-Loop Reservoir Management. *SPE J* (2012) 17(03):849–64. doi:10.2118/141314-pa
19. Chen B, Fonseca R-M, Leeuwenburgh O, and Reynolds AC. Minimizing the Risk in the Robust Life-Cycle Production Optimization Using Stochastic Simplex Approximate Gradient. *J Pet Sci Eng* (2017) 153:331–44. doi:10.1016/j.petrol.2017.04.001
20. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *nature* (2016) 529(7587):484–9. doi:10.1038/nature16961
21. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, et al. Mastering the Game of Go without Human Knowledge. *nature* (2017a) 550(7676):354–9. doi:10.1038/nature24270
22. Wang N, Chang H, and Zhang D. Efficient Uncertainty Quantification for Dynamic Subsurface Flow with Surrogate by Theory-Guided Neural Network. *Comp Methods Appl Mech Eng* (2021) 373(113):492. doi:10.1016/j.cma.2020.113492
23. Jin ZL, Liu Y, and Durlofsky LJ. Deep-learning-based Surrogate Model for Reservoir Simulation with Time-Varying Well Controls. *J Pet Sci Eng* (2020) 192:273. doi:10.1016/j.petrol.2020.107273
24. Tang M, Liu Y, and Durlofsky LJ. Deep-learning-based Surrogate Flow Modeling and Geological Parameterization for Data Assimilation in 3d Subsurface Flow. *Comp Methods Appl Mech Eng* (2021) 376(113):636. doi:10.1016/j.cma.2020.113636
25. Guevara J, Patel RG, and Trivedi JJ. Optimization of Steam Injection for Heavy Oil Reservoirs Using Reinforcement Learning. In: SPE International Heavy Oil Conference and Exhibition, Kuwait City, Kuwait, December 2018. Dallas, TX, USA: OnePetro (2018) doi:10.2118/193769-ms
26. Hourfar F, Bidgoly HJ, Moshiri B, Salahshoor K, and Elkamel A. A Reinforcement Learning Approach for Waterflooding Optimization in Petroleum Reservoirs. *Eng Appl Artif Intelligence* (2019) 77:98–116. doi:10.1016/j.engappai.2018.09.019
27. Ma H, Yu G, She Y, and Gu Y. Waterflooding Optimization under Geological Uncertainties by Using Deep Reinforcement Learning Algorithms. In: SPE Annual Technical Conference and Exhibition, Calgary, Alberta, Canada, September 2019. OnePetro (2019) doi:10.2118/196190-ms
28. Miftakhov R, Al-Qasim A, and Efremov I. Deep Reinforcement Learning: Reservoir Optimization from Pixels. In: International Petroleum Technology Conference, International Petroleum Technology Conference (2020) doi:10.2523/iptc-20151-ms

29. Peaceman DW. Interpretation of Well-Block Pressures in Numerical Reservoir Simulation with Nonsquare Grid Blocks and Anisotropic Permeability. *Soc Pet Eng J* (1983) 23(03):531–43. doi:10.2118/10528-pa
30. Sutton RS, and Barto AG. *Reinforcement Learning: An Introduction*. MIT press (2018).
31. Howard RA. *Dynamic Programming and Markov Processes*. John Wiley (1960).
32. Schulman J, Wolski F, Dhariwal P, Radford A, and Klimov O. *Proximal Policy Optimization Algorithms*. Ithaca, NY, United States: Cornell University (2017).
33. Huang S, and Ontañón S. *A Closer Look at Invalid Action Masking in Policy Gradient Algorithms*. Ithaca, NY, United States: Cornell University (2020).
34. Tang C-Y, Liu C-H, Chen W-K, and You SD. Implementing Action Mask in Proximal Policy Optimization (Ppo) Algorithm. *ICT Express* (2020) 6(3): 200–3. doi:10.1016/j.icte.2020.05.003
35. Schulman J, Levine S, Abbeel P, Jordan M, and Moritz P. Trust Region Policy Optimization. In: International Conference on Machine Learning (2015a). p. 1889–97.
36. Schulman J, Moritz P, Levine S, Jordan M, and Abbeel P. *High-dimensional Continuous Control Using Generalized Advantage Estimation*. Ithaca, NY, United States: Cornell University (2015b).
37. Kullback S, and Leibler RA. On Information and Sufficiency. *Ann Math Statist* (1951) 22(1):79–86. doi:10.1214/aoms/1177729694
38. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, et al. Mastering the Game of Go without Human Knowledge. *nature* (2017b) 550(7676):354–9. doi:10.1038/nature24270
39. He K, Zhang X, Ren S, and Sun J. Deep Residual Learning for Image Recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016). p. 770–8. doi:10.1109/cvpr.2016.90
40. Krizhevsky A, Sutskever I, and Hinton GE. Imagenet Classification with Deep Convolutional Neural Networks. In: *Advances in Neural Information Processing Systems* (2012). p. 1097–105.
41. Nair V, and Hinton GE. Rectified Linear Units Improve Restricted Boltzmann Machines. In: *ICML* (2010).
42. He J, Chen J, He X, Gao J, Li L, Deng L, et al. *Deep Reinforcement Learning with a Natural Language Action Space*. Ithaca, NY, United States: Cornell University (2015).
43. Khait M, and Voskov DV. Operator-based Linearization for General Purpose Reservoir Simulation. *J Pet Sci Eng* (2017) 157:990–8. doi:10.1016/j.petrol.2017.08.009
44. Khait M. *Delft Advanced Research Terra Simulator: General Purpose Reservoir Simulator with Operator-Based Linearization*. Delft, the Netherlands: Delft University of Technology (2019). PhD thesis.
45. Haarnoja T, Zhou A, Abbeel P, and Levine S. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor* (2018). p. 01290.
46. Espeholt L, Soyer H, Munos R, Simonyan K, Mnih V, Ward T, et al. *Impala: Scalable Distributed Deep-Rl with Importance Weighted Actor-Learner Architectures* (2018). p. 01561.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.