DeepThermal: Combustion Optimization for Thermal Power Generating Units Using Offline Reinforcement Learning

Xianyuan Zhan[1*], **Haoran Xu**[2,3], **Yue Zhang**[2], **Xiangyu Zhu**[2], **Honglei Yin**[2], **Yu Zheng**[2,3]

[1] Institute for AI Industry Research (AIR), Tsinghua University, Beijing, China
[2] JD Intelligent Cities Research, JD Technology, Beijing, China
[3] Xidian University, Xi'an, China
{zhanxianyuan, ryanxhr, zhangyuezjx, zackxiangyu, yinhonglei93}@gmail.com, msyuzheng@outlook.com

Abstract

Optimizing the combustion efficiency of a thermal power generating unit (TPGU) is a highly challenging and critical task in the energy industry. We develop a new data-driven AI system, namely DeepThermal, to optimize the combustion control strategy for TPGUs. At its core, is a new model-based offline reinforcement learning (RL) framework, called MORE, which leverages historical operational data of a TGPU to solve a highly complex constrained Markov decision process problem via purely offline training. In DeepThermal, we first learn a data-driven combustion process simulator from the offline dataset. The RL agent of MORE is then trained by combining real historical data as well as carefully filtered and processed simulation data through a novel restrictive exploration scheme. DeepThermal has been **successfully deployed** in four large coal-fired thermal power plants in China. Real-world experiments show that DeepThermal effectively improves the combustion efficiency of TPGUs. We also report the superior performance of MORE by comparing with the state-of-the-art algorithms on the standard offline RL benchmarks.

Introduction

Thermal power generation forms the backbone of the world's electricity supply and plays a dominant role in the energy structure of many countries. For example, there are more than 2,000 coal-fired thermal power plants in China, contributing to more than 60% of all electricity generated in the country. Every year, thermal power plants across the world consume an enormous amount of non-renewable coal and cause serious air pollution issues. How to improve the combustion efficiency of a *thermal power generating unit* (TPGU) has been a critical problem for the energy industry for decades. Solving this problem has huge economic and environmental impacts. For instance, by only improving **0.5%** of combustion efficiency of a 600 megawatt (MW) TPGU, a power plant can save more than 4000 tons of coal and reduce hundreds of tons of emissions (e.g. carbon dioxides CO_2 and nitrogen oxides NO_x) a year.

After decades of development and technology advances, most modern TPGUs can achieve a combustion efficiency ranging from 90% to 94%. Further improving the combustion

efficiency of TPGUs, especially through system control aspect is becoming an extremely challenging task. The difficulties arise from several aspects. First, TPGUs are highly complex and large systems, which contain lots of equipment, huge amounts of sensors and complicated operation mechanisms. The involvement of the large number of safety constraints and domain knowledge further exacerbates the difficulty of the task. Lastly, it is desirable to achieve long-term optimization with multiple objectives, such as increasing combustion efficiency while reducing NO_x emission. All these factors and requirements result in an extremely difficult problem that has not been well solved after decades of effort. Currently, most coal-fired thermal power plants still use semi-automatic control systems, and their control heavily depends on the experience and expertise of human operators.

Conventional industrial control optimization approaches, such as the widely used PID controller (Astrom and Hagglund 2006) and model predictive control (MPC) algorithms (Garcia, Prett, and Morari 1989), neither have sufficient expressive power nor scale with the increase of problem size. When facing large and complex systems, these methods will have unavoidable modeling complexity and time cost to obtain the optimal solutions. Hence most existing combustion optimization approaches decompose the TPGU into individual small sub-systems that only optimize a limited amount of state and control variables (Kalogirou 2003; Lee et al. 2007; Ma and Lee 2011; Liu and Bansal 2014). The recent advances of deep reinforcement learning (RL) provide another promising direction. Deep RL leverages expressive function approximators and has achieved great success in solving complex tasks such as games (Mnih et al. 2015; Silver et al. 2017) and robotic control (Levine et al. 2016). However, all these achievements are restricted to the online setting, where agents are allowed to have unrestricted interaction with real systems or perfect simulation environments. In real-world industrial control scenarios, an algorithm may never get the chance to interact with the system at the training stage. A problematic control policy can lead to disastrous consequences to system operation. Besides, most real-world industrial systems are overly complex or partially monitored by sensors, which makes it impossible to build a high-fidelity simulator.

Fortunately, industrial systems like TPGUs have long-term storage of the operational data collected from sensors, and the recently emerged offline RL provides an ideal frame-

work for our problem. Offline RL focuses on training RL policies from offline, static datasets without environment interaction. The main difficulty of offline RL tasks is the *distributional shift* issue (Kumar et al. 2019), which occurs when the learned policies make counterfactual queries on unknown out-of-distribution (OOD) data samples, causing non-rectifiable exploitation error during training. The key insight of recent offline RL algorithms (Fujimoto, Meger, and Precup 2019; Kumar et al. 2019; Wu, Tucker, and Nachum 2019; Yu et al. 2020) is to restrict policy learning stay close to the data distribution. However, these methods are over-conservative, hindering the chance to surpass a sub-optimal behavior policy.

In this work, we develop a new data-driven AI system, namely **DeepThermal** (Chinese name: 深燧), to optimize the combustion efficiency of real-world TPGUs. DeepThermal constructs a data-driven combustion process simulator to facilitate RL training. The core of DeepThermal is a new model-based offline RL framework, called **MORE**, which is capable of leveraging both logged datasets and an imperfect simulator to learn a policy under safety constraints and greatly surpass the behavior policy. DeepThermal has already been successfully deployed in four large coal-fired thermal power plants in China. Real-world experiments show that the optimized control strategies provided by DeepThermal effectively improve the combustion efficiency of TPGUs. Extensive comparative experiments on standard offline RL benchmarks also demonstrate the superior performance of MORE against the state-of-the-art offline RL algorithms.

Overview

Operation Mechanisms of TPGUs

Thermal power generating unit converts the chemical energy of the coal to electric power. The power generation process of a TPGU is highly complicated involving three major stages (see Figure 1). 1) **Coal pulverizing stage**: Coals from the coal-feeders are pulverized to fine-grained particles by coal mills before outputting to the burner. To ensure complete combustion, many control operations need to be properly performed, e.g. amount of coal should meet the demand load; valves of the cold and hot air blowers (primary blowers) are adjusted to ensure suitable primary air temperature. 2) **Burning stage**: Pulverized coals and air from the secondary blower are injected through 20~48 locations of the burner (depending on the specific structure of the burner). The valves of the secondary blower at each injection location need to be precisely controlled to allow a large fireball to form at the center of the burner, facilitating complete combustion. Safety and regulatory issues need also be guaranteed, such as maintaining negative internal pressure and pollutants generated below a certain level. 3) **Steam circulation stage**. The burner vaporizes water in the boiler and generates high-temperature, high-pressure steam, which drives a steam turbine to generate electricity satisfying demand load. The steam generated needs to satisfy multiple temperature and pressure requirements, which are controlled by the valves of the induced draft fan, and the amount of cooling water used, etc.

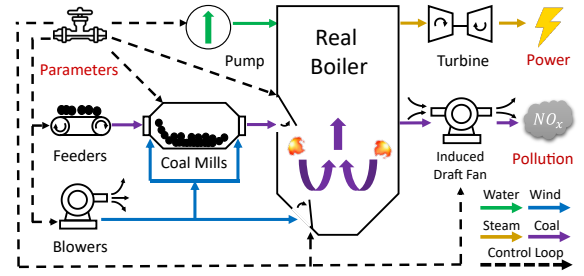Optimizing the combustion efficiency of a TPGU involves

Figure 1: Illustration of operation mechanisms of a TPGU

70~100 major continuous control variables and the chemical properties of the coal, which is extremely challenging.

Preliminaries

We model the combustion optimization problem for TPGUs as a Constrained Markov Decision Process (CMDP) (Altman 1999), which augments the standard MDP with multiple safety constraints. A CMDP is represented by a tuple $(\mathcal{S}, \mathcal{A}, T, r, c_{1:m}, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $T(s_{t+1}|s_t, a_t)$ denotes the transition dynamics. $r(s_t, a_t) > 0$ is the reward function and $c_{1:m}(s_t, a_t)$ are m cost functions. $\gamma \in (0, 1)$ is the discount factor. A policy $\pi(s)$ is a mapping from states to actions. In our problem, the state, action, reward and costs are set as follows.

States $\mathcal{S}$: We use the chemical property of the coal and sensor data that relevant to the combustion process of a TPGU as states, including temperature, pressure, wind, and water volume as well as other sensor readings of different stages in the combustion process described in previous section.

Actions $\mathcal{A}$: We consider all the key control variables that impact combustion process in a TPGU as actions, such as the adjustment of valves and baffles. All actions are continuous.

Reward function r: We model the reward as a weighted combination of combustion efficiency $Effi$ and reduction in NO_x emission Emi, i.e. $r_t = \alpha_r Effi_t + (1 - \alpha_r)Emi_t$. We set $\alpha_r = 0.8$ in our deployed systems, as improving combustion efficiency is the primary concern for many power plant.

Cost functions $c_{1:m}$: We model a series of safety constraints as costs, such as load, internal pressure, and temperature satisfaction. Violating these constraints will lead to a positive penalty value. We denote a weighted combination of costs as $\tilde{c}(s, a) = \sum_{i=1}^{m} \alpha_c^i c_i(s, a)$, where $\alpha_c^{1:m}$ are set according to experts' opinion.

In our problem, we assume no interaction with the actual TPGU and only have a static historical operational dataset $\mathcal{B} = (s, a, s', r, c_{1:m})$, generated by unknown behavior policies from TPGU operators. Our goal is to learn a policy $\pi^*(s)$ from $\mathcal{B}$ that maximizes the expected discounted reward $R(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$ while controlling the expected discounted combined costs $C(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \tilde{c}(s_t, a_t)]$ below a threshold l, mathematically:

$$\pi^* = \arg\max_\pi R(\pi)$$
$$\text{s.t.} \quad C(\pi) \le l \tag{1}$$

We also learn a data-driven combustion process simulator, which serves as an approximated dynamics model

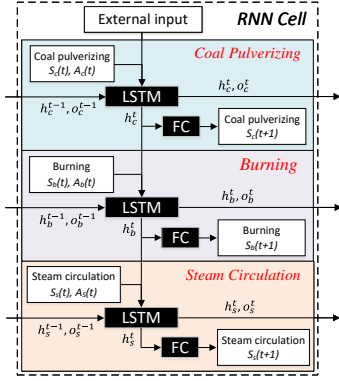Figure 2: Design of the combustion process simulator

Figure 3: The framework of MORE

$f(s_t, a_t) = \hat{s}_{t+1}, \hat{r}_{t+1}, \hat{c}_{t+1}$ to generate future states, rewards and costs in our model-based RL framework.

Combustion Process Simulator

Accurately fitting the dynamics of combustion process is very challenging. TPGUs are highly complex, large, and partially observed open systems. Due to extremely high temperature and pressure in certain parts of a TPGU, some state information is not fully captured by sensors. External factors like ambient temperature and chemical properties of the coal also impact combustion. We propose a modified deep recurrent neural network (RNN) as the combustion process simulator, with its internal cell structure specially designed according to the actual physical process. As shown in Figure 2, the input state-action pairs are split into 3 blocks to encode their physical and hierarchical dependencies, and the long short term memory (LSTM) layers are used to capture the temporal correlations. Specifically, we first model the coal pulverizing stage by considering the related states s_t^c and actions a_t^c together with the external inputs s_t^e (e.g. environment temperature and chemical properties of the coal), and predict the next coal pulverizing related states s_{t+1}^c. We then combine the impact from the coal pulverizing stage (encoded in the hidden states h_t^c) with the states s_t^b and actions a_t^b of burning stage to predict the next state s_{t+1}^b. Lastly, impacts of burning stage h_t^b are combined with the states s_t^s and actions a_t^s of steam circulation stage to predict the related states of next time step s_{t+1}^s. This design embeds domain knowledge in the network structure, which helps to alleviate the impact of missing information in the partially observed system, and greatly improves model accuracy and robustness.

The parameters of the simulator are learned by minimizing the mean squared error (MSE) of the actual and predicted states. To further strengthen the simulator, following techniques are applied: 1) **Seq2seq** and **scheduled sampling**: We use sequence to sequence structure and scheduled sampling (Bengio et al. 2015) to improve long-term prediction accuracy of the simulator. 2) **Noisy data augmentation**: We add gradually vanishing Gaussian noises on the state inputs during simulator training, which can be perceived as a means of data augmentation. This treatment helps to improve model robustness and prevent overfitting.
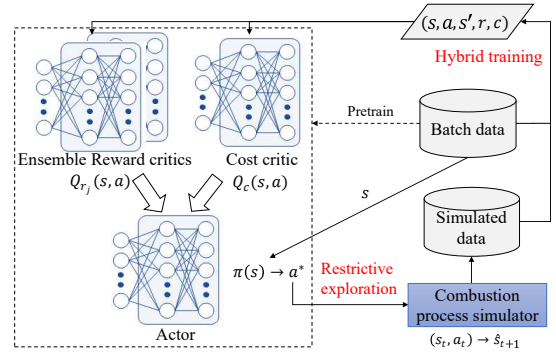
MORE: An Improved Model-based Offline RL Framework

In this Section, we introduce the core RL algorithm used in DeepThermal: Model-based Offline RL with Restrictive Exploration (MORE). MORE tackles the challenge of offline policy learning under constraints with an imperfect simulator. The framework of MORE is illustrated in Figure 3. It introduces an additional cost critic to model and enforces safety constraints satisfaction of the combustion optimization problem. MORE quantifies the risks imposed by the imperfect simulator using a novel *restrictive exploration* scheme, from the perspective of both prediction reliability (measured by *model sensitivity*) as well as the possibility of being OOD samples (measured by *data density* in the behavioral data). Specifically, MORE trusts the simulator only when it is certain about the outputs and adds reward penalty on potential OOD predictions to further guide the actor to explore in high density regions. Finally, MORE ingeniously combines the real data and carefully distinguished simulated data to learn a safe policy through a *hybrid training* procedure.

Safe Policy Optimization

MORE uses two types of Q-functions, Q_r and Q_c, for reward maximization and cost evaluation. The policy optimization is performed on the carefully combined real-simulation data $\mathcal{D}$ (see hybrid training subsection for details) as follows:

$$\pi_\theta := \max_\pi \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi} \left[\min_{j=1,2} Q_{r_j}(s, a) \right]$$
$$\text{s.t. } \mathbb{E}_{a \sim \pi}[Q_c(s, a)] \leq l \tag{2}$$

MORE adopts the Clipped Double-Q technique (Fujimoto, Hoof, and Meger 2018) by using two Q_r functions to penalize the uncertainty in Q_r and alleviate the overestimation issue that commonly occurs in offline RL. This trick is not applied to the Q_c-network as it could potentially underestimate the cost value. To solve this problem, we employ the Lagrangian relaxation procedure (Boyd, Boyd, and Vandenberghe 2004) by introducing following Lagrangian function:

$$\mathcal{L}(\pi, \lambda) = \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi} \left[\min_{j=1,2} Q_{r_j}(s, a) - \lambda \left(Q_c(s, a) - l \right) \right]$$

where λ is the Lagrangian multiplier. The original problem (Eq. 2) can be converted to following unconstrained form:

$$(\pi^*, \lambda^*) = \arg \min_{\lambda \geq 0} \max_{\pi} \mathcal{L}(\pi, \lambda). \tag{3}$$

$$\lambda \leftarrow [\lambda + \eta(\mathbb{E}_{a \sim \pi}[Q_c(s, a)] - l)]^+ \tag{4}$$

where η is the step size and $[x]^+ = \max\{0, x\}$ is the projection onto the dual space ($\lambda \geq 0$). We use the iterative primal-dual update method to solve the above unconstrained minimax problem. In the primal stage, we fix the dual variable λ and perform policy gradient update on policy π. In the dual stage, we fix the policy π and update λ by dual gradient ascent (Eq. 4).

Restrictive Exploration

Like many real-world tasks, we only have an imperfect simulator for the combustion optimization problem. The model is learned entirely from the offline data and possible to make inaccurate predictions that impact RL training. The inaccuracies are mainly from two sources: 1) lack of data to fully train the model in certain regions; 2) unable to well fit data due to limited model capability or system complexity. Note the first case is not an absolute criteria to detect model inaccuracy. Models can perform reasonably well in low density regions of data if the pattern is easy to learn. Under this setting, we should encourage exploration with the model even if the resulting samples lie outside the dataset distribution.

With this intuition, we design a new *restrictive exploration* strategy to fully utilize the generalizability of the simulator from both the model and data perspective. The key insight is to only consider the samples that the simulator is certain, and then further distinguish whether the simulated samples are in data distribution or not.

Model sensitivity based filtering. We first filter out those unreliable simulated data if the model is uncertain. Previous work (Novak et al. 2018) has shown that model sensitivity can be a viable measure for model generalizability on data samples. We use this metric to detect if the model is certain or well generalizable on simulated state-action pairs from the model's perspective. For sensitivity quantification, we inject K i.i.d. Gaussian noises $\epsilon_i \sim N(\mathbf{0}, \sigma\mathbf{I})$, $i \in \{1, \dots, K\}$ on a input state-action pair (s, a) of the model, and compute the variance of the output perturbations $u = Var[\epsilon^y]$ as the sensitivity metric, where $\epsilon^y = [f((s, a) + \epsilon_i) - f(s, a)]_{i=1}^K$, and σ controls the degree of perturbation. A large u suggests the model is sensitive to input perturbation at (s, a), which is an indication of uncertainty or lack of prediction robustness at this point (Novak et al. 2018).

Let τ_s be a batch of simulated transitions $\{(s, a, s', r, \tilde{c})\}$ at training step t and $\mathbf{u}_{s,t}$ be the sensitivity of τ_s. MORE filters problematic simulated transitions as follows:

$$\tau_m := \{\tau_s | \mathbf{u}_{s,t} < l_u\} \tag{5}$$

where l_u is a predefined threshold. In practice, we choose it to be the β_u-percentile value of sensitivity pre-evaluated at all state-action pairs in the offline dataset $\mathcal{B}$.

Data density based filtering. The lack of data in low density regions of $\mathcal{B}$ may provide insufficient information to describe

---

Algorithm 1: Restrictive exploration

1: **Require:** Simulator f, threshold l_u, l_p, batch of real data transitions $\tau_n = \{(s, a, s', r, \tilde{c})\}_n$, and rollout length H
2: **for** τ in τ_n **do**
3:     Set $\hat{s}_1 = s, \tau^+ = \emptyset, \tau^- = \emptyset$
4:     **for** Rollout step: $h = 1, .., H$ **do**
5:         Generate simulated transition $(\hat{s}_h, \pi(\hat{s}_h), \hat{s}_{h+1}, \hat{r}_h, \hat{c}_h)$, where $(\hat{s}_{h+1}, \hat{r}_h, \hat{c}_h) = f(\hat{s}_h, \pi(\hat{s}_h))$
6:         **// Model sensitivity based filtering**
7:         **if** Evaluated sensitivity $u(\hat{s}_h, \hat{a}_h) < l_u$ (follow Eq.5) **then**
8:             **// Data density based filtering** Compute data density $p_m(\hat{s}_h, \hat{a}_h)$ according to Eq.6
9:             Add $(\hat{s}_h, \pi(\hat{s}_h), \hat{s}_{h+1}, \hat{r}_h, \hat{c}_h)$ into positive sample set τ^+ or negative set τ^- according to Eq.7
10:             Add reward penalties for samples in τ^- as in Eq.8
11:         **end if**
12:     **end for**
13: **end for**
14: **Output:** (τ^+, τ^-)

---

the system dynamics completely and accurately. This can lead to unreliable OOD simulated transitions that cause exploitation error during policy learning. To address this issue, we propose the data-density based filtering to encourage exploration in high density regions, while cautioning about potential OOD samples. The key insight is to carefully distinguish between positive (in high density region) and negative (in low density or OOD) simulated samples. We trust more on positive samples while penalizing on the negative samples.

In practical implementation, we use a state-action variational autoencoder (VAE) (Kingma and Welling 2014) to fit the data distribution of $\mathcal{B}$. VAE maximize the following evidence lower bound (ELBO) objective that lower bounds the actual log probability density of data:

$$\mathbb{E}_{z \sim q_{\omega_2}} [\log p_{\omega_1}(s, a|s, a, z)] - D_{KL} [q_{\omega_2}(z|s, a) \| N(0, 1)] \tag{6}$$

where the first term represents the reconstruction loss and the second term is the KL-divergence between the encoder output and the prior $N(0, 1)$. We use ELBO to approximate the probability density of data. Let τ_m be the simulation samples that passed model sensitivity based filtering at training step t. We estimate data density p_m of state-action pairs in τ_m with Eq. 6 and split τ_m to positive samples τ_+ and negative samples τ_- with threshold l_p.

$$\tau^+ := \{\tau_m | p_m > l_p\}, \quad \tau^- := \{\tau_m | p_m \leq l_p\} \tag{7}$$

Like l_u, we choose l_p to be the β_p-percentile ELBO values pre-evaluated on all state-action pairs in the dataset $\mathcal{B}$.

Hybrid training

After quantifying the risks imposed by the imperfect simulator, MORE introduces a hybrid training strategy to differentiate the impact of positive and negative simulated samples obtained from the restrictive exploration. We keep the positive samples as their original forms to encourage fully exploiting the generalizability of the model, but penalize the rewards of negative samples to guide policy learning away

Algorithm 2: Complete algorithm of MORE

1: **Require:** Offline dataset $\mathcal{B}$
2: Pre-train actor π_θ, reward critic ensemble $\{Q_{r_i}(s,a|\phi_{r_i})\}_{i=1,2}$ and cost critic $Q_c(s,a|\phi_c)$ with real data. Initialize target networks $\{Q'_{r_i}\}_{i=1}^2$ and Q'_c with $\phi'_{r_i} \leftarrow \phi_{r_i}$ and $\phi'_c \leftarrow \phi_c$
3: **for** Training step: $t = 1, ..., T$ **do**
4:     Random sample mini-batch transitions τ_n from $\mathcal{B}$
5:     Obtain (τ^+, τ^-) using restrictive exploration (Algorithm 1)
6:     Construct local buffer $\mathcal{R} = \{(s,a,r,c,s')\}$ using τ^+, τ^- and τ_n, as well as Eq.8
7:     Set $y = \min_{i=1,2} Q'_{r_i}(s', \pi(s'))$ and $z = Q'_c(s', \pi(s'))$
8:     Update Q_{r_i} by minimizing $(Q_{r_i} - (r + \gamma y))^2$
9:     Update Q_c by minimizing $(Q_c - (c + \gamma z))^2$
10:    Update policy π_θ by Eq.3 using policy gradient
11:    Update λ by Eq.4 using dual gradient ascent
12:    Update target cost critic: $\phi'_c \leftarrow \rho\phi_c + (1-\rho)\phi'_c$
13:    Update target reward critics: $\phi'_{r_i} \leftarrow \rho\phi_{r_i} + (1-\rho)\phi'_{r_i}$
14: **end for**

from high-risk regions. Moreover, we pretrain π, Q_r and Q_c with real data in order to run RL algorithm with good initial parameters, which is observed to improve stability of training and speed up convergence.

Reward penalization on negative samples. Several previous works in online and offline RL (Yu et al. 2020; Kidambi et al. 2020; Shi et al. 2019) already use penalized rewards to regularize policy optimization against potential negative impacts of simulated samples. However, unlike prior works that penalize the reward of all simulated samples, we use a more delicate strategy by softly penalizing rewards as:

$$\hat{r}(\hat{s}_t, \hat{a}_t) \leftarrow \frac{\hat{r}(\hat{s}_t, \hat{a}_t)}{1 + [\kappa(l_p - p_m(\hat{s}_t, \hat{a}_t))]^+} \qquad (8)$$

where $[x]^+ = \max\{x, 0\}$ and κ is a hyper-parameter to control the scale of reward penalty. It's easy to find that positive samples whose approximated density $p_m(\hat{s}_t, \hat{a}_t)$ higher than l_p are not penalized (see Eq.7). Only negative samples are penalized and the penalty weight is propotional to the difference between $p_m(\hat{s}_t, \hat{a}_t)$ and l_p. This strategy encourages policy updates toward high reward directions suggested by positive samples, providing the possibility to generalize beyond the offline dataset; while also forcing policy updates away from the area of OOD negative samples to avoid potential exploitation error.

MORE constructs a special local buffer $\mathcal{R}$ to combine real, positive and negative simulated data for offline training. The full algorithm is summarized in Algorithm 2.

Experiments

Dataset and Settings
We conduct experiments on both real TPGUs and standard offline RL benchmarks. Detailed settings are as follows:
Real-world datasets and experiment settings. We used 1~2 years' historical TPGU operational data to train our models. Very old data are not used due to potentially different patterns compared with current conditions of the TPGU, mainly caused by changes and deteriorating of equipment and

Table 1: Evaluation of the combustion process simulator

Model	ARIMA	GBRT	DNN	LSTM	Ours
RMSE	3.05e-1	1.97e-1	2.05e-2	1.69e-3	**6.54e-4**
MAE	2.66e-1	2.65e-1	2.73e-2	2.50e-2	**1.55e-3**

devices. We considered more than 800 sensors and optimized about 100 control variables. A specially designed feature engineering process is used to process these sensor data into about 100~170 states and 30 ∼ 50 actions (differ for TPGUs in different power plants). Some sensor values monitoring similar state as well as control variables sharing the same operation mode are merged into single values to reduce problem dimension. Finally, we re-sampled the processed data into equal 20~30 second (depending on the quality of the sensor data) interval data, which typically results in 1~2 million records for RL training.
Datasets and settings for standard offline RL benchmarks. We evaluate and compare the performance of MORE on the standard offline RL benchmark D4RL (Fu et al. 2020). D4RL provides datasets specifically designed for the offline setting. We mainly focus on three locomotion tasks (hopper, halfcheetah and walker2d) and two dataset types (medium and mixed) that are more relevant to real-world applications. These datasets are generated as follows: **medium**: generated using a partially trained SAC policy to roll out 1 million steps. **mixed**: train a SAC policy (Haarnoja et al. 2018) until reaching a predefined performance threshold, and take the replay buffer as the dataset.

For all experiments on D4RL datasets, we model the dynamics model using fully connected neural networks.

Evaluation of the Simulator

We compare in Table 1 the performance of the combustion process simulator with four baselines that commonly used for time-series data prediction, including ARIMA, GBRT, DNN (feedforward neural network) and stacked LSTM. It is observed that our proposed combustion process simulator significantly outperforms all the baselines on both evaluation metrics (RMSE and MAE). This demonstrates the effectiveness of the proposed simulator, as well as the benefit of incorporating domain knowledge in the network design.

Real-World Experiments

To verify the effectiveness of DeepThermal, we conducted a series of before-and-after tests on real-world TPGUs. The duration of these experiments ranges from 1 to 1.5 hours. During the experiment, the human operator adjusted the control strategy of a TPGU according to the recommended actions provided by the learned RL policy.

Figure 4 presents the experiment results in CHN Energy Nanning Power Station on three different load settings. It is observed that DeepThermal effectively improves combustion efficiency in all three load settings. In the 270MW, 290MW and 310MW load setting experiments, the optimized control strategy achieved the maximum increase of 0.52%, 0.31% and 0.48% on the combustion efficiency in about 60 minutes
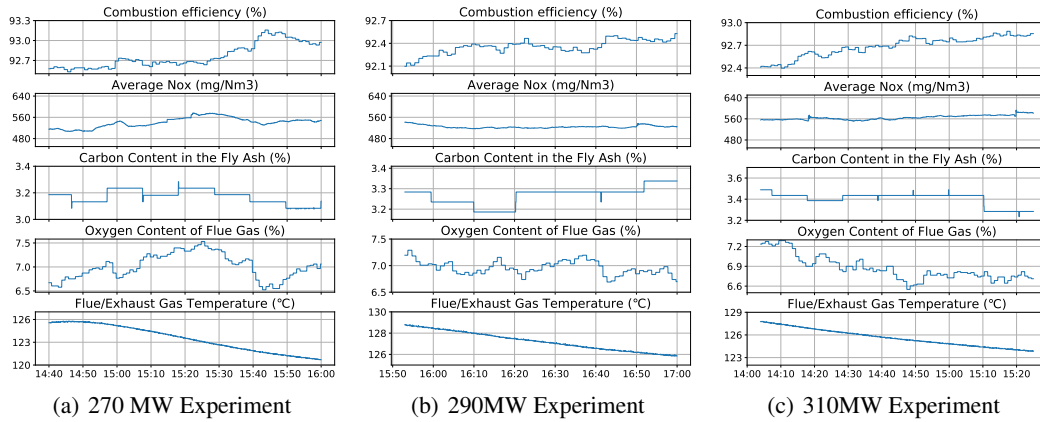
| | (a) 270 MW Experiment | (b) 290MW Experiment | (c) 310MW Experiment |

Figure 4: Real-world experiments at CHN Energy Nanning Power Station

Table 2: Results for D4RL datasets, averaged over 5 random seeds

Dataset	Batch Mean	Batch Max	BC	BEAR	BRAC-v	BCQ	MBPO	MOPO	MORE (Ours)
halfcheetah-medium	3953	4410.7	4202.7	4513.0	5369.5	4767.9	3234.4	4972.3	**5970**
hopper-medium	1021.7	3254.3	924.1	**1674.5**	1031.4	**1752.4**	139.9	891.5	1264
walker2d-medium	498.4	3752.7	302.6	2717.0	**3733.4**	2441.3	582.8	817.0	**3649**
halfcheetah-mixed	2300.6	4834.2	4488.2	4215.1	5419.2	4463.9	5593.0	**6313.0**	5790
hopper-mixed	470.5	1377.9	364.4	331.9	9.7	688.7	1600.8	**2176.8**	**2100**
walker2d-mixed	358.4	1956.5	518.5	1161.4	36.2	1057.8	1019.1	1790.7	**1947**

compared with the initial values. The average NO_x concentrations before the denitrification reactor remain at a relative stable level. We also present three key indicators that reflect sufficient combustion, including *carbon content in the fly ash*, *oxygen content of flue gas* and *flue/exhaust gas temperature*. In all three experiments, these three indicators achieved a certain level of decrease, which provides clear evidence of combustion improvement.

Evaluation on Offline RL Benchmarks

In this subsection, we further investigate the performance of the proposed model-based offline RL framework MORE on standard offline RL benchmark D4RL.

Comparative Evaluations. We compare MORE against the state-of-the-art offline RL algorithms, including model-free algorithms such as BCQ (Fujimoto, Meger, and Precup 2019), BEAR (Kumar et al. 2019) and BRAC-v (Wu, Tucker, and Nachum 2019) that constrain policy learning to stay close to the behavior policy using various divergence metrics. We also compare against model-based offline RL algorithms including MOPO (Yu et al. 2020) that follows MBPO (Janner et al. 2019) with additional reward penalties. We omit the cost critic of MORE in these experiments, as there are no safety constraints in corresponding D4RL tasks.

The comparative results are presented in Table 2. It is observed that MORE matches or outperforms both the model-free and model-based baselines in most tasks. MOPO is shown to outperform model-free methods by a large margin in the mixed datasets, while performs less well on the medium datasets due to the lack of action diversity. MORE matches the performance of MOPO on the mixed datasets

while greatly surpasses MOPO on the medium datasets. We hypothesize that conditionally removing uncertain simulated samples (via model sensitivity based filtering) as well as introducing data-density based reward penalties on OOD samples provide more reliable and informative simulated data for RL policy learning, thus lead to good results with a potentially imperfect model.

Ablation Study We conduct a series of ablation studies on halfcheetah environment to investigate how different components impact the performance of MORE.

- *Evaluation on the model sensitivity threshold β_u.* It can be shown in Figure 5(a) that, in the mixed dataset where the simulator can learn and generalize well, MORE with $\beta_u = 70\%$ outperforms $\beta_u = 40\%$ (more tolerant to encourage generalization). While in the medium dataset, MORE with $\beta_u = 70\%$ performs inferior than $\beta_u = 40\%$ due to allowing too much problematic samples from the imperfect dynamics models.

- *Evaluation on the data density threshold β_p.* We find in Figure 5(b) that smaller β_p ($\beta_p = 10\%$) performs better in the mixed dataset, while a medium value β_p ($\beta_p = 40\%$) works better in the medium dataset. This again suggests that it is beneficial to be more tolerant to potential OOD simulated samples when the dynamics model is reliable. However, when the dynamics model is imperfect, carefully controlling the ratio between positive and negative samples is important to achieve the best performance.

Additional Evaluation under Safety Constraints. We conduct additional experiments to demonstrate the performance of MORE under safety constraints. We use the halfcheetah-mixed dataset and further introduce the safety cost as the dis-

(a) Ablation of β_u

(b) Ablation of β_p
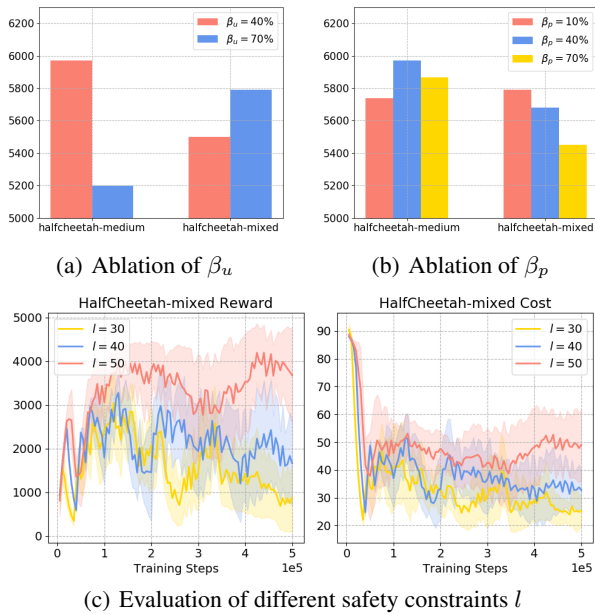

(c) Evaluation of different safety constraints l

Figure 5: Ablation study on MORE

counted cumulative torque that the agent has applied to each joint (Tessler, Mankowitz, and Mannor 2018). The per-state cost $c(s, a)$ is the amount of torque the agent decided to apply at each step, i.e. the L2-norm of the action vector, $\|a\|_2$. It should be noted that by preventing the agent from using high torque values, the agent may learn a sub-optimal policy. We test MORE under different constraint limit $l \in \{30, 40, 50\}$. It can be shown in Figure 5(c) that MORE is robust to different l. In all the tests, the cumulative costs are controlled below the given constraint limits.

Related Work

Complex System Control
PID control (Astrom and Hagglund 2006) is the most common approach for industrial system control. Although PID control ensures safe and stable control, its performance is limited due to insufficient expressive power. Model predictive control (MPC) (Garcia, Prett, and Morari 1989) is another widely used control method, that utilizes an explicit process model to predict the future response of the system and performs control optimization accordingly. MPC has been applied to many areas, such as refining, petrochemicals, food processing, mining/metallurgy and automotive applications (Qin and Badgwell 2003). However, applying MPC in large-scale stochastic systems is often infeasible due to their heavy online computational requirements.

RL overcomes the above challenges by learning the optimal strategy beforehand, a concept similar to parametric programming in explicit model predictive control (Bemporad et al. 2002). Previous works that use RL for real-world control tasks typically rely on existed high-fidelity simulators (Li 2019; Lopez et al. 2018; Todorov, Erez, and Tassa 2012), or build "virtual" simulators using diverse and large data (Shi et al. 2019; Lazic et al. 2018). However, both high-fidelity simulators and diverse data are impossible to obtain in some complex real-world tasks, using data-driven RL algorithms hold the promise of automated decision-making informed only by logged data, thus getting rid of the sim-to-real dilemma (Dulac-Arnold et al. 2020).

Offline Reinforcement Learning
Offline RL (or batch RL (Lange, Gabel, and Riedmiller 2012)) considers the problem of learning policies from offline data without environment interaction. One major challenge of offline RL is the distributional shift issue (Levine et al. 2020), which incurs when the policy distribution deviates largely from the data distribution. Recent model-free methods attempted to solve this problem by constraining the learned policy to be close to the behavior policy via implicit or explicit divergence regularization (Fujimoto, Meger, and Precup 2019; Kumar et al. 2019; Wu, Tucker, and Nachum 2019; Kumar et al. 2020). While performing well in single-modal datasets (e.g., medium datasets), model-free methods are shown to have limited improvements in multi-modal datasets (e.g., mixed datasets and real-world datasets), due to over-restricted constraints.

Model-based RL algorithms provide another solution to offline RL. They adopt a pessimistic MDP framework (Kidambi et al. 2020), where the reward is penalized if the learned dynamic model cannot make an accurate prediction. MOPO (Yu et al. 2020) extends MBPO (Janner et al. 2019) with an additional reward penalty on generated transitions with large variance from the learned dynamic model. MOReL (Kidambi et al. 2020) terminates the generated trajectories if the state-action pairs are detected to be unreliable, i.e. the disagreement within model ensembles is large. MBOP (Argenson and Dulac-Arnold 2021) learns a dynamics model, a behavior policy and a truncated value function to perform model-based offline planning, where the actions are sampled from the learned behavior model instead of the Gaussian distribution. Note that all these methods largely depend on the quality of learned dynamic models, while MORE reduces the reliance on the model by using information from both the model and offline data.

Conclusion
In this paper, we develop DeepThermal, a new data-driven AI system for optimizing the combustion control strategy for TPGUs. To the best knowledge of the authors, DeepThermal is the first offline RL application that has been deployed to solve real-world mission-critical control tasks. The core of DeepThermal is a new model-based offline RL framework, called MORE. MORE strikes the balance between fully utilizing the generalizability of an imperfect model and avoiding exploitation error on OOD samples. DeepThermal has been successfully deployed in four large coal-fired thermal power plants in China. Real-world experiments show that DeepThermal effectively improves the combustion efficiency of TPGUs. We also conduct extensive comparative experiments on standard offline RL benchmarks to demonstrate the superior performance of MORE against the state-of-the-art offline RL algorithms.

Acknowledgments

References

Altman, E. 1999. *Constrained Markov decision processes*, volume 7. CRC Press.

Argenson, A.; and Dulac-Arnold, G. 2021. Model-Based Offline Planning. In *International Conference on Learning Representations*.

Astrom, K. J.; and Hagglund, T. 2006. Advanced PID control.

Bemporad, A.; Morari, M.; Dua, V.; and Pistikopoulos, E. N. 2002. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1): 3–20.

Bengio, S.; Vinyals, O.; Jaitly, N.; and Shazeer, N. 2015. Scheduled sampling for sequence prediction with recurrent Neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, 1171–1179.

Boyd, S.; Boyd, S. P.; and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Dulac-Arnold, G.; Levine, N.; Mankowitz, D. J.; Li, J.; Paduraru, C.; Gowal, S.; and Hester, T. 2020. An empirical investigation of the challenges of real-world reinforcement learning. *arXiv preprint arXiv:2003.11881*.

Fu, J.; Kumar, A.; Nachum, O.; Tucker, G.; and Levine, S. 2020. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning*, 1587–1596.

Fujimoto, S.; Meger, D.; and Precup, D. 2019. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, 2052–2062. PMLR.

Garcia, C. E.; Prett, D. M.; and Morari, M. 1989. Model predictive control: theory and practice—a survey. *Automatica*, 25(3): 335–348.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*, 1861–1870.

Janner, M.; Fu, J.; Zhang, M.; and Levine, S. 2019. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 12519–12530.

Kalogirou, S. A. 2003. Artificial intelligence for the modeling and control of combustion processes: a review. *Progress in energy and combustion science*, 29(6): 515–566.

Kidambi, R.; Rajeswaran, A.; Netrapalli, P.; and Joachims, T. 2020. MOReL: Model-Based Offline Reinforcement Learning. In *Neural Information Processing Systems (NeurIPS)*.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In *ICLR 2014 : International Conference on Learning Representations (ICLR) 2014*.

Kumar, A.; Fu, J.; Soh, M.; Tucker, G.; and Levine, S. 2019. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, 11761–11771.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative Q-Learning for Offline Reinforcement Learning. In *Neural Information Processing Systems (NeurIPS)*.

Lange, S.; Gabel, T.; and Riedmiller, M. 2012. Batch reinforcement learning. In *Reinforcement learning*, 45–73. Springer.

Lazic, N.; Lu, T.; Boutilier, C.; Ryu, M.; Wong, E.; Roy, B.; and Imwalle, G. 2018. Data center cooling using model-predictive control. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 3818–3827.

Lee, K. Y.; Heo, J. S.; Hoffman, J. A.; Kim, S.-H.; and Jung, W.-H. 2007. Neural network-based modeling for a large-scale power plant. In *2007 IEEE Power Engineering Society General Meeting*, 1–8. IEEE.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1): 1334–1373.

Levine, S.; Kumar, A.; Tucker, G.; and Fu, J. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Li, Y. 2019. Reinforcement learning applications. *arXiv preprint arXiv:1908.06973*.

Liu, X.; and Bansal, R. 2014. Integrating multi-objective optimization with computational fluid dynamics to optimize boiler combustion process of a coal fired power plant. *Applied energy*, 130: 658–669.

Lopez, P. A.; Behrisch, M.; Bieker-Walz, L.; Erdmann, J.; Flötteröd, Y.-P.; Hilbrich, R.; Lücken, L.; Rummel, J.; Wagner, P.; and Wießner, E. 2018. Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2575–2582. IEEE.

Ma, L.; and Lee, K. Y. 2011. Neural network based superheater steam temperature control for a large-scale supercritical boiler unit. In *2011 IEEE Power and Energy Society General Meeting*, 1–8. IEEE.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Novak, R.; Bahri, Y.; Abolafia, D. A.; Pennington, J.; and Sohl-Dickstein, J. 2018. Sensitivity and Generalization in Neural Networks: an Empirical Study. In *International Conference on Learning Representations*.

Qin, S. J.; and Badgwell, T. A. 2003. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7): 733–764.

Shi, J.-C.; Yu, Y.; Da, Q.; Chen, S.-Y.; and Zeng, A.-X. 2019. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4902–4909.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359.

Tessler, C.; Mankowitz, D. J.; and Mannor, S. 2018. Reward constrained policy optimization. In *International Conference on Learning Representations (ICLR)*.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. IEEE.

Wu, Y.; Tucker, G.; and Nachum, O. 2019. Behavior Regularized Offline Reinforcement Learning. *arXiv preprint arXiv:1911.11361*.

Yu, T.; Thomas, G.; Yu, L.; Ermon, S.; Zou, J.; Levine, S.; Finn, C.; and Ma, T. 2020. MOPO: Model-based Offline Policy Optimization. In *Neural Information Processing Systems (NeurIPS)*.