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

Xianyuan Zhan[*1,2], **Haoran Xu**[*1,3], **Yue Zhang**[*1],
Yusen Huo[1], **Xiangyu Zhu**[1], **Honglei Yin**[1], **Yu Zheng**[1,2,3]
[1]JD Intelligent Cities Research, China
[2]JD iCity, JD Technology, Beijing, China
[3]Xidian University, China
{zhanxianyuan,ryanxhr,zhangyuezjx,yinhonglei93}@gmail.com
{zackxiangyu,msyuzheng}@outlook.com
{huoyusen1}@126.com

Abstract

Thermal power generation plays a dominant role in the world's electricity supply. It consumes large amounts of coal worldwide, and causes serious air pollution. 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 logged historical operational data of a TPGU to solve a highly complex constrained Markov decision process problem via purely offline training. MORE aims at simultaneously improving the long-term reward (increase combustion efficiency and reduce pollutant emission) and controlling operational risks (safety constraints satisfaction). 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 a TPGU. We also report and demonstrate the superior performance of MORE by comparing with the state-of-the-art algorithms on the standard offline RL benchmarks. To the best knowledge of the authors, DeepThermal is the first AI application that has been used to solve real-world complex mission-critical control tasks using the offline RL approach.

1 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.

[*]Equal contribution.

However, optimizing the combustion efficiency of TPGUs is an extremely challenging task. The difficulties arise from several aspects. First, TPGUs are highly complex and large systems, which contain lots of equipment and complicated operation mechanisms. A typical 600MW TPGU has more than 10,000 sensors. The combustion process alone involves 70$\sim$100 continuous control variables, which induce extremely large action space for control optimization. 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 $\mathrm{NO_x}$ 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 (proportional-integral-derivative) controller (Astrom and Hagglund, 2006) and model predictive control (MPC) algorithms (Garcia et al., 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 cost an extremely large amount of time to solve for optimal solutions. Hence most existing combustion optimization approaches decompose the TPGU into individual small sub-systems that only involve a limited amount of state and control variables (Kalogirou, 2003; Lee et al., 2007; Ma and Lee, 2011; Liu and Bansal, 2014). These treatments oversimplify the system, which is insufficient for fully modeling the complex combustion process in a TPGU.

The recent advances of deep reinforcement learning (RL) provide another promising direction. Deep RL leverages expressive function approximators (deep neural networks) and has achieved great success in solving complex tasks such as games (Mnih et al., 2013; Silver et al., 2017) and robotic control (Levine et al., 2016). However, all these achievements are restricted to the online RL domain, where agents are allowed to have unrestricted interaction with real systems or perfect simulation environments. In real-world industrial control scenarios, especially for a mission-critical task like control optimization for TPGUs, 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. Furthermore, most real-world industrial systems are overly complex or partially monitored by sensors, which makes it impossible to build a high-fidelity simulation environment for training online RL models. Fortunately, industrial systems like TPGUs have long-term storage of the operational data collected from sensors, resulting in large logged datasets. So our problem becomes, how to learn an optimized control strategy for TPGUs using RL with only logged dataset, simultaneously considering all the safety constraints, but without interacting with the real system.

The recently emerged offline RL provides an ideal framework 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 et al., 2019; Kumar et al., 2019; Wu et al., 2019; Yu et al., 2020) is to restrict policy learning stay "close" to the data distribution, which avoids the potential extrapolation error when evaluating unknown OOD samples.

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 historical datasets and an imperfect simulator to learn RL policies under constraints. 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.

The main contributions of this work are summarized as follows:

- We present the first study that applies offline RL in a large, complex real-world industrial control scenario. A task with such scale and complexity has never been tackled in literature or practice.

- We develop a new AI system, named DeepThermal, to optimize the combustion efficiency of TPGUs. The system has been successfully deployed in multiple real thermal power plants in China.
- We design a new data-driven deep-learning based combustion process simulator, with its network design guided by prior knowledge of the actual physical dynamic processes in a TPGU.
- We propose a new model-based offline RL framework, namely MORE, which fully utilizes the information in a real dataset and the generalizability of a potentially imperfect dynamics model to achieve effective constrained offline RL policy learning.
- We propose several novel strategies in MORE for model-based offline RL algorithms, including restrictive exploration and hybrid training. These strategies effectively avoid the negative impact of simulated OOD data, while at the same time, allow sufficiently exploiting the generalizability of a learned dynamics model.
- We conducted real-world experiments in multiple power plants in China, which validate the effectiveness of DeepThermal. Extensive experiments on standard offline RL benchmarks also show that MORE has superior performance compared with the state-of-the-art offline RL algorithms.

2 Overview

2.1 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 (e.g. NO_x) 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 70∼100 major continuous control variables as well as the chemical properties of the coal, which is an extremely challenging task.

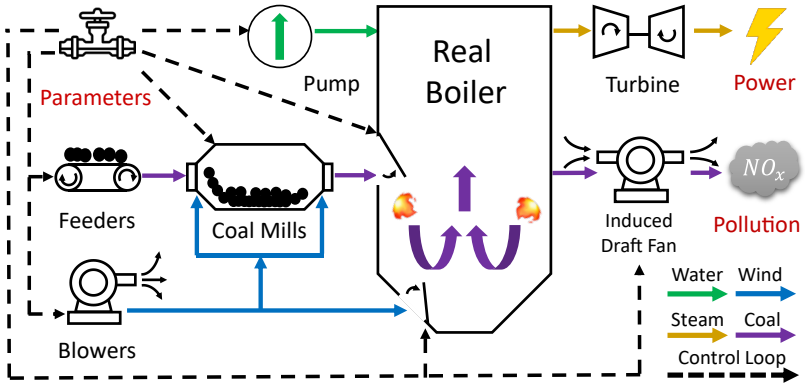


Figure 1: Illustration of operation mechanisms of a TPGU

2.2 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 combustion optimization 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 Section 2.1.
- **Actions** $\mathcal{A}$. We consider all the key control variables that impact combustion process in a TPGU as actions, such as the adjustment of the valves and baffles. All the actions are continuous.
- **Reward function** r. We model the reward as a weighted combination of combustion efficiency *Effi* and reduction in $\mathrm{NO_x}$ emission *Emi*, i.e. $r_t = \alpha_r \mathit{Effi}_t + (1 - \alpha_r)\mathit{Emi}_t$, where α_r can be adjusted according to the need of the 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 expert opinion.

In our combustion optimization problem, we assume no interaction with the actual TPGU and only have a static, offline 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 constraints of the expected discounted combined costs $C(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \tilde{c}(s_t, a_t)]$ below a given threshold l, mathematically:

$$\begin{aligned} \pi^* &= \arg\max_{\pi} \; R(\pi) \\ \text{s.t.} \quad & C(\pi) \leq l \end{aligned} \tag{1}$$

2.3 Overall System Framework

DeepThermal consists of two parts: *offline learning* and *online serving*, which is illustrated in Figure 2. Due to the complexity of the combustion process in a TPGU, it is impossible to build a high-fidelity simulation environment. Solely using 1 or 2 years' operational data may not be sufficient to find the optimized control strategy. In the offline learning part, DeepThermal learns a data-driven simulator and adopts a new model-based offline learning framework (MORE) to combat the limited data issue. The combustion simulator is used to provide supplement dynamics data to facilitate RL training. It is also used to generalize beyond the existing stereotyped control strategies of human operators recorded in data. However, as the simulator is learned from data, we do not fully trust the simulated data and use them with extra caution. Inside MORE, we introduce several specially designed strategies, including restrictive exploration and hybrid training to filter problematic simulated data and introduce reward penalties on OOD samples to guide offline RL policy learning away from high-risk areas.

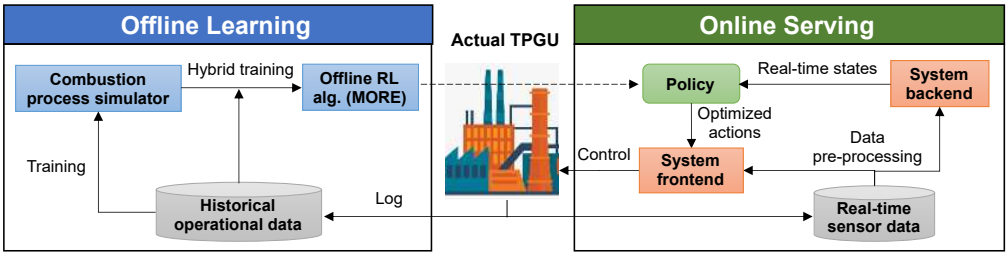


Figure 2: Overall framework of the DeepThermal system

During online serving, the learned RL policy outputs optimized actions according to the state information processed from real-time sensor data streams by the system backend. The system frontend displays the optimized control strategies to human operators, who adjust the combustion control of a TPGU.

As conditions of the equipment and devices inside a TPGU can change or deteriorate over time. DeepThermal is designed to be completely data-driven, which allows re-collecting the newly generated operational data from the TPGU for RL policy re-training and fine-tuning every few months. After every few months, we can re-collect the newly generated operational data of a TPGU to finetune the existing RL policy. This enables the models in DeepThermal to adapt to the current condition of the TPGU, providing an evolving optimization solution to a slowly changing system.

3 Combustion Process Simulator

DeepThermal learns a data-driven combustion process simulator from historical operational data of a TPGU, which serves as an approximated dynamics model $f(s_t, a_t) = \hat{s}_{t+1}$ to generated future states. Accurately fitting the dynamics of the 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 system state information is not fully captured by sensors. External factors like ambient temperature and chemical properties of the coal also impact combustion. Most importantly, the involvement of high-dimensional states and actions, complicated internal dependency structure among variables, combined with relatively limited historical operational data jointly pose great difficulty for the modeling process.

We propose a large 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 3, 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

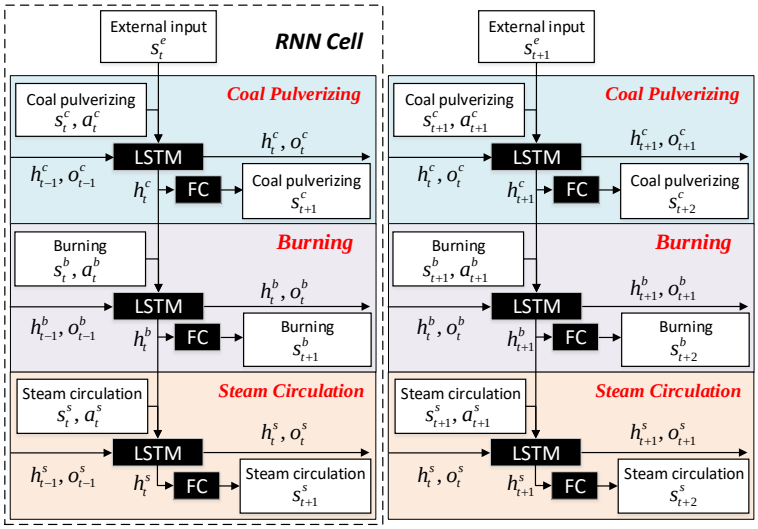


Figure 3: Design of the combustion process simulator

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:

- **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.
- **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.

4 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 4. 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.

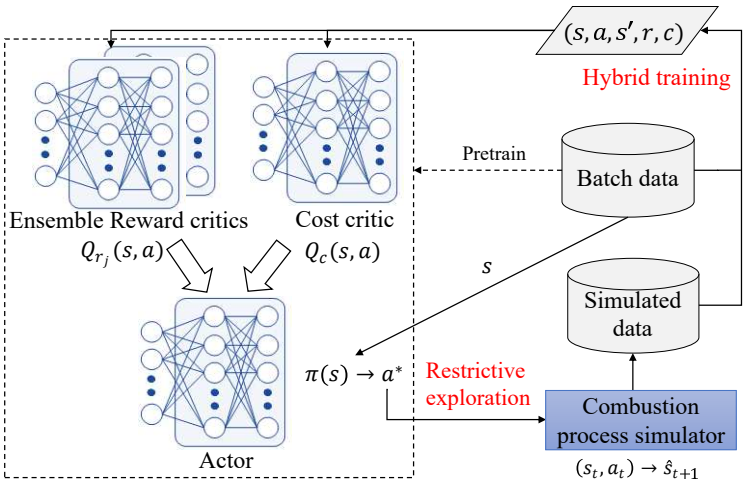


Figure 4: The framework of MORE

4.1 Safe Policy Optimization

MORE optimizes a policy to maximize long-term rewards while satisfying safety constraints requirements. It uses two Q-functions, Q_r, and Q_c, for reward maximization and cost evaluation. The policy optimization is performed on especially combined real-simulation data $\mathcal{D}$ (see hybrid training subsection for details) as follows:

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

MORE adopts the Clipped Double-Q technique (Fujimoto et al., 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 et al., 2004) by introducing following Lagrangian function:

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

where λ is Lagrangian multiplier. The original constrained 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).

4.2 Restrictive Exploration

Like many real-world tasks, we only have an imperfect simulator for the combustion optimization problem. The simulator/model is learned entirely from the 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 generalizable 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 $\epsilon_i \sim N(\mathbf{0}, \sigma\mathbf{I})$, $i \in \{1, \ldots, K\}$ on a input state-action pair (s, a) of the model, and compute the variance of the output perturbations $u = Var[\boldsymbol{\epsilon}^y]$ as the sensitivity metric, where $\boldsymbol{\epsilon}^y = \left[f((s,a) + \epsilon_i) - f(s,a)\right]_{i=1}^K$, 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 each training step t and $\mathbf{u}_{s,t}$ be the sensitivity of τ_s. MORE filters problematic simulated transitions using following strategy:

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

where l_u is a predefined threshold. In practice, we choose it to be the β_u-percentile value of sensitivity 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 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, 2013) 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:

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

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 of all state-action pairs in the offline dataset $\mathcal{B}$.

Algorithm 1 Restrictive exploration

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
for τ in τ_n **do**
  Set $\hat{s}_1 = s$, $\tau^+ = \emptyset$, $\tau^- = \emptyset$
  for Rollout step: $h = 1, .., H$ **do**
    Generate simulated transition $(\hat{s}_h, \pi(\hat{s}_h), \hat{s}_{h+1}, \hat{r}_h, \hat{c}_h)$, where $(\hat{s}_{h+1}, \hat{r}_h, \hat{c}_h) = f(\hat{s}_h, \pi(\hat{s}_h))$
    // Model sensitivity based filtering
    if Evaluated sensitivity $u(\hat{s}_h, \hat{a}_h) < l_u$ (follow Eq.5) **then**
      // Data density based filtering
      Compute data density $p_m(\hat{s}_h, \hat{a}_h)$ according to Eq.6
      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
      Add reward penalties for samples in τ^- according to Eq.8
    end if
  end for
end for
return (τ^+, τ^-)

4.3 Hybrid training

After quantifying the risks imposed by the imperfect simulator, MORE introduces a hybrid training strategy to solve two questions:

1. How to differentiate the impact of positive and negative simulated samples obtained from the restrictive exploration?
2. How to best blend real and simulated data for RL training?

In hybrid training, 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 from high-risk regions. We further combine the filtered simulated data with real data in a local buffer rather than a global replay buffer. 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) used penalized rewards to regularize policy optimization against the negative impact of OOD samples. Unlike prior works that penalize rewards of all simulation data to restrict policy update within data distribution, we propose a more delicate strategy by softly penalizing the rewards as:

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

where $[x]^+ = \max\{x, 0\}$ and η 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 based on 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 force policy updates away from the area of negative samples, so as to avoid potential exploitation error on OOD samples.

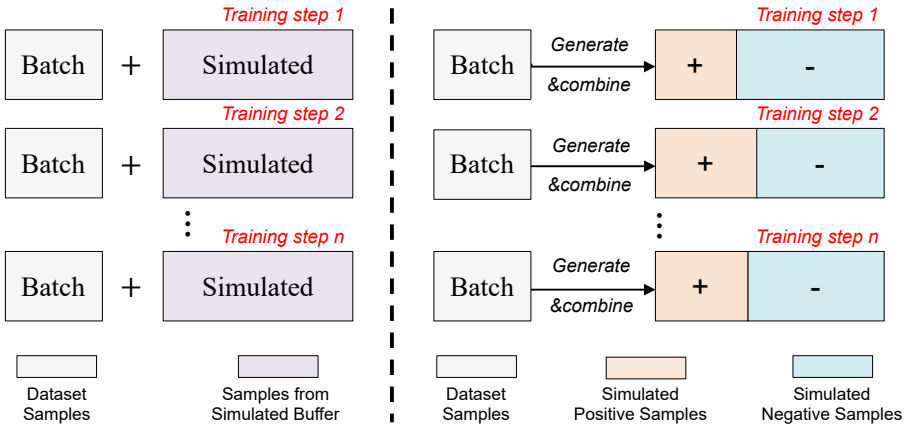


Figure 5: Illustration of hybrid training in MORE

Real-sim data combination. Rather than naïvely mixing real and simulated data as shown in the left part of Figure 5, MORE constructs a special local buffer $\mathcal{R}$ to combine real, positive and negative simulated data for training (right figure). As DeepThermal uses a RNN-based simulator, to improve the prediction accuracy, we first use trajectories in the real data batch to preheat and update the hidden states of the simulator, and then use it to roll out trajectories. We also control the proportion of positive and negative samples in the simulated data with the percentile threshold β_p. The choice of β_p controls the behavior of MORE to be either more aggressive to explore beyond the offline dataset (with a small β_p) or more conservative to avoid OOD errors (with a large β_p).

The full algorithm of MORE is summarized in Algorithm 2. Further implementation details can be found in the Appendix A.3.

Algorithm 2 Complete algorithm of MORE

Require: Offline dataset $\mathcal{B}$
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 $\phi'_c \leftarrow \phi_c$
for Training step: $t = 1, ..., T$ **do**
  Random sample mini-batch transitions τ_n from $\mathcal{B}$
  Obtain (τ^+, τ^-) using restrictive exploration (Algorithm 1)
  Construct local buffer $\mathcal{R} = \{(s,a,r,c,s')\}$ using τ^+, τ^- and τ_n
  Set $y = \min_{i=1,2} Q'_{r_i}(s', \pi(s'))$ and $z = Q'_c(s', \pi(s'))$
  Update Q_{r_i} by minimizing $(Q_{r_i} - (r + \gamma y))^2$
  Update Q_c by minimizing $(Q_c - (c + \gamma z))^2$
  Update policy π_θ by Eq.3 using policy gradient
  Update λ by Eq.4 using dual gradient ascent
  Update target cost critic: $\phi'_c \leftarrow \tau\phi_c + (1-\tau)\phi'_c$
  Update target reward critics: $\phi'_{r_i} \leftarrow \tau\phi_{r_i} + (1-\tau)\phi'_{r_i}$
end for

5 Experiments

In this section, we conduct extensive experiments on real-world TPGUs and standard offline RL benchmarks to demonstrate the effectiveness of DeepThermal and the superior performance of MORE compared with other state-of-the-art offline RL algorithms.

5.1 Dataset and Settings

We conduct experiments on both real TPGUs and standard offline RL benchmarks. The datasets and settings are described as follows:

Real-world datasets and experiment settings. DeepThermal uses 1∼2 years' historical operational data from a power plant to train its models. Very old data are not used due to potentially different

patterns compared with current conditions of the TPGU, caused by changes and deteriorating of equipment and devices. DeepThermal uses 800$\sim$1000 sensors' data from a typical TPGU. For example, in the system deployed in CHN Energy Nanning Power Station, we considered more than 800 sensors and optimized about 100 control variables. A specially designed feature engineering process is used to process these sensor data into about 100$\sim$170 states and $30 \sim 50$ actions (differ for TPGUs in different power plants). Some sensors 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 perform re-sampling on the processed state and action data into equal 20$\sim$30 second (depending on the quality of the sensor data) interval data, which typically results in 1$\sim$2 million records for RL training.

We present the results of real-world experiments conducted in CHN Energy Nanning Power Station. Additional results from CHN Energy Langfang Power Station are reported in the Appendix.

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. The datasets in D4RL are generated as follows:

- **mixed**: train a SAC policy (Haarnoja et al., 2018) until reaching a predefined performance threshold, and take the replay buffer as the dataset.
- **medium**: generated using a partially trained SAC policy to roll out 1 million steps.

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

5.2 Evaluation of the Simulator

We compare the performance of the combustion process simulator with six baselines that commonly used for time-series data prediction, including Auto-Regressive Integrated Moving Average (ARIMA), Gradient Boosted Regression Trees (GBRT), Deep Neural Network (DNN), Long Short Term Memory Network (LSTM) and Gated Recurrent Unit (GRU). Rooted mean squared error (RMSE) and the mean absolute error (MAE) are used for evaluation. The detailed evaluation results are presented in Table 1. It is observed that the proposed combustion process simulator significantly outperforms all the baselines on both evaluation metrics. This demonstrates the effectiveness of the proposed simulator, as well as the benefit of incorporating domain knowledge in the network design.

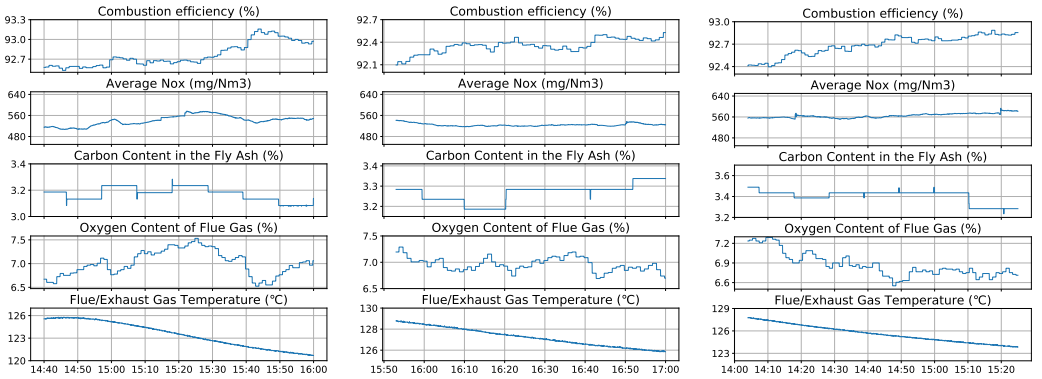
Table 1: Evaluation of the combustion process simulator

Model	ARIMA	GBRT	DNN	LSTM	GRU	Ours
RMSE	3.05e-3	1.97e-3	2.05e-3	1.69e-3	1.87e-3	**6.54e-4**
MAE	2.66e-2	2.65e-2	2.73e-2	2.50e-2	2.98e-2	**1.55e-2**

5.3 Real-World Experiments

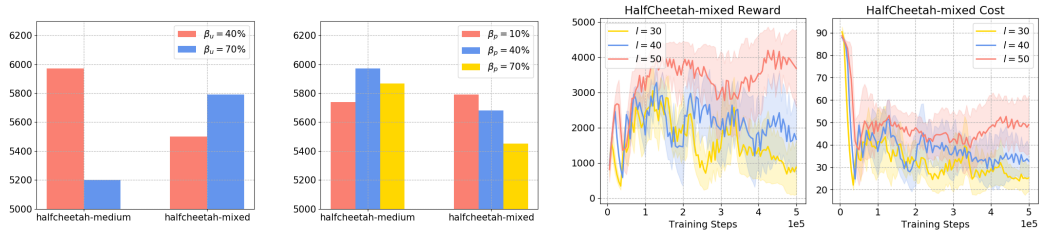
To verify the effectiveness of DeepThermal, we conduct a series of before-and-after tests on real-world TPGUs. The duration of these experiments ranges from 1 to 1.5 hours. During the experiment, the human operator adjusted the control strategy of a TPGU according to the recommended actions provided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

Figure 6 presents the experiment results on a TPGU of CHN Energy Nanning Power Station in three different load settings (270MW, 290MW, 310MW). It is observed that DeepThermal effectively improves combustion efficiency in all three load settings. In the 270MW, 290MW and 310MW 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 compared with the initial values. The average $\mathrm{NO_x}$ 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*. Carbon content in the fly ash is the remaining combustible carbon content in the fly ash of the furnace outlet, the oxygen content of flue gas measures the excess air content after combustion, and flue/exhaust gas temperature measures the



(a) 270 MW experiment (b) 290MW experiment (c) 310MW experiment

Figure 6: Real-world experiments at CHN Energy Nanning Power Station



(a) Ablation of β_u (b) Ablation of β_p (c) Evaluation of different safety constraints l

Figure 7: Ablation study on MORE

amount of heat loss. The lower of these three values, the more sufficient combustion is achieved. In all three experiments, these three indicators achieved a certain level of decrease, which provides clear evidence of combustion improvement.

Additional real-world experiment results on another power plant (CHN Energy Langfang Power Station) are reported in the Appendix A.2. due to space limit. DeepThermal achieved 0.51%∼0.65% increase in combustion efficiency after control optimization.

5.4 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 et al., 2019), BEAR (Kumar et al., 2019) and BRAC-v (Wu et al., 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 further compare against Behavior Cloning (BC) to examine whether our framework indeed performs effective RL, instead of simply copying strategies in data. 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

Table 2: Results for D4RL datasets, averaged over 3 random seeds

Dataset	BC	BEAR	BRAC-v	BCQ	MBPO	MOPO	MORE
halfcheetah-medium	4202.7	4513.0	5369.5	4767.9	3234.4	4972.3	**5970**
hopper-medium	924.1	**1674.5**	1031.4	**1752.4**	139.9	891.5	1264
walker2d-medium	302.6	2717.0	**3733.4**	2441.3	582.8	817.0	**3649**
halfcheetah-mixed	4488.2	4215.1	5419.2	4463.9	5593.0	**6313.0**	5790
hopper-mixed	364.4	331.9	9.7	688.7	1600.8	**2176.8**	**2100**
walker2d-mixed	518.5	1161.4	36.2	1057.8	1019.1	1790.7	**1947**

based reward penalties on OOD samples provide more reliable and informative simulated data for RL policy learning, to achieve 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 7(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 7(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 far from perfect, 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 discounted cumulative torque that the agent has applied to each joint. This mimics the situation that one constrains the robot from using high torque values to prolong their motor life. 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 only be able to learn a sub-optimal policy. We test MORE under different constraint limit $l \in \{30, 40, 50\}$. It can be shown in Figure 7(c) that MORE is robust to different l. In all the tests, the cumulative costs are controlled below the given constraint limits.

6 Real-World System Deployment

DeepThermal has already been successfully deployed in four large thermal power plants in China, including CHN Energy Nanning and Langfang Power Stations, Shanxi Xingneng Power Station and Huadian Xinzhou Guangyu Power Station. Real-world experiments have been conducted in all four power plants to test the effectiveness of DeepThermal. Our system achieves good results while ensures safe operation in all these four power plants, and has passed the project acceptance checks by industry experts, whom consider highly on the innovation as well as the effectiveness of our system.

The left figure in Figure 8 shows the main interface of the DeepThermal system deployed in CHN Energy Nanning Power Station. The interface displays real-time values of major states as well as optimized actions provided by the learned RL policy. The operator can easily follow the guidance of the recommended strategy to adjust their control operation, so as to improve system combustion efficiency. The right figure in Figure 8 shows the scene that the operator of the power plant using DeepThermal for reference to adjust the combustion control in the central control room. More detailed information about system implementation and deployment in other power plants can be found in the Appendix A.1.

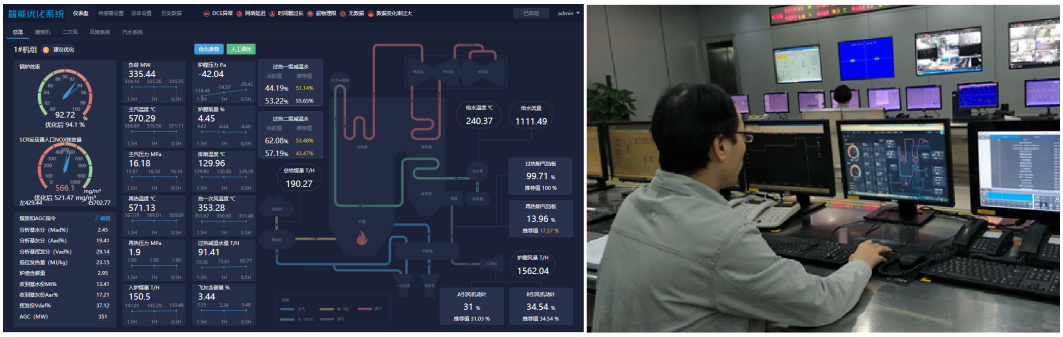


Figure 8: Interface of DeepThermal deployed in CHN Energy Nanning Power Station (left) and the control room (right)

7 Related Work

7.1 Complex System Control

PID control (Astrom and Hagglund, 2006) is the most common approach for industrial system control. Although PID ensures safe and stable control, its performance is limited due to insufficient expressive power and the inability to handle large systems. Model predictive control (MPC) (Garcia et al., 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. Many real-world MPC applications decompose the original optimization problem into meaningful smaller-scale sub-problems in order to achieve high frequency control.

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 high-fidelity simulators (Li, 2019), such as SUMO (Lopez et al., 2018) used in autonomous driving systems, Virtual-Taobao (Shi et al., 2019) used in recommendation systems, and MuJoCo (Todorov et al., 2012) used in robotic locomotion and manipulation tasks. However, high-fidelity simulators are impossible to obtain in most real-world complex system control 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).

7.2 Offline Reinforcement Learning

Offline RL (also known as batch RL (Lange et al., 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. Although off-policy RL methods (Mnih et al., 2013; Lillicrap et al., 2016) are naturally designed for tackling this problem, they typically fail to learn solely from fixed offline data, and often require a growing batch of online samples for good performance. Recent model-free methods attempted to solve this problem by constraining the learned policy to be "close" to the behavior policy. BCQ (Fujimoto et al., 2019) adds small perturbation to the behavior policy in order to stay close to the data distribution, BEAR (Kumar et al., 2019) and BRAC (Wu et al., 2019) incorporate additional distributional penalties (such as KL divergence or MMD). 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-strict behavioral constraints.

Model-based offline RL algorithms provide a nice solution to this problem, 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) learns an ensemble of dynamic models that output Gaussian distributions. It extends MBPO (Janner et al., 2019) with an additional reward penalty on generated transitions with large variance from the learned dynamic models. 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.

8 Conclusion and Perspectives

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.

Reliable control using RL in real-world scenarios is a challenging task that still needs a lot of research. Many open problems remain and worth further exploration, including 1) designing data-efficient offline RL policies under small offline datasets with limited state-action space coverage. 2) Finding better uncertainty and generalizability evaluation measures for models/simulators on unknown data samples. 3) Exploring new offline RL learning schemes to combat distributional shift while avoiding over-conservative police learning. 4) Developing better safety constraint modeling schemes to enforce strict constraint satisfaction. And 5) improving policy robustness with limited offline data and potential inaccuracies in the input data (e.g. random errors in the sensor readings. Incorporating adversarial learning into the offline RL training process is a possible direction to make policy more robust against random or targeted perturbations.

We hope our work can shed light on applying offline, data-driven RL algorithms to solve real-world complex system control tasks, where one could train RL algorithms offline using abundant logged system operational data, and provides reliable policies for safe and high-quality control.

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*, pages 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*, pages 1587–1596.

Fujimoto, S., Meger, D., and Precup, D. (2019). Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 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*, pages 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*, pages 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. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

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*, pages 11761–11771.

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

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*, pages 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*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *ICLR (Poster)*.

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)*, pages 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*, pages 1–8. IEEE.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

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, pages 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.

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*, pages 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)*.

A Appendix

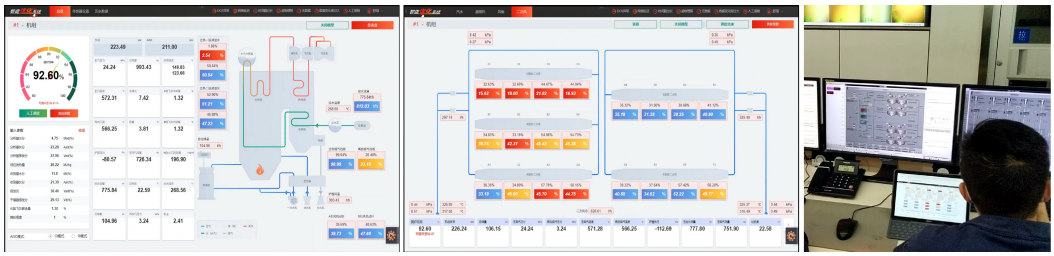
A.1 Detailed System Implementation

In this section, we provide additional information for the DeepThermal systems deployed in real-world thermal power plants. DeepThermal has been deployed and achieved some good results in four large coal-fired thermal power plants in China. Figure 9 presents the DeepThermal systems deployed in CHN Energy Langfang Power Station and Shanxi Xingneng Power Station.

The user interface of DeepThermal consists of an overview screen and several sub-interfaces displaying recommended control strategies for different combustion control stages (e.g. coal pulverizing, burning, air circulation and steamer system, etc.). The leftmost figures in Figure 9 show the overview screens of DeepThermal. Information that are not obtainable from sensors can be manually inputted in the bottom-left part in the overview screen, such as the chemical property of the coal. The figures in the middle are the sub-interfaces of the burning stage, which display recommended values for valves of the secondary blowers in the burner. Other sub-interfaces are not presented due to space limit. The rightmost figures in Figure 9 show the scene that the operator using DeepThermal to adjust their control strategy in the central control room.

DeepThermal displays two lines of values for each control variable in the interface. The value in the top line is the current control value. The value in the bottom with red, yellow or blue color marks the recommended value from the learned RL policy. The red or yellow indicate that the current control strategy has a large or medium deviation from the optimal policy, which should be adjusted. The operator in the central control room can easily adjust his control operation following the guidance of this system.

For each power plant, the user interface style of DeepThermal is customized to meet the needs of power plant clients. The interface layouts are also redesigned to match with the interface of the existing distributed control system (DCS) in the power plant. This allows operators easily locating the corresponding control element and adapting to the guidance from DeepThermal. Despite the differences in the system frontend, the RL algorithm module and system backend remain the same for different power plants.



(a) System deployed in CHN Energy Langfang Power Station

(b) System deployed in Shanxi Xingneng Power Station

Figure 9: System interfaces and control room usage of DeepThermal in two thermal power plants in China

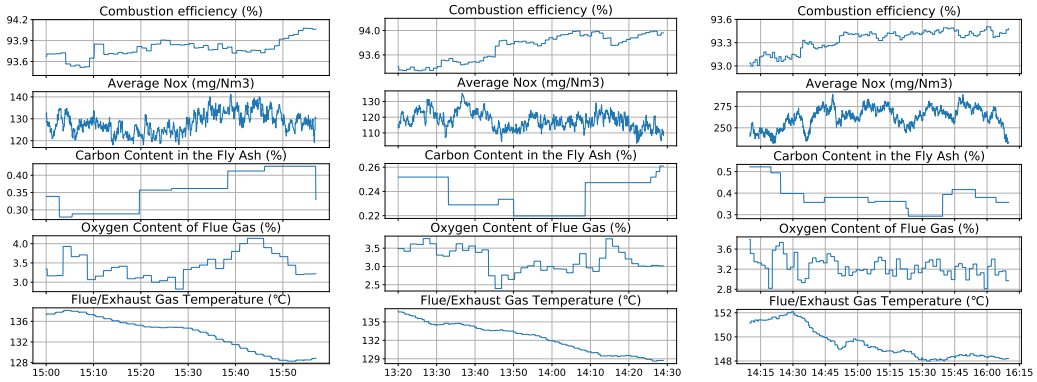
A.2 Extra Real-World Experiments

In this section, we report results from recent real-world experiments conducted at CHN Energy Langfang Power Station in Figure 10. DeepThermal system uses more than 700 sensors in the TPGU and optimizes the control strategy involving more than 70 control variables. See the trend analysis chart and the actual measurement record table for details.

Experiment (a) was conducted in the 250MW load setting on July 17, 2020. The test started at 15:20, and ended at 16:23. The optimized control strategy achieved the maximum increase of 0.56% on the combustion efficiency and the maximum decrease of 9.8°C on the exhaust gas temperature in about 60 minutes compared with the initial values. The average NO_x concentrations, the oxygen content of flue gas remain at a relatively stable level.

Experiment (b) was carried out at 14:36-15:55 on July 18, 2020 in the 200MW load setting. The initial value of combustion efficiency was 93.34%., and it reached 93.99% after the adjustment in about 60 minutes, the optimized control strategy achieved a maximum increase of 0.65%. The average NO_x concentrations reduced from $128.46mg/Nm^3$ to $118.61mg/Nm^3$, so the strategy achieved the maximum decrease of about $10mg/Nm^3$. The exhaust gas temperature also dropped from 136.7°C to 128.1°C. Other indicators also achieved a certain level of decrease.

Experiment (c) was conducted in the 300MW load setting on July 22, 2020. The test started at 14:10 and ended at 16:15. After optimization, the combustion efficiency rose from 93% to 93.51%, the optimized control strategy achieved the maximum increase of about 0.51%; The carbon content in the fly ash decreased from 0.56% to 0.38% in about 60 minutes with small turbulence. In three different load settings (200MW, 250MW,300MW), DeepThermal effectively improves combustion efficiency at a certain level and decreases the other three main indicators as well.



(a) 250 MW experiment (b) 200MW experiment (c) 300MW experiment

Figure 10: Real-world experiments at CHN Energy Langfang Power Station

A.3 Implementation Details

Hyper-parameters for Real-World Experiments. For the combustion process simulator, each LSTM cell has 256 hidden units. For the sequence to sequence structure, we set the encoder length as 20, and the decoder length as 10. The model is trained with the Adam optimizer and a learning rate of 1e-4. In the scheduled sampling, the probability of replacing the true data with the generated ones decays by 1e-4 for every training step. For the noisy data augmentation, the noises are sampled from $N(0, 0.025)$. The actor, critic and data density VAE are optimized using Adam and we use a fully connected neural network for function approximation of them. Their learning rate are $1\mathrm{e}-6$, $1\mathrm{e}-5$. and $1\mathrm{e}-3$. Both the encoder and the decoder of VAE have three hidden layers (1024,1024,1024) by default. We train the VAE for $1\mathrm{e}6$ timesteps with batch size 256. The policy and the critic have three hidden layers $(256, 128, 64)$. And they are trained for $1\mathrm{e}6$ timesteps with batch size 256.

Hyper-parameters for Offline RL Benchmarks. For all function approximators, we use fully connected neural networks with RELU activations. The pre-trained dynamics DNN has four hidden

layers (200, 200, 200, 200), the learning rate of DNN is $1e-4$. The pre-trained state-action VAE has two hidden layers (750, 750), the learning rate of VAE is $1e-4$. For policy networks, we use tanh (Gaussian) on outputs. The learning rate of Q-function is always $1e-3$. As in other deep RL algorithms, we maintain source and target Q-functions with an update rate 0.005 per iteration. We use Adam for all optimizers. The batch size is 256 and γ is 0.99. The actor network is 2-layer MLP with 300 hidden units on each layer and the reward and cost critic network is 2-layer MLP with 400 hidden units each layer. The learning rate of actor is $1e-5$. We search the model sensitivity threshold $\beta_u \in \{40, 70\}$ and the data density threshold $\beta_p \in \{10, 40, 70\}$. The rollout length are given in Table 3, we search the rollout length $H \in \{1, 5\}$ and use penalty coefficient $\eta = 5$.

Table 3: Rollout length H used in D4RL benchmarks

Env	halfcheetah		hopper		walker2d	
Type	medium	mixed	medium	mixed	medium	mixed
H	1	1	5	5	5	1