Towards a machine learning (ML) model for smarter electrolyte design

Jeonghwan Lee [*] Jaehee Park [†]

September 12, 2025

1 Introduction

To begin with, our main goal is to build a machine learning (ML) model which outputs an electrolyte with an *optimal performance* from the collection of all utilizable electrolytes given an input data consists of the pair of anode and cathode, together with several key factors that affect the performance of our main interest. In this proposal, we are mainly interested in the *capacity* of a battery for measuring performances of batteries, which quantifies how much charge it can store. In order to build an automation system for identifying an optimal electrolyte based on ML methods, we use the framework of *reinforcement learning* (RL) for system modeling. Towards this end, we first summarize key factors that influence the capacity of batteries:

(1) **Battery chemistry:** Different battery types (*e.g.*, lithium-ion, lead-acid, nickel-metal hydride) have inherently different energy densities. We consider *energy densities* as one of the quantitative variables that affect the capacity of batteries;

(2) **Temperature:** Battery performance is highly sensitive to temperature;

(3) **Discharge rate (a.k.a., C-rate):** The *faster* a battery discharges (higher C-rate), the *lower* the effective capacity due to internal resistance and inefficiencies;

(4) **State of Health (SoH):** As batteries age, their *state of health* deteriorates–internal components break down, increasing resistance and reducing capacity;

(5) **Depth of Discharge (DoD):** How deeply you discharge a battery affects its lifespan and effective capacity;

(6) **Manufacturing quality:** Variations in materials, electrode thickness, and electrolyte purity all influence the capacity of batteries;

[*]Department of Statistics at the University of Chicago. E-mail: jhlee97@uchicago.edu.
[†]Pritzker School of Molecular Engineering at the University of Chicago. E-mail: jaeheepark@uchicago.edu.

(7) **Physical size:** Larger batteries can hold more charge–this is especially relevant in applications like EVs or energy storage systems.

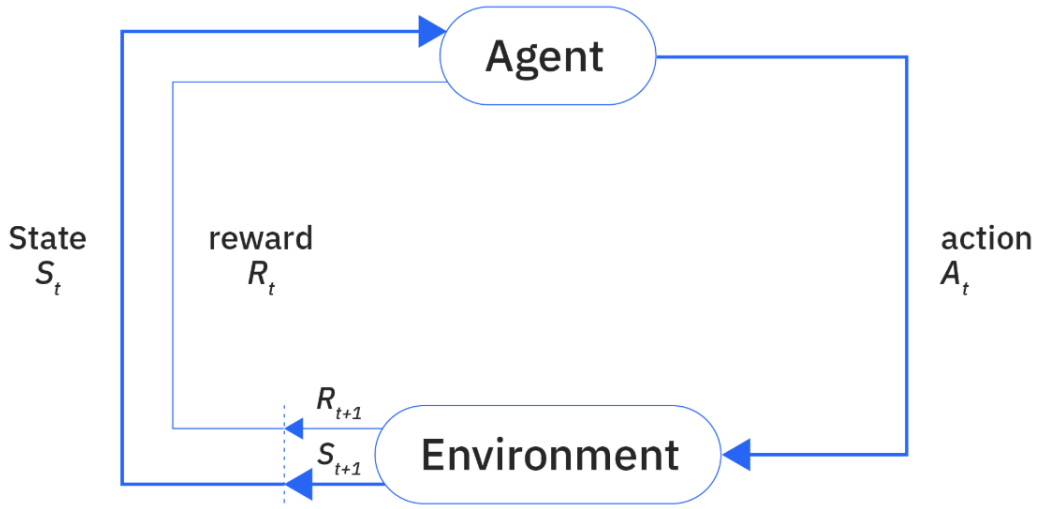
Now, let $\mathbb{S}$ denote the *state space* consists of all possible tuples of the form of

$$(\text{anode}, \text{cathode}, \text{energy density}, \text{temperature}, \text{C-rate}, \text{SoH}, \text{DoD}, \text{manufacturing quality}, \text{size}),$$

and we take a look at the charge-discharge cycles up to a fixed cycle number $H \in \mathbb{N}$. We will provide a precise definition of the *action space* $\mathbb{A}$ in order to formulate our framework using the language of RL in Section 3.2.

2 Prerequisites

2.1 Offline reinforcement learning for episodic finite-horizon MDPs



Environment: finite-horizon MDP. A *finite-horizon Markov decision process*, denoted by $\mathcal{M} := (\mathbb{S}, \mathbb{A}, H, \mathbb{P}, \mathbf{r})$, consists of the following key components:

(i) $\mathbb{S}$: the *state space*;

(ii) $\mathbb{A}$: the *action space*;

(iii) H: the horizon length;

(iv) $\mathbb{P} := (\mathbb{P}_h : h \in [H])$: the function $\mathbb{P}_h(\cdot,\cdot) : \mathbb{S} \times \mathbb{A} \to \Delta(\mathbb{S})$ denotes the *transition probability kernel* at the h-th step, where $\Delta(\mathbb{S})$ is the set of all probability distributions over the state space $\mathbb{S}$;

(v) $\mathbf{r} := (r_h : h \in [H])$: $r_h(\cdot,\cdot) : \mathbb{S} \times \mathbb{A} \to \mathbb{R}_+$ denotes the *reward function* at the h-th step.

In the language of batteries, $r_h(s,a)$ represents the capacity of the battery at the h-th cycle given a pair of $(s,a) \in \mathbb{S} \times \mathbb{A}$ of key factors $s \in \mathbb{S}$ and electrolyte $a \in \mathbb{A}$.

Agent: policy. A *policy* refers to a collection of action-selection rules $\boldsymbol{\pi} := (\pi_h : h \in [H])$ with functions $\pi_h(\cdot) : \mathbb{S} \to \Delta(\mathbb{A})$, such that $\pi_h(a|s) \in [0,1]$ specifies the probability of choosing the action $a \in \mathbb{A}$ when in state $s \in \mathbb{S}$ and the h-th step. Here, $\Delta(\mathbb{A})$ denotes the set of all probability distributions over the action space $\mathbb{A}$. When $\boldsymbol{\pi}$ is a *deterministic policy*, then we overload the notation and let $\pi_h(s) \in \mathbb{A}$ represent the action selected by $\boldsymbol{\pi}$ in state $s \in \mathbb{S}$ at the h-th step.

Evaluation: value function and Q-function. We introduce the value function $V^{\boldsymbol{\pi}} := \left(V_h^{\boldsymbol{\pi}} : h \in [H]\right)$ and the Q-function $Q^{\boldsymbol{\pi}} := \left(Q_h^{\boldsymbol{\pi}} : h \in [H]\right)$ associated with a policy $\boldsymbol{\pi} = (\pi_h : h \in [H])$. The value function $V_h^{\boldsymbol{\pi}}(\cdot) : \mathbb{S} \to \mathbb{R}_+$ of a policy $\boldsymbol{\pi}$ at step $h \in [H]$ is defined to be the expected cumulative reward from the h-th step on as a result of policy $\boldsymbol{\pi}$:

$$V_h^{\boldsymbol{\pi}}(s) := \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=h}^{H} r_t(s_t, a_t) \middle| s_h = s\right], \ \forall s \in \mathbb{S}, \tag{2.1}$$

where the expectation is taken over the randomness over the sample trajectory $\{(s_t, a_t) : t = h, h+1, \cdots, H\}$ when the policy $\boldsymbol{\pi}$ is implemented, i.e., $a_t \sim \pi_t(\cdot|s_t)$ and $s_{t+1} \sim \mathbb{P}_t(\cdot|s_t, a_t)$ for all $t = h, h+1, \cdots, H$. Correspondingly, the Q-function of a policy $\boldsymbol{\pi}$ at the h-th step is defined to be

$$Q_h^{\boldsymbol{\pi}}(s,a) := \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=h}^{H} r_t(s_t, a_t) \middle| (s_h, a_h) = (s,a)\right], \ \forall (s,a) \in \mathbb{S} \times \mathbb{A}. \tag{2.2}$$

It is well-known that there exists at least one *deterministic policy* that simultaneously maximizes the value function and the Q-function for all $(s,a,h) \in \mathbb{S} \times \mathbb{A} \times [H]$ [1]. In light of this, we shall denote by $\boldsymbol{\pi}^* := \left(\pi_h^* : h \in [H]\right)$ an *optimal deterministic policy*; this allows us to employ $\pi_h^*(s) \in \mathbb{A}$ to indicate the corresponding optimal action chosen in state $s \in \mathbb{S}$ at step $h \in [H]$. The resulting optimal value function and optimal Q-function are denoted respectively by

$$V_h^*(s) := V_h^{\boldsymbol{\pi}^*}(s) \quad \text{and} \quad Q_h^*(s,a) := Q_h^{\boldsymbol{\pi}^*}(s,a)$$

for every $(s,a,h) \in \mathbb{S} \times \mathbb{A} \times [H]$.

Offline/batch dataset. Suppose that we have access to a *batch dataset* (or *historical dataset*) $\mathcal{D}$, which comprises a collection of K i.i.d. sample trajectories generated by a *behavioral policy* $\boldsymbol{\pi}^{\mathsf{b}} := \left(\pi_h^{\mathsf{b}} : h \in [H]\right)$. More specifically, we have

$$\mathcal{D} := \left\{\tau_k := \left(s_1^k, a_1^k, r_1^k, s_2^k, a_2^k, r_2^k, \cdots, s_H^k, a_H^k, r_H^k\right) : k \in [K]\right\}, \tag{2.3}$$

which is generated by the underlying finite-horizon MDP $\mathcal{M} = (\mathbb{S}, \mathbb{A}, H, \mathbb{P}, \mathbf{r})$ under the behavior policy $\boldsymbol{\pi}^{\mathsf{b}}$ in the following manner:

$$a_h^k \sim \pi_h^{\mathsf{b}}\left(\cdot \middle| s_h^k\right), \quad s_{h+1}^k \sim \mathbb{P}_h\left(\cdot \middle| s_h^k, a_h^k\right), \quad \text{and} \quad r_h^k = r_h\left(s_h^k, a_h^k\right), \quad 1 \le h \le H. \tag{2.4}$$

Here, we note that we cannot collect additional sample trajectories via further *online* interaction with the underlying MDP.

GOAL. With the historical dataset $\mathcal{D}$ in (2.3) in hand, our goal is to compute a policy $\hat{\boldsymbol{\pi}} := (\hat{\pi}_h : h \in [H])$ that results in near-optimal values, i.e.,

$$V_1^*(s) - V_1^{\hat{\boldsymbol{\pi}}}(s) \leq \varepsilon, \ \forall s \in \mathbb{S}, \tag{2.5}$$

with high probability using as few samples as possible, where $\varepsilon > 0$ denotes the target accuracy level.

3 Design of an automated system to find an optimal electrolyte

3.1 Phase I: Assemble "experimental" datasets

Step 1. Collect electrochemical datasets: We first aggregate electrochemical datasets based on our own experimental results or from the literature with the aid of modern large language models (LLMs) such as ChatGPT or Perplexity.

Step 2. High-throughput imaging data collection: At this step, we assemble a dataset of battery material images with high throughput, e.g., microscopy of electrode surfaces, X-ray CT of cells, spectroscopic maps of composition, etc. This may involve a systematic series of imaging experiments, for instance: utilizing X-ray CT to scan dozens of cathode electrodes after various cycle counts, or performing cross-sectional FIB-SEM imaging on cells with different electrolyte formulations to visualize Lithium dendrite morphology in Lithium metal batteries.

Step 3. Image feature extraction via Vision ML: To augment imaging datasets (**Step 2**) to electrochemical datasets (**Step 1**), we need to extract key representations (or features) from imaging datasets by utilizing vision ML methods. In order to do so, we train a computer vision model to analyze imaging datasets quantitatively. One approach is to learn meaningful representations via *contrastive learning* (CL) [5, 3]. State-of-the-art CL models encompass *contrastive language-image pre-training* (CLIP) [7] and *knowledge DIstiliation with NO labels* (DINO) [2]. This enables us to connect image features with text descriptors – this could allow some level of semantics to be attached to imaging datasets. Alternatively, one can utilize the *vision transformer* (ViT) [4] or convolutional neural network (CNN) to perform tasks such as segmentation (e.g., identify and segment particles, pores, cracks, dendrites) and feature computation (e.g., size distribution of cracks, tortuosity of pores, thickness of deposited lithium).

Step 4. Augment extracted features from imaging datasets to electrochemical datasets: Finally, we take extracted features from imaging datasets from **Step 3** collectively together with electrochemical datasets from **Step 1**.

Of course, we can collect additional datasets by performing more experiments in hand. However, since our main goal throughout this proposal is to develop an automated system to identify optimal electrolytes built upon modern ML methods, we would like to generate additional datasets with the aid of *generative models*.

3.2 Phase II: High-throughput screening of electrolytes using machine learning (ML)

The main goal of the current phase is to narrow down the set of candidate electrolytes, which plays a role as the *action space* in **Phase IV** (see Section 3.4), via high-throughput screening of electrolytes. *High-throughput screening of electrolytes* using automation and ML methods is one major theme in the recent literature of battery research. This research theme is motivated from the fact that traditional *trial-and-error electrolyte design* is too slow for the vast chemical design space. In order to address this limitation, researchers have developed automated *high-throughput experimentation* (HTE) platforms for liquid electrolytes. In [8], the authors introduced a modular robotic system for formulating and testing non-aqueous electrolyte mixtures with the aid of a software pipeline called *Liquid Electrolyte Composition Analysis* (LECA). It automatically prepares electrolyte samples, measures key properties (e.g., ionic conductivity and oxidative stability), and then exploits an ML workflow in order to analyze the dataset. Various ML models (such as linear regression, random forest, neural networks, and Gaussian processes) are trained in parallel on the HTE dataset together with built-in cross-validation and uncertainty estimation. The system selects the model with superior performance, and makes use of it to predict which new electrolyte compositions will maximize ionic conductivity, thereby closing the loop between experiments and predictions.

Beyond one-shot predictions, researchers are also making efforts to adopt *closed-loop optimization* strategies. In [6], the authors established an *ML-guided* HTE robotic platform to optimize electrolyte solvents in a closed-loop fashion. In this case study, they focused on a redox-active organic molecule (2,1,3-benzothiadiazole) and sought solvents that maximize its solubility. They developed an active learning loop by combining an HTE module (which autonomously prepares and measures saturated solutions via a robotic system) together with a Bayesian optimization (BO) module. After each round of experiments, a surrogate ML model is updated on the new solubility dataset and then an acquisition function suggest the next set of solvent formulations for testing. This BO allows to efficiently navigate the solvent space by balancing exploitation vs. exploration. By iterating HTE module with BO recommendation, the system quickly identifies solvents with superior solubility, illustrating how *active learning accelerates electrolyte discovery*.

To summarize, high-throughput electrolyte screening demonstrates the value of generating large datasets with high-quality and leveraging ML methods to identify optimal electrolyte compositions far more efficiently compared to brute-force tests. Here, let us focus on leveraging these advantages of HTE electrolyte screening in order to narrow down candidate electrolytes. Let $\mathbb{A}$ denote the *action space* consists of all electrolyte candidates obtained through HTE electrolyte screening process which demonstrate superior key properties of electrolytes such as solubility, ionic conductivity, and oxidative stability. Also, let $\mathcal{D}$ denote the experimental dataset consists of samples whose electrolyte belongs to the *screened action space* $\mathbb{A}$. We finally conclude this section with reasons why we consider high-throughput electrolyte screening procedure to define the action space $\mathbb{A}$ of our interest:

(i) If we set the space of all electrolyte candidates with no screening processes as the action space for running an offline RL algorithm (Section 3.4), then the algorithm will return a deterministic

policy that could lead to a situation where one might select an electrolyte candidate with poor key properties. It would be undesirable to choose an electrolyte candidate with high capacity but poor performance in some key properties.

(ii) For MDPs with finite action space, smaller size of the action space leads to faster exploration of the action space in general. Hence, narrowing down electrolyte candidates via high-throughput electrolyte screening procedure could lead to faster implementation of offline RL algorithms in Section 3.4.

3.3 Phase III: Produce "imitative" datasets via generative modeling

In the third phase, one aims to create additional dataset that is similar to the experimental dataset $\mathcal{D}$ obtained in Section 3.1 using generative models. In brief, a *generative model* is an ML model designed to create new dataset whose *distributional behavior* is similar to its training dataset. More specifically, generative models learn the patterns and distributions of training data, and apply those understandings to generate novel content in response to new input data. Formally, let $\mu_{\mathsf{b}} \in \Delta\Big((\mathbb{S} \times \mathbb{A} \times \mathbb{R}_+)^H\Big)$ be the joint distribution of the sample trajectory

$$\tau_k := \Big(s_1^k, a_1^k, r_1^k, s_2^k, a_2^k, r_2^k, \cdots, s_H^k, a_H^k, r_H^k\Big), \quad k \in [K],$$

which is generated according to the finite-horizon MDP $\mathcal{M}$ under the behavioral policy $\boldsymbol{\pi}^{\mathsf{b}} = \Big(\pi_h^{\mathsf{b}} : h \in [H]\Big)$. In other words,

$$\mathcal{D} = \{\tau_k : k \in [K]\} \overset{\text{i.i.d.}}{\sim} \mu_{\mathsf{b}}(\cdot).$$

However, we have no exact knowledge of μ_{b} as we don't have any exact information about the underlying finite-horizon MDP $\mathcal{M}$. Thus, in order to generate new samples that imitate the distributional behavior of the batch dataset $\mathcal{D}$, we should first learn the joint distribution μ_{b} by training a generative model on the training dataset $\mathcal{D}$. For example, one can make use of *diffusion models*, also known as *diffusion probabilistic models* or *score-based generative models* (SGMs), in order to learn the joint distribution μ_{b}. In general, a diffusion model consists of three major components: (i) the forward process, (ii) the reverse process, and (iii) the sampling procedure.

Let $\hat{\mu}(\cdot) \in \Delta\Big((\mathbb{S} \times \mathbb{A} \times \mathbb{R}_+)^H\Big)$ denote the distribution learned by generative models. State-of-the-art diffusion models are known to achieve $\hat{\mu} \approx \mu_{\mathsf{b}}$. With the trained generative model $\hat{\mu}$ in hand, we now generative sufficiently many new samples

$$\tilde{\mathcal{D}} := \Big\{\tilde{\tau}_k : k \in \Big[\tilde{K}\Big]\Big\} \overset{\text{i.i.d.}}{\sim} \hat{\mu},$$

where $\tilde{K} \gg K$. Hereafter, we would like to utilize $\overline{\mathcal{D}} := \mathcal{D} \cup \tilde{\mathcal{D}}$ as a new batch dataset to run offline RL algorithms in the next section.

3.4 Phase IV: Build an automated optimal electrolyte identifier via offline RL

In our fourth phase, we want to compute a near-optimal policy $\hat{\boldsymbol{\pi}} = (\hat{\pi}_h : h \in [H])$ by running an offline RL algorithm on the new offline dataset $\overline{\mathcal{D}}$. Here, one can use various offline RL algorithms including

the off-policy policy gradient method and the value-iteration algorithm or the Q-learning algorithm by incorporating a principle called the *pessimism in the face of uncertainty*. At this point , we note that the majority of offline RL algorithms return a *deterministic policy* as an output. With the deterministic near-optimal policy $\hat{\boldsymbol{\pi}} = (\hat{\pi}_h : h \in [H])$ in hand, we finally have an automated system for identifying optimal electrolytes. For any given state $s \in \mathbb{S}$ that describes information about the electrodes and key environmental factors of our current interest, $\hat{\pi}_h(s) \in \mathbb{A}$ will represent the electrolyte that is likely to demonstrate the highest capacity of the battery at the h-th cycle.

3.5 Phase V: Verification

In the final stage, we need to go through an *experimental verification step* to verify whether the near-optimal deterministic policy $\hat{\boldsymbol{\pi}} = (\hat{\pi}_h : h \in [H])$ obtained via offline RL algorithms in Section 3.4 actually yields a capacity close to the optimal. In order to do so, for any given state $s \in \mathbb{S}$ that describes information about the electrodes and key environmental factors of our current interest, we compare the capacity of the cell using the electrolyte $\hat{\pi}_h(s) \in \mathbb{A}$ to the capacity of cells which use other several electrolytes at the h-th cycle.

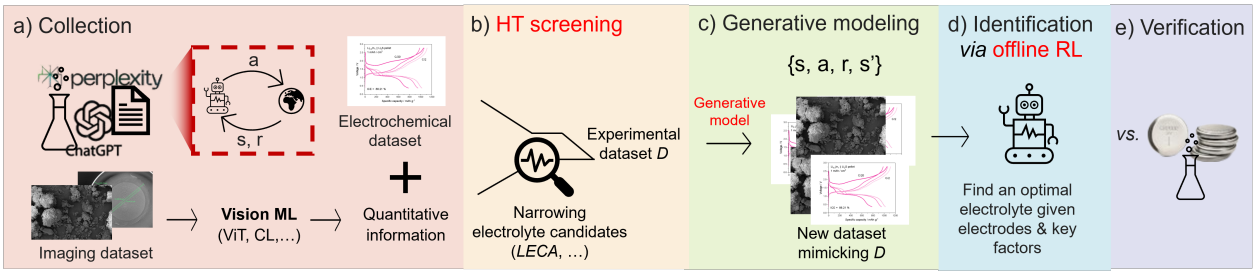


Figure 1: An overview of our proposed automated system to find an optimal electrolyte via ML.

References

[1] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[5] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[6] Juran Noh, Hieu A Doan, Heather Job, Lily A Robertson, Lu Zhang, Rajeev S Assary, Karl Mueller, Vijayakumar Murugesan, and Yangang Liang. An integrated high-throughput robotic platform and active learning approach for accelerated discovery of optimal electrolyte formulations. *Nature Communications*, 15(1):2757, 2024.

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[8] Peng Yan, Mirko Fischer, Harrison Martin, Christian Wölke, Anand Narayanan Krishnamoorthy, Isidora Cekic-Laskovic, Diddo Diddens, Martin Winter, and Andreas Heuer. Non-aqueous battery electrolytes: high-throughput experimentation and machine learning-aided optimization of ionic conductivity. *Journal of Materials Chemistry A*, 12(30):19123–19136, 2024.