Artificial Intelligence-Assisted Experimental Optimization of Water Oxidation Catalysts

Henrik Spitzenpfeil, Marius Neumann, Nick Hausen, Regina Palkovits and Stefan Palkovits*

DOI: 10.1002/cite.202400156

Artificial intelligence (AI) methods are very often used to make predictions for datasets that were created externally in arbitrary experiments or on already literature known datasets. In this work, we try to make use of active learning techniques to search for an optimal strategy for the startup-phase of bulk nickel electrodes in the oxygen evolution reaction. The data collected was afterwards reduced in dimensions and used to extract additional information that were learned via an artificial neural network (ANN) on the dataset, respectively.

Keywords: Active learning, Catalysis, Electrochemistry, Machine learning, Oxygen evolution reaction

Received: November 27, 2024; *revised:* January 16, 2025; *accepted:* February 03, 2025

1 Introduction: The Oxygen Evolution Reaction (OER) Activation Challenge

It is pretty obvious nowadays that we need to diversify the energy sector and that water splitting will contribute to the energy mix. No matter which kind of technology we favor for the oxygen evolution reaction (OER) in the future, being it alkaline or acidic water splitting, the challenge to tackle looks manageable on paper (Eq. (1)) but in reality the elementary steps taking place on the surface of an electrode are much more complex and therefore the chemistry is the topic of more than one recent article [1] discussing various aspects of water splitting [2].

$$2H_2O \rightarrow 2H_2 + O_2 \qquad (1)$$

So, which approaches can be used to gain more insights into water splitting, e.g., on nickel electrodes when they seem to be pretty sensitive with respect to their reactivity depending on the activation conditions [3]? One approach tries to keep industrial conditions in mind [4] leading to very reproducible results especially under harsh conditions. It is worth noting that there are many pitfalls when doing the experiments [5], and the precise protocols like the one from Thissen et al. [3] are time-consuming. To speed up the screening for reaction conditions one could think of a blend of high-throughput experimentation (HTE) [6, 7] and machine learning (ML) or artificial intelligence (AI) [8] and like in other disciplines also in catalysis there are already several approaches and overviews available that attempt combining approaches of the two neighboring fields [9–11].

We already tried to make some use of ML in catalysis especially with regards to water splitting [12] and ML in a broader perspective [13, 14]. We like to use these insights and tackle the experimental optimization of the startup behavior of bare nickel electrodes in alkaline electrolysis with AI approaches. But how can this be done when typically experimental data for such an optimization approach is scarce? We chose to generate the data on the fly while conducting the experiment itself with an active learning strategy (reinforcement learning, RL) and the overall method development will be described next.

2 Two Steps Forward, One Step Back - Method Development

The work on the topic went through several stages. They will be briefly explained as they are helpful to understand the method development. All stages of the development have in common that typical conditions for OER investigations were used. Unless stated elsewhere, the working electrode was a bare nickel electrode, the counter electrode was platinum. As reference electrode served an Ag/AgCl (3 M KCl) electrode, and the electrolyte was a 1 molar KOH solution. All

[1]Henrik Spitzenpfeil, [1]Marius Neumann, [1]Nick Hausen, [1,2]Prof. Dr. Regina Palkovits https://orcid.org/0000-0002-4970-2957, [1]Dr. Stefan Palkovits https://orcid.org/0000-0003-4809-2939 (stefan.palkovits@itmc.rwth-aachen.de)
[1]RWTH Aachen University, Heterogeneous Catalysis and Chemical Technology, Worringerweg 2, 52074 Aachen, Germany.
[2]Forschungszentrum Jülich GmbH, Institut für nachhaltige Wasserstoffwirtschaft 2, Marie-Curie Straße 5, 52428 Jülich, Germany.

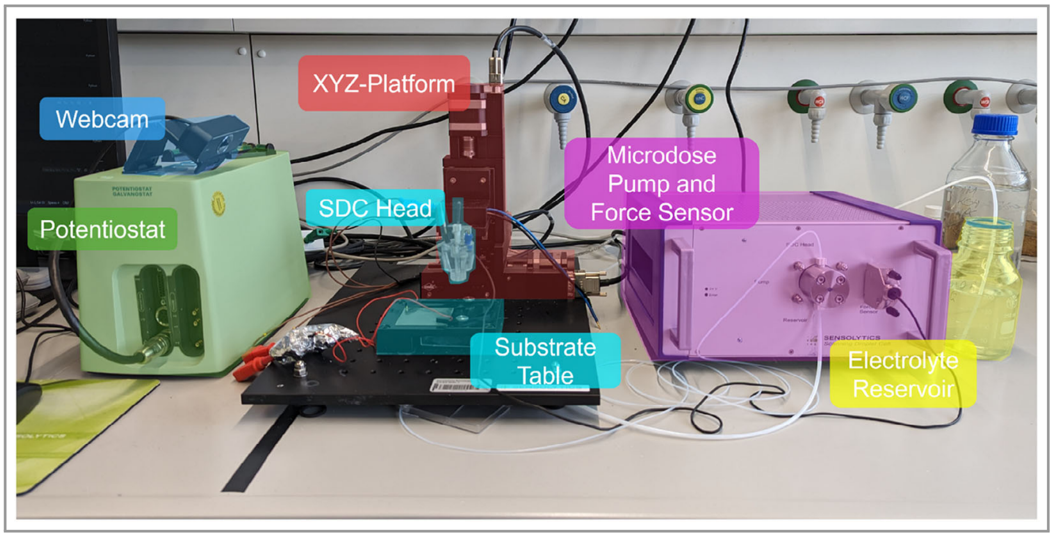


Figure 1. Scanning droplet cell used for the algorithmic optimization.

experiments were carried out at ambient temperature and pressure.

- Initial phase: In a first attempt we tried to establish a method for an optimized activation of bare nickel electrodes at all. Randomly generated potential profiles were imprinted on the electrodes and closely watched with active learning algorithms. The experiments were carried out in a self-built flow cell and resulted in hints on how to refine the strategy.
- Study on reproducibility: The initial phase in the flow cell yielded experimentally in two outcomes or advancements. On the one hand the random profiles were refined with profiles that were generated out of several sinusoidal profiles, and on the other hand the flow cell seemed to have heavy issues with respect to reproducibility. The latter was mainly addressed in the next step but could not completely be solved by the change in the experimental strategy.

To really move one step forward in method development we moved away from a flow cell. The reproducibility issues were mainly due to the phase between two measurements. As each step in the optimization should start again with a clean electrode, the regeneration step in between has to be very thorough. Up to now we have not managed to fully optimize this cleaning step. The next important development is depicted in Fig. 1.

Instead of a flow cell a scanning electrochemical microscope (SECM) equipped with a droplet cell (Sensolytics GmbH) was used. Apart from the *xyz*-stage the setup consists of a potentiostat (Metrohm PGSTAT204) and a pump together with a force sensor (Sensolytics). This enabled us to scan over a bulk nickel plate and have a fresh spot of material for each experiment. As the droplet cell has an opening of 2 mm and the electrodes have a size from 50 mm × 50 mm, up to about 250 measurements can be done without exchanging the electrode. To ensure an always fresh measuring environment, the 1 molar KOH is exchanged from the cell between the measurements.

For the control of the automated setup, several stages of software packages are developed that work together:

- autolab.py: This package is like all the rest written in Python [15]; it is the foundation and controls the potentiostat. It serves as an interface to the Metrohm Autolab SDK.
- langpy.py: The langpy package is used to control the stepper motor via the motor controller API.
- secm.py: In the secm package not only the control of the pump and the force sensor is implemented but also an abstract workflow for the SECM and its experiments.
- aec.py: The aec ("Artificial Electrochemist") package finally takes care of all the experiments and implements the AI with its algorithms.

All can be found in an online repository [16]. To qualify the setup and the packages written, a set of 100 linear sweep voltammograms were recorded on a bulk nickel plate in 1 molar KOH with a scan rate of 5 mV s^{-1} (Fig. 2, left). The measurements show a good reproducibility before a probable mass transfer limitation starts. Especially the area around the onset potential of the OER can be regarded as stable. A histogram of the onset potentials (Fig. 2, right) shows a narrow distribution around 1.64 V leading to an overpotential of around 0.398 V for all experiments. With the strategy to move from a flow cell to a droplet cell, the reproducibility issues seem to be compensated. Experimentally this leaves the challenge of the mass transfer limitation in the higher potential region which could maybe be resolved by pumping the electrolyte through the droplet cell but this is beyond the scope of this manuscript.

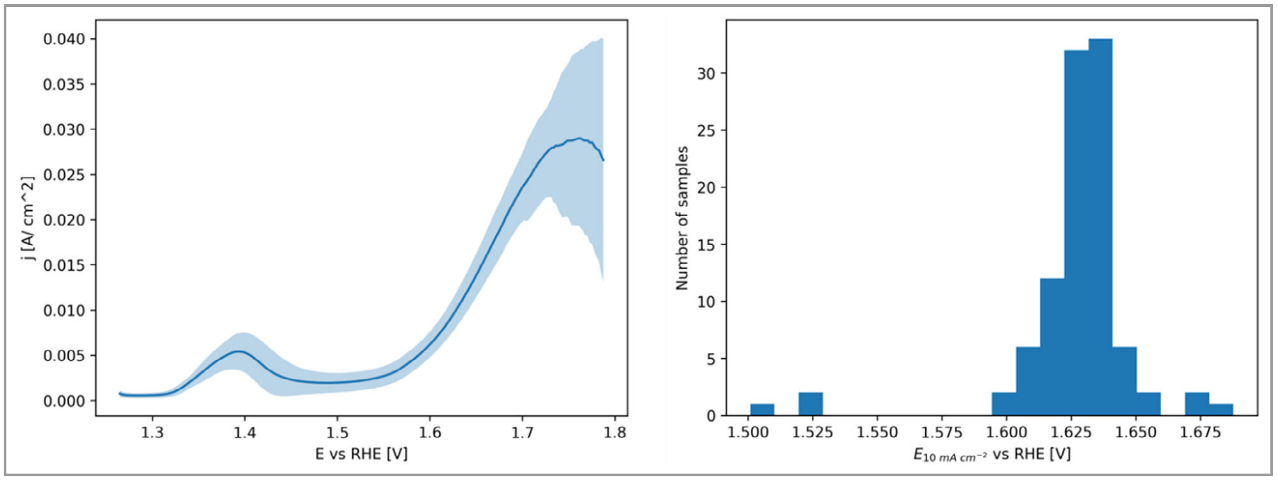


Figure 2. Cyclic voltammetry of 100 experiments on a bulk nickel plate in 1 M KOH with 5 mV s^{-1} and its error in blue (left) and a histogramm of the resulting onset potentials (right).

3 Active Learning to Optimize the Conditioning Process

With the experimental prerequisites in place, we tried now to make use of active learning strategies to optimize the OER conditioning phase on nickel. Some part of active learning is also called reinforcement learning (RL) and its principle is illustrated in Fig. 3.

The basic idea is that an agent carries out an action on a system. This agent then sees what happened to the system in question and depending on the outcome gets a reward. The agent then optimizes its actions to get a maximal reward [17, 18]. For RL to work the transition from one state to the next has only to be affected by the action and the original state (Markov decision process) [18, 19]. The reward is constructed in a way that it has its maximum when the task to optimize, here the conditioning phase of the OER, is sufficiently fulfilled. To do so, two different agents were implemented (Fig. 4).

One is a simple agent (SA) which directly gets back all observations after the action taken. The second agent is a time-resolved agent (TRA) which gets a time average over 300 episodes of the optimization. Both agents were based on Deep-Q algorithms which means that it is a model-free optimization without any knowledge of electrochemistry like, e.g., the Butler-Vollmer equation. Both agents start at a potential of 1.31 V and control the potential for 1500 steps with a length of 0.4 s per step. This is then repeated for several hundred episodes. The minimal potential is set to 1.152 V, the maximal potential is set to 1.53 V. The learning rates were 0.0023 for the SA and 0.0001 for the TRA. The neural networks used have in both cases two hidden layers with 256 neurons each. The agents were implemented in a “Gymnasium” environment (OpenAI Gym) together with the stable baselines 3 package which implements both Deep-Q agents [20, 21].

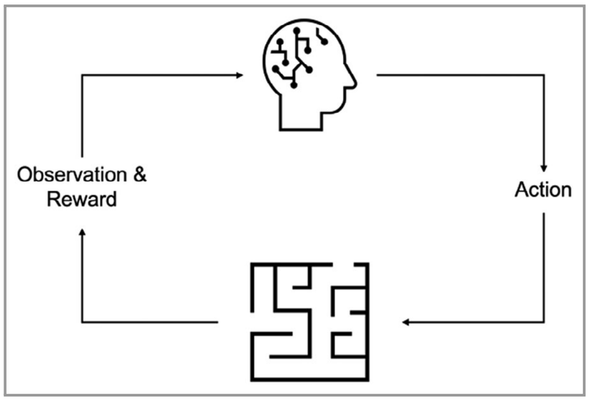
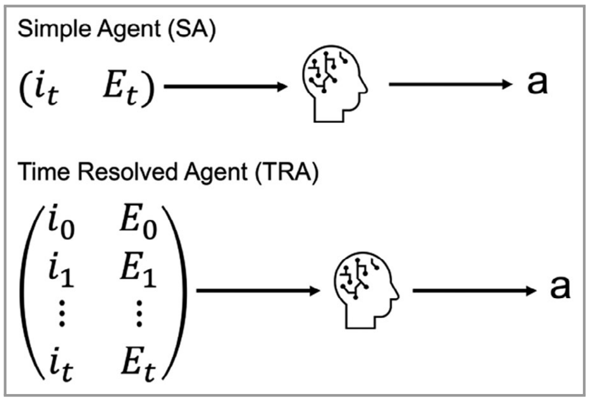


Figure 3. Principle of active/reinforcement learning.



Simple Agent (SA)
$(i_t \quad E_t)$
a
Time Resolved Agent (TRA)
$\begin{pmatrix} i_0 & E_0 \\ i_1 & E_1 \\ \vdots & \vdots \\ i_t & E_t \end{pmatrix}$
a

Figure 4. Agents used for the active learning strategies. The simple agent (SA, top) gets all observations at once, the time-resolved agent (TRA, bottom) gets a history of 300 episodes.

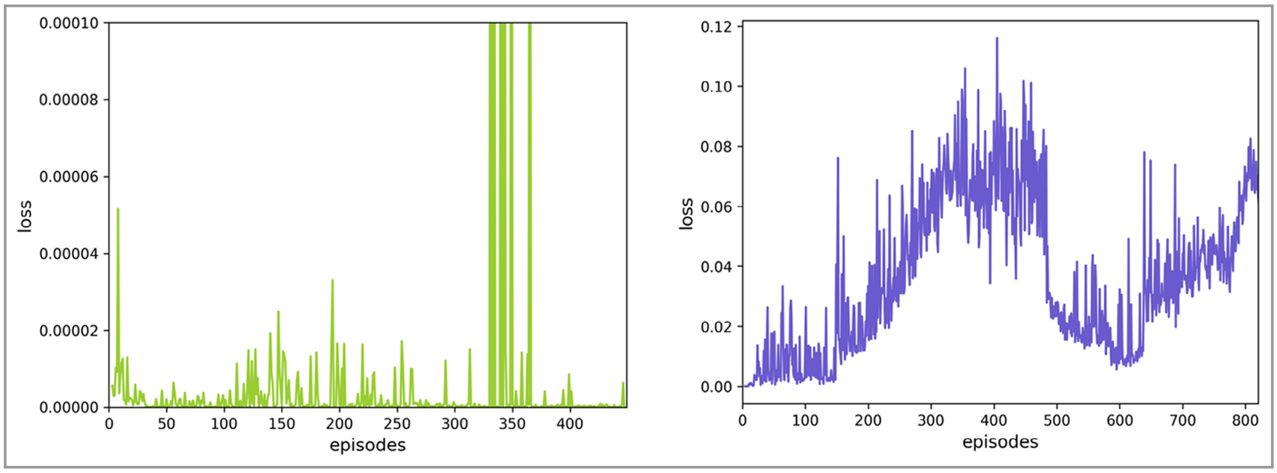


Figure 5. Loss curves of the simple agent (SA, left) and the time-resolved agent (TRA, right).

After training both agents, the resulting loss curves (Fig. 5) were analyzed. On the left side the loss curve of the SA is shown. There is no visible trend which leads to the conclusion that the SA was not able to learn anything from the training at all. There are just regular spikes showing up in the loss curve most probably because the agent only gets a reward at the end of each episode. The agent does not seem to associate any behavior of the system with respect to the rewards. The TRA on the other hand (Fig. 5, right) shows a completely different behavior in the loss curve. First, the loss rises roughly until episode 450. This is not very common as typically the loss starts at a certain level and then slowly decreases. Then the loss decreases which would be a hint for the algorithm to learn something from the actions and rewards undertaken. But then it rises again. As no more experiments were done, it is not clear if this might be, e.g., a periodic behavior. At least building a time average over the data seems to increase the information content and makes the algorithm learn from the active learning procedure.

The results of the learning phase of the TRA are summarized in Fig. 6. First, we tried to extract a time vs. potential profile from the measurements (Fig. 6, left) and surprisingly the algorithm seems to suggest to activate the electrode via a triangular potential profile which is in principle a cyclic voltammetry procedure like also proposed in many literature references [3]. The histogram of the onset potentials was again plotted for all experiments, and it peaks around 1.625 V leading to an average overpotential of 0.401 V for all 526 experiments compared to 0.398 V for the reference experiments from Fig. 2. The main advantage of the method is the time in which the optimization is executed. While the reference CVs take about 15 min each, the optimized CV takes only about 2 min for the whole conditioning process.

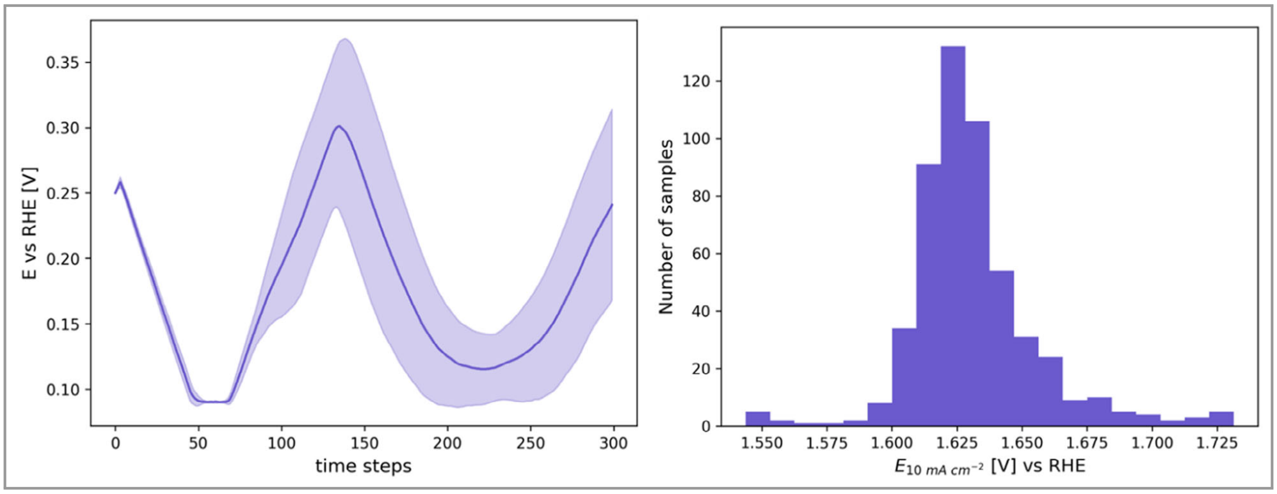


Figure 6. Results of the active learning procedure of the TRA. An extracted time vs. potential curve (left) and a histogram of the onset potentials (right).

One contribution to this short time is of course the time step and the overall episode length during the optimization.

4 Supervised Learning to Extract Further Information

After the active learning procedure, we were not completely satisfied with the outcome of the optimization. We therefore went for alternative approaches to extract data from the dataset collected with the active learning approach. But first the data had to be reshaped or reduced, respectively, as the data has a dimensionality problem. The dataset from the optimization contains finally 526 observations (table rows) but each observation contains 300 potentials (table columns). To make proper use of the data, the columns had to be reduced by roughly one order of magnitude ("curse of dimensionality") compared to the rows. So, before using supervised methods we had to employ unsupervised methods to reduce the dimensionality of the data.

To yield consistent results, Scikit-Learn [22] pipelines were used where always one dimensionality reduction algorithm was connected to an artificial neural network (ANN, MLPRegressor). Several methods were compared like principal component analysis (PCA) [23] with 100 principal components, K-means clustering with three clusters, and a window-based averaging approach with a window size of 11 features and a stride length of 4 yielding 72 features to reduce the data, respectively. The untreated raw dataset was added for comparison to the procedure. To yield optimal results, the ANN was optimized with the random search cross validation procedure from Scikit-Learn. The resulting optimal hyperparameters can be found in Tab. 1. The outcome of this optimization procedure is illustrated in Fig. 7.

Table 1. Best hyperparameters for each model as determined by random search cross validation.

Hyperparameter	Raw Data	PCA	K-Means	Window-based
Optimizer	adam	adam	lbfgs	lbfgs
Hidden layer size	[500, 500]	[500, 500]	[256, 256]	[500, 500]
α	1	1	1	0.01
ε	1e10^{-8}	1e10^{-9}	1e10^{-7}	1e10^{-7}

On the left-hand side, a comparison for the R^2 score of the predicted results of the ANN is displayed for the different dimensional reduction algorithms. The window-based approach clearly outperforms all other methods with a final score of about 0.99 for the training dataset and only a little lower 0.976 for the test set (train: 426 samples, test: 100 samples). The lower score for the test set also is a hint that not too much overfitting is occurring. To visualize the predictions of the best performing combination, a parity plot (Fig. 7, right) shows that all data points lie well along the diagonal, emphasizing the good quality of the final result.

Now with a decent predictability of the experimental dataset at hand, a procedure was needed to extract the information that is hidden in the ANN. Although nowadays methods exist to extract information from ANNs like going backwards through the layers [24], this was not possible with the ANN implementation out of Scikit-Learn and another approach had to be used. The proposed method is presented in Fig. 8.

The idea is to first train an ANN with a dataset like already shown (Fig. 7). The ANN uses a potential (voltage) profile as input variables to predict an overpotential. The ANN should now be able to predict any overpotential based on a provided potential profile. But how can we now extract a potential profile that leads to the lowest overpotential as

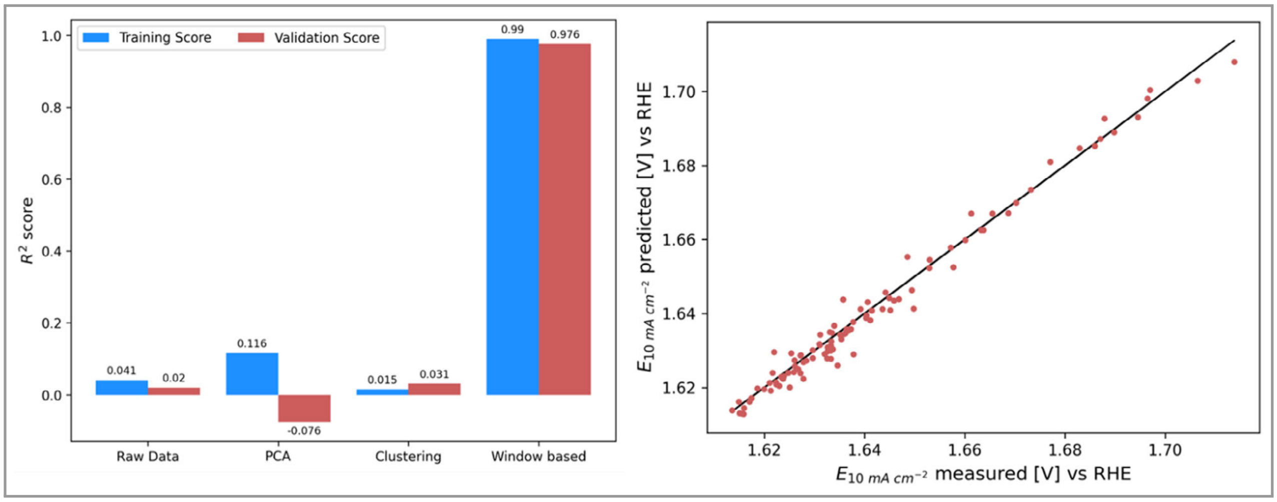


Figure 7. Resulting R^2 score for the compared dimensional reduction algorithms (right) and predicted results for the ANN after the optimized data reduction (right).

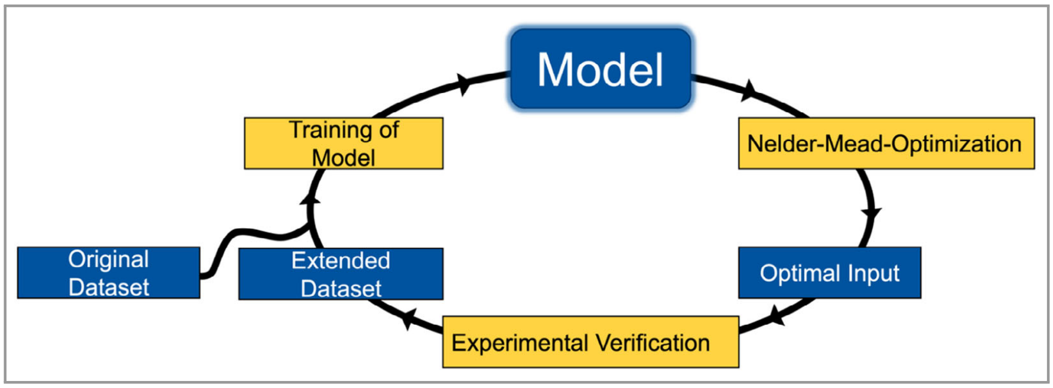


Figure 8. Procedure to make use of the hidden information in the ANN.

this information is still hidden in the ANN? We are looking for a minimum in the overpotential now, and a traditional algorithm like the Nelder-Mead-Algorithm (Simplex) [25, 26] should be well suited. The Simplex algorithm varies the voltage profile at the input until a minimum in overpotential is reached. The resulting voltage profile can be used for the experimental verification of the results and to extend the original data if you like so.

The extracted optimal voltage profile is shown in Fig. 9, left together with its experimental verification on the right. The time versus potential profile again shows the behavior of a cyclic voltammogram in the potential range between 1.15 and 1.55 V vs. RHE. The profile shows the typical cyclic potential that is often used for the conditioning of nickel electrodes, so we were slightly optimistic to also get a good prediction for the overpotential. With a prediction for the overpotential of 1.52 V we went into the experimental validation. Here (Fig. 9, right), 50 experiments were carried out and the resulting potential curves are shown on the right side of Fig. 9 together with the benchmarking experiments.

In the verification experiments some peculiarities can be seen. The peak at around 1.4 V vs. RHE which is often prescribed to the production of NiOOH [3] is not so well pronounced like before. In the higher potential range, there seems to be some mass transport limitation which fortunately is not that important for the outcome of this study. The most obvious observation is that the verification (red) and benchmarking (blue) experiments are well on top of each other, emphasizing the reproducibility of the method, but the overpotential predicted by our methodology is 1.52 V (cross) which is about 9 % off from the experiments. Regarding the aspect that this study is based on high-throughput experimentation, an error of around 9 % is still not bad but what are factors that contribute to this error?

The first aspect is that the determination of the overpotential yield sometimes errors during the experiments. This should be correctable in the future. More severe are the facts that with around 500 observations the dataset is still small. And the data for the supervised learning approach originates

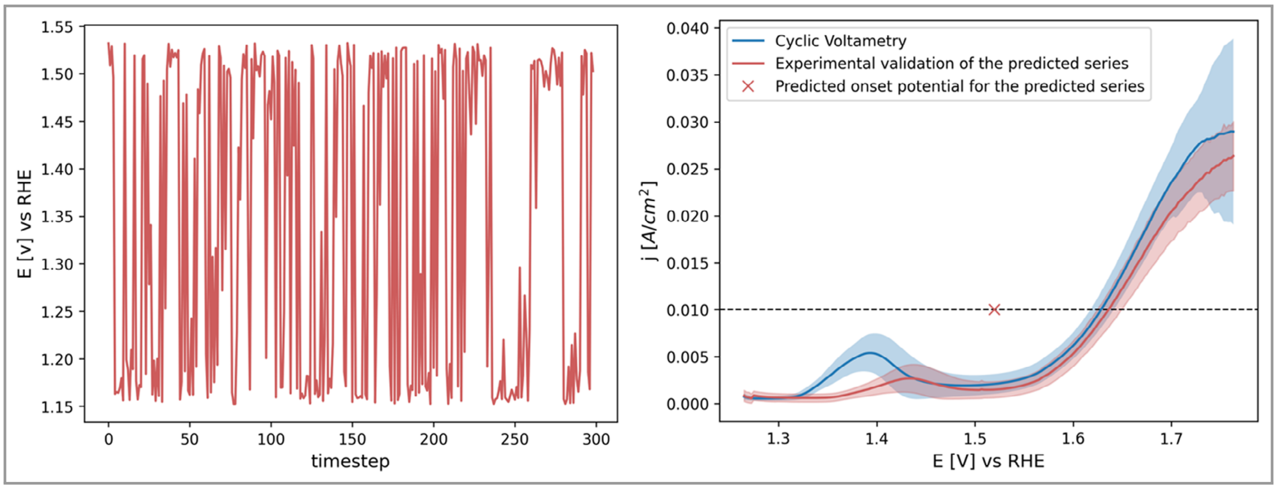


Figure 9. Resulting potential profile proposed by the ANN after supervised learning (left) and experimental validation of the proposed profile (right). The benchmark experiments and the predicted overpotential (x) are shown for comparison.

from an active learning procedure resulting in a somewhat biased dataset as the algorithms will only be chasing the states with the most reward. To fix this, a more random distribution over the OER potential range would probably help. But still the approach seems to be feasible for an automated testing of electrodes.

5 Conclusions

In this study, we could unfortunately not show a better start-up procedure for nickel electrodes as nickel was kind of unimpressed by our endeavors. But the methodology used seems to be promising to be applied on other occasions. Here, we could demonstrate two things: First, that active learning strategies can supplement high-throughput experimentation, especially when data can relatively easily be collected like in electrochemistry. Second, we could show how to extract optimal parameters in case they are hidden in an ANN. This might be a very general theme that can be employed on many occasions. Combining those two methods is not completely advisable as the biased datasets from active learning might not be advantageous for the outcome of the supervised learning process.

The source for this artificial electrochemist is available at the following git repository: https://git.rwth-aachen.de/ai4oer.

Acknowledgment

This work was performed as part of the Cluster of Excellence Fuel Science Center (EXC 2186 ID: 390919832) funded by the Excellence Initiative by the German federal and state governments to promote science and research at German universities and NFDI4Cat as funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) with the project number 441926934.

Open access funding enabled and organized by Projekt DEAL.

Abbreviations

AI	artificial intelligence
ANN	artificial neural network
ML	machine learning
PCA	pricipal component analysis
RL	reinforcement learning
SA	simple agent
SECM	scanning electrochemical microscope
TRA	time-resolved agent

References

[1] N.-T. Suen, S.-F. Hung, Q. Quan, N. Zhang, Y.-J. Xu, H. M. Chen, *Chem. Soc. Rev.* **2017**, *46*, 337–365.
[2] J. Kibsgaard, I. Chorkendorff, *Nat. Energy* **2019**, *4*, 430–433.
[3] Y. J. Son, S. Kim, V. Leung, K. Kawashima, J. Noh, K. Kim, R. A. Marquez, O. A. Carrasco-Jaim, L. A. Smith, H. Celio, D. J. Milliron, B. A. Korgel, C. B. Mullins, *ACS Catal.* **2022**, *12*, 10384–10399.
[4] N. Thissen, J. Hoffmann, S. Tigges, D. A. M. Vogel, J. J. Thoede, S. Khan, N. Schmitt, S. Heumann, B. J. M. Etzold, A. K. Mechler, *ChemElectroChem* **2024**, *11*, e202300432.
[5] U. I. Kramm, R. Marschall, M. Rose, *ChemCatChem* **2019**, *11*, 2563–2574.
[6] S. Senkan, K. Krantz, S. Ozturk, V. Zengin, I. Onal, *Angew. Chem., Int. Ed.* **1999**, *38*, 2794–2799.
[7] J. M. Newsam, F. Schüth, *Biotechnol. Bioeng.* **1999**, *61*, 203–216.
[8] J. R. Kitchin, *Nat. Catal.* **2018**, *1*, 230–232.
[9] K. McCullough, T. Williams, K. Mingle, P. Jamshidi, J. Lauterbach, *Phys. Chem. Chem. Phys.* **2020**, *22*, 11174–11196.
[10] T. Williams, K. McCullough, J. A. Lauterbach, *Chem. Mater.* **2020**, *32*, 157–165.
[11] J. Benavides-Hernández, F. Dumeignil, *ACS Catal.* **2024**, *14*, 11749–11779.
[12] R. Palkovits, S. Palkovits, *ACS Catal.* **2019**, *9*, 8383–8387.
[13] S. Palkovits, *ChemCatChem* **2020**, *12*, 3995–4008.
[14] C. L. M. von Meyenn, S. Palkovits, *Energy Adv.* **2023**, *2*, 691–700.
[15] G. van Rossum, *Python Tutorial*, Centrum voor Wiskunde en Informatica, Amsterdam **1995**.
[16] https://git.rwth-aachen.de/ai4oer
[17] *Reinforcement Learning* (Eds.: A. Nandy, M. Biswas), Apress, Berkeley, CA **2018**.
[18] R. Ris-Ala, *Fundamentals of Reinforcement Learning*, Springer Nature Switzerland, Cham **2023**.
[19] A. Scedrov, *J. Symb. Log.* **1991**, *56*, 336–337.
[20] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, N. Dormann, *J. Mach. Learn. Res.* **2021**, *22*, 1–8.
[21] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, arXiv, New York **2016**.
[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passous, D. Cournapeau, *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
[23] M. E. Tipping, C. M. Bishop, *J. R. Stat. Soc., Ser. B, Methodol.* **2002**, *61*, 611–622.
[24] F. Dalirani. (2021, May 2023), Farhad-dalirani/PytorchRevelio: PytorchRevelio-V2021.2021.2020 (Version V2021.2021.2020), Zenodo.
[25] J. A. Nelder, R. Mead, *Comput. J.* **1965**, *7*, 308–313.
[26] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, I. Polat, Y. Feng, E. W. Moore, J. VanderPlas, *Nat. Methods* **2020**, *17*, 261–272.