Supporting Information
Non-aqueous Battery Electrolytes: High Throughput Experimentation and Machine Learning-Aided Optimization of Ionic Conductivity

Peng Yan[a], Mirko Fischer[b], Harrison Martin[b], Christian Woelke[a], Anand Narayanan Krishnamoorthy[a], Isidora Cekic-Laskovic[a], Diddo Diddens[a], Martin Winter[c], and Andreas Heuer[b]

[a]Helmholtz-Institute Münster (IEK-12), Forschungszentrum Jülich GmbH, Corrensstraße 46, 48149 Münster, Germany. E-mail: andheuer@uni-muenster.de, i.cekic-laskovic@fz-juelich.de
[b]Institute of Physical Chemistry, University of Münster, Corrensstraße 28/30, 48149 Münster, Germany
[c]MEET Battery Research Center, University of Münster, Corrensstraße 46, 48149 Münster, Germany

Contents

S1 Operational procedure of HTS platform **S3**

S2 LECA - The Liquid Electrolyte Composition Analysis Package **S3**
S2.1 Module Overview . . . S4
S2.2 LECA.prep . . . S4
S2.3 LECA.fit . . . S4
S2.3.1 Initialization, Scaling, Grouping, Splitting . . . S4
S2.3.2 Adding Models, Bayesian Hyperparameter Selection and Cross-Validation Scoring S5
S2.3.3 Uncertainty Estimation . . . S5
S2.3.4 Objective Optimization Search . . . S6
S2.4 Data preparation workflow with LECA . . . S7
S2.5 LECA model fitting and evaluation workflow . . . S8

S3 Fitting $\log(\sigma)$ via Arrhenius and fitting $\log \frac{\sigma}{x_{\mathrm{LiSalt}}}$ directly **S9**

S4 Arrhenius-Fit **S10**
S4.1 Choice of $\boldsymbol{\beta_0}$ . . . S12
S4.2 Table of all Arrhenius fits . . . S14

S5 Feature correlation **S18**

S6 Feature space coverage **S19**

S7 Model description **S20**
S7.1 Random Forest (RF) . . . S20
S7.2 Optimized Linear Regression (LR opt) . . . S20
S7.3 Gaussian Process Regression (GPR) . . . S22
S7.3.1 Radial basis function kernel (A.RBF and I.RBF) . . . S22
S7.3.2 Matérn kernel (A.M and I.M) . . . S22
S7.3.3 Rational Quadratic kernel (I.RQ) . . . S22
S7.4 Neural network (NN) . . . S23
S7.5 Optimized neural network (NN opt) . . . S23

S8 Train-Test split **S24**

S9 Systematic Study of Neural networks **S25**

S10 Model timings **S27**

S11 Model validation **S29**

S12 Datasize Performance **S34**

S13 Model predictions of individual S_0, S_1 and S_2 **S35**
S13.11D slices . . . S35
S13.2Uncertainties of S_i . . . S37
S13.3Difficult to predict electrolyte compositions . . . S38

S14 Predicted 1D slices for −20°C and 60°C **S40**

S15 Predictions for different temperatures **S42**

S16 Predicted 2D slices for −20°C and 60°C **S44**

S17 2D slices of Arrhenius coefficient predictions **S45**

S18 Further Data available **S47**

S1 Operational procedure of HTS platform

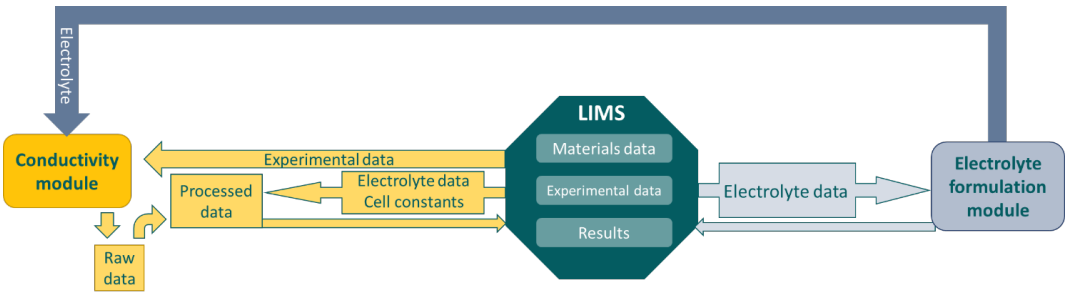


Figure S1: Schematic overview of the LIMS – conductivity and electrolyte modules interaction workflow.

S2 LECA - The Liquid Electrolyte Composition Analysis Package

The core of the LECA package is a python library built to implement and extend the data-driven approach used by[1] to model and analyze the conductivity of liquid electrolyte formulations. As depicted in Figure S2 it provides a simplified and generalized workflow to build robust machine learning regression models and facilitate a better understanding of composition performance.

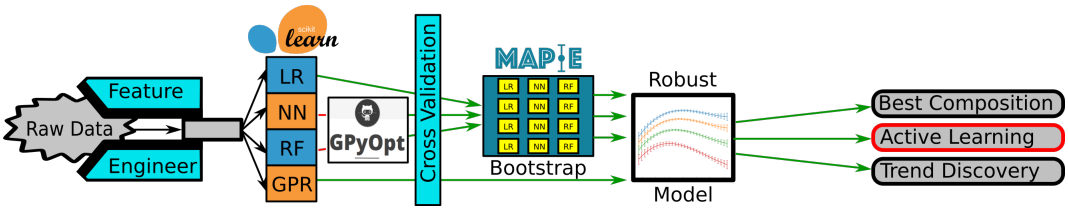


Figure S2: LECA package schematic.[2–4] This Figure is still a placeholder for figure in correct format.

The package uses Jupyter-Notebooks[5] as usage examples and customizable workflows. As a rule, the library uses popular python libraries over proprietary code when possible. A full list of libraries used is included in Table S1, but the main work underlying the LECA package is the Scikit-learn library.[4]

Table S1: Used Libraries

Library	Version	Ref.
Jupyter Notebook	6.4.11	5
Scikit-learn	1.3.1	4
NumPy	1.22.3	6
Matplotlib	3.5.1	7
Pandas	1.4.2	8,9
SciPy	1.8.1	10
Uncertainties	3.1.7	11
MAPIE	0.5.6	3
HDBSCAN	0.8.28	12
Seaborn	0.11.2	13
Sphinx	5.0.2	14
GPyOpt	1.2.6	2

S2.1 Module Overview

The LECA library is comprised of three main modules: prep, fit and analyze. In order to keep the Notebooks as concise as possible, an additional module notebook_utils is included for some frontend-specific utility functions.

S2.2 LECA.prep

Key features: data import and dataset overview, estimate feature importance, generate statistics from repeated measurements, optional Arrhenius surrogate model.

The prep module is designed to extract raw measurement data and convert it via a generic feature-engineering process to train machine learning models. The module imports the in-house high-throughput experimental measurement JSON format or standard labeled CSV tables into Pandas DataFrames for processing.[8] In order to help identify extraneous or irrelevant features, feature correlation and covariance matrices are automatically provided as well as an estimation of the relative feature-importance for each objective function (generated by fitting a default Scikit-learn random forest).

The prep module also includes a function to combine repeated measurements (identical input features) to the mean value and standard deviations. This conversion is automatically detected by the fit module and can be used to weight the bias of specific measurements or filter out measurement values with unusually high deviations.

Finally, as the LECA package was conceived to model liquid electrolyte conductivity, the prep module includes an optional function to convert the dataset with an Arrhenius surrogate model (Equation (S1)). To apply the Arrhenius surrogate model LECA groups measurement data by all features excluding the inverse temperature and fits the measured $\log\sigma$ values as a function of the inverse temperature β, the corresponding $S_0(X)$, $S_1(X)$ and $S_2(X)$ coefficients then become the new objective functions.

$$\log\sigma = S_0(X) - S_1(X)(\beta - \beta_0) - S_2(X)(\beta - \beta_0)^2 \tag{S1}$$

Where σ is the conductivity, X is the vector of non-temperature features, the onset temperature is β_0 and β is the inverse temperature (both expressed in $\left[\frac{1000}{\mathrm{K}}\right]$ units). Note, the onset temperature β_0 is a constant, defined as by default as the inverse temperature where the $S_0(X)$ and $S_1(X)$ coefficients are minimally correlated.[1]

S2.3 LECA.fit

Key features: data scaling / splitting / grouping, Linear / Random Forest / Gaussian Process / Neural Network regression models, GPyOpt hyperparameter optimization[2], cross-validation scoring, uncertainty estimation, objective optimization search.

S2.3.1 Initialization, Scaling, Grouping, Splitting

The fit module contains a single object, the WorkFlow. This object takes the prepared dataset and handles all of the rescaling, data splitting (test / train / validation splits) and scoring of models. The user can choose regression models to add to the WorkFlow, or use the autoML function to automatically select, train and score a subset of the supported regression models based on the size of the dataset. Loading the object is as simple as shown in Figure S3.

One of the main benefits of using the LECA package is that it removes the tedious task of keeping careful track of each test, train and validation split. It is designed to avoid any data-leakage, the *composition_features* argument ensures that the test/train splits are grouped in a way to avoid the easy mistake of e.g. mixing measurement results from the same electrolyte composition at different temperatures in the training and testing datasets. Doing so would result in exceptionally low prediction errors while simply modeling the temperature dependency of the objective function instead of testing the model's ability to predict novel compositions' performance.

The validation holdout data is set aside and completely excluded from any model training or data scaling to avoid data leakage. This holdout data serves as a final check to ensure that the models are indeed effectively predictive on unseen data.

Note that LECA internally scales all features, using standard scaling[?] as implemented in Scikit-Learn, whereas it stores unscaled objective functions S_0, S_1 and S_2.

```
from LECA import fit
wf = fit.WorkFlow(
    dataset_df,                          # Pandas DataFrame
    ['salt', 'ec:emc', 'temperature'], # Features
    ['conductivity', 'viscosity'],     # Objective functions
    random_state=42,                     # State passed to randomized functions for
                                             repeatability
    polynomial_degree=3,                 # Generate polynomial features to deg. 3
    validation_holdout=0.1,              # Fraction of DataFrame withheld for
                                             validation
    composition_features=['salt', 'ec:emc'], # Data grouped by these features THEN
                                              split (cross-validation /
                                             validation)
    n_jobs=-1                            # Use all possible CPU cores for parallel
                                             computations
)
```

Figure S3: Initializing a LECA workflow.

S2.3.2 Adding Models, Bayesian Hyperparameter Selection and Cross-Validation Scoring

The LECA package includes Linear[15], Random Forest[16], Gaussian Process[17–19] and Multilayer Perceptron (neural network)[15,20] regression models. LECA also supports automated bayesian hyperparameter optimization for the Random Forest and Neural Network models.[2] Furthermore, it includes an optimization routine for Linear Regression models. Once the user has selected the models to include in the WorkFlow, the comparative quality of the models is scored using cross-validation, ranking the best model by the average mean-squared-error score over all cross-validation folds.

S2.3.3 Uncertainty Estimation

Predictive models are of course useful, but accurate prediction uncertainty estimation is the tool which unlocks the possibility of active learning and Bayesian optimization.[21] Gaussian Process Regression is a particularly attractive model type as it inherently includes prediction uncertainty. For Linear, Random Forest and Multilayer Perceptron regression models the LECA package uses the MAPIE (Model Agnostic Prediction Interval Estimator) library.[3] The LECA package implements two methods based on MAPIE bootstrapping and allows the user to freely choose which to use for uncertainty estimation.

MAPIE is an implementation of the jackknife resampling strategy introduced by Foygel-Barber et al. (2020)[3,22]. LECA uses the jackknife+-after-bootstrapping method from MAPIE to estimate the confidence interval of a prediction with theoretical guarantees for coverage.[23] In practice this method involves generating bootstrapped datasets and training copies of the regression model on the bootstrapped datasets. The jackknife step then involves calculating the *conformity score*, or interval between the prediction of the bootstrapped models and values excluded from the bootstrapped data.[3] While computationally expensive, this method gives information on the variance of the regression model from the bootstrapping process, as well as the predictive quality of the model for unseen data. Finally, the confidence interval is calculated as the equivalent interval from the max prediction from the set of bootstrapped models plus the conformity scores (as the upper bound) and the min prediction minus the conformity scores (as the lower bound).

As a second option, equivalent to the method used in our previous work[1], the same bootstrapped models used by MAPIE can be used to generate a prediction distribution which reflects the impact of varying model training input on the output prediction. The standard deviations of the set of bootstrapped models' predictions can then be taken as an estimation of the *epistemic* uncertainty of the ensemble of models.

Further, having already expended the computational cost of retraining manifold bootstrapped models to estimate uncertainty for non-GPR models, by default LECA uses the mean predictions of the ensemble of bootstrapped models for predictions to help mitigate model bias caused by noisy or outlier data.

S2.3.4 Objective Optimization Search

LECA.WorkFlow also includes an objective optimization search method. This uses the SciPy optimization[10] function to search the design space with the trained models for the composition which maximizes/minimizes the objective function. Alternately it can be used to search for the point with the highest Bayesian Expected Improvement, maximal model uncertainty or upper/lower confidence bounds (objective prediction ± uncertainty).

LECA.analyze

Key features: comparative model performance overview, training data size:prediction accuracy analysis

The analysis module includes two functions to gain further insight into the relative performance of different model types. The performance_plot function is a simple visualization tool for displaying the average cross-validation scores of all of the trained models (Training time, Mean Absolute Error, Mean Squared Error and R2 score for the training and test folds).

The datasize_performance function trains copies of a model with randomized subsamples of the dataset with increasing training dataset sizes and scores the model's predictive accuracy. This tool helps show the trend of increasing data on predictive accuracy, which can inform the potential value of generating further data.

Furthermore, the analysis module includes functions to analyze the trained models further, offering the possibility to plot experimental and predicted Arrhenius fits, as well as 1-dimensional and 2-dimensional slices of predictions through the hyper-dimensional feature space.

S2.4 Data preparation workflow with LECA

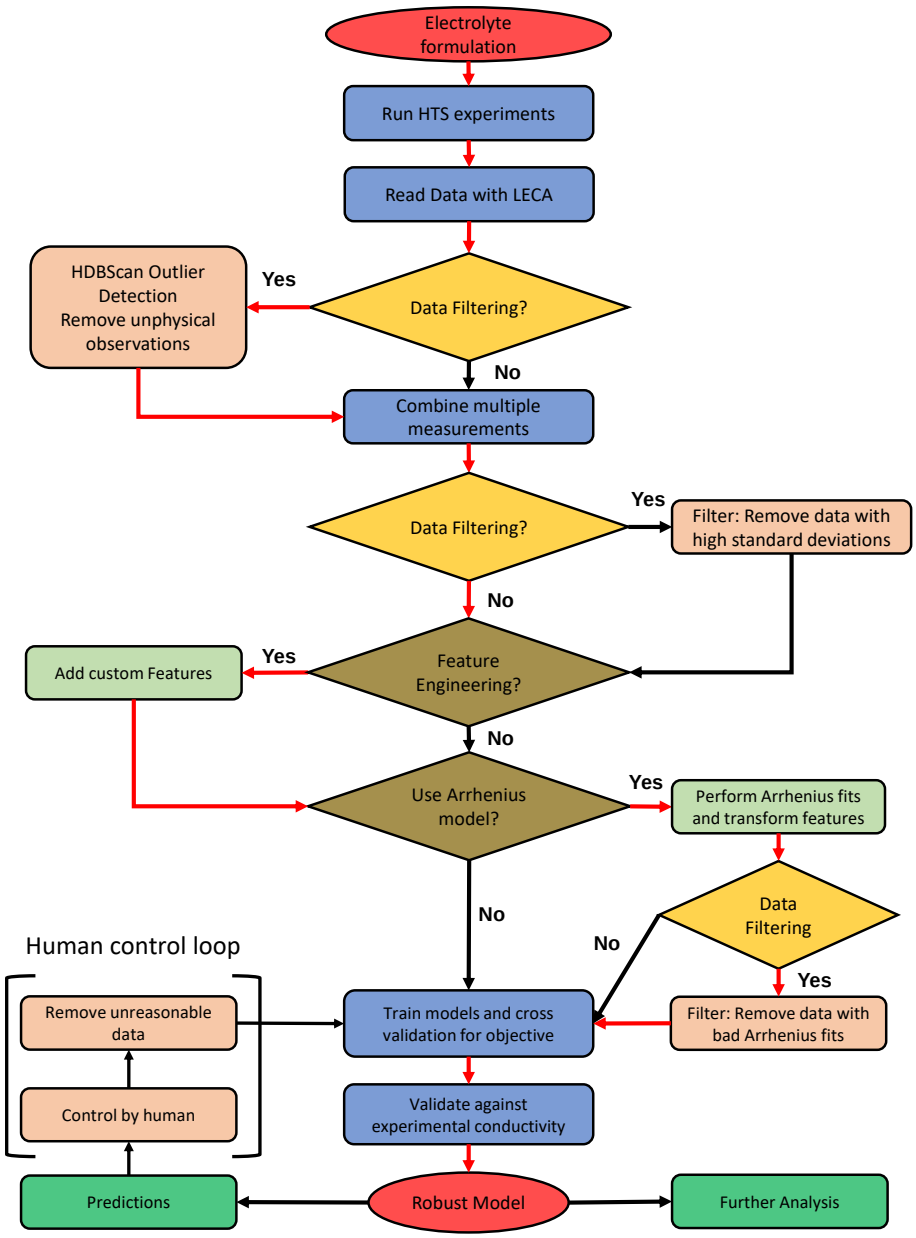


Figure S4: Flowchart of data preparation for LECA. The red arrows indicate the workflow applied to fit the models in this work. Black arrows indicate optional paths and choices one may take to fit a model.

S2.5 LECA model fitting and evaluation workflow

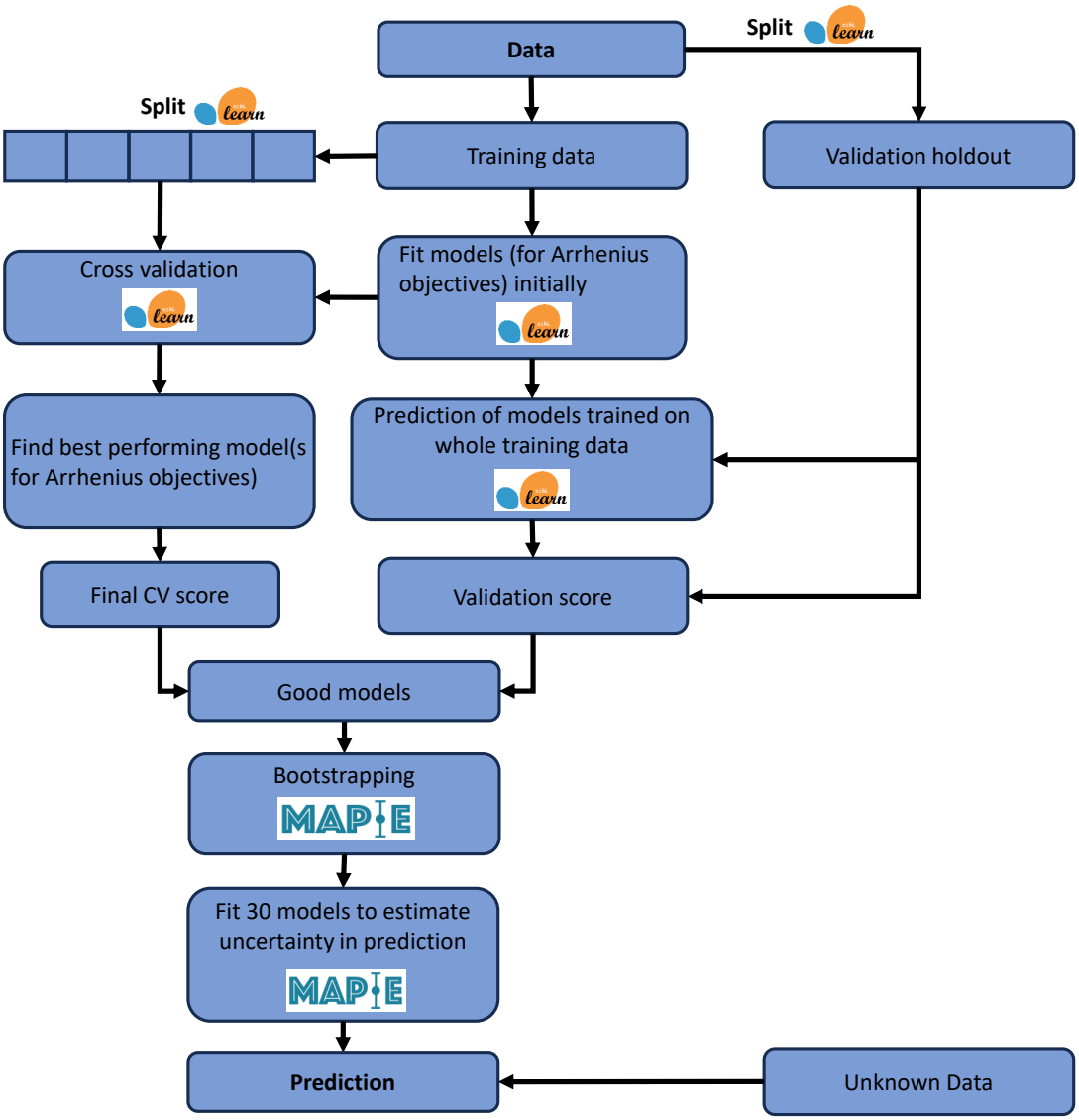
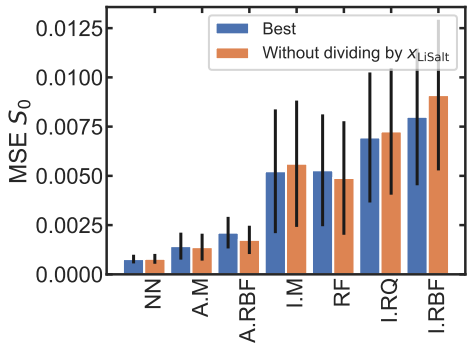


Figure S5: Flowchart of the LECA WorkFlow class. It is indicated where the libraries skikit-learn [4] and Mapie [3] are used.

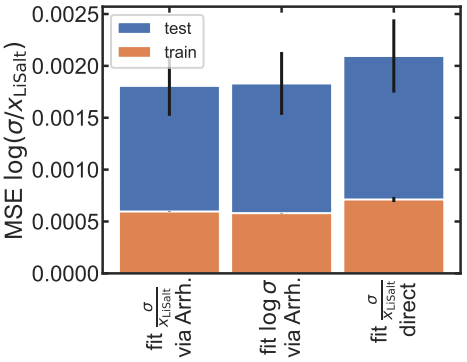
S3 Fitting $\log(\sigma)$ via Arrhenius and fitting $\log \frac{\sigma}{x_{\mathrm{LiSalt}}}$ directly

Besides fitting models to the Arrhenius objectives S_i calculated from Arrhenius fits of $\log \frac{\sigma}{x_{\mathrm{LiSalt}}}$ it is also possible to fit models to Arrhenius objectives S_i calculated from an Arrhenius fit of $\log \sigma$. In Figure S6a we show that for the NN model the MSE is well within standard errors when predicting S_0 for both $\log \frac{\sigma}{x_{\mathrm{LiSalt}}}$ and $\log \sigma$. For S_1 and S_2 the MSEs are identical, because when going from $\log \sigma$ to $\log \frac{\sigma}{x_{\mathrm{LiSalt}}}$ only S_0 is modified. The main difference is that we predict with the former approach correct behavior for $x_{\mathrm{LiSalt}} \to 0$, which means that $\sigma \to 0$. With the latter approach, we observe a finite σ for $x_{\mathrm{LiSalt}} \to 0$ (Figure S6c). The correct behavior for fitting $\log \frac{\sigma}{x_{\mathrm{LiSalt}}}$ arises from the back-transformation to sigma, where we have to multiply the right-hand side of Equation S2 with x_{LiSalt}.

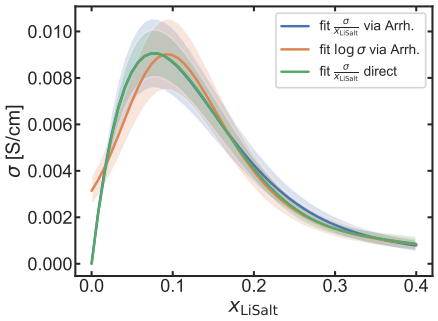
Finally, we clarify that it is advantageous to apply the Arrhenius fit and fit models for the objective functions S_i, instead of fitting $\log \frac{\sigma}{x_{\mathrm{LiSalt}}}$ directly without the Arrhenius fit, which can be seen in Figure S6b. The reason why the direct fit has a higher MSE is that we do not include physical information about the temperature dependence in the model and the inverse temperature as an additional feature leads to a higher model complexity, which makes the data more difficult to fit for the models.



(a) Comparison of averaged MSE in leave-one-out CV in predicting S_0.



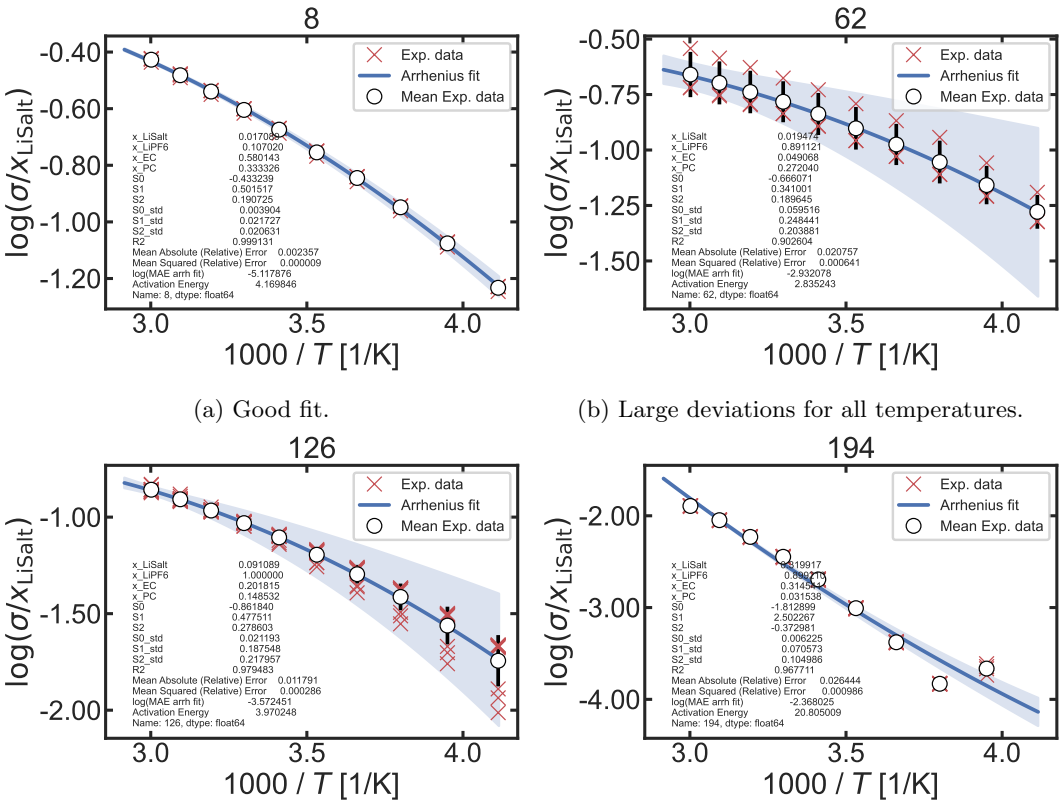
(b) Comparison of MSE in predicting $\log \frac{\sigma}{x_{\mathrm{LiSalt}}}$ in leave-one-out CV via Arrhenius fit and 10-fold CV via the direct fit.



(c) Comparing predictions of σ for varying x_{LiSalt}.

Figure S6: Comparison of fitting $\log \frac{\sigma}{x_{\mathrm{LiSalt}}}$ via Arrhenius with fitting $\log(\sigma)$ via Arrhenius and fitting $\log \frac{\sigma}{x_{\mathrm{LiSalt}}}$ directly. As a model we use NN. The black bars correspond to standard errors.

S4 Arrhenius-Fit



(a) Good fit.

(b) Large deviations for all temperatures.

(c) Large deviations for low temperatures.

(d) Nonphysical fit.

Figure S7: Example Arrhenius fits. The black open circles represent the mean of the individual measurements (red crosses). The fit is displayed in blue together with its standard deviation.

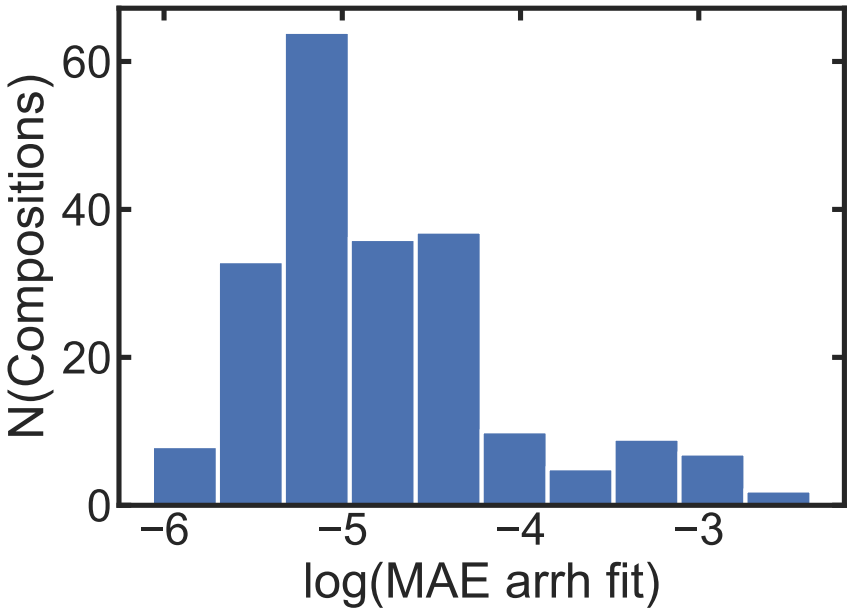
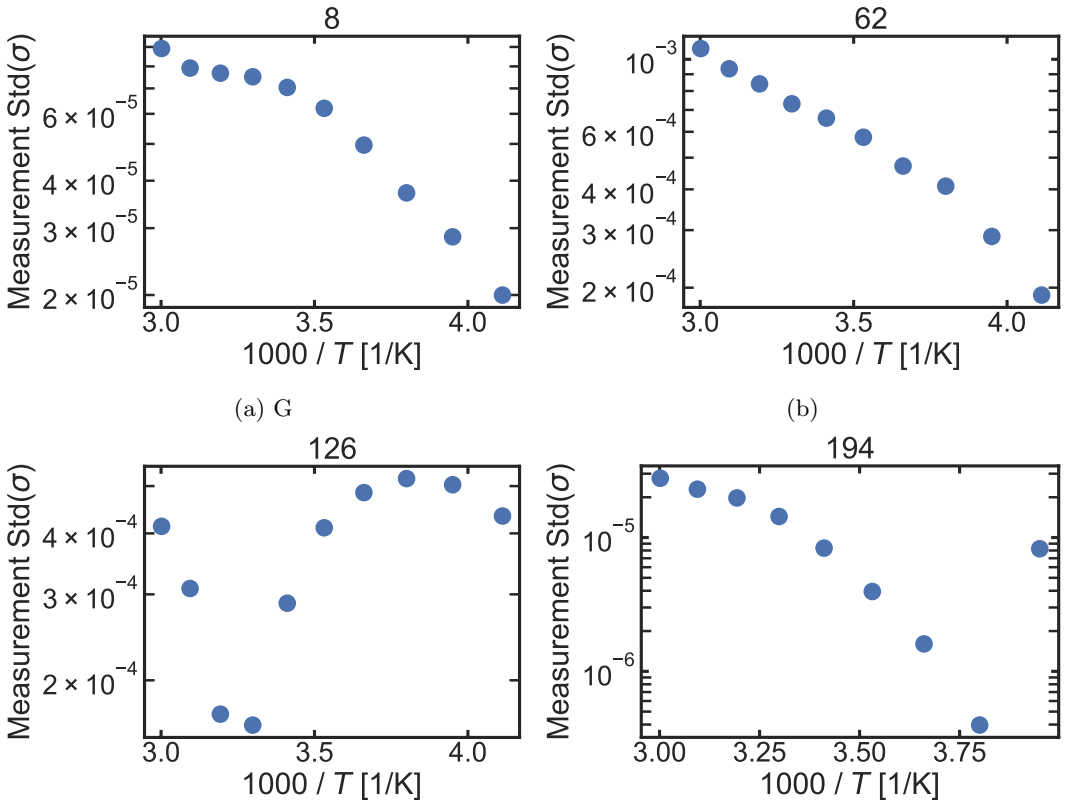


Figure S8: Histogram of log(MAE) of the Arrhenius fit.



(a) G (b)

(c) (d)

Figure S9: Standard deviation of the individual measurements for each inverse temperature for electrolyte formulations corresponding to Figure S7.

S4.1 Choice of β_0

In our previous work, we chose β_0 such that S_0 and S_1 are uncorrelated. Applying this approach we yield an onset temperature $1/\beta_0 = T_0$ of 60°C. This should lead to optimized predictions of all S_i.

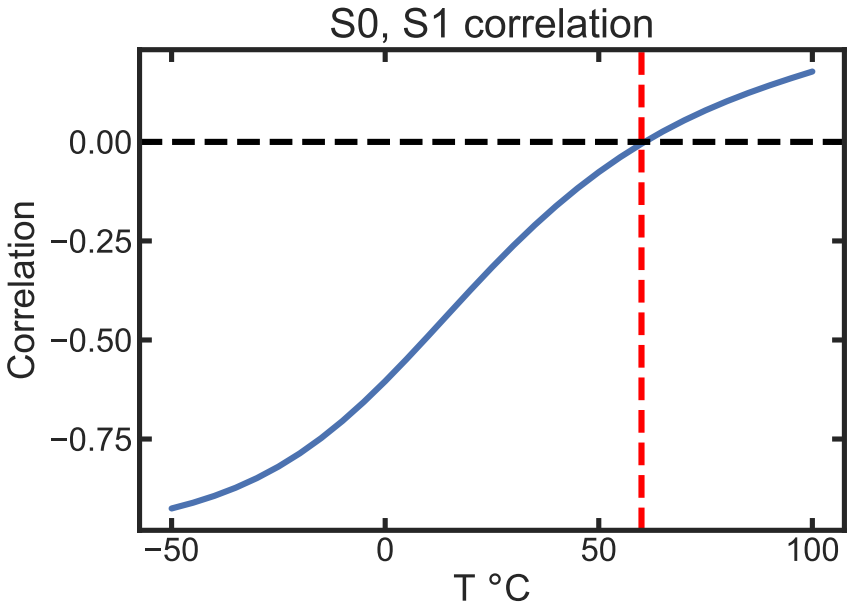


Figure S10: Correlation of S_0 and S_1 for different onset temperatures.

However, when transforming back from the S_i to the original ionic conductivity via

$$\log(\frac{\sigma}{x_{\text{LiSalt}}}) = S_0 - S_1 \cdot (\beta - \beta_0) - S_2 \cdot (\beta - \beta_0)^2 \tag{S2}$$

there is a subtle effect on the error bars when calculating them using individual errors ΔS_i

$$\Delta \log(\frac{\sigma}{x_{\text{LiSalt}}}) = \Delta S_0 - \Delta S_1 \cdot (\beta - \beta_0) - \Delta S_2 \cdot (\beta - \beta_0)^2. \tag{S3}$$

Directly for the onset temperature, only ΔS_0 is relevant (because $\beta - \beta_0 = 0$), whereas for smaller or larger temperatures also ΔS_1 and ΔS_2 are relevant and therefore the total error depends on the temperature. This also means that the error for a certain temperature is clearly dependent on the choice of β_0. Following this argument, it might be suitable to choose the onset temperature equal to the temperature for which one wants to make predictions to reduce the model error. However, comparing the overall performance of NN models trained for different onset temperatures (−20°C, 20°C and 60°C) in terms of leave-one-out cross-validation scores we find that the NN model scores best for an onset temperature of 20°C and 60°C, as shown in Figure S11.

Thus we decide to make predictions also for all other temperatures on models trained on Arrhenius fits with an onset temperature of 60°C. In practice we do not expect large deviations in predictions of models trained with different onset temperatures under the condition of similar MSEs. Another argument is that the back-transformation for this model captures the experimental deviations at low temperatures very well as shown in Figure S19c. Furthermore, the minimal correlation of S_0 and S_1 seems to be most accurate to investigate the effects of features on these single objectives. Concluding, if one wants to perform the Arrhenius fit once, it seems best to choose the onset temperature where S_0 and S_1 are mostly uncorrelated.

Since the NN architecture is partly optimized to Arrhenius fits for 60°C the results might change when testing different architectures for Arrhenius fit performed with other onset temperatures. However, due to the overall good performance of various different NN architectures, we expect that results

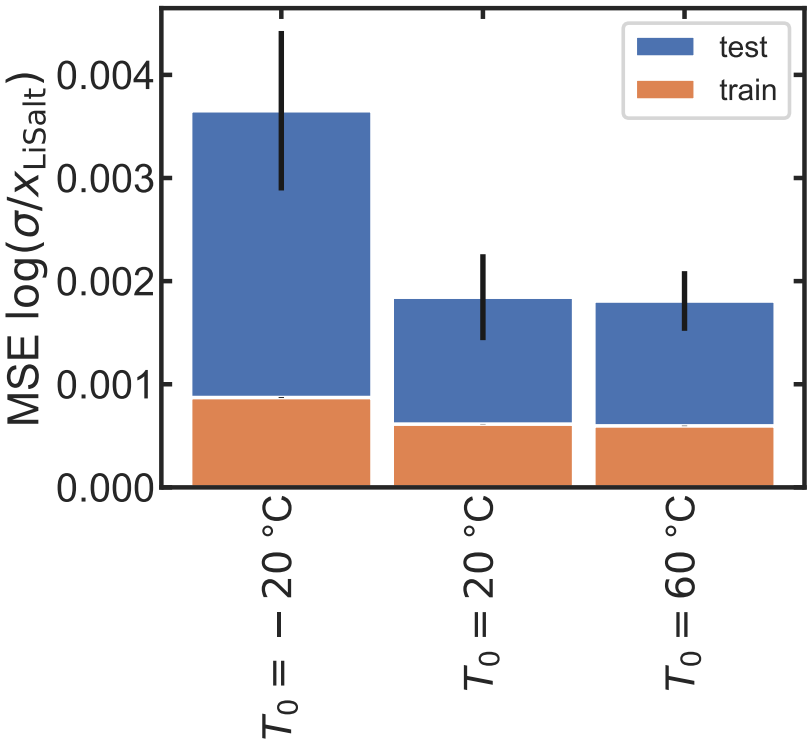


Figure S11: Comparison of MSE of NN model trained on Arrhenius fits with different onset temperatures. Leave-one-out CV scores are shown.

will be similar for other onset temperatures. This can be part of future work, to reveal the influence of the choice of β_0 in more detail.

S4.2 Table of all Arrhenius fits

Table S2: Data of Arrhenius fits for all electrolyte formulations. Electrolyte formulation 194 (marked by *) was not used for further model training and testing due to an insufficient fit (log(MAE) > -2.5). Reported are for each electrolyte formulation the Arrhenius coefficients S_0, S_1 (in units K) and S_2 (in units K^2) as well as their standard deviations. Furthermore, the R2-score and the logarithm of the mean absolute error of the Arrhenius fit are reported as well as the activation energy E_a (in units mJ/mol) at the onset temperature $T_0 = 1/\beta_0 = 60\,°C$.

	x_{LiSalt}	$x_{\mathrm{LiPF_6}}$	x_{EC}	x_{PC}	S_0	S_1	S_2	S_0 std	S_1 std	S_2 std	$R2$	log(MAE)	E_a
0	0.016	0.000	0.712	0.064	-0.379	0.483	0.405	0.007	0.082	0.080	0.991	-3.750	4.016
1	0.016	0.891	0.713	0.064	-0.346	0.519	0.517	0.014	0.138	0.162	0.990	-3.527	4.312
2	0.016	0.504	0.712	0.064	-0.369	0.511	0.428	0.011	0.084	0.091	0.992	-3.698	4.253
3	0.016	0.110	0.713	0.064	-0.376	0.391	0.583	0.008	0.063	0.072	0.986	-3.478	3.248
4	0.017	0.000	0.580	0.333	-0.411	0.485	0.201	0.005	0.021	0.017	0.999	-5.209	4.033
5	0.017	0.894	0.542	0.000	-0.390	0.203	0.722	0.035	0.190	0.197	0.980	-3.214	1.685
6	0.017	0.000	0.516	0.047	-0.357	0.425	0.223	0.038	0.196	0.181	0.951	-3.158	3.534
7	0.017	0.000	0.542	0.000	-0.424	0.109	0.752	0.020	0.164	0.198	0.984	-3.425	0.902
8	0.017	0.107	0.580	0.333	-0.433	0.502	0.191	0.004	0.022	0.021	0.999	-5.118	4.170
9	0.017	0.108	0.542	0.000	-0.419	0.282	0.532	0.035	0.301	0.353	0.940	-2.927	2.342
10	0.017	0.000	0.000	0.663	-0.449	0.484	0.181	0.006	0.025	0.021	0.999	-5.129	4.027
11	0.017	0.893	0.515	0.047	-0.420	0.315	0.304	0.007	0.057	0.067	0.996	-4.470	2.617
12	0.017	0.890	0.580	0.334	-0.414	0.497	0.209	0.003	0.012	0.012	0.999	-5.256	4.136
13	0.017	1.000	0.542	0.000	-0.249	0.183	0.730	0.014	0.075	0.059	0.994	-3.924	1.517
14	0.017	0.491	0.541	0.000	-0.405	0.273	0.482	0.027	0.235	0.294	0.966	-3.269	2.270
15	0.017	0.105	0.516	0.047	-0.341	0.408	0.210	0.056	0.215	0.181	0.953	-3.177	3.389
16	0.017	0.495	0.515	0.047	-0.415	0.282	0.447	0.028	0.264	0.321	0.962	-3.273	2.344
17	0.017	1.000	0.579	0.334	-0.429	0.465	0.239	0.007	0.028	0.022	1.000	-5.397	3.865
18	0.017	1.000	0.516	0.047	-0.408	0.327	0.381	0.022	0.204	0.258	0.971	-3.496	2.716
19	0.017	0.491	0.580	0.333	-0.415	0.515	0.188	0.003	0.012	0.011	0.999	-5.183	4.284
20	0.018	0.104	0.372	0.242	-0.439	0.455	0.156	0.004	0.016	0.014	0.999	-5.467	3.782
21	0.018	0.895	0.371	0.243	-0.459	0.451	0.183	0.003	0.015	0.014	1.000	-5.558	3.750
22	0.018	0.000	0.216	0.627	-0.449	0.479	0.201	0.004	0.016	0.014	1.000	-5.378	3.979
23	0.018	0.000	0.372	0.242	-0.443	0.445	0.160	0.005	0.019	0.015	0.999	-5.470	3.702
24	0.018	1.000	0.372	0.243	-0.461	0.450	0.185	0.005	0.023	0.019	0.999	-5.211	3.745
25	0.018	0.889	0.336	0.000	-0.531	0.412	0.157	0.002	0.012	0.012	0.999	-5.574	3.422
26	0.018	0.506	0.313	0.030	-0.531	0.432	0.135	0.004	0.013	0.011	0.999	-5.376	3.596
27	0.018	1.000	0.216	0.627	-0.469	0.460	0.237	0.002	0.012	0.011	1.000	-5.765	3.823
28	0.018	0.107	0.336	0.000	-0.525	0.415	0.136	0.003	0.010	0.008	1.000	-5.663	3.454
29	0.018	0.890	0.216	0.627	-0.459	0.504	0.203	0.006	0.026	0.023	0.999	-5.034	4.194
30	0.018	0.502	0.336	0.000	-0.530	0.424	0.141	0.004	0.015	0.012	0.999	-5.555	3.528
31	0.018	1.000	0.336	0.000	-0.523	0.441	0.146	0.003	0.016	0.014	0.999	-5.401	3.665
32	0.018	0.520	0.372	0.243	-0.469	0.450	0.177	0.004	0.025	0.022	0.999	-5.072	3.740
33	0.018	0.505	0.216	0.627	-0.457	0.501	0.195	0.003	0.016	0.015	0.999	-5.187	4.162
34	0.019	1.000	0.313	0.030	-0.536	0.424	0.152	0.003	0.017	0.015	0.999	-5.308	3.527
35	0.019	0.903	0.000	0.505	-0.486	0.447	0.174	0.003	0.017	0.015	1.000	-5.564	3.719
36	0.019	0.000	0.313	0.029	-0.526	0.451	0.109	0.006	0.021	0.016	0.999	-5.189	3.753
37	0.019	0.112	0.218	0.626	-0.470	0.501	0.183	0.004	0.017	0.015	0.999	-5.129	4.167
38	0.019	1.000	0.105	0.451	-0.494	0.451	0.173	0.004	0.016	0.014	0.999	-5.260	3.750
39	0.019	1.000	0.000	0.704	-0.484	0.453	0.241	0.004	0.015	0.012	1.000	-5.614	3.764
40	0.019	0.103	0.000	0.505	-0.476	0.448	0.154	0.003	0.017	0.017	0.999	-5.258	3.725
41	0.019	0.895	0.000	0.704	-0.478	0.453	0.248	0.002	0.013	0.015	0.999	-5.201	3.764
42	0.019	0.000	0.106	0.450	-0.474	0.463	0.147	0.004	0.017	0.016	0.999	-5.279	3.851
43	0.019	0.124	0.313	0.031	-0.528	0.424	0.135	0.005	0.026	0.022	0.999	-5.164	3.526
44	0.019	0.502	0.107	0.451	-0.474	0.448	0.172	0.002	0.008	0.006	1.000	-6.041	3.721
45	0.019	0.898	0.105	0.451	-0.471	0.484	0.150	0.004	0.014	0.011	0.999	-5.363	4.027
46	0.019	0.000	0.000	0.505	-0.475	0.443	0.159	0.005	0.024	0.021	0.999	-5.102	3.682
47	0.019	0.873	0.313	0.029	-0.535	0.439	0.141	0.003	0.014	0.014	0.999	-5.341	3.647

Continued on next page

	x_{LiSalt}	$x_{\mathrm{LiPF_6}}$	x_{EC}	x_{PC}	S_0	S_1	S_2	S_0 std	S_1 std	S_2 std	$R2$	log(MAE)	E_a
48	0.019	0.106	0.000	0.704	-0.469	0.486	0.198	0.004	0.016	0.013	1.000	-5.483	4.039
49	0.019	0.511	0.000	0.705	-0.481	0.431	0.243	0.006	0.020	0.015	1.000	-5.691	3.586
50	0.019	0.504	0.000	0.505	-0.487	0.448	0.165	0.001	0.006	0.006	1.000	-5.622	3.721
51	0.019	0.497	0.203	0.148	-0.552	0.424	0.133	0.005	0.025	0.024	0.998	-4.981	3.524
52	0.019	0.106	0.106	0.451	-0.470	0.435	0.173	0.005	0.023	0.021	0.999	-5.158	3.618
53	0.019	1.000	0.202	0.148	-0.554	0.407	0.159	0.003	0.012	0.010	1.000	-5.694	3.383
54	0.019	0.000	0.203	0.149	-0.547	0.409	0.131	0.003	0.016	0.015	0.999	-5.240	3.402
55	0.019	0.113	0.202	0.148	-0.545	0.412	0.132	0.003	0.010	0.009	0.999	-5.469	3.429
56	0.019	1.000	0.048	0.272	-0.604	0.354	0.188	0.004	0.014	0.010	1.000	-5.798	2.941
57	0.019	0.505	0.000	0.304	-0.584	0.398	0.145	0.005	0.020	0.016	0.999	-5.398	3.308
58	0.019	0.888	0.202	0.148	-0.552	0.392	0.169	0.001	0.007	0.006	1.000	-6.068	3.262
59	0.019	0.000	0.048	0.273	-0.560	0.387	0.144	0.004	0.017	0.014	0.999	-5.408	3.221
60	0.019	1.000	0.000	0.304	-0.602	0.373	0.171	0.006	0.021	0.015	1.000	-5.772	3.102
61	0.019	0.495	0.049	0.272	-0.563	0.370	0.167	0.008	0.046	0.045	0.995	-4.503	3.075
62	0.019	0.891	0.049	0.272	-0.666	0.341	0.190	0.060	0.248	0.204	0.903	-2.932	2.835
63	0.020	0.884	0.000	0.304	-0.594	0.396	0.155	0.003	0.010	0.008	1.000	-5.818	3.290
64	0.020	0.000	0.000	0.304	-0.572	0.408	0.131	0.005	0.021	0.019	0.999	-5.271	3.396
65	0.020	0.102	0.000	0.304	-0.584	0.399	0.138	0.003	0.011	0.009	1.000	-5.708	3.316
66	0.020	0.146	0.050	0.276	-0.555	0.388	0.150	0.006	0.032	0.032	0.998	-4.839	3.224
67	0.021	1.000	0.000	0.000	-2.281	0.022	0.157	0.003	0.018	0.015	0.993	-5.291	0.181
68	0.021	0.000	0.000	0.000	-1.814	0.202	0.133	0.002	0.007	0.006	0.998	-5.445	1.677
69	0.021	1.000	0.000	0.000	-2.149	0.090	0.119	0.004	0.021	0.020	0.990	-5.015	0.750
70	0.074	1.000	0.735	0.000	-0.676	0.420	0.830	0.037	0.337	0.404	0.965	-2.834	3.489
71	0.075	0.899	0.734	0.000	-0.680	0.390	0.918	0.034	0.319	0.383	0.974	-3.008	3.239
72	0.076	1.000	0.713	0.064	-0.682	0.591	0.376	0.002	0.010	0.009	1.000	-5.167	4.915
73	0.076	0.499	0.735	0.000	-0.671	0.438	0.755	0.047	0.450	0.558	0.951	-2.683	3.643
74	0.076	0.899	0.714	0.065	-0.687	0.471	0.603	0.041	0.391	0.493	0.973	-3.229	3.916
75	0.077	0.102	0.734	0.000	-0.672	0.561	0.604	0.041	0.389	0.481	0.957	-2.819	4.668
76	0.077	0.502	0.714	0.065	-0.690	0.586	0.371	0.003	0.013	0.011	1.000	-5.081	4.872
77	0.077	0.000	0.733	0.000	-0.691	0.445	0.604	0.041	0.373	0.453	0.961	-2.932	3.698
78	0.078	0.102	0.712	0.066	-0.690	0.581	0.298	0.001	0.008	0.008	1.000	-5.188	4.835
79	0.078	0.000	0.713	0.065	-0.690	0.570	0.307	0.004	0.018	0.015	0.999	-5.073	4.739
80	0.081	1.000	0.577	0.333	-0.742	0.596	0.352	0.003	0.011	0.009	1.000	-5.050	4.956
81	0.081	1.000	0.542	0.000	-0.733	0.509	0.357	0.003	0.016	0.014	1.000	-5.278	4.231
82	0.082	0.898	0.541	0.000	-0.722	0.529	0.333	0.004	0.020	0.017	0.999	-5.117	4.400
83	0.082	1.000	0.515	0.047	-0.724	0.545	0.328	0.003	0.013	0.011	1.000	-5.095	4.529
84	0.082	0.501	0.580	0.334	-0.740	0.605	0.377	0.003	0.013	0.011	1.000	-5.051	5.029
85	0.082	0.900	0.515	0.047	-0.737	0.522	0.333	0.006	0.023	0.018	0.999	-4.948	4.344
86	0.083	0.498	0.542	0.000	-0.730	0.508	0.316	0.003	0.013	0.012	1.000	-5.294	4.226
87	0.083	0.000	0.580	0.333	-0.738	0.534	0.342	0.003	0.018	0.017	1.000	-5.231	4.438
88	0.083	0.499	0.515	0.048	-0.733	0.510	0.310	0.004	0.017	0.013	0.999	-5.033	4.238
89	0.083	0.106	0.580	0.334	-0.753	0.561	0.335	0.003	0.016	0.016	0.999	-5.059	4.667
90	0.084	0.100	0.542	0.000	-0.715	0.511	0.271	0.004	0.022	0.020	0.999	-5.146	4.246
91	0.084	0.000	0.542	0.000	-0.719	0.456	0.313	0.006	0.030	0.029	0.999	-4.767	3.795
92	0.084	0.108	0.515	0.047	-0.722	0.505	0.288	0.004	0.017	0.015	0.999	-5.138	4.199
93	0.085	0.000	0.515	0.047	-0.735	0.472	0.279	0.006	0.027	0.028	0.999	-5.075	3.928
94	0.085	0.904	0.582	0.333	-0.779	0.615	0.459	0.004	0.015	0.013	0.999	-4.850	5.113
95	0.086	1.000	0.371	0.243	-0.772	0.549	0.322	0.004	0.017	0.015	0.999	-4.960	4.566
96	0.086	0.899	0.371	0.243	-0.762	0.545	0.318	0.004	0.017	0.015	0.999	-5.048	4.535
97	0.086	0.899	0.214	0.627	-0.791	0.598	0.388	0.003	0.015	0.013	1.000	-4.993	4.971
98	0.087	1.000	0.219	0.626	-0.800	0.618	0.439	0.005	0.020	0.016	1.000	-4.898	5.141
99	0.087	0.499	0.226	0.623	-0.794	0.583	0.342	0.016	0.060	0.048	0.998	-4.423	4.848
100	0.087	0.499	0.371	0.243	-0.772	0.523	0.294	0.006	0.028	0.023	0.999	-4.946	4.346
101	0.087	0.900	0.000	0.704	-0.809	0.585	0.379	0.006	0.027	0.023	0.999	-4.866	4.865
102	0.087	1.000	0.000	0.704	-0.800	0.598	0.384	0.003	0.017	0.020	0.999	-4.905	4.973
103	0.088	1.000	0.336	0.000	-0.812	0.478	0.297	0.002	0.008	0.008	1.000	-5.334	3.971
104	0.088	0.000	0.228	0.624	-0.787	0.562	0.311	0.002	0.008	0.006	1.000	-5.114	4.674
105	0.088	0.099	0.371	0.242	-0.772	0.499	0.283	0.004	0.026	0.028	0.999	-5.017	4.150

Continued on next page

	x_{LiSalt}	$x_{\mathrm{LiPF_6}}$	x_{EC}	x_{PC}	S_0	S_1	S_2	S_0 std	S_1 std	S_2 std	$R2$	log(MAE)	E_a
106	0.088	0.105	0.223	0.625	-0.786	0.564	0.319	0.008	0.032	0.027	0.999	-4.903	4.693
107	0.088	0.000	0.371	0.243	-0.775	0.479	0.297	0.006	0.033	0.036	0.999	-4.939	3.980
108	0.088	0.898	0.336	0.000	-0.826	0.442	0.311	0.005	0.024	0.024	0.999	-5.175	3.673
109	0.089	1.000	0.313	0.029	-0.834	0.407	0.345	0.004	0.022	0.022	1.000	-5.381	3.385
110	0.089	0.899	0.313	0.029	-0.829	0.436	0.311	0.008	0.035	0.029	0.999	-4.869	3.624
111	0.089	0.500	0.000	0.704	-0.799	0.556	0.364	0.004	0.018	0.016	1.000	-5.130	4.625
112	0.089	0.101	0.000	0.704	-0.798	0.547	0.338	0.001	0.010	0.010	1.000	-5.393	4.548
113	0.089	1.000	0.106	0.451	-0.797	0.511	0.349	0.003	0.016	0.015	1.000	-5.181	4.252
114	0.090	0.497	0.337	0.000	-0.804	0.470	0.265	0.008	0.063	0.073	0.997	-4.550	3.904
115	0.090	0.899	0.106	0.451	-0.792	0.522	0.320	0.004	0.017	0.014	1.000	-5.196	4.336
116	0.090	0.000	0.000	0.704	-0.806	0.533	0.337	0.003	0.016	0.014	1.000	-5.289	4.434
117	0.090	0.498	0.313	0.029	-0.815	0.414	0.321	0.005	0.032	0.038	0.998	-4.916	3.445
118	0.090	1.000	0.000	0.505	-0.822	0.553	0.303	0.007	0.041	0.039	0.998	-4.545	4.594
119	0.090	0.900	0.000	0.505	-0.809	0.511	0.310	0.003	0.014	0.012	1.000	-5.332	4.250
120	0.090	0.102	0.336	0.000	-0.808	0.432	0.265	0.004	0.019	0.016	0.999	-5.193	3.592
121	0.091	0.000	0.336	0.000	-0.807	0.423	0.265	0.010	0.039	0.032	0.999	-4.889	3.520
122	0.091	0.100	0.312	0.030	-0.804	0.437	0.254	0.008	0.030	0.025	0.999	-5.170	3.636
123	0.091	0.000	0.314	0.029	-0.804	0.436	0.249	0.007	0.025	0.020	0.999	-5.374	3.621
124	0.091	0.498	0.106	0.451	-0.791	0.521	0.296	0.002	0.014	0.014	0.999	-5.189	4.333
125	0.091	0.899	0.202	0.148	-0.853	0.451	0.264	0.004	0.014	0.011	1.000	-5.692	3.747
126	0.091	1.000	0.202	0.149	-0.862	0.478	0.279	0.021	0.188	0.218	0.979	-3.572	3.970
127	0.092	0.499	0.202	0.148	-0.845	0.432	0.269	0.006	0.046	0.057	0.998	-4.831	3.595
128	0.092	0.100	0.106	0.451	-0.798	0.512	0.269	0.003	0.014	0.012	1.000	-5.276	4.255
129	0.092	0.501	0.000	0.505	-0.804	0.506	0.266	0.010	0.048	0.043	0.998	-4.515	4.203
130	0.092	0.000	0.106	0.451	-0.793	0.522	0.252	0.004	0.016	0.013	0.999	-5.137	4.340
131	0.093	0.000	0.000	0.505	-0.812	0.470	0.292	0.002	0.012	0.010	1.000	-5.476	3.906
132	0.093	0.898	0.049	0.273	-0.862	0.460	0.267	0.004	0.019	0.017	0.999	-5.192	3.822
133	0.093	1.000	0.049	0.272	-0.864	0.469	0.262	0.003	0.014	0.012	1.000	-5.374	3.902
134	0.093	0.000	0.202	0.149	-0.826	0.427	0.248	0.006	0.028	0.024	0.999	-5.164	3.547
135	0.094	1.000	0.000	0.304	-0.902	0.430	0.261	0.014	0.068	0.071	0.996	-4.404	3.572
136	0.094	0.103	0.202	0.148	-0.832	0.433	0.248	0.005	0.023	0.021	0.999	-5.174	3.602
137	0.094	0.899	0.000	0.304	-0.890	0.421	0.268	0.010	0.047	0.049	0.998	-4.707	3.498
138	0.094	0.498	0.048	0.272	-0.866	0.447	0.247	0.002	0.010	0.010	1.000	-5.718	3.713
139	0.095	0.500	0.000	0.304	-0.862	0.462	0.253	0.008	0.054	0.061	0.997	-4.562	3.844
140	0.096	0.101	0.048	0.273	-0.855	0.435	0.237	0.003	0.021	0.019	0.999	-5.195	3.617
141	0.096	0.000	0.050	0.273	-0.856	0.427	0.237	0.005	0.021	0.019	0.999	-5.194	3.550
142	0.096	0.100	0.000	0.304	-0.855	0.453	0.227	0.005	0.024	0.025	0.999	-5.086	3.770
143	0.097	0.000	0.000	0.304	-0.845	0.461	0.285	0.005	0.046	0.054	0.998	-4.782	3.832
144	0.099	1.000	0.000	0.000	-1.222	0.333	0.180	0.001	0.006	0.005	0.999	-5.615	2.772
145	0.102	0.000	0.000	0.000	-1.114	0.353	0.173	0.001	0.007	0.006	0.999	-5.518	2.935
146	0.102	0.000	0.000	0.000	-1.129	0.427	0.112	0.006	0.046	0.041	0.994	-4.488	3.547
147	0.108	0.932	0.000	0.505	-0.881	0.562	0.361	0.005	0.021	0.017	0.999	-5.032	4.669
148	0.111	0.085	0.202	0.148	-0.902	0.467	0.212	0.007	0.029	0.023	0.999	-5.087	3.885
149	0.151	0.500	0.733	0.000	-1.060	0.653	0.851	0.005	0.043	0.039	0.993	-3.471	5.425
150	0.154	0.101	0.734	0.000	-1.079	0.678	0.658	0.005	0.021	0.019	0.999	-4.311	5.634
151	0.154	0.000	0.734	0.000	-1.060	0.677	0.637	0.006	0.023	0.018	0.999	-4.473	5.627
152	0.156	1.000	0.581	0.334	-1.201	0.701	1.072	0.001	0.004	0.003	0.999	-4.037	5.825
153	0.157	1.000	0.543	0.000	-1.139	0.460	1.045	0.010	0.044	0.035	0.999	-4.428	3.821
154	0.157	0.899	0.580	0.333	-1.185	0.694	0.955	0.002	0.007	0.006	0.999	-4.123	5.771
155	0.158	0.899	0.542	0.000	-1.127	0.564	0.927	0.006	0.042	0.037	0.999	-4.183	4.687
156	0.161	0.499	0.581	0.335	-1.170	0.711	0.791	0.003	0.017	0.016	0.999	-4.216	5.911
157	0.161	0.499	0.543	0.000	-1.128	0.646	0.692	0.004	0.017	0.015	0.999	-4.323	5.372
158	0.164	0.100	0.581	0.333	-1.141	0.681	0.660	0.005	0.017	0.014	0.999	-4.515	5.661
159	0.165	1.000	0.372	0.244	-1.211	0.632	0.819	0.007	0.033	0.030	0.999	-4.138	5.254
160	0.165	1.000	0.000	0.305	-1.220	0.580	0.559	0.004	0.014	0.012	0.999	-4.533	4.820
161	0.165	0.101	0.543	0.000	-1.103	0.622	0.597	0.003	0.013	0.010	0.999	-4.580	5.173
162	0.166	0.000	0.580	0.333	-1.157	0.640	0.661	0.011	0.048	0.048	0.999	-4.308	5.321
163	0.166	0.896	0.372	0.244	-1.206	0.638	0.800	0.007	0.024	0.020	0.999	-4.188	5.307

Continued on next page

	x_{LiSalt}	$x_{\mathrm{LiPF_6}}$	x_{EC}	x_{PC}	S_0	S_1	S_2	S_0 std	S_1 std	S_2 std	$R2$	log(MAE)	E_a
164	0.167	0.000	0.542	0.000	-1.102	0.623	0.563	0.004	0.021	0.021	0.999	-4.459	5.183
165	0.168	1.000	0.000	0.704	-1.284	0.696	0.949	0.013	0.044	0.033	0.999	-4.209	5.787
166	0.169	0.899	0.000	0.704	-1.275	0.708	0.882	0.005	0.026	0.025	0.999	-4.202	5.886
167	0.169	1.000	0.337	0.000	-1.190	0.589	0.670	0.003	0.012	0.012	0.999	-4.332	4.897
168	0.170	1.000	0.000	0.305	-1.224	0.582	0.608	0.003	0.009	0.007	0.999	-4.601	4.841
169	0.170	0.498	0.372	0.243	-1.178	0.637	0.652	0.001	0.008	0.008	0.999	-4.401	5.297
170	0.171	0.897	0.336	0.000	-1.184	0.596	0.634	0.002	0.006	0.005	0.999	-4.491	4.957
171	0.173	0.498	0.000	0.704	-1.257	0.730	0.715	0.007	0.027	0.023	0.999	-4.344	6.067
172	0.173	1.000	0.000	0.505	-1.235	0.641	0.750	0.002	0.008	0.006	0.999	-4.356	5.329
173	0.174	1.000	0.204	0.149	-1.224	0.576	0.650	0.005	0.021	0.017	0.999	-4.395	4.790
174	0.174	0.899	0.205	0.150	-1.241	0.547	0.622	0.015	0.055	0.043	0.998	-4.235	4.552
175	0.175	0.500	0.337	0.000	-1.184	0.575	0.566	0.003	0.011	0.010	0.999	-4.675	4.784
176	0.175	0.000	0.375	0.243	-1.160	0.638	0.530	0.003	0.011	0.008	0.999	-4.701	5.302
177	0.175	0.100	0.372	0.243	-1.167	0.662	0.540	0.004	0.016	0.014	0.999	-4.474	5.506
178	0.177	0.100	0.000	0.704	-1.244	0.676	0.622	0.002	0.008	0.006	0.999	-4.592	5.619
179	0.177	0.898	0.000	0.506	-1.295	0.680	0.780	0.006	0.024	0.020	0.999	-4.264	5.652
180	0.178	0.000	0.000	0.704	-1.225	0.674	0.606	0.009	0.041	0.041	0.999	-4.413	5.606
181	0.178	0.101	0.337	0.000	-1.154	0.569	0.475	0.001	0.004	0.003	1.000	-4.889	4.729
182	0.179	0.000	0.336	0.000	-1.166	0.548	0.478	0.007	0.028	0.022	0.999	-4.838	4.560
183	0.180	0.501	0.203	0.149	-1.201	0.587	0.531	0.008	0.025	0.018	0.999	-4.775	4.881
184	0.180	0.495	0.000	0.504	-1.241	0.654	0.619	0.004	0.015	0.013	0.999	-4.453	5.437
185	0.183	0.000	0.000	0.501	-1.215	0.602	0.529	0.012	0.058	0.050	0.998	-4.295	5.002
186	0.183	0.102	0.204	0.149	-1.179	0.599	0.438	0.003	0.014	0.012	1.000	-5.056	4.984
187	0.184	0.000	0.204	0.149	-1.181	0.592	0.425	0.003	0.011	0.008	0.999	-4.800	4.924
188	0.185	0.499	0.000	0.305	-1.234	0.572	0.493	0.003	0.011	0.009	0.999	-4.822	4.754
189	0.185	0.101	0.000	0.505	-1.238	0.633	0.522	0.008	0.031	0.025	0.999	-4.455	5.264
190	0.188	1.000	0.000	0.000	-1.311	0.530	0.368	0.003	0.012	0.009	0.999	-4.932	4.404
191	0.189	0.103	0.000	0.305	-1.221	0.558	0.440	0.003	0.011	0.009	0.999	-4.891	4.635
192	0.189	0.000	0.000	0.305	-1.220	0.562	0.417	0.005	0.018	0.015	0.999	-4.908	4.669
193	0.199	0.000	0.000	0.000	-1.262	0.504	0.310	0.002	0.008	0.007	1.000	-5.411	4.191
194*	0.320	0.899	0.315	0.032	-1.813	2.502	-0.373	0.006	0.071	0.105	0.968	-2.368	20.805
195	0.333	0.498	0.336	0.000	-1.931	1.523	0.851	0.006	0.044	0.037	0.999	-4.052	12.665
196	0.334	0.500	0.314	0.031	-1.918	1.270	1.084	0.004	0.019	0.019	1.000	-4.801	10.561
197	0.335	0.896	0.045	0.273	-1.906	1.351	1.046	0.005	0.027	0.031	1.000	-4.652	11.237
198	0.341	0.499	0.202	0.148	-1.985	0.800	1.722	0.061	0.726	1.093	0.987	-2.936	6.651
199	0.346	0.099	0.338	0.000	-1.855	1.115	0.895	0.002	0.010	0.012	1.000	-4.633	9.269
200	0.347	0.495	0.059	0.272	-1.982	1.308	0.922	0.005	0.051	0.076	1.000	-4.943	10.874
201	0.350	0.000	0.337	0.000	-1.838	1.104	0.855	0.002	0.022	0.032	1.000	-4.664	9.178
202	0.355	0.100	0.201	0.150	-1.897	1.118	0.840	0.002	0.022	0.026	0.999	-4.402	9.299
203	0.356	0.482	0.000	0.305	-1.986	1.340	0.883	0.003	0.017	0.017	1.000	-4.758	11.137
204	0.359	0.000	0.203	0.150	-1.871	1.088	0.828	0.001	0.009	0.011	0.999	-4.287	9.046
205	0.359	0.000	0.313	0.032	-1.796	1.019	0.801	0.003	0.013	0.012	0.999	-4.299	8.475
206	0.362	0.100	0.048	0.272	-1.913	1.081	0.831	0.004	0.029	0.032	1.000	-4.633	8.991
207	0.366	0.000	0.048	0.272	-1.906	1.039	0.818	0.004	0.025	0.029	0.999	-4.359	8.641
208	0.367	0.000	0.000	0.304	-1.914	1.026	0.839	0.005	0.044	0.054	0.999	-4.304	8.534
209	0.373	0.096	0.000	0.305	-1.937	1.107	0.813	0.004	0.017	0.018	1.000	-4.896	9.207
210	0.381	0.000	0.000	0.000	-1.924	0.884	0.745	0.008	0.045	0.051	0.999	-4.310	7.350

S5 Feature correlation

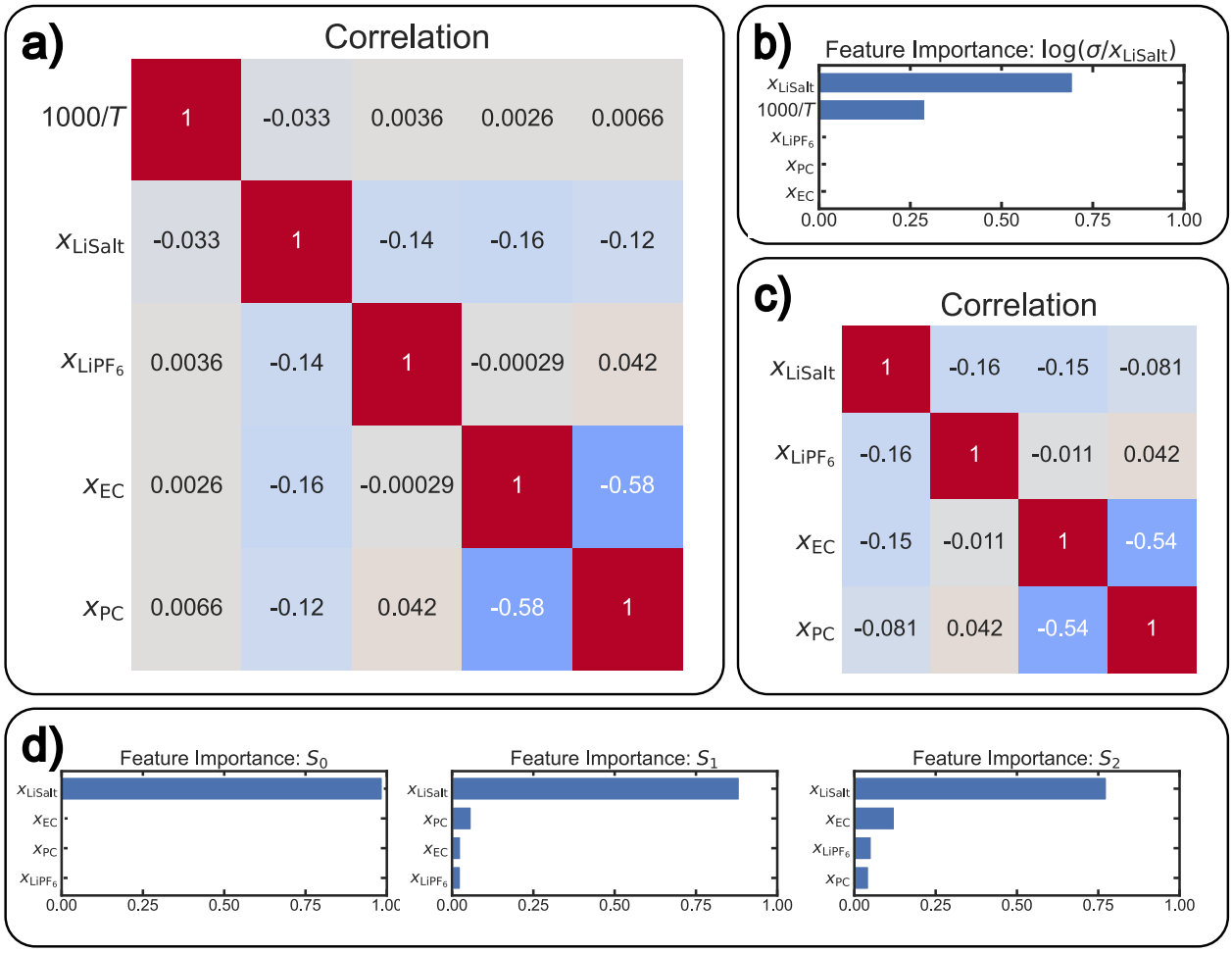


Figure S12: (a,b): Feature correlations and Feature importance for features before the Arrhenius fit, using each individual measured data point for the calculation. Due to different numbers of measurements for each electrolyte formulation, caused by different measurement uncertainties for different electrolyte formulations, every electrolyte formulation is weighted differently. (c,d): Feature correlations and Feature importance for features after the Arrhenius fit, removing data points with an insufficient Arrhenius fit and using averaged data for each electrolyte composition. Each electrolyte formulation is weighted equally. The correlations are calculated using the Pandas library with the Pearson method and the Feature importance is calculated by a simple Random Forest model as implemented in Scikit-Learn.

S6 Feature space coverage

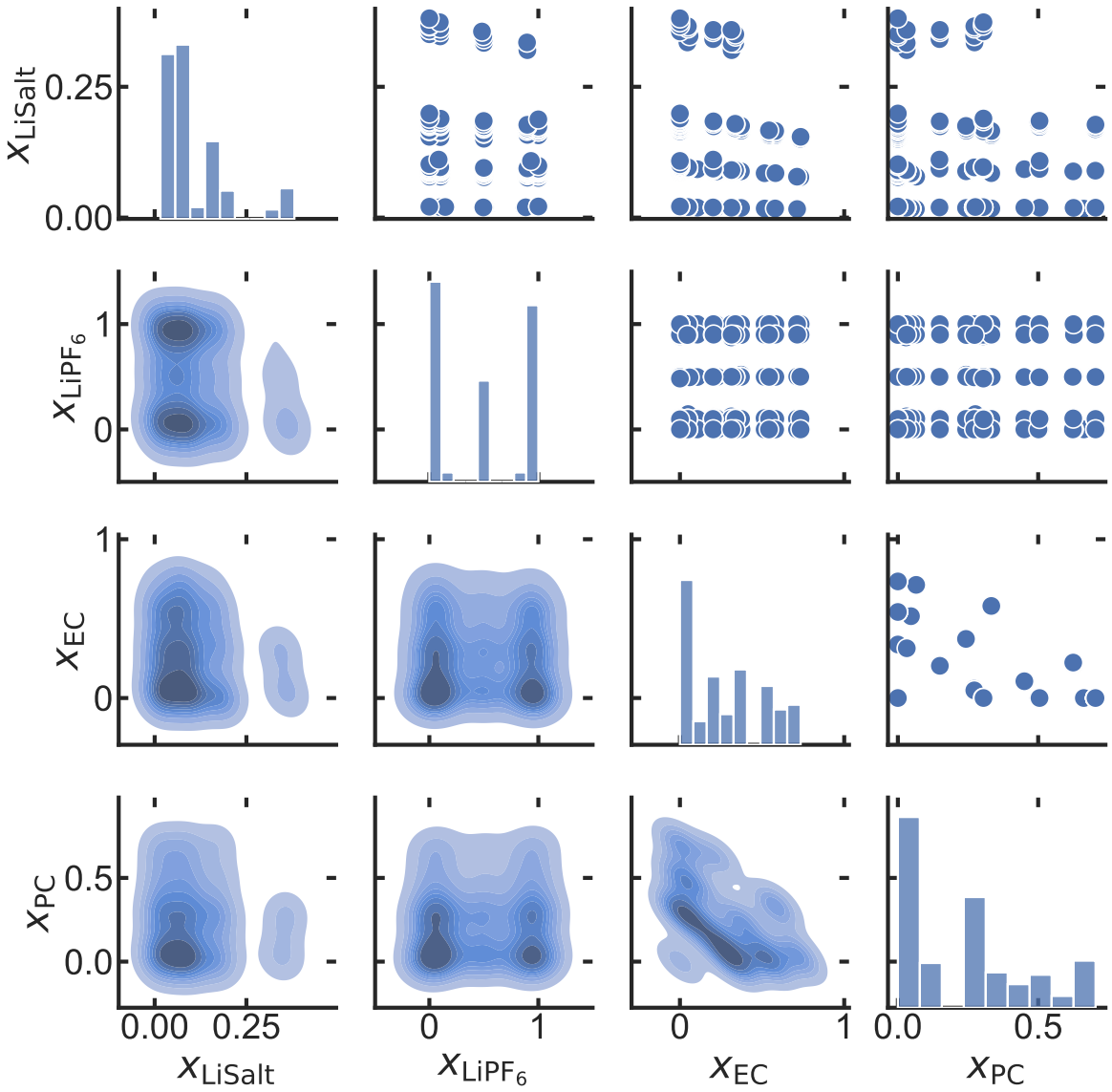


Figure S13: Pairplots to illustrate feature space coverage of all features used for the predictions of Arrhenius objectives S_0, S_1 and S_2. The upper diagonal shows explicit data points, and the lower diagonal shows the density of data.

S7 Model description

This section aims to describe the models used in our paper briefly, focussing on hyperparameter settings. For a deeper understanding of each model, we refer to the documentation of scikit-learn [4]. For all models, we use a random seed of 42 to ensure our results are reproducible.

S7.1 Random Forest (RF)

The Random Forest (RF) [16] model is based on an ensemble of decision trees. We use the standard implementation of scikit-learns RandomForestRegressor with its default hyperparameters. The most important ones are the number of 100 decision trees, a number of 2 samples per node to split it further and a minimum number of 1 sample per leaf. For building the decision trees, bootstrapped samples are used.

S7.2 Optimized Linear Regression (LR opt)

The optimized Linear Regression model (LR opt) is constructed from a polynomial of degree 3. The polynomial includes all possible combinations of features up to this degree.

$$p = \sum_i a_i x_i + \sum_{ij} a_{ij} x_i x_j + \sum_{ijk} a_{ijk} x_i x_j x_k \tag{S4}$$

with $i, j, k \in [x_{\mathrm{LiSalt}}, x_{\mathrm{LiPF_6}}, x_{\mathrm{EC}}, x_{\mathrm{PC}}]$ and corresponding weights a. We then remove polynomials, which do not increase the training error, when removed. Therefore, we calculate the training error η^2 (MSE) for infinite training size N_{inf}, by performing a linear regression on the function

$$\eta^2 = m \cdot \frac{1}{N} + b \tag{S5}$$

with the number of training data N, slope m and intercept b, which corresponds directly to η^2_{inf}. The detailed method is described in our previous work [1].

We perform the regression with 20 different values for N with each trained excluding one polynomial from the total set. The polynomial whose elimination results in the minimal estimated error for N_{inf}. Recursively, this process is repeated until removing any polynomial results in an increased η^2_{inf} larger than 10 % from the best error N_{inf} seen so far in the whole process. We start with removing polynomials of higher degrees and then proceed with polynomials of lower degrees. Below are the tables of polynomials considered for the optimized model. It is remarkable, that only a few polynomials include $x_{\mathrm{LiPF_6}}$, indicating that it is less important. For S_1 it is only one single polynomial.

Table S3: Polynomials in Optimized LR model for S_0.

poly_index	x_{LiSalt}	$x_{\mathrm{LiPF_6}}$	x_{EC}	x_{PC}	coefs	std	rel_std
*	0	0	0	0	-0.875023	0.002235	0.002554
0	1	0	0	0	12.218531	0.212923	0.017426
1	0	1	0	0	-0.112419	0.005834	0.051893
2	0	0	1	0	6.734287	0.064513	0.009580
3	0	0	0	1	7.185093	0.069567	0.009682
4	2	0	0	0	-49.873777	1.059288	0.021239
5	1	1	0	0	-0.389274	0.017569	0.045132
6	1	0	1	0	-41.903746	0.505475	0.012063
7	1	0	0	1	-43.582825	0.532575	0.012220
9	0	1	1	0	0.198567	0.008227	0.041433
10	0	1	0	1	0.168711	0.008414	0.049870
11	0	0	2	0	-8.255223	0.113815	0.013787
12	0	0	1	1	-17.088289	0.197576	0.011562
13	0	0	0	2	-9.980385	0.128979	0.012923
14	3	0	0	0	49.882205	1.209635	0.024250
16	2	0	1	0	59.585975	1.254787	0.021058
17	2	0	0	1	59.743359	1.294362	0.021665
21	1	0	2	0	23.692574	0.331104	0.013975
22	1	0	1	1	46.543972	0.559897	0.012029
23	1	0	0	2	27.588520	0.373083	0.013523
30	0	0	3	0	3.381360	0.068299	0.020199
31	0	0	2	1	9.256496	0.151613	0.016379
32	0	0	1	2	11.314440	0.167111	0.014770
33	0	0	0	3	4.500586	0.079047	0.017564

Table S4: Polynomials in Optimized LR model for S_1.

poly_index	x_{LiSalt}	$x_{\mathrm{LiPF_6}}$	x_{EC}	x_{PC}	coefs	std	rel_std
*	0	0	0	0	0.534251	0.001053	0.001971
2	0	0	1	0	1.146745	0.024096	0.021013
3	0	0	0	1	0.393108	0.003765	0.009577
4	2	0	0	0	5.081773	0.041250	0.008117
11	0	0	2	0	-2.958723	0.084152	0.028442
12	0	0	1	1	-1.398687	0.029247	0.020910
15	2	1	0	0	2.778388	0.112186	0.040378
21	1	0	2	0	1.936693	0.049419	0.025517
30	0	0	3	0	2.248034	0.075330	0.033509
31	0	0	2	1	2.327605	0.055255	0.023739

Table S5: Polynomials in Optimized LR model for S_2.

poly_index	x_{LiSalt}	$x_{\mathrm{LiPF_6}}$	x_{EC}	x_{PC}	coefs	std	rel_std
*	0	0	0	0	0.397774	0.001366	0.003435
0	1	0	0	0	-4.112804	0.071528	0.017391
1	0	1	0	0	-0.138173	0.004075	0.029494
3	0	0	0	1	0.081101	0.006142	0.075733
4	2	0	0	0	37.620234	0.497493	0.013224
5	1	1	0	0	1.780458	0.039904	0.022412
7	1	0	0	1	1.301105	0.043369	0.033332
9	0	1	1	0	0.285936	0.006658	0.023285
10	0	1	0	1	0.142192	0.006149	0.043246
11	0	0	2	0	0.736755	0.005841	0.007928
12	0	0	1	1	-3.034302	0.034587	0.011399
14	3	0	0	0	-63.381723	0.907281	0.014315
22	1	0	1	1	13.287443	0.183829	0.013835
32	0	0	1	2	3.235468	0.054295	0.016781

S7.3 Gaussian Process Regression (GPR)

For Gaussian Process Regression[17–19], we employ different kernels to test which works best with the given data. The hyperparameters of each kernel (typically a length scale) are optimized automatically by scikit-learn when fitting it to the data within the given boundary (10^{-5}, 10^{5}). If available, we use the isotropic kernel (one length scale for all dimensions) and the anisotropic kernel (one length scale per dimension).

S7.3.1 Radial basis function kernel (A.RBF and I.RBF)

The Radial Basis Function (RBF) kernel (sometimes also Squared Exponential or Gaussian kernel)[17,24] is given as

$$k(x_i, x_j) = \exp\left(-\frac{|x_i - x_j|^2}{2l^2}\right) \tag{S6}$$

with the length scale l. $|x_i - x_j|$ denotes the euclidean distance.

S7.3.2 Matérn kernel (A.M and I.M)

The Matèrn kernel[17,25] is given as

$$k(x_i, x_j) = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\frac{\sqrt{2\nu}}{l}|x_i - x_j|\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}}{l}|x_i - x_j|\right) \tag{S7}$$

with the gamma function Γ and a Bessel function K_ν. ν is a parameter that can be chosen. We chose $\nu = 1.5$ for our analysis, which results in a one times differentiable function. l is again a length scale.

S7.3.3 Rational Quadratic kernel (I.RQ)

The Rational Quadratic (RQ) kernel[17,24] is given as

$$k(x_i, x_j) = \left(1 + \frac{|x_i - x_j|}{2\alpha l^2}\right)^{-\alpha} \tag{S8}$$

with the length scale l and a mixture parameter a. Both are optimized during the fitting process.

S7.4 Neural network (NN)

Based on a more detailed analysis of different Neural Network[15,20] models (section S9) we have chosen one specific model for further investigations. The chosen model has two hidden layers with 16 neurons per hidden layer. We use the Rectified Linear Unit (ReLU)[26] activation function for the output layer and hyperbolic tangent activation functions for hidden layers. We use a constant learning rate of 0.001, a value of 0.008 for L2 regularization, and the L-BFGS[27] solver, which is recommended for small datasets. The number of maximum iterations is set to 4000. For other parameters, we adopt the defaults from scikit-learn.

S7.5 Optimized neural network (NN opt)

To automatically determine hyperparameters for NN models, we employ Bayesian hyperparameter Optimization using the GPyOpt[2] library. It automatically varies the network architecture up to 4 hidden neurons with up to 20 neurons. Different activation functions for the hidden layers can be evaluated, choosing from ReLU[26], hyperbolic tangent[28], logistic and identity. The value for L2 regularization can be chosen from the interval (0.0001, 0.01). The batch size is varied between 10 and 200 in steps of 5. The solver can be chosen from 'L-BFGS'[27], 'Adam'[29] and stochastic gradient descent with a number of maximum iterations between 500 and 5000, varied in steps of 500. Other parameters are not varied and adopted from scikit-learn's default. We search for optimized hyperparameters for a maximum of 300 seconds or 100 iterations, whatever condition is first met. Depending on the random seed, each time other optimized parameters are found. For a random seed of 42 we find the optimized hyperparameters listed below and used for further analysis.

Table S6: Optimized hyperparameters for model to predict S_0, S_1 and S_2.

Hyperparameter	S_0	S_1	S_2
Neurons in hidden layer 1	13	8	14
Neurons in hidden layer 2	14	8	9
Neurons in hidden layer 3	16	3	12
Neurons in hidden layer 4	7	15	11
Activation function	hyperbolic tangent	ReLU	ReLU
L2 regularization	0.0068	0.01	0.01
Batch size	10	10	10
Solver	L-BFGS	Adam	Adam
Maximum number of iterations	500	4500	3000

S8 Train-Test split

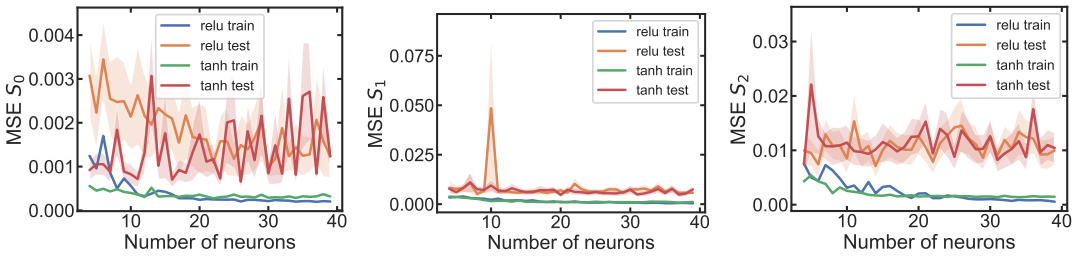
To compare several models and ensure reproducibility we ensure that for a similar random state each time equal data is used for the training and the test set. With our random state of 42, we obtain the following split of data points listed below. The indices of the data points correspond to the Arrhenius fits that are listed in Table S2.

Table S7: Indices of electrolyte formulations that were used for training and test set of the final model. The formulations were chosen randomly.

Train	Test (Validation holdout)
0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 16, 17, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 74, 76, 77, 78, 79, 80, 81, 82, 83, 85, 86, 87, 88, 89, 90, 91, 92, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 127, 128, 129, 130, 131, 132, 133, 134, 135, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 156, 157, 158, 159, 160, 161, 162, 163, 164, 166, 167, 169, 170, 171, 173, 174, 175, 176, 177, 178, 179, 180, 181, 183, 184, 185, 186, 188, 189, 190, 191, 192, 193, 195, 196, 198, 199, 201, 202, 203, 204, 205, 206, 207, 209, 210	9, 15, 18, 30, 45, 60, 73, 75, 84, 93, 126, 136, 155, 165, 168, 172, 182, 187, 197, 200, 208

S9 Systematic Study of Neural networks

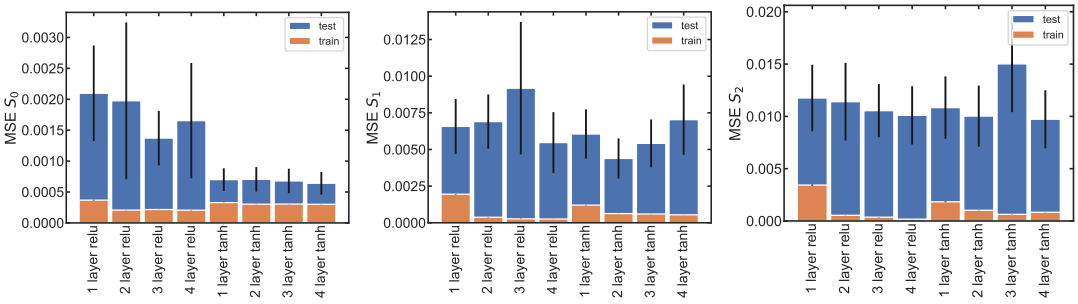
From all tested models Neural Networks offer the largest variety in hyperparameters. Often expert knowledge about the Neural Network itself and the dataset are required in order to achieve high accuracy. This can be circumvented by Bayesian hyperparameter optimization, leading to the NN opt model. However, we would like to understand the effect of different hyperparameters on the model, especially the Neural Network architecture in more detail. Therefore, we systematically investigate the effect of choosing different numbers of neurons per hidden layer. For simplicity we restrict ourselves to one hidden layer for this kind of analysis. Furthermore, we compare the ReLU and hyperbolic tangent activation functions. We calculate the averaged MSE from leave-one-out CV. As can be seen in Figure S14b and c both activation functions perform similarly for S_1 and S_2, nearly regardless of the number of neurons per hidden layer. Only for less than 10 neurons models might perform worse. For S_0 in Figure S14 we observe the trend that the hyperbolic tangent activation function gives more stable results for a number of neurons below 20. For more neurons the MSE increases slightly. However, there is one exception for 13 neurons per layer, where the hyperbolic tangent (tanh) activation function performs much worse. So far, this is not understood. For the ReLU activation function the MSE decreases, when increasing the number of neurons up to 20. Except for 13 neurons, the MSE for the ReLU activation function is larger than for the hyperbolic tangent. Starting from 20 neurons both perform similar. From this analysis we can identify several good architectures and decide to continue with 16 neurons per layer for all objectives S_i, as this seems to be the architecture for the hyperbolic tangent activation function with the lowest MSE for S_0.



(a) S_0 (b) S_1 (c) S_2

Figure S14: Dependence of MSE of network architecture. The number of nodes is varied for a NN with 1 hidden layer. The transparent areas indicate standard errors.

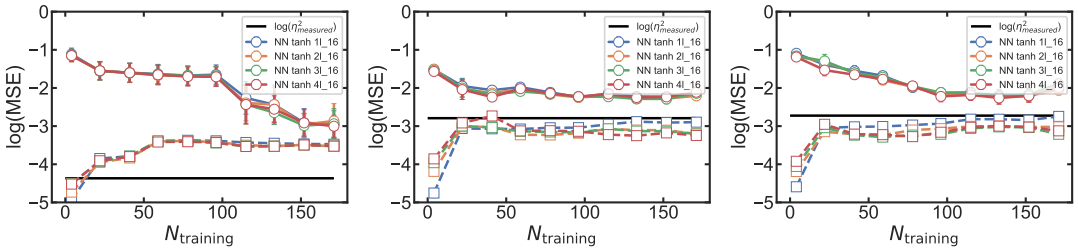
In Figure S15 we keep the number of neurons per layer constant, but vary the number of hidden layers from 1 to 4. For S_0 and ReLU three hidden layers give rise to the lowest MSE, but within standard errors (indicated by black bars) all models are similar. The similarity is even more pronounced for the hyperbolic tangent activation function. For S_1 and S_2 within standard errors all architectures are similar. However, for the hyperbolic tangent activation function and S_1 objective two hidden layers has the best accuracy.



(a) S_0 (b) S_1 (c) S_2

Figure S15: Dependence of MSE of network architecture. The number of hidden layers is varied with 16 neurons per hidden layer. Also, the two activation functions ReLU and hyperbolic tangent are compared.

Finally, we investigate in Figure S16 if the performance of NN models with 16 neurons per layer and different numbers of layers depends on the number of training data. Here, we observe that for a given number of training data all architectures perform similarly well within standard errors. A similarity is observed when varying the number of neurons in one hidden layer.

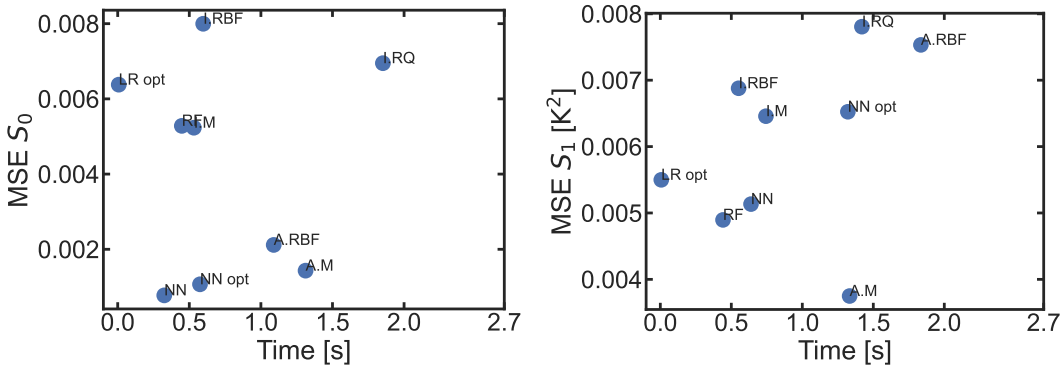


(a) S_0 (b) S_1 (c) S_2

Figure S16: Datasize performance of NN with different numbers of hidden layers and 16 neurons per hidden layer. The black line represents the experimental variance of the S_i. The error bars represent standard errors.

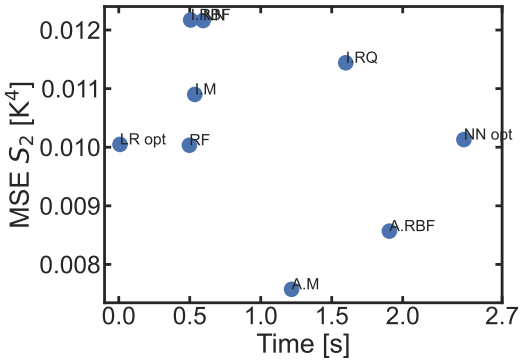
As a final note we should try to avoid too large Neural Network architectures to reduce the number of parameters and thus prevent overfitting. However, when choosing the Neural Network architecture too small it might be possible that the MSE increases.

S10 Model timings



(a) MSE vs. fitting time for S_0.

(b) MSE vs. fitting time for S_1.



(c) MSE vs. fitting time for S_2.

Figure S17: Average MSE from leave-one-out CV plotted against the average time to fit a model once for all models.

Beyond accuracy, also the computational cost to fit a model is of relevance. In general, for small datasets, this does not matter too much. However, for larger datasets it is necessary to have efficient models in both, accuracy and time needed to fit a model. Therefore, to find the optimal trade-off, we can plot the MSE of each S_i vs. the average time needed to fit a model as shown in Figure S17. The MSE and fitting times are calculated during the leave-one-out CV. We highlight that our models can be fitted on commonly available laptops or desktop computers and do not need any specific hardware. The analysis discussed in this paper and the measured timings in Figure S17 were done on an AMD Ryzen 7 4800H CPU with 8 cores (16 threads) and a maximum clock rate of 4.2 GHz.

The cheapest model is always LR opt needing less than 0.01 s. However, especially for S_0 it is one of the least accurate models. There the NN and NN opt models are superior in terms of accuracy and compared to all other models except LR opt also in fitting time (0.4 and 0.7 s). Thus the NN models are the best option for fitting S_0. GPR models are in general more expensive, where the exact fitting time depends on the chosen kernel.

However, NN models might be also more expensive, depending on the maximum number of iterations and network architecture. However, the fitting time only scales with N, whereas it scales with N^3 for GPR models.

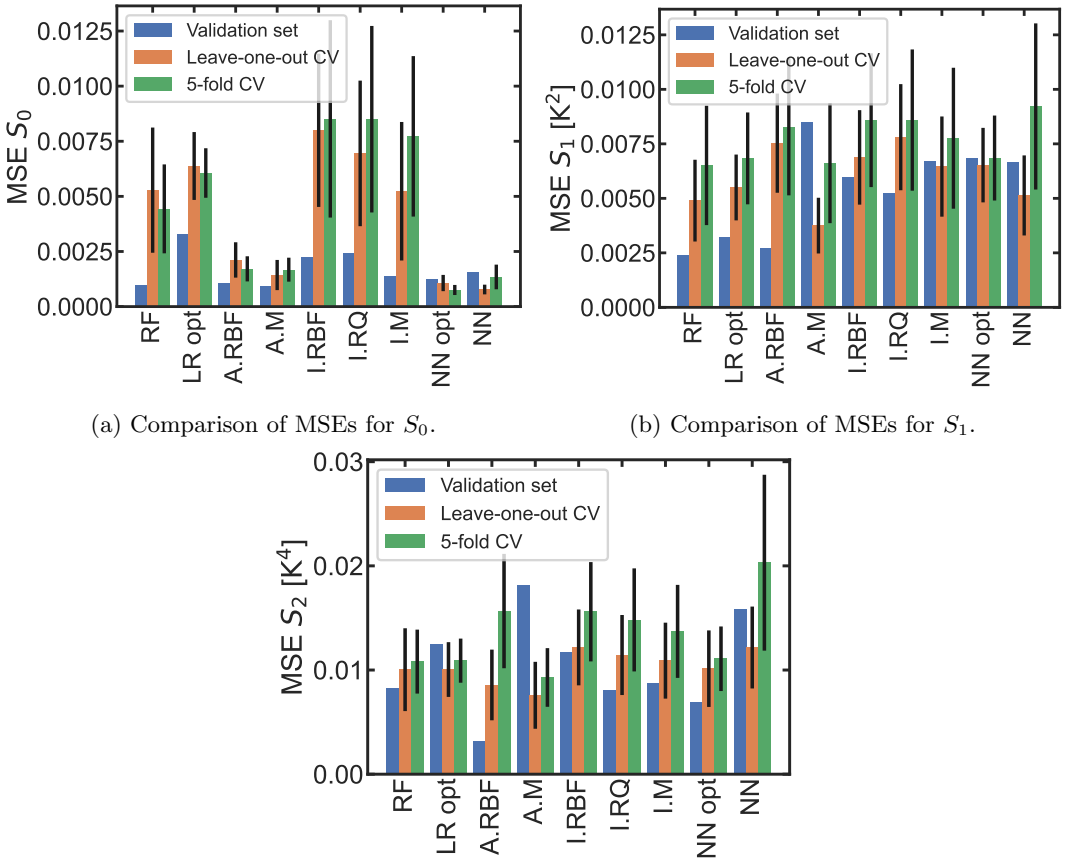
For S_1 and S_2 A.M has the lowest MSE, however, it is also expensive in terms of computation time. Here the NN model is superior in terms of fitting time (0.7 s for S_1 and 0.6 s for S_2) with still acceptable MSEs. The NN opt model needs more time to fit, which is explained by more hidden layers and a higher number of maximum iterations. Besides NN the I.M model offers a good balance between fitting time (0.5 s for S_1 and 0.7 s for S_2) and MSE.

Interestingly, the preferable models with respect to fitting times are exactly the models we identified for further analysis in our paper only based on the MSE and physical reasonability.

Finally, when increasing the size of the dataset and thus also computational costs, it is possible to switch from leave-one-out CV to K-fold CV, lowering the computational costs to obtain statistics by simultaneously also losing some statistical significance. However, this is a common approach in the ML community. This would not reduce individual fitting times of models, but the total time to calculate CV scores.

S11 Model validation

To validate our model, we use different representations. First, we start by comparing the Mean Squared Errors (MSEs) calculated from the leave-one-out CV to the MSEs obtained for the test set. In Figure S18, we also compare the MSEs calculated from leave-one-out CV with the MSEs calculated from a 5-fold CV. In general, both are in good agreement, however, the MSEs from leave-one-out CV are mostly lower because we used more data to train a model, which increases the models' performance. The MSEs for the test set are also in good agreement within the standard errors of the MSEs calculated from CV.



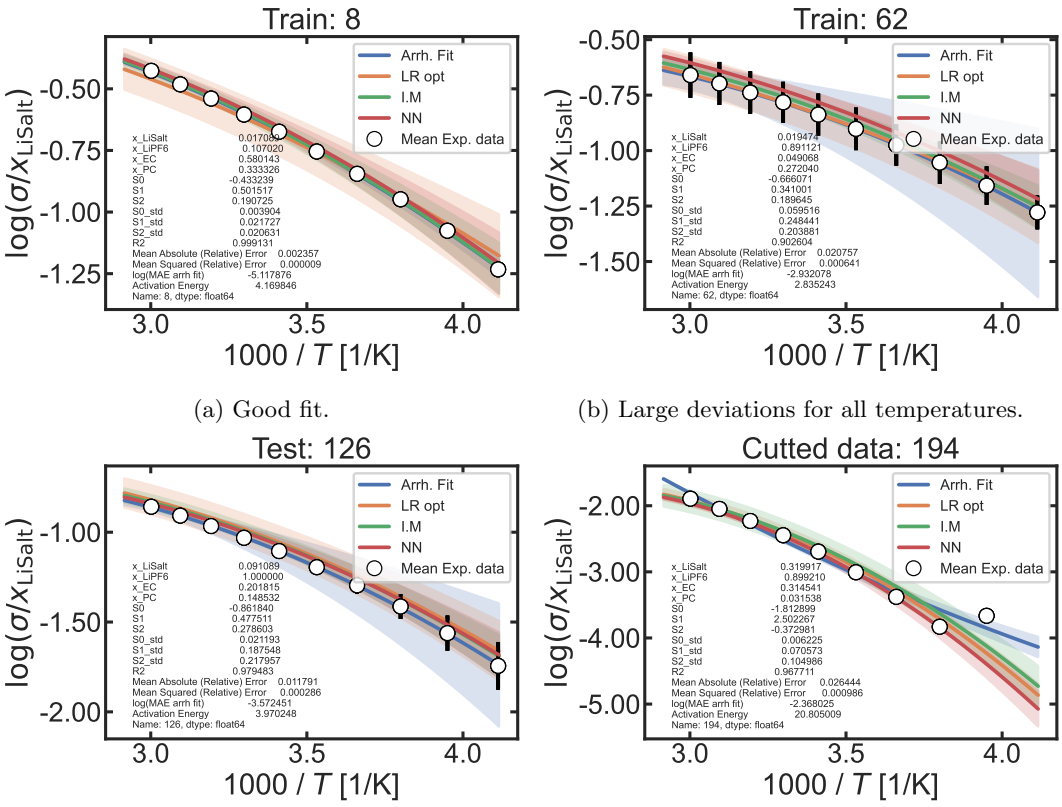
(a) Comparison of MSEs for S_0.

(b) Comparison of MSEs for S_1.

(c) Comparison of MSEs for S_2.

Figure S18: Comparison of the MSEs for individual S_i, calculated from leave-one-out CV, 5-fold CV and calculated for the test set. The black bars correspond to standard errors.

Second, we can calculate the model's prediction for a certain electrolyte composition for every measured temperature. If there is a measurement for this specific composition, we can compare both. Figure S19a and b show typical Arrhenius fits and model predictions for the Arrhenius fits for electrolyte compositions within the training set, which are expected to be fitted well. Figure S19c and d show the predicted Arrhenius fit for a composition from the test set (c), which the models have never seen before. The measurement data, the experimental Arrhenius fit and the predicted Arrhenius fits of all models agree well for this specific composition. We observe similar good predictions for the electrolyte composition which was filtered out (d) due to a large Mean Absolute Error for the Arrhenius fit, which is caused by the measurement at −30°C. Here, the model predicts different behavior. Yet, it is not clear whether the measurements are incorrect or there is an interesting behavior for this specific electrolyte composition, deviating from normal Arrhenius behavior.



(a) Good fit.

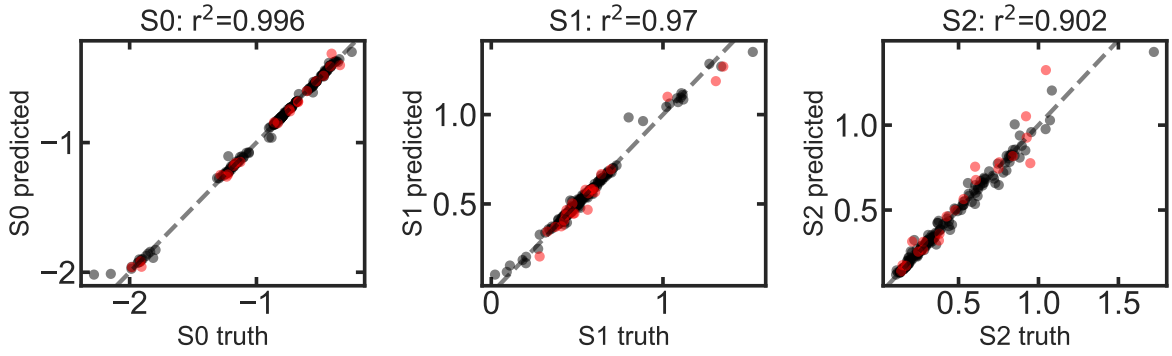
(b) Large deviations for all temperatures.

(c) Large deviations for low temperatures.

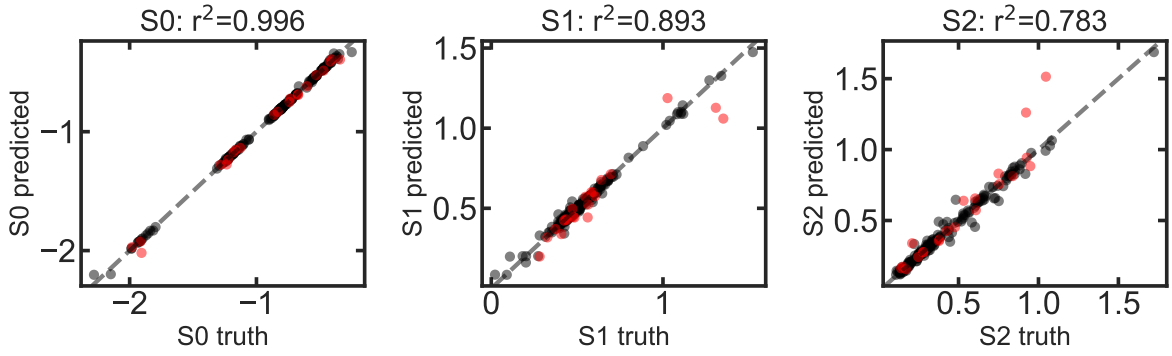
(d) Unphysical fit.

Figure S19: Prediction of Arrhenius fits. The black open circles represent the mean of the individual measurements with their standard deviation as error bars. The Arrhenius and model predictions are displayed together with their uncertainties. For each subplot it is given, if the electrolyte composition belongs to the training or test set or was filtered out before the train-test-split.

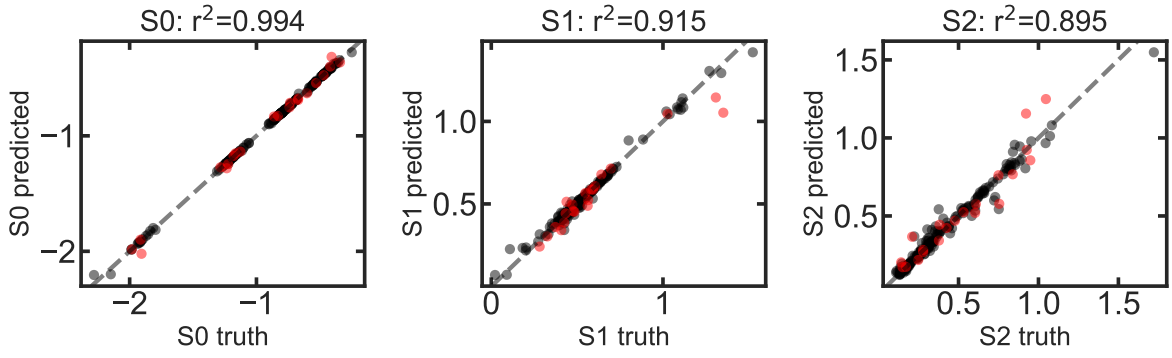
In Figure S20 we plot for each individual S_i the predicted value against the true value for each electrolyte composition in the training set (black circles) and in the test set (red circles). We show these plots for all models that are used to make predictions (see e.g. Figure 6 in the main paper). Furthermore, we calculate the R2-score (a score of 1 is a perfect fit) for the validation set. From these plots we observe that the S_0 (all R2-scores > 0.98) objective function is easier to fit than the S_1 objective function, which is easier to fit than the S_2 objective function. This is indicated by generally decreasing R2-scores from S_0 to S_2. Only in the NN opt model the R2-scores for S_1 and S_2 are similar, which might be related to statistical reasons and separate optimization for each objective. Because the R2-scores are only calculated for one test set without another level of CV they can not be used to compare models, but indicate that all models are trained well without too much overfitting. A general trend is also that lower values of S_i in the test set are fitted with a lower absolute error than larger values, which can be especially seen for S_1 and S_2.



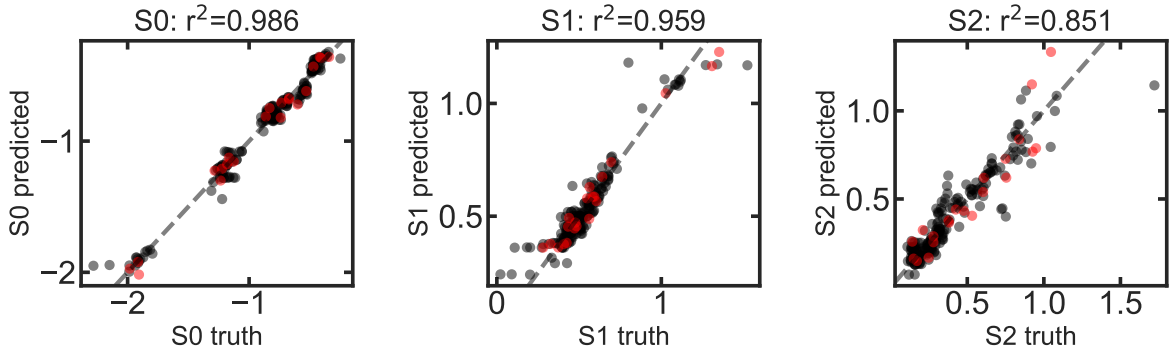
(a) Random Forest.



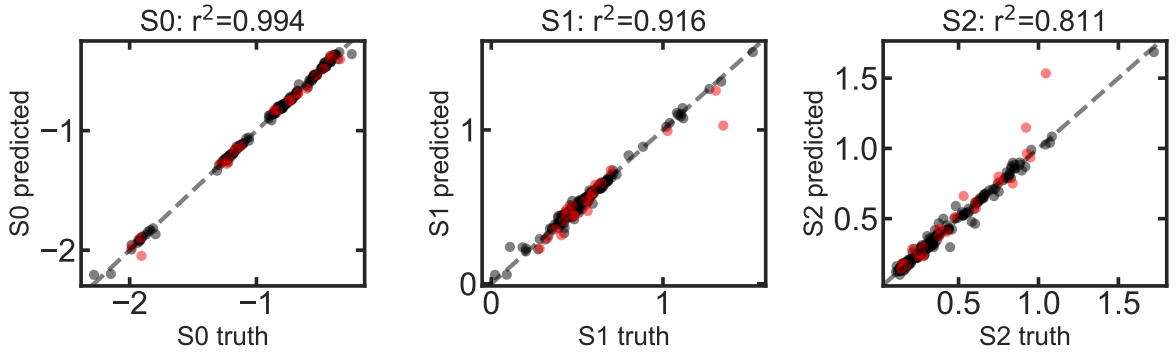
(b) GPR with Anisotropic Matern kernel.



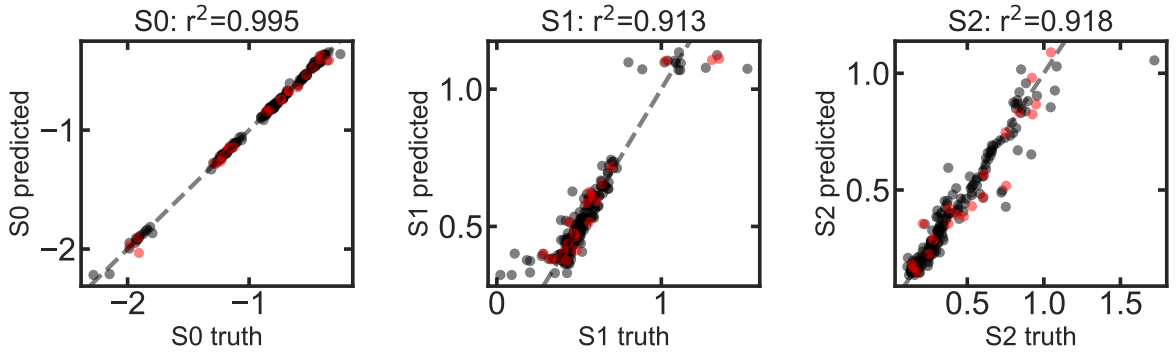
(c) GPR with Isotropic Matern kernel.



(d) Optimized Linear Regression.



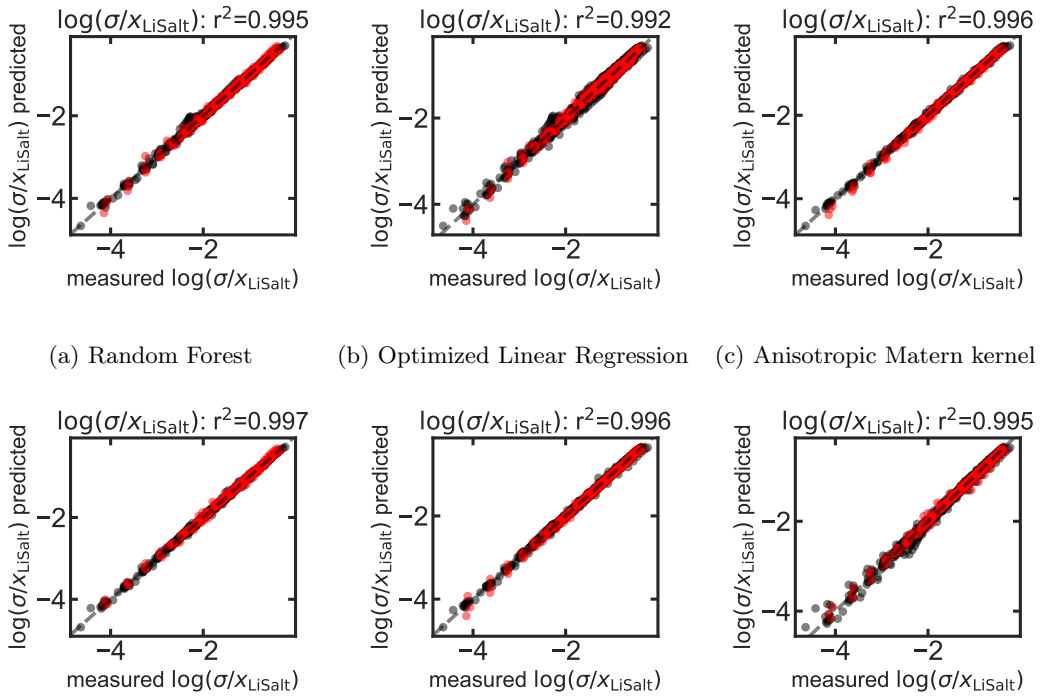
(e) Neural Network.



(f) Optimized Neural Network.

Figure S20: R2 score of individual S_0, S_1 and S_2, for similar models as in Figure 5 in the main paper.

Besides R2-scores for individual S_i we can also calculate the R2-score for the back-transformed $\log \frac{\sigma}{x_{\text{LiSalt}}}$ by showing also the prediction versus the true value in Figure S21 for similar models as in Figure S20. For all models, we observe a R2-score in the test set larger than 0.99 indicating that the models are fitted well. The plots for Random Forest, Optimized Neural Network and the Optimized Linear Regression also show some difficult to fit electrolyte compositions around $\log \frac{\sigma}{x_{\text{LiSalt}}} \approx 2$, indicated by small deviations from the diagonal. This can not be seen for other models.



(a) Random Forest (b) Optimized Linear Regression (c) Anisotropic Matern kernel

(d) Isotropic Matern kernel (e) Neural Network (f) Optimized Neural Network

Figure S21: R^2-scores of the ionic conductivity for chosen models.

S12 Datasize Performance

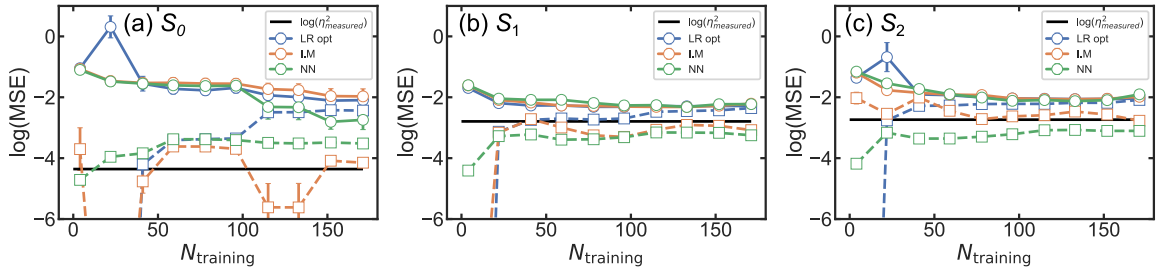


Figure S22: Datasize performance plots for S_0, S_1 and S_2. The straight line represents the MSE in the test set and the dashed line is the MSE in the train set. The black line represents the mean standard deviations of the S_i from the randomized 200 Arrhenius fits for each composition. The error bars represent standard errors across randomized training samples.

The size of the training dataset can affect the models' performance significantly. It is intuitively clear that models with a larger number of parameters, like NN, need more data for training than models with only a few parameters like I.M or LR opt. Figure S22 shows the averaged MSE over 100 repetitions in the test set (straight line) and training set (dashed line) as a function of the number of randomly picked data points for training the objectives S_0 (a), S_1 (b) and S_2 (c).

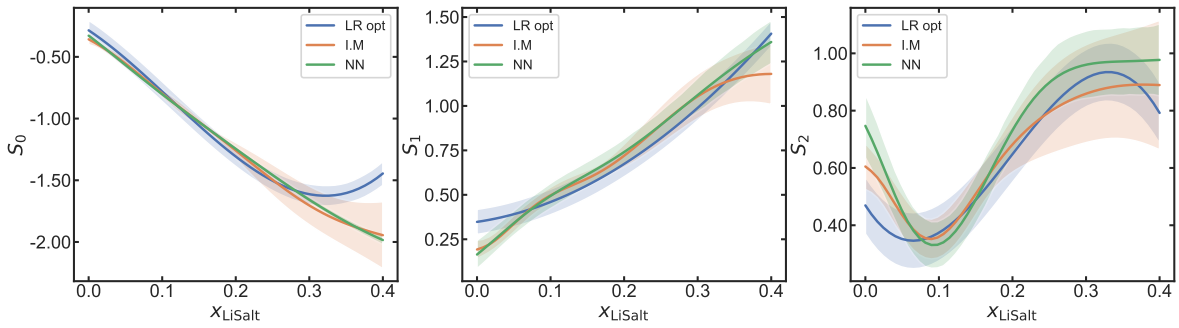
The largest dependence on the number of data points can be observed for the S_0 objective. For up to 100 data points, all models perform similarly in terms of the MSE of the test set, except for 22 data points, where the LR opt model performs worse. However, when increasing the number of data points in the training set, the NN model profits the most, whereas there is only minor improvement for the LR opt and I.M models. For more than 100 data points MSE in train and test set approach each other, indicating that the LR opt model would not benefit significantly from even more training data. For the NN model, the MSE in the train set is stable for a various range of training set sizes, approaching $3.0 \cdot 10^{-3}$ for 171 data points. However, we do not expect that the NN model hits experimental variance (black line) which is $4.4 \cdot 10^{-5}$, when adding more data, highlighting the very high experimental accuracy.

For S_1 and S_2 all three models perform roughly similarly in terms of MSE of the test set for any given number of training data larger than approximately 100. For both, S_1 and S_2, we observe that the MSE in the test set is approximately one order of magnitude larger than the MSE in the training set, which approaches the experimental variance of $1.6 \cdot 10^{-3}$ for S_1 and $1.8 \cdot 10^{-3}$ for S_2, respectively.

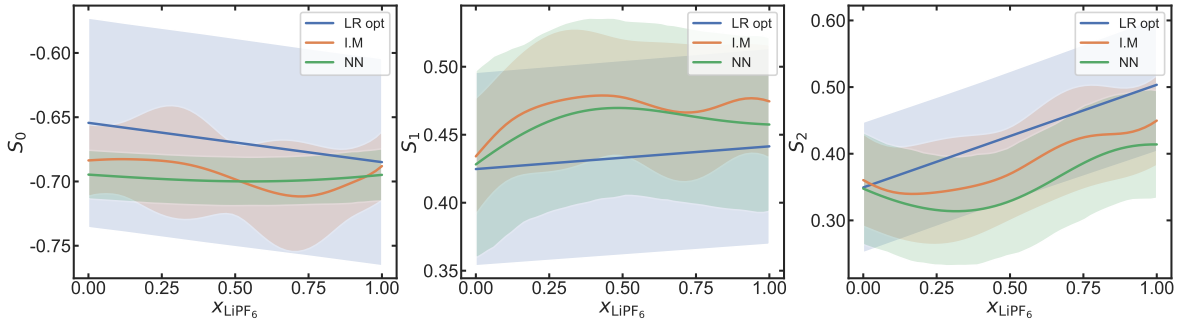
That the experimental variance can only be achieved for the S_1 and S_2 and not for S_0 objective reflects on the one hand a higher precision in measuring S_0 experimentally and on the other hand the more complicated dependencies of S_0 on the composition. Although S_0 is mainly determined by the salt concentration, the other components play also a minor and more subtle role, which needs to be understood in more detail. This was already observed in our previous work [1], where a polynomial of third order was needed to fit S_0^*, whereas a polynomial of the second and first order, respectively, were sufficient to fit S_1 and S_2, respectively. To sum this up, already a few data points are enough to fit meaningful models for S_1 and S_2 with experimental accuracy but more data is needed to fit models for S_0. Also, our models would benefit from more data.

S13 Model predictions of individual S_0, S_1 and S_2

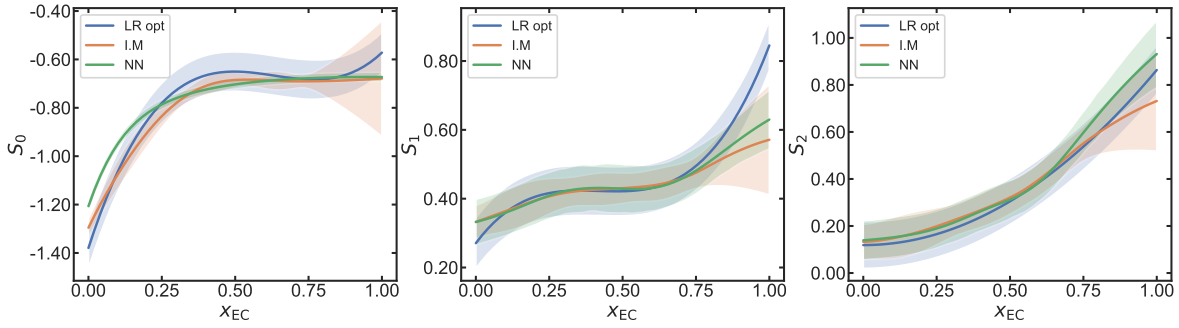
S13.1 1D slices



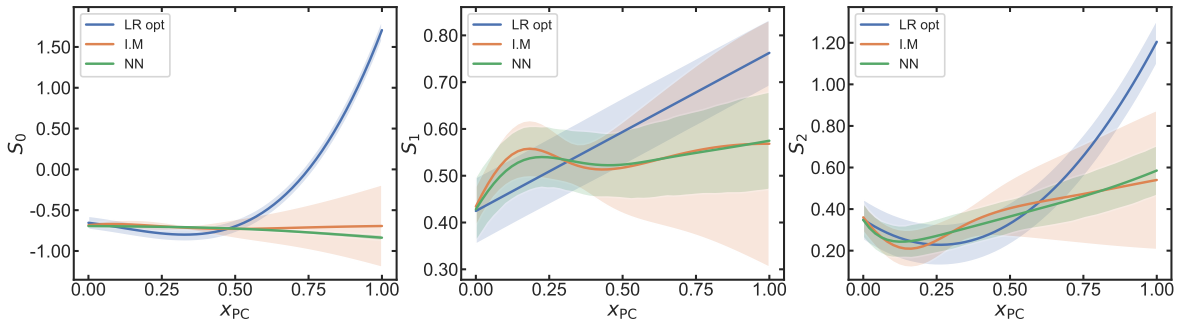
(a) x_{LiSalt}: varied, $x_{\mathrm{LiPF_6}}$=0, x_{EC}=0.56, x_{PC}=0.



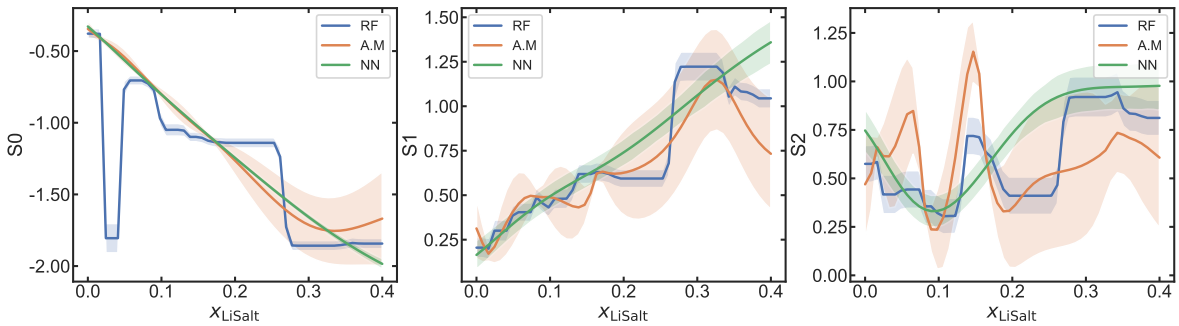
(b) x_{LiSalt}=0.077, $x_{\mathrm{LiPF_6}}$: varied, x_{EC}=0.56, x_{PC}=0.



(c) x_{LiSalt}=0.077, $x_{\mathrm{LiPF_6}}$=0, x_{EC}: varied, x_{PC}=0.



(d) x_{LiSalt}:=0.077, $x_{\mathrm{LiPF_6}}$=0, x_{EC}=56, x_{PC}: varied.



(e) x_{LiSalt}: varied, $x_{\mathrm{LiPF_6}}$=0, x_{EC}=0.56, x_{PC}=0.

Figure S23: Prediction of S_0, S_1 and S_2 for different physical reasonable models (a-d) and non-physical models (e). Similar features are varied and fixed as in Figure 6 in the main article.

Corresponding to Figure 6 in the main part of the article, we can also plot predictions for individual S_i instead of back-transformed ionic conductivity (Figure S23). This provides deeper insights into the models and highlights differences between models in a more detailed way. In general, we observe that the NN and I.M models are similar within uncertainties, whereas the LR opt model deviates mainly for varying x_{PC}. However, for $x_{\mathrm{PC}} < 0.5$ where measurement data is available under the condition of fixed features, the LR opt model also performs similar. It is worth highlighting the predictions of S_2 for varying X_{LiSalt}. Here we observe a narrow window in which all models predict S_2 to be minimal, which is around 0.1, corresponding to the area in which we observe the maximum in σ. Finally, in agreement with that bad predictive capability of log $\frac{\sigma}{x_{\mathrm{LiSalt}}}$ for the RF model all objectives S_i are predicted with large nonphysical fluctuations. These fluctuations are not so pronounced for the A.M model when only showing the prediction of log $\frac{\sigma}{x_{\mathrm{LiSalt}}}$. However, the individual predictions of S_i reveal large fluctuations especially for S_1 and S_2 rendering this model as not usable for predictions.

S13.2 Uncertainties of S_i

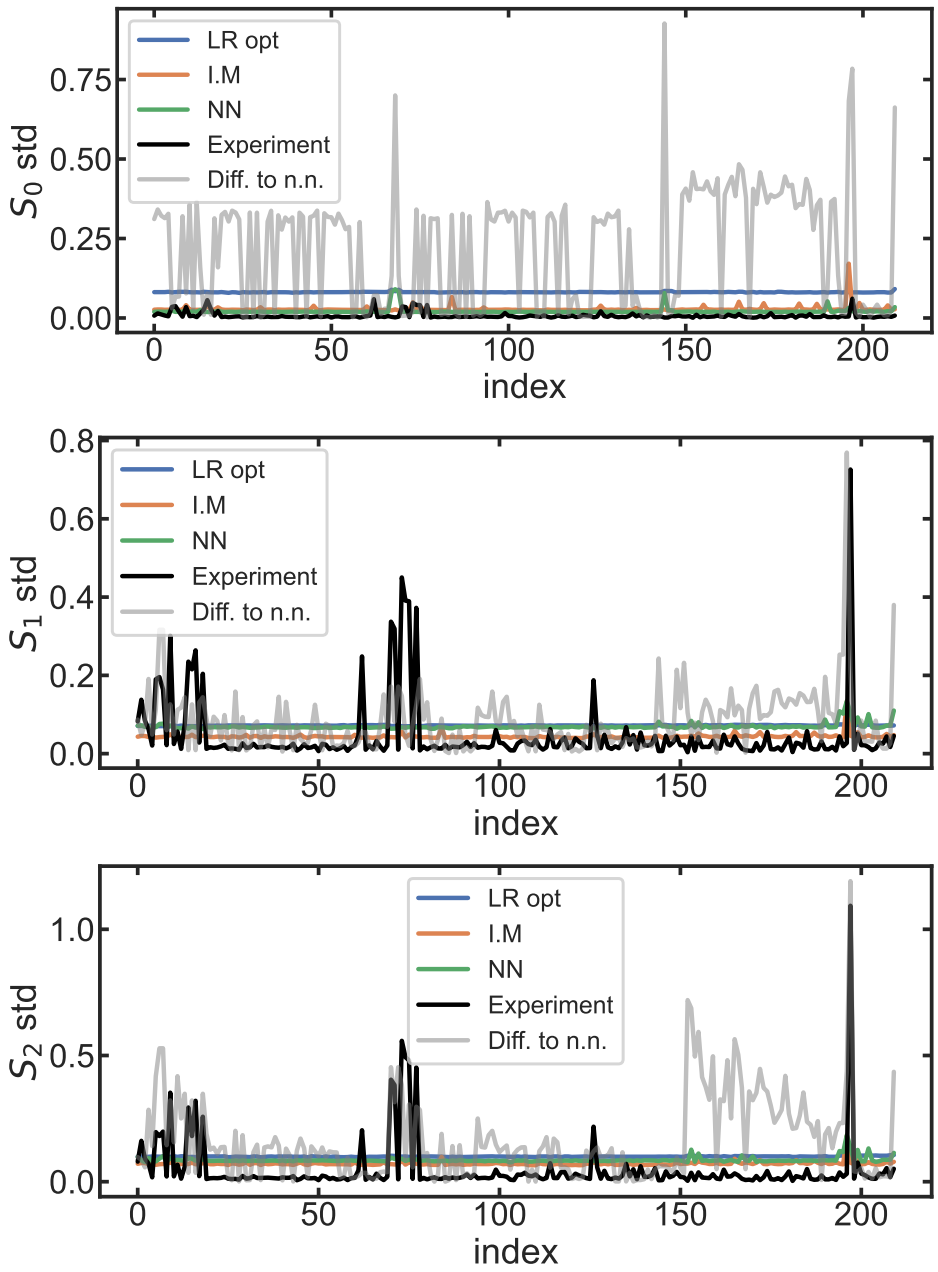


Figure S24: Comparison of standard deviations for experimental Arrhenius fit and uncertainties in the prediction of S_0, S_1 and S_2. The gray line shows the difference between the S_i of the electrolyte composition indicated by index and the nearest experimentally measured neighboring electrolyte composition in feature space, using euclidean distance.

In Figure S24 we compare the predicted uncertainties of the Arrhenius objectives S_i with the experimentally measured standard deviations. We observe that the uncertainties for all models are nearly constant. A key distinction here is the difference in uncertainty estimation method for GPR models (I.M) and models using the MAPIE[3] jackknife+-after-bootstrapping algorithm[22,23] (LR opt, NN). Gaussian Process Regressors inherently can return a model prediction distribution[30]. For the MAPIE models there is a baseline minimum uncertainty defined by the conformity score of the models (this scales with the bootstrapped models' prediction accuracy). This is clearly reflected by the higher baseline uncertainty of the LR opt model for S_0 predictions (see also the higher S_0 MSE for LR

opt compared to NN in Figure S18a). The second component of the MAPIE uncertainty estimation approach used by LECA is the variation in the distribution of the bootstrapped model predictions. The spikes in the uncertainty estimation for the NN model tend correspond to the largest differences in S_i values to neighboring compositions, which makes intuitive sense as these would be the data whose inclusion or exclusion in training the bootstrapped models would lead to the largest deviations in model predictions. None of the methods consistently match the magnitude of the highest experimental uncertainty values, but both GPR and MAPIE show some sensitivity to the especially large outlier uncertainty of measurement 198 (see Table S2). On the other hand, both approaches appear to reasonably approach the correct magnitude for a baseline uncertainty, and the estimations err on the side of slightly overestimating the uncertainty.

S13.3 Difficult to predict electrolyte compositions

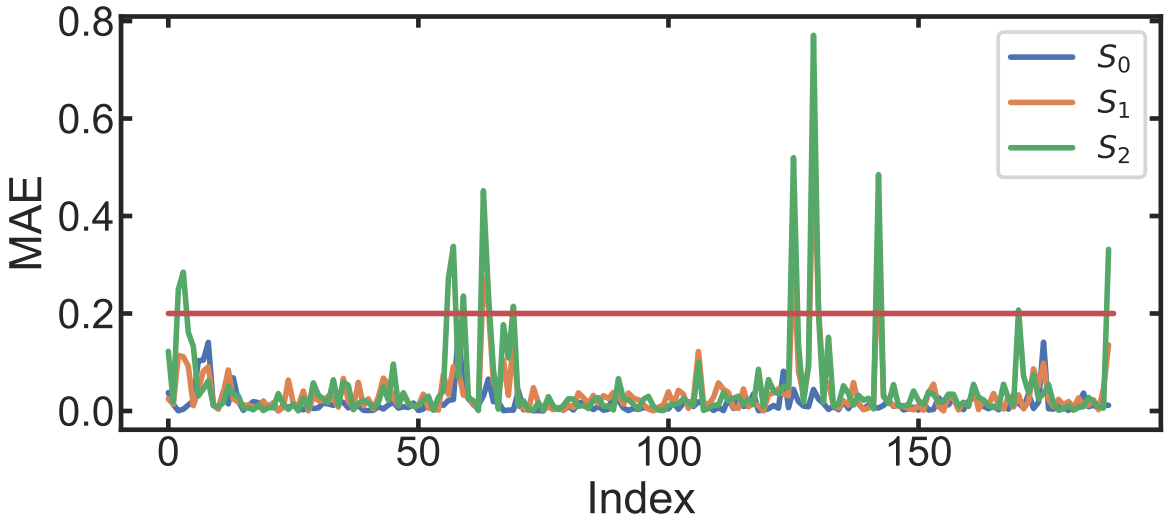


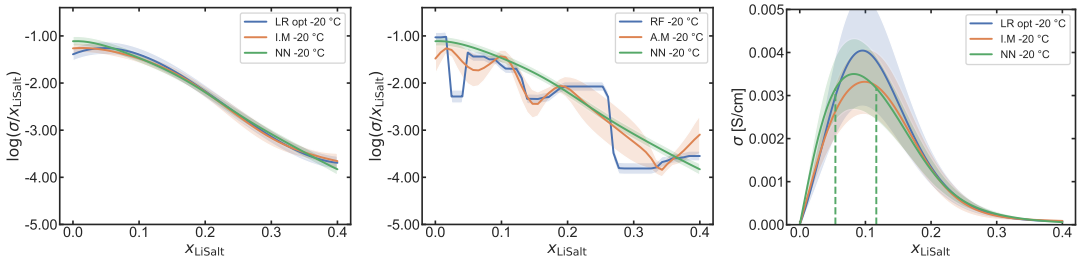
Figure S25: MAE in prediction of S_0, S_1 and S_2 of NN model for leave one out CV. This may help to indicate which data is difficult to fit when every electrolyte formulation except the one with the index is included.

Besides uncertainties, the mean absolute error between prediction and true value of any S_i can be used as an indicator for difficult to fit electrolyte compositions (Figure S25. It is strongly correlated to the analysis above, but is much more simple. To identify difficult to predict electrolyte compositions, we set an arbitrary threshold of 0.2 for predictions of the NN model. If any MAE of the S_i for a specific electrolyte composition exceeds this threshold, we print it out. The largest MAEs are observed for S_2 and smaller MAEs for S_0 and S_1. In total, we find 14 compositions. Nearly for all electrolyte compositions one or more features are close to the minimum or maximum value which was included in the training set. We indicate the number of features with such values as Edge features. They are naturally difficult to fit due to their position at the border or edge in the feature hyperspace.

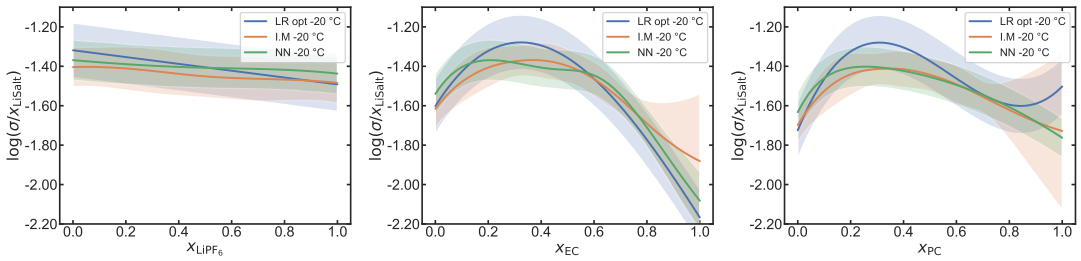
Table S8: Difficult to fit electrolyte formulations in a leave-one-out CV. For each leave out electrolyte formulation, the MAE is given for the prediction of a Neural Network model trained on the data excluding that specific point. The number of edge features counts how many features are either 0 or 1. For x_{LiSalt} every value < 0.025 also counts as an edge feature because there are no measurements with $x_{\mathrm{LiSalt}} = 0$. Also $x_{\mathrm{EC}} > 0.70$ and $x_{\mathrm{PC}} > 0.70$ count as edge features, because there are no measurements for higher values.

	x_{LiSalt}	$x_{\mathrm{LiPF_6}}$	x_{EC}	x_{PC}	MAE S0	MAE S1	MAE S2	Edge features
2	0.016	0.504	0.712	0.064	0.000	0.114	0.249	1
3	0.016	0.110	0.713	0.064	0.004	0.111	0.285	1
56	0.019	1.000	0.048	0.272	0.022	0.034	0.272	2
57	0.019	0.505	0.000	0.304	0.023	0.091	0.338	2
59	0.019	0.000	0.048	0.273	0.046	0.033	0.236	2
63	0.020	0.884	0.000	0.304	0.031	0.273	0.452	2
64	0.020	0.000	0.000	0.304	0.066	0.212	0.241	3
69	0.021	1.000	0.000	0.000	0.002	0.145	0.215	3
125	0.091	0.899	0.202	0.148	0.044	0.318	0.520	0
129	0.092	0.501	0.000	0.505	0.044	0.549	0.771	1
130	0.092	0.000	0.106	0.451	0.025	0.212	0.201	1
142	0.096	0.100	0.000	0.304	0.006	0.333	0.485	1
170	0.171	0.897	0.336	0.000	0.016	0.033	0.207	1
188	0.185	0.499	0.000	0.305	0.012	0.136	0.332	1

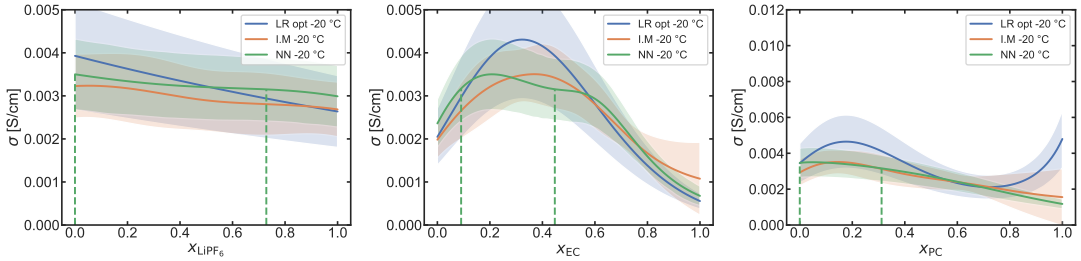
S14 Predicted 1D slices for −20°C and 60°C



(a) x_{LiSalt}: varied, $x_{\text{LiPF}_6} = 0$, $x_{\text{EC}} = 0.20$, $x_{\text{PC}} = 0.04$. (b) x_{LiSalt}: varied, $x_{\text{LiPF}_6} = 0$, $x_{\text{EC}} = 0.20$, $x_{\text{PC}} = 0.04$. (c) x_{LiSalt}: varied, $x_{\text{LiPF}_6} = 0$, $x_{\text{EC}} = 0.20$, $x_{\text{PC}} = 0.04$.

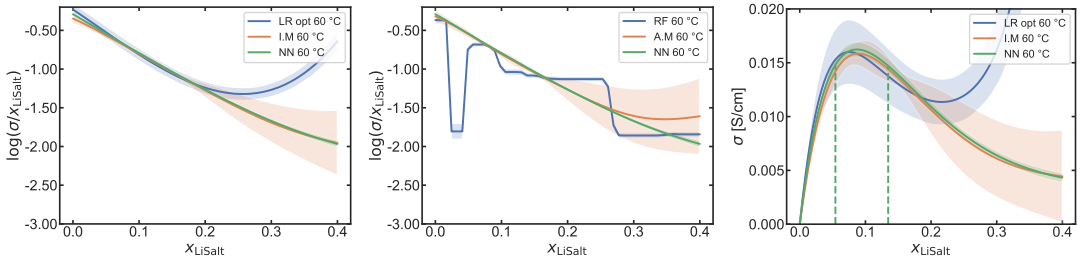


(d) $x_{\text{LiSalt}} = 0.082$, x_{LiPF_6}: varied, $x_{\text{EC}} = 0.20$, $x_{\text{PC}} = 0.04$. (e) $x_{\text{LiSalt}} = 0.082$, $x_{\text{LiPF}_6} = 0$, x_{EC}: varied, $x_{\text{PC}} = 0.04$. (f) $x_{\text{LiSalt}} = 0.082$, $x_{\text{LiPF}_6} = 0$, $x_{\text{EC}} = 0.20$, x_{PC}: varied.



(g) $x_{\text{LiSalt}} = 0.082$, x_{LiPF_6}: varied, $x_{\text{EC}} = 0.20$, $x_{\text{PC}} = 0.04$. (h) $x_{\text{LiSalt}} = 0.082$, $x_{\text{LiPF}_6} = 0$, x_{EC}: varied, $x_{\text{PC}} = 0$. (i) $x_{\text{LiSalt}} = 0.082$, $x_{\text{LiPF}_6} = 0$, $x_{\text{EC}} = 0.20$, x_{PC}: varied.

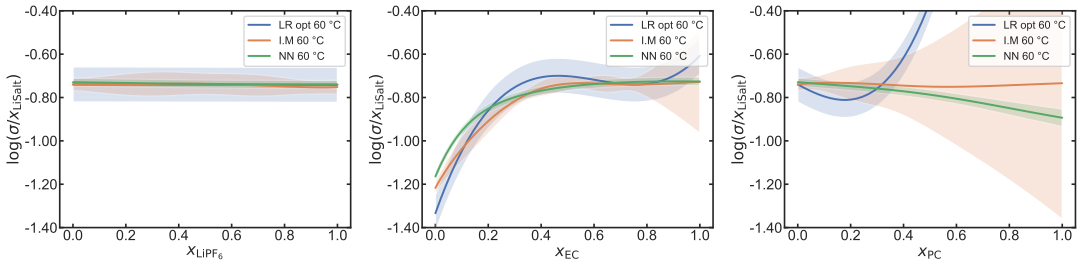
Figure S26: 1D slices corresponding to Figure 6 in the paper at a temperature of −20°C of predictions for physically reasonable LR opt, I.M and NN models (a, c-i) and non-physical RF and A.M models (b). The slices are chosen such that they always include the composition with the maximum ionic conductivity predicted by the NN model at −20°C, except when varying x_{PC}, where we set $x_{\text{PC}} = 0$ to compare both directly for a similar amount of EC. (a,b,d-f) show $\log \frac{\sigma}{x_{\text{LiSalt}}}$ and (c, g-i) shows σ directly. The area between the dashed vertical lines indicates compositions for which the ionic conductivity, predicted by the NN model, is larger than 0.9 times the maximum ionic conductivity. (a-c) show slices in which x_{LiSalt} is varied, x_{LiPF_6} is varied in (d,g), x_{EC} is varied in (e,h) and x_{PC} is varied in (f,i). The fixed features are described in the subplots caption.



(a) x_{LiSalt}: varied, $x_{\mathrm{LiPF_6}} = 0$, $x_{\mathrm{EC}} = 0.73$, $x_{\mathrm{PC}} = 0$.

(b) x_{LiSalt}: varied, $x_{\mathrm{LiPF_6}} = 0$, $x_{\mathrm{EC}} = 0.73$, $x_{\mathrm{PC}} = 0$.

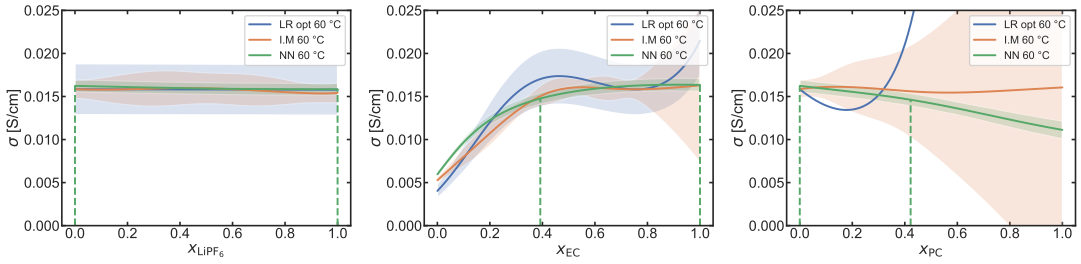
(c) x_{LiSalt}: varied, $x_{\mathrm{LiPF_6}} = 0$, $x_{\mathrm{EC}} = 0.73$, $x_{\mathrm{PC}} = 0$.



(d) $x_{\mathrm{LiSalt}} = 0.087$, $x_{\mathrm{LiPF_6}}$: varied, $x_{\mathrm{EC}} = 0.73$, $x_{\mathrm{PC}} = 0$.

(e) $x_{\mathrm{LiSalt}} = 0.087$, $x_{\mathrm{LiPF_6}} = 0$, x_{EC}: varied, $x_{\mathrm{PC}} = 0$.

(f) $x_{\mathrm{LiSalt}} = 0.087$, $x_{\mathrm{LiPF_6}} = 0$, $x_{\mathrm{EC}} = 0.73$, x_{PC}: varied.



(g) $x_{\mathrm{LiSalt}} = 0.087$, $x_{\mathrm{LiPF_6}}$: varied, $x_{\mathrm{EC}} = 0.73$, $x_{\mathrm{PC}} = 0$.

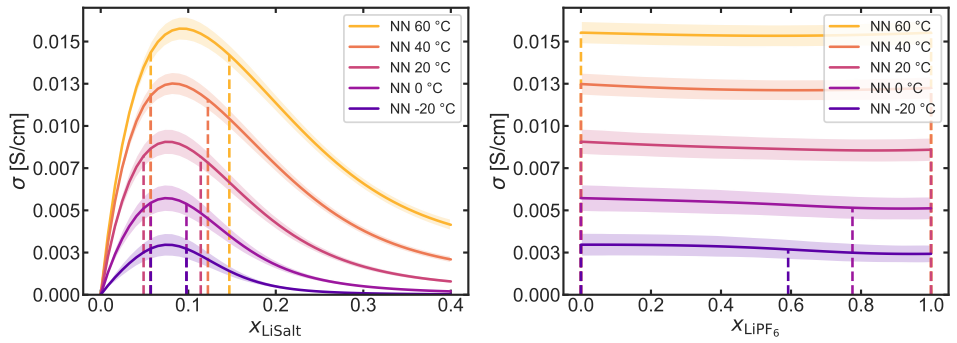
(h) $x_{\mathrm{LiSalt}} = 0.087$, $x_{\mathrm{LiPF_6}} = 0$, x_{EC}: varied, $x_{\mathrm{PC}} = 0$.

(i) $x_{\mathrm{LiSalt}} = 0.087$, $x_{\mathrm{LiPF_6}} = 0$, $x_{\mathrm{EC}} = 0.73$, x_{PC}: varied.

Figure S27: 1D slices corresponding to Figure 6 in the paper at a temperature of 60°C of predictions for physically reasonable LR opt, I.M, NN models (a, c-i) and non-physical RF and A.M models (b). The slices are chosen such that they always include the composition with the maximum ionic conductivity predicted by the NN model at 60°C, except when varying x_{PC}, where we set $x_{\mathrm{PC}} = 0$ to compare both directly for a similar amount of EC. (a,b,d-f) show $\log \frac{\sigma}{x_{\mathrm{LiSalt}}}$ and (c, g-i) shows σ directly. The area between the dashed vertical lines indicates compositions for which the ionic conductivity, predicted by the NN model, is larger than 0.9 times the maximum ionic conductivity. (a-c) show slices in which x_{LiSalt} is varied, $x_{\mathrm{LiPF_6}}$ is varied in (d,g), x_{EC} is varied in (e,h) and x_{PC} is varied in (f,i). The fixed features are described in the subplots caption.

S15 Predictions for different temperatures

Here, we show 1D slices (Figure S28) where we vary one feature and keep the other features constant at the optimum composition predicted by the NN model for 20°C. Then we vary the temperature to observe the effects of varying the temperature for one specific electrolyte composition. In general, the ionic conductivity σ increases, when increasing the temperature. When varying either $x_{\mathrm{LiPF_6}}$ or x_{PC} under this conditions nothing interesting happens. However, when increasing x_{LiSalt} the maximum shifts towards larger values and the area with an ionic conductivity σ larger than 90% of the maximum broadens. Similar happens when increasing x_{EC}. The optimum shifts to larger values with a strong broadening of the optimal area around the maximum. However, we have to clarify here once again that we do not show the optimal composition for larger temperatures, but the prediction at 60°C of the optimal composition at 20°C.



(a) x_{LiSalt}: varied, $x_{\mathrm{LiPF_6}}$=0, x_{EC}=0.56, x_{PC}=0.

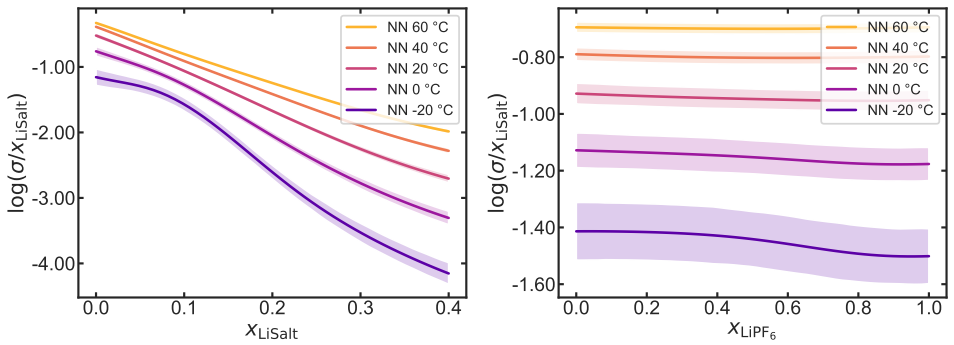
(b) x_{LiSalt}=0.077, $x_{\mathrm{LiPF_6}}$: varied, x_{EC}=0.56, x_{PC}=0.

(c) x_{LiSalt}=0.077, $x_{\mathrm{LiPF_6}}$=0, x_{EC}: varied, x_{PC}=0.

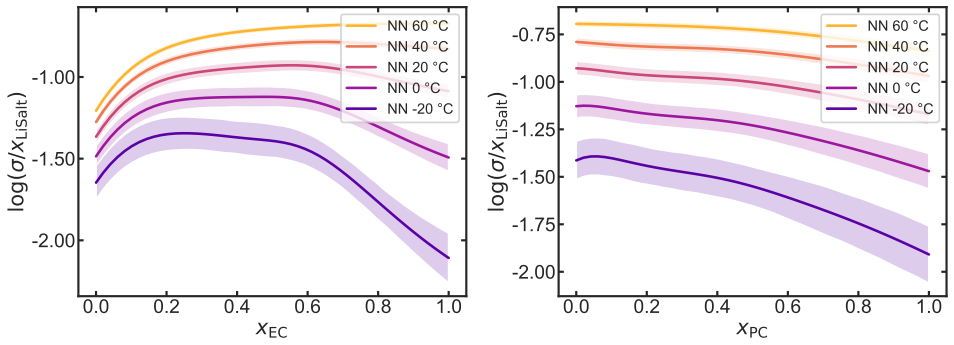
(d) x_{LiSalt}:=0.077, $x_{\mathrm{LiPF_6}}$=0, x_{EC}=56, x_{PC}: varied.

Figure S28: Neural Network model predictions for different electrolyte formulations analog to Figure 6 in the main article and different temperatures.

To highlight the pyhsical insight gained from Figure 8 and Figure S28 we plot Figure S28 also in logarithmic scale (Figure S29). Especially the dependence of the logarithmic ionic conductivity on x_{EC} (Figure S29c) shows interesting behavior. From Figure 8b, we can learn that the activation energy for small amounts of lithium salt is nearly independent of x_{EC}. This is reflected by a similar gradient for low amounts of EC ($x_{\mathrm{EC}} < 0.2$) in Figure S29c for all temperatures. Particularly for 40°C and 60°C the gradient is similar for all amounts of EC, whereas we observe large deviations for lower temperatures from this behavior. This can be explained by the fact that the deviations from Arrhenius behavior significantly depend on x_{EC} and increase with the EC content (as can be learned from Figure 8c). With increasing temperature difference between onset temperature and prediction temperature, these deviations become more important compared to the activation energy and govern the overall behavior of the ionic conductivity when varying the EC content.



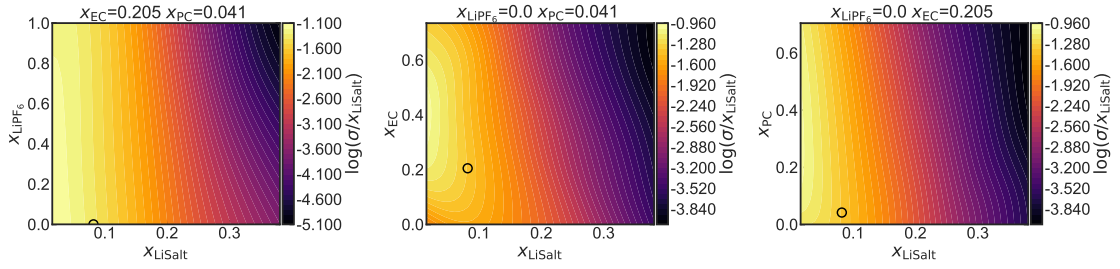
(a) x_{LiSalt}: varied, $x_{\mathrm{LiPF_6}}$=0, x_{EC}=0.56, x_{PC}=0. (b) x_{LiSalt}=0.077, $x_{\mathrm{LiPF_6}}$: varied, x_{EC}=0.56, x_{PC}=0.



(c) x_{LiSalt}=0.077, $x_{\mathrm{LiPF_6}}$=0, x_{EC}: varied, x_{PC}=0. (d) x_{LiSalt}:=0.077, $x_{\mathrm{LiPF_6}}$=0, x_{EC}=56, x_{PC}: varied.

Figure S29: Neural Network model predictions for different electrolyte formulations analog to Figure S28 in logarithmic scale.

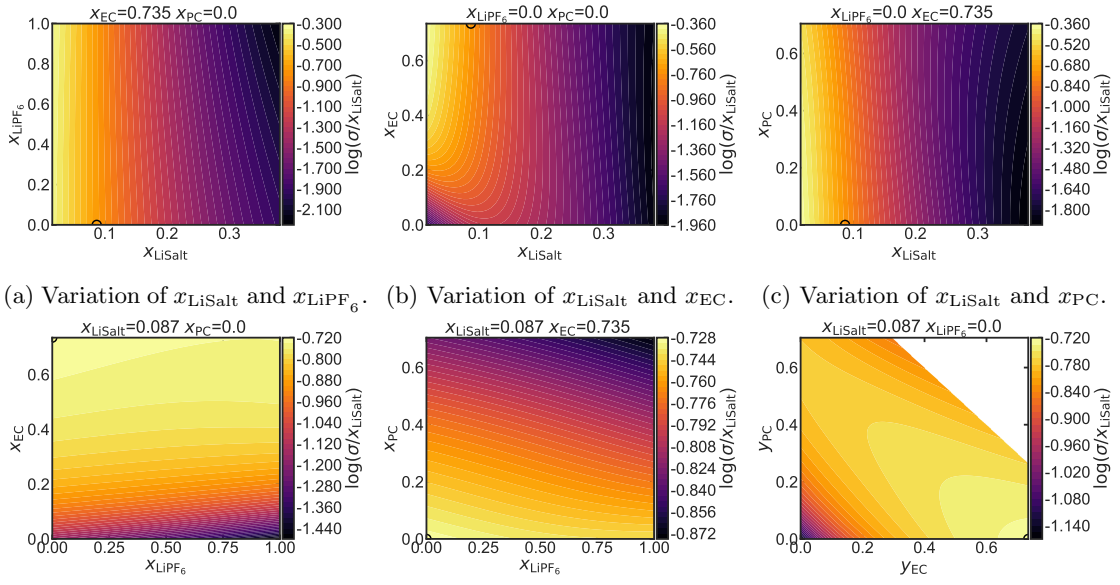
S16 Predicted 2D slices for −20°C and 60°C



(a) Variation of x_{LiSalt} and $x_{\mathrm{LiPF_6}}$. (b) Variation of x_{LiSalt} and x_{EC}. (c) Variation of x_{LiSalt} and x_{PC}.

(d) Variation of $x_{\mathrm{LiPF_6}}$ and x_{EC}. (e) Variation of $x_{\mathrm{LiPF_6}}$ and x_{PC}. (f) Variation of x_{EC} and x_{PC}.

Figure S30: 2D slices for the predictions of the NN model at −20°C corresponding to Figure 7 in the main article. Two features are fixed and two features are varied for each subfigure. The fixed features are chosen in a way that the composition with the global maximum ionic conductivity σ at −20°C is included in each slice (indicated by a black circle. The circle does not correspond to the maximum necessarily, because we show $\log \frac{\sigma}{x_{\mathrm{LiSalt}}}$).

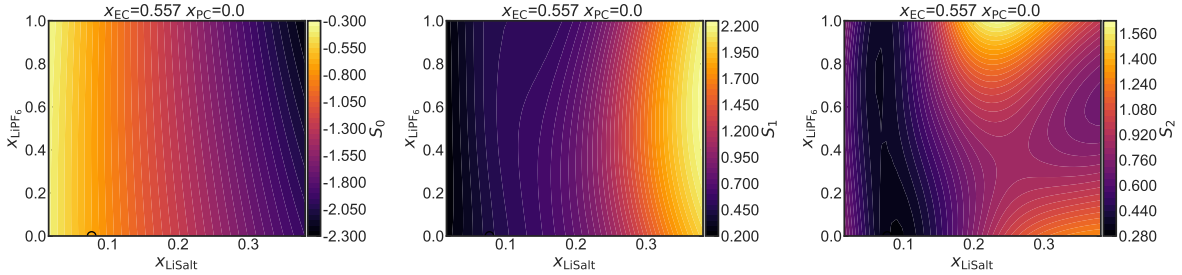


(a) Variation of x_{LiSalt} and $x_{\mathrm{LiPF_6}}$. (b) Variation of x_{LiSalt} and x_{EC}. (c) Variation of x_{LiSalt} and x_{PC}.

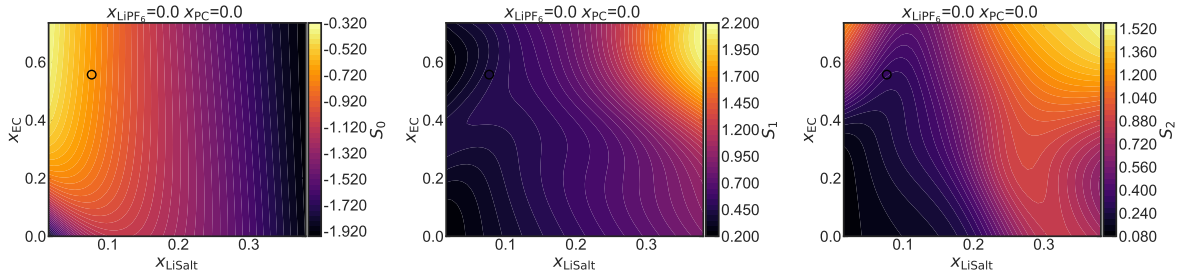
(d) Variation of $x_{\mathrm{LiPF_6}}$ and x_{EC}. (e) Variation of $x_{\mathrm{LiPF_6}}$ and x_{PC}. (f) Variation of x_{EC} and x_{PC}.

Figure S31: 2D slices for the predictions of the NN model at 60°C corresponding to Figure 7 in the paper. Two features are fixed and two features are varied for each subfigure. The fixed features are chosen in a way that the composition with the global maximum ionic conductivity σ at 20°C is included in each slice (indicated by a black circle. The circle does not correspond to the maximum necessarily, because we show $\log \frac{\sigma}{x_{\mathrm{LiSalt}}}$).

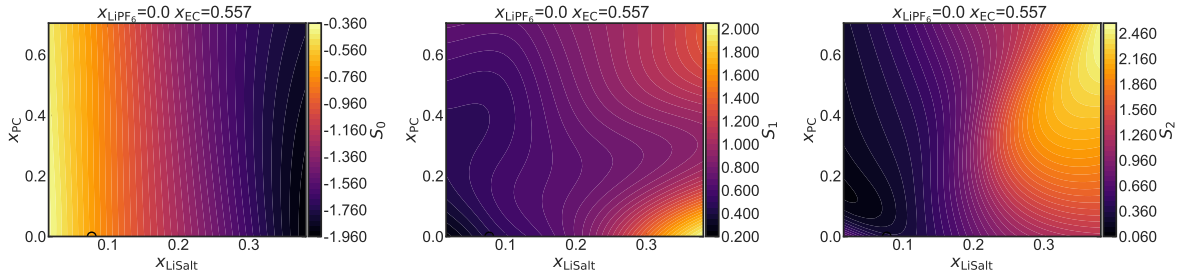
S17 2D slices of Arrhenius coefficient predictions



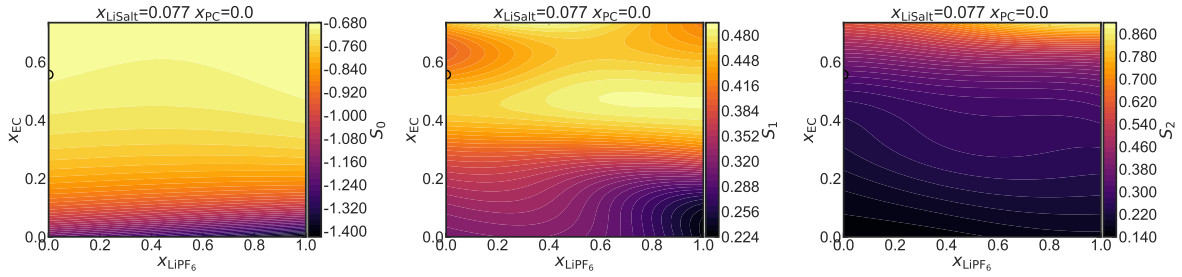
(a) x_{LiSalt} and $x_{\mathrm{LiPF_6}}$ varied, x_{EC}=0.56, x_{PC}=0.



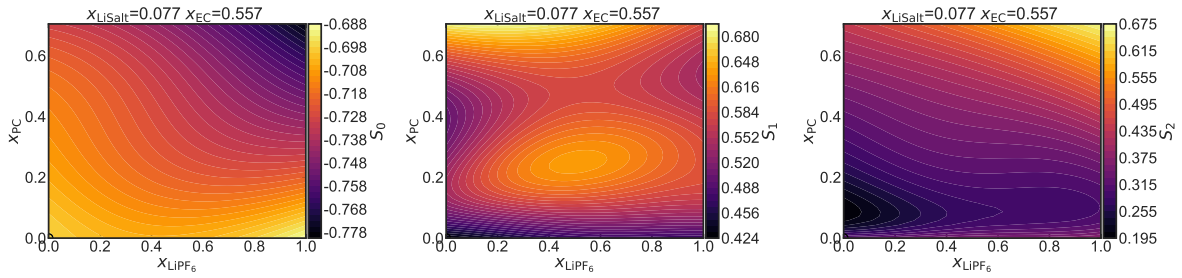
(b) x_{LiSalt} and x_{EC} varied, $x_{\mathrm{LiPF_6}}$=0, x_{PC}=0.



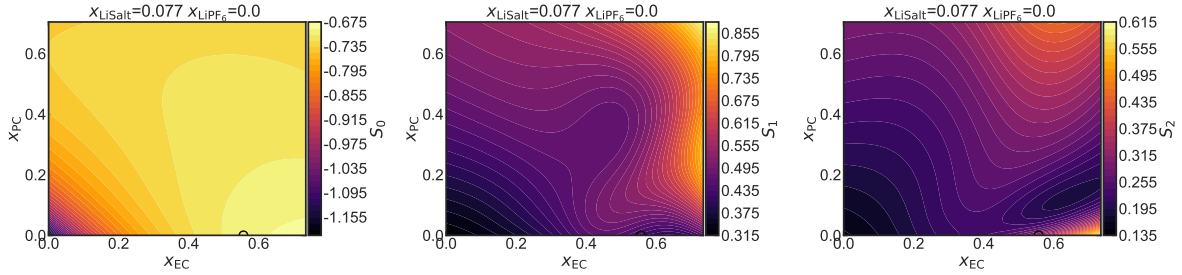
(c) x_{LiSalt} and x_{PC} varied, $x_{\mathrm{LiPF_6}}$=0, x_{EC}=0.56.



(d) x_{LiSalt}=0.077, $x_{\mathrm{LiPF_6}}$=0 and x_{EC} varied, x_{PC}=0.



(e) x_{LiSalt}=0.077, $x_{\mathrm{LiPF_6}}$ and , x_{PC} varied, x_{EC}=0.56.



(f) x_{LiSalt}=0.077, $x_{\mathrm{LiPF_6}}$=0, x_{EC} and x_{PC} varied.

Figure S32: 2D slices of Arrhenius coefficient predictions for the NN model with an onset temperature of 60°C. All 2D slices bisect the predicted optimal composition at 20°C analog to Figure 7 in the main article and span the training feature space.

S18 Further Data available

The LECA code is available under https://github.com/Harrison-Teeg/LECA/ and its documentation under https://leca.readthedocs.io. Jupyter-notebooks to create the graphics in the paper and ESI will be made available after publication. The complete dataset is available.

References

[1] A. Narayanan Krishnamoorthy, C. Wölke, D. Diddens, M. Maiti, Y. Mabrouk, P. Yan, M. Grünebaum, M. Winter, A. Heuer and I. Cekic-Laskovic, *Chemistry–Methods*, 2022, **2**, e202200008.

[2] The GPyOpt authors, *GPyOpt: A Bayesian Optimization framework in python*, `http://github.com/SheffieldML/GPyOpt`, 2016.

[3] *MAPIE - Model Agnostic Prediction Interval Estimator*, `https://mapie.readthedocs.io/en/latest/index.html`, Version: 0.4.1.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, *Journal of Machine Learning Research*, 2011, **12**, 2825–2830.

[5] B. E. Granger and F. Pérez, *Computing in Science & Engineering*, 2021, **23**, 7–14.

[6] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke and T. E. Oliphant, *Nature*, 2020, **585**, 357–362.

[7] J. D. Hunter, *Computing in Science & Engineering*, 2007, **9**, 90–95.

[8] The pandas development team, *pandas-dev/pandas: Pandas*, 2020, `https://doi.org/10.5281/zenodo.6408044`.

[9] W. McKinney, Proceedings of the 9th Python in Science Conference, 2010, pp. 56 – 61.

[10] 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, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt and SciPy 1.0 Contributors, *Nature Methods*, 2020, **17**, 261–272.

[11] E. O. Lebigot, *Uncertainties: a Python package for calculations with uncertainties*, `http://pythonhosted.org/uncertainties/`.

[12] L. McInnes, J. Healy and S. Astels, *The Journal of Open Source Software*, 2017, **2**, 1–2.

[13] M. L. Waskom, *Journal of Open Source Software*, 2021, **6**, 3021.

[14] G. Brandl, *URL http://sphinx-doc.org/sphinx.pdf*, 2021.

[15] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning*, Springer-Verlag, 2009.

[16] L. Breiman, *Machine Learning*, 2004, **45**, 5–32.

[17] C. E. Rasmussen, *Gaussian processes for machine learning*, MIT Press, Cambridge, Mass., 2006.

[18] E. Schulz, M. Speekenbrink and A. Krause, *Journal of Mathematical Psychology*, 2018, **85**, 1–16.

[19] V. L. Deringer, A. P. Bartók, N. Bernstein, D. M. Wilkins, M. Ceriotti and G. Csányi, *Chemical Reviews*, 2021, **121**, 10073–10141.

[20] M. Kirk, *Thoughtful Machine Learning: A Test-Driven Approach*, O'Reilly & Associates, Sebastopol, 1st edn, 2014.

[21] T. Lookman, P. V. Balachandran, D. Xue and R. Yuan, *npj Computational Materials*, 2019, **5**, 1–17.

[22] R. F. Barber, E. J. Candès, A. Ramdas and R. J. Tibshirani, *The Annals of Statistics*, 2021, **49**, 486 – 507.

[23] B. Kim, C. Xu and R. F. Barber, *Predictive Inference Is Free with the Jackknife+-after-Bootstrap*, 2020, `https://arxiv.org/abs/2002.09025`.

[24] M. G. Genton, *J. Mach. Learn. Res.*, 2002, **2**, 299–312.

[25] B. Minasny and A. B. McBratney, *Geoderma*, 2005, **128**, 192–207.

[26] K. He, X. Zhang, S. Ren and J. Sun, *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*.

[27] R. H. Byrd, J. Nocedal and R. B. Schnabel, *Mathematical Programming*, 1994, **63**, 129–156.

[28] X. Glorot and Y. Bengio, Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Chia Laguna Resort, Sardinia, Italy, 2010, pp. 249–256.

[29] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*.

[30] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2005.