Example 1: Echo State Network for Triple-Cation PL Output

Here we explain the contents of this notebook for a non-coding audience, breaking down each part of the process according to our 5-step machine learning roadmap (described in the main text of the paper).

(1) Identify the materials question of interest

The question here is whether we can predict the absolute PL from a triple-cation perovskite thin film based on the ambient humidity level (holding the ambient temperature constant). These predictions would allow us to estimate how well the material would perform in high vs low humidity environments.

(2) Obtain sufficient data for model training

The data for this project was collected using a custom-built environmental sample chamber. PL data was collected in a confocal microscope over three experiments, each lasting 4 hours. This is a small amount of data for complex ML algorithms like neural networks, so we augment the raw data during the pre-processing step. Note that, in practice, more experimental data is desired for ML implementations.

(3) Pre-process the data

Information about the raw data:

- In total: 720 time-series data points, where each point contains a humidity level and a PL value. Data is in the form of five 4-hour runs of humidity cycling on samples with identical compositions. Each run contains three humidity cycles and 240 data points.

The pre-processing steps are as follows:

- Create a pandas dataframe and load the data
- Clean the data files (remove gaps, cut to equal duration) and resample such that data points are 15 seconds apart
- Scale the values for relative humidity (rH) and PL from -1 to 1 (since ML algorithms are sensitive to the relative scale of the features)
- Augment the data using a linear interpolation method. Essentially, this creates several "intermediate" rH-PL runs that are weighted linear combinations of the experimental runs. This routine thus does not introduce any data points outside of the bounds of the raw data. After augmentation, the data set consists of 14400 time-series data points.
- Randomly stitch together the runs, one after another, to create a single (much longer) time-series. This final data set contains over 60 hours of data, and is graphed in the "Data pre-processing" section of this notebook.

(4) Apply feature engineering as needed

We create additional features for model training:

- Humidity gradient (captures how quickly the humidity is rising or falling)
- Cumulative moisture exposure

(5) Optimize and test the model

The model is constructed using the pyESN implementation of the ESN algorithm (https://github.com/cknd/pyESN). The data is split into 80% training and 20% validation sets. Hyperparameters are optimized using a grid search approach, which systematically steps through different hyperparameter values to locate the combination with the lowest validation error. Hyperparameters of interest are:

- random noise applied to each neuron in the network, serves as a regularization term to mitigate overfitting
- spectral radius of the recurrent weight matrix which links neurons together

Below are instructive graphics (also shown in the main text of the paper) which demonstrate how ESN predictive performance changes based on the amount of training data supplied and the selected hyperparameters. Validation set NRMSE values range from 29.2% (worst case) to 14.3% (best case) depending on hyperparameter selection.

Example 2: Long Short-Term Memory for MAPI Devices

Here we explain the contents of this notebook for a non-coding audience, breaking down each part of the process according to our 5-step machine learning roadmap (described in the main text of the paper).

(1) Identify the materials question of interest

The question here is how to forecast long-term degradation in MAPI solar cells under environmental stress (in this case, temperature).

(2) Obtain sufficient data for model training

We use data from the publication: Holzhey, P., et al. A chain is as strong as its weakest link – Stability study of MAPbI3 under light and temperature. Materials Today 29, 10-19, doi:10.1016/j.mattod.2018.10.017 (2019). The authors track solar cell power output over a long time frame (500 hours) for cells held at five different temperatures (-10, 20, 50, 65, and 95C).

(3) Pre-process the data

Information about the raw data:

- In total: five 500-hour runs on MAPI solar cell samples, with the temperature held constant throughout each run. Each time-series data point contains the time (in hours) and device power.

The pre-processing steps are as follows:

- Create a pandas dataframe and load the data
- Clean the data files (cut to equal duration) and resample such that data points are 60 seconds apart
- Normalize the power data
- Hold out one entire experimental data file (at 50C) and set aside for testing
- Augment the data using a linear interpolation method. Essentially, this creates several "intermediate" temperature-power runs that are weighted linear combinations of the experimental

runs. This routine thus does not introduce any data points outside of the bounds of the raw data, and does not include the held out data file. After augmentation, the data set consists of 11000 time-series data points.

- Randomly stitch together the runs, one after another, to create a single (much longer) time-series.
- Split the stitched-together time series into 70% training and 30% validation sets

(4) Apply feature engineering as needed

We do not use feature engineering in this case.

(5) Optimize and test the model

A long short-term memory (LSTM) model is constructed constructed and trained using the keras deep learning framework. The network architecture consists of:

- Five input units
- A leaky rectified linear unit (ReLU) as the activation function
- A dropout layer to mitigate overfitting
- A dense layer to output the prediction

Interested readers can find more information about each of these layers at https://www.tensorflow.org/api_docs/python/tf/keras/layers

The LSTM is trained over the course of 200 epochs, where each epoch invovles stepping through all of the training and validation data. Finally, the model is evaluated on the unseen test set: the experimental run at 50C that was held our prior to data augmentation and training. The LSTM predicts the entire 500 hour time-series with an NRMSE of only 5.5%.

Example 3: Convolutional Neural Network for HTL conductance

Here we explain the contents of this notebook for a non-coding audience, breaking down each part of the process according to our 5-step machine learning roadmap (described in the main text of the paper).

(1) Identify the materials question of interest

The question here is whether we can predict the conductance value for PSC hole-transport layers (Spiro-OMeTAD) based on optical images alone. Such a prediction links optical and electronic properties, and can save time by reducing the number of characterization steps necessary to evaluate each thin film.

(2) Obtain sufficient data for model training

We use data from the publication: MacLeod, B. P., et al. Self-driving laboratory for accelerated discovery of thin-film materials. Sci Adv 6 (2020). The authors use high-throughput robotic synthesis and characterization to quickly acquire large amounts of data.

(3) Pre-process the data

Information about the raw data:

- In total: 2x35=70 samples
- For each sample, there are 7 different I-V data for conductance measurement taken at different location (70x7=490)
- For each sample, there are 3 images for spin-coated Spiro-OMeTAD layer after annealing (70x3=210), taken from different areas of the sample. All the original image files are of the size (4000, 3000) for width and height, respectively

The pre-processing steps are as follows:

- For each sample, average the 7 different conductance data for labeling
- Duplicate each conductance data point twice, so they align with the image data
- Normalize the conductance data
- Convert the PIL image instances into numpy array
- Split the data into train and test sets

(4) Apply feature engineering as needed

We do not use explicit feature engineering, but we do augment the data through random rotations, width and height shifting, flipping, and zooming of the raw data images. This is a very common practice for machine vision tasks, where it is essential to provide the model with representative training data (ideally this will encompass all types of images the model may encounter during later testing and usage.

(5) Optimize and test the model

The convolutional neural network (CNN) is constructed and trained using the keras deep learning framework. The network architecture consists of:

- Three 2D convolution layers (3x3x32, 3x3x64, 3x3x64), each followed by a max pooling layer (2x2)
- Regularization for each convolutional layer to mitigate overfitting
- Two fully connected layers (100 and 64 in size)
- Single-value output layer that predicts conductance for each image

Interested readers can find more information about each of these layers at https://www.tensorflow.org/api_docs/python/tf/keras/layers.

The model is trained on 64% of the data. 16% of the data serves as the validation set. Finally, the CNN is evaluated on the remaining 20% of the data (test set). Our final CNN is able to predict conductance values with an NRMSE of 21% based on dark field images alone, an impossible task for human visual analysis. The CNN_data_analysis notebook contains further visualization of the model results.

Example 4: Echo State Network for MAPI Devices

(Supplemental example, not in main text)

Here we explain the contents of this notebook for a non-coding audience, breaking down each part of the process according to our 5-step machine learning roadmap (described in the main text of the paper).

(1) Identify the materials question of interest

The question here is whether we can predict the power output from a MAPI solar cell device based on the incident light intensity and ambient temperature. These predictions would allow us to estimate how well the device would perform in various locations and environments.

(2) Obtain sufficient data for model training

We use data from the publication: Tress, W., et al. Performance of perovskite solar cells under simulated temperature-illumination real-world operating conditions. Nature Energy 4, 568-574, doi:10.1038/s41560-019-0400-8 (2019). The authors collect solar cell data over a long time frame (500+ hours), which provides sufficient data to train a simple recurrent neural network, in this case an Echo State Network (ESN).

(3) Pre-process the data

Information about the raw data:

- In total: 106920 data points, taken approximately 10 seconds apart

The pre-processing steps are as follows:

- Create a pandas dataframe and load the data
- Extract three variables of interest from the data, which serve as the model inputs (intensity, temperature) and output (power)
- Scale the values for each variable from -1 to 1 (since ML algorithms are sensitive to the relative scale of the features)
- Split the data into train and test sets

(4) Apply feature engineering as needed

We do not use feature engineering in this case.

(5) Optimize and test the model

The model is constructed using the pyESN implementation of the ESN algorithm (https://github.com/cknd/pyESN). The hyperparameters are set to typical values and not tuned in this case. Therefore, we do not include a validation set and instead split the data in 20% training and 80% testing sets. We can use a small training set for this project since there is a known physical correlation between light intensity and power output for solar cells. This is a simple example of how physics domain knowledge can be incorporated into ML.

The ESN predicts the solar cell power from intensity and temperature with an NRMSE of 3.1%.