Supporting Information for: Convolutional Neural Networks for High Throughput Screening of Catalyst Layer Inks for Polymer Electrolyte Fuel Cells

Mohammad J. Eslamibidgoli[a,*], Fabian P. Tipp[b], Jenia Jitsev[c], Jasna Jankovic[d], Michael H. Eikerling[a,e], Kourosh Malek[a,*]

[a] *Institute of Energy and Climate Research, Theory and Computation of Energy Materials (IEK-13), Forschungszentrum Jülich, 52425 Jülich, Germany*
[b] *Department of Chemistry, University of Cologne, Greinstr. 4-6, 50939 Cologne, Germany*
[c] *Jülich Supercomputing Center, Forschungszentrum Jülich, 52425 Jülich, Germany*
[d] *Department of Materials Science and Engineering, University of Connecticut, 97 North Eagleville Road, Unit 3136, Storrs, CT 06269-3136, USA*
[e] *Chair of Theory and Computation of Energy Materials, Faculty of Georesources and Materials Engineering, RWTH Aachen University, Aachen 52062, Germany*

1. Data preprocessing

Six catalyst ink imaging sets were considered in this work. Three ink powders were studied with Nafion ionomer, EA50 I33 ink, F50 I33 ink and V50 I33 ink (EA - graphitized, F - stabilized and V - Vulcan carbon support with Pt from TKK, and 33 wt% Nafion ionomer), and three with Aquivion ionomer, EA50 I33 ink, F50E I33 ink and V50 I33 ink. Catalyst powders Tec10EA50E, Tec10F50E and Tec10V50E, containing ≈ 50 wt% platinum on carbon were received from Tanaka Kikinzoku Kogyo (TKK) (further denoted as EA50, V50 and F50). The catalyst inks with 33 wt% ionomer were prepared by sonication of the catalyst in 50 vol% n-Propanol in water for 15 minutes before deposition of two drops onto a lacey carbon 300 mesh formvar coated TEM grid (TedPella). All samples were imaged using a FEI Tecnai G2 200 kV Transmission Electron Microscope. For the samples with Nafion, fifteen images were taken for each sample at 22,000x magnification. For the Aquivion class there were 3 images for V50, 2 images for

*Corresponding author
Email addresses: `m.eslamibidgoli@fz-juelich.de` (Mohammad J. Eslamibidgoli), `k.malek@fz-juelich.de` (Kourosh Malek)

F50E and 1 image for EA50 at 13,000 magnification. The images belonging to the Nafion ionomer have a resolution of 2048 × 2115 pixels with the scale bar indicating 500 nm, and those belonging to Aquivion have a resolution of 1024 × 1059 with the scale bar of 200 nm. As there is also a black bar with the scaling information at the bottom of each image, the images were cropped to remove it. This resulted in a resolution of 1024 × 1024 for the Aquivion images and 2048 × 2048 for the Nafion images. The Nafion images were then resized to 1024 × 1024 in order to have a uniform image dataset for further processing. To decrease the required memory resources for running the region proposal algorithm (i.e. Selective Search) at a later step, each high-resolution image (1024 × 1024) was split up into four lower resolution images with a resolution of 512 × 512.

At this point, the dataset were split into approximately 75%-25% for the training and the validation purposes. Hereby, it was ensured that there is no information leakage between the training set and the hold out set. For example, for the Nafion class 12 out of 15 images were used for training and 3 for the test purpose. The only exception was the Aquivion-EA50 class as there was only one original image available. Therefore, one of the four 512 × 512-pixel images that resulted from the same high-resolution image of the Aquivion-EA50 class was placed into the hold out set, while the other three were placed into the training set.

Thereafter, selective search algorithm was applied as explained in the article. Table 1 summarizes the distribution of the images in the classes after application of the selective search algorithm.

2. Model Training

The preprocessed images were loaded into memory through the keras flow_from_directory function and were preprocessed with the keras ImageDataGenerator. The images of the training and validation sets were rescaled by dividing each pixel's value by 255. The mean of each input image was also set to zero through the samplewise center option and each image was divided by its standard deviation through the option samplewiese std normalization.

Table 1: Distribution of the images in the classes after application of the selective search algorithm.

Class	Training Set	Hold out Set
Aquivion-EA50	1979	1869
Aquivion-F50E	1885	1890
Aquivion-V50	1847	1921
Nafion-EA50	1837	1863
Nafion-F50	1874	1882
Nafion-V50	1885	1607

For the images of the training set, data augmentation was also performed by randomly flipping the images along the vertical and/or the horizontal for each epoch. For the custom CNN, the image size was set to 100×100 and the colormode was set to grayscale for all images. For the transfer learning with the first layers of VGG16, the image size was also set to 100×100 but the colormode was set to rgb. The colormode was chosen to be rgb to make the images compatible with the convolutional layers of VGG16.

For the approach using logistic regression, the image size was set to 224×224 and the colormode was set to rgb. The image size was changed to be able to directly use the input layer of VGG16. The custom CNN was created as a sequential model in the keras API. For training, the categorical crossentropy loss function and the Adam optimizer was used.

To ensure the overlapping samples do not affect our conclusion about the importance of applying zoom-in and zoom-out data augmentation, we created another smaller dataset with non-overlapping 100×100 patches using the SW approach. This gives us about 50 training examples in each class. As shown in Fig. 1, training the custom network on this small dataset and validating on the SS dataset also results in poor model performance; however, training on SS and validating on the non-overlapping images further confirms the hypothesis that zoom-in and zoom-out data augmentation must be accounted for in the training phase.

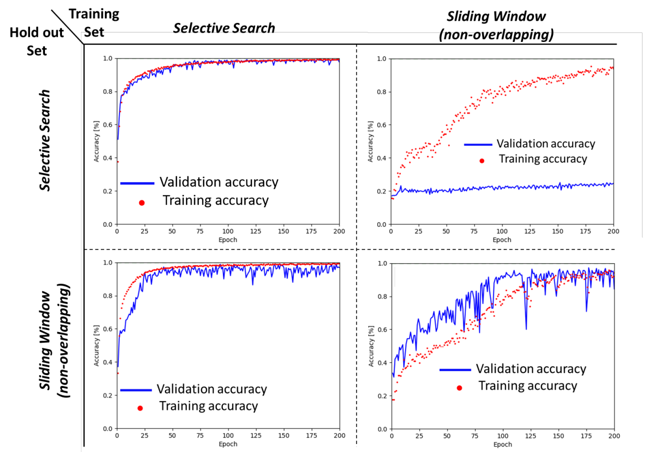

Figure 1: Comparison of the learning curves when the training and hold out sets are Selective Search and non-overlapping Sliding Window.

Models were trained on the computational resources at Jülich Supercomputing Centre.[S1] TensorFlow version 2.3.1 was used for the backend through version 2.4.3 of Keras.[S2] The logistic regression model was constructed through version 0.23.2 of scikit-learn.[S3]

3. References

[S1] Jülich Supercomputing Centre. (2019). JUWELS, Modular Tier-0/1 Supercomputer at the Jülich Supercomputing Centre. Journal of large-scale research facilities, 5, A135. doi: 10.17815/jlsrf-5-171

[S2] Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J. et al. Tensorflow: A system for large-scale machine learning. In 12th USENIX symposium on operating systems design and implementation, 2016, OSDI 16, 265-283.

[S3] Pedregosa, F. ; Varoquaux, G. ; Gramfort, A. ; Michel, V. ; et al. "Scikit-learn: Machine learning in Python." the Journal of machine Learning research, 2011, 12, 2825-2830.