Supporting Information for:
A Neural Network to Decipher Organic Electrochemical Transistors' Multivariate Responses for Cation Recognition

Sébastien Pecqueur[1]*, Dominique Vuillaume[1], Željko Crljen[2], Ivor Lončarić[2], Vinko Zlatić[2]*

[1] Univ. Lille, CNRS, Centrale Lille, Univ. Polytechnique Hauts-de-France, UMR 8520 - IEMN, F-59000 Lille, France
[2] Ruđer Bošković Institute, Bijenička cesta 54, 10000 Zagreb, Croatia
* E-mail: sebastien.pecqueur@*iemn*.fr; vinko.zlatic@*irb*.hr

Class	KCl (0.1 M)	NaCl (0.1 M)	$CaCl_2$ (0.1 M)	Precision	Sensitivity
1	0	0	1	0.49	0.62
2	0	0.1	0.9	0.54	0.67
3	0	0.01	0.99	0.41	0.30
4	0	0.5	0.5	0.66	0.41
5	0	0.9	0.1	0.52	0.61
6	0	0.99	0.01	0.41	0.38
7	0	1	0	0.50	0.48
8	0.1	0	0.9	0.51	0.41
9	0.01	0	0.99	1	1
10	0.1	0.1	0.8	0.43	0.48
11	0.01	0.01	0.98	0.42	0.28
12	0.1	0.8	0.1	0.53	0.72
13	0.1	0.9	0	0.6	0.75
14	0.01	0.98	0.01	0.48	0.65
15	0.01	0.99	0	0.34	0.17
16	0.5	0	0.5	0.75	0.93
17	0.5	0.5	0	0.63	0.3
18	0.8	0.1	0.1	0.26	0.23
19	0.9	0	0.1	0.45	0.51
20	0.9	0.1	0	0.5	**0.04**
21	0.33	0.33	0.33	0.60	0.83
22	0.98	0.01	0.01	0.25	**0.03**
23	0.99	0	0.01	0.24	0.16
24	0.99	0.01	0	0.39	0.75
25	1	0	0	0.31	0.65

Table S1 Classes used in Fig. 6 with the corresponding ion concentrations (in 0.1M). Precision = fraction of cases in which ANN gave output Class and its target was exactly that Class out of all the times it gave output the Class. Sensitivity = fraction of cases in which ANN gave output the Class and it was exactly that class out of all the cases in which target was the Class. Random choice would give sensitivity 0.04 and random trial would be bellow sensitivity 0.08 in 95% of trials. In two highlighted cases, the majority of the output was assigned to Class 25 and 24 thus method correctly identified that majority of the ions are Potassium but did not manage to correctly classify the Class.

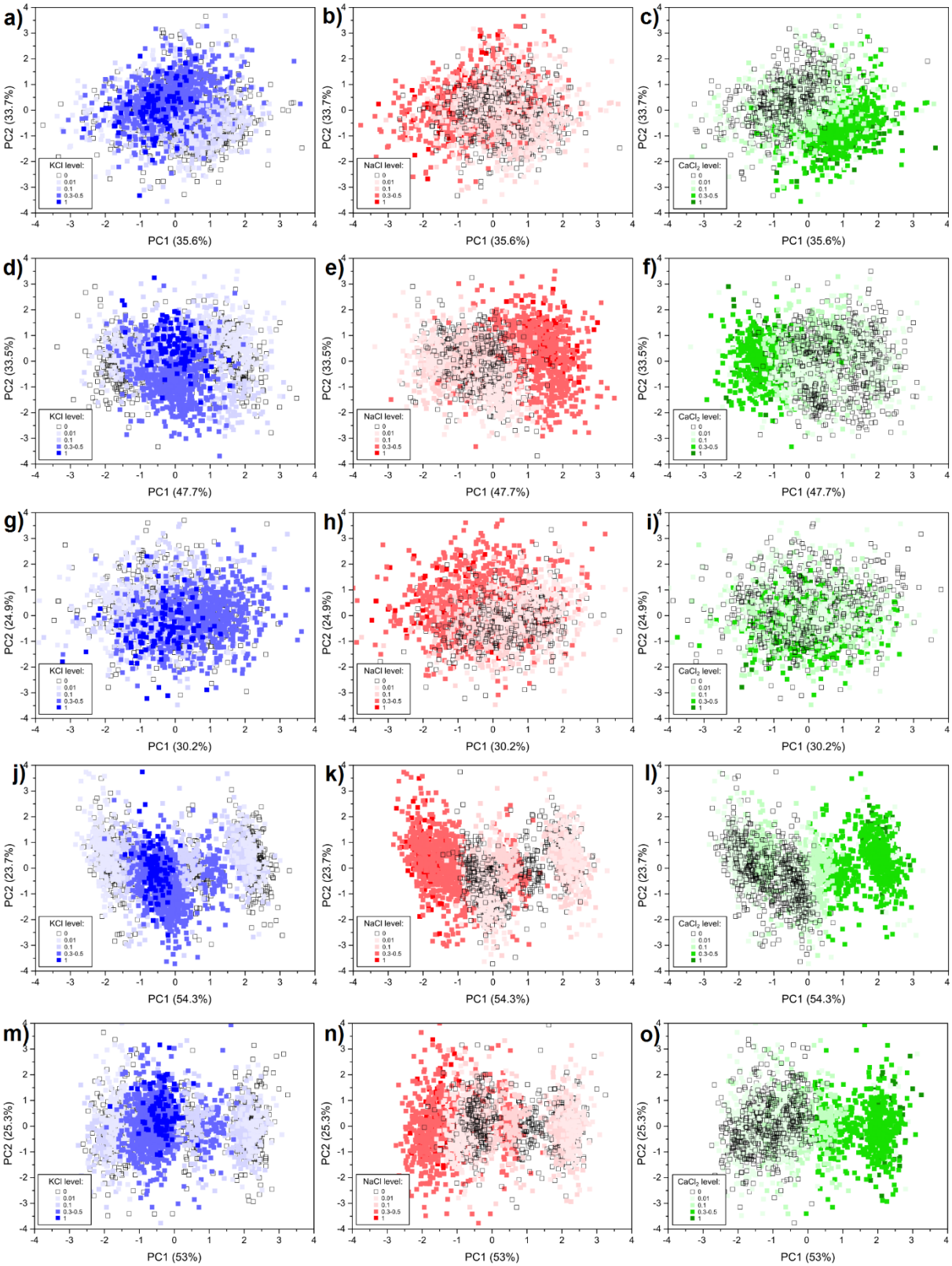

Figure S1 PCA with reduced number of descriptors: a-c) omitting 350 mV data, d-f) omitting 100 mV data, g-i) omitting spike data, j-l) omitting steady-state data and m-o) omitting transient data.

	True Positive	True Negative	False Positive	False Negative	True	False
K ≥ 1	0.00	0.96	0.00	0.04	0.96	0.04
K ≥ 0.98	0.61	0.86	0.39	0.14	0.86	0.14
K ≥ 0.9	0.85	0.96	0.15	0.04	0.93	0.07
K ≥ 0.8	0.97	1.00	0.03	0.00	0.99	0.01
K ≥ 0.5	0.8	0.89	0.19	0.1	0.86	0.14
K ≥ 0.333	0.87	0.97	0.13	0.03	0.92	0.08
K ≥ 0.1	0.86	0.84	0.14	0.16	0.82	0.18
K ≥ 0.01	0.63	0.78	0.37	0.22	0.74	0.26
Na ≥ 1	0.5	0.96	0.5	0.04	0.96	0.04
Na ≥ 0.98	0.90	0.98	0.10	0.02	0.97	0.03
Na ≥ 0.9	0.97	0.99	0.03	0.01	0.98	0.02
Na ≥ 0.8	0.92	0.98	0.08	0.02	0.96	0.04
Na ≥ 0.5	0.99	0.94	0.01	0.06	0.95	0.05
Na ≥ 0.333	0.97	0.87	0.03	0.13	0.90	0.1
Na ≥ 0.1	0.76	0.67	0.24	0.33	0.73	0.27
Na ≥ 0.01	0.77	0.66	0.24	0.36	0.75	0.25
Ca ≥ 1	0.90	0.97	0.10	0.03	0.96	0.04
Ca ≥ 0.98	0.80	0.97	0.20	0.03	0.97	0.03
Ca ≥ 0.9	0.86	0.99	0.14	0.01	0.96	0.04
Ca ≥ 0.8	1	1	0	0	1	0
Ca ≥ 0.5	0.97	0.98	0.03	0.02	0.98	0.02
Ca ≥ 0.333	1	1	0	0	1	0
Ca ≥ 0.1	0.95	0.78	0.05	0.22	0.85	0.15
Ca ≥ 0.01	0.83	0.62	0.17	0.38	0.77	0.23

Table S2 Classification if solution contains more or equal concentration of ions than the one given in the left column, taken from one learning case. In **K ≥ 1** case method systematically never even attempts to positively classify any sample. This is the only case in which disbalance of sampling really affected our method.

Supplementary Information S4 Script for ANN creation (non-defined functions are a standard set of MATLAB libraries).

Script1

```
% Solve a Pattern Recognition Problem with a Neural Network
% Script generated by NPRTOOL
% This script assumes these variables are defined:
%   PulseInputs - input data.
%   Pattern - target data.
inputs = PulseInputs';
targets = Pattern';
% Create a Pattern Recognition Network
% Following two entries for optimal configuration otherwise delete them and call script without them with script in S5 + in all outputs add counter.
hiddenLayerSize = 77;
net = patternnet(hiddenLayerSize);
% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};
% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerandperclass';  % Divide data randomly within each class;
net.divideMode = 'sample';  % Divide up every sample;
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
% training function 'trainlm':
% training functions type: help nntrain
net.trainFcn = 'trainlm';  % Levenberg-Marquardt;
% Choose a Performance Function
net.performFcn = 'mse';  % Mean squared error;
% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist','plotregression', 'plotfit'};
```

```
% Train the Network

    [net,tr] = train(net,inputs,targets);

% Test the Network

    outputs = net(inputs);
    errors = gsubtract(targets,outputs);
    performance = perform(net,targets,outputs);

% Recalculate Training, Validation and Test Performance

    trainTargets = targets .* tr.trainMask{1};
    valTargets = targets  .* tr.valMask{1};
    testTargets = targets  .* tr.testMask{1};

% Here for screening change trainPerformance in trainPerformance(Count) and similar for others...

    trainPerformance = perform(net,trainTargets,outputs);
    valPerformance = perform(net,valTargets,outputs);
    testPerformance = perform(net,testTargets,outputs);

% Screening script for testing network architectures,

    Count=0;
    for j=1:3
    for i=1:ceil(100./j)
    Count=Count+1;
    if j==1
    net = patternnet([i])
    elseif j==2
    net = patternnet([i,i])
    elseif j==3
    net = patternnet([i,i,i])
    end
```

Script1;