Supporting information

Automatically identifying electrode reaction mechanisms using deep neural networks

Gareth F. Kennedy[†], *Jie Zhang*[*,†,‡] *and Alan M. Bond*[*,†,‡]

† School of Chemistry, Monash University, Victoria 3800, Australia

‡ ARC Centre of Excellence for Electromaterials Science, School of Chemistry, Monash University

Corresponding Author

*alan.bond@monash.edu

*jie.zhang@monash.edu

Contents

```
# build the model using sequential layers
model = Sequential()
# 3 sequential 2D convolution layers followed by Rectified Linear Unit (Relu)
#   activation function and a 2D max pooling layer
model.add(Conv2D(32, (3, 3), input_shape=x_train.shape[1:]))    # n_paras = 320
model.add(Activation('relu'))                                   # out = (98, 98, 32)
model.add(MaxPooling2D(pool_size=(2,2)))                        # out = (49, 49, 32)
model.add(Conv2D(32, (3, 3)))                                   # n_paras = 9248
model.add(Activation('relu'))                                   # out = (47, 47, 32)
model.add(MaxPooling2D(pool_size=(2,2)))                        # out = (23, 23, 32)
model.add(Conv2D(64, (3, 3)))                                   # n_paras = 18496
model.add(Activation('relu'))                                   # out = (21, 21, 64)
model.add(MaxPooling2D(pool_size=(2,2)))                        # out = (10, 10, 64)
# flatten to 1D data
model.add(Flatten())                                            # out = (6400)
# dense connection to a hidden layer of variable number of nodes
model.add(Dense(dnnHLNodeNumber))                               # n_paras = 448070
# another Relu activation function
model.add(Activation('relu'))                                   # out = (dnnHLNodeNumber)
# randomly set 50% of input units to 0: helps prevent overfitting
model.add(Dropout(0.5))
# final output layer of the 3 classifications (E, EE and EC mechanisms)
model.add(Dense(n_labels))                                      # n_paras = 213
# sigmoid activation function for the final classification layer
model.add(Activation('sigmoid'))                                # out = (3)
# compile the model using the ADAM optimizer for a specified learning rate
adam_opt = keras.optimizers.Adam(lr=dnnLearningRate)
model.compile(optimizer=adam_opt, loss='categorical_crossentropy', metrics=['accuracy'])
# Train the model, iterating dnnEpochs times over the data in batches of dnnBatch samples
model.fit(x_train, y_train, epochs=dnnEpochs, batch_size=dnnBatch)
```

Figure S1. Python code using the Keras library in TensorFlow to give the DNN architecture schematically shown in Figure 2. The initial training data is a normalized greyscale image of 100x100 pixels which goes through 3 convolution layers before being flattened and densely connected to a hidden layer of a variable number of nodes and finally to an output layer with one node for each classification. The output image and number of filters (32 or 64) from each layer as well as the number of parameters for that layer are shown as a comment at the end of each line. The final classification output layer contains a series of 3 values corresponding to the probabilities that the input mechanism is an E, EE or EC mechanism.

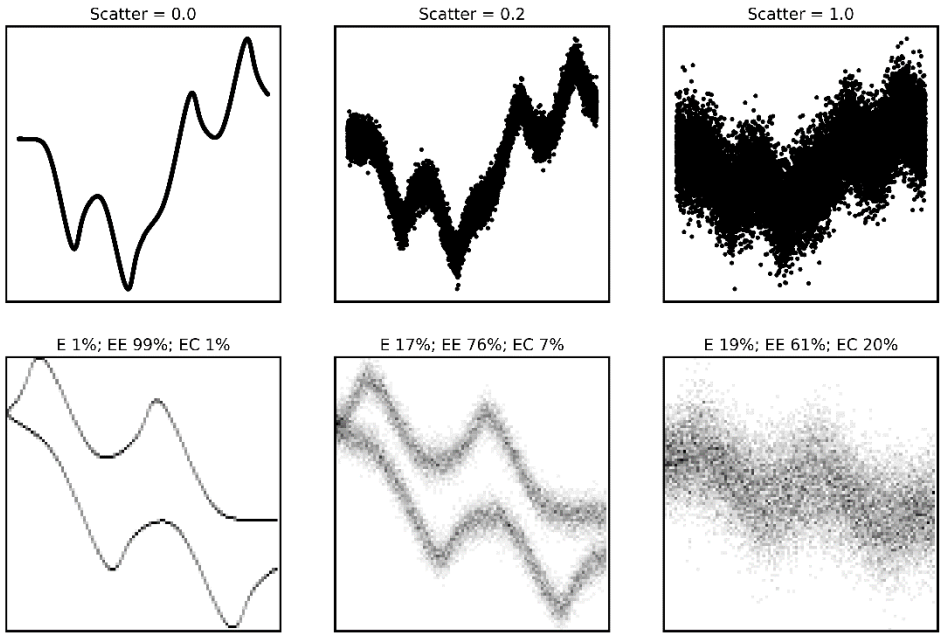

Figure S2. Effect of adding noise to the current as a function of time (top panels) for an EE mechanism example and the associated DNN input images (bottom panels). The simulated data for the EE mechanism before noise is added uses the default parameters given in Table 1 except for R_u = 50 Ω, k^0_1 = k^0_2 = 1 cm s^{-1} and E^0_1 = 0.2 V and E^0_2 = -0.2 V which were chosen to further confuse the classifier.

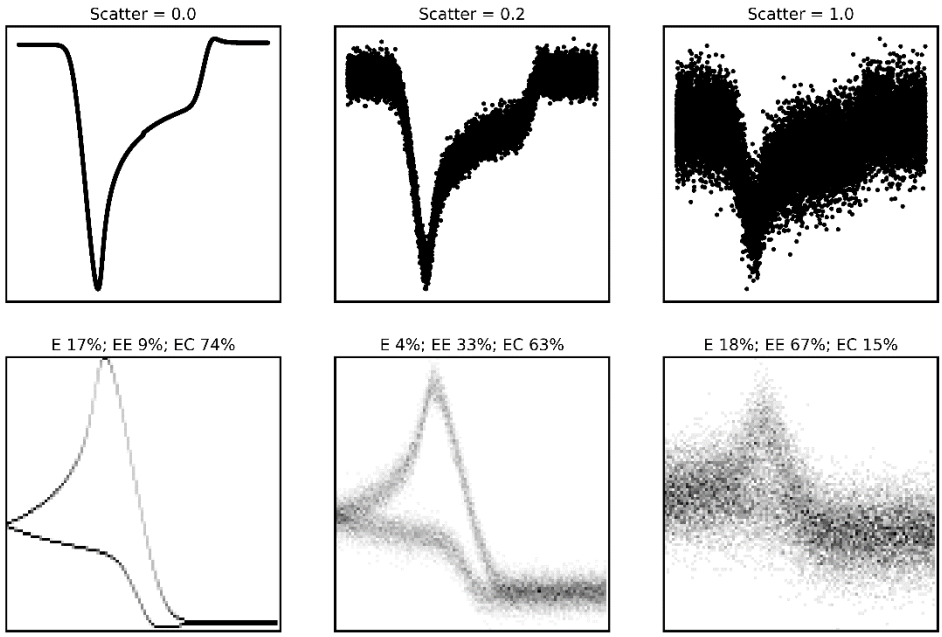

Figure S3. Effect of adding noise to the current as a function of time (top panels) for an EC mechanism example and the associated DNN input images (bottom panels). The simulated data for the EC mechanism before noise is added uses the default parameters given in Table 1 except for R_u = 50 Ω, k^0 = 1 cm s^{-1} and k_f = 1 s^{-1} which were chosen to further confuse the classifier.

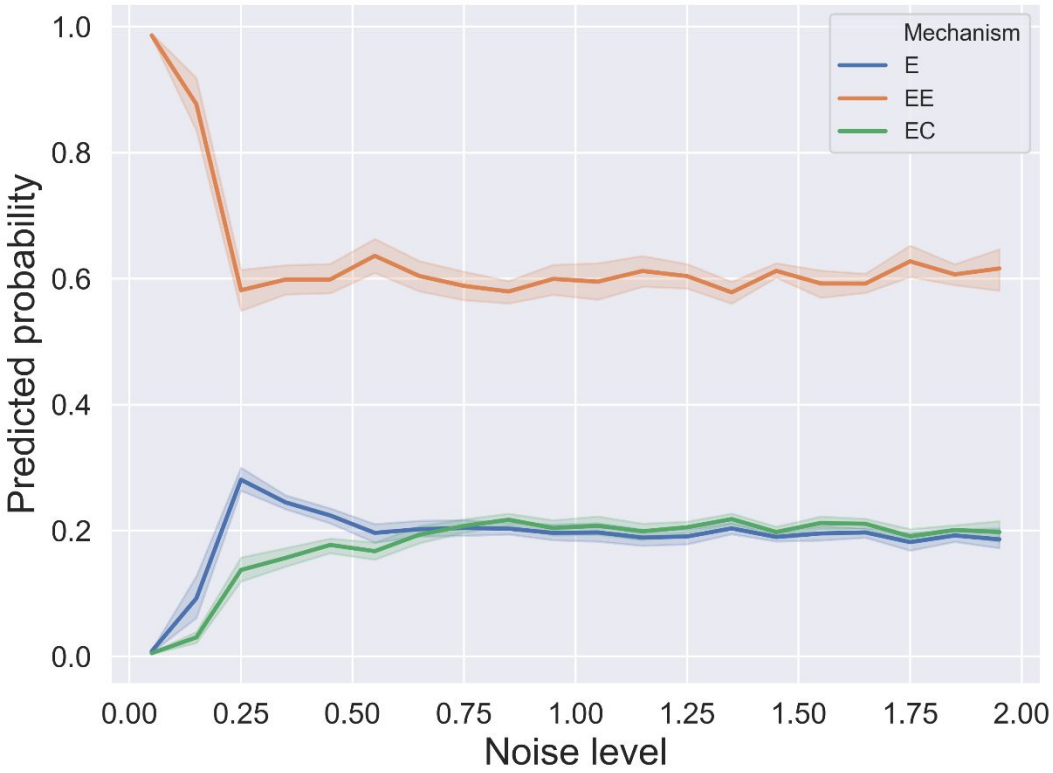

Figure S4. Effect on the classification probabilities of increasing the noise added to the simulated data for an EE mechanism shown in Figure S2.

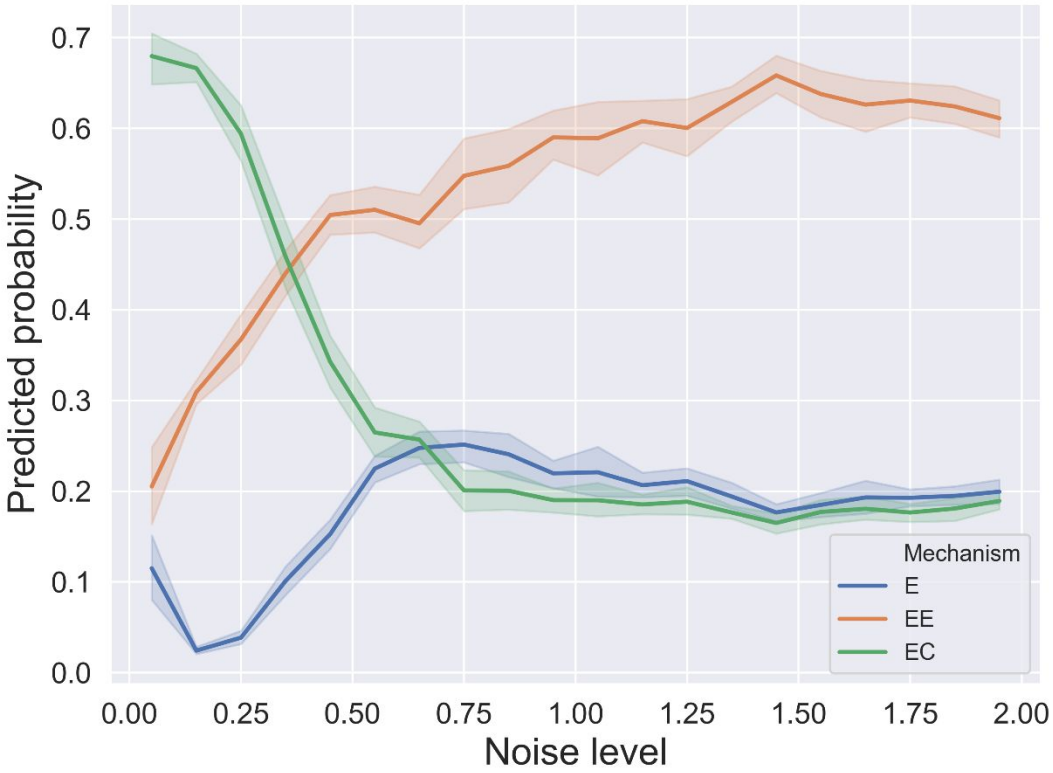

Figure S5. Effect on the classification probabilities of increasing the noise added to the simulated data for an EC mechanism shown in Figure S3.