Experiment Setup

1. Given dataset

In the given dataset [1] (available: http://archive.ics.uci.edu/ml/machine-learning-databases/00361/), the resistances captured by each of the eight sensors (in KOhm) on four types of gas (Ethylene, Ethanol, Carbon Monoxide, and Methane) of ten levels of concentration are gathered, together with the elapsed time being 600 seconds, as a 9-column array. The dataset contains 640 samples in total. While the capture rates differ, sample sizes are different. We define the resolution of a sensor, or sensor resolution, as the number of captured points during the time window, in this case the window size is 600s.

In Table 1, we list the number of samples whose sensor resolution falls into one of the identified ranges. Over 98.5% of samples (631 out of 640) have a resolution greater than 50000. For the convenience of classifier fitting, we first trim the original dataset by dropping the samples with the resolution smaller than 50000 and shear the tail data beyond the 50000th captured point. Therefore, we get a trimmed dataset containing 631 samples of the equal size. It is named as "Original_trimmed" and denoted as "ori_t" for simplicity. The dimension of ori_t is (50000, 8, 631), thus (# rows of a sample, # columns of a sample, # samples). The in the original dataset, there are nine columns in a sample with the first column being the time. We drop the time column and keep the rest eight columns as eight curves that can be plotted in a graph. All the following datasets developed in our experiment are built on it.

Table 1 The distribution of sensor resolution

Sensor resolution	[19289, 50k)	[50k, 57k)	[57k, 58k)	[58k, 59k)	[59k, 60k)	[60k, 60001]
# sample	9	21	22	30	415	143

2. Data compression (Or other names such as Preprocessing, Dimension Reduction, Compression etc.)

High-resolution censored data can be a burden in classification tasks. They are computationally heavy and, in our case, highly redundant. In Figure 1, we plot the original censored data after shearing the tail. Its resolution is 50000 and curves are fuzzy. To reduce the computational complexity and smoothen the curves, we propose the following down-sampling strategies based on the convolution operation.

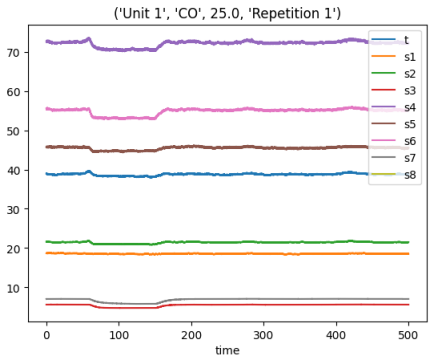


Figure 1 Plot of sample "B1_GCO_F010_R1.txt" in ori_t: s# is the sensor number

2.1 Down sampling

The shape of a sample data in ori_t is (50000, 8) and each column is the resistance numbers captured by a sensor. Therefore, we can plot eight curves as shown above. To lower the resolution of each curve, we

apply a convolution operation on the sample data using a (c, 1) kernel at stride lengths being (1, c). Thus, it moves column by column horizontally and slides vertically in a non-overlapping manner. In Figure 2, we provide an example of convolving using a (3, 1) kernel whose strides are (1, 3). This convolution reduces the size of each column from 12 to 4 and the new value is the mean of the corresponding three neighbors. The down sampling strategies are essentially the designs of convolution kernel.

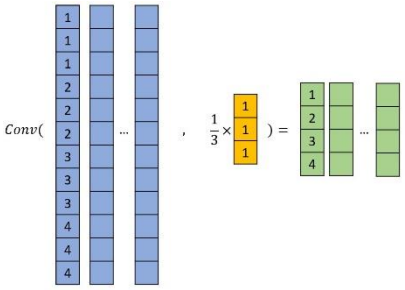


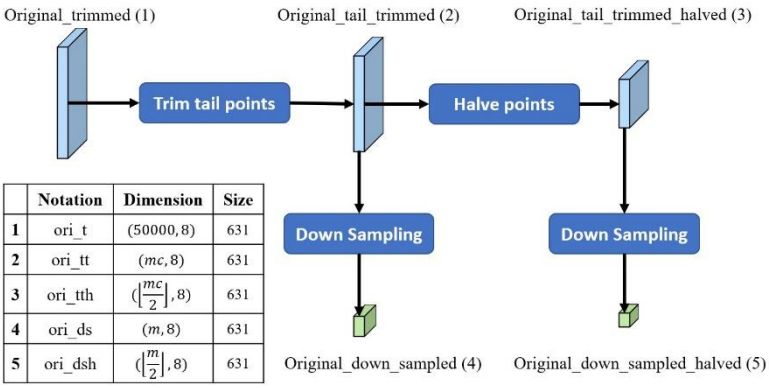
Figure 2 Convolve using a (3,1) kernel: a demo

We adopt the following notations throughout our experiment. The shape of a sample data is (m, n) where m is the sensor resolution, or the number of captured points during 500s, and n is the number of sensors, which is fixed to 8. The size of a dataset is k, which is the number of sample data. In the original dataset, m has various values and k = 640. In ori_t, m = 50000 and k = 631.

To apply down sampling, or to lower the sensor resolution, we set the target resolution (m) as the initial parameter. Then we define the compression factor (c) as floor(50000/m). Sometimes m does not evenly divide 50000, in this case, the curve is trimmed from 50000 to mc. This tail-trimmed dataset is called "Original_tail_trimmed" and denoted using ori_tt.

Based on our observations, the second half of the censored data tends to generate a flat curve. This lack of fluctuation suggests the useful features are likely to be absent. On the other hand, obvious changes in the curve are present in the first half. We are curious if the first half of the data are sufficient for the classification tasks. Therefore, we halve ori_tt and generate a new dataset called "Original_tail_trimmed_halved" (ori_tth).

We apply various down sampling operations on both ori_tt and ori_tth and construct the corresponding down sampled dataset, denoted as ori_ds and ori_dsh. The above descriptions of dataset preparations and the information of each dataset are depicted in Figure 3.



	Notation	Dimension	Size
1	ori_t	$(50000, 8)$	631
2	ori_tt	$(mc, 8)$	631
3	ori_tth	$(\lfloor \frac{mc}{2} \rfloor, 8)$	631
4	ori_ds	$(m, 8)$	631
5	ori_dsh	$(\lfloor \frac{m}{2} \rfloor, 8)$	631

Figure 3 Datasets preparations

In Figure 4, we list five designs of the down sampling kernel, whose dimensions are (c, 1) and strides are (1, c). Here c is consistent with compression factor (c). Below we concisely discuss each design.

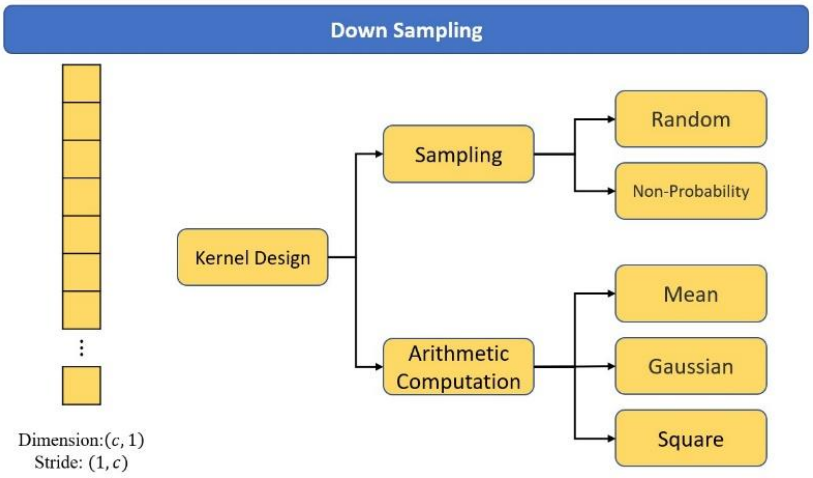


Figure 4 Kernel designs

2.1.1 Sampling

To design a kernel for sampling, it is first initialized as a (c, 1) vector filled with 0. Then one of them is set to 1 based on certain selection rules. Two rules are:

- Random Sampling (RS): The index is selected by generating a random integer number ranging from 0 to c-1. Then the number at the random index is set to 1 while others are 0.
- Non-Probability Sampling (NPS): We design a naïve selection rule as the index is fixed to 0. In other words, the kernel is $[1, 0, 0, \ldots, 0]^T$.

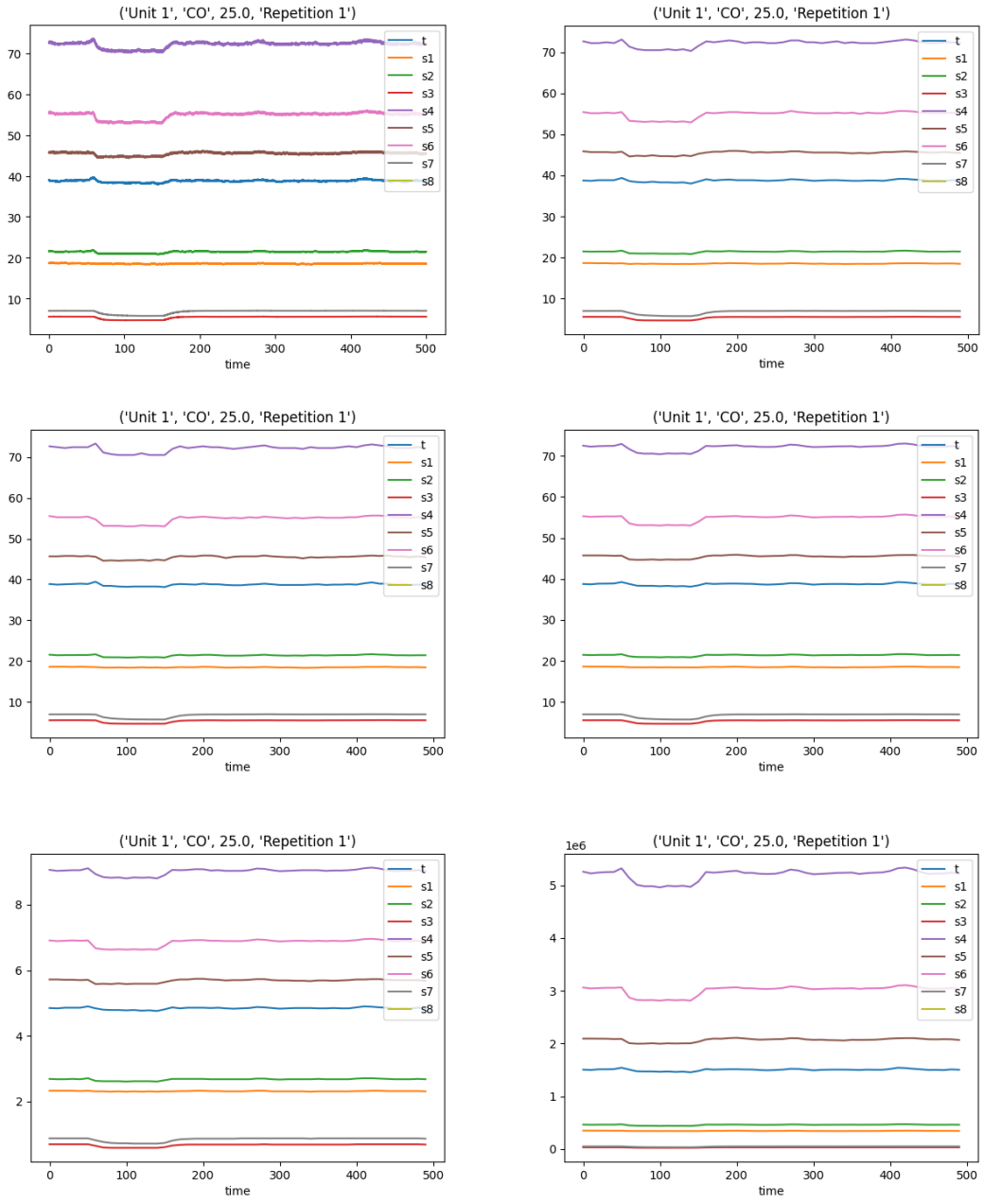
2.1.2 Arithmetic Computation

Different from sampling kernels, more complicated computations are introduced in arithmetic computation kernels.

- Mean: A mean kernel is configured to be 1/c * $[1, 1, \ldots, 1]^T$.
- Gaussian: A Gaussian kernel is a set of numbers that are sampled from a standard Gaussian distribution ranging from -4 to 4, which covers over 99.73% of the whole distribution. It is implemented using "Gau_kernel = norm.pdf(np.arange(-4, 4, 8/c), 0, 1).reshape((c, 1))", where "norm" is imported from "scipy.stats", "np" is the imported "numpy" library, and "c" is the kernel size, or compression factor.
- Square: We would like to enlarge the difference among the captured data points so that the fluctuation/jitter can be captured as features for classification tasks. In other words, we would like to increase the variance of the captured data points. One economical way to realize this amplification is applying the square operation. To implement sum of square operation using a kernel, an additional step is needed: AA^T, where A is a sample data from ori_tt/ori_tth. Then we design a summation kernel as $[1, 1, \ldots, 1]^T$ and convolve it with AA^T.

2.2 Applying down sampling kernels: a demo

Below, we apply the proposed five kernels to a sample data in ori_tt. The compression factor c is set to 50. Since 50000 can be evenly divided by c, ori_tt and ori_t are identical.



The plots, from left to right, top to bottom, are before down sampling, applying RS, NPS, Mean, Gaussian, and Square. We observe the new curves are smoother, while some on the similar order of magnitude (RS, NPS, Mean), Gaussian on a smaller magnitude and Square on a much larger magnitude. With slight shape differences among the first four designs, we observe the difference among captured data points are enlarged after applying the square operation, which is as expected.

2.3 Flatten the data

We mainly focus on reducing the size of a single sample data in previous parts. With the compressed sample data, we now look at the whole dataset. As mentioned in Figure 3, all the datasets are 3-dimensional because they contain 2-D sample data.

Since a sample data consists of eight columns of captured resistance numbers by eight independent sensors, it is safe to conclude that there is no spatial relationship among these eight curves. Therefore, we can concatenate the columns and flatten the sample data to 1-D. As a result, the dataset becomes 2-dimensional. For ori_tt, its shape is (8mc, 631). For ori_ds, it is (8m, 631).

3. Classification and test accuracies

We adopt the K-nearest neighbors (KNN) algorithm for this four-gas classification task. Sklearn library provides an implementation of KNN classifiers as "KNeighborsClassifier". The input k value specifies the threshold of classification. We experimented with k value being {1, 2, 3, 4, 5}.

Two steps are required to determine the test accuracy of the classifier: fitting the labeled training data, and comparing the difference between the predictions of the unlabeled test data and ground truth labels.

While we only have a single dataset (ori_ds/ori_dsh), we randomly split it into a training part and a test part by specifying the test data proportion using "sklearn.model_selection.train_test_split()" function. The proportion ranges from 0 to 1 and the respective training data proportion equals 1- test data proportion. Therefore, if the test proportion is too large, the training data may not be sufficient. We experimented with the test proportions being {0.1, 0.2, 0.3, 0.4, 0.5, 0.6}.

To better justify the effectiveness of KNN classifiers on this task, we also experimented with various target sensor resolutions (*m*). For simplicity, we only consider *m* that evenly divide 50000 in this preliminary experiment. They are selected as {10, 20, 40, 50}. We define the overall test accuracy in each target resolution as the mean of the best accuracies generated by the k value among all the test proportions. In Table 2, we list the overall test accuracies of four target resolutions with various kernel options. The full results from the perspective of the target resolution are provided in Appendix A.

Table 2 Classification accuracies with various kernels and target resolutions

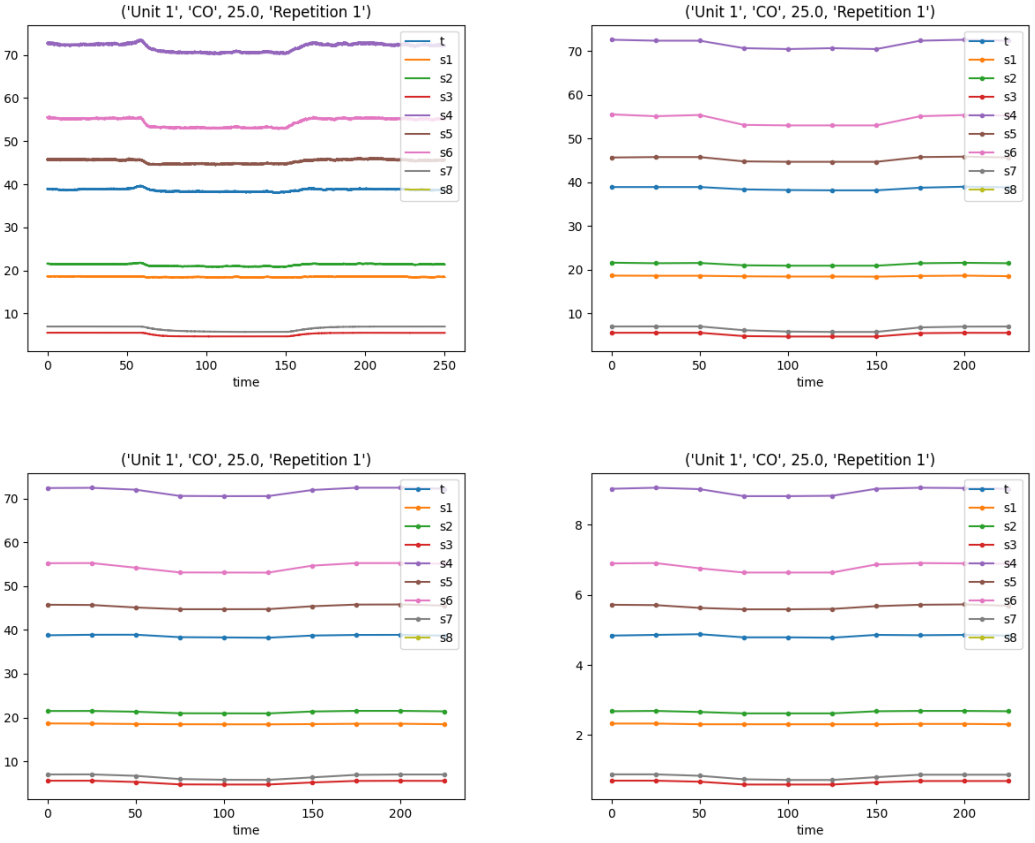
kernel \ m	10		20		40		50	
RS	0.9072	0.9209	0.9125	0.9239	0.9405	0.944	0.9478	0.9457
NPS	0.9341	**0.9434**	0.9493	0.9416	0.9561	0.9422	0.9463	0.9529
Mean	**0.9608**	**0.9504**	0.9486	0.9431	0.9401	0.9589	0.9574	0.9492
Gaussian	0.9443	**0.9434**	0.9462	0.9491	0.9589	0.9468	0.9396	0.9405
Square	0.9161	0.9247	0.9126	0.9174	0.9213	0.9296	0.9203	0.9125

When the accuracies are comparable, smaller value of target resolution (*m*) is preferred because it suggests that fewer captured censor data is sufficient for the classification task. Therefore, we mainly focus on the accuracies under the condition that $m = 10$. The corresponding kernel size is (5000, 1). The best accuracy (0.9608 or 96.08%) is achieved when compressing the whole data using a mean kernel. On halved data, NPS kernel and Gaussian kernel generate comparable accuracies (94.34%) to the mean kernel (95.04%). Below, we plot compressed sample data using kernels with top performance. They are

NPS, mean and Gaussian kernel on a halved sample data. On the top left is the halved original data from ori_tth. On the top right is the data after applying NPS kernel. On the bottom left, is the halved data applying mean kernel and on the bottom right is that applying Gaussian kernel. In the plot of the compressed data, each curve is constructed using 10 points (as $m = 10$). These points are explicitly shown.

Regarding the difficulty of implementation of these three kernels (NPS, mean, and Gaussian), NPS is a clear winner because it is indeed not a convolution operation. While mean kernel and Gaussian kernel must involve a convolution operation, which is more complicated to implement using our hardware. Another advantage of NPS kernel is that no floating-point computation is needed because all the numbers in the NPS kernel are 0s and one of them is 1. In mean kernel and Gaussian kernel, most numbers are floating-point, which increases the implementation complexity and slower to compute.

To conclude, each censor only needs to sample or capture 10 resistance data points during the first half of the gas release window. These captured data are sufficient to distinguish the gas using a KNN classifier.



Appendix A

A.1 RS kernel

m=10	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9219	0.8976	0.8947	0.913	0.8956	0.8892	0.9375	0.9291	0.9368	0.9012	0.9209	0.8997
2	0.9219	0.8268	0.8263	0.8735	0.8291	0.8232	0.9062	0.8898	0.9158	0.8814	0.8924	0.8734
3	0.9531	0.7953	0.8474	0.8498	0.8354	0.8259	0.8906	0.8819	0.9105	0.8775	0.8987	0.8997
4	0.9062	0.7638	0.8263	0.8221	0.8259	0.8179	0.875	0.8425	0.9053	0.8577	0.8892	0.8786
5	0.8906	0.7795	0.8211	0.834	0.8291	0.8074	0.875	0.8583	0.9158	0.8538	0.8639	0.8496
Best	0.9531	0.8976	0.8947	0.913	0.8956	0.8892	0.9375	0.9291	0.9368	0.9012	0.9209	0.8997
Mean	0.9072						0.9209					

m=20	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9375	0.937	0.9263	0.9209	0.8639	0.8892	0.9219	0.9134	0.9263	0.9644	0.9335	0.8839
2	0.8906	0.9134	0.8947	0.8735	0.8323	0.8522	0.9062	0.8583	0.8947	0.9289	0.8956	0.8707
3	0.9062	0.8898	0.9158	0.8893	0.8259	0.8575	0.9062	0.8504	0.9105	0.9486	0.9051	0.8734
4	0.8906	0.9055	0.9	0.8617	0.7975	0.8417	0.8906	0.8583	0.8842	0.9407	0.8861	0.8654
5	0.9062	0.8976	0.9053	0.8538	0.7911	0.8549	0.9062	0.8346	0.9053	0.9328	0.8797	0.876
Best	0.9375	0.937	0.9263	0.9209	0.8639	0.8892	0.9219	0.9134	0.9263	0.9644	0.9335	0.8839
Mean	0.9125						0.9239					

m=40	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9688	0.9449	0.9474	0.9289	0.9241	0.9288	0.9688	0.9449	0.9474	0.9289	0.9557	0.9182
2	0.9219	0.8976	0.9053	0.9012	0.9019	0.8918	0.9531	0.9213	0.9263	0.8972	0.9146	0.8865
3	0.9219	0.9213	0.9211	0.9091	0.8892	0.8971	0.9688	0.9291	0.9211	0.9091	0.9241	0.9024
4	0.9062	0.8976	0.9105	0.9012	0.8671	0.8734	0.9531	0.9291	0.9316	0.913	0.9019	0.876
5	0.9062	0.9134	0.9053	0.9012	0.8449	0.8786	0.9531	0.937	0.9105	0.917	0.9177	0.8786
Best	0.9688	0.9449	0.9474	0.9289	0.9241	0.9288	0.9688	0.9449	0.9474	0.9289	0.9557	0.9182
Mean	0.9405						0.944					

m=50	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9688	0.9528	0.9789	0.9486	0.9272	0.9103	0.9844	0.9528	0.9316	0.9526	0.9241	0.9288
2	0.9062	0.9134	0.9263	0.917	0.8956	0.8892	0.9688	0.8976	0.9	0.9091	0.8829	0.9103
3	0.9062	0.937	0.9316	0.9328	0.9051	0.8786	0.9688	0.9055	0.9211	0.9091	0.8924	0.9103
4	0.9062	0.8898	0.9053	0.9091	0.8734	0.8628	0.9531	0.9055	0.8947	0.9091	0.8861	0.8892
5	0.9375	0.8898	0.9053	0.913	0.8956	0.8654	0.9531	0.8898	0.8947	0.913	0.8734	0.9024
Best	0.9688	0.9528	0.9789	0.9486	0.9272	0.9103	0.9844	0.9528	0.9316	0.9526	0.9241	0.9288
Mean	0.9478						0.9457					

A.2 NPS

m=10	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9688	0.9213	0.9368	0.9368	0.9304	0.9103	0.9688	0.9528	0.9684	0.917	0.9272	0.9261
2	0.8906	0.874	0.9263	0.9091	0.8734	0.8681	0.9531	0.9213	0.9211	0.9012	0.8892	0.905
3	0.9219	0.8819	0.9105	0.9051	0.8797	0.8734	0.9531	0.9291	0.9368	0.9012	0.8987	0.8997
4	0.875	0.8504	0.8842	0.8775	0.8671	0.8549	0.9531	0.8976	0.9053	0.8893	0.8734	0.8839
5	0.9062	0.8661	0.8684	0.8893	0.8703	0.847	0.9531	0.9055	0.9211	0.8735	0.8797	0.8839
Best	0.9688	0.9213	0.9368	0.9368	0.9304	0.9103	0.9688	0.9528	0.9684	0.917	0.9272	0.9261
Mean	0.9341						0.9434					

m=20	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9688	0.9606	0.9737	0.9526	0.9114	0.9288	0.9531	0.9291	0.9632	0.9526	0.9335	0.9182
2	0.9531	0.9213	0.9263	0.9012	0.8797	0.8892	0.9219	0.9055	0.9211	0.8893	0.8987	0.8839
3	0.9688	0.937	0.9368	0.9091	0.8797	0.8945	0.9375	0.9291	0.9263	0.913	0.9051	0.8892
4	0.9375	0.9291	0.9105	0.8893	0.8671	0.8496	0.9375	0.8976	0.9053	0.9051	0.8766	0.8813
5	0.9531	0.9449	0.9105	0.9091	0.8829	0.8707	0.9531	0.9134	0.9053	0.8933	0.8892	0.8813
Best	0.9688	0.9606	0.9737	0.9526	0.9114	0.9288	0.9531	0.9291	0.9632	0.9526	0.9335	0.9182
Mean	0.9493						0.9416					

m=40	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9688	0.9606	0.9789	0.9486	0.9589	0.9208	0.9375	0.9528	0.9421	0.9447	0.9367	0.9393
2	0.9062	0.9055	0.9421	0.8893	0.9272	0.8865	0.8906	0.9134	0.9	0.913	0.9019	0.8892
3	0.9375	0.9134	0.9474	0.8972	0.9051	0.8997	0.9062	0.9134	0.9053	0.917	0.9082	0.8997
4	0.9062	0.8898	0.9421	0.8735	0.9114	0.8839	0.875	0.8898	0.9053	0.9012	0.8861	0.8813
5	0.9219	0.8976	0.9368	0.8617	0.9082	0.8892	0.9062	0.9055	0.9053	0.917	0.8956	0.8971
Best	0.9688	0.9606	0.9789	0.9486	0.9589	0.9208	0.9375	0.9528	0.9421	0.9447	0.9367	0.9393
Mean	0.9561						0.9422					

m=50	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9531	0.9764	0.9474	0.9447	0.9272	0.9288	0.9531	0.9843	0.9684	0.9644	0.9209	0.9261
2	0.9219	0.937	0.9263	0.8854	0.8766	0.8892	0.9375	0.9449	0.9316	0.9249	0.8892	0.876
3	0.9062	0.937	0.9105	0.8893	0.8829	0.8839	0.9375	0.9449	0.9474	0.9249	0.9019	0.8945
4	0.9219	0.8976	0.8895	0.8617	0.8576	0.8707	0.9375	0.937	0.9158	0.9051	0.8703	0.8707
5	0.9219	0.9213	0.9053	0.8577	0.8639	0.847	0.9375	0.937	0.9421	0.917	0.8956	0.8734
Best	0.9531	0.9764	0.9474	0.9447	0.9272	0.9288	0.9531	0.9843	0.9684	0.9644	0.9209	0.9261
Mean	0.9463						0.9529					

A.3 Mean

m=10	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9844	0.9685	0.9632	0.9526	0.962	0.934	0.9688	0.9449	0.9632	0.9407	0.9589	0.9261
2	0.9062	0.937	0.9211	0.9209	0.9146	0.8786	0.9375	0.9213	0.9368	0.913	0.9177	0.8786
3	0.9062	0.9528	0.9368	0.9091	0.9114	0.8654	0.9531	0.9213	0.9368	0.9289	0.9335	0.8918
4	0.875	0.9213	0.9316	0.9091	0.8829	0.8391	0.9375	0.9055	0.9421	0.9051	0.9146	0.8654
5	0.875	0.937	0.9211	0.8854	0.8956	0.847	0.9375	0.9213	0.9263	0.9091	0.9177	0.8813
Best	0.9844	0.9685	0.9632	0.9526	0.962	0.934	0.9688	0.9449	0.9632	0.9407	0.9589	0.9261
Mean	0.9608						0.9504					

m=20	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9688	0.9528	0.9842	0.9368	0.9494	0.8997	0.9688	0.9449	0.9526	0.9249	0.9335	0.934
2	0.9062	0.9291	0.9526	0.9051	0.9114	0.8417	0.9531	0.8976	0.9105	0.8972	0.9051	0.8997
3	0.9375	0.937	0.9526	0.917	0.9177	0.8734	0.9688	0.9055	0.9158	0.8893	0.9051	0.9024
4	0.875	0.9055	0.9474	0.9012	0.8987	0.8232	0.9375	0.9055	0.9	0.8893	0.8892	0.8839
5	0.9062	0.9213	0.9316	0.9091	0.8987	0.8522	0.9531	0.9134	0.9053	0.8814	0.8829	0.8865
Best	0.9688	0.9528	0.9842	0.9368	0.9494	0.8997	0.9688	0.9449	0.9526	0.9249	0.9335	0.934
Mean	0.9486						0.9431					

m=40	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9844	0.9528	0.9316	0.9051	0.9589	0.9077	0.9688	0.9843	0.9632	0.9684	0.9399	0.9288
2	0.9844	0.9134	0.8579	0.8814	0.9146	0.8575	0.9219	0.9685	0.9263	0.917	0.9019	0.8918
3	0.9688	0.9291	0.8842	0.8933	0.9209	0.8522	0.9375	0.9606	0.9368	0.9209	0.9114	0.8865
4	0.9844	0.9055	0.8579	0.8419	0.8892	0.8391	0.9219	0.9449	0.9158	0.8933	0.8956	0.8918
5	0.9688	0.9055	0.8842	0.8696	0.8734	0.8311	0.9375	0.9449	0.9211	0.8933	0.8987	0.8786
Best	0.9844	0.9528	0.9316	0.9051	0.9589	0.9077	0.9688	0.9843	0.9632	0.9684	0.9399	0.9288
Mean	0.9401						0.9589					

m=50	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9844	0.9843	0.9737	0.9328	0.943	0.9261	0.9531	0.9843	0.9526	0.9447	0.9209	0.9393
2	0.9375	0.937	0.9368	0.9209	0.8987	0.8865	0.9219	0.9449	0.9053	0.917	0.8956	0.8997
3	0.9375	0.9528	0.9474	0.9249	0.9051	0.8945	0.9375	0.9528	0.9158	0.9328	0.9114	0.8971
4	0.9375	0.9213	0.9421	0.8972	0.8766	0.876	0.9062	0.9528	0.8842	0.9012	0.8956	0.8892
5	0.9375	0.937	0.9316	0.8972	0.8892	0.8813	0.9062	0.9528	0.8789	0.9091	0.8797	0.8654
Best	0.9844	0.9843	0.9737	0.9328	0.943	0.9261	0.9531	0.9843	0.9526	0.9447	0.9209	0.9393
Mean	0.9574						0.9492					

A.4 Gaussian

m=10	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9375	0.9528	0.9632	0.9644	0.9241	0.9235	0.9688	0.9134	0.9474	0.9605	0.9494	0.9208
2	0.9062	0.8976	0.9316	0.9289	0.8797	0.8628	0.9062	0.8898	0.9	0.9209	0.8956	0.8865
3	0.9062	0.9134	0.9421	0.9407	0.8861	0.8496	0.9219	0.9055	0.9105	0.9209	0.9209	0.8865
4	0.8906	0.8819	0.9316	0.9051	0.8671	0.8602	0.9062	0.8898	0.8895	0.913	0.8987	0.8865
5	0.8906	0.8819	0.9316	0.9091	0.8671	0.8443	0.9219	0.8898	0.8842	0.9091	0.9019	0.8839
Best	0.9375	0.9528	0.9632	0.9644	0.9241	0.9235	0.9688	0.9134	0.9474	0.9605	0.9494	0.9208
Mean	0.9443						0.9434					

m=20	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9531	0.9449	0.9737	0.9605	0.9399	0.905	0.9844	0.9606	0.9579	0.9565	0.9146	0.9208
2	0.9219	0.8976	0.9105	0.9209	0.8924	0.8681	0.9531	0.937	0.9053	0.8972	0.8797	0.8734
3	0.9219	0.9213	0.9158	0.9447	0.9146	0.8839	0.9531	0.937	0.9105	0.9209	0.8861	0.8786
4	0.9375	0.8661	0.9105	0.913	0.8892	0.8628	0.9375	0.937	0.8895	0.8933	0.8703	0.8522
5	0.9062	0.8819	0.9263	0.9209	0.8987	0.8734	0.9375	0.9449	0.9	0.9051	0.8829	0.8654
Best	0.9531	0.9449	0.9737	0.9605	0.9399	0.905	0.9844	0.9606	0.9579	0.9565	0.9146	0.9208
Mean	0.9462						0.9491					

m=40	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9844	0.9764	0.9579	0.9684	0.943	0.9235	0.9688	0.9843	0.9421	0.9447	0.9146	0.9261
2	0.9219	0.9449	0.9263	0.9447	0.9114	0.8839	0.9219	0.937	0.9158	0.9051	0.8797	0.8918
3	0.9062	0.9528	0.9263	0.9447	0.9304	0.8918	0.9375	0.937	0.9263	0.9249	0.8924	0.8918
4	0.875	0.9213	0.9263	0.9368	0.8924	0.8786	0.8906	0.937	0.9105	0.913	0.8797	0.8918
5	0.8906	0.9055	0.9211	0.913	0.8829	0.8549	0.9219	0.9291	0.9105	0.917	0.8797	0.8839
Best	0.9844	0.9764	0.9579	0.9684	0.943	0.9235	0.9688	0.9843	0.9421	0.9447	0.9146	0.9261
Mean	0.9589						0.9468					

m=50	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9219	0.9685	0.9368	0.9328	0.9462	0.9314	0.9688	0.9606	0.9579	0.9289	0.9272	0.8997
2	0.875	0.9134	0.9105	0.8933	0.8956	0.905	0.9219	0.9134	0.9316	0.913	0.8861	0.8734
3	0.8906	0.9213	0.9158	0.9051	0.8987	0.9156	0.9531	0.9291	0.9368	0.917	0.8924	0.8602
4	0.875	0.9055	0.8947	0.8696	0.8671	0.8997	0.9375	0.9213	0.9368	0.8972	0.8861	0.8522
5	0.8594	0.9134	0.8895	0.8893	0.8829	0.8813	0.9531	0.9213	0.9474	0.8933	0.8924	0.8364
Best	0.9219	0.9685	0.9368	0.9328	0.9462	0.9314	0.9688	0.9606	0.9579	0.9289	0.9272	0.8997
Mean	0.9396						0.9405					

A.5 Square

m=10	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9219	0.9449	0.9211	0.8972	0.9146	0.8971	0.9531	0.9449	0.9263	0.913	0.9241	0.8865
2	0.8906	0.9291	0.8789	0.8656	0.8956	0.8602	0.9219	0.9134	0.8842	0.8854	0.8734	0.8681
3	0.8906	0.9449	0.8789	0.8893	0.8734	0.8602	0.8906	0.9134	0.9211	0.8893	0.8671	0.8628
4	0.8438	0.9213	0.8737	0.8735	0.8861	0.8417	0.8906	0.9055	0.8895	0.8893	0.8766	0.8628
5	0.875	0.8898	0.8684	0.8696	0.8734	0.8232	0.9062	0.8976	0.8789	0.8893	0.8576	0.8206
Best	0.9219	0.9449	0.9211	0.8972	0.9146	0.8971	0.9531	0.9449	0.9263	0.913	0.9241	0.8865
Mean	0.9161						0.9247					

m=20	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9375	0.9055	0.9105	0.9289	0.8987	0.8945	0.9531	0.937	0.9263	0.917	0.9082	0.8628
2	0.9219	0.8976	0.8947	0.8854	0.8608	0.8575	0.9219	0.8819	0.8947	0.9051	0.9082	0.8391
3	0.9062	0.8898	0.9	0.8814	0.8797	0.8654	0.9375	0.9055	0.9053	0.8933	0.8987	0.8364
4	0.9219	0.8898	0.8789	0.8775	0.8703	0.8338	0.9375	0.8661	0.8789	0.8854	0.8766	0.8232
5	0.9219	0.8976	0.9	0.8617	0.8513	0.8338	0.9375	0.9055	0.8842	0.8893	0.8734	0.8443
Best	0.9375	0.9055	0.9105	0.9289	0.8987	0.8945	0.9531	0.937	0.9263	0.917	0.9082	0.8628
Mean	0.9126						0.9174					

m=40	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9844	0.8819	0.9316	0.8893	0.9304	0.9103	0.9375	0.9606	0.9263	0.9328	0.9177	0.9024
2	0.9688	0.8504	0.9	0.8458	0.9146	0.8549	0.9062	0.9134	0.8895	0.8538	0.9019	0.8628
3	0.9688	0.8425	0.9211	0.8458	0.9019	0.847	0.9062	0.9213	0.8895	0.8775	0.8892	0.876
4	0.9531	0.8268	0.8895	0.8419	0.8861	0.8338	0.9062	0.9134	0.8737	0.8577	0.8924	0.8681
5	0.9531	0.8504	0.9158	0.8419	0.8797	0.8285	0.9062	0.9213	0.8684	0.8696	0.8734	0.8522
Best	0.9844	0.8819	0.9316	0.8893	0.9304	0.9103	0.9375	0.9606	0.9263	0.9328	0.9177	0.9024
Mean	0.9213						0.9296					

m=50	Full						Half					
k \ p	0.1	0.2	0.3	0.4	0.5	0.6	0.1	0.2	0.3	0.4	0.5	0.6
1	0.9219	0.9291	0.9	0.9328	0.9146	0.9235	0.8906	0.9528	0.9316	0.9407	0.9019	0.8575
2	0.875	0.8976	0.8895	0.8933	0.8797	0.8865	0.8594	0.8976	0.9	0.8933	0.8829	0.8285
3	0.875	0.9134	0.8842	0.8656	0.8829	0.8945	0.8906	0.9213	0.9263	0.8893	0.8671	0.8127
4	0.875	0.874	0.8842	0.8577	0.8671	0.8417	0.875	0.8819	0.9	0.8775	0.8671	0.7968
5	0.9219	0.8819	0.8632	0.8617	0.8766	0.8496	0.875	0.9055	0.8895	0.8577	0.8386	0.7916
Best	0.9219	0.9291	0.9	0.9328	0.9146	0.9235	0.8906	0.9528	0.9316	0.9407	0.9019	0.8575
Mean	0.9203						0.9125					

Reference

[1] Fonollosa, J., et al. "Calibration transfer and drift counteraction in chemical sensor arrays using Direct Standardization." *Sensors and Actuators B: Chemical* 236 (2016): 1044-1053.