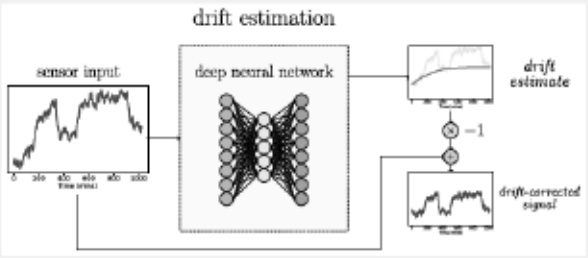
Real-Time Low-Cost Drift Compensation for Chemical Sensors Using a Deep Neural Network With Hadamard Transform and Additive Layers

Diaa Badawi, Agamyrat Agambayev, Sule Ozev, and A. Enis Cetin, *Fellow, IEEE*

***Abstract*—In this paper, we propose a computationally efficient deep learning framework to address the issue of sensitivity drift compensation for chemical sensors. The framework estimates the underlying drift signal from sensor measurements by means of a deep neural network with a multiplication-free Hadamard transform based layer. In addition, we propose an additive neural network which can be efficiently implemented in real-time on low-cost processors. The temporal additive neural network structure performs only one multiplication per "convolution" operation. Both the regular network and the additive network can have Hadamard transform based layers that implement orthogonal transforms over feature maps and perform soft-thresholding operations in the transform domain to eliminate noise. We also investigate the use of the Discrete Cosine Transform (DCT) and compare it with the Hadamard transform. We present experimental results demonstrating that the Hadamard transform outperforms the DCT.**



***Index Terms*—Chemical sensor drift, chemical sensor, time series analysis, discrete cosine transform, Hadamard transform, convolutional neural networks.**

I. Introduction[1]

DRIFT correction is a crucial pre-processing step for reliable and accurate gas analyte detection and identification in chemical sensors and Electronic nose (E-nose) systems [1]–[5]. Sensor drift causes the characteristics of a chemical sensor's response to change over time. It is due to multiple factors, such as variations in temperature and humidity, aging and the so-called sensor poisoning [6]. An electronic nose system can neither be reliable nor accurate without addressing the sensor drift problem [7].

Manuscript received March 23, 2021; revised May 10, 2021; accepted May 11, 2021. Date of publication May 27, 2021; date of current version August 13, 2021. This work was supported in part by the National Science Foundation (NSF) under Grant 1739396 [University of Illinois at Chicago (UIC)] and Grant 1739451 [Arizone State University (ASU)]. The associate editor coordinating the review of this article and approving it for publication was Prof. Irene Taurino. *(Corresponding author: Diaa Badawi.)*

Diaa Badawi and A. Enis Cetin are with the Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL 60607 USA (e-mail: dbadaw2@uic.edu; ecyy@uic.edu).

Agamyrat Agambayev and Sule Ozev are with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: aagambay@asu.edu; sule.ozev@asu.edu).

Digital Object Identifier 10.1109/JSEN.2021.3084220

[1]A preliminary version of this work was submitted to The international Conference on Acoustics, Speech, & Signal Processing (ICASSP) 2021.

Recently, there has been great interest in drift correction in the sensors community by utilizing machine learning algorithms. Zhang *et al.* [8] propose an unsupervised subspace projection method, named Domain Regularized Component Analysis (DRCA), which aims at adapting the distribution of the drifted data to that of drift-free data for analyte classification. Liu *et al.* [9] propose an online active-learning algorithm that calibrates sample drift for class identification. In [10], Tao *et al.* propose learning drift invariant features in an adversarial manner by minimizing the Wasserstein distance to perform domain adaptation between the drifted data and the drift-free data domains. In [11], we proposed a generative adversarial framework to train a discriminator-classifier network to learn drift-invariant feature parameters using the chemical sensor dataset collected by Vergara *et al.* [6].

While the previous work addresses long-term drift in chemical sensors, machine learning algorithms therein are implemented over several extracted features, such as the maximum and minimum values of the original time-series data. Unfortunately, the original time-series measurement data is not available in [6], [9]–[11]. Huang *et al.* propose a Papoulis-Gerchberg (PG) algorithm-based method for drift correction. The PG algorithm-based algorithm first extrapolates the drift signal from the observed data by assuming that the drift

signal is a band-limited low-pass signal [12]. This PG-based algorithm is not applicable to on-line, real-time applications since PG algorithm is an iterative method that requires computing successive Fourier Transforms (FT) between time and frequency domains until a satisfactory convergence level is achieved. Furthermore, some baseline drift samples should be known for convergence. Other chemical sensor drift estimation approaches include Kalman filtering and shallow neural network-based methods [7], [13]–[17].

In this paper, we propose using deep learning to estimate the sensor drift signals from raw time-series data in real-time. Deep convolutional neural networks (CNNs) have excelled in various time-series related tasks, such as prediction, interpolation, and classification [18]. Researchers have recently used CNNs in the analyte classification problem [11], [19], [20]. CNNs have been increasingly preferred to recurrent neural networks in time-series recognition problems thanks to their highly flexible architectures and relative ease to train [21]. Since our goal is to carry out drift correction in real-time using a causal regression framework, we propose to use novel temporal convolutional neural networks (TCNNs) for sensor drift estimation. TCNNs have been able to outperform recurrent neural networks over a number of benchmark data sets [21], [22]. In TCNNs, convolutional layers implement causal convolution (or correlation), meaning that the current output only depends on the current and previous input values. Convolutions are carried out at different dilation rate, thus enabling the network to learn long- and short-term features for the task.

To take advantage of the slowly varying nature of the sensor drift signal, we propose incorporating orthogonal transforms and thresholding layers in the TCNNs architecture to produce smooth intermediate features, which in turn will generate a smooth drift estimate. This approach also removes the noise in the observed data. In particular, we compute the Hadamard Transform and Discrete Cosine Transform (DCT) over sliding windows of the past and current intermediate feature maps and apply soft thresholds to the high-frequency components in the transform domain. The transform layer will essentially suppress the high-frequency components and regularize the features. The thresholding parameters are also learned during training using artificially created data. Both transforms are fast and can be implemented efficiently using $O(n \log n)$ operations. Hadamard transform is even faster than the DCT as it is a multiplication-free transform, which can be constructed from the Haar wavelet transform [23]. Furthermore, we replace the convolutional layers of TCNN with multiplication-free additive layers that can be implemented efficiently on a low-performance processor to design an energy-efficient and low-cost system.

Our results show that the proposed framework can accurately provide smooth and slowly varying drift estimates from the sensor measurements in real-time, even for severely degraded sensors.

The organization of this paper is as follows: In Sec. II, we review the sensor drift problem. In Sec. III we describe the TCNN framework, the transform domain layers, and the additive convolutional layers. In Sec. IV, we present and discuss our experimental results. Finally, we provide our conclusion in Sec. V.

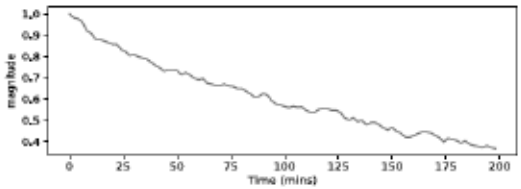


Fig. 1. Example of drift signal showing the low-frequency nature of the drift signal.

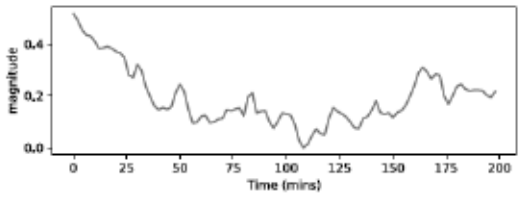


Fig. 2. Example of drift signal showing that the drift signal can increase and decrease.

II. The Sensor Drift Problem

Sensor drift is a common problem in chemical sensors, such as the E-nose technology, that can lead to inaccurate measurements. There are many sources of drift in chemical sensors, such as binding of molecules to the sensor surface, electronic aging of components, environmental contamination, and temperature variations. As an example, Figures 1 and 2 show the sensor measurements in a stable environment for two electronic nose (E-nose) sensor devices, characterized by Jet Propulsion Lab (JPL) [35]. The sensors are designed to detect the methane gas.

In Fig. 1, the signal decays over time although there is no methane gas excitation. Comparatively, in Fig. 2 the baseline drift signal first decreases until around $t = 110$ minutes and slowly increases afterwards. Ideally, both sensors should not have produced any output. The measured output therefore is the offset, which drifts in time. Furthermore, although the sensors are identical in design, they exhibit different drift signals. These two examples show that it is difficult to rely on analytical models to characterize drift signals due to the individual nature of each sensor.

While sensor drift can vary significantly from device to device, the common consensus is that the sensor drift signal is a slowly varying baseline signal [6], [12], [24].

In [12], it is further assumed that the drift signal is a band-limited lowpass signal and the drift estimation is formulated as an interpolation problem. The iterative Papoulis-Gerchberg (PG) algorithm [25]–[27] is used to construct missing parts of the drift signal. In particular, this drift estimation method assumes that the sensor is not exposed to a chemical gas initially and at the end of the measurement cycle. PG algorithm imposes time and frequency domain information in an iterative manner to perform interpolation operation until both time and frequency constraints are satisfied or a satisfactory convergence level is achieved. The shortcoming of the PG method is the need for prior information about the gas exposure to be able to interpolate section of the drift signal

corresponding to gas exposure [12]. As pointed above, it is also necessary to assume a low-pass bandwidth for the drift signal in [12]. On the contrary, we do not assume any prior bandwidth information. The neural network automatically imposes sparsity constraints on the drift signal in the transform domain during training by learning the transform domain soft thresholds. Furthermore, we do not use future samples during the DCT and Hadamard transform computations to estimate the drift signal. We use only the past and current samples of sensor measurement signal $y(t)$ to estimate the drift signal $d(t)$. As a result, we compute a real-time estimate of $d(t)$.

In this paper, we study the real valued DCT and Hadamard transforms instead of the Discrete Fourier Transform, which is complex. The Hadamard transform is based on the Haar wavelet transform and can be computed without performing any actual multiplication operations.

III. TCNN With Spectral Transform Domain Layers

TCNNs have been widely used in time-series related tasks [21]. Typically, TCNNs are made up of successive blocks of convolutional layers and residual connections. The convolutional layers carry out causal one-dimensional convolution at different dilation rates. The so-called dilated convolution is expressed mathematically as follows:

$$y_r^a[n] := \sum_{c=0}^{C-1} \sum_{k=0}^{K-1} h^a[k,c] \times x[n - rk, c], \tag{1}$$

where n is the time index, c is the input channel index, a is the output feature map index, and r is the dilation rate. One block typically is made of dilated convolutional layer, followed by 1×1 convolutional layer, another dilated convolutional layer, and finally a residual connection between the output and the input of the block.

In this paper, we are interested in finding a drift estimate $\hat{d}[n]$ at time $t = nT_s$ given the sensor time series $y[t]$ for $t \in \{0, T_s, 2\ T_s, \ldots, nT_s\}$, where T_s is the sampling period. Notice that we have a causal baseline drift signal estimation framework. After we estimate $\hat{d}[n]$, we can obtain an estimate of the desired sensor response signal $\hat{p}[n]$ by subtracting the drift signal from $y[n]$. The TCNN structure is suitable for this causal time-series estimation task because the dilated convolution operation not only uses recent samples (short-term features) but also past samples (long-term features) to estimate the current output.

In Section 3.2 we explain how we implement transform domain processing as a part of a TCNN structure and in Section 3.2, we describe how we can develop an additive-TCNN using a novel operator, which we introduced in [28], [29]. In Section 3.3, we describe the architecture of the deep network, which we used in both regular and additive-TCNNs.

A. Transform Domain Thresholding Blocks

The sensor drift signal is a slowly varying signal without any high-frequency noise. Therefore, we need a way to obtain a smooth estimate using the deep learning structure. We perform denoising and smoothing using orthogonal transforms [30], [31]. We propose novel orthonormal transform-based blocks to perform smoothing and denoising as a part of the deep neural network. The orthonormal blocks with soft-thresholding feature serve as smoothing units. We perform orthonormal transform operations in sliding causal windows to make it compatible with the online estimation task because feature parameters will be shifted by one time step at a time.

Let $\{f_n^i\}_{n\in\mathbb{N}}$ be the i-th feature map of a specific layer resulting from the earlier convolutional layers of the neural network at time step n. The corresponding feature vector $\mathbf{f}_n^i$ is defined using a sliding window as follows:

$$\mathbf{f}_n^i := \left[f^i[n]\ f^i[n-1]\ \ldots\ f^i[n-N+1]\right]^T \tag{2}$$

where N is the size of the causal sliding window. We select N to be power of 2 to take advantage of fast $O(N\log N)$ efficiency of the Hadamard transform and the DCT. The transformed feature vector, denoted by $\mathbf{F}_n^i$, is defined as:

$$\mathbf{F}_n^i = \mathbf{W}_N \mathbf{f}_n^i \tag{3}$$

where $\mathbf{W}_N \in \mathbb{R}^{N\times N}$ is the transform matrix.

The Hadamard Transform (HT) is based on an orthogonal matrix defined using the recursive relation:

$$\mathbf{H}_N = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \mathbf{H}_{N/2} \tag{4}$$

with $\mathbf{H}_0 = [1]$ and $\otimes$ is the Kronecker product. The matrix $\mathbf{W}_N = \frac{1}{\sqrt{N}}\mathbf{H}_N$ is the unitary version of the orthogonal HT. It is essentially constructed from butterflies and it does not require any actual multiplication operation to compute the transform domain coefficients. It can also be constructed from the Haar wavelet transform [23]. It is worth mentioning that there are different ordering conventions for the Hadamard matrix. For example, the four-by-four sequency-ordered Walsh-Hadarmard transform matrix is defined as follows:

$$\mathbf{H}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{5}$$

The first row of the matrix approximates the action of two successive halfband lowpass filtering and down-sampling operations and generates the DC (or low-low) output with an approximate bandwidth of $[0, \pi/4]$. The second row of the matrix generates the so-called low-high output with the approximate bandwidth of $[\pi/4, \pi/2]$, the third-row is the high-low and the last row is the high-high with approximate bandwidths $[\pi/2, 3\pi/4]$, $[3\pi/4, \pi]$, respectively.

In the case of DCT, $\mathbf{W}_N$ is given by the N-point type-II unitary DCT matrix, whose entries are defined as follows:

$$[\mathbf{W}_N]_{ij} = \begin{cases} \sqrt{\frac{2}{N}}\cos\left(\frac{\pi j}{2N}(2i+1)\right) & i \neq 0 \\ \sqrt{\frac{1}{N}} & i = 0 \end{cases} \tag{6}$$

It is well-known that the DCT approximates the Karhunen-Loeve transform when the correlation between the entries of the input vector is high.

In the transform domain we apply a soft-thresholding function to each coefficient except the DC value to obtain a sparse transform domain representation as follows:

$$\text{SoftTh}(x) = \begin{cases} x - b, & x \geq b \\ 0, & |x| \leq b \\ x + b, & x \leq -b, \end{cases} \tag{7}$$

where b is the soft-thresholding parameter that we learn during the training process. In practice, we do not apply the soft-thresholding function to the first $N_o = 3$ values of the DCT transformed feature parameters and the first value (DC) value of the Hadamard transformed features. This is required to preserve the DC level of the drift estimate over the sliding windows. This is also a common practice in denoising [30], [31]. After we obtain the thresholded transform domain coefficients, we compute the inverse transform and we feed the resulting smoothed feature vector to the next layer of the deep neural network. We repeat the same process whenever we get a new reading from the sensor.

Unlike the work of [12], we do not assume any specific bandwidth for the low frequency drift signal, the soft-thresholding parameters of each transform domain coefficient are automatically learned during training. It is worth mentioning that we call the transform domain thresholding blocks as spectral domain thresholding blocks in this paper as both Hadamard and the DCT transforms approximate the DFT.

B. Additive TCNN With Transform Domain Layers

While CNNs are very powerful, they are computationally expensive because of the large number of add-multiply operations needed for the inference phase. This becomes more of a pressing issue when deploying CNNs on embedded systems for real-time monitoring, in which case the power and computational capabilities are often limited. In our bid to reduce the computational cost of CNNs, we utilize the so-called additive (multiplication-free) layers introduced in [28]. Additive nets also improve the energy efficiency of the network because they perform only one multiplication per convolution operation [18]. Additive nets have been applied to many recognition tasks in fire detection, gas leak detection, and time-series prediction [18], [20], [32]. The additive neural networks are based on the multiplication-free operator defined for the scalar case as follows:

$$x \oplus y := \text{sgn}(xy)(|x| + |y|) \tag{8}$$

where sgn(.) is the signum operation determining the sign of regular multiplication, and the operation defined in Equation (8) is generalized to vectors as follows:

$$\mathbf{x}^T \oplus \mathbf{y} := \sum_i x_i \oplus y_i = \sum_i \text{sgn}(x_i y_i)(|x_i| + |y_i|) \tag{9}$$

for two vectors $\mathbf{x}$ and $\mathbf{y} \in \mathbb{R}^D$. As one can see from Equation (9), each of the summation terms can be computed by adding two numbers ($|x_i|$ and $|y_i|$), and the sign can be obtained using the logical XOR operation between the sign bits of x_i and y_i. When $\mathbf{x} = \mathbf{y}$, $\mathbf{x}^T \oplus \mathbf{x} = 2||\mathbf{x}||_1$, which is the scaled version of the ℓ_1 norm of $\mathbf{x}$. Furthermore, Equation (8) can be expressed as

$$x \oplus y = \text{sgn}(x)y + \text{sgn}(y)x \tag{10}$$

Therefore, the vector operation defined in Equation (9) can be implemented using binary operations similar to the Hadamard transform and there is no need to perform any multiplications to compute the $\oplus$-"dot" product. In the same fashion, the $\oplus$-based dilated "convolution" can be defined as follows:

$$y_r^a[n] := \beta \sum_{c=0}^{C-1} \sum_{k=0}^{K-1} h^a[k, c] \oplus x[n - rk, c], \tag{11}$$

where β is a normalization factor used to scale down the resulting $\oplus$ product. Although normalization by β requires a multiplication, it is performed once for each $y_r^a[n]$ value, as opposed to $C \times K$ multiplication operations required in regular dilated convolution defined in Equation (1). We set $\beta = \frac{1}{\sqrt{C \times K}}$. By selecting a number equal to the power of 2 as β, it is possible to eliminate all the multiplications in the dilated convolution defined in Equation (1).

C. TCNN Architecture

We combine the convolutional units, the transform domain thresholding blocks, and the residual connections to construct our TCNN. At time step n, the input to the TCNN is a vector of size M containing the sensor measurements $y[n], y[n-1], \ldots, y[n-M+1]$ and the output is a single value $\hat{d}[n]$, which is the estimate of the drift signal at time instant $t = nT_s$.

Our TCNN design is summarized in Fig. 3 and Algorithm 1, where `Conv1d(k,r,D)` is the one-dimensional causal convolutional layer with filter size k, dilation rate r, and D output features. `Transform`(32) represents the orthogonal transformation of size 32. L is the number of convolutional blocks and we set it to 7. Each convolutional block carries out temporal convolution, followed by 1×1 convolution, and finally another temporal convolution. We apply orthogonal transforms after the 4-th block. We use the LeakyReLU with leakage factor = 0.2 as our nonlinearity throughout the TCNN.

The network has residual connections between successive blocks similar to the well-known ResNet [33] architecture, which introduced the "identity shortcut connection". These connections skip one or more layers. In our case, the so-called skip connection linearly combines the input of a block, after scaling it by 0.5, to the output of that block. The next block of the network processes $F_{n-1}(x) + 0.5\,x$, where x represents the input to the previous block, and F_{n-1} represents the output of the previous block. The scaling by 0.5 is important so that the magnitude of the final output is stable. This is necessary since we carry out a regression task. We only apply bias in the 1×1 convolutional layers. We do not have the bias term in the transform domain. As it can be seen from Algorithm 1, we have a bottleneck layer that maps from 64 feature channels to 32 and we then map back to 64 feature channels in each block. Using 64 feature channels makes the network powerful enough while not being very computationally expensive, while the bottleneck layer (mapping to 32 channels) regularizes these features.

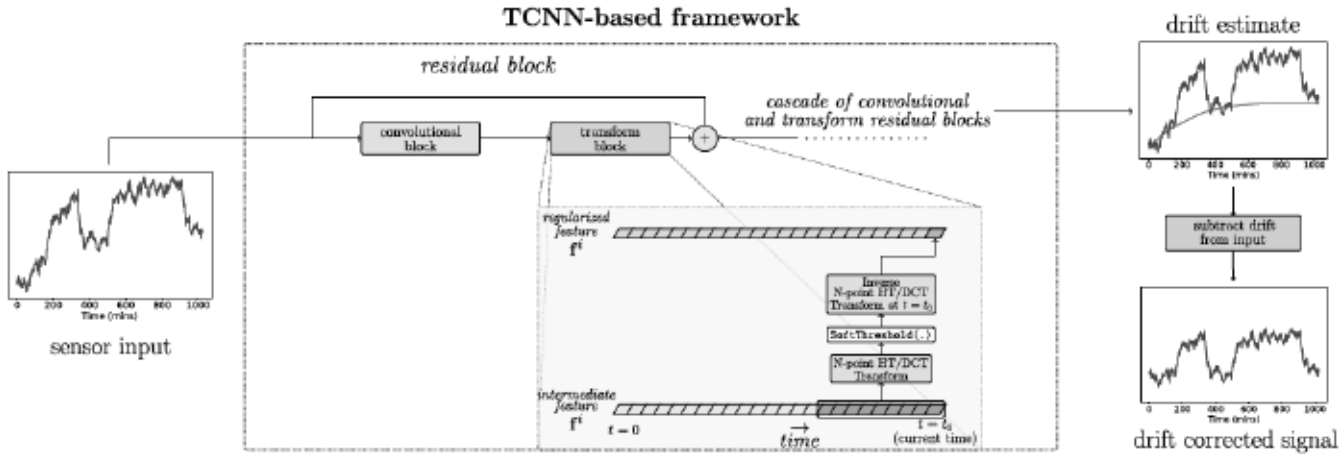


Fig. 3. Block diagram of the proposed system: The system receives sensor measurements and estimates the drift using a TCNN network with transform domain layers. The TCNN network is made up of a cascade of dilated convolutional and orthogonal transform residual blocks. The system then subtracts the drift estimate from the input signal and generates the drift-corrected signal in real-time.

Algorithm 1 Pseudocode of the Design of TCNN. `Conv1D` (k, r, D) Is a Convolutional Layer Of Filters of Sizes k, Dilation Rates r, and Outputs D Feature Maps

procedure TCNN(Input)
  Out = Conv1D(3,1,64)(Input)
  OutRes = LeakyReLU(Out)
  for idx $\in \{1, 2, \ldots, L\}$ **do**
    $r = 2^{\text{idx}}$
    Out = Conv1D(3,r,64)(OutRes)
    Out = LeakyReLU(Out)
    Out = Conv1D(1,1,32)(Out)
    Out = LeakyReLU(Out)
    Out = Conv1D(3,r,64)(Out)
    Out = LeakyReLU(Out)
    if idx>=4 **then**
      Out = Transform(32)(Out)
      Out = SoftThresholding(Out)
      Out = InverseTransform(32)(Out)
    end if
    if idx> 1 **then**
      OutRes = $\frac{1}{2}$Out + $\frac{1}{2}$OutRes
    else
      OutRes = Out
    end if
  end for
  Out = Conv1D(1,1,1)(OutRes)
  return Out
end procedure

The number of blocks is tied to the dilation rates and the input size. Because we double the dilation rate, the number of blocks is in order of the logarithm of the input size. In our case, we have 7 blocks and the dilation rate of the last block is $2^7 = 128$, which means the filters of the last block can look back in time by this amount (128) multiplied by the filter size. The actual "effective" length will still be larger, given the contribution from earlier blocks. increasing the dilation rate after some point will result in having the effective filter size larger than the input size. This means that early filter coefficients will lie outside the input support.

The orthogonal transform can be the Hadamard transform or the DCT depending on the network and it is applied after the 4-th round of convolutions. This is because the size of the features exceed the transform block size of 32 after the 4-th round successive convolutions. "Soft -thresholding" operations have threshold values which are learned during training. We do not apply thresholds to the DC value of the transforms to maintain continuity between the blocks as in many image and audio coding algorithms.

As mentioned earlier, we train the TCNN networks to find a drift estimate $\hat{d}[n]$. In order to do so, we minimize the following regularized cost function:

$$\mathcal{J} := \sum_{n=0}^{N-1} \left(d[n] - \hat{d}[n]\right)^2 + \lambda \sum_{n=1}^{N-1} |\hat{d}[n] - \hat{d}[n-1]| - \gamma \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} \log b_{ij} \quad (12)$$

where the first term in Equation (12) is the reconstruction square errors and the second term is the Total Variation (TV) regularization term [34], [35], which also imposes smoothness on the drift estimates because it minimizes the difference between the neighboring samples. The last term is a log penalty for the soft-thresholding parameters b_{ij} for layer i and channel j. This term is required to make sure that the threshold parameters are pushed away from zero. We use synthetically generated data using Equation 13 and Equation 14 to train the networks, as detailed in Sec. IV.

IV. Experimental and Simulation Results

Datasets: In our experiments, we used data from Electronic-nose (E-nose) sensors used for air quality monitoring. This dataset is collected by the Jet Propulsion Lab (JPL) [36]

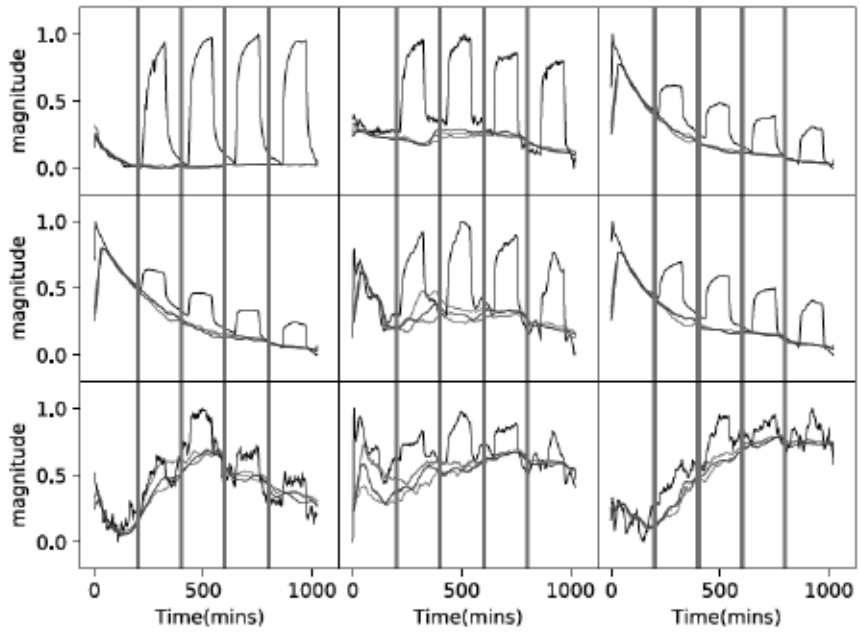


Fig. 4. Test examples from the JPL data set (in black) and the estimated drift from JPL data set using regular (multiplicative) TCNNs with no spectral thresholding (in red), with DCT based layers (in green), and with HT based layers (in blue). Vertical lines mark the beginnings of gas excitations.

using 32 different carbon-polymer sensors. We use data from 16 sensors for the experimental evaluation of the proposed technique. We also collected our own data using the commercially available MQ-137 ammonia sensor which has a detection range of 5-500 ppm [37].

Some sensor recording examples are shown in Fig. 4 and 5. In what follows, we refer to this data set as JPL sensor data. We considered recordings corresponding to single-gas (methanol) exposure experiments as in [12]. In E-nose sensors, the sensor reacts with the vapor upon contact with its surface. Once the sensor is no longer exposed to the gas vapor, it starts exuding the vapor it had absorbed earlier. Absorbing and exuding the vapor depends on the sensor material, the analyte type, and the environment. The ideal sensor response in the absence of drift can be approximated analytically as follows [38]:

$$p(t) = \begin{cases} 0 & t \le T_s \\ \beta\tau \tan^{-1}\left(\frac{t - T_s}{\tau}\right) & T_s \le t \le T_s + \Delta T \\ \beta\tau\left[\tan^{-1}\left(\frac{t - T_s}{\tau}\right) - \tan^{-1}\left(\frac{t - T_s - \Delta T}{\tau}\right)\right] & t \ge T_s + \Delta T \end{cases} \quad (13)$$

where T_s is the starting time of the exposure, and ΔT is the exposure duration.

The sensors are exposed to the gas vapor after 200, 400, 600 and 800 minutes in Fig. 4 and the exposure duration is about 150 minutes. The sensor responses more or less obey the model described in Equation (13), in the first two rows of Fig. 4. Even in some of these sensors, it may not be possible to set a threshold without estimating the drift waveform to detect the VOC gas because the drift waveform is decaying. In the last row of Fig. 4, there are "noninformative" sensors and the pulses due to gas are irregular and noisy.

Three typical sensor recordings obtained from the ammonia sensors are shown in Fig. 6 and Fig. 7. We refer to this data set as the Ammonia data set. The real E-nose data was used as our test set.

We trained and validated our TCNN models on synthetic data that we created. The drift-correct signal modelling is based on Equation (13). We created a total of 10,000 time series with the different levels of gas exposure duration. The length of the time series is resampled to 512 samples. We set the upper bound on the duration of the gas exposure as 500 min in the JPL dataset. We randomized the starting time and the end time of each gas exposure session as well as the parameters β and τ in synthetic training data. To generate slowly varying drift signal, we sampled data from a Gaussian process of mean μ and the covariance given by:

$$\mathrm{Cov}\big(x(t_1), x(t_2)\big) = \exp\Big(\frac{(t_1 - t_2)^2}{\sigma^2}\Big), \quad (14)$$

where σ^2 is a hyperparameter controlling how strongly correlated the samples of the realizations are. The process is locally smooth, and by selecting a large σ^2 value, we can generate slowly varying realizations of the random process. In particular we select σ^2 to be equal to either 2048, 4096 or 8192. The mean value μ is also chosen randomly. Afterwards, the synthetic sensor measurements are created by adding the synthetic drift signal and signals created using Equation (13). Finally, we add zero-mean white noise with various standard deviation levels to the training waveforms.

We trained our networks using the synthetic training data for 80 epochs and we employed early stopping criterion during training based on the reconstruction loss calculated over our validation data set, starting from the tenth epoch. We used mini batches of size 32. We used Adam Optimizer with a learning

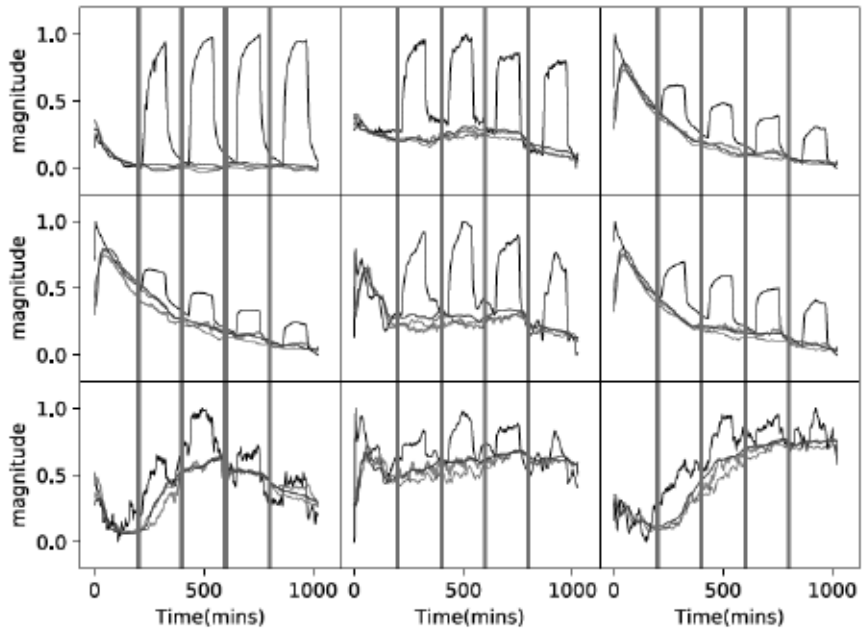


Fig. 5. Test examples from the JPL data set (in black) and the estimated drift using additive (multiplication-free) TCNNs with no spectral thresholding (in red), with DCT based layers (in green), and with HT based layers (in blue).

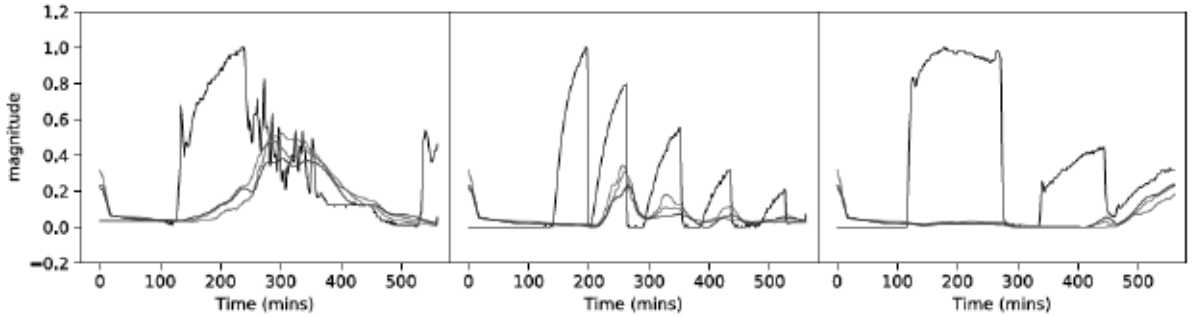


Fig. 6. Results over the Ammonia data set obtained by the three TCNN models. The original signal is in black. The red-colored signals are obtained by the baseline TCNN, while the green- and blue-colored are obtained by the TCNNs with DCT and HT layers, respectively.

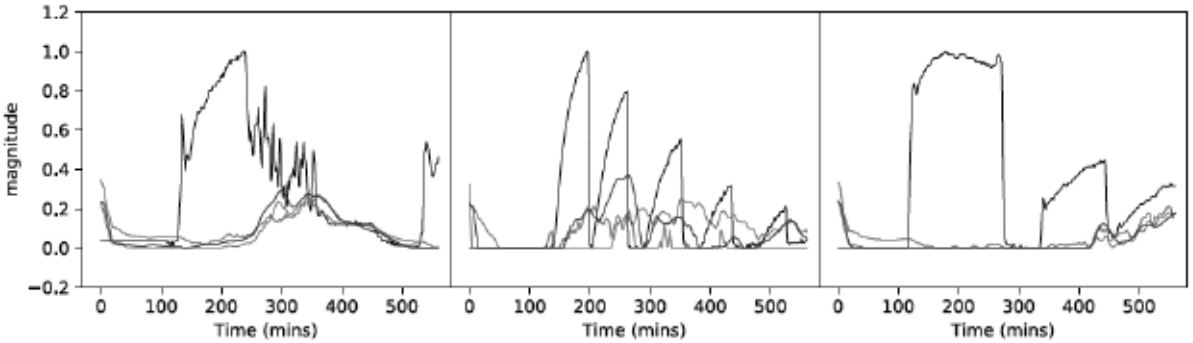


Fig. 7. Results over the Ammonia data set obtained by the three additive TCNN models. The original signal is in black. The red-colored signals are obtained by the baseline TCNN, while the green- and blue-colored are obtained by the TCNNs with DCT and HT layers, respectively.

rate equal to 10^{-3}, $\beta_1 = 0.9$, and $\beta_2 = 0.99$. We set the TV parameter λ in Equation (12) to 0.1 and the log penalty parameter γ to 10^{-3}. Our numerical results over the JPL data set and the Ammonia data set, are shown in Table I and Table II, respectively. We also show the drift estimates for some examples in Fig. 4 for baseline TCNN, TCNN with Hadamard Transform (HT) thresholding layers, and TCNN with DCT thresholding layers. The drift estimates for the additive neural networks are shown in Fig. 5. We included only nine examples in Fig. 4 and 5 to save some space.

TABLE I

MEAN SQUARE ERROR (MSE) OVER THE JPL DATA SET FOR SIX TCNN MODELS AND THE PAPOULIS-GERCHBERG (PG) ALGORITHM: THREE REGULAR AND THREE ADDITIVE (MULTIPLICATION-FREE) MODEL. WE IMPLEMENTED THREE MODELS USING THE REGULAR TCNN AND ADDITIVE TCNN, RESPECTIVELY: "NONE" MEANS REGULAR TCNN, THE SECOND AND THE 3RD COLUMNS REFER TO DCT AND HT BASED TCNNS, RESPECTIVELY. THE LAST THREE COLUMNS REPORT THE MSE RESULTS OF THE PG ALGORITHM FOR DIFFERENT BANDWIDTH SELECTION. THE NUMBERS 8,16, AND 24 CORRESPOND TO THE CUTOFF FREQUENCY INDEX FOR A DFT OF SIZE 4096 USED IN PG ALGORITHM. WE ALSO SHOW THE AVERAGE MEAN-ABSOLUTE ERROR FOR THE DIFFERENT ALGORITHMS USED

Example	Regular Network			Additive Network			PG Algorithm $\left(\frac{f_{cutoff}}{4096}\right)$		
	None	DCT	HT	None	DCT	HT	8	16	24
JPL 0	3.41	4.59	**2.52**	4.17	2.98	4.59	12.42	11.27	5.96
JPL 1	**0.75**	0.98	0.97	1.12	1.09	1.20	2.82	4.48	7.54
JPL 2	1.02	0.97	**0.78**	1.10	1.02	1.23	2.74	6.20	8.85
JPL 3	2.27	**1.52**	1.74	1.55	1.81	1.91	2.21	5.87	7.45
JPL 4	0.75	**0.40**	0.52	0.44	0.88	0.54	13.55	7.15	5.78
JPL 5	1.02	**0.50**	0.73	1.35	0.70	0.34	16.56	9.32	6.04
JPL 6	**4.41**	5.21	5.93	3.02	2.40	2.34	11.43	13.52	20.13
JPL 7	2.39	1.87	**1.56**	2.86	1.65	3.91	4.43	1.86	7.28
JPL 8	1.52	2.45	**1.25**	2.03	1.47	1.15	5.56	10.13	12.47
JPL 9	1.96	3.08	**1.55**	2.10	1.02	1.21	2.51	7.89	11.43
JPL 11	1.11	**0.47**	0.60	1.11	1.49	0.74	19.51	12.33	10.92
JPL 12	5.23	8.27	**4.37**	7.17	3.89	4.90	15.71	12.60	13.86
JPL 13	5.62	8.65	**5.01**	5.88	5.86	6.34	21.23	12.44	11.29
JPL 14	**1.39**	4.14	2.53	1.35	2.99	1.45	7.47	9.81	11.23
JPL 15	3.11	3.92	**2.25**	6.58	8.24	4.49	12.82	8.15	10.77
Average (MSE)	2.63	3.22	**2.35**	3.04	3.20	**2.60**	8.46	8.84	11.62
Average (MAE)	0.07	0.07	0.06	0.07	0.06	0.06	0.23	0.25	0.30

TABLE II

MEAN SQUARE ERROR (MSE) OVER THE AMMONIA DATA SET FOR SIX TCNN MODELS: THREE REGULAR AND THREE ADDITIVE (MULTIPLICATION-FREE) MODEL. "NONE" MEANS A REGULAR TCNN STRUCTURE (WITHOUT TRANSFORM DOMAIN LAYERS). WE ALSO REPORT THE AVERAGE MEAN-ABSOLUTE ERROR FOR THE DIFFERENT MODELS USED

Example	Regular Network			Additive Network		
	None	DCT	HT	None	DCT	HT
Ammonia 1	22.30	28.32	17.95	5.10	10.22	12.85
Ammonia 2	2.99	4.34	2.06	2.12	3.56	4.23
Ammonia 3	1.00	1.86	2.18	2.07	2.70	3.06
Average (MSE)	8.67	11.51	**7.39**	**3.09**	5.49	6.71
Average (MAE)	**0.08**	0.10	0.09	**0.08**	0.10	0.12

As it can be seen from results in Fig. 4 and 5, the TCNN structures with Hadamard Transform (HT) and DCT layers can accurately estimate the drift. The TCNN with the HT layer achieves the lowest average MSE in both regular and additive systems in the JPL data set. The sensors in the 3rd row of Fig. 4 and Fig. 5 generate very noisy data. In all cases, the methane gas was applied to the sensors after roughly 250 min four times. The DCT based network overfits the data compared to the regular TCNN and TCNN with HT layer in one case. In JPL sensor 12 (shown in the first column, third row in Fig. 4), the TCNN with DCT layer misses the first pulse (the green curve). The network follows the sensor measurements very closely. As a result, this methane gas exposure cannot be detected by thresholding the difference between the sensor measurement and the estimated drift signal because they follow the baseline drift level of the sensor. On the other hand, the TCNN with HT layer (blue) and regular TCNN (red) can detect the first pulse due to methane exposure because they generate drift curves which are well-below the sensor measurement data. Therefore, our results in Fig. 4 and Fig. 5 indicate that degraded sensors can be used for methane detection when the sensor data is processed using the TCNN with HT layers in real-time.

In the second set of experiments we used the Ammonia data set. Our sensors are new sensors and we do not observe any baseline drift in the Ammonia data set. Since they are not degraded sensors the baseline level is zero as shown in 6 and 7. The proposed TCNN based algorithms should not generate any false curves in this data set. All of the network models more or less follow the zero baseline level. The results for the Ammonia data set are shown in Fig. 6 and 7 corresponding to regular and additive TCNNs, respectively. It is possible to detect all the ammonia gas pulses because the drift signals are well below the pulse levels except for the additive network with the DCT transform block in the middle signal. Although the overall MSE values of the regular networks are lower than the additive networks as shown in Table II, the estimated drift levels of the regular networks are very close to zero whenever the gas excitation occurs in all the three sensors as shown in Fig. 6. The additive TCNN with HT and DCT layers provide inferior results compared to the regular networks in the middle plot. They may miss the 5th gas exposure in the middle plot of Fig. 7. Only the regular additive network without any transforms (red curve) can detect the 5th pulse of the middle experiment in Fig. 7.

The TCNN with the HT layer generates the lowest MSE among the three regular network models as shown in Table II. The additive networks do not produce as good results as the regular networks but they can be used in low-cost processor embedded systems, such as IoT devices.

The sensor at the bottom right of Fig. 4 and 5 is a "noninformative" sensor. After subtracting the estimated drift waveform one can clearly detect the gas exposure as shown in the demonstrative figure (Fig. 3). We can set up thresholds to detect gas leaks because the sensor signal has an approximate zero baseline level.

It is worth mentioning that the additive network replaces the regular dot-product operations used in convolutions with a correlation operation related with the ℓ_1 norm. As a result, intermediate features obtained in additive network learning turn out to be different from those in traditional networks. This is reflected in our finding that the additive network can achieve better performance on some of the examples in terms of the MSE. We also compared the waveforms using the ℓ_1 norm-based Mean Absolute Error (MAE) and it turns out that the average MAE results of the regular (additive) network with DCT and Hadamard layers are 0.07 (0.06) and 0.06 (0.06), respectively.[2]

A. Comparison With Papoulis-Gerchberg (PG) Algorithm-Based Method

Huang *et al.* developed a PG algorithm-based method for sensor drift estimation method as described in Section 2 [12]. We compared our TCNN-based networks with the method introduced in [12] using the same JPL data set. In our case, we assume that we only know that there is no gas exposure in the initial portions of the sensor recording and an upper bound on the duration of the gas exposure. This is an advantage of our method compared to the PG algorithm based method because in [12], it is assumed they know the exact time that the gas exposure ends, which is not a realistic assumption for a typical gas leak detection application in an open field or inside a building [11], [39].

Next, we present a short summary of the PG method. Let $x[n]$ be the desired discrete-time signal for $n \in \{0, \ldots, N-1\}$. let $I[n \in \mathcal{S}]$ be the indicator function which is equal to 1 if the predicate is true, and zero otherwise. Let $\mathcal{S}$ is the set of "available" samples, which is the first 100 samples out of 512 in JPL sensor recordings. The PG algorithm aims at reconstructing all $x[n]$ values from $x[n]I[n \in \mathcal{S}]$ through the following iteration:

$$\hat{x}^i[n] = x[n] \times I[n \in \mathcal{S}] + \hat{y}^i[n] \times (1 - I[n \in \mathcal{S}]) \quad (15)$$

where

$$\hat{y}^i[n] = \mathcal{F}^{-1}[P(e^{j\omega})\hat{X}^i(e^{j\omega})] \quad (16)$$

where $\hat{X}^i(e^{j\omega})$ is the Discrete-Time Fourier transform (DTFT) of $\hat{x}^i[n]$, $P(e^{j\omega})$ is an ideal low-pass filter performing band-limiting operation, $\mathcal{F}^{-1}$ is the inverse-DTFT operator and i is the iteration index. The algorithm is globally convergent regardless of the initial estimate [25]. One weakness of the PG algorithm is that one has to know the bandwidth of the desired signal *a priori*, which is not possible in general. In addition, the algorithm is very sensitive to the noise present in the available portion of the data [26], [31]. It generates high mean-square-error results as shown in the last three columns of Table I especially for noisy sensor signals shown in the third row of Fig. 4.

To estimate the drift signals, we implemented the algorithm using the Discrete Fourier Transform (DFT) of size 4096. We tried three different bandwidth choices with the following cut-off frequencies: 8/4096, 16/4096, and 24/4096. The MSE results are presented in the last three columns of Table I. As it can be seen from the table, the PG based algorithm does not produce as good results as the deep learning based TCNN structures. Furthermore, the PG algorithm is very sensitive to the quality of the samples. We conclude that the PG algorithm is not suitable for gas leak detection. However, it can be useful when used as an interpolator as in [12].

B. Comparison With the Shallow Multi-Layer Perceptron-Based Predictor

We also compared the deep learning based TCNN methods with the shallow neural network. In [40], a Radial-Basis Function (RBF) neural network is trained to predict the sensor drift using the current and past samples. We also implemented a shallow multi-layer-perceptron (MLP) neural network. In [40], the authors gathered long time series recordings of three different chemical sensors. The recordings correspond to baseline drift signals and the first portion of each time-series are used to train a shallow one-step RBF neural network predictor.

In the JPL dataset, the initial portions of the time series sensor data correspond to the baseline sensor drift.

$$\hat{d}[n] = f(y[n-1], y[n-2], \ldots, y[n-N+1]), \quad (17)$$

where $f(.)$ represents the neural network function, N is the number of past samples used in the nonlinear neural network-based predictor, $y[n] = d[n] + p[n]$ is the augmented input signal at time n containing the underlying drift $d[n]$ and the added pulse signal. We then trained the neural network to minimize the MSE between $d[n]$ and $\hat{d}[n]$ over the training data. Once the MSE converges, we infer the drift over the remaining portion of the signal using the following expression:

$$\hat{d}[n] = \epsilon f(y[n-1], \ldots y[n-N+1]) + (1-\epsilon)\hat{d}[n-1], \quad (18)$$

where $0 < \epsilon \leq 1$ is a memory factor. We use ϵ to account for the fact that the pulse signal might last longer than the predictor order N, in which case the rule in Equation (17) has no means of distinguishing the drift from the additive signal $p[n]$. We trained a one-hidden neural network with input size $N = 16$ with hyperbolic tangent activation function, and a hidden layer of size 32. The results of two examples from the JPL data set are shown in Fig. 8 and Fig. 9. In Fig. 8, we see that it is possible to get a reasonable drift prediction when using an appropriate smoothing factor ϵ. In contrast, in Fig. 9, the initial portion of the drift does not look similar to the remaining portion, resulting in a total failure of the shallow predictor in generating any meaningful drift estimate. This is in contrast to our proposed approach, in which the network can learn all sort of slowly varying signals during the

[2]We did not include the MAE results per example in order not to overcrowd the tables.

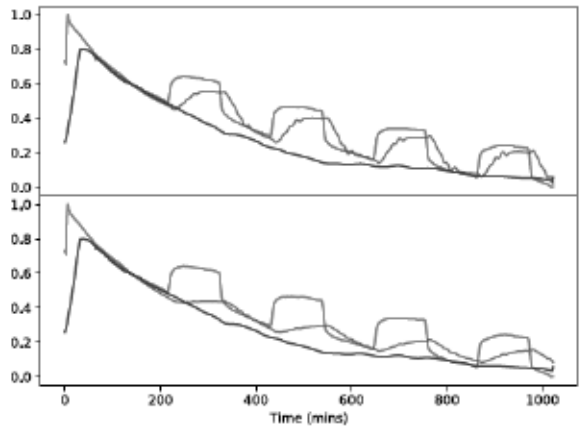


Fig. 8. JPL Sensor 5: Original Signal (in red) and the predicted drift using a MLP predictor (in green), with $\epsilon = 0.1$(top), and 0.7(bottom), compared to the drift estimate of the HT based TCNN (in blue).

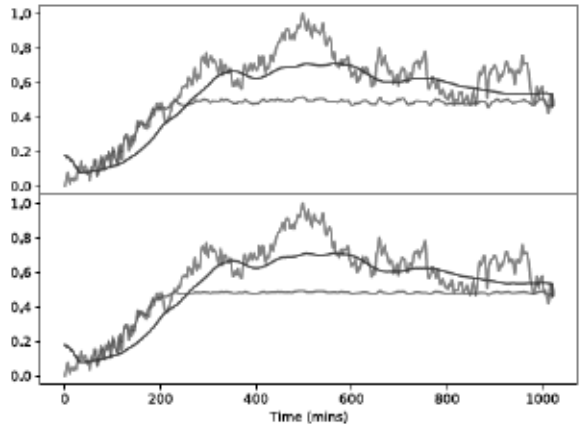


Fig. 9. JPL Sensor 13: Original Signal (in red) and the predicted drift using a MLP predictor (in green), with $\epsilon = 0.1$(top), and 0.7(bottom), compared to the drift estimate of the HT based TCNN (in blue).

training because the TCNN is deep with many layers, which can "store" and "learn" the large training set of possible slowly varying signals.

V. Conclusion

In this paper, we introduced a novel deep learning-based framework for sensor drift estimation. The proposed TCNN structures has built-in Hadamard and additive transform based layers, which smooths and denoises the input signal because the sensor drift signal is a slowly varying signal. The Hadamard transform is related with the Haar wavelet transform and it can be implemented without performing any real multiplication operations. The Hadamard transform can be implemented using binary operations. Similarly, the proposed additive layers can be implemented using only binary operations.

We experimentally observed that Hadamard transform performs better than the DCT, which approximates the Karhunen-Loeve Transform, in our dataset. The DCT is used in signal, image and data compression and it fits the input very efficiently. However, it overfits the data, which are feature parameters of the previous layer in our case. Since our goal is smoothing and denoising, the Haar wavelet transform based Hadamard transform is more suitable compared to DCT. Other wavelet transforms can be also used in this problem but they are not as computationally efficient as the Hadamard transform.

Acknowledgment

The authors, D. Badawi and A. E. Cetin, additionally thank NVIDIA for an equipment grant.

References

[1] D. Y. Chen and P. K. Chan, "An intelligent ISFET sensory system with temperature and drift compensation for long-term monitoring," *IEEE Sensors J.*, vol. 8, no. 12, pp. 1948–1959, Dec. 2008.

[2] K.-M. Chang, C.-T. Chang, K.-Y. Chao, and C.-H. Lin, "A novel pH-dependent drift improvement method for zirconium dioxide gated pH-ion sensitive field effect transistors," *Sensors*, vol. 10, no. 5, pp. 4643–4654, May 2010.

[3] L. Bousse, N. F. De Rooij, and P. Bergveld, "Operation of chemically sensitive field-effect sensors as a function of the insulator-electrolyte interface," *IEEE Trans. Electron Devices*, vol. ED-30, no. 10, pp. 1263–1270, Oct. 1983.

[4] C. Z. D. Goh, P. Georgiou, T. G. Constandinou, T. Prodromakis, and C. Toumazou, "A CMOS-based ISFET chemical imager with auto-calibration capability," *IEEE Sensors J.*, vol. 11, no. 12, pp. 3253–3260, Dec. 2011.

[5] B. C. Jacquot, N. Muñoz, D. W. Branch, and E. C. Kan, "Non-faradaic electrochemical detection of protein interactions by integrated neuromorphic CMOS sensors," *Biosensors Bioelectron.*, vol. 23, no. 10, pp. 1503–1511, May 2008.

[6] A. Vergara, S. Vembu, T. Ayhan, M. A. Ryan, M. L. Homer, and R. Huerta, "Chemical gas sensor drift compensation using classifier ensembles," in *Discrete Wavelet Transforms: Biomedical Applications*. Sep. 2011, p. 177.

[7] A. Rudnitskaya, "Calibration update and drift correction for electronic noses and tongues," *Frontiers Chem.*, vol. 6, p. 433, Sep. 2018.

[8] L. Zhang, Y. Liu, Z. He, J. Liu, P. Deng, and X. Zhou, "Anti-drift in E-nose: A subspace projection approach with drift reduction," *Sens. Actuators B, Chem.*, vol. 253, pp. 407–417, Dec. 2017.

[9] T. Liu, D. Li, and J. Chen, "An active method of online drift-calibration-sample formation for an electronic nose," *Measurement*, vol. 171, Feb. 2021, Art. no. 108748.

[10] Y. Tao, C. Li, Z. Liang, H. Yang, and J. Xu, "Wasserstein distance learns domain invariant feature representations for drift compensation of E-nose," *Sensors*, vol. 19, no. 17, p. 3703, Aug. 2019.

[11] D. Badawi, T. Ayhan, S. Ozev, C. Yang, A. Orailoglu, and A. E. Cetin, "Detecting gas vapor leaks using uncalibrated sensors," *IEEE Access*, vol. 7, pp. 155701–155710, 2019.

[12] D. Huang and H. Leung, "Reconstruction of drifting sensor responses based on Papoulis–Gerchberg method," *IEEE Sensors J.*, vol. 9, no. 5, pp. 595–604, May 2009.

[13] D. Kumar, S. Rajasegarar, and M. Palaniswami, "Automatic sensor drift detection and correction using spatial kriging and Kalman filtering," in *Proc. IEEE Int. Conf. Distrib. Comput. Sensor Syst.*, May 2013, pp. 183–190.

[14] D. Kumar, S. Rajasegarar, and M. Palaniswami, "Geospatial estimation-based auto drift correction in wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 11, no. 3, pp. 1–39, May 2015.

[15] S. Ma, J. Li, H. Hao, and S. Jiang, "Structural response recovery based on improved multi-scale principal component analysis considering sensor performance degradation," *Adv. Structural Eng.*, vol. 21, no. 2, pp. 241–255, Jan. 2018.

[16] C. Distante, M. Leo, and K. C. Persaud, "Wavelet transform for electronic nose signal analysis," *Discrete Wavelet Transforms, Biomed. Appl.*, p. 177, Sep. 2011.

[17] L. Zhang and X. Peng, "Time series estimation of gas sensor baseline drift using ARMA and Kalman based models," *Sensor Rev.*, vol. 36, no. 1, pp. 34–39, Jan. 2016.

[18] T. Ergen, A. H. Mirza, and S. S. Kozat, "Energy-efficient LSTM networks for online learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 3114–3126, Aug. 2020.

[19] P. Peng, X. Zhao, X. Pan, and W. Ye, "Gas classification using deep convolutional neural networks," *Sensors*, vol. 18, no. 2, p. 157, Jan. 2018.

[20] D. Badawi, H. Pan, S. C. Cetin, and A. E. Çetin, "Computationally efficient spatio-temporal dynamic texture recognition for volatile organic compound (VOC) leakage detection in industrial plants," *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 4, pp. 676–687, May 2020.

[21] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1803.01271*. [Online]. Available: http://arxiv.org/abs/1803.01271

[22] L. S. Mokatren, A. E. Cetin, and R. Ansari, "Deep layered LMS predictor," 2019, *arXiv:1905.04596*. [Online]. Available: http://arxiv.org/abs/1905.04596

[23] A. E. Cetin, O. N. Gerek, and S. Ulukus, "Block wavelet transforms for image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, no. 6, pp. 433–435, Dec. 1993.

[24] F. Karabacak, U. Obahiagbon, U. Ogras, S. Ozev, and J. B. Christen, "Making unreliable chem-FET sensors smart via soft calibration," in *Proc. 17th Int. Symp. Qual. Electron. Design (ISQED)*, Mar. 2016, pp. 456–461.

[25] A. Papoulis, "A new algorithm in spectral analysis and band-limited extrapolation," *IEEE Trans. Circuits Syst.*, vol. CS-22, no. 9, pp. 735–742, Sep. 1975.

[26] P. L. Combettes, "The foundations of set theoretic estimation," *Proc. IEEE*, vol. 81, no. 2, pp. 182–208, Feb. 1993.

[27] A. E. Cetin, O. N. Gerek, and Y. Yardimci, "Equiripple FIR filter design by the FFT algorithm," *IEEE Signal Process. Mag.*, vol. 14, no. 2, pp. 60–64, Mar. 1997.

[28] A. Afrasiyabi, D. Badawi, B. Nasir, O. Yildi, F. T. Y. Vural, and A. E. Cetin, "Non-Euclidean vector product for neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 6862–6866.

[29] H. Tuna, I. Onaran, and A. E. Cetin, "Image description using a multiplier-less operator," *IEEE Signal Process. Lett.*, vol. 16, no. 9, pp. 751–753, Sep. 2009.

[30] H. Krim, D. Tucker, S. Mallat, and D. Donoho, "On denoising and best signal representation," *IEEE Trans. Inf. Theory*, vol. 45, no. 7, pp. 2225–2238, Nov. 1999.

[31] A. E. Cetin and M. Tofighi, "Projection-based wavelet denoising [lecture notes]," *IEEE Signal Process. Mag.*, vol. 32, no. 5, pp. 120–124, Sep. 2015.

[32] H. Pan *et al.*, "Additive neural network for forest fire detection," *SIViP*, vol. 14, pp. 675–682, 2020, doi: 10.1007/s11760-019-01600-7.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[34] C. R. Vogel and M. E. Oman, "Iterative methods for total variation denoising," *SIAM J. Sci. Comput.*, vol. 17, no. 1, pp. 227–238, Jan. 1996.

[35] M. Tofighi, O. Yorulmaz, K. Köse, D. C. Yıldırım, R. Çetin-Atalay, and A. E. Cetin, "Phase and TV based convex sets for blind deconvolution of microscopic images," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 1, pp. 81–91, Feb. 2016.

[36] H. Zhou, M. Homer, A. Shevade, and M. Ryan, "Nonlinear least-squares based method for identifying and quantifying single and mixed contaminants in air with an electronic nose," *Sensors*, vol. 6, no. 1, pp. 1–18, Dec. 2005.

[37] *Ammonia Gas Sensor (Model MQ137)*, Zhengzhou Winsen Electron. Technol. Co., Ltd., Zhengzhou, China, 2015.

[38] L. Carmel, S. Levy, D. Lancet, and D. Harel, "A feature extraction method for chemical sensors in electronic noses," *Sens. Actuators B, Chem.*, vol. 93, nos. 1–3, pp. 67–76, Aug. 2003.

[39] C. Yang, P. Cronin, A. Agambayev, S. Ozev, A. E. Cetin, and A. Orailoglu, "A crowd-based explosive detection system with two-level feedback sensor calibration," in *Proc. 39th Int. Conf. Comput.-Aided Design*, Nov. 2020, pp. 1–9.

[40] L. Zhang, F. Tian, S. Liu, L. Dang, X. Peng, and X. Yin, "Chaotic time series prediction of E-nose sensor drift in embedded phase space," *Sens. Actuators B, Chem.*, vol. 182, pp. 71–79, Jun. 2013.

Diaa Badawi received the M.Sc. degree in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2016. He is currently pursuing the Ph.D. degree in electrical engineering with the University of Illinois at Chicago. His research interests include intelligent monitoring systems, energy-efficient implementations of neural networks, and machine learning algorithms.

Agamyrat Agambayev was born in Mary, Turkmenistan, in 1992. He received the B.S. degree in physics with double majoring in electrical and electronics engineering and the M.S. degree in biomedical engineering from Fatih University, Istanbul, Turkey, in 2012 and 2014, respectively. He is currently pursuing the Ph.D. degree in electrical engineering with the King Abdullah University of Science and Technology, Thuwal, Saudi Arabia. His research interests include modeling, designing, application of the fractional-order circuit elements, and nanocomposites for energy storage systems.

Sule Ozev received the Ph.D. degree in computer science and engineering from the University of California, San Diego, in 2002. She is currently an Associate Professor with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe. Her research interests include self-test and self-calibration for wireless transceivers, analysis and mitigation of process variations for mixed signal and digital circuits, fault-tolerant and reconfigurable heterogeneous systems, and mixed signal circuit testing.

A. Enis Cetin (Fellow, IEEE) received the B.Sc., M.S.E., and Ph.D. degrees in systems engineering from the Moore School of Electrical Engineering, University of Pennsylvania. He studied electrical engineering with the Orta Dogu Teknik Universitesi (METU). From 1987 to 1989, he was an Assistant Professor in electrical engineering with the University of Toronto, Canada. He is on leave from Bilkent University, Ankara, Turkey. From 2016 to 2017, he was a Visiting Professor with the UCSD. He is currently a Research Professor with the University of Illinois at Chicago. He is one of the founders of GrandEye that produces omnidirectonal cameras.