A Deep Learning Approach to Sensor Fusion Inference at the Edge

T. Becnel and P-E. Gaillardon
University of Utah, Salt Lake City, U.S.
{thomas.becnel, pierre-emmanuel.gaillardon}@utah.edu

***Abstract*— The advent of large scale urban sensor networks has enabled a paradigm shift of how we collect and interpret data. By equipping these sensor nodes with emerging low-power hardware accelerators, they become powerful edge devices, capable of locally inferring latent features and trends from their fused multivariate data. Unfortunately, traditional inference techniques are not well suited for operation in edge devices, or simply fail to capture many statistical aspects of these low-cost sensors. As a result, these methods struggle to accurately model nonlinear events. In this work, we propose a deep learning methodology that is able to infer unseen data by learning complex trends and the distribution of the fused time-series inputs. This novel hybrid architecture combines a multivariate *Long Short-Term Memory* (LSTM) branch and two convolutional branches to extract time-series trends as well as short-term features. By normalizing each input vector, we are able to magnify features and better distinguish trends between series. As a demonstration of the broad applicability of this technique, we use data from a currently deployed pollution monitoring network of low-cost sensors to infer hourly ozone concentrations at the device level. Results indicate that our technique greatly outperforms traditional linear regression techniques by 6× as well as state-of-the-art multivariate time-series techniques by 1.4× in mean squared error. Remarkably, we also show that inferred quantities can achieve lower variability than the primary sensors which produce the input data.**

I. Introduction

Constructing healthy and sustainable urban societies requires efficient environmental monitoring and rapid interpretation of the data. The proliferation of large-scale Wireless Sensor Networks (WSN) in recent years has greatly contributed to a vast procurement of data spanning many applications, including water quality, pollution, automotive traffic, and distributed power systems [1]–[4]. However, the large influx of environmental data sampled by WSNs brings its own challenges. Where data is traditionally transferred to a centralized cloud for processing, the large amount of data collected by hundreds or thousands of network nodes can make this unfeasible due to network bandwidth or processing limitations [5]. Thanks to the simultaneous advancement of powerful edge processors and hardware accelerators, a great deal of computation is moved away from the cloud and into the WSN nodes themselves [6].

One type of WSN analysis that transfers well to the edge computing paradigm is time series forecasting. A single WSN node is typically comprised of several low-cost environmental sensors that sample independently and fuse their temporal measurements. Due to the loose coupling and periodicity of many environmental events, there exist rich unobserved phenomena that can be extracted directly from the sampled data. For examples, authors in [7] use a network of sparse and intermittent mobile pollution monitors to create a dense spatiotemporal model of airborne particulate matter concentrations in an urban area. In [8], authors use a number of environmental factors, such as wind speed, direction, temperature, and atmospheric pressure to predict short-term wind vectors.

Traditional forecasting techniques generally make use of *AutoRegressive* (AR) methodologies because of their ability to use to past observations to make estimates, with one of the most popular being the *Nonlinear Autoregressive Network with Exogenous Inputs* (NARX) model [9]. However, in recent years, hybrid *Deep Neural Network* (DNN) models have proven to be effective tools for time series tasks, and often outperform AR techniques. In [10], the authors propose a hybrid framework to forecast loading on power grids. In [11], the authors propose a similar network topology for time series classification and show state-of-the-art performance. Despite the predominate success of the aforementioned techniques, they fail to address the inherent statistical properties of datasets generated from low-cost environmental sensors, such as the posterior dependencies between these exogenous inputs.

In this work, we present DEENET (***DEE****p inference using* ***N****ormalized* ***E****xogenous* ***T****ime series*), a lightweight, deep learning inference technique specifically for deployment on edge devices. DEENET is targeted at estimating latent time series by analyzing the nonlinear response of fused multi-sensor inputs. Specifically, we employ a novel multi-branch DNN architecture comprised of a *Long Short-Term Memory* (LSTM) branch and two multi-layer *Fully Convolutional* (FC) branches. A major contribution of this work is a lightweight normalization stage that rescales the input features prior to input to the FC branch. By providing both raw and normalized feature sets in parallel, our network is able to model general relationships between features as well as extract dynamics that would otherwise not trigger neural responses because of their subdued scale. To demonstrate our framework, we task DEENET with hourly ozone inference using strictly exogenous input from an active network of low-cost pollution monitors. We show $1.4\times$ mean squared error improvement over current state-of-the-art DNN techniques. We also show the remarkable effect that inferred quantities show lower inter-model variability than the input driving sensors, which addresses the ability of the network to perform non-linear calibration of the inputs. Specifically, our contributions in this work are:

- A novel 3-branch architecture, coined DEENET, for fused

978-3-9819263-5-4/DATE21/©2021 EDAA

multi-sensor time-series inference of latent time series from strictly exogenous inputs.
- A back-propagation based training technique that teaches the network important distributions between multi-source driving series.
- A real-world case study of urban ozone inference using a deployed network of low-cost pollution monitors.

The rest of the paper is organized as follows: Section II introduces the background behind our proposed algorithm. Section III describes our algorithm in detail. Section IV provides an experimental evaluation of the technique using data from a low-cost pollution monitoring network deployed in an urban metropolitan area. Section V provides concluding remarks for the work described here.

II. Related Work

This section summarizes related work on time series forecasting and classification tasks, as well as the foundational components used in the proposed methodology.

A. Time Series Forecasting and Classification

Traditional forecasting techniques are generally based on *AutoRegressive* (AR) models, due to their ability to incorporate past values of the variable of interest into future prediction. The *Autoregressive Integrated Moving Average* (ARIMA) model has seen extensive use on univariate time series prediction since it's inception [12]. A more generalized successor is the *Nonlinear Autoregressive Network with Exogenous Inputs* (NARX) model, which uses multiple exogenous driving series, as well as previous predictions from the target series, to make predictions at the next time step. The success of the NARX model has led to a large pool of research to improve the technique. In [13], authors present an auto-regressive network to predict short-term thermal behavior in buildings which exhibits state-of-the-art performance.

In recent years, hybrid deep neural network (DNN) models have proven to be an effective tool in time series inference and classification tasks, and often outperform vanilla AR or RNN techniques. In [10], the authors propose a parallel LSTM-CNN framework for the task of short-term load forecasting of power grids. While the LSTM branch learns long-term trends in the driving series, the convolutional block learns interesting short-term events that help distinguish non-periodic phenomena. In [11], authors extend the idea of a parallel LSTM-CNN architecture and propose a novel multivariate time series classification scheme. By including Squeeze-and-Excite (SE) blocks in the convolutional branch, feature maps are able to analyse interdependencies and adaptively alter their scales. The authors show this work achieves state-of-the-art performance on a large majority of classification tasks.

B. Recurrent Neural Networks and Long Short-Term Memory

Recurrent Neural Networks (RNN) are a class of neural networks capable of learning temporal patterns from the input data by maintaining a hidden state vector which is updated during the back-propagation phase of training [14].

However, RNNs are quite prone to the vanishing gradient problem [15], which lends to their inability to learn long sequences and thus vanilla RNNs are not typically used in practice. The successor to RNNs, *Long Short-Term Memory* (LSTM) networks, combat this problem by employing gating mechanisms on the internal state vector. During back-propagation, the smart gating mechanism adaptively decides what information is important, and error is either accumulated or allowed to pass through the cell. In this way, LSTM networks are able to long input sequences, which makes them ideal candidates for time series inference tasks. In an LSTM architecture, the hidden state is replaced by a cell consisting of gating functions and the internal state. For a detailed description of LSTM networks, we refer the reader to [16].

C. Convolutional Neural Networks

Convolutional operations are a well-established technique to extract features from time series data [17], and meany research works have successfully employed Convolutional Neural Networks (CNN) to solve time-series tasks [18]–[20]. Each convolutional block is typically comprised of a set of 1-dimensional convolutional kernels, followed by ReLU activation and batch normalization. The input matrix, or feature map output from a previous layer, is convolved with several convolutional kernels.

Specifically, following the notations derived in [21], start with an input time series feature vector of length F_0: $X_t \in \mathcal{R}^{F_0}$, for time step $0 < t \leq T$. Note that in our application, T is constant across all features and therefore does not require a padding mechanism. Given L convolutional layers in the convolutional block, we apply a set of 1-dimensional convolutional kernels that capture dynamics of the input. The filters for each layer are parameterized by the tensor $W^l \in \mathcal{R}^{F_l \times d \times F_{l-1}}$ and biases $b^l \in \mathcal{R}^{F_l}$, where d is the filter duration and $l \in \{l, \ldots, L\}$ is the layer index. For the l-th layer of the convolutional block, the i-th filter activation $\hat{E}_t^l \in \mathcal{R}^{F_l}$ is function of the activation matrix $E^{l-1} \in \mathcal{R}^{F_l \times T_{l-1}}$ from the previous layer

$$\hat{E}_{i,t}^l = ReLU\left(b_i^l + \sum_{t'=1}^{d} \left\langle W_{i,t',\cdot}^l, E_{\cdot,t+d-t'}^{l-1} \right\rangle\right) \qquad (1)$$

Convolutional block outputs are immediately succeeded by a max pooling layer. As introduced in [22], max pooling is a nonlinear down-sampling technique that takes the maximum value over a given window size, across a number of pooling regions Q. Effectively, the resolution of the learned temporal features is reduced to prevent overfitting.

The next section will cover the proposed architecture, and make use of the definitions outlined in this section.

III. DEENET - A Time Series Inference Framework

In this section, we introduce and formally define DEENET, a hybrid DNN methodology which aims to infer latent features from fused sensor time series input.

A. Methodology Rationale

We begin by formulating the inference task as a type of multivariate time-series forecasting. However, unlike classical

forecasting techniques, which generally use past values of the observed phenomena of interest to perform one-step-ahead predictions, our targeted application is the inference of latent features at the current time step. Therefore, we must instead rely solely on extracting trends and relationships between the exogenous inputs to estimate the target. To this end, we propose a novel 3-branch architecture that is particularly performant at inference from loosely periodic, multivariate time series data acquired by sensors. By providing training data from multiple sources of a given feature (such as multiple identical sensors), the proposed model learns the *a priori* distribution of sensor calibration parameters. The trained model can then be ported to any arbitrary network node (not included in the training phase) containing the same platform of sensors and perform high quality inference of the desired target.

B. Network Input and Normalized Preprocessing Stage

A novel step in the proposed framework is a small preprocessing stage of normalization along the temporal axis of the multivariate input tensor. Rationale for this approach can be demonstrated by way of example. Figure 1a shows one year of hourly samples from a temperature sensor and a *Carbon Monoxide* (CO) sensor deployed in a dense urban, outdoor environment. Each feature vector has gone through an initial normalization step to fit the datasets in the range $[0, +1]$ to aid in visualization. Temperature measurements display an annual trend with the changing seasons. CO concentrations do not correlate well to temperature, and while there is a large disparity in the scale of the observations between the two features in July, 2019, the scales converge in December of the following year. Figure 1b shows a 100-hour window of the two vectors, taken from Figure 1a, as well as vectors that have been normalized to this window. It is then easy to discern the role that normalization plays in amplifying short temporal events. When considering a training a FC network on the fused tensor input displayed in Figure 1b, the large disparity in scales will result in convolutional filters that learn general trends, whereas filters trained on the normalized feature sets will tend to learn the relationship between feature dynamics. Indeed, whereas a FC block operating on the raw inputs will tend to learn magnitudes and trends between features, the same FC block operating on the normalized inputs will learn to pick out relational temporal dynamics between features.

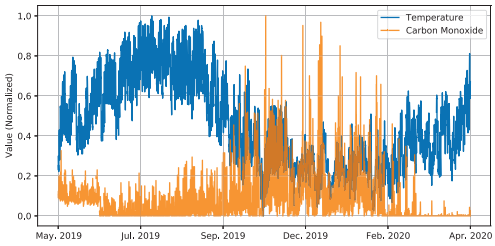
The normalization step is formally defined as

$$\hat{X}_{m,n} = \frac{X_{m,n} - \min(X_m)}{\max(X_m) - \min(X_m)} \tag{2}$$

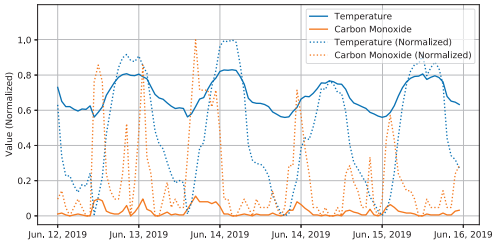
where M is the feature vector and N is the subregion time vector.

C. Proposed Architecture

The proposed model, shown in Figure 2 is is a novel 3-branch architecture comprised of a Long Short-Term Memory (LSTM) block stacked with two Fully Convolutional (FC) blocks. Network input tensors of shape (N, M) are bifurcated and passed through a simple stage of feature-wise normalization. Raw



(a) Annual collection of hourly observations for a temperature sensor and Carbon Monoxide sensor.



(b) 100 consecutive hour subregion from 1a

Fig. 1: Time series data from a temperature sensor and Carbon Monoxide sensor located in a dense urban, outdoor location. Large disparities between the two feature vectors can be removed in each subregion by re-normalizing in the context of the subregion.

inputs are fed into the LSTM branch and FC branch, while the normalized inputs are fed into the second FC branch. In this way our model is able to learn short-term dynamics between input features via the normalized-input FC branch, as well long-term trends and periodic behavior from the LSTM branch and raw-input FC branch.

The LSTM branch is comprised of a vanilla LSTM layer with an internal state vector dimensionality of 8. The LSTM layer is followed by a dropout layer with a dropout rate of 0.8 to prevent overfitting of the training data [23].

Following the work proposed in [24], each FC block contains three stacked convolutional layers with $\{128, 256, 128\}$ filters in each layer. Each FC layer is followed by batch normalization and ReLU activation. Convolutional operations are performed by 1-D kernels with sizes $\{8, 5, 3\}$, respective to each FC layer. Following the approach taken in [11], we append Squeeze-and-Excite (SE) blocks to the end of the first two ReLU activations. SE blocks were introduced by Hu *et al.* in [25] as a technique to model interdependencies between channels and adaptively recalibrate features. The SE block has the effect of helping the network generalize between inputs and prevent overfitting. To this end, we found the inclusion of the SE blocks in our network improved accuracy at a marginal cost of a 2-10% increase in the network size. A diagram of the full convolutional block is depicted in Figure 2a.

We note that through experimental evaluation we found that the improved accuracy of adding a second LSTM branch trained on normalized inputs is marginal and therefore unjustified for the network size increase. For evaluation purposes the proposed model was generated using the Keras python library [26] with

Fig. 2: The proposed architecture. The Fully Convolution (FC) block is shown in (a), and the complete network is shown in (b).

Tensorflow [27] as the backend.

IV. Experimental Results

The proposed architecture was trained and evaluated using a publicly available dataset from an actively deployed pollution monitoring network [2]. The model is evaluated and compared against vanilla *Multiple Linear Regression* (MLR) *Multi-Layer Perceptron* (MLP) models to form a baseline, as well as a NARX model and MLSTM-FCN, a state-of-the-art multivariate time series classification technique adapted for for the task of latent time series inference [11].

In this section we first describe the sensor network from which we procured the experimental dataset, then we cover preparing the dataset for the training and inference regimen, then training, the comparison models, the evaluation metrics, and finally a discussion of the experimental results.

A. Environmental Pollution Dataset Description

The experiment performed in this work uses a publicly available dataset known as the AirU Pollution Monitoring Network [2]. AirU is a network of WiFi-enabled IoT edge devices, herein referred to as network *nodes*, which monitor airborne pollutants. Each node contains temperature, humidity, particulate matter (PM), oxidizing gas species (OX) (primarily NO_2), and reducing gas species (RED) (primarily CO) sensors. In this work we use 8 nodes collocated with two reference instruments, described below. For more information regarding the AirU network, we refer the reader to [2].

Several high-grade reference instruments maintained by the United States Department of Air Quality (DAQ) are deployed throughout the AirU region. These DAQ sites produce hourly observations for temperature, relative humidity, barometric pressure, $PM_{2.5}$, nitric oxide (NO), NO_2, CO, and ozone (O_3). The measurements are considered extremely accurate, and thus are referred to as the "ground-truth" data. At two DAQ sites, dubbed Hawthorne (HW) and Rose Park (RP), there are 8 collocated AirU nodes (4 at each site), which act as calibration nodes for the entirety of the network. The dataset used is this experiment was generated by these 8 nodes, with the ozone targets taken from the DAQ ozone reference instruments.

To help describe the research interest in the task of ozone inference, we refer to Figure 3, which displays the correlation between collocated AirU sensors and the corresponding DAQ reference. In lieu of space we only show sensors at the DAQ Hawthorne site, but the Rose Park site contains very similar results. The figure shows that the AirU temperature, relative humidity, and $PM_{2.5}$ sensors all correlate quite well between one another, as well as the DAQ reference instruments. The gas sensors, however, exhibit very low correlations with the DAQ NO_2, CO, and O_3 reference instruments, likely due to the poor selectivity and nonlinear response of the gas sensors, as discussed in [28]. However, the relatively high correlations between the low-cost gas sensors tell us that they are functioning properly. We then note that ozone concentrations do not respond linearly to the gas sensors on the AirU platform, and that a nonlinear inference scheme will likely outperform a simple linear regression. Indeed, we show that MLR is not suitable to model the complex relationship between ozone and the observed parameters.

Data from the DAQ sites is sampled at 1-hour intervals. To match this interval, minute-by-minute data from the collocated AirU nodes is averaged into matching 1-hour intervals. Ozone measurements obtained by the DAQ reference instruments are concatenated to each AirU node dataset and matched according to hourly timestamps.

B. Training and Validation

Training data is provided by 7 of the 8 collocated nodes, and the 8th node is defined as the holdout. In this way, the statistics of the environmental sensors of the node are completely unknown during validation and testing. The nonlinear mapping of the model structure allows the model to make probabilistic assumptions of the various sensor distributions in order to make the most accurate ozone concentration inference. Each feature is scaled to the range $[0, +1]$ across the time frame. For the 7 training sensors, data is sectioned into individual samples via a temporal sliding window with width 100, such that each sample is a 2-D tensor $X_i \in \mathcal{R}^{M \times N}$, where M is the 7 input features (temperature, humidity, $PM_{1.0}$, $PM_{2.5}$, $PM_{10.0}$, RED, and OX values), and N is 100 consecutive hourly time steps, where $\{-100 < n \leq 0\}$. Each sample is assigned a corresponding target measurement, which is the collocated DAQ ozone concentration at $\{n = 0\}$ (current time), relative to the sample time frame.

Samples are then added indiscriminately to the training pool and randomly shuffled prior to each epoch. Similarly, samples for the holdout node are added to the testing pool. In total there are roughly 55,000 training samples and 8,000 test samples for each of the 8 models. Network models are trained across 300 epochs with a batch size of 128. We define mean-squared error

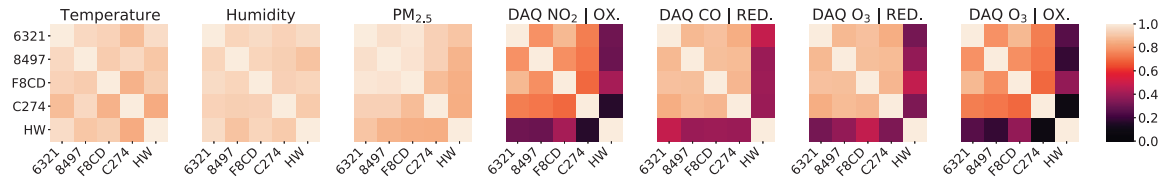


Fig. 3: R^2 values of each environmental sensor on the AirU Platform and corresponding DAQ reference instrument. The labels `6321`, `8497`, `F8CD`, `C274` are the designators of the four AirU nodes located at the DAQ HW site.

as the loss metric and use the Adam optimization algorithm as the back-propagation technique with a static learning rate of 0.01.

Similar to a cross-validation approach, we train 8 separate models, with each model targeting one of the 8 AirU nodes as the holdout. In this way we can demonstrate robust and generalized results across multiple nodes, as well as compare the variability between models.

C. Comparison Models and Evaluation Metrics

We compare our model to two state-of-the-art inference techniques: NARX [9], and MLSTM-FCN, a state-of-the-art multivariate time-series classification neural network architecture which we have adapted for real-valued inference [11]. We also establish a baseline by training Multiple Linear Regression (MLR) [29] and a vanilla Multi-layer Perceptron (MLP) with a single hidden layer [30].

The described models are evaluated using *Mean Squared Error* (MSE), *Median Absolute Deviation* (MAD), and *Coefficient of Determination* (R^2). We also use *Mean Error* (ME) to define inter-sensor variance. These metrics are well-regarded and used extensively throughout the field of time series modelling. For descriptions and detailed analysis of these metrics we refer the reader to [31].

D. Inference Results and Statistics

Table I shows the experimental results of the described models on the inference of ozone concentrations from a dataset of exogenous sensor data. We begin by noting that MLR achieves the lowest scores across all evaluation metrics, which is unsurprising as the dataset exhibits nonlinear behavior.. Similarly, a simple MLP does not contain the required complexity to sufficiently learn and model the numerous temporal and inter-feature dependencies and relationships. We use MLR and MLP to establish a baseline across the evaluation metrics. The NARX model shows surprisingly poor error and deviation results, but outperforms the baseline in correlation. This behavior is expected as the NARX model is able to incorporate past dependencies. On this task DEENET displays state-of-the-art performance, beating MLSTM-FCN on average by $1.2\times$, $1.41\times$, $1.02\times$ across MAD, MSE, and R^2, respectively. Our technique also outperforms MLSTM-FCN on every holdout node across the evaluation metrics, including up to $2.2\times$ improvement in MSE.

DEENET also displays the remarkable ability to reduce variance across models when compared to the variance of the input sensors types. Figure 4 shows the variance for the input features across the 8 AirU nodes, where variance is calculated from the *Mean Error* (ME) for each sensor on each node. The inferred ozone measurements exhibit $25.4\times$, $15.1\times$, and $13.7\times$ lower model-to-model variance than RED, humidity, and OX sensors, respectively. Effectively this demonstrates that the network has the ability to perform internal calibration on the input features, and we suspect that as more nodes are added to the training pool, the variance of the ozone inference will decrease further.

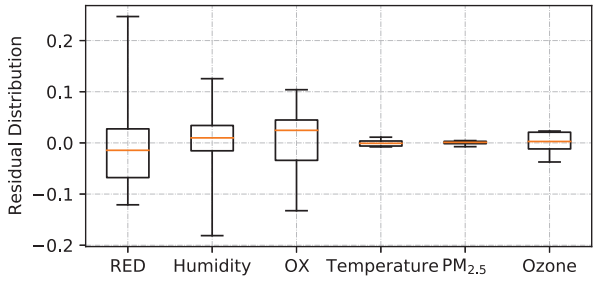


Fig. 4: Inter-sensor variability for each of the environmental sensors used in this study and inferred ozone concentrations. The underlying data for each boxplot is generated from the mean error (ME) for each of the 8 sensors. The variance for each distribution is $\{1.3e^{-2},\ 7.7e^{-3},\ 7.0e^{-3},\ 5.2e^{-5},\ 1.7e^{-5},\ 5.1e^{-4}\}$ for RED, Humidity, OX, Temperature, $PM_{2.5}$, and Ozone, respectively.

Finally we conclude the discussion by investigating the feasibility of deploying DEENET to low power edge devices, such as the Google Edge TPU [32], NVIDIA Jetson Nano [33], or Intel Movidius [34], though for the sake of brevity we choose to only assess the Google Edge TPU. In [6], the authors benchmark the Edge TPU on MobileNet V1 [35], which contains 4.2 million parameters. By comparison, our network contains 756,000 parameters (82% fewer parameters than MobileNet V1). The authors report 23 mW/inference and an average inference time of 27 ms. By extrapolating the network size reduction to the reported power, we can expect a power cost of roughly 1.2 mW/inference, which is well within the power budget for many edge networks.

V. Conclusion

In this work we address the task of multivariate time-series inference of latent variables with the proposal of a novel multi-branch DNN architecture. By providing both raw and

TABLE I: Performance comparison of the proposed model

Holdout Node	MLR			MLP			NARX			MLSTM-FCN			DEENET (This Work)		
	MAD	MSE	R^2	MAD	MSE	R^2	MAD	MSE	R^2	MAD	MSE	R^2	MAD	MSE	R^2
075B	0.064	0.0167	0.546	0.068	0.0162	0.567	0.131	0.0241	0.871	0.024	0.0015	0.960	0.021	0.0011	0.971
6321	0.065	0.0102	0.731	0.066	0.0094	0.752	0.044	0.0052	0.865	0.037	0.0019	0.950	0.032	0.0017	0.954
8497	0.070	0.0063	0.828	0.068	0.0072	0.810	0.059	0.0070	0.851	0.039	0.0022	0.940	0.034	0.0015	0.960
F8CD	0.068	0.0066	0.826	0.063	0.0071	0.809	0.130	0.0286	0.771	0.029	0.0015	0.960	0.027	0.0014	0.964
C274	0.084	0.0079	0.782	0.083	0.0075	0.801	0.054	0.0081	0.880	0.037	0.0021	0.943	0.036	0.0020	0.945
EA5D	0.080	0.0162	0.545	0.082	0.0144	0.612	0.050	0.0064	0.858	0.030	0.0016	0.956	0.027	0.0014	0.962
FD74	0.071	0.0180	0.509	0.072	0.0161	0.567	0.058	0.0077	0.834	0.044	0.0031	0.916	0.038	0.0021	0.944
8AC9	0.080	0.0081	0.766	0.079	0.0075	0.797	0.107	0.0158	0.885	0.048	0.0057	0.834	0.029	0.0026	0.925
Mean	0.073	0.0113	0.691	0.073	0.0107	0.714	0.079	0.0129	0.852	0.036	0.0024	0.933	0.030	0.0017	0.953

normalized exogenous inputs, our network is able to capture trends between input features as well as short temporal dynamics that are normally washed out due to the input feature scales. The inputs are fed into a novel 3-branch network architecture comprised of a LSTM block and two parallel Fully Convolutional blocks. When employed with the task of ozone inference using inputs from a network of low-cost environmental sensors and no observed ozone measurements, our network outperforms current DNN techniques by $1.4\times$. Our model also displays a remarkable decrease in variance between models when compared to the inter-sensor variance of the supplementary driving sensors, which demonstrates the ability of our technique to act as a form of nonlinear calibration of the inputs.

ACKNOWLEDGMENT

This work was supported by DARPA under Grant No. D19AP00028. This work does not necessarily reflect the views of the United States government. Pierre-Emmanuel Gaillardon has financial interest in the company *Tetrad: Sensor Network Solutions, LLC.*, which manufactures air quality sensing solutions and provides engineering services.

REFERENCES

[1] Y. Chen et al. Water quality monitoring in smart city: A pilot project. *Automation in Construction*, 89:307–316, 2018.
[2] T. Becnel et al. A Distributed Low-Cost Pollution Monitoring Platform. *IEEE Internet of Things Journal*, 6(6), 2019.
[3] S. Jain et al. Smart City: Traffic Management System Using Smart Sensor Network. *Journal of Physics: Conference Series*, 1362:012129, 2019.
[4] S. Ryu et al. Deep Neural Network Based Demand Side Short Term Load Forecasting. *Energies*, 10(1):3, 2017.
[5] L. Qu et al. Daily long-term traffic flow forecasting based on a deep neural network. *Expert Systems with Applications*, 121:304–312, 2019.
[6] A. Reuther et al. Survey and Benchmarking of Machine Learning Accelerators. 2019.
[7] R. Ma et al. Generative Model Based Fine-Grained Air Pollution Inference for Mobile Sensing Systems. SenSys '18, pages 426–427, 2018.
[8] M. Dalto et al. Deep neural networks for ultra-short-term wind forecasting. In *2015 IEEE International Conference on Industrial Technology (ICIT)*, pages 1657–1663, 2015.
[9] K. S. Narendra et al. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.
[10] C. Tian et al. A Deep Neural Network Model for Short-Term Load Forecast Based on Long Short-Term Memory Network and Convolutional Neural Network. *Energies*, 11(12):3493, 2018.
[11] F. Karim et al. Multivariate LSTM-FCNs for time series classification. *Neural Networks*, 116:237–245, 2019.
[12] S. C. Hillmer et al. An ARIMA-Model-Based Approach to Seasonal Adjustment. *Journal of the American Statistical Association*, 77(377):63–70, 1982.
[13] F. Ferracuti et al. Data-driven models for short-term thermal behaviour prediction in real buildings. *Applied Energy*, 204:1375–1387, 2017.
[14] R. Pascanu et al. How to Construct Deep Recurrent Neural Networks. 2013.
[15] S. Hochreiter et al. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02):107–116, 1998.
[16] A. Graves et al. Supervised Sequence Labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks*, pages 5–13. 2012.
[17] S. Mallat et al. *A Wavelet Tour of Signal Processing*. 1999.
[18] C. Szegedy et al. Going Deeper With Convolutions. pages 1–9, 2015.
[19] Y. Zheng et al. Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks. In *Web-Age Information Management*, volume 8485, pages 298–310. 2014.
[20] Z. Cui et al. Multi-Scale Convolutional Neural Networks for Time Series Classification. 2016.
[21] C. Lea et al. Temporal Convolutional Networks: A Unified Approach to Action Segmentation. 2016.
[22] D. Scherer et al. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In *Artificial Neural Networks – ICANN 2010*, pages 92–101, 2010.
[23] N. Srivastava et al. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
[24] Z. Wang et al. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1578–1585, 2017.
[25] J. Hu et al. Squeeze-and-Excitation Networks. 2017.
[26] Keras: the Python deep learning API.
[27] M. Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. 2016.
[28] T. Sayahi et al. Long-term calibration models to estimate ozone concentrations with a metal oxide sensor. *Environmental Pollution*, 267:115363, 2020.
[29] D.F. Andrews. A Robust Method for Multiple Linear Regression: Technometrics: Vol 16, No 4.
[30] D.W. Ruck et al. The multilayer perceptron as an approximation to a Bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1(4):296–298, 1990.
[31] G. E. P. Box et al. *Time series analysis. Forecasting and control*. 1976.
[32] Google Edge TPU. 2019. Available online: https://cloud.google.com/edge-tpu/.
[33] NVIDIA. Jetson Nano. 2019. Available online: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/.
[34] Hruska. New movidius myriad x vpu packs a custom neuralcompute engine. 2017. Available online: https://www.extremetech.com/computing/254772-new-movidius-myriad-x-vpu-packs-custom-neural-compute-engine.
[35] A. G. Howard et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017.