Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2019.DOI

An End-to-End Trainable Feature Selection-Forecasting Architecture Targeted at the Internet of Things

MERT NAKIP[1], (Student Member, IEEE), KUBİLAY KARAKAYALI[2], CÜNEYT GÜZELİŞ[3], AND VOLKAN RODOPLU[4], (Member, IEEE)

[1]Institute of Theoretical and Applied Informatics, Polish Academy of Sciences (PAN), ul. Baltycka 5, 44100 Gliwice, Poland
[2]ETECube, Izmir Technology Development Zone in Izmir Institute of Technology, 35437, Izmir, Turkey
[3, 4]Department of Electrical and Electronics Engineering, Yaşar University, 35100 , Izmir, Turkey

Corresponding author: Volkan Rodoplu (e-mail: volkan.rodoplu@yasar.edu.tr).

This work has been funded by the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No. 846077, entitled "Quality of Service for the Internet of Things in Smart Cities via Predictive Networks".

ABSTRACT We develop a novel end-to-end trainable feature selection-forecasting (FSF) architecture for predictive networks targeted at the Internet of Things (IoT). In contrast with the existing filter-based, wrapper-based and embedded feature selection methods, our architecture enables the automatic selection of features dynamically based on feature importance score calculation and gamma-gated feature selection units that are trained jointly and end-to-end with the forecaster. We compare the performance of our FSF architecture on the problem of forecasting IoT device traffic against the following existing (feature selection, forecasting) technique pairs: Autocorrelation Function (ACF), Analysis of Variance (ANOVA), Recurrent Feature Elimination (RFE) and Ridge Regression methods for feature selection, and Linear Regression, Multi-Layer Perceptron (MLP), Long Short Term Memory (LSTM), 1 Dimensional Convolutional Neural Network (1D CNN), Autoregressive Integrated Moving Average (ARIMA), and Logistic Regression for forecasting. We show that our FSF architecture achieves either the best or close to the best performance among all of the competing techniques by virtue of its dynamic, automatic feature selection capability. In addition, we demonstrate that both the training time and the execution time of FSF are reasonable for IoT applications. This work represents a milestone for the development of predictive networks for IoT in smart cities of the near future.

INDEX TERMS Forecasting, feature selection, machine learning, neural network, Internet of Things (IoT), predictive network, smart city

I. INTRODUCTION

Smart cities of the near future will be made up of smart buildings, factories, hospitals, transportation services, schools as well as smart homes, all of which will be at the disposal of the digitally-equipped inhabitants of these cities. The Internet of Things (IoT), which refers to a plethora of sensors dispersed in a smart city, is expected to become the main enabler of the services that the smart city will offer. The sensors that make up the IoT infrastructure will report their data via a telecommunication infrastructure network to the cloud that runs Artificial Intelligence (AI) algorithms that interpret and act upon the data reported by the sensors.

Predictive networking [1]–[10] is a new paradigm in which the telecommunication infrastructure network itself uses AI-based algorithms to form predictions of the demand upon the network in order to allocate networking resources in advance. Predictive networks stand in sharp contrast with the traditional reactive networks that merely respond to the current demand. It is expected that the development of accurate forecasting schemes will play a crucial role for realizing predictive networks in smart cities of the near future [11].

The goal of this paper is to develop a general architecture for forecasting that is targeted at forecasting IoT data traffic but which also potentially has applications to many areas beyond IoT. The forecasting architecture that we develop in this paper stands at a point between Deep Learning (DL) techniques at one end and the rest of the Machine Learning (ML) techniques at the other end of the spectrum. A key difference between these two classes of techniques is in regard to "features", namely, aspects of the past data that are singled out as being important for forming accurate forecasts: A DL technique discovers the features itself but typically requires a long past data stream and a high training time. In contrast, an ML technique that does not use DL typically utilizes "feature selection", namely a selection among or a transformation of the past data such that only these features (not the entire past data) are fed as input to the forecaster.

The feature selection-forecasting architecture that we develop in this paper, which we abbreviate as FSF, enables automatic discovery of features in order to minimize the forecasting error while retaining reasonable space and time complexity. The key differences between our FSF architecture and the existing literature are as follows:

- The fact that unimportant features are strictly eliminated in FSF distinguishes our approach from DL techniques.
- The fact that feature discovery is completely automated in FSF contrasts sharply with the existing categories of filter-based, wrapper-based and embedded feature selection techniques [12] in the literature. Our architecture falls in none of these existing categories.
- Our FSF architecture is *not* a simple combination of feature selection and forecasting methods. Instead, it is a general architecture that dynamically selects features based on the computed feature importance scores and performs forecasting based only on the selected features.

An artificial neural network (ANN) is said to be "end-to-end trainable" if all of the parameters of the ANN between the input and output terminals are trainable. We achieve the dynamic, automatic feature selection in our FSF architecture by training both the feature selection and forecasting modules jointly as an end-to-end trainable ANN. In this paper, we demonstrate that our architecture offers a high generalization ability that produces highly accurate forecasts.

We compare the performance of our FSF architecture on the problem of forecasting IoT device traffic against a competitive subset of existing (feature selection, forecasting) pairs and show that our FSF architecture achieves either the best or close to the best performance by virtue of its dynamic, automatic feature selection capability. In addition, we demonstrate that both the training time and the execution time of FSF are reasonable for IoT applications.

The rest of this paper is organized as follows: In Section II, we describe the relationship between this work and the existing literature. In Section III, we describe the detailed design of our FSF architecture. In Section IV, we present the performance comparison of our FSF architecture with the state-of-the-art techniques for forecasting the traffic generation patterns of IoT devices. In Section V, we present our conclusions.

II. RELATIONSHIP TO THE STATE OF THE ART

In this section, we contrast our work with the state of the art in two categories: (1) We contrast our work with DL forecasting techniques as well as statistical models, none of which use feature selection. (2) We state the differences between our work and ML techniques that utilize feature selection. We subdivide this second category into two subcategories: (2.1) non-adaptive (i.e. filter-based) feature selection, and (2.2) adaptive feature selection techniques that utilize either wrapper-based or embedded feature selection. After contrasting our work with each of the above categories for feature selection and forecasting, we describe the relationship of our work to the recent schemes that have appeared in the literature on forecasting the traffic generation patterns of IoT devices, which is the immediate domain of application of our FSF architecture in this paper.

In regard to the first category, in the current forecasting literature, both deep neural networks and statistical models are used for forecasting without feature selection. We shall first address DL techniques and then turn to the comparison with statistical models. The architectural designs of DL techniques, such as LSTM and 1D CNN, are able to perform internal feature extraction in order to minimize the forecasting error: LSTM is used for short-term forecasting of the electric load in [13]. By using 1D CNN, Reference [14] forecasts the daily electricity load. Moreover, a hybrid of 1D CNN and LSTM is used for forecasting Particulate Matter concentration in smart cities in [15]. The results of the works [13]–[15] show that DL techniques are able to achieve relatively high performance for each of these distinct forecasting problems. However, these techniques need data sets with a relatively high number of samples because of their complex internal architectures. In contrast, we shall show that our architecture FSF is able to reach higher performance than these DL techniques on smaller data sets since the internal architecture of FSF is composed of simpler operations than those of the DL techniques.

Now, we address the differences between statistical forecasting models and our work. Statistical models such as ARIMA and Seasonal ARIMA (SARIMA) capture the linear relationship among the future, current, and the past values of a time series, if such an underlying relationship exists in the data. In [16], the authors use ARIMA for forecasting the software clone evolution over time; their results show that ARIMA underperforms with respect to the ANN models due to the nonlinear structure in the data. SARIMA is used for long-term forecasting of the significantly large wave heights in [17]. The models in [16], [17] are able to achieve high performance for data that have a linear underlying structure. In contrast with ARIMA and SARIMA, our FSF architecture is able to capture both linear and nonlinear relationships due to the flexibility inherent in our architecture.

In regard to the second category, we shall first address the difference between our work and the first subcategory, namely non-adaptive (also known as filter-based) feature selection techniques. The articles in this subcategory select the relevant features with respect to feature importance scores. In this category, the following set of articles calculate the feature importance scores based mostly on statistical functions: Reference [18] computes both the ACF and the partial ACF with a correlation threshold in order to perform forecasting on univariate time series data by using ANN forecasters. The authors in [19] propose the method of predominant correlation and use this method to calculate a feature importance score in order to select features with a constant threshold value. In [20], initially, the relevant features are selected empirically based on mutual information; subsequently, the power load of the smart grid is forecast using an ANN.

In the same subcategory of filter-based methods, the following articles calculate the feature importance scores by using ML or fuzzy logic: In [21], the most important features are selected empirically based on the feature importance scores that are calculated by the authors' proposed methodology that is based on a fuzzy curve. In [22], if a connection weight of a perceptron that has been trained to make predictions by using all of the available features is greater than a fixed threshold, a feature that is the input of that connection weight is selected. In the filter-based feature selection methods, typically, a fixed (that is, untrainable) threshold value or a policy is applied on the importance scores in order to select the relevant features. In contrast, our FSF architecture trains a neural network based on *trainable* feature importance scores and a *trainable* value of the threshold.

Now, we turn to the second subcategory of feature selection techniques, namely adaptive feature selection, which is comprised of wrapper-based feature selection and embedded feature selection methods. We shall first contrast our work against wrapper-based feature selection methods for forecasting in the literature. These feature selection methods minimize the forecasting error iteratively. Reference [23] combines ant colony optimization and a genetic algorithm to select features in order to forecast the 24-hours-ahead electricity load using an MLP. The authors in [24] forecast the electricity load by using a multi-output SVR with the firefly algorithm for feature selection. Reference [25] performs wrapper-based feature selection by updating the features according to the summation of the connection weights of a single-layer ANN. In [26], a two-stage wrapper based feature selection algorithm is proposed for forecasting financial time series data. References [27]–[31] perform a filter-based feature selection prior to the iterations of the wrapper-based feature selection. In contrast with all of these past wrapper-based feature selection methods, our FSF architecture selects the important features by minimizing the forecasting error via backpropagation in the place of performing a search for those features.

Within the second subcategory of adaptive feature selection methods for forecasting, embedded feature selection methods minimize the forecasting error by performing weighted or standard feature selection during the training of the model. Reference [32] proposes an algorithm based on Lasso for forecasting solar intensity. Reference [33] uses Bayesian Ridge Regression for forecasting wind speed and direction 1 to 24 hours ahead. Reference [34] proposes the group method of data handling based on the Elastic Net in order to forecasting the load of a power grid. Although the works in this subcategory select features during the training of the forecaster, some parameters of these methods have to be set before training. In contrast with these past embedded feature selection methods for forecasting, all of the parameters in the design of our FSF architecture are trainable. Furthermore, our architecture can be combined and trained with any forecasting model via backpropagation.

Finally, we describe the relationship of our work to the past work on forecasting the traffic generation patterns of IoT devices, which is the immediate application domain of our FSF architecture. Forecasting the traffic generation patterns of individual IoT devices was proposed in [11]. In that work, the authors demonstrated the performance of MLP, LSTM, 1D CNN and ARIMA using feature selection that is based on ACF, embedding dimension as well as Analysis of Variance (ANOVA). Furthermore, Reference [4] used MLP, LSTM, and ARIMA, and each of [5] and [6] used MLP using ACF-based feature selection for forecasting the traffic generation pattern of an IoT device. In contrast with these past articles that are based on existing feature selection methodologies for forecasting IoT traffic, in this paper, we propose a novel architecture for forecasting IoT traffic generation patterns, which combines dynamic, automatic feature selection and forecasting.

III. END-TO-END TRAINABLE FEATURE SELECTION-FORECASTING ARCHITECTURE

In this section, we describe our end-to-end trainable feature selection-forecasting architecture FSF. Since our immediate application domain is that of forecasting the traffic generation pattern of an IoT device, we shall refer to the time series data as the "traffic generation pattern" of an IoT device in defining all of our variables. Table 1 gives the list of the mathematical symbols that we use in the order in which they appear in this paper.

Our FSF architecture is shown in Fig. 1. We let $\{x_t\}$ denote the traffic generation pattern of an individual IoT device, where t denotes the discrete time index.[1] As shown in the figure, the input of our FSF architecture is the subset of the past samples, denoted by $\{x_{t-l}\}_{l\in\{0,\dots,L+1\}}$, at current time t. The output of our architecture, denoted by $\{\hat{x}_{t+k}\}_{k\in\{1,\dots,K\}}$, is the set of forecasts at current time t over the next K samples.

As shown in Fig. 1, our FSF architecture is comprised of three modules: Feature Importance Score Calculation (FISC) module, Gamma-Gated Feature Selection (GG-FS) module, and the Forecasting module. The main idea behind FSF is as follows: The FISC module transforms the input vector of the past data into a vector of importance scores. The GG-FS module compares these importance scores against a threshold

[1]In practice, an IoT device typically generates traffic at multiples of a traffic generation period for that device; hence, we use a discrete time index t in order to refer to those instances at which the IoT device can potentially generate traffic. Note that this choice also allows the case in which the IoT device may generate zero traffic at a subset of these discrete-time instances.

TABLE 1: List of Symbols

Symbol	Meaning
x_t	Number of bits of traffic generated by an IoT device at current time t
$\hat{x}_{t+k}$	k^{th}-step ahead forecast at current time t
$s^m_{t,l}$	Feature importance score that corresponds to x_{t-l} at iteration m
γ^m	Value of the feature selection threshold at iteration m
$\tilde{x}^m_{t-l}$	The l^{th} past traffic generation sample for iteration m at current time t *after* the application of feature selection
L	Number of samples into the past used by the FSF architecture
K	Total number of steps into the future for which forecasts are formed
E	Total number of MLP layers
n_e	Number of neurons at layer e of the MLP
E_{LSTM}	Number of fully connected layers of LSTM
h_e	Number of neurons at fully connected layer e of LSTM
h_{LSTM}	Total number of LSTM units
c_{CNN}	Number of filters in the convolution layer of 1D CNN
c_e	Number of neurons in the fully connected layer e of 1D CNN
p	Order of observation lags in ARIMA
q	Number of samples that fall in one Moving Average (MA) window for ARIMA
d	Order of differencing for ARIMA

in order to determine dynamically which elements of the input vector will be selected as features to be passed onto the Forecasting module. Finally, the Forecasting module performs forecasting based on these selected features. The key novelty in our design is that *all* of the parameters in our FSF architecture are trainable end-to-end via backpropagation.

Backpropagation completes a forward and a backward pass through the entire FSF architecture in Fig. 1 in each iteration while updating the values of all of the parameters. Throughout this paper, the iteration index m shall appear as a superscript on each of the parameters of our architecture. For example, in Fig. 1, the input to the GG-FS module, labeled as γ^m, denotes the value of the threshold parameter at iteration m of the backpropagation through the FSF architecture.

We now describe the operation of the modules of our FSF architecture in Fig. 1. First, the FISC module computes the importance score for each of the elements of the input vector by measuring the pairwise relationship between the current input x_t and each of the past inputs $x_{t-l}, \forall l \in \{1, \dots, L+1\}$. In the figure, $s^m_{t,l}$ is the importance score for x_{t-l} at iteration m; it measures the importance of x_{t-l} in minimizing the forecasting error at the output of the FSF architecture.[2]

Second, each of the features x_{t-l} and the corresponding feature importance score $s^m_{t,l}$ of that feature is passed onto the GG-FS module. The GG-FS module passes only the selected features $\{\tilde{x}^m_{t-l}\}_{l\in\{0,\dots L\}}$ to the Forecasting module, where the values of the features that have *not* been selected are set to zero.[3]

Third, the Forecasting module performs K-step ahead forecasting. Any forecasting architecture with trainable parameters can be used for this Forecasting module. The parameters of the Forecasting module are trained jointly with the parameters of the FISC and GG-FS modules in our FSF architecture.

We shall now delve into the inner architecture of each of the FISC, GG-FS and Forecasting modules of FSF.

A. FEATURE IMPORTANCE SCORE CALCULATION (FISC) MODULE

The inner architecture of the FISC module is given in Fig. 2. In the first layer of the FISC module (which is the left-most layer in the figure), which we shall call the "customized layer", the lth neuron learns the relationship between x_t and x_{t-l}, where $l \in \{1, \dots, L+1\}$. The remainder of the FISC module is a fully connected 2-layer perceptron as shown in the figure.[4]

For each layer of the FISC, we set the number of neurons equal to the number of input pairs, which is $L + 1$. We represent the importance scores in the range $(0, 1)$. To this end, we set the activation function of each neuron in the last (i.e. output) layer of FISC to the unipolar sigmoid function. We also set the activation function of each neuron in the customized layer as well as the first layer of the fully-connected 2-layer perceptron to the tangent hyperbolic ($tanh$) function. Furthermore, each of the connection weights and the biases $\{(\tilde{w}_{l,1}, \tilde{w}_{l,2}, \tilde{b}_l)\}_{l\in\{1,\dots,L+1\}}$ is initialized as the realization of a uniform random variable on the interval $[0, 1]$. Note that the training of FISC is performed during the training of the entire FSF architecture; that is, this module is *not* trained in a stand-alone fashion.

B. GAMMA-GATED FEATURE SELECTION (GG-FS) MODULE

In Fig. 3, we display the inner architecture of the GG-FS module. This module has two input vectors, which are (1)

[2]The reason that the outputs of the FISC module are correctly interpreted as importance scores is that only a selected subset of these scores, which are deemed important, will be passed onto the Forecasting module by the GG-FS module. The operation of the GG-FS module will be described shortly.

[3]One must keep in mind that our FSF architecture selects features dynamically. Hence, at the end of training, after all of the parameters of the entire architecture have converged, the FISC module computes the importance scores dynamically. Hence, the particular features that are selected by GG-FS are also determined dynamically.

[4]For the FISC module, in the place of a 3-layer perceptron, which would be a general ANN architecture, we have chosen an architecture that possesses a customized design for the first layer in order to circumvent the greater number of local optima that would result if a fully-connected 3-layer design were used.

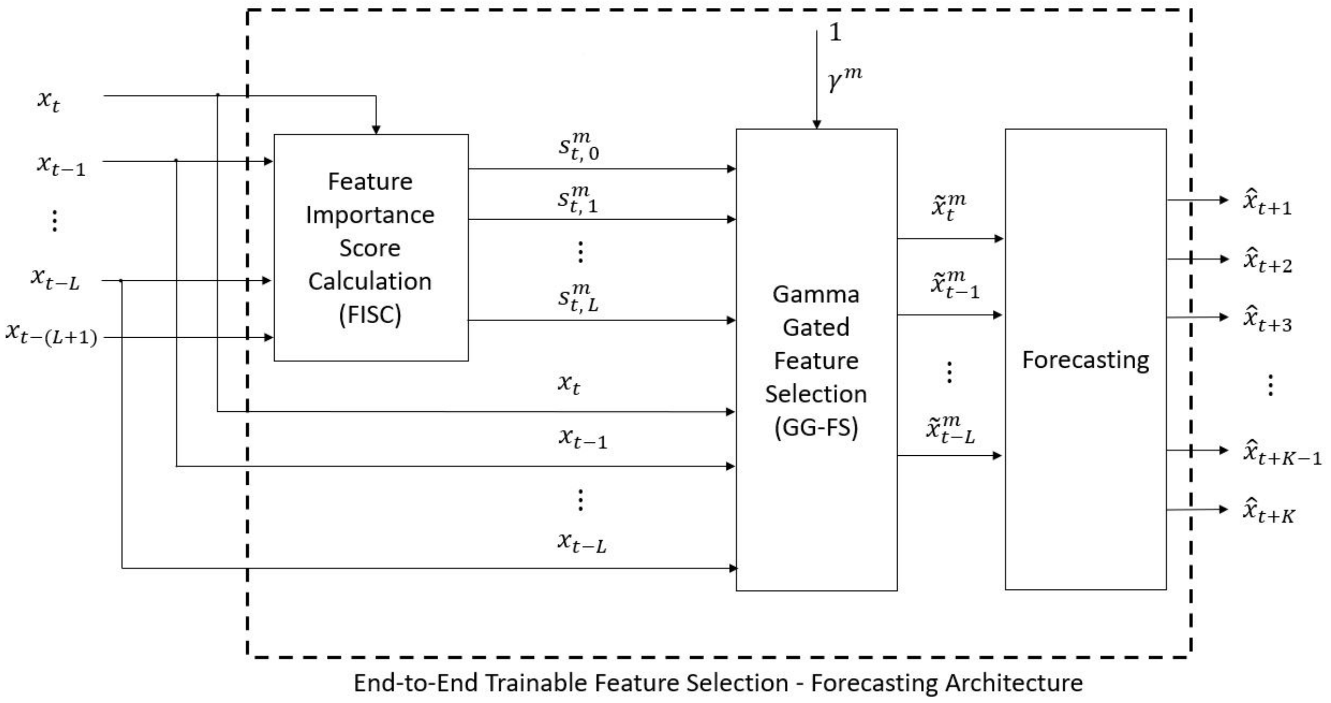


FIGURE 1: End-to-end trainable Feature Selection-Forecasting architecture (FSF), which is comprised of the Feature Importance Score Calculation (FISC), Gamma-Gated Feature Selection (GG-FS) and Forecasting modules

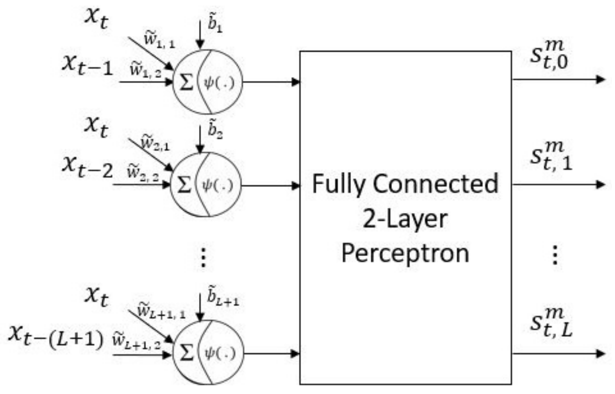


FIGURE 2: Inner architecture of the Feature Importance Score Calculation (FISC) module

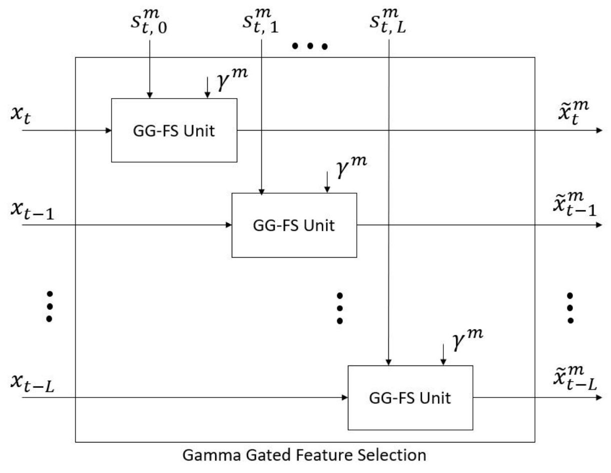


FIGURE 3: Inner architecture of the Gamma-Gated Feature Selection (GG-FS) module

the data points of the time series $\{x_{t-l}\}_{l\in\{0,\ldots,L\}}$ and (2) the feature importance scores $\{s^m_{t,l}\}_{l\in\{0,\ldots,L\}}$, which are the outputs of the FISC module.

As shown in Fig. 3, at iteration m, for each value of l, the GG-FS module computes the value of $\tilde{x}^m_{t-l}$ by using x_{t-l}, $s^m_{t,l}$ in the lth GG-FS Unit as well as the trainable parameter γ^m, which is common to *all* of the GG-FS units. That is, there is only a single trainable threshold parameter γ^m for the entire FSF architecture.[5]

Fig. 4 displays the inner architecture of a GG-FS Unit. In this figure, x_{t-l} and $s^m_{t,l}$ are the inputs, and $\tilde{x}^m_{t-l}$ is the output. In this figure, $\otimes$ denotes ordinary multiplication, and $u(\cdot)$ denotes the unit step function which outputs 0 if its argument is negative and outputs 1 otherwise. If $s^m_{t,l} < \gamma^m$, x_{t-l} is not

[5]Recall that the value of γ^m is updated by using backpropagation across the *entire* FSF architecture at each training iteration m.

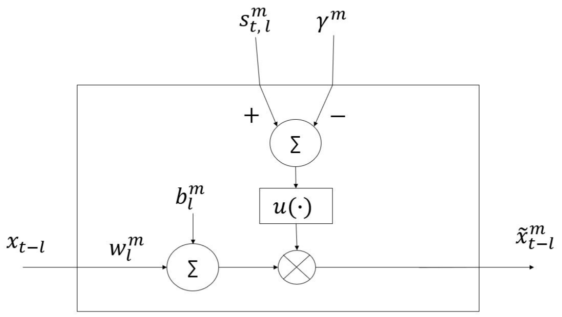


FIGURE 4: Gamma-Gated Feature Selection (GG-FS) Unit

selected as a feature. In this case, $\tilde{x}^m_{t-l}$ is set to 0. If $s^m_{t,l} \geq \gamma^m$, then x_{t-l} is selected as a feature. In this case, the GG-FS Unit multiplies x_{t-l} by the connection weight w^m_l and adds the bias term b^m_l and obtains $\tilde{x}^m_{t-l}$; that is, in the case that x_{t-l} is selected as a feature, it is transformed into $\tilde{x}^m_{t-l}$ via a single linear neuron.

Now, it is important to keep in mind that in order for the entire FSF architecture to be end-to-end trainable via backpropagation, all of the functions in the FSF architecture must be differentiable. Since the unit step function $u(\cdot)$ is not a differentiable function, we implement a differentiable approximation to the unit step function as $u(x) \approx 1/(1+e^{-\alpha x})$, which is a sigmoid function with steepness parameter α. In our implementation, we set $\alpha = 10$.

Recall that $s^m_{t,l}$ is a real number in the range $(0, 1)$. Note that if $\gamma^m \geq 1$, then none of the features will be selected by the GG-FS module. Conversely, if $\gamma^m \leq 0$, then all of the features will be selected. In order to prevent the FSF architecture from getting stuck at selecting none or all of the features, we restrict the value of γ^m such that it lies between 0 and 1.

C. FORECASTING MODULE

Any forecasting scheme can used as the Forecasting module of the FSF architecture in Fig. 1. The only criterion for the selection of a forecasting scheme is that the forecasting scheme be trainable by using backpropagation. This is a necessary condition in order to retain the end-to-end trainability of the entire FSF architecture. In Section IV, we examine the performance of the FSF architecture under the MLP and LSTM forecasting schemes.

As shown in Fig. 1, at each iteration m, the input to the Forecasting module is the vector of selected features $\{\tilde{x}^m_{t-l}\}_{l\in\{0,\dots,L\}}$, and the output of the Forecasting module is the vector of 1- to K-step ahead forecasts $\{\hat{x}_{t+k}\}_{k\in\{1,\dots,K\}}$.

D. NUMBER OF TRAINABLE PARAMETERS FOR FEATURE SELECTION IN THE FISC AND GG-FS MODULES OF THE FSF ARCHITECTURE

We now compute the order of the number of trainable parameters in the FISC and GG-FS modules of the FSF architecture, which carry out feature selection.[6]

To this end, first, we note that there are $L + 1$ neurons at each layer of the FISC module in Fig. 2. Since the first layer is not a fully connected layer and there are only three parameters (namely two connection weights and one bias parameter) for each neuron, the total number of parameters in the first layer of FISC is equal to $3(L+1)$. In addition, the rest of the FISC module is comprised of two fully connected layers, each of which consists of $L+1$ neurons; hence, there are $2(L + 1)$ bias parameters for these layers. The number of connection weights across all of the layers of FISC is $2(L + 1)^2$. Thus, the total number of parameters in FISC is $2L^2+9L+7$. Furthermore, since there are $L+1$ GG-FS Units in the GG-FS module, and there are only two parameters in each GG-FS Unit, the total number of parameters in the GG-FS module is $2(L + 1)$. In addition, the parameter γ^m is an input to the GG-FS module. Hence, the total number of parameters for the GG-FS module is $2L + 3$. As a result, the total number of parameters in the FISC and the GG-FS modules is $2L^2 + 11L + 10$. Hence, the total number of parameters for the FISC and the GG-FS modules is $\mathcal{O}(L^2)$.

IV. RESULTS

In this section, by using either an MLP or an LSTM forecaster in the FSF architecture in Fig. 1, we aim to compare the forecasting performance of our FSF architecture against the performance obtained by a selected subset of the following (feature selection, forecaster) pairs: Autocorrelation Function (ACF), Analysis of Variance (ANOVA), Recurrent Feature Elimination (RFE) and Ridge Regression for feature selection, and Linear Regression, Multi-Layer Perceptron (MLP), Long Short Term Memory (LSTM), 1 Dimensional Convolutional Neural Network (1D CNN), Autoregressive Integrated Moving Average (ARIMA), and Logistic Regression for forecasting. [7]

To this end, in this section, we first present the collection and processing methodology for IoT traffic data sets on which we have obtained our results. Second, we present the methodology for competing feature selection methods and for tuning the hyperparameters of competing forecasting models. Third, we describe the 10-fold cross-validation method for the performance evaluation of all of the forecasting models under examination. Fourth, we present the results on the forecasting performance as well as the training and execution times.

[6]We calculate the number of parameters excluding those in the Forecasting module because the number of parameters in the Forecasting module depends on the particular forecasting scheme.

[7]In order to compare the performance of the FSF architecture against that of representative (feature selection, forecaster) pairs, we have selected ACF-based and ANOVA-based feature selection as representatives of filter-based, RFE as a representative of wrapper-based, and Ridge Regression as a representative of embedded feature selection methods.

A. COLLECTION AND PROCESSING METHODOLOGY FOR INTERNET OF THINGS TRAFFIC DATA SETS

In [11], the traffic generation patterns of individual IoT devices were classified into four distinct classes as follows: First, if the IoT device generates only a constant number of bits, its traffic generation pattern is said to be "Fixed Bit"; otherwise, it is "Variable Bit". Furthermore, if the generation intervals of the traffic of the IoT device are constant, its traffic is called "Periodic"; otherwise, it is called "Aperiodic". According to this classification, the traffic generation of an individual IoT device falls in one of the four classes: (1) Fixed Bit Periodic (FBP), (2) Fixed Bit Aperiodic (FBA), (3) Variable Bit Periodic (VBP), and (4) Variable Bit Aperiodic (VBA). Since both the number of bits and the generation interval for the FBP class are known in advance, forecasting is required only for the traffic generation patterns in the VBP, VBA and the FBA classes. In this paper, we present our results on the traffic generation patterns of three distinct IoT devices for each of these classes.

For the VBP device class, we collected sensor measurements on Relative Humidity (RH) and Light Dependent Resistor (LDR) sensors, and obtained the data for the Water Level sensor from [35]. For the VBA class, we obtained the actual sensor readings for the NO_2 sensor from [36], [37], and we collected the data for the Temperature and Elevator Button[8] sensors.[9] For the FBA class, we obtained the sensor readings for the NMHC sensor from [36], [37]; for the Smart Home Energy Generation sensor from [38], and for the Wind Speed sensor from [39].

We converted each sequence of actual readings obtained for each sensor listed in the VBP and VBA classes above to the sequence of number of bits that the sensor needs to send by performing Huffman compression on the fixed-bit representation of each such reading.[10] On each sensor reading for each sensor listed in the FBA class above, we used a fixed-bit representation.

For all of the (feature selection, forecasting) pairs as well as FSF under examination, we set the total number of features (namely, the number of successive data samples into the past) to 120.[11]

B. HYPERPARAMETER SEARCH AND FEATURE SELECTION FOR FORECASTING MODELS AGAINST WHICH FSF IS COMPARED

In order to ensure that we compare the performance of our FSF architecture against the minimum forecasting error that can be obtained by each competing model, we perform a grid search for tuning the hyperparameters of the MLP, LSTM, ARIMA, 1D CNN, Logistic Regression and Ridge Regression models in this section.

To this end, first, in Table 2, we present the hyperparameters and the search set for each forecasting model. The first column of this table shows the names of the forecasting models. The second column displays the names of the hyperparameters. For all forecasting models except ARIMA, the third column shows the values for the hyperparameters that are set empirically or the set of values over which grid search is performed. For the ARIMA model, the third column shows the tuning methods for its hyperparameters.

In Table 3, we present the feature selection methods against which our FSF architecture is compared. The first column of this table displays the names of the data sets. The second column shows the features which are selected via ACF-based Feature Selection. The third, fourth and the fifth columns describe the methodology for ANOVA-based Feature Selection, RFE, and Ridge Regression, respectively.

C. CROSS-VALIDATION: TRAINING AND TEST

In order to observe the performance of the forecasting models, we train and test each of the forecasting models by using 10-fold cross-validation (CV).[13] We calculate the performance of the forecaster at each fold f and compute the mean performance over all of the folds. At each fold f, during training, we minimize the Mean Absolute Error (MAE) for the VBP class, and the Categorical Cross-Entropy for the VBA and the FBA classes[14]. In order to prevent overfitting, instead of using a fixed number of epochs, we use a variable number of epochs across all of the folds of any given data set.

We define $^1/_2$-sMAPE as sMAPE divided by 2.[15]. Furthermore, in the rest of this paper, misclassification error will refer to the average of the fraction of classification errors across the length K vector of forecasts in Fig. 1.

As the output of CV, we obtain three different measurements: (1) the forecasting error $^1/_2$-sMAPE for the VBP and

[8]We measure the number of times that an elevator button is pressed in each minute.

[9]We have collected the sensor readings for RH, LDR, Temperature and Elevator Button data sets in our laboratory over 2.5 months.

[10]Huffman coding thus leads to a variable number of bits to represent each sensor reading.

[11]The total number of samples for the data set of each sensor is as follows: 9357 for RH; 65535 for LDR; 25000 for Water Level; 2288 for NO_2; 7371 for Temperature; 40080 for Elevator Button; 7715 for NMHC; 8692 for Smart Home; and 25000 for Wind Speed. In addition, the number of quantization levels (i.e. the number of unique values for the number of bits) is 2 for the FBA class and is as follows for each data set in VBP and VBA classes: 6 for RH; 213 for LDR; 60 for Water Level; 7 for NO_2; 10 for Temperature; and 15 for the Elevator Button.

[12]By empirically analyzing the PACF and ACF, we set the value of each of the p and q hyperparameters to the greatest index of the sample whose score, in Partial ACF (PACF) and ACF respectively, is greater than a threshold.

[13]In order to obtain a 10-fold CV, we divide the time series data set into 10 disjoint subsets of an equal number of elements in each subset. Then, in each step (namely, "fold") f of the 10-fold CV, we use the f^{th} subset as the test set and the rest nine subsets as the training set. That is, by the end of the CV, we have use each subset nine times as the training set and exactly once as the test set.

[14]We use *softmax* at the output of the forecaster for the VBA and the FBA classes. As a result, we minimize the Categorical Cross-Entropy for the misclassification error between the output classes that are constituted by the distinct number of bits that a sensor can generate.

[15]This serves to normalize this error measure such that it lies in the range $[0\%, 100\%]$ rather than in the range $[0\%, 200\%]$ (as is the case for the original sMAPE measure).

TABLE 2: GRID SEARCH FOR TUNING HYPERPARAMETERS OF FORECASTING MODELS

Model Name	Hyperparameters	Value / Search Set
MLP	The number of layers E	4
	The number of neurons n_E at layer E	K
	n_e at each layer $e \in \{1, \ldots, E-1\}$	$\{2^j\}_{j\in\{1,\ldots,8\}}$
	The activation function of each neuron at each hidden layer	$ReLU$
	The activation function of each neuron at output layer	$Linear$ (for VBP) $softmax$ (for VBA & FBA)
LSTM	The number of LSTM layers	1
	The number of LSTM units h_{LSTM} at each LSTM layer	$\{2^j\}_{j\in\{1,\ldots,8\}}$
	The number of fully connected layers E_{LSTM}	3
	The number of neurons $h_{E_{LSTM}}$ at layer E_{LSTM}	K
	h_e at each fully connected layer $e \in \{1, \ldots, E_{LSTM}-1\}$	$\{2^j\}_{j\in\{1,\ldots,8\}}$
	The activation function of each LSTM unit and neuron at hidden fully connected layers	$ReLU$
	The activation function of each neuron at output layer	$Linear$ (for VBP) $softmax$ (for VBA & FBA)
1D CNN	The number of convolution layers	1
	The number of filters c_{CNN} at each convolution layer	$\{2^j\}_{j\in\{1,\ldots,8\}}$
	The kernel size of each convolution filter	$(3, 3)$
	The stride of each convolution filter	$(2, 1)$
	The number of max pooling layers	1
	The kernel size of each max pooling filter	$(3, 3)$
	The number of fully connected layers E_{CNN}	4
	The number of neurons $c_{E_{CNN}}$	K
	c_e at each fully connected layer $e \in \{1, \ldots, E_{CNN}-1\}$	$\{2^j\}_{j\in\{1,\ldots,8\}}$
	The activation function of each convolution and hidden fully connected layers	$ReLU$
	The activation function of each neuron at output layer	$Linear$ (for VBP) $softmax$ (for VBA & FBA)
Logistic Regression	The regularization term	$\{0.01, 0.1, 1, 10\}$
	The method of penalty	{none, $l1$, $l2$, elasticnet}
	Solver	Newton's method, / limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm, / library for large linear classification, / stochastic average gradient, / stochastic average gradient with non-smooth penalty
Model Name	**Hyperparameters**	**Tuning Method**
ARIMA	The order of observation lags, Autoregression (AR in ARIMA), p	Analysis based on PACF[12]
	The degree of difference for the samples in the raw time series data, d	Dickey Fuller Test
	The number of samples that fall in one Moving Average (MA) window, q	Analysis based on ACF

TABLE 3: FEATURE SELECTION METHODS AGAINST WHICH FSF IS COMPARED

Data Sets	**ACF-based Feature Selection**	**ANOVA-based Feature Selection**	**Recurrent Feature Elimination (RFE)**	**Ridge Regression**
RH	$\{1,2,3,4,5\} \cup \{12 * j\}_{j\in\{1,\ldots 10\}}$	For each feature, we first compute the F-ratio [40] between that feature and the desired output by using ANOVA in the *statsmodels* [41] library in Python. We sort the features with respect to their F-ratios in descending order and select the first twelve features from this sorted sequence of features.	We use the RFE algorithm from *scikit-learn* library whose inputs are the forecasting model and the desired number of features (denoted by N). We selected the forecasting model as Linear Regression. We build an outer loop for N from 1 to L in order to find the value of N that achieves the minimum forecasting error.	We use the *scikit-learn* [42] library in Python. By using this library, we normalize the data at the output of Ridge (which is the input of the regressor) by subtracting the mean of the data and by dividing the result by the $l2$-norm of the data. We search for the value of the coefficient of the $l2$-regularization term, α, in the set $\{10^j\}_{j\in\{-7,\ldots,-1\}}$.
NO_2	The first 3 samples			
NMHC	$\{23 * j\}_{j\in\{1,\ldots,5\}}$			
LDR	The highest 12 values of the ACF			
Water Level				
Temperature				
Elevator Button				
Wind Speed				
Smart Home Energy Generation				

the VBA classes, and the misclassification error for the FBA class, (2) the training time, and (3) the execution time.

D. PERFORMANCE COMPARISON OF THE FSF ARCHITECTURE AGAINST EXISTING FORECASTING MODELS

In this section, we aim to present the performance evaluation of the FSF architecture and its comparison against the existing forecasting models. To this end, for each data set in VBP, VBA and FBA classes, we measure and display the performance of each forecasting model averaged over the values of K from 1 to 15.[16]

First, in Fig. 5, we present the average $^1/_2$-sMAPE performance of our FSF architecture under each of the MLP and LSTM forecasters (namely FSF-MLP and FSF-LSTM respectively) for the RH sensor, LDR, and the Water Level data sets, each of which falls in the VBP class. In this figure, we see that for all of these data sets, FSF-MLP achieves the lowest forecasting error among all of the forecasting models. We also note that the performance of FSF-MLP almost equals that of FSF-LSTM for the LDR and the Water Level data sets.

For the RH data set in Fig. 5(a), we see that the linear forecasting models (RFE, Ridge, Linear Regression) achieve the minimum forecasting error among the models under comparison (excluding FSF-MLP and FSF-LSTM), which suggests that an approximately linear relationship is inherent in the data. In Fig. 5(b), for the LDR data set, we see that the nonlinear forecasting models outperform the linear and statistical models, which suggests that a nonlinear relationship is inherent in the data. In Fig. 5(c), for the Water Level data set, we see that the nonlinear models that possess feature selection capability (namely ACF-MLP and ANOVA-MLP) or those with feature extraction in their internal architecture (namely LSTM and 1D CNN) outperform the other existing models. The fact that FSF-MLP outperforms the best existing models in each of these three categories suggests that FSF is able to capture both linear and nonlinear relationships as well as performing well in capturing trends in data for which feature extraction is crucial. Jointly, for the VBP class, these results point to the advantages of our highly flexible, end-to-end trainable FSF architecture.

Second, in Fig. 6, we present the $^1/_2$-sMAPE performance (averaged over K) of our FSF-MLP and FSF-LSTM architectures for the NO_2, Temperature, and the Elevator Button data sets that fall in the VBA class. Our results in this figure show that both the FSF-MLP and FSF-LSTM architectures achieve at least the same performance as those existing models that achieve the minimum forecasting error. In addition, in Fig. 6(a), LSTM and ANOVA-MLP are seen to be the forecasting models that are the most competitive against our FSF architecture for the NO_2 data set. The reason is that l approaches L, the F-ratio increases and thus the features x_{t-l} that have values near L become more important for this data set. Accordingly, LSTM performs well because it captures the long-term relationships, and ANOVA-MLP performs well because it selects the features that have a high F-ratio. This implies that the FSF architecture is able to select the important features, each of which has a long-term relationship with the future samples.

[16]In the Appendix, we present the performance comparison for each value of K separately.

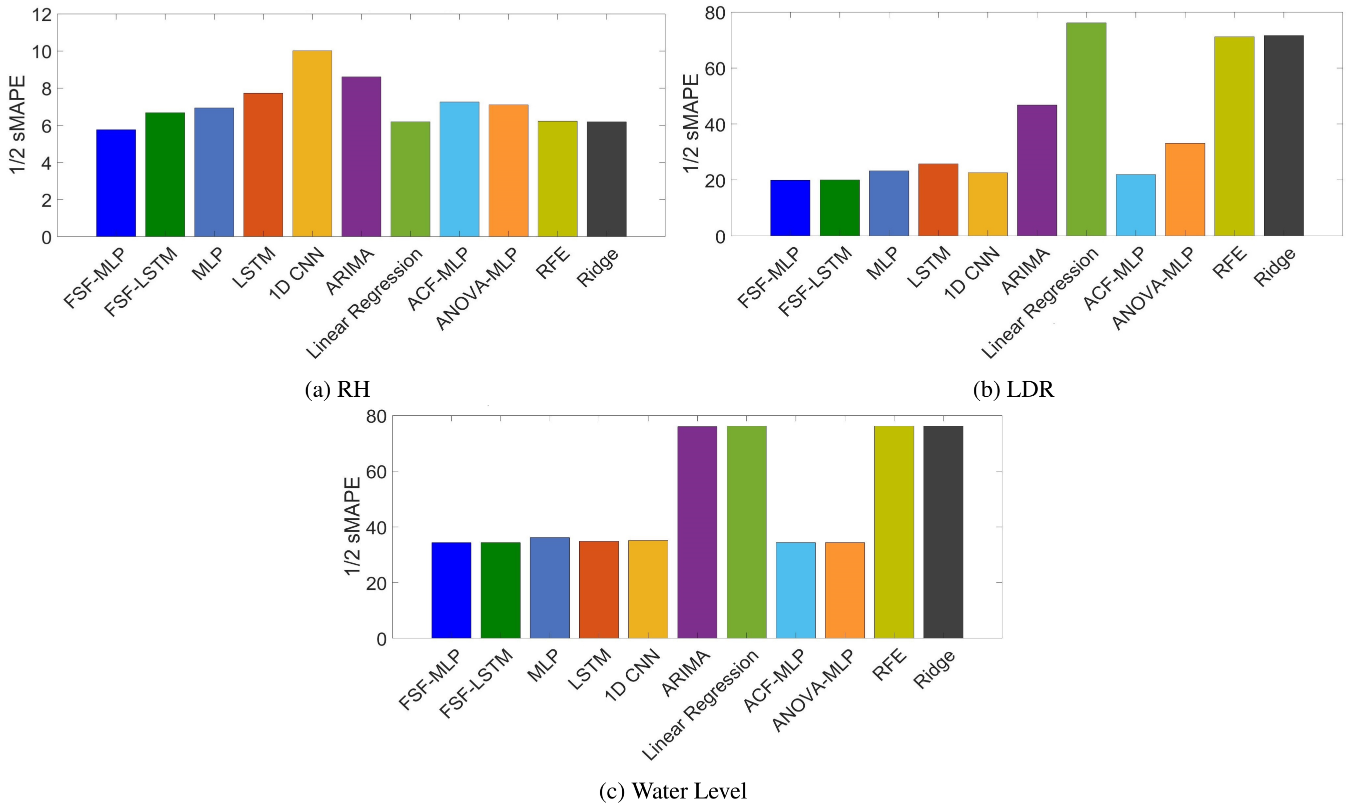


FIGURE 5: Comparison of FSF-MLP and FSF-LSTM against existing models with respect to the average $^1/_2$-sMAPE over the values of K for (a) RH, (b) LDR, and (c) Water Level data sets, each of which falls in the VBP class

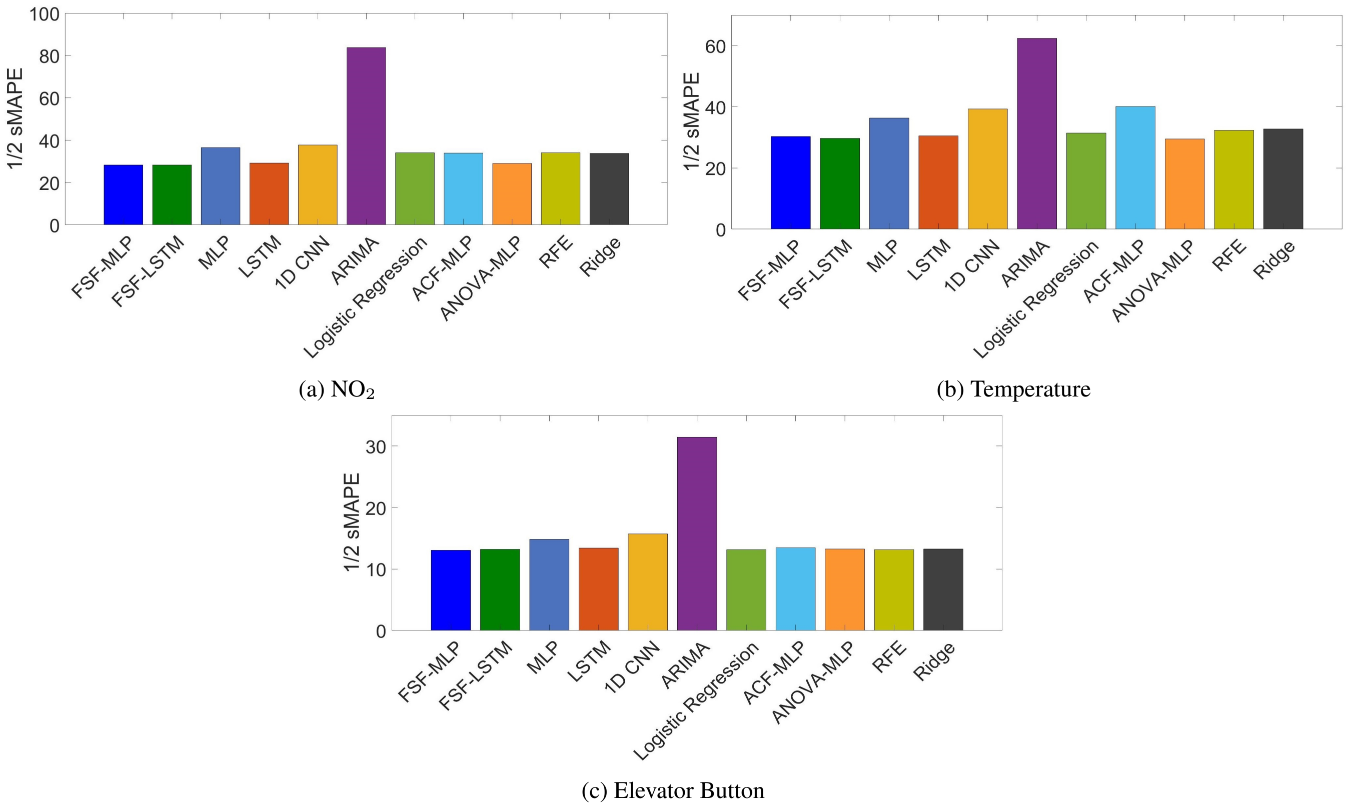
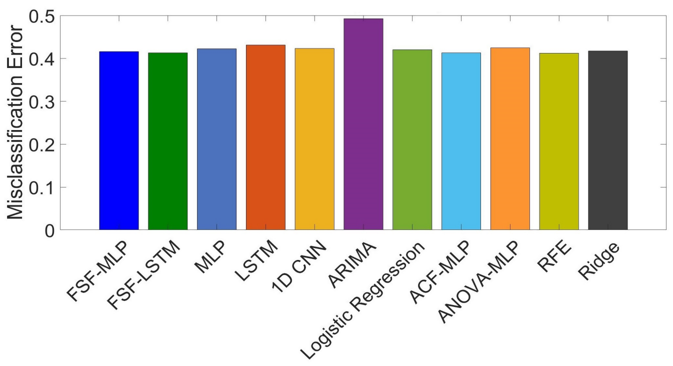
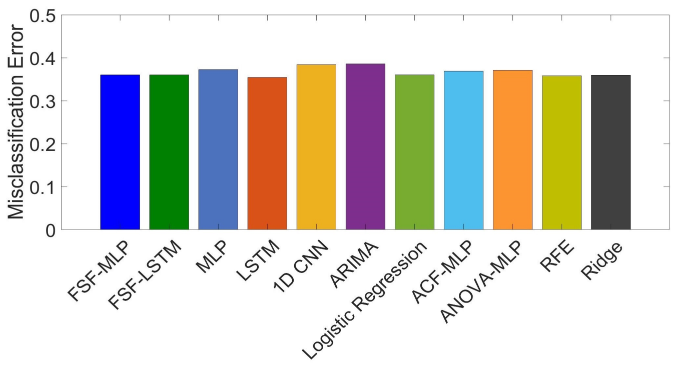


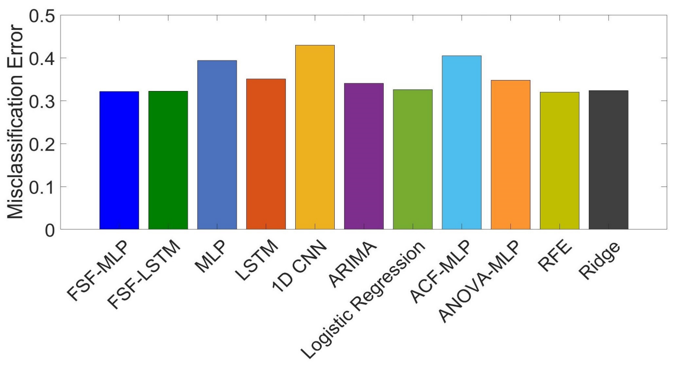
FIGURE 6: Comparison of FSF-MLP and FSF-LSTM against existing models with respect to the average $^1/_2$-sMAPE over the values of K for (a) NO$_2$, (b) Temperature, and (c) Elevator Button data sets, each of which falls in the VBA class



(a) NMHC



(b) Wind Speed



(c) Smart Home Energy Generation

FIGURE 7: Comparison of FSF-MLP and FSF-LSTM with existing models with respect to average $^1/_2$-sMAPE over the values of K for the (a) NMHC, (b) Wind Speed, and the (c) Smart Home Energy Generation data sets that fall in the FBA class.

Third, in Fig. 7, we present the average misclassification error of FSF-MLP and FSF-LSTM for the NMHC, Wind Speed, and the Smart Home Generation data sets that fall in the FBA class. Our results in this figure show that both FSF-MLP and FSF-LSTM are able to achieve either the minimum error or an error close to the minimum that is achieved by the existing forecasting models.

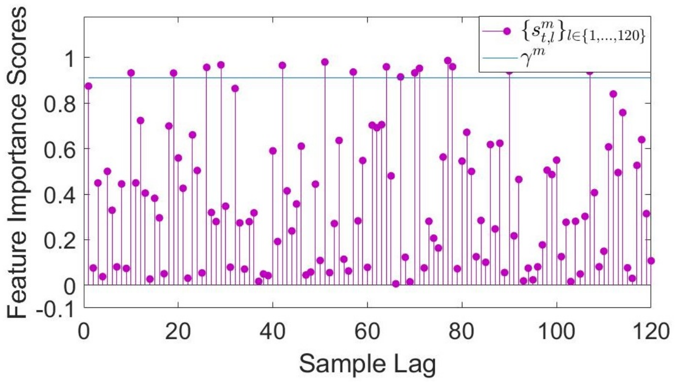
In summary, the above results jointly show that FSF-MLP and FSF-LSTM achieve either the best performance or a performance that is competitive with the best-performing models for all data sets. In addition, our results suggest that the generalization ability of the FSF architecture across widely different data sets is high. On the other hand, we also see that the models that utilize the FSF architecture significantly outperform the existing forecasting models for the data sets in the VBP class. Furthermore, the FSF architecture is able to achieve a performance that is the same as or close to that of the best existing models for the VBA and the FBA classes.

We now aim to display the differences between the feature importance scores that are calculated by the FISC module in the FSF architecture and those are calculated by the other feature selection methods. To this end, in Fig. 8 for the RH data set, we show the average (over samples) of the feature importance scores in FSF-MLP[17], the coefficients of Linear Regression, the values of the ACF, and the normalized F-ratios of ANOVA.

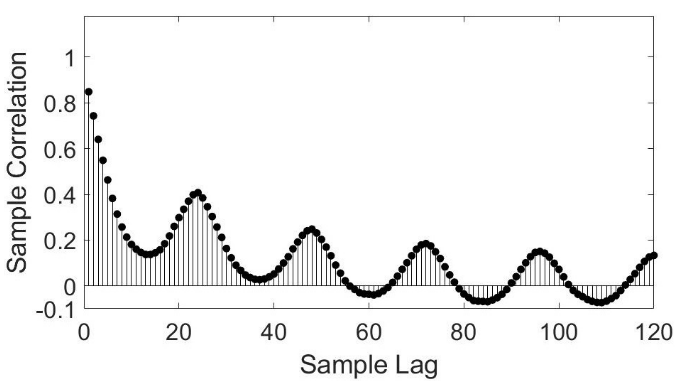
Fig. 8(a) shows the vector of feature importance scores of FSF-MLP, which appears at the output of the FISC module. We see that the FSF architecture captures the important features that do not possess any seasonal trend and which thus cannot be found by a simple linear operation. In Fig. 8(b), we see that ACF captures the seasonal trends of the VBP traffic generation; however, feature selection based only on these seasonal trends is suboptimal due to the linear structure of the ACF.

Fig. 8(c) presents the coefficients in ANOVA, for which the scores of the features are defined based on variance. In this figure, we see that the features x_{t-l} for smaller values of l are found to be more important by ANOVA. In Fig. 8(d), we see that the Linear Regression model amplifies the data sample with a time lag of 1, namely x_{t-1}, which has the highest covariance with x_{t+1} according to ANOVA. That is, we see that the FSF architecture is able to capture this nonlinear relationship (that is not captured by any other method) between the features and the desired output of the forecaster in order to maximize the forecasting performance.

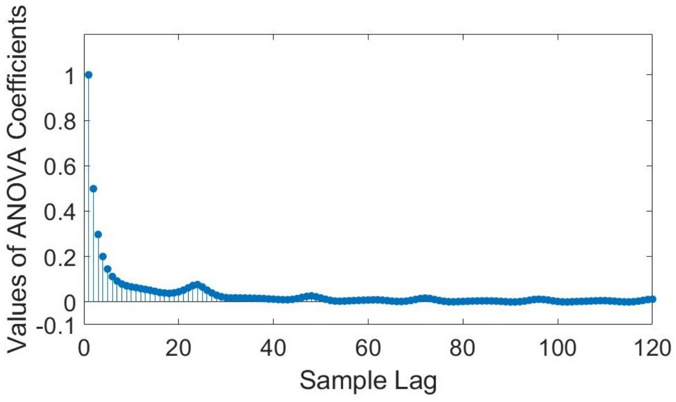
[17]For the RH data set, we only present the feature importance scores for FSF-MLP because FSF-MLP outperforms FSF-LSTM in this case.



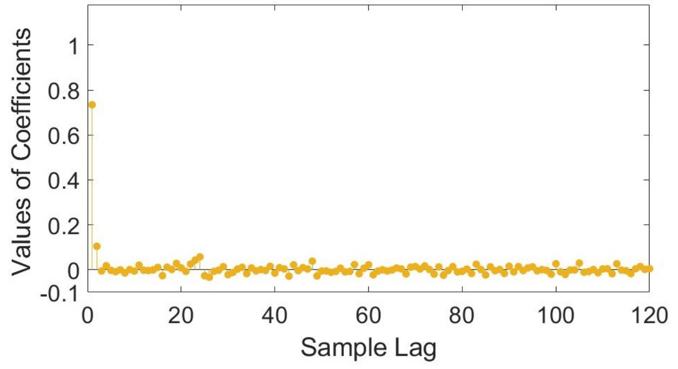
(a) Feature importance scores of FSF-MLP



(b) Autocorrelation Function (ACF)



(c) ANOVA Coefficients



(d) Linear Regression Coefficients

FIGURE 8: Coefficients of distinct schemes as a function of the sample lag for the RH data set, which falls in the VBP class

E. COMPARISON OF THE FSF ARCHITECTURE AGAINST EXISTING FORECASTING MODELS WITH RESPECT TO TRAINING AND EXECUTION TIMES

In order to present the trade-off between the forecasting performance and the computation (training and execution) time of the FSF architecture, we now present our results on the training and the execution time of each model in the previous section. We have obtained these results on the Google Colab computational platform that has a Tensor Processing Unit (TPU) accelerator.

1) Training Time

In this section, we present the comparison of the forecasting models with respect to training time. Note that training time depends not only on the architecture of the forecasting model but also on the number of samples in the data set; hence, the training times of a forecasting model on distinct data sets are not directly comparable.

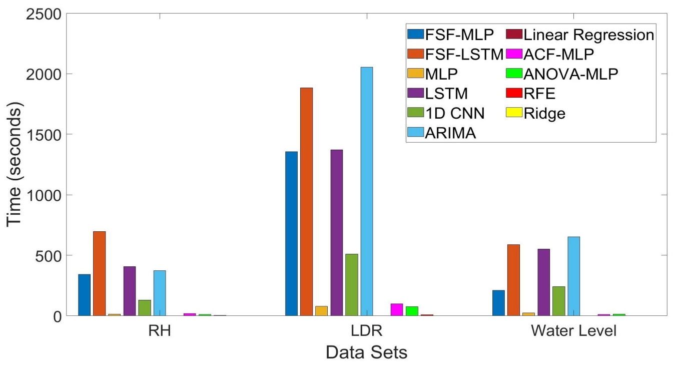
In Fig. 9, we present the training time of each forecasting model for the data sets in the VBP, VBA and the FBA classes. In this figure, we see that the training time of the FSF architecture is comparable to that of LSTM, 1D CNN, and ARIMA. In addition, the training times of FSF-MLP, FSF-LSTM, LSTM, 1D CNN, and ARIMA models are higher than those of the stand-alone MLP and the linear forecasting models.

2) Execution Time

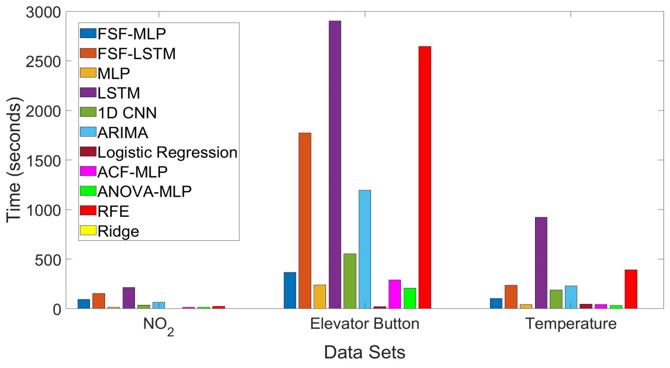
In Fig. 10, we present the execution time of each forecasting model for the data sets in the VBP, VBA, and the FBA classes. We see that for all data sets, the execution times of FSF-MLP and FSF-LSTM are less than 10 ms, which is highly acceptable for the IoT applications that require subsecond computation time. On the other hand, we also see that FSF-MLP, FSF-LSTM, and ARIMA have the highest and Linear Regression, Logistic Regression, RFE, and Ridge have the lowest execution times. Although the execution time of the FSF architecture is comparable to that of ARIMA in the majority of cases, it is 1 to 3 orders higher than those of LSTM, MLP, ACF-MLP, ANOVA-MLP, 1D CNN, Linear Regression, Logistic Regression, RFE, and Ridge.

V. CONCLUSION

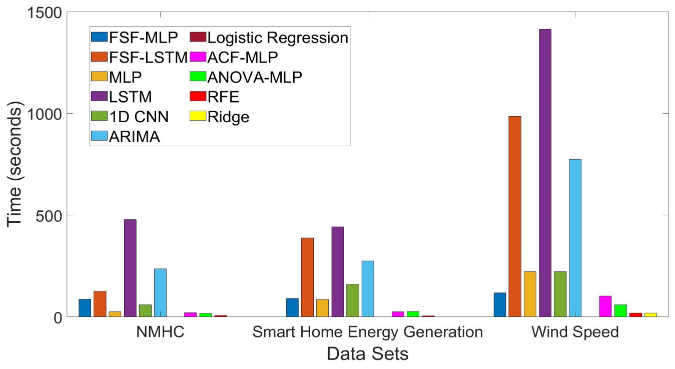
We have developed a novel feature selection-forecasting architecture, called FSF, that is able to perform automatic, dynamic selection of features in order to minimize the fore-



(a) Training Time for VBP Data Sets

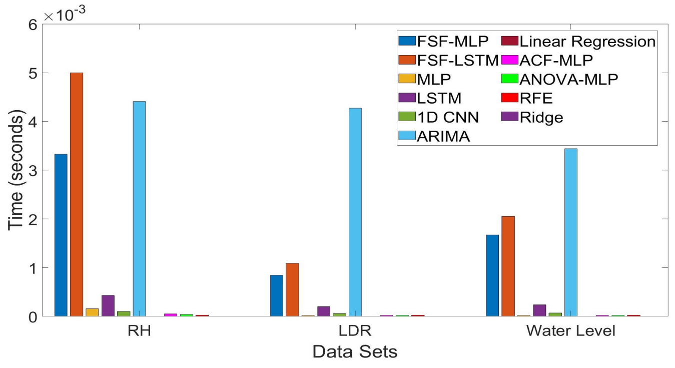


(b) Training Time for VBA Data Sets

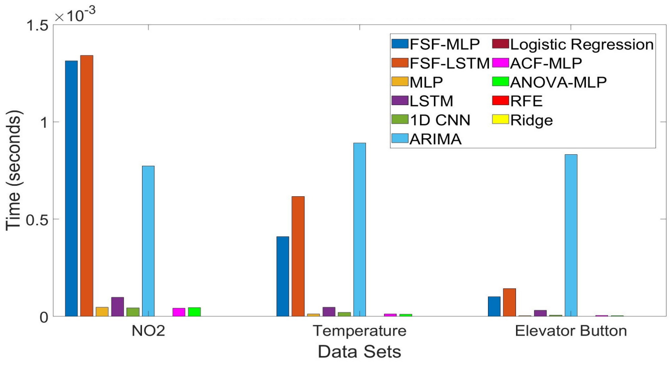


(c) Training Time for FBA Data Sets

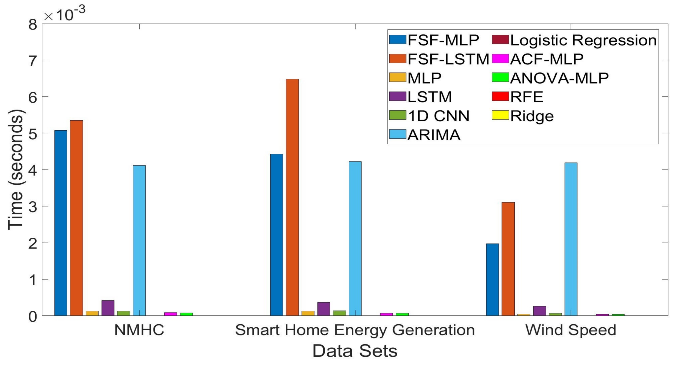
FIGURE 9: Training time of the forecasting models on each data set in the following device classes: (a) VBP, (b) VBA, (c) FBA



(a) Execution Time for VBP Data Sets



(b) Execution Time for VBA Data Sets



(c) Execution Time for FBA Data Sets

FIGURE 10: Execution time per sample of the forecasting models on each data set in the following device classes: (a) VBP, (b) VBA, (c) FBA

casting error. Our architecture stands at a point between Deep Learning techniques that discover all of the features themselves but require long data for training, and the rest of the Machine Learning techniques that select features based on existing filter-based, wrapper-based and embedded feature selection methods.

We have demonstrated the performance of our FSF architecture on the problem of forecasting the future traffic generation patterns of individual IoT devices. We have found that FSF achieves either the best or close to the best performance among a competitive selection of existing (feature selection, forecasting) technique pairs. Our architecture achieves such high performance by virtue of the fact that all of the parameters of the architecture are end-to-end trainable via backpropagation.

The execution time of FSF per sample is below 7 ms on the Google Colab platform. Such an execution time is reasonable for massive IoT applications, in which the delay constraints are on the order of seconds to hours. As a result, we expect that the FSF architecture can be used as a forecaster to achieve predictive resource allocation in next-generation networks for massive IoT.

Even though we have developed FSF in order to forecast the traffic of individual IoT devices, we note that FSF is a general architecture that can be applied to *any* forecasting problem. In particular, whenever the number of data samples is not sufficient for the Deep Learning-based forecasting algorithms to converge, FSF provides a high-performance solution that can only be approximated by the existing (feature selection, forecasting) method pairs. Fine-tuning the hyperparameters of all of the feature selection methods and the forecasters separately in order to find the best-performing solution is an arduous task. FSF obviates this task by discovering a high-performance solution automatically. As a result, we expect that FSF will have an impact on multiple areas beyond IoT.

In our future work, first, we shall examine how to reduce the execution time of our FSF architecture. Second, we plan to apply FSF to novel problems in next-generation predictive networks. We expect that smart cities of the near future will rely heavily on prediction in all layers of the communication protocol stack as well as in providing network services to the inhabitants of these cities. As a flexible, automatic, dynamic forecasting architecture, we expect that FSF will emerge as a key enabler of such predictive networking solutions.

APPENDIX. PERFORMANCE COMPARISON WITH RESPECT TO THE STEP-AHEAD PARAMETER

In this appendix, we show how forecasting error varies with respect to K, the step-ahead forecasting parameter. We note that the results that appear in Fig. 5, 6 and7 show the forecasting error that has been averaged over the values of K that appear in Tables 4,5, 6, respectively. The first column of each of these tables displays the range of values of the number of bits of traffic generated by each sensor, and the number of samples. The second column shows the models, and the remaining columns display the forecasting error for the values of K in our measurements. The lowest value of the forecasting error has been indicated in boldface for each K. Our key observation is that FSF-MLP and FSF-LSTM either have the lowest or close to the lowest forecasting error across all schemes across all K over all of the data set classes. Hence, we see that the main conclusion that we have drawn in Section IV-D regarding the performance of our FSF architecture holds not only with respect to the average of the forecasting error over all K but also for each value of K in the underlying set of results that led to that average.

REFERENCES

[1] J. Brown and J. Y. Khan, "A predictive resource allocation algorithm in the LTE uplink for event based M2M applications," IEEE Transactions on Mobile Computing, vol. 14, no. 12, pp. 2433–2446, 2015.
[2] R. Atawia, H. S. Hassanein, and A. Noureldin, "Optimal and robust QoS-aware predictive adaptive video streaming for future wireless networks," in GLOBECOM 2017-2017 IEEE Global Communications Conference. IEEE, 2017, pp. 1–6.
[3] N. Khambari, B. Ghita, and L. Sun, "QoE-driven video enhancements in wireless networks through predictive packet drops," in 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). IEEE, 2017, pp. 355–361.
[4] M. Nakip, V. Rodoplu, C. Güzeliş, and D. T. Eliiyi, "Joint forecasting-scheduling for the Internet of Things," in 2019 IEEE Global Conference on Internet of Things (GCIoT). IEEE, 2019, pp. 1–7.
[5] V. Rodoplu, M. Nakıp, D. T. Eliiyi, and C. Güzelis, "A multi-scale algorithm for joint forecasting-scheduling to solve the massive access problem of IoT," IEEE Internet of Things Journal, vol. 7, no. 9, pp. 8572–8589, 2020.
[6] V. Rodoplu, M. Nakip, R. Qorbanian, and D. T. Eliiyi, "Multi-channel joint forecasting-scheduling for the internet of things," IEEE Access, vol. 8, pp. 217 324–217 354, 2020.
[7] M. Li, X. Guan, C. Hua, C. Chen, and L. Lyu, "Predictive pre-allocation for low-latency uplink access in industrial wireless networks," in IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, 2018, pp. 306–314.
[8] C. Yao, J. Guo, and C. Yang, "Achieving high throughput with predictive resource allocation," in 2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP). IEEE, 2016, pp. 768–772.
[9] R. Atawia, H. S. Hassanein, H. Abou-Zeid, and A. Noureldin, "Robust content delivery and uncertainty tracking in predictive wireless networks," IEEE Transactions on Wireless Communications, vol. 16, no. 4, pp. 2327–2339, 2017.
[10] J. Wen, M. Sheng, J. Li, and K. Huang, "Assisting intelligent wireless networks with traffic prediction: Exploring and exploiting predictive causality in wireless traffic," IEEE Communications Magazine, vol. 58, no. 6, pp. 26–31, 2020.
[11] M. Nakip, B. C. Gül, V. Rodoplu, and C. Güzeliş, "Comparative study of forecasting schemes for IoT device traffic in machine-to-machine communication," in Proceedings of the 2019 4th International Conference on Cloud Computing and Internet of Things, 2019, pp. 102–109.
[12] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," Computers & Electrical Engineering, vol. 40, no. 1, pp. 16–28, 2014.
[13] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," IEEE Transactions on Smart Grid, vol. 10, no. 1, pp. 841–851, 2017.
[14] S. Khan, N. Javaid, A. Chand, A. B. M. Khan, F. Rashid, and I. U. Afridi, "Electricity load forecasting for each day of week using deep CNN," in Workshops of the International Conference on Advanced Information Networking and Applications. Springer, 2019, pp. 1107–1119.
[15] C.-J. Huang and P.-H. Kuo, "A deep CNN-LSTM model for particulate matter (pm2. 5) forecasting in smart cities," Sensors, vol. 18, no. 7, p. 2220, 2018.
[16] J. Pati, B. Kumar, D. Manjhi, and K. K. Shukla, "A comparison among ARIMA, BP-NN, and MOGA-NN for software clone evolution prediction," IEEE Access, vol. 5, pp. 11 841–11 851, 2017.

TABLE 4: CROSS-VALIDATION $^1/_2$-sMAPE RESULTS FOR THE VBP DATA SETS

DATA SETS	MODELS	K = 1	K = 2	K = 3	K = 4	K = 5	K = 10	K = 15
RH 1 kbit - 7 kbits 9357 samples	FSF-MLP	3.38	**4.33**	**5.25**	**5.77**	**6.16**	**7.63**	**7.83**
	FSF-LSTM	3.43	4.89	6.74	7.06	7.82	7.94	8.88
	MLP	5.24	5.99	6.50	6.93	7.15	8.00	8.75
	LSTM	6.10	6.44	7.41	7.52	7.89	8.95	9.80
	1D CNN	9.81	9.09	9.34	10.68	9.58	10.41	11.17
	ARIMA	5.98	7.30	8.19	8.78	9.18	10.22	10.56
	Linear Regression	3.50	4.76	5.68	6.30	6.75	7.94	8.43
	ACF-MLP	5.23	6.02	6.78	7.49	7.70	8.65	8.84
	ANOVA-MLP	5.76	6.29	6.89	6.98	6.96	7.96	8.83
	RFE	**3.23**	4.73	5.76	6.48	6.98	7.93	8.44
	Ridge	3.49	4.76	5.67	6.29	6.75	7.94	8.43
LDR 2 kbits - 1984 kbits 65535 samples	FSF-MLP	20.01	**18.18**	**20.08**	20.19	20.19	**20.39**	**20.64**
	FSF-LSTM	**19.45**	19.59	21.29	**18.16**	**19.86**	21.05	20.84
	MLP	27.14	22.63	21.09	23.53	24.04	21.92	22.45
	LSTM	25.03	25.05	24.55	23.83	20.50	25.16	36.54
	1D CNN	22.95	23.96	22.11	23.03	22.08	22.16	22.42
	ARIMA	45.35	46.26	46.67	46.93	47.10	47.57	47.85
	Linear Regression	74.59	75.52	75.65	75.97	76.11	77.08	77.69
	ACF-MLP	20.92	21.21	23.53	23.53	21.23	21.27	21.80
	ANOVA-MLP	42.31	40.17	39.90	25.48	41.21	21.57	21.52
	RFE	70.23	70.12	70.59	71.19	71.02	72.28	72.58
	Ridge	70.42	70.83	71.09	71.37	71.61	72.51	73.19
Water Level 6 kbits - 543 kbits 25000 samples	FSF-MLP	34.41	34.55	34.52	34.44	**34.31**	34.43	**33.60**
	FSF-LSTM	35.16	**34.06**	**34.04**	**33.92**	34.35	34.35	34.34
	MLP	35.88	35.67	36.72	34.96	35.83	35.96	37.80
	LSTM	34.96	34.98	34.51	34.22	36.09	34.57	34.65
	1D CNN	35.96	34.89	34.44	34.35	35.61	34.90	35.66
	ARIMA	76.02	76.02	76.02	76.01	76.02	76.05	76.07
	Linear Regression	76.20	76.21	76.21	76.21	76.21	76.22	76.22
	ACF-MLP	34.35	34.34	34.35	34.35	34.35	**34.34**	34.34
	ANOVA-MLP	**34.34**	34.34	34.34	34.34	34.35	**34.34**	34.34
	RFE	76.20	76.20	76.21	76.21	76.21	76.22	76.22
	Ridge	76.20	76.21	76.21	76.21	76.21	76.22	76.22

TABLE 5: CROSS-VALIDATION $^1/_2$-sMAPE RESULTS FOR THE VBA DATA SETS

Data Sets	Models	K = 1	K = 2	K = 3	K = 4	K = 5	K = 10	K = 15
NO_2 max. 6 kbits 2288 samples	FSF-MLP	18.24	24.81	27.72	**29.31**	**30.84**	33.30	**33.57**
	FSF-LSTM	**17.73**	**22.28**	28.67	30.94	31.48	33.42	33.86
	MLP	36.97	39.26	41.35	34.85	34.62	33.95	34.13
	LSTM	19.96	27.30	**24.24**	30.42	31.82	32.73	37.48
	1D CNN	44.08	34.14	43.11	36.97	35.63	36.19	33.99
	ARIMA	81.15	82.77	83.57	84.02	84.31	85.10	85.60
	Logistic Regression	32.45	34.98	34.61	34.41	33.95	34.08	33.92
	ACF-MLP	33.86	33.86	33.85	33.84	33.85	33.87	33.87
	ANOVA-MLP	19.74	25.90	29.58	29.74	31.74	**32.52**	33.70
	RFE	32.96	34.85	34.54	34.37	34.25	34.06	33.98
	Ridge	33.27	33.79	33.81	33.91	33.91	34.07	34.13
Temperature max. 11 kbits 7371 samples	FSF-MLP	29.55	29.85	29.62	29.76	31.81	30.39	30.84
	FSF-LSTM	**29.08**	28.94	29.55	29.48	**29.53**	30.40	31.00
	MLP	38.15	39.04	39.83	32.48	34.70	33.91	36.18
	LSTM	29.15	**28.50**	28.70	31.49	32.07	30.72	33.17
	1D CNN	36.97	41.39	36.28	40.91	41.86	36.59	40.92
	ARIMA	61.21	61.98	62.25	62.25	62.46	63.43	63.48
	Logistic Regression	30.40	30.41	30.86	31.02	31.30	32.64	33.60
	ACF-MLP	40.17	40.13	40.07	40.02	40.17	40.04	40.23
	ANOVA-MLP	30.90	29.75	**27.72**	**27.42**	30.98	**29.83**	**29.95**
	RFE	30.24	30.32	30.53	31.78	32.69	34.85	36.12
	Ridge	31.96	31.99	32.22	32.40	32.63	33.621	34.53
Elevator Button max. 34 kbits 40080 samples	FSF-MLP	12.96	**12.96**	**12.99**	**13.08**	**13.08**	**13.25**	**13.20**
	FSF-LSTM	13.09	13.15	13.17	13.19	13.32	13.32	13.25
	MLP	15.41	15.71	14.10	15.29	15.11	13.77	14.59
	LSTM	13.22	13.15	13.43	13.85	13.39	13.51	13.42
	1D CNN	16.91	16.28	15.65	16.96	14.65	15.69	13.74
	ARIMA	29.83	30.17	30.55	31.24	31.66	32.97	33.85
	Logistic Regression	12.84	12.98	13.04	13.12	13.19	13.36	13.45
	ACF-MLP	13.51	13.48	13.47	13.39	13.43	13.46	13.47
	ANOVA-MLP	**12.80**	13.06	13.09	13.25	13.88	13.35	13.39
	RFE	12.84	12.97	13.07	13.12	13.20	13.41	13.49
	Ridge	13.14	13.20	13.23	13.25	13.27	13.33	13.35

TABLE 6: CROSS-VALIDATION MISCLASSIFICATION RESULTS FOR THE FBA DATA SETS

Data Sets	Models	K = 1	K = 2	K = 3	K = 4	K = 5	K = 10	K = 15
NMHC 34 kbits 7715 samples	FSF-MLP	0.416	0.415	0.414	0.417	0.416	0.414	0.417
	FSF-LSTM	0.413	**0.412**	**0.412**	**0.412**	0.413	0.413	**0.412**
	MLP	0.462	0.414	**0.412**	0.417	0.421	0.417	**0.412**
	LSTM	0.469	0.466	**0.412**	0.426	**0.412**	0.415	0.417
	1D CNN	0.444	0.421	0.443	**0.412**	0.416	**0.412**	0.413
	ARIMA	0.490	0.493	0.492	0.492	0.492	0.493	0.495
	Logistic Regression	0.419	0.419	0.420	0.421	0.419	0.421	0.421
	ACF-MLP	**0.412**	**0.412**	0.413	**0.412**	0.413	**0.412**	0.413
	ANOVA-MLP	0.465	0.415	0.439	0.413	**0.412**	**0.412**	0.413
	RFE	**0.412**	**0.412**	**0.412**	**0.412**	**0.412**	**0.412**	**0.412**
	Ridge	0.417	0.417	0.417	0.417	0.417	0.417	0.418
Smart Home Energy Generation 16 kbits 8692 samples	FSF-MLP	0.258	**0.291**	0.315	0.330	**0.333**	0.357	0.366
	FSF-LSTM	0.258	0.297	0.315	**0.324**	0.338	0.360	0.363
	MLP	0.332	0.3802	0.389	0.428	0.423	0.401	0.402
	LSTM	0.272	0.367	0.331	0.352	0.377	0.379	0.379
	1D CNN	0.448	0.448	0.402	0.425	0.444	0.402	0.436
	ARIMA	0.268	0.319	0.341	0.352	0.359	0.371	0.376
	Logistic Regression	0.258	0.296	0.319	0.334	0.343	0.362	0.369
	ACF-MLP	0.411	0.403	0.408	0.408	0.402	0.401	0.402
	ANOVA-MLP	0.282	0.315	0.334	0.369	0.376	0.378	0.382
	RFE	**0.255**	0.299	**0.313**	0.326	0.335	**0.353**	**0.359**
	Ridge	0.258	0.293	0.318	0.332	0.341	0.359	0.367
Wind Speed 64 kbits 25000 samples	FSF-MLP	0.356	0.360	0.358	0.357	0.359	0.364	**0.365**
	FSF-LSTM	0.356	0.355	0.360	0.360	0.361	**0.362**	0.366
	MLP	0.369	0.393	0.369	0.369	0.369	0.369	0.369
	LSTM	**0.272**	0.367	0.362	**0.352**	0.377	0.371	0.379
	1D CNN	0.402	0.379	0.387	0.379	0.369	0.405	0.369
	ARIMA	0.390	0.389	0.389	0.389	0.387	0.380	0.378
	Logistic Regression	0.354	0.357	0.358	0.359	0.361	0.365	0.368
	ACF-MLP	0.369	0.369	0.369	0.369	0.369	0.369	0.369
	ANOVA-MLP	0.373	0.359	0.374	0.371	0.377	0.376	0.369
	RFE	0.352	**0.352**	**0.354**	0.356	**0.357**	0.368	0.368
	Ridge	0.354	0.357	0.358	0.359	0.359	0.364	0.366

[17] S. Yang, Z. Zhang, L. Fan, T. Xia, S. Duan, C. Zheng, X. Li, and H. Li, "Long-term prediction of significant wave height based on SARIMA model in the South China Sea and adjacent waters," IEEE Access, vol. 7, pp. 88 082–88 092, 2019.

[18] J. H. F. Flores, P. M. Engel, and R. C. Pinto, "Autocorrelation and partial autocorrelation functions to improve neural networks models on univariate time series forecasting," in The 2012 International Joint Conference on Neural Networks (IJCNN). IEEE, 2012, pp. 1–8.

[19] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in Proceedings of the 20th international conference on machine learning (ICML-03), 2003, pp. 856–863.

[20] A. Ahmad, N. Javaid, N. Alrajeh, Z. A. Khan, U. Qasim, and A. Khan, "A modified feature selection and artificial neural network-based day-ahead load forecasting model for a smart grid," Applied Sciences, vol. 5, no. 4, pp. 1756–1772, 2015.

[21] A. Kavousi-Fard, "A new fuzzy-based feature selection and hybrid TLA–ANN modelling for short-term load forecasting," Journal of Experimental & Theoretical Artificial Intelligence, vol. 25, no. 4, pp. 543–557, 2013.

[22] M. E. Aminanto, R. Choi, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, "Deep abstraction and weighted feature selection for Wi-Fi impersonation detection," IEEE Transactions on Information Forensics and Security, vol. 13, no. 3, pp. 621–636, 2017.

[23] M. Sheikhan and N. Mohammadi, "Neural-based electricity load forecasting using hybrid of GA and ACO for feature selection," Neural Computing and Applications, vol. 21, no. 8, pp. 1961–1970, 2012.

[24] Z. Hu, Y. Bao, R. Chiong, and T. Xiong, "Mid-term interval load forecasting using multi-output support vector regression with a memetic algorithm for feature selection," Energy, vol. 84, pp. 419–431, 2015.

[25] J. Deraeve and W. H. Alexander, "Fast, accurate, and stable feature selection using neural networks," Neuroinformatics, vol. 16, no. 2, pp. 253–268, 2018.

[26] T. Niu, J. Wang, H. Lu, W. Yang, and P. Du, "Developing a deep learning framework with two-stage feature selection for multivariate financial time series forecasting," Expert Systems with Applications, vol. 148, p. 113237, 2020.

[27] C. Feng, M. Cui, B.-M. Hodge, and J. Zhang, "A data-driven multi-model methodology with deep feature selection for short-term wind forecasting," Applied Energy, vol. 190, pp. 1245–1257, 2017.

[28] Z. Hu, Y. Bao, T. Xiong, and R. Chiong, "Hybrid filter–wrapper feature selection for short-term load forecasting," Engineering Applications of Artificial Intelligence, vol. 40, pp. 17–27, 2015.

[29] Ö. Uncu and I. Türkşen, "A novel feature selection approach: combining feature wrappers and filters," Information Sciences, vol. 177, no. 2, pp. 449–466, 2007.

[30] O. Abedinia, N. Amjady, and H. Zareipour, "A new feature selection technique for load and price forecast of electrical power systems," IEEE Transactions on Power Systems, vol. 32, no. 1, pp. 62–74, 2016.

[31] S. F. Crone and N. Kourentzes, "Feature selection for time series prediction–A combined filter and wrapper approach for neural networks," Neurocomputing, vol. 73, no. 10-12, pp. 1923–1936, 2010.

[32] N. Tang, S. Mao, Y. Wang, and R. Nelms, "Solar power generation forecasting with a LASSO-based approach," IEEE Internet of Things Journal, vol. 5, no. 2, pp. 1090–1099, 2018.

[33] C. M. van der Walt and N. Botha, "A comparison of regression algorithms for wind speed forecasting at Alexander Bay," in 2016 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech). IEEE, 2016, pp. 1–5.

[34] W. Liu, Z. Dou, W. Wang, Y. Liu, H. Zou, B. Zhang, and S. Hou, "Short-term load forecasting based on elastic net improved GMDH and difference degree weighting optimization," Applied Sciences, vol. 8, no. 9, p. 1603, 2018.

[35] L. Bronchal, Venezia Water Level Data Set. [Online]. Available: https://www.kaggle.com/lbronchal/venezia

[36] U. of California Irvine Machine Learning Repository, Air Quality Data Set. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/air+quality

[37] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. Di Francia, "On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario," Sensors and Actuators B: Chemical, vol. 129, no. 2, pp. 750–757, 2008.

[38] T. Singhl, Smart Home Energy Generation Data Set. [Online]. Available: https://www.kaggle.com/taranvee/smart-home-dataset-with-weather-information

[39] B. Erisen, Wind Speed Data Set. [Online]. Available: https://www.kaggle.com/berkerisen/wind-turbine-scada-dataset

[40] R. G. Lomax, Statistical concepts: A second course. Lawrence Erlbaum Associates Publishers, 2007.

[41] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with Python," in Proceedings of the 9th Python in Science Conference, vol. 57. Austin, TX, 2010, p. 61.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

MERT NAKIP obtained his B.Sc. degree, with graduation rank #1, from the Electrical-Electronics Engineering at Yaşar University (Izmir, Turkey) in 2018. His design of a multi-sensor fire detector via machine learning methods was ranked #1 nationally at the Industry-Focused Undergraduate Graduation Projects Competition organized by TÜBİTAK (Turkish Scientific and Technological Research Council). He completed his M. Sc. thesis in Electrical-Electronics Engineering at Yaşar University (Izmir, Turkey) in 2020. His thesis focused on the application of machine learning methods to IoT and was supported by the National Graduate Scholarship Program of TÜBİTAK 2210C in High-Priority Technological Areas. He is currently a Research Assistant and a Ph.D. candidate at the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences (Gliwice, Poland).

KUBİLAY KARAKAYALI obtained his B.Sc. degree in the Mathematics for Computer Science program in the Department of Mathematics at Ege University (Izmir, Turkey) in 2018. He has been working as the team leader of Explainable Artificial Intelligence and Augmented Intelligence at ETECube (Izmir, Turkey). Simultaneously, he is pursuing his M.Sc. thesis at the International Computer Institute at Ege University. His thesis focuses on the development of locomotion of the humanoid robot based on reinforcement learning. His current research interests are time-series forecasting, reinforcement learning and computer vision.

CÜNEYT GÜZELİŞ received the B.Sc., M.Sc., and Ph.D. degrees in Electrical Engineering from Istanbul Technical University (Istanbul, Turkey) in 1981, 1984, and 1988, respectively. He was a visiting researcher and lecturer between 1989 and 1991 in the Department of Electrical Engineering and Computer Sciences at the University of California, Berkeley (Berkeley, CA). He served as a full-time faculty member at Istanbul Technical University from 1991 to 2000, where he became full professor in 1998. He was Professor of Electrical and Electronics Engineering at Dokuz Eylül University (İzmir, Turkey) from 2000 to 2011, where he has served as the Dean of the Faculty of Engineering, and at İzmir University of Economics (İzmir, Turkey) from 2011 to 2015, where he has served as the Director of the Graduate School of Natural and Applied Science. Since 2015, he has been Professor of Electrical and Electronics Engineering at Yaşar University (İzmir, Turkey), where he has served as the Director of the Graduate School. He has supervised 17 M.S. and 14 Ph.D. students and published over 50 SCI indexed journal papers, 6 peer-reviewed book chapters, and more than 80 peer-reviewed conference papers. He has participated in over 20 scientific research projects funded by national and international institutions, such as the British Council and the French National Council for Scientific Research. His research interests include artificial neural networks, biomedical signal and image processing, nonlinear circuits-systems and control as well as educational systems.

VOLKAN RODOPLU obtained his B.S. degree (summa cum laude) in Electrical Engineering from Princeton University in 1996, and his M.S. and Ph.D. degrees in Electrical Engineering from Stanford University in 1998 and 2003, respectively. He has worked for the Wireless Research Division of Texas Instruments (Dallas, TX) and for Tensilica Inc. (Santa Clara, CA). He served as an Assistant Professor of Electrical Engineering at the University of California Santa Barbara from 2003 to 2009, where he was promoted to the position of Associate Professor with tenure. He is currently Professor of Electrical Engineering at Yaşar University (Izmir, Turkey) and Marie Skłodowska-Curie Fellow of the European Commission. His current research focuses on the Internet of Things, predictive networks, smart cities, and visible light communication. He is the winner of the TUBITAK (Scientific and Technological Research Council of Turkey) Achievement Award, the Research Achievement Award at Yaşar University, the National Science Foundation CAREER Award (USA), the University of California Regents' Junior Faculty Fellowship, Stanford Department of Electrical Engineering Outstanding Service Award, Stanford Graduate Fellowship (as Andreas Bechtolsheim Fellow), Stanford Department of Electrical Engineering Fellowship, the John W. Tukey Award from the American Statistical Association, G. David Forney Award, and the George B. Wood Legacy Prize.