YAŞAR UNIVERSITY

GRADUATE SCHOOL

MASTER'S THESIS

JOINT FORECASTING-SCHEDULING FOR THE INTERNET OF THINGS VIA SUBSPACE-BASED APPLICATION-SPECIFIC ERROR METRIC EMULATION

ALPEREN HELVA

THESIS ADVISOR: PROF. VOLKAN RODOPLU
CO-ADVISOR: PROF. CÜNEYT GÜZELİŞ

ELECTRICAL AND ELECTRONICS ENGINEERING

PRESENTATION DATE: 25.08.2021

BORNOVA / İZMİR
AUGUST 2021

ABSTRACT

JOINT FORECASTING-SCHEDULING FOR THE INTERNET OF THINGS VIA SUBSPACE-BASED APPLICATION-SPECIFIC ERROR METRIC EMULATION

HELVA, Alperen
MSc, Electrical and Electronics Engineering
Advisor: Prof. Volkan RODOPLU
Co-Advisor: Prof. Cüneyt GÜZELİŞ
August 2021

The massive access problem refers to the challenge posed in uplink wireless communication from a massive number of Internet of Things (IoT) devices to an IoT gateway, base station or access point. In this thesis, first, we present an Application-Specific Error Function (ASEF), which measures the impact of the forecasting error on network performance for Joint Forecasting-Scheduling (JFS). Second, we propose a Neural Network (NN)-based emulation of ASEF on a subspace of forecasting errors, which we call "Emulation of ASEF" (E-ASEF), and develop a novel algorithm, "Motion On a Subspace under Adaptive Learning rate" (MOSAL), which moves on this subspace of forecasting errors while minimizing the application-specific error metric at the output of MAC-layer scheduling. Our results show that MOSAL improves the performance of the JFS system while achieving a low execution time. This work paves the way to achieving high network performance at an IoT Gateway that has a massive number of IoT devices in its coverage area.

Keywords: Internet of Things (IoT), Forecasting, Scheduling, Massive Access Problem, Artificial Neural Network (ANN), Machine-to-Machine (M2M) Communication, Joint Forecasting-Scheduling, Subspace Learning

ÖZ

NESNELERİN İNTERNETİ İÇİN ALTUZAY TABANLI UYGULAMAYA ÖZGÜ HATA METRİĞİ ÖYKÜNMESİ İLE BÜTÜNLEŞİK TAHMİNLEME-ÇİZELGELEME

HELVA, Alperen
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği
Danışman: Prof. Dr. Volkan RODOPLU
Yardımcı Danışman: Prof. Dr. Cüneyt GÜZELİŞ
Ağustos 2021

Devasa erişim problemi, çok sayıda Nesnelerin İnterneti (IoT) cihazından bir IoT ağ geçidi, baz istasyonu veya erişim noktasına kadar olan kablosuz iletişimde ortaya çıkan zorluğu ifade eder. Bu tezde ilk olarak, Ortak Tahmin-Çizelgeleme (JFS) için tahmin hatasının ağ performansı üzerindeki etkisini ölçen bir Uygulamaya Özgü Hata Fonksiyonu (ASEF) sunuyoruz. İkinci olarak, "ASEF Öykünmesi" (E-ASEF) olarak adlandırdığımız tahmin hatalarının bir alt uzayı üzerinde Sinir Ağı (NN) tabanlı bir ASEF öykünmesi öneriyoruz ve geliştirdiğimiz “Uyarlanabilir Öğrenme Hızı Yöntemi ile Alt Uzayda Hareket” (MOSAL) olarak adlandırdığımız, MAC katmanı çizelgelemesinin çıktısında uygulamaya özel hata metriğini en aza indirirken tahmin hatalarının bu alt alanı üzerinde hareket eden yeni bir algoritma sunuyoruz. Sonuçlarımız, MOSAL’in düşük uygulama süresi elde ederken JFS sisteminin performansını iyileştirdiğini göstermektedir. Bu çalışma, kapsama alanında çok sayıda IoT cihazı bulunan bir IoT ağ geçidinde yüksek ağ performansı elde etmenin yolunu açmaktadır.

Anahtar Kelimeler: Nesnelerin İnterneti (IoT), Tahminleme, Çizelgeleme, Devasa Erişim Problemi, Yapay Sinir Ağı (ANN), Makinelerarası (M2M) İletişim, Bütünleşik Tahmin-Çizelgeleme (JFS), Altuzay Öğrenmesi

ACKNOWLEDGEMENTS

During the progress of this work, I would like to extend my sincere and heartfelt obligation towards all of the people who helped me in this endeavor. Firstly, I would like to thank my advisor Prof. Volkan Rodoplu and my co-advisor Prof. Cüneyt Güzeliş for providing guidance and feedback throughout this thesis.

I would like to thank my fiancee Şevval Aytaş for supporting me whenever I needed her. I wish to show my appreciation to my teammates and colleagues Mert Nakıp, Kubilay Karakayalı, Oytun Uzun and Onur Çopur for their helpfulness and support. I am also thankful to the faculty members in the Department of Electrical and Electronics Engineering at Yaşar University for all the considerate guidance. In addition, I would like to thank Michelle Rajotte at the Department of Foreign Languages at Yaşar University for her feedback on the writing in this thesis.

I would like to acknowledge the following institutions and programs that have provided employment or financial support during my Master's studies: Yaşar University, and the TÜBİTAK 1001 Master's Scholarship in Project # 118E277.

Finally, I cannot forget to thank my family for their unconditional support.

Alperen HELVA
İzmir, 2021

TEXT OF OATH

I declare and honestly confirm that my study, titled “JOINT FORECASTING-SCHEDU-LING FOR THE INTERNET OF THINGS VIA SUBSPACE-BASED APPLICATION-SPECIFIC ERROR METRIC EMULATION” and presented as a Master’s Thesis, has been written without applying to any assistance inconsistent with scientific ethics and traditions. I declare, to the best of my knowledge and belief, that all content and ideas drawn directly or indirectly from external sources are indicated in the text and listed in the list of references.

ALPEREN HELVA

August 8, 2021

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SYMBOLS AND ABBREVIATIONS

ABBREVIATIONS:

Abbreviation	**Meaning**
IoT	Internet of Things
ANN	Artificial Neural Network
M2M	Machine-to-Machine
ASEF	Application-Specific Error Function
E-ASEF	Emulation of ASEF
JFS	Joint Prediction-Scheduling
MAC	Medium Access Control
MOSAL	Motion On a Subspace under Adaptive Learning rate
ML	Machine Learning
QoS	Quality of Service
MLP	Multi Layer Perceptron
LSTM	Long short-term memory
Adaline	Adaptive Linear Neuron or later Adaptive Linear Element
BoF	Bank of Forecasters
PAL	Priority based on Average Load
MSE	Mean Square Error
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
sMAPE	symmetric Mean Absolute Percentage Error
MSA	Multi-Scale Algorithm
MC-LAPAL	Multi-Channel Look Ahead Priority based on Average Load
HSDPA	High Speed Downlink Packet Access
PCA	Principal Component Analysis
EML	Error Metric Learner
FBP	Fixed Bit Periodic
VBP	Variable Bit Periodic
FBA	Fixed Bit Aperiodic
VBA	Variable Bit Aperiodic

List of Symbols used in Chapters 1, 2 and 3

SYMBOLS:

Symbol	Meaning
G	Coverage Area of Gateway
$\mathscr{N}$	Set of N IoT devices
N	Number of IoT devices
T_{sch}	Duration of Scheduling Window
$\mathbf{X_{past}}$	Past Traffic Generation Pattern
$x_i[m-s]$	Entry of Input Matrix of device i, range $[0,m]$
$\hat{\mathbf{X}}$	Forecast Traffic Generation Pattern
$\hat{x}_i[m+k]$	Entry of $\hat{\mathbf{X}}$ at the kth Step in the Traffic Generation Pattern of Device i
$\tilde{\mathbf{X}}$	Accumulated Actual Past Traffic Generation Pattern
$x_i[m+k]$	Entry of $\tilde{\mathbf{X}}$ at the kth Step in the Traffic Generation Pattern of Device i
$\mathbf{S}(\hat{\mathbf{X}})$	Forecast Schedule Matrix
$\eta(\mathbf{S}(\hat{\mathbf{X}}),\tilde{\mathbf{X}})$	Forecast Network performance of Scheduler Based on $\mathbf{S}(\hat{\mathbf{X}})$ and $\tilde{\mathbf{X}}$
$\mathbf{S}(\tilde{\mathbf{X}})$	Accumulated Schedule Matrix
$\eta(\mathbf{S}(\tilde{\mathbf{X}}),\tilde{\mathbf{X}})$	Accumulated Network Performance of Scheduler Based on $\mathbf{S}(\tilde{\mathbf{X}})$ and $\tilde{\mathbf{X}}$
Φ	Throughput Difference
$\hat{\Phi}$	Estimated Throughput Difference
$\mathbf{D}$	Next Layer's Forecasting Error Difference Matrix
$\bar{d}$	Average Vector of $\mathbf{D}$
τ_{MAC}	Duration of a MAC-layer Slot
a	Adding a Constant Number of Bits
$\mathbf{A}^-$	Negative Vector of a
$\mathbf{A}^+$	Positive Vector of a
Φ^-	Outputs of E-ASEF for the Corresponding $\mathbf{A}^-$
Φ^+	Outputs of E-ASEF for the Corresponding $\mathbf{A}^+$
s	Index of a Scheduling Window
$x_i[s]$	Total Number of Bits Generated by Device $i \in \mathscr{N}$.
j	Burst
r	Generation Time
d	Deadline of r
Δ	Delay Constraint
p	Processing Time

Continued on the next page

Continuation of the List of Symbols used in Chapters 1, 2 and 3

Symbol	Meaning
R	Uplink Data Rate
m	Current Slot
K	Total Traffic Step
$\mathscr{V}$	Subspace of D
$\tilde{R}$	Relation of α and Φ
f	Function of Relation $\tilde{R}$
$\mathbf{W}_i^t$	Connection Weights and Biases
$\mathbf{b}_i^t$	Biases of the Output Neurons at Training Step t and Device i
t	Training Step
$\mathscr{E}_{\text{over}}^t$	Average Overestimation Error of Forecaster$_i$ at t
$\mathscr{E}_{\text{under}}^t$	Average Underestimation Error of Forecaster$_i$ at t
$\mathscr{E}_{\text{under}}^{min}$	Absolute Value of the Minimum Underestimation Error on the Training Set
ζ	Learning Rate
θ	Positive Threshold
ζ^0	Value to Which ζ is Reinitialized
L	Number of Channel Taps
W	Bandwidth
κ	Multiply by the Coefficient of Normalized Variance of Each Real and Imaginary One-touch Part in the Wireless Channel Model
N_o	Ratio of the One-sided Power Spectral Density of All Devices in N to the Total Number of Bursts Successfully Transmitted Over the Entire Slot Channel Grid
P	Total Transmit Power of IoT Device
T	Traffic Generation Period

CHAPTER 1

INTRODUCTION

The main purpose of this thesis is the design of subspace-based application-specific error metric emulation for Joint Forecasting-Scheduling (JFS). In this chapter, first, we introduce the concept of the Internet of Things (IoT), Machine-to-Machine (M2M) communication. Second, we describe the Massive Access Problem. Third, we summarize the JFS methodology, for which we have specially designed the error metric emulation in this thesis. Fourth, we distinguish our work from the past articles in the literature. Fifth, we give an outline of this thesis.

1.1 IoT and M2M Communication

IoT refers to that part of the Internet, which is made up of "things", namely, devices that act without human intervention. In this regard, traditional devices, such as laptops, cell phones and tablets, all of which are human-operated, are not part of IoT, but devices such as smart utility meters, tele-health devices, fleet management devices, smart lamp posts, and smart garbage bins are part of IoT.

As a result of the proliferation of IoT devices in recent years, we expect that there will be more machine-to-machine (M2M) communication, in which an IoT device communicates with a server, which also functions without human intervention. The number of IoT devices per base station is expected to increase from 5,000 to 35,000 devices (Vodafone, Apr. 2010). The overall number of IoT devices, which was 8.6 billion in 2018, is estimated to increase to 22.3 billion in 2024 (Cerwall et al., 2015). The problem of giving access to a massive number of IoT devices in the coverage area of an IoT gateway, base station or access point is called the *Massive Access Problem*.

1.2 Massive Access Problem

IoT device types, that are broadly referred to as "massive IoT" and for which the Massive Access Problem is posed typically have data rates or data volumes of these devices that are lower than the devices in other segments (such as broadband or critical IoT devices). Although such massive IoT devices do not create a large traffic load for base stations, access points or gateways in terms of rate and volume, the number of connection attempts that massive IoT devices would make constitutes the essence of the Massive Access Problem. JFS is a methodology that has been specifically designed so that it will work on the MAC layer to obviate such a massive number of connection requests.

Much of the previous work on the Massive Access Problem (Ghavimi & Chen, 2014)(Lien, Liau, Kao, & Chen, 2011)(Lin, Lee, Cheng, & Chen, 2014)(Jin, Toor, Jung, & Seo, 2017) (J. Liu, Song, et al., 2017) (Tello-Oquendo, Leyva-Mayorga, et al., 2018) (Tello-Oquendo, Pacheco-Paramo, Pla, & Martinez-Bauset, 2018) (Aijaz & Aghvami, 2015)(Aijaz, Ping, Akhavan, & Aghvami, 2014) (Ali, Saad, & Rajatheva, 2018) (Paul & Rho, 2016) (Pang, Chao, Lin, & Wei, 2014) (Park, Kim, & Har, 2014) (Liang, Xu, Cao, & Jia, 2018) (Si, Yang, Chen, & Xi, 2014) (Shirvanimoghaddam, Dohler, & Johnson, 2017) (Zanella et al., 2013) (Y. Liu, Yuen, Cao, Hassan, & Chen, 2014) (Shahin, Ali, & Kim, 2018) (Alavikia & Ghasemi, 2018) is based on reactive solutions, in which the network reacts to the current traffic demand. Furthermore, all of these studies[1] use the assumption of "random arrivals" at the MAC layer. However, recent work (Nakip, Gül, Rodoplu, & Güzeliş, 2019) has demonstrated that the traffic generated by IoT devices in M2M communication is predictable. Based on this work, a novel technique, called Joint Forecasting-Scheduling (JFS) was proposed in (Nakip, Rodoplu, Güzeliş, & Eliiyi, 2019) and further developed in (Rodoplu, Nakıp, Eliiyi, & Güzelis, 2020) and (Rodoplu, Nakip, Qorbanian, & Eliiyi, 2020). In JFS, an IoT gateway predicts the future traffic that will be generated by each IoT device in its coverage area and schedules the uplink transmissions from these IoT devices in advance. Thus, JFS enables scheduling of future IoT traffic in order to eliminate contention and collision at IoT gateways and base stations. In this thesis, we develop novel algorithms for JFS, the details of which we shall describe in the upcoming chapters.

[1] In addition to these studies, the assumption of "random arrivals" is also seen in (Bockelmann et al., 2016) (Mobasheri, Kim, & Kim, 2020).

1.3 Review of Joint Forecasting-Scheduling (JFS)

JFS was first presented in (Nakip, Rodoplu, et al., 2019) at the 2019 IEEE Global Conference on the Internet of Things. The original JFS system has been designed to be implemented at an IoT gateway in order to forecast and schedule the future IoT traffic from IoT devices that fall in the coverage area of this IoT gateway. JFS aims to maximize the total number of bits in successfully delivered bursts without any collision and resulting data loss. In this work, in order to achieve this aim, we model the throughput difference between that under forecasts and that under perfect forecasting. Furthermore, we design an ANN to emulate this throughput difference.

The JFS system consists of a Bank of Forecasters (BoF) and a Scheduler as the two main modules. These main modules will be described below.

1.3.1 Bank of Forecasters (BoF)

Figure 1.1. BoF of JFS

The BoF is the module in which the forecasting is performed in parallel for all of the devices in the coverage area of the IoT gateway. BoF has forecasters, each of which forecasts the future traffic generation pattern of a distinct IoT device. In this work, we use the Multi-Layer Perceptron (MLP) for each forecaster. As can be seen in the Fig. 1.1, the past traffic generation pattern is the input to the BoF. In addition, the output of the BoF is fed into the Scheduler module.

1.3.2 Scheduler

The Scheduler schedules the forecast traffic across the all of the forecasters in the JFS system. In this work, the Scheduler uses the Priority based on Average Load (PAL) heuristic scheduling method, whose output is a Schedule matrix. While

Figure 1.2. Scheduler of JFS

creating the matrix, the forecast traffic generation pattern produced in the BoF is used as input as shown in Fig. 1.2. The schedule, which appears as the output, indicates to which device each MAC layer is allocated (or left idle) over the duration of a scheduling window.

1.3.3 Measurement of Network Performance under JFS

In this subsection, we shall describe the performance benchmark of JFS. The performance metric, called "throughput", is defined to be the ratio of the total number of bits in successfully transmitted bursts to the total number of bits in the traffic that all of the devices offer to the network over a scheduling window.[2] The method that we have developed in this thesis, which can improve network performance, is a general framework that can be used for different performance metrics, although it is specific to throughput in our particular treatment.

In this thesis, ASEF makes use of the scheduling results of wireless communication sources at the MAC layer. It examines the relationship between the scheduler that measures the network performance in the JFS system and the forecaster that gives the forecasting error result. Learning the result of the relationship between them, ASEF measures the effect of the results on the forecasting algorithm in the JFS system.

1.3.4 Relationship between Forecasting Error and Network Performance

The proactive nature of the JFS system described above allows resources to be pre-allocated based on IoT traffic forecasts to maximize network throughput. As a result of studies (Nakip, Rodoplu, et al., 2019; Rodoplu, Nakıp, et al., 2020; Rodoplu, Nakip, et al., 2020) evaluating JFS performance, JFS performed well in terms of network

[2]We point out that this definition of throughput differs from the standard definition of throughput.

throughput and energy consumption. However, the source of its performance is the forecasting scheme that it uses. Therefore, the interaction between forecasting error and network performance directly affects JFS performance. For this reason, the main purpose of this thesis is to examine the use of this relationship to maximize the efficiency of JFS.

Forecasting error metrics, which are used to observe and minimize forecasting errors in the literature, such as Mean Square Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) or symmetric Mean Absolute Percentage Error (sMAPE), provide analysis for general errors. However, these metrics are not developed specifically for each application and do not take these differences into account.[3] To avoid such problems, in this thesis we propose an Application-Specific Error Function (ASEF) that can use forecasting results. However, in order to design the working structure of the ASEF function, we also present the Emulation of Application-Specific Error Function (E-ASEF) in the thesis to enable learning according to ASEF by imitating the application on which the function will work.

1.4 Relationship to the State of the Art

In this section, we present the differences between our work and the works in the current literature in four categories: (1) the works that use the JFS system for resource allocation at the MAC layer; (2) the references that utilize traditional forecasting error metrics for IoT traffic at layers above the MAC layer; (3) the articles that perform subspace learning in machine learning models in contexts outside of IoT.[4] (4) the works that propose a custom training algorithm for forecasting aggregate network traffic.

First, Reference (Nakip, Gül, et al., 2019) showed that the traffic generation patterns of individual IoT devices have structure and can be predicted via machine learning models. Reference (Nakip, Rodoplu, et al., 2019) proposed the Joint Forecasting-Scheduling technique. Furthermore, the authors in (Rodoplu, Nakıp, et al., 2020) developed the Multi-Scale Algorithm (MSA), which extends the accurate forecasting horizon of the JFS system by aggregating the traffic patterns over multiple temporal scales. Reference

[3] In other forecasting systems, more than one traditional metric can be used to enable learning. In addition to this, the forecaster may not have any information about the applications to be able to forecast.

[4] To the best of the authors' knowledge, there do not currently exist works in the literature that perform subspace learning for the prediction of the traffic generation patterns of individual IoT devices.

(Rodoplu, Nakip, et al., 2020) developed multi-channel JFS, in which scheduling is scales. Reference (Rodoplu, Nakip, et al., 2020) developed multi-channel JFS, in which scheduling is performed over a slot-channel grid and proposed the Multi-Channel Look Ahead Priority based on Average Load (MC-LAPAL) heuristic. Even though these works represent significant advances, the forecasters therein have been trained in order to minimize the traditional MSE metric. In contrast, in this work, we develop a novel application-specific error metric, which we emulate via an ANN, that quantifies the impact of forecasting on the scheduling performance. Furthermore, we develop a novel algorithm that moves on the subspace of forecasting errors while minimizing the application-specific error metric.

Second, we note that Reference (Nakip, Gül, et al., 2019) trains forecasters for individual IoT devices by minimizing MSE and measures performance via the sMAPE. Reference (Lopez-Martin, Carro, & Sanchez-Esguevillas, 2019) forecasts the total traffic volume in an IoT network via a neural network model and measures the performance of the neural network via traditional forecasting error metrics such as R^2, MSE, and MAE. Reference (Lopez-Martin, Carro, & Sanchez-Esguevillas, 2020) uses gradient boosting neural networks to forecast the type of the IoT traffic and measures the forecasting performance via traditional classification metrics such as accuracy and the F1 score. While the works in this category use traditional performance metrics during both training and testing of the forecasting models, we use a novel application-specific error metric that measures the performance at the output of the application that utilizes the forecasts.

Third, Reference (Pehlevan, Hu, & Chklovskii, 2015) performs subspace learning by using a hebbian/anti-hebbian neural network for which the input data are projected onto the principal subspace. Furthermore, subspace learning has been used for pattern recognition in (Jiang, 2011), for spectral regression in (Cai, He, & Han, 2007) and for computer vision via robust Principal Component Analysis (PCA) in (Li, 2004) and (De La Torre & Black, 2003). In (Gu, Li, & Han, 2011) and (Wang, He, Wang, Wang, & Tan, 2015), the authors perform feature selection and subspace learning jointly. Whereas all of these works project the inputs onto a subspace in order to reduce the input dimension, we train our model by using samples that reside on a *given* subspace such that the application-specific error function becomes learnable.

Fourth, Reference (Zhang & Patras, 2018) introduced an Ouroboros Training Scheme (OTS) that uses earlier forecasts when ground truth values are unavailable for forecasting the total cellular traffic under the High Speed Downlink Packet Access

(HSDPA) protocol. Reference (Lawal, Abdulkarim, Hassan, & Sadiq, 2016) combined the Firefly and Resilient propagation algorithms in order to train a neural network model for forecasting total cellular traffic volume. Whereas the algorithms in these works minimize well-known error metrics to train the forecaster, we propose a novel algorithm, namely Motion On a Subspace under Adaptive Learning rate (MOSAL), which enables the training of the forecaster by minimizing the estimated loss in the IoT network performance.

1.5 Outline of this Thesis

We now give an outline of the rest of this thesis: In Chapter 2, we describe a new methodology for investigating the relationship between forecasting error and network performance in JFS via the Application-Specific Error Function (ASEF). In Chapter 3, we describe a new approach, called "Emulation of ASEF (E-ASEF), in which an Artifi-cial Neural Network (ANN) is trained on a subspace to learn the difference in network performance. Furthermore, we develop our "Motion on a Subspace under Adaptive Learning rate" (MOSAL) algorithm in this chapter. In Chapter 4, we summarize the contributions of this thesis and describe new directions for our future work.

CHAPTER 2

Subspace-Based Emulation of the Relationship Between Forecasting Error and Network Performance in Joint Forecasting-Scheduling for the Internet of Things

2.1 Introduction

As we have discussed in Chapter 1, IoT is a key technology for the smart cities of the near future (Hasan, Hossain, & Niyato, 2013). The application areas of IoT widen every day (Bello & Zeadally, 2019)(Fortino, Russo, Savaglio, Viroli, & Zhou, 2017). These applications include fleet management, environmental monitoring and control systems, traffic controlling as well as smart buildings (Kuhlins, Rathonyi, Zaidi, & Hogan, Jan. 2020). Although IoT is one of the technologies that make the daily life of humans easier, the challenges that must be overcome in order to unleash IoT grow as the number of IoT devices increases. In the near future, it is expected that there will be 30 billion IoT devices on the Internet (Cisco, Mar. 2020). In addition, more than half of IoT devices are expected to fall in the Massive IoT segment (Ericsson, Nov. 2019), in which a base station or a gateway will cover a massive number of low-cost IoT devices. The connection requests of the massive number of IoT devices to a single gateway will result in a significant access problem in cellular networks (Ghavimi & Chen, 2014) (Zanella et al., 2013). This problem is referred to as the "Massive Access Problem" of IoT.

In order to solve the Massive Access Problem, Reference (Nakip, Rodoplu, et al., 2019) proposed Joint Forecasting-Scheduling (JFS), which is a machine learning based proactive resource allocation technique. JFS ensures that it allocates the resources in advance based on forecasts of IoT traffic in order to maximize the network throughput. The performance of JFS is evaluated in a number of works (Nakip, Rodoplu, et al., 2019; Rodoplu, Nakıp, et al., 2020; Rodoplu, Nakip, et al., 2020). The results of these works show that although JFS performs well in terms of network throughput and energy consumption, its performance depends highly on the performance of the forecasting

scheme that it utilizes. Thus, knowledge of the relationship between the forecasting error and network performance would provide crucial information in improving the performance of JFS. One of the major foci of this work[1] is the examination of this relationship.

The focus of the past literature on forecasting has been to minimize the forecasting error, which is captured by any of the forecasting error metrics such as Mean Square Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) or symmetric Mean Absolute Percentage Error (sMAPE). These traditional forecasting error metrics do not take into account the particular application that will utilize the forecasting metric.[2] In contrast, in this chapter, we propose an error metric that is specific to the application that will utilize the results of forecasting. In contrast with the traditional forecasting error metrics in which forecasting is decoupled from the application that utilizes the forecasts, in this chapter, we propose an Application-Specific Error Function (ASEF). In the particular example of JFS, which we examine in this chapter, the application that utilizes the forecasting results is MAC-layer scheduling of the wireless resources. In this regard, our aim is to discover the relationship between the forecasting error (which is measured at the output of the forecaster) and the network performance (which is measured at the output of the scheduler) for the JFS system. In this context, ASEF measures how the performance of the forecasting algorithm in JFS affects the resulting network performance.

In addition to the above conceptual contribution, the major methodological contribu-tion of this chapter to the literature is the emulation of the relationship between the forecasting error and the network performance by an Artificial Neural Network (ANN). In our investigations, we have found that it is extremely difficult for an ANN to learn this relationship if the space of forecasting errors is taken to be the original space of all possible forecasting errors produced by the forecaster. One of our key contributions is the idea that if the forecasting errors are constrained to lie on a subspace of the original space of forecasting errors, then it becomes possible for the ANN to learn the relationship between the forecasting error and the network performance. In this work, we design an ANN that learns this relationship successfully on a subspace of the entire space of forecasting errors. Furthermore, we evaluate the performance of this ANN, which we call “Emulator for ASEF”, or

[1] The technical content in this chapter has been published as a conference paper (Nakip, Helva, Güzeliş, & Rodoplu, 2021b) at the IEEE Internet of Things World Forum (WF-IoT) 2021.

[2] In many instances, more than one application may utilize the metric. Furthermore, the forecaster might have no knowledge of which application will utilize the forecasts.

E-ASEF, for short, via simulations that show that E-ASEF achieves high performance with very fast execution time.

The rest of this chapter is organized as follows: In Section 2.2, we state our assumptions. In Section 2.3, we present ASEF for the JFS system. In Section 2.4, we describe the design of an ANN, called E-ASEF, that emulates ASEF on a subspace. In Section 2.5, we evaluate the performance of E-ASEF. In Section 2.6, we present our conclusions.

2.2 Assumptions

We assume[3] that there is a set of N IoT devices, denoted by N, in the coverage area of Gateway G. In addition, each device i in N has a direct wireless link to G.

We let m denote the index of a MAC-layer slot (in short, "slot"). Furthermore, we let $x_i[m]$ denote the total number of bits generated by device i in slot m. We assume that the generation of traffic by each IoT device occurs in bursts. Moreover, we assume that every time that an IoT device i transmits to G, it compresses its traffic pattern since its last transmission and sends this along with its actual data. For each IoT device, Gateway G pieces together these traffic patterns and thus has access to the entire past traffic pattern of that device. Gateway G then performs the k-step ahead forecast of the future traffic generation pattern for all values of $k \in \{1,\ldots,K\}$ for each IoT device. Based on these forecasts, G schedules the uplink transmissions from all of these IoT devices in advance.[4]

For a JFS system, the throughput is defined as the ratio of the total number of bits in successfully transmitted bursts to the total number of bits in offered traffic over a scheduling window of duration T_{sch}.[5] While the framework that we develop for ASEF is general and works for any network performance metric, throughput will be the main metric by which we measure network performance in this chapter.

2.3 Application Specific Error Function (ASEF) for JFS

In this section, we describe the ASEF for the JFS system. After we have defined ASEF, we shall emulate this function via an ANN in Section 2.4.

In Fig. 2.1, we present the block diagram through which we define ASEF for the JFS

[3]The assumptions of this work are identical to those in (Nakip, Rodoplu, et al., 2019), which describes the JFS system.

[4]For simplicity, we do not model mobile devices that change coverage areas in this work.

[5]We note that this differs from the traditional definition of throughput. In this definition, the throughput serves as a measure of efficiency of the JFS system; it measures the efficiency with which the system is able to deliver bits successfully to the Gateway at the MAC layer.

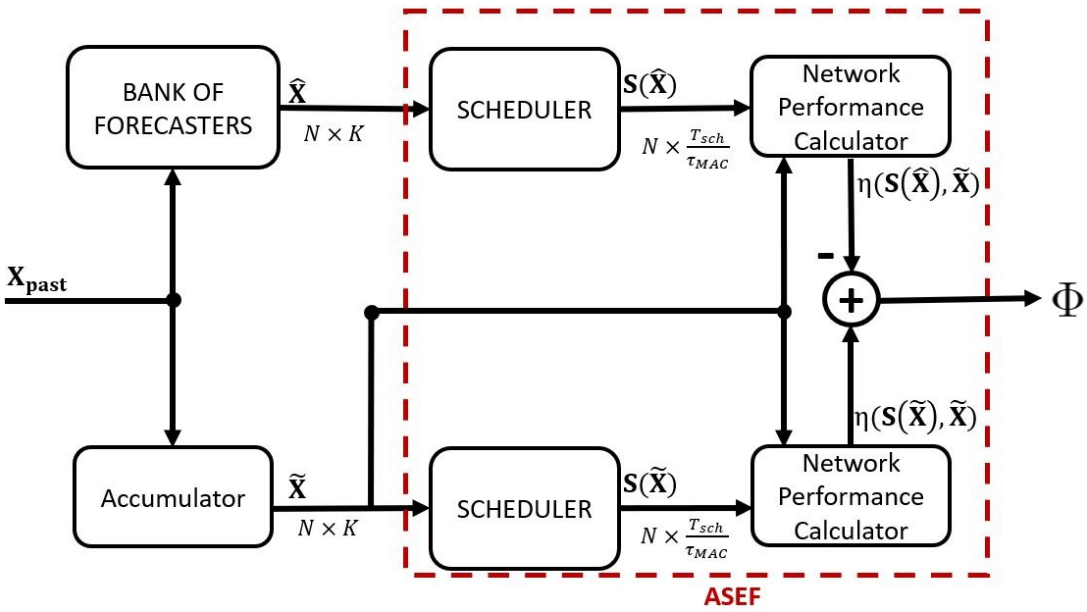

Figure 2.1. ASEF is defined as a "black box" by the set of components that appear in the dashed box in this figure. Our goal will be to emulate this ASEF via an ANN.

system. For the definition of ASEF, we use two JFS systems that run in parallel. The upper branch of this figure is a JFS system that consists of a Bank of Forecasters (BoF) and a Scheduler[6]. The lower branch of this figure is a JFS system under the actual traffic generation patterns (a.k.a. "perfect forecasts") of the devices. The ASEF is defined as the function that computes the difference between the performance of the application under forecasts versus the one under perfect forecasts.[7]

In this figure, at each discrete time m, $\mathbf{X}_{\mathbf{past}}$ denotes the input matrix of the BoF whose entry (i,s) is $x_i[m-s]$, where s can take values in the range of integers in $[0,m]$. The is the output matrix of the BoF whose entry (i,k) is the forecast of the number of bits at the kth step in the traffic generation pattern of device i, denoted by $\hat{x}_i[m+k]$, where k can take values in the range of integers $[1,K]$. In addition, the matrix $\tilde{\mathbf{X}}$ denotes the

---

developed in (Nakip, Rodoplu, et al., 2019). Although any scheduling algorithm can be used in our methodology, in this article, the reason that we use PAL is that the previous studies (Nakip, Rodoplu, et al., 2019; Rodoplu, Nakıp, et al., 2020) showed that PAL is a fast heuristic with relatively high performance.

[7] Note that in an actual system, the performance under perfect forecasts can be measured only after the traffic of each IoT device has been realized. The system diagram that appears in Fig. 2.1 is used for network simulation during which the inputs to the system as well as the performance difference at the output are collected as data that will be used later in order to train the ANN that will emulate the ASEF block shown in this figure.

actual past traffic generation pattern that has been accumulated, whose entry (i,k) is $x_i[m+k]$.

Furthermore, the red dashed box in Fig. 2.1 shows the ASEF for the JFS system, where the inputs of ASEF are $\tilde{\mathbf{X}}$ and $\hat{\mathbf{X}}$. In this figure, ASEF is comprised of three parts that appear in this red dashed box: the upper branch of the figure in the box; the lower branch of the figure in the box; and the difference between the outputs of the Network Performance Calculators that appear in this figure.

First, on the upper branch of ASEF in Fig. 2.1, based on the forecast future traffic generation matrix $\hat{\mathbf{X}}$, the Scheduler produces a schedule matrix, denoted by $\mathbf{S}(\hat{\mathbf{X}})$, whose entry (i,m) is the binary variable that equals 1 if MAC-slot m has been allocated to device i, and equals 0 otherwise. Then, the Network Performance Calculator in this figure calculates the network performance $\eta(\mathbf{S}(\hat{\mathbf{X}}),\tilde{\mathbf{X}})$ based on $\mathbf{S}(\hat{\mathbf{X}})$ and $\tilde{\mathbf{X}}$. Throughout this chapter, the performance metric will be the network throughput (as defined in Section 2.2).

Second, on the lower branch of ASEF in Fig. 2.1, we compute the network performance of JFS under perfect forecasts, which is denoted by $\eta(\mathbf{S}(\tilde{\mathbf{X}}),\tilde{\mathbf{X}})$. Note that perfect forecasts are available only after the actual traffic has been realized in practice. In our simulations, we use these perfect forecasts in order to measure the difference in the network performance attained under forecasting versus the one under the actual traffic realizations.

Third, after the operations in each of the upper and the lower branches are completed, we calculate the throughput difference, denoted by Φ, which is the output of the ASEF. Thus, the output of ASEF is

$$\Phi = \eta(\mathbf{S}(\tilde{\mathbf{X}}),\tilde{\mathbf{X}}) - \eta(\mathbf{S}(\hat{\mathbf{X}}),\tilde{\mathbf{X}}) \tag{2.1}$$

2.4 Emulation of ASEF (E-ASEF) via an ANN

Recall that the main objective in computing the ASEF is to pave the way to algorithms that will significantly improve the JFS system by taking the advantage of the knowledge of Φ. However, since ASEF requires computing the schedule matrices as well as the throughput metrics, the execution time of ASEF in its original form is too high for its use in practical algorithms that can improve the performance of the JFS system. In order to solve this problem, we now aim to emulate ASEF by an ANN. By using an ANN in the place of ASEF, we will generate Φ such that none of the blocks that appear

under ASEF, such as the Scheduler or the Network Performance Calculator will be needed. Thus, the ANN that replaces ASEF shall calculate the value of Φ based on the inputs to the ASEF.

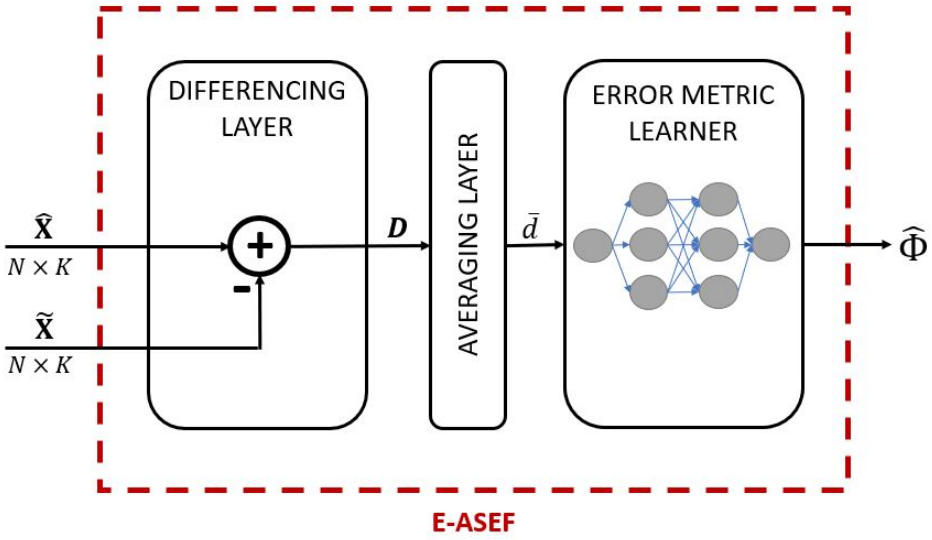

Figure 2.2. E-ASEF, which is the ANN-based emulator for ASEF

In Fig. 2.2, in the design of the emulator, the inputs are the matrices $\tilde{\mathbf{X}}$ and $\hat{\mathbf{X}}$, and the output is the value of Φ that will be estimated by the ANN. We shall denote this estimate by $\hat{\Phi}$. In our design, the E-ASEF consists of three layers: the Differencing Layer, the Averaging Layer and the Error Metric Learner (EML).

At the first layer, the emulator computes $\mathbf{D} = \hat{\mathbf{X}} - \tilde{\mathbf{X}}$ and passes the matrix $\mathbf{D}$ of the forecasting error to the next layer.

At the second layer, we take the average over all of the elements of $\mathbf{D}$, which generates $\bar{d}$. The reason that we use averaging is two-fold: (1) During our experimental work, we observed that the value of Φ is highly learnable for a subspace of ASEF on which the forecasting error is kept constant across all of the devices. That is, in this subspace, the value of $\bar{d}$ equals to the constant value of forecasting error; that is $\bar{d} = \mathbf{D}[i,k], \forall i,k$. (2) Instead of using $\tilde{\mathbf{X}}$ and $\hat{\mathbf{X}}$ to form the estimate $\hat{\Phi}$, using $\bar{d}$ significantly decreases the input dimension and hence the computational complexity of the model.

At the third layer, we use ANN as the EML in order to form the estimate $\hat{\Phi}$ based on $\bar{d}$. Since E-ASEF is a model that can be used in other algorithms to improve performance of JFS, the computational complexity of E-ASEF is as important as its learning performance. Thus, we aim to use a relatively simple ANN architecture in

the EML layer. To this end, we compare the Linear Perceptron (namely, Adaline), the Nonlinear Perceptron, and the Multi-Layer Perceptron (MLP). For each of these models, we select the architecture and the activation functions as follows: For the Linear Perceptron, note that there is no activation function or any other hyper-parameter. The Nonlinear Perceptron is comprised of a single neuron with an activation function that is selected as the tangent hyperbolic tanh[8]. For the MLP, we use a single hidden layer and an output layer, where the hidden layer contains 5 neurons. The activation function of each neuron in the MLP is selected as tanh.

2.5 Results

2.5.1 IoT Data

We use the dataset in (*IoT Traffic Generation Pattern Dataset*, 2021), which is comprised of the traffic generation patterns of 10000 bootstrapped IoT devices. According to the classification of Reference (Nakip, Gül, et al., 2019), there are four distinct IoT classes: Fixed Bit Periodic (FBP), Variable Bit Periodic (VBP), Fixed Bit Aperiodic (FBA) and Variable Bit Aperiodic (VBA). In this classification, "Fixed Bit" states that the IoT device generates a fixed number of bits in each burst while "Variable Bit" states otherwise. In addition, "Periodic" states that the inter-arrival time between the bursts of a device is constant while "Aperiodic" states otherwise. Since both number of bits and the generation time of each burst of each device in the FBP class are known in advance, forecasting is needed only for the VBP, FBA, and the VBA classes. Thus, in our simulations, we include only these three classes in order to obtain a system in which we evaluate the performance of E-ASEF.

2.5.2 Performance Evaluation of E-ASEF

We now evaluate the performance of E-ASEF under each of the Linear Perceptron, Nonlinear Perceptron and the MLP forecasting models on the following four distinct network set-ups: $\mathscr{N}$ is comprised of (1) equal percentages of VBP, FBA and VBA classes; (2) only devices in the VBP class; (3) only devices in the FBA class; (4) only devices in the VBA class.

[8] We selected the activation function as tanh because the value of Φ is in the range $[-1, 1]$ for our application.

For each network set-up, we aim to evaluate the performance in a subspace in which the forecasting error is constant across all of the forecasters. To this end, we replace each forecaster in the BoF by adding a constant number of bits, which we denote by a, to the actual traffic generation of each of the devices.[9] Thus, whenever $\hat{x}_i[m] = x_i[m] + a$, we say that the forecaster overestimates the value of the traffic generation pattern for device i if $a > 0$, and underestimates it if $a < 0$. (The estimate matches the actual whenever $a = 0$.) In our experiments, we increment a in multiples of 10 from the lowest to the highest value in the range $[-100, 100]$. In addition, we set the number of IoT devices in the coverage area of the Gateway G as $N = 2000$. We set the duration of a MAC-layer slot, denoted by τ_{MAC}, to 0.1 s.

In our experiments, we observed that the behavior of ASEF is distinct for each of the negative and positive values of a. For convenience, we separated the values of a into two categories as follows: We hold the negative values of a in the vector $\mathbf{A}^-$ and the positive values in the vector $\mathbf{A}^+$. Then, we let Φ^- denote the outputs of E-ASEF for the corresponding $\mathbf{A}^-$, and let Φ^+ denote those for $\mathbf{A}^+$. A single E-ASEF structure should not be expected to learn both the Φ^- and Φ^+ functions due to the structural differences that we observed between these functions in our experiments.[10] In this chapter, E-ASEF is trained separately for the Φ^- and the Φ^+ functions.

Training of E-ASEF

Based on the above observations, we shall now explain how we train E-ASEF. First, for each value of a, we simulate the JFS system for 1000 distinct scheduling windows, each of whose duration equals 15 minutes. For each scheduling window, we randomly select the IoT devices from each device class for the following four distinct experimental set-ups: (1) We select an equal number of devices from each of the VBP, FBA and the VBA classes. (2) All of the devices are in the VBP class. (3) All of the devices are in the FBA class. (4) All of the devices are in the VBA class. Let s be the index of a scheduling window. We shall append a superscript s to each of the variables that appear in Fig. 2.2 in order to indicate to which scheduling window that variable belongs. Now, for each experimental set-up, we extract the traffic generation patterns $\tilde{\mathbf{X}}^{(s)}$ from the dataset and calculate $\hat{\mathbf{X}}^{(s)}$ for a. Based on $\tilde{\mathbf{X}}^{(s)}$ and $\hat{\mathbf{X}}^{(s)}$, we compute the schedule via the PAL scheduling algorithm and calculate the network performance difference $\Phi^{-(s)}$

[9] The fact that this constant a does not have an i-index indicates that it is constant across all of the devices.

[10] The structural differences between Φ^- and Φ^+ will be shown in Fig. 2.3-2.6.

if $a < 0$ and calculate $\Phi^{+(s)}$ otherwise.

Then, for each s, the inputs of E-ASEF are $\hat{\mathbf{X}}^{(s)}$ and $\tilde{\mathbf{X}}^{(s)}$. The $\Phi^{-(s)}$ is the output in the case of underestimation, and $\Phi^{+(s)}$ is the output in the case of overestimation. The connection weights and the biases of EML are updated via backpropagation using the gradient descent algorithm in pattern mode.

Learning Performance of E-ASEF

Fig. 2.3 shows how each of the Linear Perceptron, the Nonlinear Perceptron and the MLP fits to each function Φ^- and Φ^+. In this figure, we see that MLP fits both functions almost perfectly. In addition, both the Linear Perceptron and the Nonlinear Perceptron can fit only the average of each of the underestimation and overestimation segments of the function. In this figure, the performance of E-ASEF under MLP shows that ASEF for JFS is highly predictable.

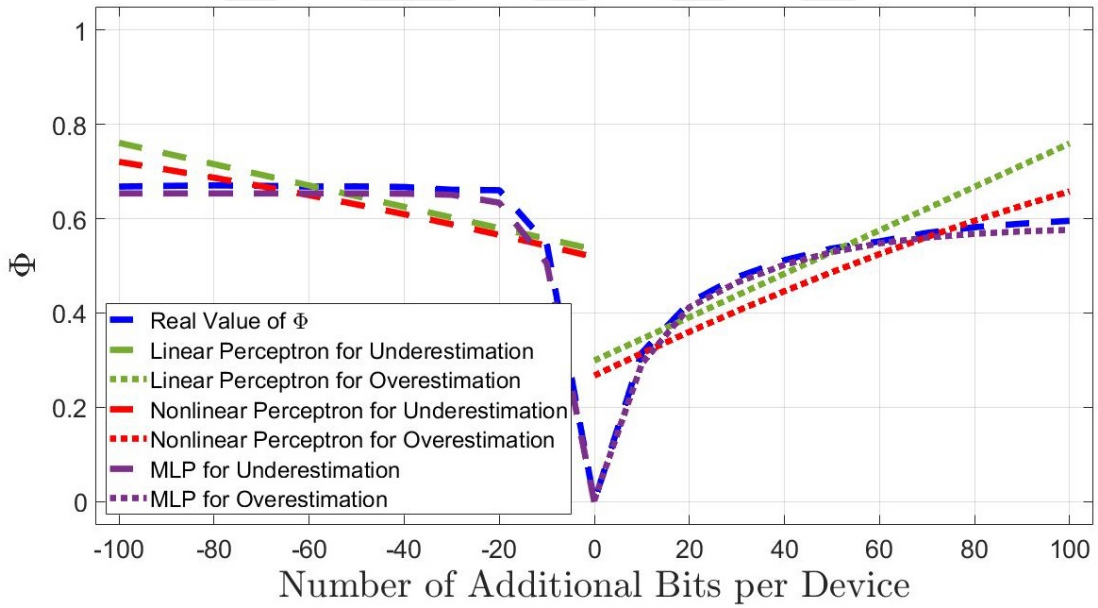

Figure 2.3. Learning performance of E-ASEF under each of the Linear Perceptron, Nonlinear Perceptron, and MLP for the throughput difference Φ under the network set-up that has 1/3 VBP, 1/3 FBA and 1/3 VBA as the fractions in each device class.

In Fig. 2.4, for the simulation that is comprised of IoT devices, each of which falls in the VBP class, we see that the MLP is the best predictor for the majority of the values of a. However, for E-ASEF under MLP, the difference between $\hat{\Phi}$ and Φ slowly increases

as the $|a|$ increases. In addition, we see that since the Linear Perceptron fits well to the flat segment of the function which corresponds to Φ^-, it is not able capture the sharp decrease at $a = -10$. On the other hand, the Nonlinear Perceptron is not able to capture Φ^-, but it successfully captures the sharp decrease at $a = -10$.

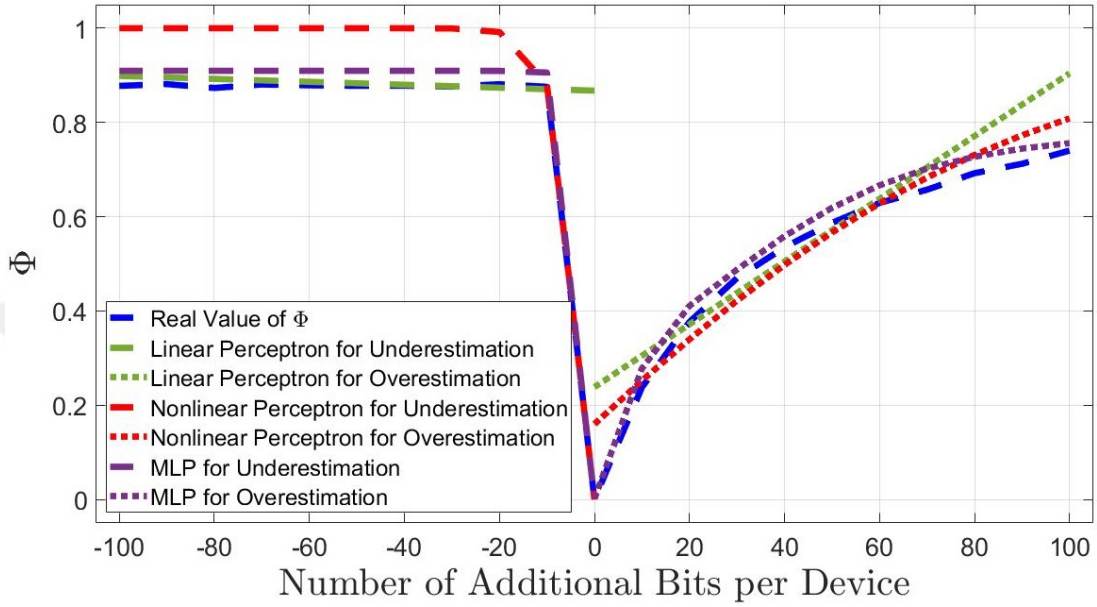

Figure 2.4. The learning performance of E-ASEF under each of the Linear Perceptron, Nonlinear Perceptron, and MLP for the throughput difference Φ for a network that is constituted by only VBP devices.

In Fig. 2.5, for the simulation that is comprised of the IoT devices, each of which falls in the FBA class, we see that MLP predicts Φ almost perfectly and achieves the most accurate result compared with those for the Linear Perceptron and the Nonlinear Perceptron models for the majority of the values of a.

In Fig. 2.6, we present the performance of E-ASEF under each of the Linear Perceptron, Nonlinear Perceptron, and the MLP for the simulation that is comprised of the IoT devices each of which falls in VBA class. As seen in this figure, $\hat{\Phi}$ under MLP is almost equal to Φ, where MLP is able to predict the sharp decrease between $a = -10$ and $a = 0$. In addition, while the Linear Perceptron fits to the mean of each of Φ^- and Φ^+ and the Nonlinear Perceptron also fits close to the mean of each of those, the MLP is the best-performing model as it is for all of the other simulation set-ups.

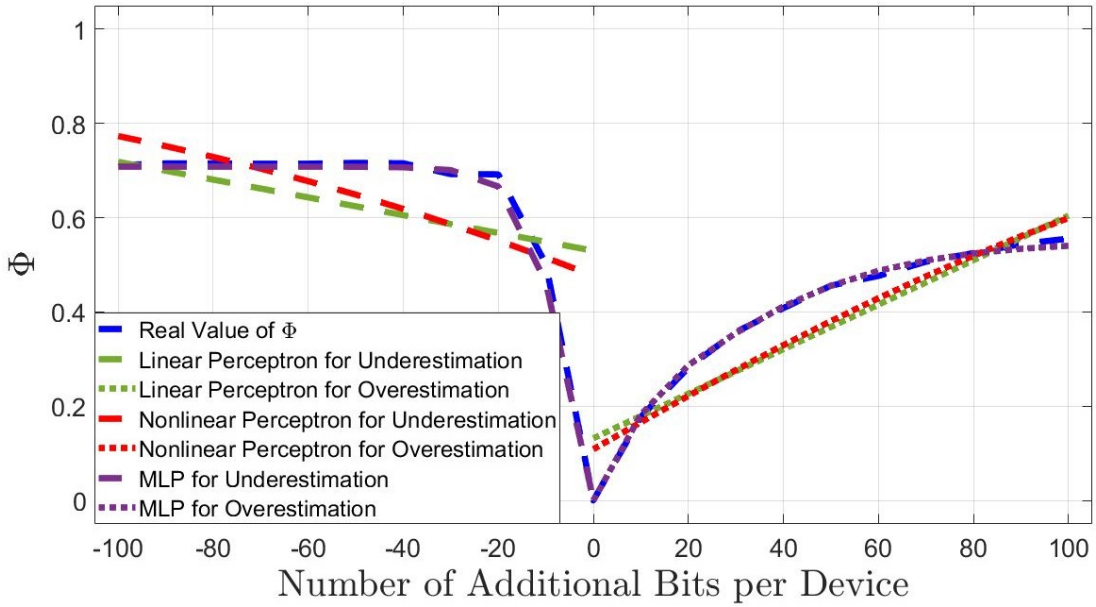

Figure 2.5. Learning performance of E-ASEF under each of the Linear Perceptron, Nonlinear Perceptron, and the MLP for the throughput difference Φ under the network set-up that has 100% of the IoT devices in the FBA class.

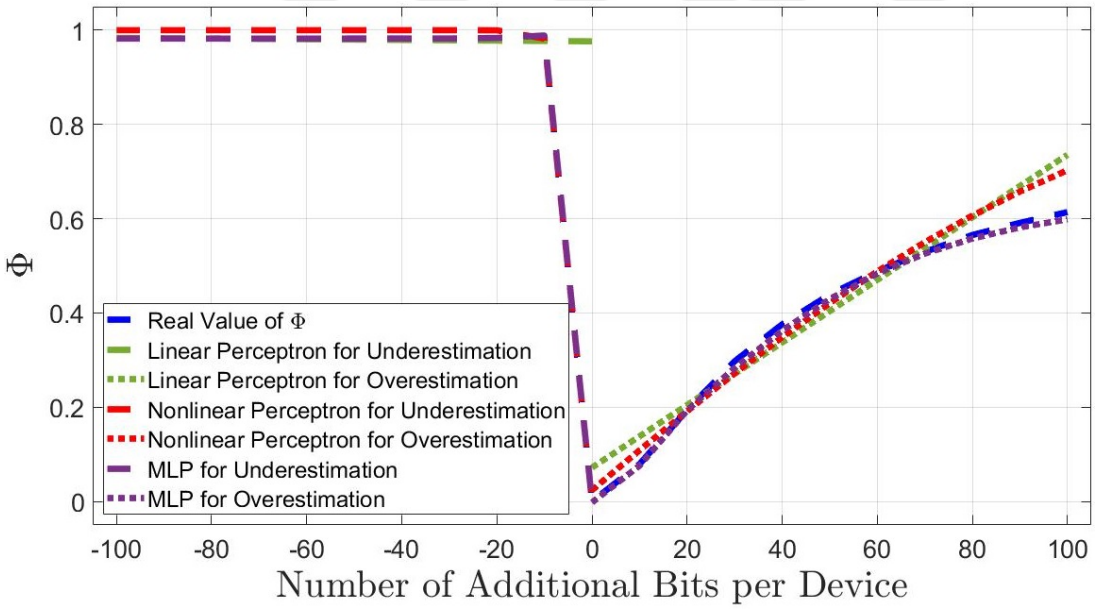

Figure 2.6. Learning performance of E-ASEF under each of the Linear Perceptron, Nonlinear Perceptron, and MLP for the throughput difference Φ under the network set-up that has devices only in the VBA class.

2.5.3 Execution Time of E-ASEF

In Table 2.1, we present the execution time of E-ASEF. We calculate the mean and the standard deviation of the execution time over 100 simulation runs on the Google Colab platform with no accelarator. In this table, we see that the execution time of E-ASEF is under 20 μs for each of the Linear Perceptron, Nonlinear Perceptron and the MLP models. This shows that E-ASEF is a practical emulation methodology in this setting. In addition, we see that the execution time difference between E-ASEF under MLP and that under the other models is less than 1 μs. Thus, MLP is the best selection among the models that we have examined for the EML block in E-ASEF, since it achieves the best performance at an execution time that is comparable to those of the other models.

Table 2.1. Execution Time of E-ASEF [μs]

EML Model in E-ASEF	Mean	Standard Deviation
Linear Perceptron	18.68	0.63
Nonlinear Perceptron	19.16	0.85
MLP	19.42	0.22

2.6 Summary

In this chapter, we have developed a novel methodology in order to investigate the relationship between the forecasting error and network performance in Joint Forecasting-Scheduling (JFS). Our methodology has been to build an Artificial Neural Network (ANN) whose input is the forecasting error and whose output is the difference between the network performance obtained under forecasting versus that obtained under the actual network traffic realization. The key novel aspect of our work is the idea of a subspace-based emulation where only a subspace of the entire space of forecasting errors is employed. We have demonstrated that a Multi-Layer Perceptron (MLP) can successfully learn the relationship between the forecasting error and the network performance difference on this subspace.

CHAPTER 3

Subspace-Based Application-Specific Error Metric Emulation

3.1 Introduction

As we have discussed in Chapter 1, IoT is expected to continue to play a major role in the evolution of cellular as well as dedicated industrial networks in the near future. A key problem faced by all such networks is the tremendous number of IoT devices that fall in the coverage area of a single IoT gateway (or base station or access point) (Ghavimi & Chen, 2014). In urban areas, smart utility meters alone are expected to reach a density of 5000 to 50,000 devices per base station (Vodafone, Apr. 2010). Such Machine-to-Machine (M2M) communication, in which traffic is generated by a machine to be communicated to a server without human intervention, is expected to pose a major challenge due to the number of connections that must be maintained by the network. This problem is referred to as the Massive Access Problem of M2M communication.

Even though there have been a plethora of past articles[1] to solve the Massive Access Problem, the key assumption behind almost all of these articles has been that the traffic of an IoT device at the Medium Access Control (MAC) layer is well-modeled by "random arrivals". This assumption has led to the development of MAC-layer protocols that react to the *current* traffic demand. However, recent work (Nakip, Gül, et al., 2019) has shown that the traffic generation patterns for M2M communication have structure and can be predicted. This has opened the way to the development of Joint Forecasting-Scheduling (JFS) in (Nakip, Rodoplu, et al., 2019; Rodoplu, Nakıp, et al., 2020; Rodoplu, Nakip, et al., 2020) in which the IoT gateway forecasts the future traffic generation patterns of all of the IoT devices in its coverage area and allocates the MAC-layer resources in advance based on these forecasts.

Although significant advances have been achieved in these works[2], forecasters have not been designed by quantifying precisely the impact of forecasting on the network

[1] See (Rodoplu, Nakıp, et al., 2020) and the categorization of the references therein.

[2] The technical content in this chapter has been submitted as a journal paper (Nakip, Helva, Güzeliş, & Rodoplu, 2021a) to IEEE Access.

performance measured at the output of MAC-layer scheduling. In this work, first, we develop a novel methodology by which we form an application-specific error metric that quantifies this impact. Second, we show that this application-specific error metric can be emulated by an Artificial Neural Network (ANN), if the metric is confined to lie on a subspace of the forecasting errors. Third, we develop an algorithm, which we call Motion On a Subspace under Adaptive Learning rate (MOSAL), which moves on this subspace of forecasting errors while minimizing the application-specific error metric at the output of MAC-layer scheduling. While our exposition in this chapter is specific to JFS, the methodology that we develop can be applied to any problem in which an application utilizes forecasts and a metric exists that quantifies the performance of the application.[3] In this sense, the techniques that we develop in this chapter are expected to impact the design of forecasting-based systems beyond the immediate context of IoT.

The rest of this chapter is organized as follows: In Section 3.2, we review the basic JFS system. In Section 3.3, we describe our mathematical framework. In Section 3.4, we present our results. Finally, in Section 3.5, we present our conclusions.

3.2 Review of Joint Forecasting-Scheduling (JFS)

We assume[4] that there is a set of N IoT devices, denoted by $\mathcal{N}$, in the coverage area of an IoT Gateway G. Each device has a direct uplink wireless link to G. We let τ_{MAC} denote the duration of a MAC-layer slot (in short, "slot"). Throughout this chapter, we assume that there is a single channel[5] on which all of the devices have been slot-synchronized. In slot s, we let $x_i[s]$ denote the total number of bits generated by device $i \in \mathcal{N}$.

We assume that the generation of traffic by each IoT device occurs in bursts. Each burst j is defined by its number of bits a_j; its generation time (namely the slot in which it is generated) denoted by r_j; and its deadline (namely the slot by the beginning of which it must be delivered) denoted d_j. Thus, for burst j that is generated by device i, the delay constraint is $\Delta_j = d_j - r_j$. The "processing time" p_j of burst j is the number of slots required to send that burst. Thus, $p_j = \lceil a_j/(\tau_{MAC} R_i) \rceil$, where R_i is uplink data rate of device i. Furthermore, we assume that Gateway G has access to the past traffic pattern of each IoT device $i \in \mathcal{N}$ by virtue of compressed, incremental past traffic patterns that

[3] In the specific example of JFS, the application is MAC-layer scheduling and the metric that quantifies the performance of scheduling is network performance, such as throughput or energy consumption.

[4] The assumptions of this chapter are identical to those in (Nakip, Rodoplu, et al., 2019).

[5] We shall address the case of multiple channels in our future work.

are piggy-backed to each transmission from i to G.

Based on these assumptions, the recent works (Nakip, Rodoplu, et al., 2019; Rodoplu, Nakıp, et al., 2020) proposed Joint Forecasting-Scheduling (JFS), which is summarized in Fig. 3.1. The JFS system resides at the IoT Gateway G and is comprised of the cascade of the Bank of Forecasters (BoF) and the Scheduler. We shall now describe each of these modules.

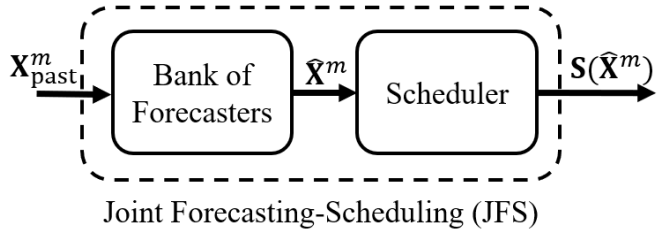

Figure 3.1. Joint Forecasting-Scheduling (JFS) system

Bank of Forecasters (BoF)

The BoF contains N forecasters, each of which forecasts the future traffic generation pattern of an individual IoT device. At the current MAC-layer slot m, the input to the BoF is the matrix $\mathbf{X}^m_{\text{past}}$ of the past traffic generation patterns of IoT d evices. The (i,l)th entry of this matrix is $x_i[m-l]$, where l takes integer values on $[1,L]$ for some window size L into the past. We let $\hat{x}_i[s]$ denote the forecast number of bits that are generated by device i in slot s. The output of BoF is the matrix $\hat{\mathbf{X}}^m$ of the forecast traffic generation patterns of the devices. The (i,k)th entry of this matrix is $\hat{x}_i[m+k]$, where k takes integer values on $[0,K-1]$ for some window size K into the future.

Scheduler

The Scheduler allocates the MAC-layer resources to all of the IoT devices in $\mathcal{N}$ based on the forecast future traffic $\hat{\mathbf{X}}^m$ over a scheduling window of duration T_{sch}. (We assume that T_{sch} is a multiple of τ_{MAC}.) To this end, the Scheduler solves an optimization program to generate a binary schedule, denoted by $\mathbf{S}(\hat{\mathbf{X}}^m)$, whose (i,s)th entry equals 1 if the uplink slot s in the scheduling window is allocated to device i, and equals 0 otherwise.

In this chapter, for the particular implementation of the Scheduler, we use the Priority based on Average Load (PAL) algorithm (Nakip, Rodoplu, et al., 2019), which is a greedy heuristic. The PAL algorithm works as follows: At any slot s of the current scheduling window, we let $\mathscr{J}_{\text{active}}[s]$ denote the set of forecast bursts that have not yet been scheduled, whose deadlines have not yet expired and whose entire set of bits can be scheduled by the end of the scheduling window. We also let $\tilde{\Delta}_j[s]$ denote the number of remaining slots starting with slot s until the beginning of slot d_j. The PAL algorithm starts at the first slot of the current scheduling window ($s = 1$) with a null schedule and takes the following steps: 1) It computes $\mathscr{J}_{\text{active}}[s]$. If $\mathscr{J}_{\text{active}}[s]$ is the null set, PAL returns the resulting schedule. 2) For each burst $j \in \mathscr{J}_{\text{active}}[s]$, PAL calculates $\tilde{\Delta}_j[s]$ as well as the average load $\gamma_j[s] \equiv a_j/(\tilde{\Delta}_j[s])$. 3) It finds a burst j^* with the largest average load; that is, $j^* = \arg\max_{j \in \mathscr{J}_{\text{active}}[s]} \gamma_j[s]$. 4) PAL reserves the next p_{j^*} slots for device i that generated burst j^*. 5) The value of s is updated as $s \leftarrow s + p_{j^*}$; and 5) If $s \leq (T_{sch}/\tau_{MAC})$, the algorithm goes to step 1); otherwise, it returns the resulting schedule.

3.3 Mathematical Framework

In this section, we develop our mathematical framework. First, we define the Application-Specific Error Function (ASEF) in the context of JFS and describe how we emulate the ASEF on a subspace of the forecasting errors via an ANN. Second, we describe how we train the JFS forecasters on this subspace and state our MOSAL algorithm.

3.3.1 Application Specific Error Function (ASEF) and Its Emulation (E-ASEF)

We define the "throughput" of the network as the total number of successfully delivered bits divided by the total number of generated bits. Furthermore, we let $\mathbf{X}^m$ denote the $N \times K$ matrix of the actual traffic patterns of IoT devices accumulated by the current slot m: The (i,k)th entry of this matrix is $x_i[m+k]$, where k takes integer values on $[0, K-1]$. We let $\eta(\mathbf{S}(\hat{\mathbf{X}}^m), \mathbf{X}^m)$ denote the throughput performance of the JFS system under the realization $\mathbf{X}^m$ when the schedule $\mathbf{S}(\hat{\mathbf{X}}^m)$ produced by the forecasting scheme is used in the BoF. Furthermore, we let $\eta(\mathbf{S}(\mathbf{X}^m), \mathbf{X}^m)$ denote the throughput performance of the JFS system under perfect forecasts.

We define the Application-Specific Error Function (ASEF) for JFS as the difference Φ^m

(at the current slot m) between $\eta(\mathbf{S}(\mathbf{X}^m), \mathbf{X}^m)$ and $\eta(\mathbf{S}(\hat{\mathbf{X}}^m), \mathbf{X}^m)$. That is,

$$\Phi^m \equiv \eta(\mathbf{S}(\mathbf{X}^m), \mathbf{X}^m) - \eta(\mathbf{S}(\hat{\mathbf{X}}^m), \mathbf{X}^m) \tag{3.1}$$

Note that ASEF measures the impact of forecasting errors on the resulting network performance (which is taken as throughput in this chapter).

One of our main ideas is that if this ASEF can be emulated by an ANN, then the BoF in the JFS system can be trained by exploiting this a priori knowledge of the impact of forecasting errors on the network performance. To this end, we shall let $\mathbf{D}^m \equiv \hat{\mathbf{X}}^m - \mathbf{X}^m$ denote the "forecasting error matrix". Note that $\mathbf{D}^m(i,k)$ is the (i,k)th entry of $\mathbf{D}^m$, namely the forecasting error for the kth step ahead forecast of device i.

In our investigations, we have found that it is extremely difficult to train an ANN to learn the ASEF on the entire space of forecasting errors (which is the space spanned by all possible $\mathbf{D}^m$ matrices). However, we have discovered that it is possible to emulate the ASEF via an ANN on a *subspace* of these errors. We call the ANN that emulates ASEF on such a subspace the Emulated ASEF (E-ASEF).

In this chapter, we take this subspace as that defined by the condition that the forecasting error is identical over all devices and all time-step ahead forecasts; that is, we take the subspace $\mathscr{V} \equiv \{\mathbf{D}^m \mid \text{All of the entries of } \mathbf{D}^m \text{ are identical.}\}$. Let $\mathbf{D}_\alpha$ denote a matrix all of whose entries have the same value $\alpha \in (-\infty, +\infty)$. Thus, $\mathbf{D}_\alpha$ is a point of the subspace $\mathscr{V}$. Note that there is a one-to-one correspondence between $\mathbf{D}_\alpha$ and its parameter α. In our experiments, we have investigated the range of the values of Φ that result from the empirically observed values of the matrices $\mathbf{X}^m$ and $\hat{\mathbf{X}}^m$ and found that the Φ's fall in a relatively restricted range. Let $\tilde{R}$ denote the relation that exists between the parameter α and the range of the application-specific error Φ obtained at the output of the Scheduler. In our design, at the end of training, E-ASEF will converge to a function f that satisfies the relation $\tilde{R}$; that is, f will map each α to a value that falls in the empirically observed range of Φ.

Now, assume that the training of E-ASEF has been completed. We let $\mathbf{D}^m_{\text{test}}$ denote the forecasting error matrix *during the testing* of E-ASEF. Note that it is possible that $\mathbf{D}^m_{\text{test}}$ lies outside $\mathscr{V}$. We shall associate a $\tilde{\mathbf{D}}^m$ that lies on $\mathscr{V}$ with $\mathbf{D}^m_{\text{test}}$ as follows: First, let $\mathbf{X}^m_{\text{test}}$ and $\hat{\mathbf{X}}^m_{\text{test}}$ denote the $\mathbf{X}^m$ and the $\hat{\mathbf{X}}^m$ during the testing of E-ASEF, respectively. Note that $\mathbf{D}^m_{\text{test}} = \hat{\mathbf{X}}^m_{\text{test}} - \mathbf{X}^m_{\text{test}}$. Second, we define the average forecasting error during testing as

$$\bar{d}^m_{\text{test}} = \frac{\sum_{i \in \mathcal{N}} \sum_{k=0}^{K-1} \mathbf{D}^m_{\text{test}}(i,k)}{NK} \tag{3.2}$$

We form the matrix $\tilde{\mathbf{D}}^m$ by setting $\tilde{\mathbf{D}}^m(i,k) = \bar{d}^m_{\text{test}}\ \forall (i,k)$. Since $\tilde{\mathbf{D}}^m$ lies on $\mathcal{V}$, we may now use the function f to which E-ASEF has converged (by the end of its training) in order to estimate the ASEF (during testing) at current slot m as

$$\hat{\Phi}^m = f(\bar{d}^m_{\text{test}}) \tag{3.3}$$

3.3.2 Training Architecture and the MOSAL Algorithm

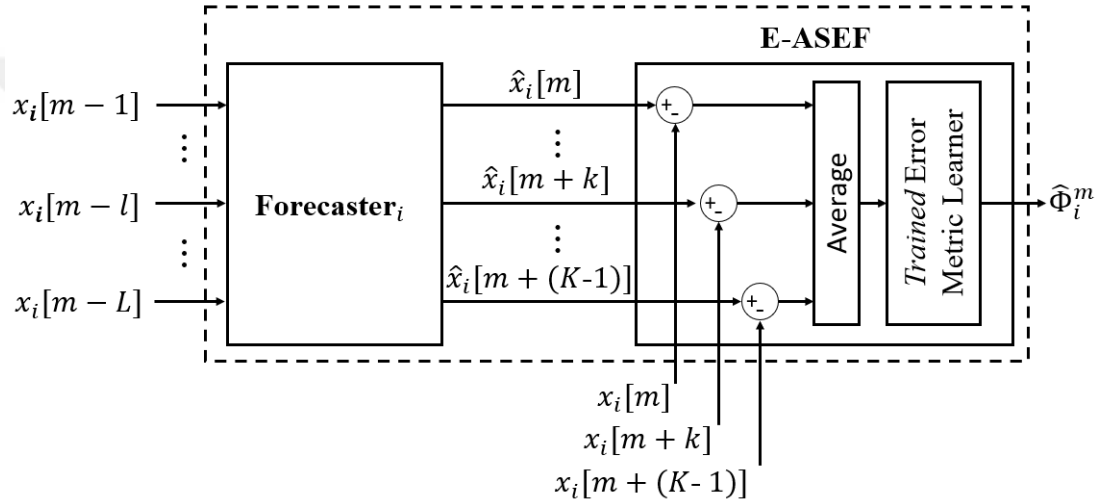

Figure 3.2. Our proposed architecture in which we cascade Forecaster$_i$ and the E-ASEF whose training has been completed. We train the Forecaster$_i$ via back-propagation by minimizing $\hat{\Phi}^m_i$ at the output of this architecture.

In this section, we describe a novel architecture by which to train each forecaster in the BoF of JFS in Fig. 3.1 by minimizing $\hat{\Phi}^m$. Fig. 3.2 displays our architecture in which we cascade the forecaster of device i and E-ASEF whose training has been completed. (This architecture is replicated for the forecaster of each device i.)

At each slot m, the inputs to this architecture are $\{x_i[m-l]\}_{l \in \{1,\ldots L\}}$ and $\{x_i[m+k]\}_{k \in \{0,\ldots K-1\}}$, and the output is the throughput difference $\hat{\Phi}^m_i$. This architecture is an end-to-end trainable neural network, which enables the training of Forecaster$_i$ via back-propagation starting with $\hat{\Phi}^m_i$. Note that in this architecture, the parameters of the E-ASEF module, whose training has been completed, are kept constant; that is, they are untrainable during the training of Forecaster$_i$.

Now, in order to train Forecaster$_i$, we develop a novel algorithm, which we call "Motion On a Subspace under Adaptive Learning rate" (MOSAL). This algorithm trains Forecaster$_i$ by keeping the forecasting error matrix on the subspace $\mathscr{V}$ while minimizing $\hat{\Phi}_i^m$.

To this end, we assume that the dataset has already been partitioned into training, validation and test sets. (Only the training and validation sets are used in training Forecaster$_i$ via the MOSAL algorithm.) Furthermore, we let $\mathbf{W}_i^t$ denote the collection of parameters, namely the connection weights and the biases, of Forecaster$_i$ at training step t. In this set of parameters, we single out the vector of biases of the output neurons at training step t and denote this vector by $\mathbf{b}_i^t$.

We say that Forecaster$_i$ "overestimates" if Forecaster$_i$ produces a positive forecasting error. We define the "average overestimation error" $\mathscr{E}_{\text{over}}^t$ of Forecaster$_i$ at training step t as the overestimation error averaged over the samples in the validation set for which Forecaster$_i$ overestimates and over the values of $k \in [0, K-1]$. We say that the Forecaster$_i$ "underestimates" if Forecaster$_i$ produces a negative forecasting error. Similarly, we define the "average underestimation error" $\mathscr{E}_{\text{under}}^t$ of Forecaster$_i$ at training step t as the underestimation error averaged over the samples in the validation set for which Forecaster$_i$ underestimates and over the values of $k \in [0, K-1]$.

Let $\mathscr{E}_{\text{under}}^{min}$ denote the absolute value of the minimum underestimation error on the training set.

The parameters of our algorithm are as follows: ζ denotes the learning rate; $r \in (0,1)$ denotes the update ratio for ζ; θ denotes a positive threshold for the change in the forecasting error; T denotes the maximum number of training iterations; and M denotes the maximum number of steps taken in a given direction on the subspace $\mathscr{V}$. We let ζ^0 denote the value to which ζ is reinitialized. Furthermore, we initialize $\zeta = \zeta^0$ at the beginning of the algorithm.

We shall now describe the MOSAL algorithm: We fix the parameter $\tilde{\alpha}$ as any positive real number. At the initial step $t = 0$, for each device i, the algorithm initializes $\mathbf{W}_i^0$ by training Forecaster$_i$ via gradient descent on the training set such that $\mathbf{D}^m(i,k) = \tilde{\alpha} \quad \forall k$. In addition, the algorithm computes $\mathscr{E}_{\text{over}}^0$ and $\mathscr{E}_{\text{under}}^0$ and calculates $\hat{\Phi}_i^0$ via a forward pass on the architecture in Fig. 3.2. We shall let ε_Φ denote the desired output of the architecture. Then, for the inputs $\{\mathbf{X}_{\text{past}}^m(i,l)\}_{l\in\{1,\ldots,L\}}$ and $\{\mathbf{X}^m(i,k)\}_{k\in\{0,\ldots,K-1\}}$ and the desired output ε_Φ, the MOSAL algorithm, which runs on this architecture, appears in Fig. 3.3.

Once the architecture in Fig. 3.2 has been trained via MOSAL for each device $i \in \mathscr{N}$,

forecasterType **MOSAL_Algorithm** {

0 $t = 0; \tilde{t} = 0;$

1 $t = t + 1;$

2 if($\tilde{t} == 2M \;||\; t == T$)
  Return Forecaster$_i$ as "untrained" model;

3 Compute Forecaster$_i \rightarrow \mathbf{W}_i^t$ via gradient descent for 3 epochs to minimize the mean squared error between ε_Φ and $\hat{\Phi}_i^{t-1}$;

4 Compute $\hat{\Phi}_i^t$ via a forward pass on the architecture;

5 if($|\hat{\Phi}_i^t| \leq \varepsilon_\Phi$) {

6 Stop training;

7 Go to Line 19; }

8 Compute $\mathscr{E}_{\text{over}}^t$ and $\mathscr{E}_{\text{under}}^t$;

9 if($\mathscr{E}_{\text{over}}^t \leq \mathscr{E}_{\text{over}}^{t-1}$) {

10 if($\tilde{t} > 0$) $\tilde{t} = 0$ and $\zeta = \zeta^0$;

11 if($\mathscr{E}_{\text{over}}^{t-1} - \mathscr{E}_{\text{over}}^t \geq \theta$) $\zeta = (1-r)\zeta$;

12 else $\zeta = (1+r)\zeta$;

13 } else {

14 $\tilde{t} = \tilde{t} + 1;$

15 Forecaster$_i \rightarrow \mathbf{W}_i^t$ = Forecaster$_i \rightarrow \mathbf{W}_i^{t-1}$;

16 if($\tilde{t} \leq M$) $\zeta = 2^{\tilde{t}}\zeta^0$;

17 else ζ = -$2^{(\tilde{t}-M)}\zeta^0$; }

18 Go to Line 1;

19 Compute $\mathscr{E}_{\text{under}}^{min}$;

20 Forecaster$_i \rightarrow \mathbf{b}_i^t$ = Forecaster$_i \rightarrow \mathbf{b}_i^t + \mathscr{E}_{\text{under}}^{min}\mathbf{1}$;

21 Return Forecaster$_i$;

22 }

Figure 3.3. The MOSAL Algorithm

we take the following additional steps: For a given device class, let n be the index of a device that MOSAL has trained successfully and which achieves the minimum $\hat{\Phi}^m$ for that class. (We assume that such a device exists for each device class.) Let $\mathbf{W}_n^{\text{final}}$ denote the parameters of Forecaster$_n$ when the MOSAL algorithm returns for device n. For each device i that belongs to the same class as n and whose Forecaster$_i$ is untrained when MOSAL returns for device i, we set $\mathbf{W}_i^0 = \mathbf{W}_n^{\text{final}}$.

3.4 Results

3.4.1 Methodology

IoT Traffic Dataset

We use the dataset in (*IoT Traffic Generation Pattern Dataset*, 2021), which is com-prised of the traffic generation patterns of 10000 bootstrapped IoT devices. Reference (Nakip, Gül, et al., 2019) classifies the IoT traffic into four distinct classes: Fixed Bit Aperiodic (FBA), Fixed Bit Periodic (FBP), Variable Bit Aperiodic (VBA) and Variable Bit Periodic (VBP). According to this classification, if an IoT device generates a fixed number of bits, its traffic generation is called "Fixed Bit"; otherwise, the traffic generation is called "Variable Bit". Moreover, if the inter-arrival time between the bursts of a device is constant, its traffic generation is called "Periodic"; otherwise, the traffic generation is called "Aperiodic". Since the traffic generation pattern for FBP class is known in advance, in our simulations, we focus on only the VBP, FBA and the VBA classes.

Wireless Channel Model

We use the Rayleigh fading model, which is described in (Rodoplu, Nakip, et al., 2020) for a JFS system. Furthermore, we set the parameters as follows: the duration of a MAC-layer slot τ_{MAC} = 0.1 s; the duration of the scheduling window T_{sch} = 900 s; the number of channel taps L = 10; the channel bandwidth W = 1 MHz; κ = 0.1 (for variance $\kappa/2L$); the value of the single-sided power spectral density of the white noise at the receiver of Gateway $N_O = 10^{-3}$ Watts/Hz; and the total transmit power of each IoT device P = 1 mW.

Parameters of the Proposed Architecture and the MOSAL Algorithm

For the particular implementation of Forecaster$_i$ in the architecture shown in Fig. 3.2, we use a Multi-Layer Perceptron (MLP) model, which is comprised of three hidden layers and an output layer. The number of neurons across the layers are (in vector notation) $[L, \lceil L/2 \rceil, \lceil L/4 \rceil, K]$, where $L = 120$ is the number of neurons in the first hidden layer, and $K = 5$ is the number of neurons in the output layer. In addition, the activation function of all neurons is set to the Rectified Linear Unit (*ReLU*).

Furthermore, for the MOSAL algorithm, we set $\zeta^0 = 0.05$; $r = 0.25$; $\theta = 0.1$; $T = 10000$; $M = 30$; $\varepsilon_\Phi = 0.05$; and $\tilde{\alpha} = 20$. We shall say that a parameter is "critical" if we expect the ultimate performance to show the highest sensitivity to perturbations in the value of this parameter. In this list of parameters, the first critical parameter is ε_Φ. If this parameter is too large, then learning will cease early, thus resulting in a low throughput. In contrast, if this parameter is too small, then oscillations will result, which will hamper the learning process. The second critical parameter is θ, which is a threshold parameter. If this parameter is too large, then the algorithm cannot attain this threshold. If this parameter is too small, then the learning rate will oscillate. Our selection of the values of these parameters has been carried out carefully in accordance with these considerations.

3.4.2 Performance Evaluation

We now evaluate the performance of the JFS system when the BoF, comprised of MLP forecasters, has been trained via our architecture in Fig. 3.2.

We shall let MLP-MOSAL denote an MLP forecasting model that has been trained by the MOSAL algorithm. Furthermore, we refer to an MLP forecasting model that is trained via gradient descent as MLP-GD. In this section, we compare the performance of MLP-MOSAL against that of MLP-GD. To this end, $\mathcal{N}$ contains an equal percentage for each of the VBP, VBA and the FBA classes from the dataset (*IoT Traffic Generation Pattern Dataset*, 2021).

In Fig. 3.4, we display the throughput of the JFS system (that uses the PAL scheduling heuristic) under MLP-MOSAL as well as that under the MLP-GD forecaster. In this figure, first, we see that the JFS system under MLP-MOSAL significantly outperforms that under MLP-GD. That is, the minimization of ASEF rather than that of the traditional

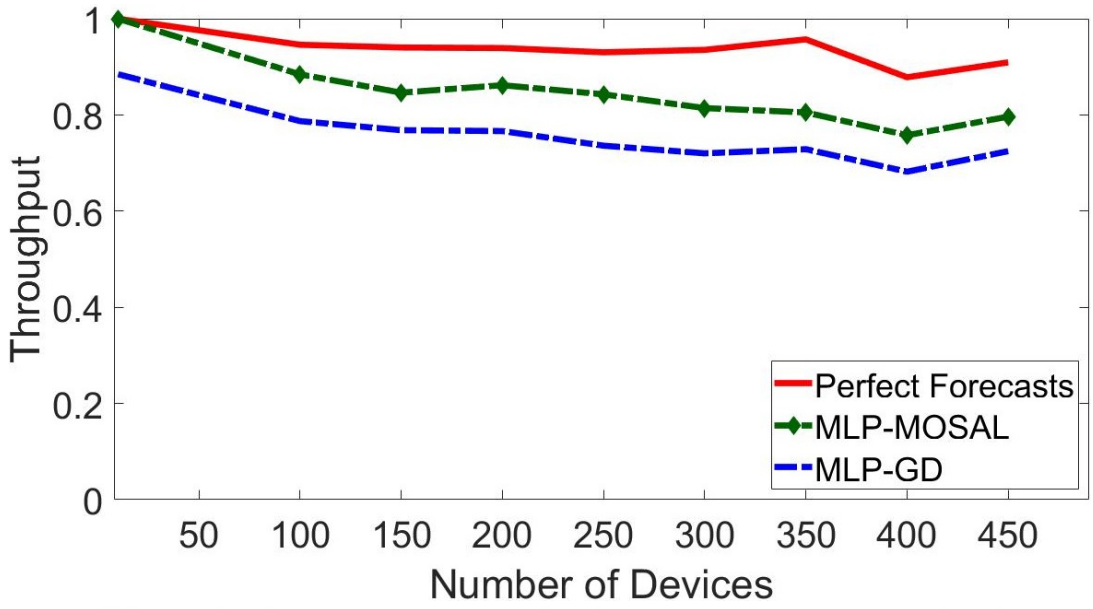

Figure 3.4. Throughput performance of JFS under perfect forecasts, MLP-MOSAL and MLP-GD

error metrics significantly improves the throughput performance. Second, we see that using MLP-MOSAL in the place of MLP-GD closes a significant fraction of the gap to the performance under perfect forecasts.

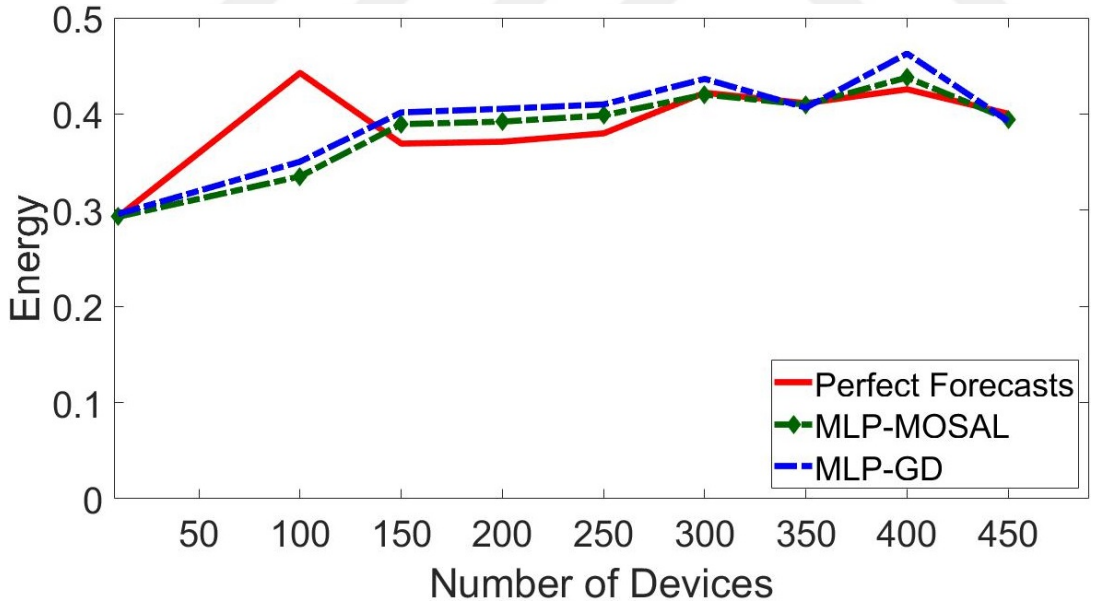

Figure 3.5. Energy consumption of JFS under perfect forecasts, MLP-MOSAL and MLP-GD

In Fig. 3.5, we present the energy consumption of the JFS system. We see that the energy consumption under MLP-MOSAL and MLP-GD are comparable across all N.

Furthermore, both of them are close to one under perfect forecasts for $N > 150$.

3.4.3 Computation Time

In Table 3.1, we present the mean and the standard deviation (STD) of the training and execution times of MLP-MOSAL and MLP-GD. In this table, we see that the training time of MLP-MOSAL is higher than that of MLP-GD, where the difference is one order of magnitude. Furthermore, in Table 3.1, we see that the execution time of MLP-MOSAL is comparable with that of MLP-GD.The execution time of both forecasters are 0.18 ms which is acceptable for real-time IoT applications.

Table 3.1. Training and Execution Times of MLP-MOSAL and MLP-GD

Forecasting	Training Time (s)		Execution Time (ms)	
Models	Mean	STD	Mean	STD
MLP-MOSAL	56.17	2.64	0.18	0.15
MLP-GD	5.40	2.66	0.18	0.23

3.5 Summary

We have developed a novel methodology for Joint Forecasting-Scheduling (JFS) in which the forecasters are trained with respect to an application-specific error metric that is learned via an Artificial Neural Network (ANN) model that emulates on a subspace the difference in network performance between that under perfect forecasts and that under Multi-Layer Perceptron (MLP) based forecasting. Based on this emulation, we have designed an algorithm, called Motion On a Subspace under Adaptive Learning rate (MOSAL) which achieves a throughput performance that surpasses those of the past JFS forecaster training algorithms at reasonable computation time. In our future work, we plan to apply this methodology to a broad range of forecasting-based systems beyond the context of IoT.

CHAPTER 4

CONCLUSIONS

In this thesis, we have developed a subspace-based Application-Specific Error Function (ASEF) for Joint Forecasting-Scheduling. In addition, we have proposed a method for the emulation of ASEF, called E-ASEF. Furthermore, we proposed a novel algorithm, MOSAL, that moves on the subpsace on which E-ASEF has been trained. We now review each of these contributions.

In Chapter 2, we examined how our novel methodology can improve the JFS system by using the relationship between forecasting error and network performance. One of the findings of our study is that on a subspace, we can emulate the behavior of ASEF. Furthermore, we have shown that a Multi-Layer Perceptron (MLP) can be successfully used for this emulation.

In Chapter 3, we have presented a novel algorithm that we have developed, namely MOSAL, which has improved the performance of JFS at low execution time. The main idea behind MOSAL is that the forecasting error across all of the forecasters at the IoT gateway can be kept on the subspace of forecasting errors that was identified via E-ASEF while reducing the forecasting error towards zero during training. MOSAL achieves this convergence by adapting the rate at which it moves at every step as the forecasting error is being reduced.

Hence, we can summarize the contributions of this thesis as follows; (1) We have developed an Application-specific Error Function (ASEF); (2) We have emulated this function by an ANN on a subspace; (3) On this subspace, we have developed a novel algorithm, called MOSAL. In the future, we intend to apply this system that we have developed to forecasting-based systems that are outside of the context of IoT.

REFERENCES

Aijaz, A., & Aghvami, A. H. (2015). Cognitive machine-to-machine communications for Internet-of-Things: A protocol stack perspective. *IEEE Internet of Things Journal*, *2*(2), 103–112.

Aijaz, A., Ping, S., Akhavan, M. R., & Aghvami, A.-H. (2014). CRB-MAC: A receiver-based MAC protocol for cognitive radio equipped smart grid sensor networks. *IEEE Sensors Journal*, *14*(12), 4325–4333.

Alavikia, Z., & Ghasemi, A. (2018). Collision-aware resource access scheme for LTE-based machine-to-machine communications. *IEEE Transactions on Vehicular Technology*, *67*(5), 4683–4688.

Ali, S., Saad, W., & Rajatheva, N. (2018). A directed information learning framework for event-driven M2M traffic prediction. *IEEE Communications Letters*, *22*(11), 2378–2381.

Bello, O., & Zeadally, S. (2019). Toward efficient smartification of the Internet of Things (IoT) services. *Future Generation Computer Systems*, *92*, 663–673.

Bockelmann, C., Pratas, N., Nikopour, H., Au, K., Svensson, T., Stefanovic, C., ... Dekorsy, A. (2016). Massive machine-type communications in 5G: Physical and MAC-layer solutions. *IEEE Communications Magazine*, *54*(9), 59–65.

Cai, D., He, X., & Han, J. (2007). Spectral regression for efficient regularized subspace learning. In *2007 IEEE 11th International Conference on Computer Vision* (p. 1-8). doi: 10.1109/ICCV.2007.4408855

Cerwall, P., Jonsson, P., Möller, R., Bävertoft, S., Carson, S., Godor, I., ... Lindberg, P. (2015). Ericsson mobility report. *On the Pulse of the Networked Society. Hg. v. Ericsson*.

Cisco. (Mar. 2020). *Cisco Annual Internet Report (2018–2023).* Retrieved from `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html`

De La Torre, F., & Black, M. J. (2003). A framework for robust subspace learning. *International Journal of Computer Vision*, *54*(1), 117–142.

Ericsson. (Nov. 2019). *Ericsson Mobility Report*. Retrieved from https://www.ericsson.com/en/mobility-report

Fortino, G., Russo, W., Savaglio, C., Viroli, M., & Zhou, M. (2017). Modeling opportunistic IoT services in open IoT ecosystems. In *WOA* (pp. 90–95).

Ghavimi, F., & Chen, H.-H. (2014). M2M communications in 3GPP LTE/LTE-A networks: Architectures, service requirements, challenges, and applications. *IEEE Communications Surveys & Tutorials*, *17*(2), 525–549.

Gu, Q., Li, Z., & Han, J. (2011). Joint feature selection and subspace learning. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence* (Vol. 22, p. 1294).

Hasan, M., Hossain, E., & Niyato, D. (2013). Random access for machine-to-machine communication in LTE-advanced networks: Issues and approaches. *IEEE Communications Magazine,* 51(6), 86–93.

IoT Traffic Generation Pattern Dataset. (2021, Jan). Retrieved from https://www.kaggle.com/tubitak1001118e277/iot-traffic-generation-patterns

Jiang, X. (2011). Linear subspace learning-based dimensionality reduction. *IEEE Signal Processing Magazine*, *28*(2), 16–26.

Jin, H., Toor, W. T., Jung, B. C., & Seo, J.-B. (2017). Recursive pseudo-Bayesian access class barring for M2M communications in LTE systems. *IEEE Transactions on Vehicular Technology*, *66*(9), 8595–8599.

Kuhlins, C., Rathonyi, B., Zaidi, A., & Hogan, M. (Jan. 2020). White paper: Cellular networks for massive IoT. Retrieved from https://www.ericsson.com/en/reports-and-papers/white-papers/cellular-networks-for-massive-iot--enabling-low-power-wide-area-applications

Lawal, I. A., Abdulkarim, S. A., Hassan, M. K., & Sadiq, J. M. (2016). Improving HSDPA traffic forecasting using ensemble of neural networks. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 308–313).

Li, Y. (2004). On incremental and robust subspace learning. *Pattern Recognition*, 37(7), 1509–1518.

Liang, L., Xu, L., Cao, B., & Jia, Y. (2018). A cluster-based congestion-mitigating access scheme for massive M2M communications in Internet of Things. *IEEE*

Internet of Things Journal, *5*(3), 2200–2211.

Lien, S.-Y., Liau, T.-H., Kao, C.-Y., & Chen, K.-C. (2011). Cooperative access class barring for machine-to-machine communications. *IEEE Transactions on Wireless Communications*, *11*(1), 27–32.

Lin, T.-M., Lee, C.-H., Cheng, J.-P., & Chen, W.-T. (2014). PRADA: Prioritized random access with dynamic access barring for MTC in 3GPP LTE-A networks. *IEEE Transactions on Vehicular Technology*, *63*(5), 2467–2472.

Liu, J., Song, L., et al. (2017). A novel congestion reduction scheme for massive machine-to-machine communication. *IEEE Access*, *5*, 18765–18777.

Liu, Y., Yuen, C., Cao, X., Hassan, N. U., & Chen, J. (2014). Design of a scalable hybrid MAC protocol for heterogeneous M2M networks. *IEEE Internet of Things Journal*, *1*(1), 99–111.

Lopez-Martin, M., Carro, B., & Sanchez-Esguevillas, A. (2019). Neural network architecture based on gradient boosting for IoT traffic prediction. *Future Generation Computer Systems*, *100*, 656–673.

Lopez-Martin, M., Carro, B., & Sanchez-Esguevillas, A. (2020). IoT type-of-traffic forecasting method based on gradient boosting neural networks. *Future Generation Computer Systems*, *105*, 331–345.

Mobasheri, M., Kim, Y., & Kim, W. (2020). Toward developing fog decision making on the transmission rate of various IoT devices based on reinforcement learning. *IEEE Internet of Things Magazine*, *3*(1), 38–42.

Nakip, M., Gül, B. C., Rodoplu, V., & Güzeliş, C. (2019). 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* (pp. 102–109).

Nakip, M., Helva, A., Güzeliş, C., & Rodoplu, V. (2021a). Subspace-based application-specific error metric emulation in predictive networks to enable massive access for the Internet of Things. In *2021 IEEE* Access, submitted.

Nakip, M., Helva, A., Güzeliş, C., & Rodoplu, V. (2021b). Subspace-based emulation of the the relationship between forecasting error and network performance in joint forecasting-scheduling for the Internet of Things. In *2021 IEEE World Forum on the Internet of Things (WF-IoT 2021)* (pp. 1–6).

Nakip, M., Rodoplu, V., Güzeliş, C., & Eliiyi, D. T. (2019). Joint forecasting-scheduling for the Internet of Things. In *2019 IEEE Global Conference on Internet of Things (GCIoT)* (pp. 1–7).

Pang, Y.-C., Chao, S.-L., Lin, G.-Y., & Wei, H.-Y. (2014). Network access for

M2M/H2H hybrid systems: A game theoretic approach. *IEEE Communications Letters*, *18*(5), 845–848.

Park, I., Kim, D., & Har, D. (2014). MAC achieving low latency and energy efficiency in hierarchical M2M networks with clustered nodes. *IEEE Sensors Journal*, *15*(3), 1657–1661.

Paul, A., & Rho, S. (2016). Probabilistic model for M2M in IoT networking and communication. *Telecommunication Systems*, *62*(1), 59–66.

Pehlevan, C., Hu, T., & Chklovskii, D. B. (2015). A hebbian/anti-hebbian neural network for linear subspace learning: A derivation from multidimensional scaling of streaming data. *Neural Computation*, *27*(7), 1461–1495.

Rodoplu, V., Nakıp, M., Eliiyi, D. T., & Güzelis, C. (2020). A multi-scale algorithm for joint forecasting-scheduling to solve the massive access problem of IoT. *IEEE Internet of Things Journal*, *7*(9), 8572-8589. doi: 10.1109/JIOT.2020.2992391

Rodoplu, V., Nakip, M., Qorbanian, R., & Eliiyi, D. T. (2020). Multi-channel joint forecasting-scheduling for the Internet of Things. *IEEE Access*, *8*, 217324–217354.

Shahin, N., Ali, R., & Kim, Y.-T. (2018). Hybrid slotted-CSMA/CA-TDMA for efficient massive registration of IoT devices. *IEEE Access*, *6*, 18366–18382.

Shirvanimoghaddam, M., Dohler, M., & Johnson, S. J. (2017). Massive non-orthogonal multiple access for cellular IoT: Potentials and limitations. *IEEE Communications Magazine*, *55*(9), 55–61.

Si, P., Yang, J., Chen, S., & Xi, H. (2014). Adaptive massive access management for QoS guarantees in M2M communications. *IEEE Transactions on Vehicular Technology*, *64*(7), 3152–3166.

Tello-Oquendo, L., Leyva-Mayorga, I., Pla, V., Martinez-Bauset, J., Vidal, J.-R., Casares-Giner, V., & Guijarro, L. (2018). Performance analysis and optimal access class barring parameter configuration in LTE-A networks with massive M2M traffic. *IEEE Transactions on Vehicular Technology*, *67*(4), 3505–3520.

Tello-Oquendo, L., Pacheco-Paramo, D., Pla, V., & Martinez-Bauset, J. (2018). Reinforcement learning-based ACB in LTE-A networks for handling massive M2M and H2H communications. In *2018 IEEE International Conference on Communications (ICC)* (pp. 1–7).

Vodafone. (Apr. 2010). RACH intensity of time controlled devices. *3GPP TSG RAN WG2 #69bis, R2-102296*. Retrieved from `http://www.3gpp.org/DynaReport/TDocExMtg--R2-69b--28031.htm`

Wang, K., He, R., Wang, L., Wang, W., & Tan, T. (2015). Joint feature selection and subspace learning for cross-modal retrieval. *IEEE Transactions on Pattern*

Analysis and Machine Intelligence, *38*(10), 2010–2023.

Zanella, A., Zorzi, M., dos Santos, A. F., Popovski, P., Pratas, N., Stefanovic, C., … Norp, T. A. (2013). M2M massive wireless access: Challenges, research issues, and ways forward. In *2013 IEEE GLOBECOM Workshops* (pp. 151–156).

Zhang, C., & Patras, P. (2018). Long-term mobile traffic forecasting using deep spatio-temporal neural networks. In *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing* (pp. 231–240).