6-2018

Exploring the Role of Semi-Supervised Deep Reinforcement Learning and Ensemble Methods in Support of the Internet of Things

Mehdi Mohammadi
Western Michigan University, mehdi.mka@gmail.com

EXPLORING THE ROLE OF SEMI-SUPERVISED DEEP REINFORCEMENT LEARNING AND ENSEMBLE METHODS IN SUPPORT OF THE INTERNET OF THINGS

by

Mehdi Mohammadi

A dissertation submitted to the Graduate College
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
Computer Science
Western Michigan University
June 2018

Doctoral Committee:

Ala Al-Fuqaha, Ph.D., Chair
Ajay Gupta, Ph.D.
Alvis Fong, Ph.D.
Mohsen Guizani, Ph.D.

EXPLORING THE ROLE OF SEMI-SUPERVISED DEEP REINFORCEMENT LEARNING AND ENSEMBLE METHODS IN SUPPORT OF THE INTERNET OF THINGS

Mehdi Mohammadi, Ph.D.

Western Michigan University, 2018

Smart services are an important element of the Internet of Things (IoT) ecosystem where insights are drawn from raw data through the use of machine learning techniques. However, the pathway to develop IoT smart services is complicated as IoT data presents several challenges for machine learning, including handling big data, shortage of labeled data, and the need to benefit from the spatio-temporal relations hidden in the training data.

In this dissertation, after reviewing the state-of-the-art deep learning (DL) and deep reinforcement learning (DRL) techniques and their use in support of IoT applications, this study proposes to extend DRL to semi-supervised settings using Variational Autoencoders (VAEs). The proposed semi-supervised DRL algorithm exploits the statistical inference of VAEs to benefit from unlabeled data and improve the accuracy of the trained agent to take the best action on the environment.

In IoT applications where most data is spatio-temporal, this study investigates the creation of a dynamic ensemble from distributed deep learning models by considering the spatio-temporal relationships embedded in the training data. The dynamic ensemble does not depend on offline configurations. Instead, it exploits the spatio-temporal relationships embedded in the training data to generate dynamic weights for the underlying weak distributed deep learners to create a stronger learner.

We also present an approach for path planning based on Generative Adversarial Networks (GANs) and crowd-sourced data for wayfinding applications in IoT-enabled environments. In our approach, a GAN is used to recommend accurate and reliable paths to desired destinations. We apply the proposed methods to real-world smart city scenarios including smart energy, smart transportation, and smart buildings. Our results showcase the potential of DL techniques in the development of IoT smart services in general and assert the superiority of our proposed approach compared to baseline methods presented in the recent literature.

Acknowledgments

I would like to acknowledge and thank my adviser, Professor Ala Al-Fuqaha, for the guidance, support, advice, and insightful comments I received from him during my doctoral study and research. Without his constant support, it would not have been possible to conduct this research. All of Professor Al-Fuqaha's contributions of time and ideas made my Ph.D. experience productive and stimulating.

My sincere thanks also go to dissertation committee members Dr. Ajay Gupta, Dr. Alvis Fong, and Dr. Mohsen Guizani for the time they spent in reviewing my work and their insightful and critical comments and suggestions. I also would like to express my appreciation to Dr. Sameh Sorour for his precious comments and editing input on Chapter 2 of this dissertation.

I would like to express my gratitude to the faculty and staff of the Computer Science Department for their help during my years at Western Michigan University. I am grateful to the Western Michigan University Libraries for funding my graduate assistantship throughout my Ph.D. program. The Libraries staff and faculty, especially those in Information Technology Services, were an immense support during these years. I am also thankful for all the assistance I received from the Graduate College faculty and staff.

Last but not least, I would like to thank my wonderful wife, Sepideh, for all her faithful support and patience during my doctoral study; my beloved daughter, Liana, who is always cheering me up; and my parents, parents-in-law, family members, and friends whose love, encouragement, and support is invaluable to me.

Mehdi Mohammadi

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Introduction

Internet of Things applications can greatly benefit from accurate analytic services and prediction models. Good prediction models highly depend on the quality and quantity of the training data. However, IoT applications usually generate data that can be erroneous and sometimes altogether missing. At the same time, they generate large amounts of data in a dynamic and evolving environment that can be hardly handled by traditional machine learning techniques. In addition, machine learning models need to benefit from the spatio-temporal patterns embedded in the training data. So, new analytic services are needed to get the most value out of IoT data in real deployment scenarios.

In this dissertation, we address these challenges and propose solutions as follows.

Survey of Deep Learning for IoT

We provide a thorough overview on using Deep Learning (DL) techniques to facilitate the analytics and learning in the IoT domain (Chapter 2). We articulate IoT data characteristics and identify two major treatments for IoT data from a machine learning perspective; namely, IoT big data analytics and IoT streaming data analytics. We also discuss why DL is a promising approach to achieve the desired analytics in these types of data and applications. The potential of using emerging DL techniques for IoT data analytics are then discussed, and its promises and challenges are introduced. We present a comprehensive background on different DL architectures and algorithms. We also analyze and summarize major reported research attempts that leveraged DL in the IoT domain. The smart IoT devices that have incorporated DL in their intelligence background are also discussed. DL implementation approaches on the fog and cloud centers in support of IoT applications are also surveyed. In addition, we shed light on some challenges and potential directions for future research.

Semi-supervised Deep Reinforcement Learning for IoT and Smart City

In Chapter 3, we propose a semi-supervised deep reinforcement learning model that fits smart city applications as it consumes both labeled and unlabeled data to improve the performance and accuracy of the learning agent. The model utilizes Variational Autoencoders (VAE) as the inference engine for generalizing optimal policies.

Ensemble of Distributed Deep Learners using Spatio-Temporal Patterns

In Chapter 4, we investigate the creation of a dynamic ensemble from distributed deep learning models by considering the spatio-temporal relationships embedded in the training data. Our dynamic ensemble does not depend on offline configurations. Instead, it exploits the spatio-temporal patterns embedded in the training data to generate dynamic weights for the underlying weak distributed deep learners to create a stronger learner.

Generative Adversarial Networks for Smart Mobility

Smart city applications can benefit from crowd-sourced data where end-users participate in data gathering and annotation tasks through their mobile or wearable devices. In Chapter 5, we investigate the use of Generative Adversarial Networks (GANs) to generate individualized paths that meet the users' needs for path planning use cases. In this work, we propose an architecture that utilizes GAN to learn from crowd-sourced trajectories and recommend accurate and reliable paths based on the expectation of the user.

Problem Statement

IoT data presents several challenges for machine learning, including handling big data, shortage of labeled data, and the need to benefit from the spatio-temporal relations hidden in the training data. Traditional machine learning techniques are not efficient to handle these challenges.

Purpose of the Research

In this research we aim to achieve the following goals.

- Develop machine learning approaches capable of learning the best policies to take actions on the environment while having a limited supervised data. Moreover, our goal is to achieve high accuracy even in situations where data is missing or incomplete.

- Decrease the efforts needed for data pre-processing and making them ready to train prediction models. This includes developing analytic architectures based on crowd-sourced data and raw data features.

Significance of the Study

This study is important since it addresses different challenges that arise in different IoT application domains. The results are of interest to IoT and machine learning researchers and developers to implement more accurate prediction models based on limited training datasets. The results contribute to the development of many IoT systems in different application domains including health, energy, and agriculture to name a few. Moreover, the level of intelligence provided by the proposed techniques is of interest to predict events of interest to the public as in the case of Google Flu Trend [1]. Our ensemble approach has the potential to improve the prediction accuracy of such applications.

Research Questions

In this study, we address the following questions:

1. To what extent, IoT applications can leverage a training mechanism with limited supervision?
2. How deep reinforcement learning techniques can be adopted in IoT applications to train models that adapt their actions according to the dynamic environment of the system?
3. How can we exploit the spatio-temporal patterns embedded in training data to improve the prediction accuracy?

Contributions

The main contributions of this dissertation are as follows.

- We identify and highlight the characteristics of IoT Big data and streaming data and the challenges they present for deep learning techniques. We also review a wide range of IoT applications in different domains that utilize DL architectures in their context. Moreover, we provide a comparison and a guideline for using different DL techniques in the various IoT domains and applications.
- We review the recent approaches and technologies for deploying DL on all levels of the IoT hierarchy from resource constrained devices to the fog and the cloud. We also identified some key challenges

in each of these paths and suggested several future research directions for the successful and fruitful integration of DL and IoT applications.

- We propose a semi-supervised deep reinforcement learning model that fits smart city applications as it consumes both labeled and unlabeled data to improve the performance and accuracy of the learning agent. The model utilizes Variational Autoencoders (VAE) as the inference engine for generalizing optimal policies. As a case study, we focus on smart buildings and apply the proposed model to the problem of indoor localization based on BLE signal strength. Our model learns the best action policies that lead to a close estimation of the target locations with an improvement of 23% in terms of distance to the target and at least 67% more received rewards compared to the supervised DRL model.

- We investigate the creation of a dynamic ensemble from distributed deep learning models by considering the spatio-temporal patterns embedded in the training data. Our dynamic ensemble does not depend on offline configurations. Instead, it exploits the spatio-temporal patterns embedded in the training data to generate dynamic weights for the underlying weak distributed deep learners to create a stronger learner. Our evaluation experiments using three real-world datasets in the context of smart city show that our proposed dynamic ensemble strategy leads to an improved error rate of up to 33% compared to the baseline strategy.

- We evaluate the use of Generative Adversarial Networks (GANs) for path planning in support of smart mobility applications. We propose an architecture based on GANs to recommend accurate and reliable paths for navigation applications. The proposed system can use crowd-sourced data to learn the trajectories and infer new ones. The system provides users with generated paths that help them navigate from their local environment to reach a desired location. Our experiments assert that the generated paths are correct and reliable. The accuracy of the classification task for the generated paths is up to 99% and the quality of the generated paths has a mean opinion score of 89%.

Moreover, this research work has resulted in the following journal and conference publications:

1. M. Mohammadi, A. Al-Fuqaha, "Enabling Cognitive Smart Cities Using Big Data and Machine Learning: Approaches and Challenges," IEEE Communications Magazine, Vol. 56, No. 2, pp. 94-101, Feb 2018.

2. A. Gharaibeh, A. Khreishah, M. Mohammadi, A. Al-Fuqaha, I. Khalil, A. Rayes, "Online Auction of Cloud Resources in Support of the Internet of Things," IEEE Internet of Things Journal, Vol. 4, No. 5, 2017.

3. A. Al-Fuqaha, A. Khreishah, M. Guizani, A. Rayes, M. Mohammadi, "Toward Better Horizontal Integration Among IoT Services," IEEE Communications Magazine, Vol. 53, No. 9, 2015.

4. A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols and Applications," IEEE Communications Surveys and Tutorials, Vol. 17, No. 4, 2015.

We also compiled two datasets that are available online for use by other researchers.

- Localization Dataset: http://archive.ics.uci.edu/ml/datasets/BLE+RSSI+Dataset+for+Indoor+localization+and+Navigation
- Wayfinding, Path Planning, and Navigation Dataset: https://www.kaggle.com/mehdimka/path-planning

Structure of the Dissertation

This dissertation consists of a collection of four papers. The focus of the papers is to investigate the role of deep and reinforcement machine learning techniques to achieve smart services for Internet of Things applications. The rest of the dissertation is organized as follows.

1. Chapter II. Deep Learning for IoT Big Data and Streaming Analytics: A Survey (*Accepted*)
2. Chapter III. Semi-supervised Deep Reinforcement Learning in Support of IoT and Smart City Services (*Published*)
3. Chapter IV. Exploiting the Temporal Relationships in an Ensemble of Distributed Learners in Support of IoT Applications (*Submitted*)
4. Chapter V. Path Planning in Support of Smart Mobility Applications using Generative Adversarial Networks (*Accepted*)
5. Chapter VI. Conclusion and Future Work.

Chapter 2

Deep Learning for IoT Big Data and Streaming Analytics: A Survey

Introduction

The vision of the Internet of Things (IoT) is to transform traditional objects to being smart by exploiting a wide range of advanced technologies, from embedded devices and communication technologies to Internet protocols, data analytics, and so forth [2]. The potential economic impact of IoT is expected to bring many business opportunities and to accelerate the economic growth of IoT-based services. Based on McKinsey's report on the global economic impact of IoT [3], the annual economic impact of IoT in 2025 would be in the range of $2.7 to $6.2 trillion. Healthcare constitutes the major part, about 41% of this market, followed by industry and energy with 33% and 7% of the IoT market, respectively. Other domains such as transportation, agriculture, urban infrastructure, security, and retail have about 15% of the IoT market totally. These expectations imply the tremendous and steep growth of the IoT services, their generated data and consequently their related market in the years ahead.

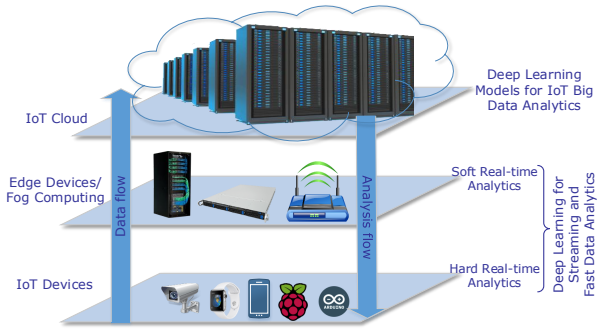


Figure 1: IoT data generation at different levels and deep learning models to address their knowledge abstraction.

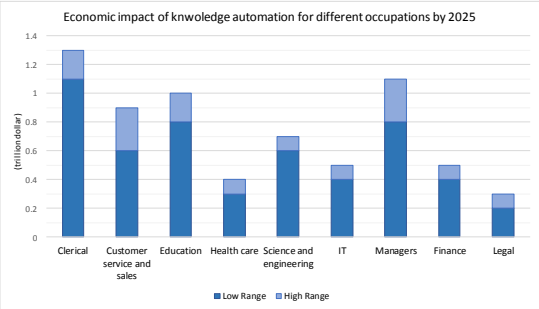


Figure 2: The break down of estimated economic impact of $5.2 trillion to $6.7 trillion per year for machine learning in 2025.

Indeed, machine learning (ML) will have effects on jobs and the workforce, since parts of many jobs may be "suitable for ML applications" [4]. This will lead to increase in demand for some ML products and the derived demand for the tasks, platforms, and experts needed to produce such products. The economic impact of machine learning in McKinsey's report [3] is defined under knowledge work automation; "the use of computers to perform tasks that rely on complex analyses, subtle judgments, and creative problem solving". The report mentions that advances in ML techniques, such as deep learning and neural networks, are the main enablers of knowledge work automation. Natural user interfaces, such as speech and gesture recognition are other enablers that are highly benefiting from ML technologies. The estimated potential economic impact of knowledge work automation could reach $5.2 trillion to $6.7 trillion per year by 2025. Figure shows the break down of this estimate in different occupations. Compared to the economic impact of IoT, this estimation asserts the more attention toward the extraction of value out of data and the potential impacts of ML on the economic situation of individuals and societies. These economic impacts have serious consequences on individuals and countries, since people need to adapt to new means of earning income suitable for them to maintain their desired living standard.

In recent years, many IoT applications arose in different vertical domains, i.e., health, transportation, smart home, smart city, agriculture, education, etc. The main element of most of these applications is an intelligent learning mechanism for prediction (i.e., regression, classification, and clustering), data mining and pattern recognition or data analytics in general. Among the many machine learning approaches, Deep Learning (DL) has been actively utilized in many IoT applications in recent years. These two technologies (i.e., DL and IoT) are among the top three strategic technology trends for 2017 that were announced at Gartner Symposium/ITxpo 2016 [5]. The cause of this intensive publicity for DL refers to the fact that traditional machine learning approaches do not address the emerging analytic needs of IoT systems. Instead, IoT systems need different modern data analytic approaches and artificial intelligence (AI) methods according to the hierarchy of IoT data generation and management as illustrated in Figure 1.

The growing interest in the Internet of Things (IoT) and its derivative big data need stakeholders to clearly understand their definition, building blocks, potentials and challenges. IoT and big data have a

two way relationship. On one hand, IoT is a main producer of big data, and on the other hand, it is an important target for big data analytics to improve the processes and services of IoT [6]. Moreover, IoT big data analytics have proven to bring value to the society. For example, it is reported that, by detecting damaged pipes and fixing them, the Department of Park Management in Miami has saved about one million USD on their water bills [7].

IoT data are different than the general big data. To better understand the requirements for IoT data analytics, we need to explore the properties of IoT data and how they are different from those of general big data. IoT data exhibits the following characteristics [7]:

- Large-Scale Streaming Data: A myriad of data capturing devices are distributed and deployed for IoT applications, and generate streams of data continuously. This leads to a huge volume of continuous data.
- Heterogeneity: Various IoT data acquisition devices gather different information resulting in data heterogeneity.
- Time and space correlation: In most of IoT applications, sensor devices are attached to a specific location, and thus have a location and time-stamp for each of the data items.
- High noise data: Due to tiny pieces of data in IoT applications, many of such data may be subject to errors and noise during acquisition and transmission.

Although obtaining hidden knowledge and information out of big data is promising to enhance the quality of our lives, it is not an easy and straightforward task. For such a complex and challenging task that goes beyond the capabilities of the traditional inference and learning approaches, new technologies, algorithms, and infrastructures are needed [8]. Luckily, the recent progresses in both fast computing and advanced machine learning techniques are opening the doors for big data analytics and knowledge extraction that is suitable for IoT applications.

Beyond the big data analytics, IoT data calls for another new class of analytics, namely fast and streaming data analytics, to support applications with high-speed data streams and requiring time-sensitive (i.e., real-time or near real-time) actions. Indeed, applications such as autonomous driving, fire prediction, driver/elderly posture (and thus consciousness and/or health condition) recognition demands for fast processing of incoming data and quick actions to achieve their target. Several researchers have proposed approaches and frameworks for fast streaming data analytics that leverage the capabilities of cloud infrastructures and services [9, 10]. However, for the aforementioned IoT applications among others, we need fast analytics in smaller scale platforms (i.e., at the system edge) or even on the IoT devices themselves. For example, autonomous cars need to make fast decisions on driving actions such

as lane or speed change. Indeed, this kind of decisions should be supported by fast analytics of possibly multi-modal data streaming from several sources, including the multiple vehicle sensors (e.g., cameras, radars, LIDARs, speedometer, left/right signals, etc.), communications from other vehicles, and traffic entities (e.g., traffic light, traffic signs). In this case, transferring data to a cloud server for analysis and returning back the response is subject to latency that could cause traffic violations or accidents. A more critical scenario would be detecting pedestrians by such vehicles. Accurate recognition should be performed in strict real-time to prevent fatal accidents. These scenarios imply that fast data analytics for IoT have to be close to or at the source of data to remove unnecessary and prohibitive communication delays.

Survey Scope

DL models in general bring two important improvements over the traditional machine learning approaches in the two phases of training and prediction. First, they reduce the need for hand crafted and engineered feature sets to be used for the training [11]. Consequently, some features that might not be apparent to a human view can be extracted easily by DL models. In addition, DL models improve the accuracy[1].

In this paper, we review a wide range of deep neural network (DNN) architectures and explore the IoT applications that have benefited from DL algorithms. The paper identifies five main foundational IoT services that can be used in different vertical domains beyond the specific services in each domain. It will also discuss the characteristics of IoT applications and the guide to matching them with the most appropriate DL model. This survey focuses on the confluence of two emerging technologies, one in communication networks, i.e., IoT and the other in artificial intelligence, i.e., DL, detailing their potential applications and open issues. The survey does not cover traditional machine learning algorithms for IoT data analytics as there are some other attempts, mentioned in section 2, that have covered such approaches. Moreover, this survey also does not go into the details of the IoT infrastructure from a communications and networking perspective.

[1] Accuracy in this work in general refers to the degree to which the result of the prediction conforms to the ground truth values. Readers may also face top-2 or top-3 accuracy in the text. In general, top-N accuracies refers to considering the N highest-probability answers of the prediction model and checking whether that set contains the expected value or not. Therefore, top-1 accuracy refers to the output with the highest probability. Likewise, top-3 accuracy refers to the three most probable predictions. For example, if we feed a picture of a tiger to a model that recognizes animal images, and it returns the list of possible outputs as dog:0.72, tiger:0.69, and cat:0.58, the top-1 accuracy will output the answer set containing only “dog”, which is counted as wrong. On the other hand, the top-2 and top-3 accuracies will result in output sets containing “tiger” as an answer, and are thus counted as correct.

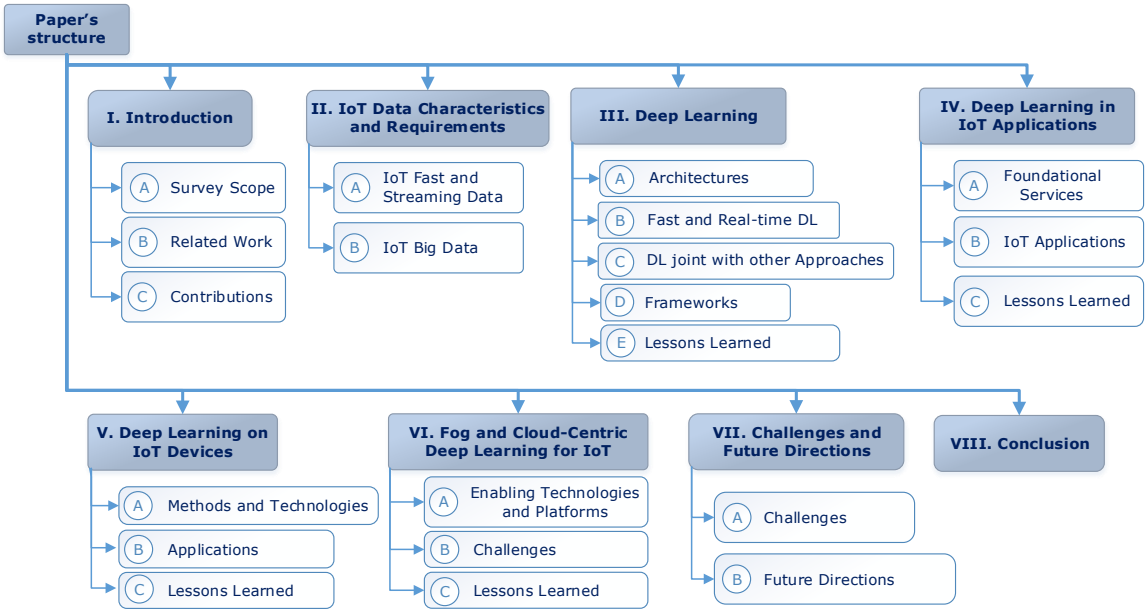


Figure 3: Structure of the survey.

Related Work

To the best of our knowledge, there does not exist an article in the literature that is dedicated to surveying the specific relation between IoT data and DL as well as applications of DL methods in IoT. There are few works presenting common data mining and machine learning methods that have been used in IoT environments. The work presented in [12] by Tsai *et al.* focused on data mining approaches in IoT. It addressed different classification, clustering, and frequent pattern mining algorithms for the IoT infrastructure and services. However, that work did not consider DL approaches, which is the focus of our survey. Moreover, their focus is mainly on offline data mining, while we also consider learning and mining for both real-time (i.e., fast) and big data analytics.

In [13], Perera *et al.* have reviewed different classes of machine learning approaches (supervised and unsupervised, rules, fuzzy logic, etc.) in the reasoning phase of a context-aware computing system, and have discussed the potentials of applying those methods in IoT systems. Nonetheless, they also did not study the role of DL on the context reasoning.

The work in [14] by Alsheikh *et al.* provides a survey of machine learning methods for wireless sensor networks (WSNs). In that work, the authors studied machine learning methods in the functional aspects of WSNs, such as routing, localization, and clustering, as well as non-functional requirements, such as security and quality of service. They reviewed several algorithms in supervised, unsupervised, and reinforcement learning approaches. This work focuses on the infrastructure of WSN (which is one potential infrastructure for implementing IoT applications), while our work is not dependent on the sources of data (i.e., IoT infrastructures) and covers a wide range of IoT applications and services.

Moreover, the focus of [14] was on traditional machine learning methods, whereas this article focuses on advanced and DL techniques.

Finally, Fadlullah *et al.* [15] addressed DL approaches in network traffic control systems. While this work primarily focuses on the infrastructure of network, it differs from our work that focuses on the usage of DL in IoT applications.

Beyond the specific works on the IoT, Qiu *et al.* [16] reviewed several traditional machine learning techniques along with several advanced techniques including DL for processing general big data. In specific, they highlighted the connection of different machine learning techniques with signal processing technologies to process and analyze timely big data applications.

Contributions

This paper is intended for IoT researchers and developers who want to build analytics, AI systems, and learning solutions on top of their IoT infrastructure, using the emerging DL machine learning approaches. The contributions of this paper can be summarized as follows:

- In order to adopt DL approaches in the IoT ecosystems, we identify the key characteristics and issues of IoT data.
- Compared to some related work in the literature that have addressed machine learning for IoT, we review the state-of-the-art DL methods and their applicability in the IoT domain both for big data and streaming data analytics.
- We review a wide range of IoT applications that have used DL in their context. We also provide a comparison and a guideline for using different types of DNN in the various IoT domains and applications.
- We review the recent approaches and technologies for deploying DL on all levels of IoT hierarchy from resource constrained devices to the fog and the cloud.
- We highlight the challenges and future research directions for the successful and fruitful merging of DL and IoT applications.

The rest of this paper is organized as follows. In section 2, we highlight the IoT data characteristics and describe what IoT big data as well as fast and streaming data are, and how they are different from the general big data. Section 2 presents several common and successful architectures of DNNs. It also includes a brief description of advancements toward real-time and fast DL architectures as well as state-of-the-art algorithms that are joint with DL. A succinct review of several frameworks and tools with different capabilities and algorithms that support DNNs is also presented. IoT applications in different

domains (e.g., healthcare, agriculture, ITS, etc.) that have used DL will be surveyed in section 2. Section 2 reviews the attempts to bring DNN to the resource constraint devices. Section 2 explains the works that investigated bringing the DNN models to the scale of fog and cloud computing. Future research direction and open challenges are presented in section 2. The paper is concluded in Section 4 with a summary of its main take-away messages. Figure 3 depicts the structure of the paper.

IoT Data Characteristics and Requirements for Analytics

IoT data can be streamed continuously or accumulated as a source of big data. Streaming data refers to the data generated or captured within tiny intervals of time and need to be promptly analyzed to extract immediate insights and/or make fast decisions. Big data refers to huge datasets that the commonly used hardware and software platforms are not able to store, manage, process, and analyze. These two approaches should be treated differently since their requirements for analytic response are not the same. Insight from big data analytics can be delivered after several days of data generation, but insight from streaming data analytics should be ready in a range of few hundreds of milliseconds to few seconds.

Data fusion and sharing play a critical role in developing ubiquitous environments based on IoT data. This role is more critical for time-sensitive IoT applications where a timely fusion of data is needed to bring all pieces of data together for analysis and consequently providing reliable and accurate actionable insights. Alam *et al.* [17] presented a survey paper in which data fusion techniques for IoT environments are reviewed followed by several opportunities and challenges.

IoT Fast and Streaming Data

Many research attempts suggested streaming data analytics that can be mainly deployed on high-performance computing systems or cloud platforms. The streaming data analytics on such frameworks is based on data parallelism and incremental processing [18]. By data parallelism, a large dataset is partitioned into several smaller datasets, on which parallel analytics are performed simultaneously. Incremental processing refers to fetching a small batch of data to be processed quickly in a pipeline of computation tasks. Although these techniques reduce time latency to return a response from the streaming data analytic framework, they are not the best possible solution for time-stringent IoT applications. By bringing streaming data analytics closer to the source of data (i.e., IoT devices or edge devices) the need for data parallelism and incremental processing is less sensible as the size of the data in the source allows it to be processed rapidly. However, bringing fast analytics on IoT devices introduces its own challenges such as limitation of computing, storage, and power resources at the source of data.

IoT Big Data

IoT is well-known to be one of the major sources of big data, as it is based on connecting a huge number of smart devices to the Internet to report their frequently captured status of their environments. Recognizing and extracting meaningful patterns from enormous raw input data is the core utility of big data analytics as it results in higher levels of insights for decision-making and trend prediction. Therefore, extracting these insights and knowledge from the big data is of extreme importance to many businesses, since it enables them to gain competitive advantages. In social sciences, Hilbert [19] compares the impact of big data analytics to that of the invention of the telescope and microscope for astronomy and biology, respectively.

Several works have described the general features of big data from different aspects [19, 20, 21, 22] in terms of volume, velocity, and variety. However, we adopt the general definition of big data to characterize the IoT big data through the following "6V's" features:

- Volume: Data volume is a determining factor to consider a dataset as big data or traditional massive/ very large data. The quantity of generated data using IoT devices is much more than before and clearly fits this feature.

- Velocity: The rate of IoT big data production and processing is high enough to support the availability of big data in real-time. This justifies the needs for advanced tools and technologies for analytics to efficiently operate given this high rate of data production.

- Variety: Generally, big data comes in different forms and types. It may consist of structured, semi-structured, and unstructured data. A wide variety of data types may be produced by IoT such as text, audio, video, sensory data and so on.

- Veracity: Veracity refers to the quality, consistency, and trustworthiness of the data, which in turn leads to accurate analytics. This property needs special attention to hold for IoT applications, especially those with crowd-sensing data.

- Variability: This property refers to the different rates of data flow. Depending on the nature of IoT applications, different data generating components may have inconsistent data flows. Moreover, it is possible for a data source to have different rates of data load based on specific times. For example, a parking service application that utilizes IoT sensors may have a peak data load in rush hours.

- Value: Value is the transformation of big data to useful information and insights that bring competitive advantage to organizations. A data value highly depends on both the underlying processes/services and the way that data is treated. For example, a certain application (e.g., medical

vital sign monitoring) may need to capture all sensor data, while a weather forecast service may need just random samples of data from its sensors. As another example, a credit card provider may need to keep data for a specific period of time and discard them thereafter.

Beyond the aforementioned properties, researchers [19][21] have identified other characteristics such as:

- Big data can be a byproduct or footprint of a digital activity or IoT interplay. The use of Google's most common search terms to predict seasonal flu is a good example of such digital byproduct [1].
- Big data systems should be horizontally scalable, that is, big data sources should be able to be expanded to multiple datasets. This attribute also leads to the complexity attribute of big data, which in turn imposes other challenges like transferring and cleansing data.

Performing analytics over continuous data flows are typically referred to as stream processing or sometimes complex event processing (CEP) in the literature. Strohbach *et al.* [23] proposed a big data analytics framework for IoT to support the volume and velocity attributes of IoT data analytics. The integration of IoT big data and streaming data analytics, an open issue that needs more investigation, has been also studied as part of that work. However, their proposed framework is designed to be deployed on cloud infrastructures. Moreover, their focus is on the data management aspect of the framework and did not use advanced machine learning models such as DL. Other off-the-shelf products such as Apache Storm are also available for real-time analytics on the cloud. A big gap in this area is the lack of frameworks and algorithms that can be deployed on the fog (i.e., system edge) or even on the IoT devices. When DL comes to play in such cases, a trade-off between the depth and performance of the DNN should be considered.

Deep Learning

DL consists of supervised or unsupervised learning techniques based on many layers of Artificial Neural Networks (ANNs) that are able to learn hierarchical representations in deep architectures. DL architectures consist of multiple processing layers. Each layer is able to produce non-linear responses based on the data from its input layer. The functionality of DL is imitated from the mechanisms of human brain and neurons for processing of signals.

DL architectures have gained more attention in recent years compared to the other traditional machine learning approaches. Such approaches are considered as being shallow-structured learning architectures versions (i.e., a limited subset) of DL. Figure 4 shows the searching trend of five popular machine learning algorithms in Google trends, in which DL is becoming more popular among the others. Although ANNs

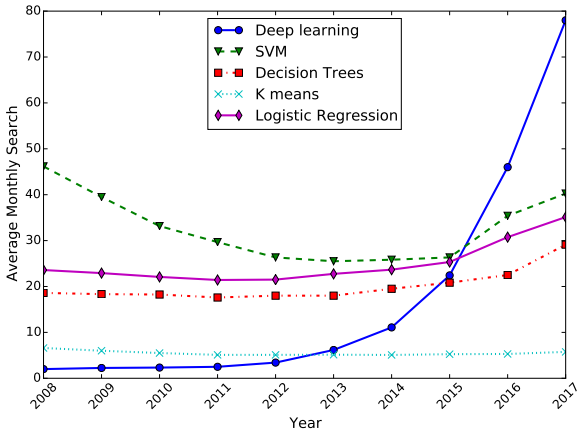


Figure 4: Google trend showing more attention toward deep learning in recent years.

have been introduced in the past decades, the growing trend for DNNs started in 2006 when G. Hinton *et al.* presented the concept of deep belief networks [24]. Thereafter, the state-of-the-art performance of this technology has been observed in different fields of AI including image recognition, image retrieval, search engines and information retrieval, and natural language processing.

DL techniques have been developed on top of traditional ANNs. Feed-forward Neural Networks (FNNs) [25] (a.k.a Multilayer Perceptrons - MLPs) have been used in the past decades to train systems, but when the number of layers is increased, they become difficult to train [26]. The small size of training data was another factor that results in overfitted models. Moreover, the limitation in computational capabilities in those days prohibited the implementation of efficient deeper FNNs. These computational limitations have been resolved lately due to hardware advances in general and the development of Graphics Processing Units (GPUs) and hardware accelerators specifically. Beyond the structural aspects and significance of depth of DL architectures, as well as hardware advances, DL techniques have benefited from advancements in effective training algorithms of deep networks including:

- Using Rectified Linear Units (ReLUs) as activation function [27],
- Introducing dropout methods [28],
- Random initialization for the weights of the network[29],
- Addressing the degradation of training accuracy by residual learning networks [30],
- Solving vanishing gradient problem as well as exploding gradient problem by introducing and enhancing Long Short-Term Memory networks [31, 32].

One advantage of DL architectures, compared to the traditional ANNs, is that DL techniques can learn hidden features from the raw data [11]. Each layer trains on a set of features based on the previous

layer's outputs. The inner-most layers can recognize more complex features, since they aggregate and recombine features from the previous layers. This is called the hierarchy of features. For example, in case of a face recognition model, raw image data of portraits as vector of pixels are fed to a model in its input layer. Each hidden layer can then learn more abstract features from the previous layer's outputs, e.g., the first hidden layer identifies the lines and edges, the second layer identifies face parts such as nose, eyes, etc., and the third layer combines all the previous features to generate a face.

However, the reported improvements of DL models are based on empirical evaluations, and there is still no concrete analytical foundation to answer why DL techniques outperform their shallow counterparts. Moreover, there is no clear boundary between deep and shallow networks based on the number of hidden layers. Generally, neural networks with two or more hidden layers that incorporate the recent advanced training algorithms are considered as deep models. Also, recurrent neural networks with one hidden layer are considered as deep since they have a cycle on the units of the hidden layer, which can be unrolled to an equivalent deep network.

Architectures

In this section, we present a brief overview of several common DL models as well as the most cutting-edge architectures that have been introduced in recent years. Interested readers can refer to other literature that surveyed the models and architectures of DL in more details, such as [33]. Table 1 summarizes these models, their attributes, characteristics, and some sample applications.

A DNN consists of an input layer, several hidden layers, and an output layer. Each layer includes several units called neurons. A neuron receives several inputs, performs a weighted summation over its inputs, then the resulting sum goes through an activation function to produce an output. Each neuron has a vector of weights associated to its input size as well as a bias that should be optimized during the training process. Figure 5 depicts the structure of a neuron.

In the training process, the input layer assigns (usually randomly) weights to the input training data and passes it to the next layer. Each subsequent layer also assigns weights to their input and produces their output, which serves as the input for the following layer. At the last layer, the final output representing the model prediction is produced. A loss function determines the correctness of this prediction by computing the error rate between the predicted and true values. An optimization algorithm such as Stochastic Gradient Descent (SGD) [34] is used to adjust the weight of neurons by calculating the gradient of the loss function. The error rate is propagated back across the network to the input layer (known as backpropagation algorithm [35, 36]). The network then repeats this training cycle, after balancing the weights on each neuron in each cycle, until the error rate falls below a desired threshold. At this point, the DNN is trained and is ready for inference. In Figure 6, the high level mechanism of

Table 1: Summary of deep learning models.

Model	Category	Learning model	Typical input data	Characteristics	Sample IoT Applications
AE	Generative	Unsupervised	Various	• Suitable for feature extraction, dimensionality reduction • Same number of input and output units • The output reconstructs input data • Works with unlabeled data	• Machinery fault diagnosis • Emotion recognition
RNN	Discriminative	Supervised	Serial, time-series	• Processes sequences of data through internal memory • Useful in IoT applications with time-dependent data	• Identify movement pattern • Behavior detection
RBM	Generative	Unsupervised, Supervised	Various	• Suitable for feature extraction, dimensionality reduction, and classification • Expensive training procedure	• Indoor localization • Energy consumption prediction
DBN	Generative	Unsupervised, Supervised	Various	• Suitable for hierarchical features discovery • Greedy training of the network layer by layer	• Fault detection classification • Security threat identification
LSTM	Discriminative	Supervised	Serial, time-series, long time dependent data	• Good performance with data of long time lag • Access to memory cell is protected by gates	• Human activity recognition • Mobility prediction
CNN	Discriminative	Supervised	2-D (image, sound, etc.)	• Convolution layers take biggest part of computations • Less connection compared to DNNs. • Needs a large training dataset for visual tasks.	• Plant disease detection • Traffic sign detection
VAE	Generative	Semi-supervised	Various	• A class of Auto-encoders • Suitable for scarcity of labeled data	• Intrusion detection • Failure detection
GAN	Hybrid	Semi-supervised	Various	• Suitable for noisy data • Composed of two networks: a generator and a discriminator	• Localization and wayfinding • Image to text
Ladder Net	Hybrid	Semi-supervised	Various	• Suitable for noisy data • Composed of three networks: two encoders and one decoder	• Face recognition • Authentication

training for DL models is illustrated.

In a broad categorization, DL models fall into three categories, namely generative, discriminative, and hybrid models. Though not being a firm boundary, discriminative models usually provide supervised learning approaches, while generative models are used for unsupervised learning. Hybrid models incorporate the benefits of both discriminative and generative models.

Convolutional Neural Networks (CNNs)

For vision-based tasks, DNNs with a dense connection between layers are hard to train and do not scale well. One important reason is the translation-invariance property of such models. They thus do not learn the features that might transform in the image (e.g., rotation of hand in pose detection). CNNs have solved this problem by supporting translation-equivariance computations. A CNN receives a 2-D input (e.g., an image or speech signal) and extracts high level features through a series of hidden

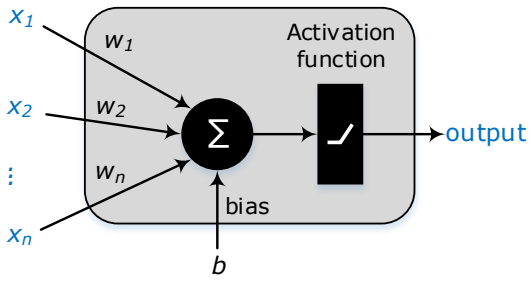


Figure 5: A neuron is a unit of artificial neural networks, with several inputs and trainable weights and bias.

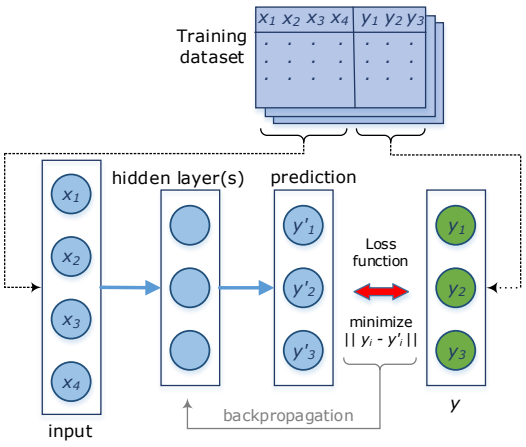


Figure 6: The overall mechanism of training of a DL model.

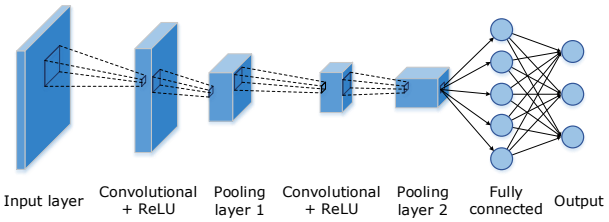


Figure 7: Architecture of a CNN.

layers. The hidden layers consist of convolution layers as well as fully connected layers at the end. The convolution layer is at the core of a CNN and consists of a set of learnable parameters, called filters, that have the same shape as the input's shape but with smaller dimensions. In the training process, the filter of each convolutional layer goes through the whole input volume (e.g., in case of an image, it goes across the width and length of the image) and calculates an inner product of the input and the filter. This computation over the whole input leads to a feature map of the filter.

Another building block of a CNN is the pooling layers, which operate on the feature maps. The objective of having pooling layers is to reduce the spatial size of the representation, in order to both cut down the number of parameters and computation times and to reduce the chance of overfitting. *Max pooling* is a common approach that partitions the input space into non-overlapping regions and picks the maximum value for each region.

The last important component in CNN is ReLU, which consist of neurons with activation function in the form of $f(x) = \max(0, x)$. The introduction of this activation function in CNN results in a faster training time without affecting the generalization of the network in a sensible negative way [37]. Figure 7 depicts the structure of a CNN.

A main difference between CNNs and fully connected networks is that each neuron in CNNs is connected only to a small subset of the input. This decreases the total number of parameters in the network and enhances the time complexity of the training process. This property is called local connectivity.

Many IoT devices, such as drones, smart phones, and smart connected cars, are equipped with cameras. The CNN architecture and its variations have been investigated for a variety of application scenarios that involve these devices. Some typical applications include flood or landslide prediction through drone images, plant disease detection using plant pictures on smart phones, and traffic sign detection using vehicles' cameras.

Recurrent Neural Networks (RNNs)

In many tasks, prediction is dependent on several previous samples such that, in addition to classifying individual samples, we also need to analyze the sequences of inputs. In such applications, a feed-forward neural network is not applicable since it assumes no dependency between input and output layers. RNNs

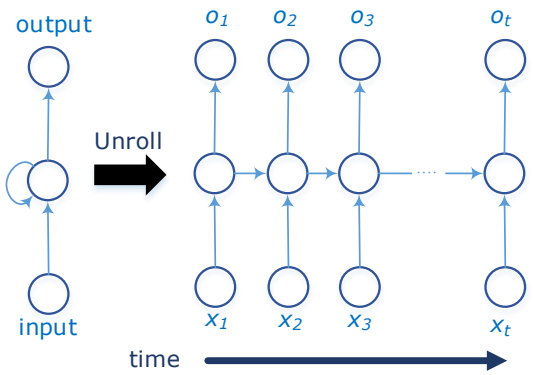


Figure 8: Structure of a recurrent neural network.

have been developed to address this issue in sequential (e.g., speech or text) or time-series problems (sensor's data) with various length. Detecting drivers' behaviors in smart vehicles, identifying individual's movement patterns, and estimating energy consumption of a household are some examples where RNNs can be applied. The input to an RNN consists of both the current sample and the previous observed sample. In other words, the output of an RNN at time step $t-1$ affects the output at time step t. Each neuron is equipped with a feedback loop that returns the current output as an input for the next step. This structure can be expresses such that each neuron in an RNN has an internal memory that keeps the information of the computations from the previous input.

To train the network, an extension of the backpropagation algorithm, called Backpropagation Through Time (BPTT) [38], is used. Due to the existence of cycles on the neurons, we cannot use the original backpropagation here, since it works based on error derivation with respect to the weight in their upper layer, while we do not have a stacked layer model in RNNs. The core of BPTT algorithm is a technique called unrolling the RNN, such that we come up with a feed-forward network over time spans. Figure 8 depicts the structure of an RNN and unrolled concept.

Traditional RNNs can be considered as deep models since they can be seen as several non-linear layers of neurons between the input layer and the output layer when they are unfolded in time [39]. However, considering the architecture and the functionality of RNNs, the hidden layers in RNNs are supposed to provide a memory instead of a hierarchical representation of features [40]. There are several approaches to make RNNs deeper, including adding more layers between the input and hidden layers, stacking more hidden layers, and adding more layers between hidden layers and the output layer [39].

Long Short Term Memory (LSTM)

LSTM is an extension of RNNs. Different variations of LSTM have been proposed, though most of them have followed the same design of the original network [31]. LSTM uses the concept of gates for its units, each computing a value between 0 and 1 based on their input. In addition to a feedback loop

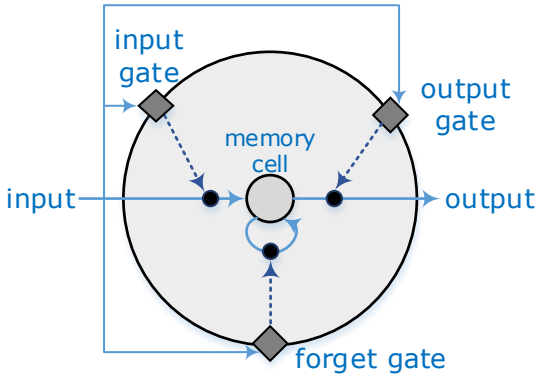


Figure 9: Structure of a LSTM memory cell where the solid arrow lines show the flow of data and dashed arrow lines show the signals coming from gates.

to store the information, each neuron in LSTM (also called a memory cell) has a multiplicative forget gate, read gate, and write gate. These gates are introduced to control the access to memory cells and to prevent them from perturbation by irrelevant inputs. When the forget gate is active, the neuron writes its data into itself. When the forget gate is turned off by sending a 0, the neuron forgets its last content. When the write gate is set to 1, other connected neurons can write to that neuron. If the read gate is set to 1, the connected neurons can read the content of the neuron. Figure 9 depicts this structure.

An important difference of LSTMs compared to RNNs is that LSTM units utilize forget gates to actively control the cell states and ensure they do not degrade. The gates can use *sigmoid* or *tanh* as their activation function. In fact, these activation functions cause the problem of vanishing gradient during backpropagation in the training phase of other models using them. By learning what data to remember in LSTMs, stored computations in the memory cells are not distorted over time. BPTT is a common method for training the network to minimize the error.

When data is characterized by a long dependency in time, LSTM models perform better than RNN models [41]. This long lag of dependency can be observed in IoT applications such as human activity recognition, predicting educational performance in online programs, and disaster prediction based on environmental monitoring, to name a few.

Autoencoders (AEs)

AEs consist of an input layer and an output layer that are connected through one or more hidden layers. AEs have the same number of input and output units. This network aims to reconstruct the input by transforming inputs into outputs with the simplest possible way, such that it does not distort the input very much. This kind of neural networks has been used mainly for solving unsupervised learning problems as well as transfer learning [42]. Due to their behavior of constructing the input at the output layer, AEs are mainly used for diagnosis and fault detection tasks. This is of great interest for industrial

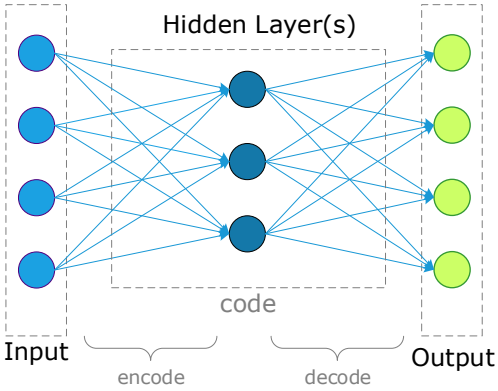


Figure 10: Structure of an autoencoder network.

IoT to serve many applications such as fault diagnosis in hardware devices and machines, and anomaly detection in the performance of assembly lines.

AEs have two main components: An encoder and a decoder. The encoder receives the input and transforms it to a new representation, which is usually called a code or latent variable. The decoder receives the generated code at the encoder, and transforms it to a reconstruction of the original input. The training procedure in AEs involves minimizing reconstruction error, i.e., the output and input showing minimal difference. Figure 10 illustrates the structure of a typical AE. There are several variations and extensions of AEs like denoising AE, contractive AE, stacked AE, sparse AE, and variational AE.

Variational Autoencoders (VAEs)

VAEs, introduced in 2013, are a popular generative model framework whose assumptions on the structure of the data is not strong, while having a fast training process through backpropagation [43]. Moreover, this model has been used for semi-supervised learning [44]. Therefore, it is a good fit for IoT solutions that deal with diverse data and the scarcity of labeled data. Such applications include failure detection in sensing or actuating levels and intrusion detection in security systems. For each data point $\mathbf{x}$, there is a vector of corresponding latent variables denoted by $\mathbf{z}$. The training architecture of a VAE consists of an encoder and a decoder with parameters ϕ and θ, respectively. A fixed form distribution $q_\phi(\mathbf{z}|\mathbf{x})$ helps the encoder in estimating the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$. The model consists of two networks: One generating samples and the other performing approximate inference. A schematic of the VAE is depicted in Figure 22.

Generative Adversarial Networks (GANs)

GANs, introduced by Goodfellow *et al.* [45], consist of two neural networks, namely the generative and discriminative networks, which work together to produce synthetic and high-quality data. The former network (a.k.a. the generator) is in charge of generating new data after it learns the data distribution

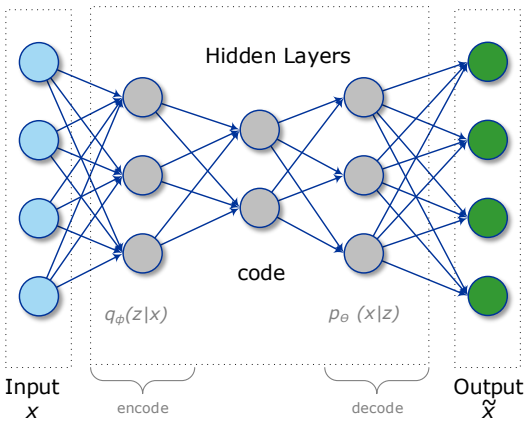


Figure 11: Structure of a variational autoencoder network.

from a training dataset. The latter network (a.k.a. the discriminator) performs discrimination between real data (coming from training data) and fake input data (coming from the generator). The generative network is optimized to produce input data that is deceiving for the discriminator (i.e., data that the discriminator cannot easily distinguish whether it is fake or real). In other words, the generative network is competing with an adversary discriminative network. Figure 44 depicts the concept of GANs.

The objective function in GANs is based on minimax games, such that one network tries to maximize the value function and the other network wants to minimize it. In each step of this imaginary game, the generator, willing to fool the discriminator, plays by producing a sample data from random noise. On the other hand, the discriminator receives several real data examples from the training set along with the samples from the generator. Its task is then to discriminate real and fake data. The discriminator is considered to perform satisfactorily if its classifications are correct. The generator also is performing well if its examples have fooled the discriminator. Both discriminator and generator parameters then are updated to be ready for the next round of the game. The discriminator's output helps the generator to optimize its generated data for the next round.

In IoT applications, GANs can be applied for scenarios that require the creation of something new out of the available data. This can include applications in localization and way-finding, where a generator network in GAN produces potential paths between two points, while the discriminator identifies which paths look viable. GANs are also very helpful for developing services for visually impaired people, such as images-to sound-converters using both GANs to generate descriptive texts from a given image [46] and another DL model to perform text-to-speech conversion. In an image processing research using GANs, a large number of real celebrity snapshots have been analyzed to create new fake images such that a human cannot identify if they are real images or not [47].

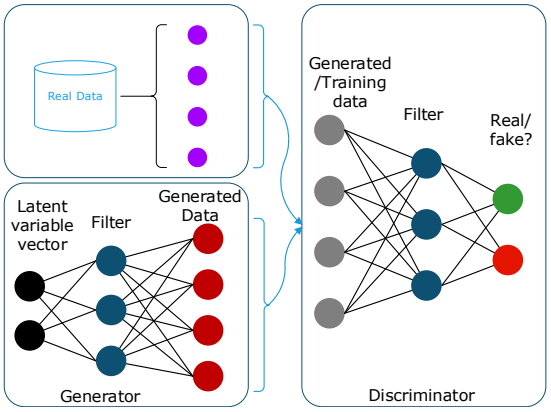


Figure 12: Concept of a generative adversarial network.

Restricted Boltzmann Machine (RBMs)

An RBM is a stochastic ANN that consists of two layers: A visible layer that contains the input that we know, and a hidden layer that contains the latent variables. The restriction in RBMs is applied to the connectivity of neurons compared to Boltzmann machine. RBMs should build a bipartite graph, such that each visible neuron should be connected to all hidden neurons and vice versa, but there is no connection between any two units in a same layer. Moreover, the bias unit is connected to all of the visible and hidden neurons. RBMs can be stacked to form DNNs. They are also the building block of deep belief networks.

The training data is assigned to visible units. The training procedure can use backpropagation and gradient descent algorithms to optimize the weights of the network. The objective of training RBM is to maximize the product of all probabilities of the visible units. The functionality of RBM is similar to the AEs as it uses forward feeding to compute the latent variables, which are in turn used to reconstruct the input using backward feeding. The structure of an RBM is shown in Figure 13.

RBMs can perform feature extraction out of input data. This happens through modeling a probability distribution over a set of inputs that is represented in a set of hidden units. For example, having a set of favorite movies of individuals, an RBM model can have a visible layer consisting of as many neurons as the number of available movies, and a hidden layer consisting of three neurons to represent three different genres such as drama, action and comedy. So, based on the application, the hidden layer can be considered as the output layer. Or it can be complemented with an additional classifier layer to perform classification based on extracted features.

From the types of potential applications where RBMs can be used, we name indoor localization, energy consumption prediction, traffic congestion prediction, posture analysis, and generally any application that benefits from extracting the most important features out of the available ones.

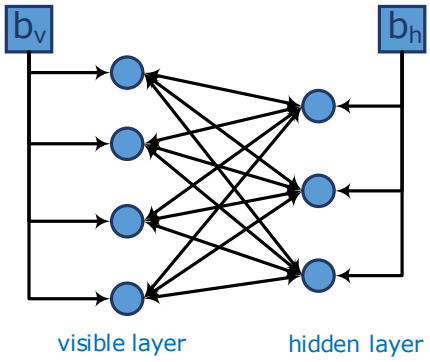


Figure 13: Structure of a restricted Boltzmann machine.

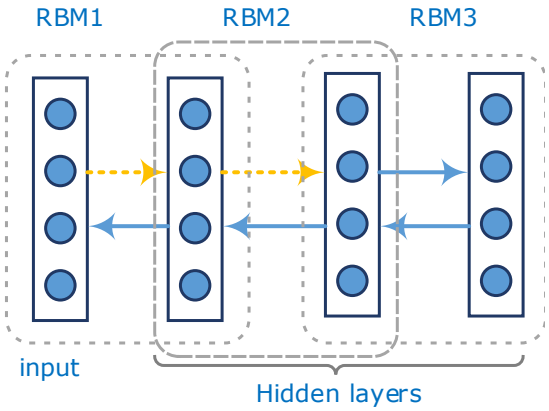


Figure 14: Structure of a deep belief network where the dash arrows show the feature extraction path and solid arrows show the generative path.

Deep Belief Network (DBNs)

DBNs are a type of generative ANNs that consist of a visible layer (corresponding to the inputs) and several hidden layers (corresponding to latent variables). They can extract hierarchical representation of the training data as well as reconstruct their input data. By adding a classifier layer like softmax, it can be used for prediction tasks.

The training of a DBN is performed layer by layer, such that each layer is treated as an RBM trained on top of the previous trained layer. This mechanism makes a DBN an efficient and fast algorithm in DL [48]. For a given hidden layer in DBN, the hidden layer of previous RBM acts as the input layer. Figure 14 shows the structure of a typical DBN.

Several applications can benefit from the structure of DBNs, such as fault detection classification in industrial environments, threat identification in security alert systems, and emotional feature extraction out of images.

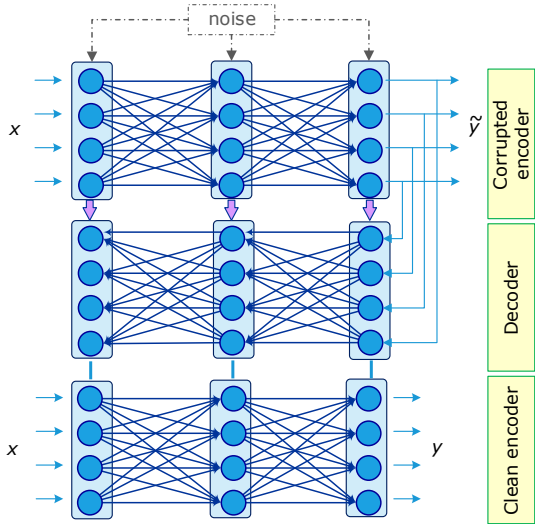


Figure 15: Ladder network structure with two layers.

Ladder Networks

Ladder networks were proposed in 2015 by Valpola *et al.* [49] to support unsupervised learning. Later, they were extended to work in semi-supervised settings [50] and have shown state-of-the-art performance for several tasks, such as handwritten digits recognition and image classification. The architecture of a ladder network consists of two encoders and one decoder. The encoders act as the supervised part of the network and the decoder performs unsupervised learning. One of the encoders, called clean encoder, produces the normal computations while the other encoder, called corrupted encoder, adds Gaussian noise to all layers.

Using a denoising function, the decoder can reconstruct the representations at each layer given the corresponding corrupted data. The difference between the reconstructed and clean data at each layer is used for computing the denoising cost of that layer. In the encoder side, the cost function uses the difference between the corrupted output of encoder layers and the corresponding clean outputs. The training objective is to minimize the sum of cost in the supervised part and unsupervised network.

The initial experimental evaluations of ladder networks [49] are limited to some standard tasks, such as handwritten digits classification over the Modified National Institute of Standards and Technology (MNIST) datasets [51] or image recognition tasks on the datasets of the Canadian Institute for Advanced Research (CIFAR)-10 [52]. Though it has not been used widely in IoT scenarios, ladder networks have the potential to be used in many vision-based IoT analytics where semi-supervision is a great bonus. Figure 15 shows the structure of a ladder network.

Fast and Real-time DL Architectures

The research works for fast and real-time analytics using DL models over the stream of data are still in their infancy. An initial work in this area that utilizes ANNs is done by Liang *et al.* [53]. It has extended the extreme learning machine (ELM) networks to apply an online sequential learning algorithm to single hidden layer feed-forward networks. Their framework, called OS-ELM, learns the training data one-by-one as well as chunk-by-chunk, and only newly arrived data go through the training process. This architecture is the base for the real-time manufacturing execution system that is proposed in [54]. In this work, OS-ELM has been used for shop floor object localization using RFID technology. Zou *et al.* [55] have also reported using this architecture for an indoor localization algorithm based on WiFi fingerprinting, in which the OS-ELM model can bear well the dynamic environmental changes while still showing good accuracy.

For convolutional networks, the architecture proposed by Ren *et al.*, called Faster R-CNN [56] (based on Fast R-CNN [57]), aims to detect objects in images in real-time. Object detection in images needs more computations and hence consumes more energy compared to the image classification tasks, since the system has a large number of potential object suggestions that need to be evaluated. The proposed architecture is based on applying region proposal algorithms in full CNNs that perform object bounds prediction and objectness score computation at each position at the same time. Their evaluation of the proposed object detection architecture indicates that the run time of the system is between 5-17 frames per second (fps) given that the original input frames are re-scaled such that the shortest side of the image would be 600 pixels. Mao *et al.* [58] also used Fast R-CNN for embedded platforms reporting a run time of 1.85 fps with frames scaled to 600 pixels in the shortest side in embedded CPU+GPU platform, which have been shown to be energy-efficient with a close-to-real-time performance. However, for image processing tasks, we can consider an approach to be truly real-time when it can process and analyze 30 fps or better. Redmon *et al.* [59] developed You Only Look Once (YOLO) that has reached the performance of 45 fps for input images resized to 448×448, and even a smaller version of it, Fast YOLO, achieving 155 fps, which are suitable for smart cameras.

Joint DL with Other Approaches

DL architectures also have been used jointly in other machine learning approaches to make them more efficient. The nonlinear function approximation of DL models that can support thousands or even billions of parameters is a strong motivation to use this method in other machine learning approaches in need of such functions. Moreover, the automatic feature extraction in deep models is another motivating reason to exploit these models jointly with other approaches. In the following subsections, a summary

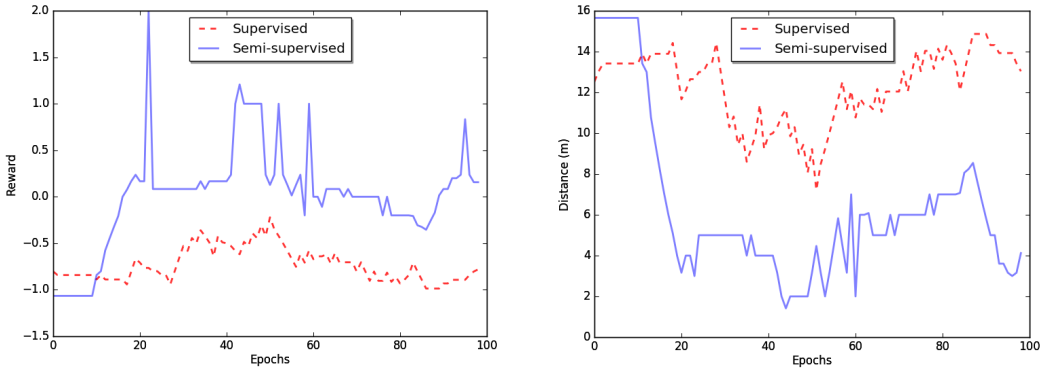


Figure 16: Deep reinforcement learning (supervised and semi-supervised): obtaining rewards (left) and their corresponding accuracy measurement (right) [61].

of such approaches that are suitable for IoT scenarios is provided.

Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) [60] is a combination of reinforcement learning (RL) with DNNs. It aims to create software agents that can learn by themselves to establish successful policies for gaining maximum long-term rewards. In this approach, RL finds the best policy of actions over the set of states in an environment from a DNN model. The need for a DNN in an RL model becomes evident when the underlying environment can be represented by a large number of states. In such situation, traditional RL is not efficient enough. Instead, a DL model can be used to approximate the action values in order to estimate the quality of an action in a given state. Systems that use DRL in their context are in their infancy, but already have showed very promising results. In the field of IoT, the work presented in [61] uses DRL in a semi-supervised setting for localization in smart campus environments. The aim of this work is to localize the users based on received signals from multiple Bluetooth Low Energy (BLE) iBeacons. The learning agent uses DRL to find the best action to perform (i.e., moving in a direction like North, North-West, etc. from a starting point). The reward function is the reciprocal of the distance error to a predefined target, such that the learning agent receives more rewards when it gets closer to its intended target and vice versa. Figure 16 shows a sample result of such method when a DNN model helps for gaining more rewards in a semi-supervised setting (left sub-figure in Figure 16) and its reward interpretation to the accuracy (right sub-figure).

Transfer Learning with Deep Models

Transfer learning, which falls in the area of domain adaptation and multi-task learning, involves the adaptation and improvement of learning in a new domain by transferring the knowledge representation that has been learned from data of a related domain [62]. Transfer learning is an interesting potential

solution for many IoT applications where gathering training data is not an easy task. For example, considering the training of a localization system through BLE or WiFi fingerprinting using smart phones, the RSSI values at a same time and location for different platforms (e.g., iOS and Android) vary. If we have a trained model for one platform, the model can be transferred to the other platform without re-collecting another set of training data for the new platform.

DL models are well matched to transfer learning due to their ability to learn both low-level and abstract representations from input data. Specifically, Stacked denoising AEs [62] and other variations of AEs [63] have been shown to perform very well in this area. Transfer learning with DNNs is still an ongoing and active research area in AI community, and we have not seen reported real-world applications in IoT.

Online Learning Algorithms joint with DL

As the stream of data generated from IoT applications goes through the cloud platforms for analysis, the role of online machine learning algorithms becomes more highlighted, as the training model needs to be updated by the incremental volume of data. This is opposed to what the current technologies support, which is based on batch learning techniques, where the whole training data set should be available for training and, thereafter, the trained model cannot evolve by new data. Several research works report applying online learning techniques on various DL models, including stacked denoising AEs [64], sum-product networks [65], and RBMs [66].

Frameworks

The rapid growth of interest to use DL architectures in different domains has been supported by introducing several DL frameworks in recent years. Each framework has its own strength based on its supported DL architectures, optimization algorithms, and ease of development and deployment [67]. Several of these frameworks have been used widely in research for efficient training of DNNs. In this section, we review some of these frameworks.

H2O: H2O is a machine learning framework that provides interfaces for R, Python, Scala, Java, JSON, and CoffeeScript/JavaScript [68]. H2O can be run in different modes including standalone mode, on Hadoop, or in a Spark Cluster. In addition to common machine learning algorithms, H2O includes an implementation of a DL algorithm, which is based on feed-forward neural networks that can be trained by SGD with backpropagation. H2O's DL AE is based on the standard deep (multi-layer) neural net architecture, where the entire network is learned together, instead of being stacked layer-by-layer.

Tensorflow: Initially developed for Google Brain project, Tensorflow is an open source library for machine learning systems using various kinds of DNNs [69]. It is used by many Google products

including Google Search, Google Maps and Street View, Google Translate, YouTube and some other products. Tensorflow uses graph representations to build neural network models. Developers can also take advantage of TensorBoard, which is a package to visualize neural network models and observe the learning process including updating parameters. Keras[2] also provides a high level of programming abstraction for Tensorflow.

Torch: Torch is an open source framework for machine learning containing a wide range of DL algorithms for easy development of DNN models [70]. It has been developed upon Lua programming language to be light-weight and fast for training DL algorithms. It is used by several companies and research labs like Google, Facebook, and Twitter. It supports developing machine learning models for both CPUs and GPUs, and provides powerful parallelization packages for training DNNs.

Theano: Theano is an open source Python-based framework for efficient machine learning algorithms, which supports compiling for CPUs and GPUs [71]. It uses the CUDA library in optimizing the complicated codes that need to be run on GPUs. It also allows parallelism on CPUs. Theano uses graph representations for symbolic mathematical expressions. Through this representation, symbolic differentiation of mathematical expressions is supported in Theano. Several wrappers including Pylearn2, Keras, and Lasagne provide easier programming experience on top of Theano [72].

Caffe: Caffe [73] is an open source framework for DL algorithms and a collection of reference models. It is based on C++, supports CUDA for GPU computations, and provides interfaces for Python and Matlab. Caffe separates model representation from its implementation. This has been made possible by defining models by configurations without hard-coding them in the source code. Switching between platforms (e.g., CPU to GPU or mobile devices) is easy by only changing a flag. Its speed on GPU is reported to be 1 ms/image for prediction and 4 ms/image for training.

Neon: Neon[3] is another open source DL framework based on Python with high performance for modern DNNs, such as AlexNet [37], Visual Geometry Group (VGG) [74], and GoogleNet [75]. It supports developing several commonly used models, such as CNNs, RNNs, LSTMs, and AEs, on both CPUs and GPUs. The list is being extended as they implemented GANs for semi-supervised learning using DL models. It also supports easy changing of the hardware platform back-ends.

Bahrampour *et al.* in [67] have provided a comparative study for four of the aforementioned tools namely, Caffe, Neon, Theano and Torch. Although the performance of each tool varies in different scenarios, Torch and Theano showed the overall best performance in most of the scenarios. Another benchmarking is provided in [76], comparing the running performance of Caffe, TensorFlow, Torch, CNTK, and MXNet. Table 2 summarizes and compares different DL frameworks.

[2]https://keras.io/
[3]http://neon.nervanasys.com

Table 2: Properties of frameworks for developing deep learning (As of September 2017).

Frameworks	**Core Language**	**Interface**	**Pros**	**Cons**	**Used in IoT Application**
H2O	Java	R, Python, Scala, REST API	• Wide range of interfaces	• Limited number of models are supported • Is not flexible	[77]
Tensorflow	C++	Python, Java, C, C++, Go	• Fast on LSTM training • Support to visualize networks	• Slower training compared to other Python-based frameworks	[78]
Theano	Python	Python	• Supports various models • Fast on LSTM training on GPU	• Many low level APIs	[79]
Torch	Lua	C, C++	• Supports various models • Good documentation • Helpful error debugging messages	• Learning a new language	[78] [80]
Caffe	C++	Python, Matlab	• Provides a collection of reference models • Easy platform switching • Very good at convolutional networks	• Not very good for recurrent networks	[81, 82, 83]
Neon	Python	Python	• Fast training time • Easy platform switching • Supports modern architectures like GAN	• Not supporting CPU multi-threading	[84]
Chainer [85]	Python	Python	• Supports modern architectures • Easier to implement complex architectures • Dynamic change of model	• Slower forward computation in some scenarios	[86]
Deeplearning4j	Java	Python, Scala, Clojure	• Distributed training • Imports models from major frameworks (e.g., TensorFlow, Caffe, Torch and Theano) • Visualization tools	• Longer training time compared to other tools	[87, 88]

Lessons Learned

In this section, we reviewed several common DL architectures that can serve in the analytics component of various IoT applications. Most of these architectures work with various types of input data generated by IoT applications. However, to get better performance for serial or time-series data, RNNs and their variations are recommended. In particular, for long term dependencies among data points, LSTM is more favorable due to its concept of gates for memory cells. For cases where the input data is more than one-dimensional, variations of CNNs work better. RBM, DBN, and variations of AE perform well in handling high-dimensionality reduction, and hierarchical feature extraction. Combined with a classification layer, they can be used for a variety of detection and forecasting scenarios. More recent architectures including VAE, GAN, and Ladder Networks are expected to have a great impact on IoT applications since they cover semi-supervised learning. Those are more favorable for IoT applications where a huge amount of data is generated while a small fraction of that can be annotated for machine learning. The role of these architectures can be emphasized by knowing that only about 3% of all universe data by 2012 was annotated, and is hence useful for supervised machine learning [89]. Table 1 summarizes DL architectures.

A few attempts toward making DL architectures fast and real-time responsive were also discussed. This avenue needs more exploration and research to be applicable in many time-sensitive IoT applications. Emerging machine learning architectures and techniques that both benefit from DL and address the specific IoT application requirements were also highlighted. Indeed, DRL can support autonomousness of IoT applications, transfer learning can fill the gap of lack of training data sets, and online learning matches the need for stream analysis of IoT data.

We also reviewed several common and powerful frameworks for the development of DL models. For IoT applications, training times, run times, and dynamic update of the trained models are determining factors for a reliable and efficient analytic module. Most of the current frameworks follow the pattern of "define-and-run" instead of "define-by-run" [85]. The former does not allow dynamic updates of the model while the latter supports such modifications. Chainer [85] is a framework that follows the latter pattern and can handle dynamic changes of the model.

DL Applications in IoT

DL methods have been shown promising with state-of-the-art results in several areas, such as signal processing, natural language processing, and image recognition. The trend is going up in IoT verticals. Some neural network models work better in special domains. For example, convolutional networks provide better performance in applications related to vision, while AEs perform very well with anomaly detection,

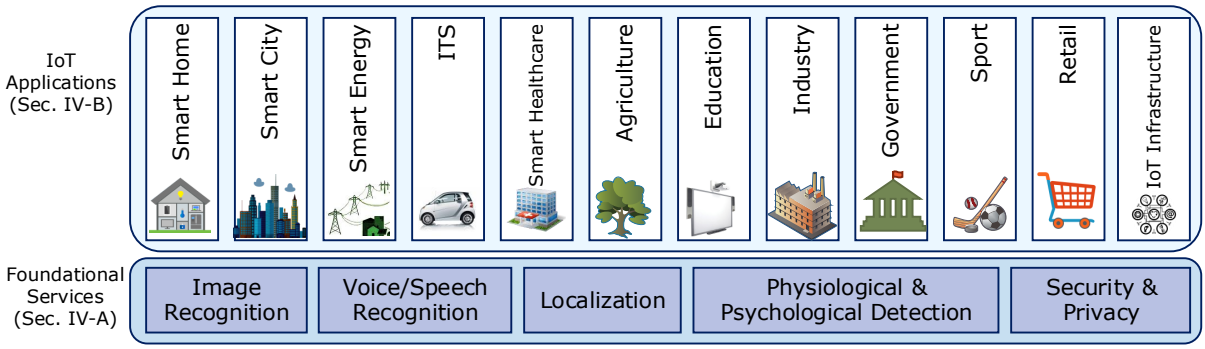


Figure 17: IoT applications and the foundational services.

data denoising, and dimensionality reduction for data visualization. It is important to make this link between the kind of neural network model that best fits each of the different application domains.

In this section, we review successful applications of DL in IoT domains. Based on our observation, many IoT related applications utilize vision and image classification (like traffic sign recognition, or plant disease detection that we will discuss in Section 2) as their base intelligent service. There are other services, such as human pose detection, which are utilized for smart home applications or intelligent car assistance. We identify several kinds of these services as foundational services on which other IoT applications can be built. The common property of these services is that they should be treated in a fast analytic mode instead of piling their data for later analytics. Indeed, each domain may have specific services beyond these foundational services. Figure 17 shows the foundational services and the IoT applications on top of them.

In the following subsections, we first review foundational services of IoT that use DL as their intelligence engine, then highlight the IoT applications and domains where a combination of foundational services as well as specific ones may be utilized.

Foundational Services

Image Recognition

A large portion of IoT applications address scenarios in which the input data for DL is in the form of videos or images. Ubiquitous mobile devices equipped with high resolution cameras facilitate generating images and videos by everyone, everywhere. Moreover, intelligent video cameras are used in many places like smart homes, campuses, and manufacturers for different applications. Image recognition/classification and object detection are among the fundamental usages of such devices.

One issue with the IoT-related systems that have addressed image recognition is the use of specific source datasets for evaluation of their performance. Most of these systems employ the available common image datasets such as the MNIST dataset of handwritten digits [51], VGG face dataset [74], CIFAR-10

and CIFAR-100 tiny images dataset, etc. Though being good for comparison with other approaches, those datasets do not show the specific characteristics of IoT systems. For example, the input for the task of vehicle detection in smart connected cars would not be always a clear image, and there are cases where the input image is at night, or in a rainy or foggy weather. These cases are not handled through the available datasets and hence the models trained based on these datasets are not comprehensive enough.

Speech/Voice Recognition

With the massive proliferation of smart mobile devices and wearables, automatic speech recognition is becoming a more natural and convenient way for people to interact with their devices [90]. Also, the small size of mobile devices and wearables nowadays lower the possibility of having touch screens and keyboards as a means of input and interaction with these devices. However, the main concern for providing speech/voice recognition functionality on resource-constrained devices is its energy-intensiveness, especially when the data is processed through neural networks. In a typical speech recognition neural network model, voice data is represented as the raw input to the network. The data is processed through the hidden layers, and the likelihood of the voice data to a particular speech sound is presented at the output layer.

Price *et al.* [91] have reported that they have built a special-purpose low-power DL chip for automatic speech recognition. The new specialized chip consumes a tiny amount of energy between 0.2 and 10 milliwatts, 100 times lesser than the energy consumption for running a speech recognition tool in current mobile phones. In the new chip, DNNs for speech recognition have been implemented. For the sake of energy saving, three levels of voice activity recognition are designed with three separate neural networks, each of which having a different level of complexity. A lowest complexity network, thus consuming the lowest amount of energy, detects voice activity by monitoring the noise in the environment. If this network identifies a voice, the chip runs the next complexity level recognition network whose task is acoustic modeling to identify if the voice looks like speech. If the output of this network is a high likelihood, then the third network, having the highest energy consumption, is triggered to run to identify individual words.

Indoor Localization

Providing location aware services, such as indoor navigation and location aware marketing in retailers, are becoming prevalent in indoor environments. Indoor localization may also have applications in other sectors of IoT, such as in smart homes, smart campuses, or hospitals. The input data generated from such applications usually comes from different technologies, such as vision, visible light communication (VLC), infrared, ultrasound, WiFi, RFID, ultrawide band, and Bluetooth. For the approaches based on

WiFi or Bluetooth, most of the literature have used mobile phones for receiving signals from the fixed transmitters (i.e., access points or iBeacons), which are called fingerprints. Among these fingerprinting approaches, several attempts reported the use of DL models to predict the location [92, 93, 94].

DL has been used successfully to locate indoor positions with high accuracy. In a system called DeepFi [92], a DL method over fingerprinting WiFi channel state information data has been utilized to identify user positions. This system consists of offline training and online localization phases. In the offline training phase, DL is exploited to train all the weights based on the previously stored channel state information fingerprints. Other works [93], [94] report using variations of DL models in conjunction with other learning methods to extract features and estimate positions. These experiments assert that the number of hidden layers and units in DL models has a direct effect on the localization accuracy. In [95], a CNN is used for indoor localization by fusion of both magnetic and visual sensing data. Moreover, a CNN has been trained in [96] to determine the indoor positions of users by analyzing an image from their surrounding scene.

Lu *et al.* have also used LSTM networks for localizing soccer robots [97]. In this application, data collected from several sensors, namely Inertia Navigation System (INS) and vision perceptions, are analyzed to predict the position of the robot. The authors reported improved accuracy and efficiency compared to two baseline methods, namely standard Extended Kalman Filtering (EKF) and the static Particle Filtering.

Physiological and Psychological State Detection

IoT combined with DL techniques has been also employed to detect various physiological and psychological states of humans, such as pose, activity, and emotions. Many IoT applications incorporate a module for human pose estimation or activity recognition to deliver their services, e.g., smart homes, smart cars, entertainment (e.g., XBox), education, rehabilitation and health support, sports, and industrial manufacturing. For example, convenient applications in smart homes are built based on the analysis of occupant's poses. The cameras transfer the video of the occupant to a DNN to find out the pose of the person and take the most appropriate action accordingly. Toshev *et al.* [98] report a system employing a CNN model to achieve this goal. This sort of services can also be used in education to monitor the attention of students, and in retail stores to predict the shopping behavior of customers [99].

Ordonez *et al.* [100] have proposed a DL framework that combines the strength of CNN and LSTM neural networks for human activity recognition from wearable sensor data (accelerometer, gyroscope, etc.). Their model consists of four convolutional layers with ReLUs followed by two LSTM layers and a softmax layer. They showed that this combination outperformed a baseline model that is just based on convolutional layers by 4% on average. The work of Tao *et al.* [101] also used LSTM architecture for

human activity recognition based on mobile phone sensor's data. Li *et al.* [102] also report the usage of raw data from passive FRID tags for detecting medical activities in a trauma room (e.g., blood pressure measurement, mouth exam, cardiac lead placement, etc.) based on a deep CNN.

In [103], a combined model of CNN and RNN was proposed for gesture recognition in video frames. This model showed better results compared to the models without such combination, and asserted the importance of the recurrence component for such task. In [104], Fragkiadaki *et al.* proposed a DNN model called Encoder-Recurrent-Decoder (ERD) for human body pose recognition and motion prediction in videos and motion capture data sets. The proposed model consisted of an RNN with an encoder before the recurrent layers and a decoder after them. This architecture was shown to outperform Conditional Restricted Boltzmann Machines (CRBMs) for this application.

Beyond the physical movements, emotion estimation of humans from video frames has been also investigated in [105] using a model that consists of a CNN, DBN, and AE. Furthermore, the work in [106] used mobile inertial sensor data for motion detection. It confirmed that human motion patterns can be used as a source of user identification and authentication. The employed model in this system is a combination of convolutional layers and clockwork RNN.

Security and Privacy

Security and privacy is a major concern in all IoT domains and applications. Smart homes, ITS, Industry, smart grid, and many other sectors consider security as a critical requirement. Indeed, the validity of the functionality of the systems depends on protecting their machine learning tools and processes from attackers.

False Data Injection (FDI) is a common type of attack on data-driven systems. In [107], He *et al.* proposed a Conditional DBN to extract FDI attack features from the historical data of smart grids, and use these features for attack detection in real-time. The work in [108] is also related to anomaly detection that may occur in vehicle networks.

Smart phones as great contributers to IoT data and applications are also under serious threats of hacker attacks. Consequently, protecting these devices from a variety of security issues is necessary for IoT perspectives beyond the users' concerns. Yuan *et al.* [109] proposed a DL framework to identify malwares in Android apps. Their architecture is based on a DBN by which they reported accuracy of 96.5% to detect malware apps.

The security and privacy preservation of deep machine learning approaches are the most important factors for the acceptance of using these methods in IoT sectors. Shokri *et al.* [110] proposed a method to address the privacy preservation issues in DL models when they are subject to distributed learning. Their approach was able to preserve both the privacy of participants' training data and the accuracy of

the models at the same time. The core of their approach is based on the fact that stochastic gradient descent optimization algorithms, used in many DL architectures, can be performed in a parallel and asynchronous way. Individual participants can thus independently train the model on their own data and share a portion of their model parameters with other participants. Abadi *et al.* [111] also proposed a method for privacy guarantee in DL models using differentially private stochastic gradient descent algorithm.

Applications

Smart Homes

The concept of smart homes involve a broad range of applications based on IoT, which can contribute to enhancing homes' energy usage and efficiency, as well as the convenience, productivity, and life-quality of their occupants. Nowadays, home appliances can connect to the Internet and provide intelligent services. For example, Microsoft and Liebherr in a collaborative project are applying Cortana DL to the information gathered from inside the refrigerator [112]. These analytics and predictions can help the household to have a better control on their home supplies and expenses, and, in conjunction with other external data, can be used for monitoring and predicting health trends.

Over one third of the generated electricity in the U.S. is consumed by the residential sector [113], with HVAC and lighting devices consisting the largest source of such consumption in buildings. This demand is expected to grow in a slower pace by smart management of energy as well as the efficiency improvements in appliances. Hence, the ability to control and improve energy efficiency and predict the future need is a must for smart home systems. In the smart home applications, electricity load prediction are the most common applications that employ different DL networks to figure out the task. Manic *et al.* [113] performed a comparison analysis of load forecasting for home energy consumption using three DL architectures, including LSTM, LSTM Sequence-to-Sequence (S2S) and CNN. Their results show that LSTM S2S predicts the future usage better than the other architectures, followed by CNN, and then LSTM. They also compared the same dataset over a conventional ANN, and all of the aforementioned models outperformed the ANN model.

Smart City

Smart city services span over several IoT domains, such as transportation, energy, agriculture [114], etc. However, this area is more interesting from a machine learning perspective as the heterogeneous data coming from different domains lead to big data, which can result in high-quality output when analyzed using DL models. Smart city benefits from advances in other domain to achieve efficient

resource management for the whole city. For example, to improve public transportation infrastructure and offer new improved services, getting analytics and patterns out of public transportation behaviors is of interest for local authorities.

Toshiba has recently developed a DL testbed jointly with Dell Technologies, and used this testbed in a Smart Community Center in Kawasaki, Japan, to evaluate the data collected in the Center [115]. The aim of running the testbed is to measure the effectiveness of using DL architectures in IoT ecosystems, and identify the best practices for service improvement including increasing machines' availability, optimizing monitoring sensors, and lowering maintenance expenses. The big data that feeds the testbed were gathered from building management, air conditioning and building security.

One important issue for smart city is predicting crowd movements patterns, and their use in public transportation. Song *et al.* [116] developed a system based on DNN models to achieve this goal on a city level. Their system is built upon a four-layer LSTM neural network to learn from human mobility data (GPS data) joint with their transportation transition modes (e.g., stay, walk, bicycle, car, train). They treated the prediction of people's mobility and transportation mode as two separated tasks instead of joining all these features together. Consequently, their learning system is based on a multi-task deep LSTM architecture to jointly learn from the two sets of features. The choice of LSTM was driven by the spatio-temporal nature of human mobility patterns. The authors assert that their approach based on multi-task deep LSTM achieves better performance compared to both shallow LSTMs having only one single LSTM layer as well as deep LSTMs without multi-tasking.

Liang *et al.* [117] presented a real-time crowd density prediction system in transportation stations that leverages the mobile phone users' telecommunication data known as caller detail record (CDR). CDR data are gathered when a user takes a telecommunication action (i.e., call, SMS, MMS, and Internet access) on the phone, which usually includes data about user ID, time, location, and the telecommunication action of the user. They built their system based on an RNN model for metro stations, and reported more accurate predictions compared to nonlinear autoregressive neural network models.

Waste management and garbage classification is another related task for smart cities. A straightforward method to perform this automation is through vision-based classifications using deep CNNs as it has been done in [81]. Monitoring air quality and predicting the pollution is another direction for city management. Li *et al.* [118] developed a DL-based air quality prediction model using a stacked AE for unsupervised feature extraction and a logistic regression model for regression of the final predictions.

Amato *et al.* in [119] developed a decentralized system to identify the occupied and empty spots in parking lots using smart cameras and deep CNNs. The authors deployed a small architecture of a CNN on smart cameras, which are equipped with Raspberry Pi 2 model. These embedded devices in smart cameras can thus run the CNN on each device to classify images of individual parking spaces as

Table 3: Typical IoT-based services in smart city.

Service	Reference	Input data	DL model
Crowd density/ transportation prediction	[116]	GPS/ transition mode	LSTM
	[117]	Telecommunication data/CDR	RNN
Waste management	[81]	Garbage images	CNN
Parking lot management	[119, 120]	Images of parking spaces	CNN

occupied or empty. The cameras then send only the classification output to a central server. Valipour *et al.* [120] also developed a system for detecting parking spots using CNN, which has shown improved results compared to SVM baselines. Table 3 summarizes the aforementioned attempts.

Energy

The two way communication between energy consumers and the smart grid is a source of IoT big data. In this context, smart meters are in the role of data generation and acquisition for the fine grained level of energy consumption measurement. Energy providers are interested to learn the local energy consumption patterns, predict the needs, and make appropriate decisions based on real-time analytics. Mocanu *et al.* in [121] have developed a kind of RBM to identify and predict the buildings' energy flexibility in real-time. Energy flexibility is about modifying a household's electricity consumption while minimizing the impact on the occupants and operations. In the mentioned work, time-of-use and consumption of individual appliances are predicted to achieve flexible energy control. The advantage of this model beyond showing good performance and accuracy is that flexibility identification can be performed with flexibility prediction concurrently. In [122], two variations of RBMs are used to forecast energy consumption for short term intervals in residential houses. The model includes a Conditional RBM (CRBM) and a Factored Conditional RBM (FCRBM). Their results indicate that FCRBM performs better that CRBM, RNN and ANN. Moreover, by extending the forecasting horizon, FCRBM and CRBM show more accurate predictions than the RBM and ANN.

On the smart grid side, forecasting the power from solar, wind, or other types of natural sustainable sources of energy is an active research field. DL is increasingly used in many applications in this domain. For example, in [123], Gensler *et al.* investigate the performance of several DL models, such as DBNs, AEs, and LSTMs, as well as MLP for predicting the solar power of 21 photovoltaic plants. For solar power prediction, a main element of the input is a numeric value for weather forecasting in a given time horizon. From their evaluation, the combination of AEs and LSTMs (Auto-LSTM) has been shown to produce the best results compared to other models, followed by DBN. The reason for obtaining a good

prediction score by Auto-LSTM is that they are able to extract features from raw data, which is not the case for ANN and MLP. In [86], an online forecasting system based on LSTM is proposed to predict the solar flare power 24 hours ahead.

Intelligent Transportation Systems

Data from Intelligent Transportation Systems (ITS) is another source of big data that is becoming ubiquitous every day. Ma *et al.* [79] presented a system of transportation network analysis based on DL. They have employed RBM and RNN architectures as their models in a parallel computing environment, and GPS data from participating taxies as the input of the models. The accuracy of their system to predict traffic congestion evolution over one hour of aggregated data is reported to be as high as 88% which was computed within less than 6 minutes. [124] also reported the investigation on short-term traffic flow prediction. They used LSTM as their learning model and reported better accuracy for LSTM compared to other methods including SVM, simple feed forward neural networks, and stacked AEs. For different intervals (15, 30, 45, and 60 min) LSTM showed the lowest mean absolute percentage error (MAPE) rate. However, for short intervals of 15 minutes, the error rate of SVM is slightly higher than the LSTM model. This result can be interpreted by the fact that the small number of data points in short intervals does not make stronger discrimination boundaries for the classification task in the LSTM model compared to the SVM model. In another study [108], ITS data are exposed to an intrusion detection system based on DNN to improve the security of in-vehicular network communications.

ITS also motivate the development of methods for traffic sign detection and recognition. Applications such as autonomous driving, driver assistance systems, and mobile mapping need such sort of mechanisms to provide reliable services. Cireşan *et al.* [125] presented a traffic sign recognition system based on DNNs of convolutional and max-pooling layers. They introduced a multi-column DNN architecture that includes several columns of separate DNNs, and reported increased accuracy with this approach. The input is preprocessed by several different preprocessors, and a random number of columns receives the preprocessed input to proceed with training. The final prediction output is the average of all the DNNs' outputs. Their results show that this proposed method, achieving a recognition rate of 99.46%, has been able to recognize traffic signs better than the humans on the task with 0.62% more accuracy.

In order to be applicable in real scenarios, these analytics need to be performed in real-time. Lim *et al.* in [126] proposed a real-time traffic sign detection based on CNN that has been integrated with a general purpose GPU. They reported F1 measure of at least 0.89 in their results with data having illumination changes. To have a faster inference engine, they used a CNN with two convolutional layers.

Furthermore, self-driving cars use DNNs in performing many tasks, such as detecting pedestrians, traffic signs, obstacles, etc. There are several startups that use DL in their self-driving cars to perform

different tasks when driving in the streets [127].

Healthcare and Wellbeing

IoT combined with DL has been also employed in providing healthcare and wellbeing solutions for individuals and communities. For instance, developing solutions based on mobile apps to accurately measure dietary intakes is a track of research that can help control the health and wellbeing of individuals. Liu *et al.* in [82] and [128] developed a system to recognize food images and their relevant information, such as types and portion sizes. Their image recognition algorithm is based on CNNs that achieved competitive results compared to the baseline systems.

DL for classification and analysis of medical images is a hot topic in the healthcare domain. For example, Pereira *et al.* [129] used the idea of recognizing handwritten images by CNNs to help identifying Parkinson's disease in its early stages. Their model learns features from the signals of a smart pen that uses sensors to measure handwritten dynamics during the individual's exam. Muhammad *et al.* [130] propose a voice pathology detection system using IoT and cloud frameworks, in which patients' voice signals are captured through sensor devices and are sent to a cloud server for analytics. They used an extreme learning machine trained by voice signals to diagnose the pathology. In [131], DL was employed for detection of cardiovascular diseases from mammograms. In their study, Wang *et al.* used breast arterial calcification (BAC) revealed in mammograms as a sign of coronary artery disease. They developed a CNN with twelve layers to identify the existence of BAC in a patient. Their results show that the accuracy of their DL model is as good as the human experts. Although this work has been done offline, it shows the potential of developing or extending mammogram devices in IoT contexts for online and early detection of such diseases.

Feng *et al.* [132] report the use of RBMs and DBNs for fall detection in a home care environment. Normal postures in such environment are standing, sitting, bending, and lying. Lying on the floor longer than a threshold is considered as a fallen posture. Their evaluation shows that RBM outperforms DBN in terms of classification accuracy. The lack of large datasets and performing offline detection are the restrictions of their method.

Researchers also used time series medical data in conjunction with RNN based models for early diagnosis and prediction of diseases. Lipton *et al.* [133] investigated the performance of LSTM networks to analyze and recognize patterns in multivariate time series of medical measurements in intensive care units (ICUs). The input data in their system consist of sensor data of vital signs as well as lab test results. Their performance results show that an LSTM model trained on raw time-series data outperforms a MLP network. A survey of DL in health informatics is provided in [134].

Agriculture

Producing healthy crops and developing efficient ways of growing plants is a requirement for a healthy society and sustainable environment. Disease recognition in plants using DNNs is a direction that has shown to be a viable solution. In a study that is reported by Sladojevic *et al.* [83], the authors developed a plant disease recognition system based on the classification of leave images. They have used a deep convolutional network model implemented using the Caffe framework. In this model, diseased leaves in 13 categories can be identified from the healthy ones with an accuracy of about 96%. Such recognition model can be exploited as a smart mobile applications for farmers to identify the fruit, vegetable, or plant disease based on their leaf images captured by their mobile devices. It can also allow them to select remedies or pesticides in conjunction with complementary data.

DL also has been used in remote sensing for land and crop detection and classification [135] [136] [137]. The direction established in these works enabled the automated monitoring and management of the agricultural lands in large scales. In most of such works, deep convolutional networks are used to learn from images of the land or crops. In [135], it is reported that using CNN has yielded an accuracy of 85% in detecting major crops, including wheat, sunflower, soybeans, and maize, while outperforming other approaches such as MLP and random forest (RF).

Furthermore, DL has been reported to be utilized for prediction and detection tasks for automatic farming. For example, [138] has used a DL model based on deep CNNs for obstacle detection in agricultural fields, which enables autonomous machines to operate safely in them. The proposed system was able to detect a standardized object with an accuracy between 90.8% to 99.9%, based on the field (e.g., row crops or grass mowing).

Moreover, fruit detection and finding out the stage of fruit (raw or ripe) is critical for automated harvesting. In [139], Sa *et al.* used a variation of CNN, called Region-based CNN, for image analysis of fruits. The input image of the system comes in two modes: one containing RGB colors and the other is near-infrared. The information of these images are combined in the model and has achieved detection improvement compared to pixel-based training models.

Education

IoT and DL contribute to the efficiency of education systems, from kindergarten to higher education. Mobile devices can gather learners' data and deep analytical methods can be used for prediction and interpretation of learners progress and achievements. Augmented reality technology combined with wearables and mobile devices are also potential applications for DL methods in this area to make students motivated, lessons and studies to be interesting, and make educational learning methods to be efficient[140, 141]. Moreover, DL can be used as a personalized recommendation module [142] to recom-

mend more relevant content to the educator. The applications of DL in other domains, such as natural language translation and text summarization, would be of help for smart education when it comes to online learning on mobile devices.

Furthermore, the advent of Massive Open Online Courses (MOOCs) and their popularity among the students has led to generating a huge amount of data from the learners' behavior in such courses. MOOCs analysis can help identify struggling students in early sessions of a course, and provide sufficient support and attention from instructors to those students to achieve a better performance. Yang *et al.* [143] proposed a method for predicting student grades in MOOCs. They use clickstream data collected from lecture videos when students are watching the video and interacting with it. Clickstream data are fed to a time series RNN that learns from both prior performance and clickstream data. In addition, Piech *et al.* applied RNN and LSTM networks to model the prediction of educator answers to exercises and quizzes, based on their past activities and interactions in MOOCs [144]. Results showed improvement over Bayesian Knowledge Tracing (BKT) methods, which employ a Hidden Markov Model (HMM) for updating probabilities of single concepts. Mehmood *et al.* [77] also used DNNs for a personalized ubiquitous e-teaching and e-learning framework, based on IoT technologies, aiming for the development and delivery of educational content in smart cities. Their proposed framework is built on top of an IoT infrastructure (e.g., smart phone sensors, smart watch sensors, virtual reality technologies) connecting the users in order to optimize the teaching and learning processes. They used DNN for human activity recognition to deliver adaptive educational content to the students.

Classroom occupancy monitoring is another application that has been investigated by Conti *et al.* [145]. In this work, the authors propose two methods for head detection and density estimation, both based on CNN architecture for counting students in a classroom. The algorithms have been deployed on off-the-shelf embedded mobile ARM platform. Their algorithm receives the images that are taken from the cameras in three classrooms with a rate of three pictures every 10 minutes. They report that the root-mean-square (RMS) error of their algorithms is at most 8.55.

Industry

For the industry sector, IoT and cyber-physical systems (CPS) are the core elements to advance manufacturing technologies toward smart manufacturing (a.k.a Industry 4.0). Providing high-accuracy intelligent systems is critical in such applications, as it directly leads to increased efficiency and productivity in assembly/product lines, as well as decreased maintenance expenses and operation costs. Therefore, DL can play a key role in this field. Indeed, a wide range of applications in industry (such as visual inspection of product lines, object detection and tracking, controlling robots, fault diagnosis, etc.) can benefit from introduction of DL models.

In [78], visual inspection is investigated using CNN architectures including AlexNet and GoogLeNet over different platforms (Caffe, Tensorflow, and Torch). In this work, several images of produced vehicles in the assembly line along with their annotation are submitted to a DL system. It has been found that the best performance is achieved using Tensorflow with accuracy of 94%. Moreover, Tensorflow was the fastest framework in terms of training time, where the model reached its peak accuracy in a shorter time, followed by Torch and then Caffe.

Shao *et al.* [146] used DNNs for feature extraction in a fault diagnosis (also referred as fault detection and classification (FDC)) system for rotating devices. Models using denoising auto-encoder (DAE) and contractive auto-encoder (CAE) were developed. The learned features from these models were both refined using a method called locality preserving projection (LPP), and fed to a softmax classifier for fault diagnosis. The input to the system is vibration data of a rotating device. In their system, seven operating conditions were considered, including normal operation, rubbing fault, four degrees of unbalance faults and compound faults (rub and unbalance). Given the vibration data, the diagnosis system identifies whether the device is in normal condition or in one of the fault conditions. Based on their experiments for fault diagnosis of rotor and locomotive bearing devices, the proposed approach is reported to outperform CNN and shallow learning methods.

In another study reported by Lee [147], a DBN model was proposed in conjunction with an IoT deployment and cloud platform to support fault detection of defect types in cars' headlight modules in a vehicle manufacturer setting. Their results confirmed the superior performance of the DBN model over two baseline methods, using SVM and radial basis function (RBF), in terms of error rate in test dataset. However, the reported error rate for their training dataset in the DBN model is comparable to that of the SVM model.

For the problem of fault detection and classification (FDC) in noisy settings, [148] employed stacked denoising AEs (SdA) to both reduce the noise of sensory data caused by mechanical and electrical disturbances, and perform fault classification. Their system was applied for fault detection in wafer samples of a photolithography process. Results show that SdA leads to 14% more accuracy in noisy situations compared to several baseline methods including K-Nearest Neighbors and SVM. Yan *et al.* [149] have also used SdA joint with extreme learning machines for anomaly detection in the behavior of gas turbine combustion system. Based on their results, the use of learned features by SdA leads to a better classification accuracy compared to the use of hand crafted features in their system.

Government

Governments can gain great potential advantages through enhanced and intelligent connectivity that comes from the convergence of IoT and DL. Indeed, a wide variety of tasks that pertains to the gov-

ernments or city authorities require precise analysis and prediction. For instance, the recognition and prediction of natural disasters (landslide, hurricane, forest fires, etc.) and environmental monitoring is of high importance for governments to take appropriate actions. Optical remote sensing images that are fed to a deep AEs network and softmax classifiers were proposed by Liu *et al.* [150] to predict geological landslides. An accuracy of 97.4% was reported for the proposed method, thus outperforming SVM and ANN models. In another study [151], an LSTM network is used for the prediction of earthquakes. They used the historical data from US Geological Survey website for training. Their system was shown to achieve an accuracy of 63% and 74% with 1-D and 2-D input data, respectively. In another study by Liu *et al.* [84], a CNN architecture is used for detection of extreme climate events, such as tropical cyclones, atmospheric rivers and weather fronts. Training data in their system included image patterns of climate events. The authors developed their system in Neon framework and achieved accuracy of 89%-99%.

In addition, damage detection in the infrastructures of the cities, such as roads, water pipelines, etc., is another area where IoT and DL can provide benefits to governments. In [152], the problem of road damage detection was addressed using DNNs that gets its data through crowd-sourcing, which can be enabled by IoT devices. Citizens can report the damage through a mobile app to a platform. However, these citizens have no expert knowledge to accurately assess the status of road damage, which may lead to uncertain and/or wrong assessments. To eliminate these instances, the app can determine the status of the road damage by analyzing the image of the scene. The analysis is performed by a deep CNN that is trained by citizen reports as well as road manager inspection results. Since the training phase is out of the capability of mobile phones, the DL model is created on a server and trained everyday. An Android application can then download the latest model from the server upon each launch, and identify the status of road damages reported by images. Evaluations showed a damage classification accuracy of 81.4% in 1 second of analysis on the mobile devices.

Sport and Entertainment

Sports analytics have been evolving rapidly during the recent years and plays an important role to bring a competitive advantage for a team or player. Professional sport teams nowadays have dedicated departments or employees for their analytics [153]. Analytics and predictions in this field can be used to track the players' behavior, performance, score capturing, etc. DL is new to this area and only few works have used DNNs in different sports.

In [154], a DL method has been proposed for making an intelligent basketball arena. This system makes use of SVM to choose the best camera for real-time broadcasting from among the available cameras around the court. They also fed basketball energy images[4] to a CNN to capture the shoot and scoring

[4]Basketball energy image is the spatial-temporal tracks of basketballs in the hotspot area. Hotspot area includes

clips from the non-scoring ones, hence providing accurate online score reporting and interesting highlight clips. This system was shown to achieve an accuracy of 94.59% in capturing score clips with 45 ms of processing time for each frame.

In another work by Wang *et al.* [155], an RNN has been used for classification of offensive basketball plays in NBA games. The authors used video clips of the games from SportVU[5] dataset. This dataset provides videos of the rate of 25 frames per second to detect players' unique ID, their location on the court, and the position of the ball. Their model is shown to achieve accuracy of 66% and 80% for top-1 and top-3 accuracies, respectively. Similarly, [156] used an RNN with LSTM units over the same dataset to predict the success rates of three-point shots, and reported better classification accuracy compared to gradient boosted machine (GBM) and generalized linear model (GLM).

Kautz *et al.* [157] investigated players' activity recognition in volleyball. Wearable sensor data and CNN were employed to achieve this task, and a classification accuracy of 83.2% to identify players activities was observed.

Group activity recognition is another interesting direction for sport teams. Ibrahim *et al.* [158] investigated this option in a volleyball team using a hierarchical LSTM model. In this work, a single LSTM model was built to derive the activities of each player, and a top-level LSTM model was designed to aggregate the individual models to identify the overall behavior of the team. A CNN model was utilized to extract features from video frames, and feed them to the individual LSTM models. Compared to several baseline models, the proposed hierarchical model obtained better classification results.

Retail

Due to the proliferation of mobile devices, online shopping has increased greatly. A recent shift toward product image retrieval through visual search techniques was noticed [159]. CNNs have been used for this visual search of clothes and fashion market, to find items in online shops that are identical or similar to what you have seen in a movie [160] or in the street [161].

Moreover, shopping for visually impaired people needs to be made convenient. A combination of IoT technologies, including smart carts, integrated with DL methods can be a solution to this problem. In [162], a visual grocery assistance system that includes smart glasses, gloves, and shopping carts was designed to help visually impaired people in shopping. This system also used a CNN to detect items in the aisles.

Moreover, check-out counters in retail stores are usually the bottlenecks where people queue up to pay their shoppings. The development of smart carts can enable real-time self check-out and enhancing

basketball backboard, hoop, and basket.

[5]http://go.stats.com/sportvu

such system with prediction capabilities can offer an item that a customer may need based on his/her past shopping.

Furthermore, recommending items to shoppers is a popular application of IoT for retails that uses different technologies, like BLE signals or visual cameras. The latter approach can be done through identifying the shop items or shoppers actions (e.g., reach to a shelf, retract from a shelf, etc.) [163] and providing a list of related items for the detected action.

To analyze the customer interest in merchandise, Liu *et al.* [99] proposed a customer pose and orientation estimation system based on a DNN consisting of a CNN and RNN. The input data comes from surveillance cameras. The CNN network is used to extract image features. The image features and the last predicted orientation features are then fed to an RNN to get the output pose and orientation.

Smart IoT Infrastructure

IoT environments consist of a large number of sensors, actuators, media and many other devices that generate big M2M and network traffic data. Therefore, the management, monitoring, and coordination of these devices are subject to processing such big data, with advanced machine learning techniques, to identify bottlenecks, improve the performance, and guarantee the quality of service.

One popular task for infrastructure management would be anomaly detection. For example, spectrum anomaly detection in wireless communications using AE was proposed by Feng *et al.* in [164]. In this work, an AE model was developed to detect the anomaly that may happen due to sudden change in signal-to-noise ratio of the communications channel. The model is trained on features based on the time–frequency diagram of input signals. Their result showed that a deeper AE performs better than the conventional shallow networks. Lopez-Martin *et al.* [165] and Shone *et al.* [166] have used conditional VAE and deep AEs, respectively, for network intrusion detection. In the conditional VAE, the labels of the samples are used in addition to the latent variables as extra inputs to the decoder network.

The tiny traces of IoT traffic may not lead to congestion at the backbone. However, the need to access the channel simultaneously by a large number of IoT devices can lead to contention during the channel access phase. The contention in channel access turns to a severe problem when the access delays are increased [167]. Therefore, load balancing is a viable solution that can be performed by DL models to predict the traffic metrics and propose alternate routes. Kim *et al.* [168] used DBNs to perform load balancing in IoT. Their DL model is trained on a large amount of user data and network loads. Interference identification can also be handled by DNNs as demonstrated by Schmidt *et al.* [169], where a wireless interference identification systems based on CNN was proposed. Ahad *et al.* [170] provided a survey focused on the application of neural networks to wireless networks. They reviewed related literature on quality of service and quality of experience, load balancing, improved security, etc.

Since the emerging 5th Generation (5G) cellular networks constitute one of the main pillars of IoT infrastructure, it is necessary to utilize cutting-edge technologies to enhance the different aspects of cellular networks, including radio resource management, mobility management, service provisioning management, self-organization, and to find an efficient and accurate solution for complicated configuration problems [171]. As part of these efforts, using crowd-sourced cellular networks data (e.g., signal strength) can help to come up with reliable solutions. For example, such big data can be used to create more precise coverage maps for cellular networks to improve the performance of the network [172], as was performed by the OpenSignal mobile application [173].

Lessons Learned

In this section, we have identified five classes of IoT services as the foundational services that can be used in a wide range of IoT applications. We discussed how DL has been used to achieve these services. Moreover, we went through a wide range of IoT domains to find out how they exploit DL to deliver an intelligent service. Table 4 shows the works that utilized foundational services in IoT domains.

Many IoT domains and applications have greatly benefited from image recognition. The interest is expected to grow faster as the high-resolution cameras embedded in smart phones will result in easier generation of image and video data. The usage of other fundamental applications, especially physiological and psychological detections as well as localization, can be seen in different fields. However, the utilization of security and privacy services is shown to be limited. This is the gap in developing intelligent IoT applications, where the potential activities of hackers and attackers are ignored. Also, voice recognition with DL has not been used widely in IoT applications belonging to several domains, such as smart homes, education, ITS, and industry. There are works that use voice recognition with traditional machine learning approaches. Voice recognition has shown remarkable advancement with DL. One reason for the few appearance of this technique in IoT applications is the lack of comprehensive training datasets for each domain, as there is a need for large training datasets to train voice recognition DNNs.

Foundational services need fast data analytics to be efficient in their context. Despite several works in this direction, IoT fast data analytics based on DL has many spaces for development of algorithms and architectures.

Table 5 summarizes the research in each domain, and their DL model. Figure 18 also depicts the frequency of different models that have been used in the different research works. About 43% of the papers have used CNN in building their proposed systems while DBN are less used compared to other models (about 7%). RNNs and LSTMs together, as time-series models, have been used in 30% of the works. The table also emphasizes the great impact of works related to image recognition on IoT applications. Moreover, one third of the IoT applications are related to time-series or serial data, in which employing

Table 4: Usage of foundational services in IoT domains.

		IoT Foundational Services				
		Image Recognition	**Voice Recognition**	**Physiological & Psychological Detection**	**Localization**	**Security & Privacy**
IoT Domains	Smart Home					
	Smart City	[81, 119, 120]			[116, 117]	
	Energy					[107]
	ITS	[125, 126]				[108]
	Healthcare	[82, 128, 129, 131]	[130]	[132]		
	Agriculture	[83, 135, 136, 137, 138, 139]				
	Education	[145]		[77]		
	Industry	[78, 147]			[54]	
	Government	[150, 152, 84]				
	Sport	[154, 155, 156]		[157, 158]	[97]	
	Retail	[159, 160, 161, 162]		[99]	[163]	

RNNs is a helpful approach.

Complexity vs. Performance

Canziani *et al.* [174] analyzed several state-of-the-art DNN models to find out the relationship between their accuracy, memory usage, operations count, inference time, and power consumption. They found out that the accuracy and inference time show a hyperbolic relationship such that a minor increase in accuracy leads to a long computational time. They also illustrated that the number of operations in a network model have a linear relationship with the inference time. Their results also indicated that imposing an energy constraint would limit the maximum achievable accuracy. Regarding the memory footprint and batch size, the results showed that the maximum memory usage is constant during the initial memory allocation of the model and then linearly increases with the batch size. Given that the neurons are the main building blocks of a model that performs the operations, the number of operations is proportional to the number of neurons. So, the complexity can be expressed as the number of neurons in the network, such that increasing the number of neurons directly impacts the run-time.

However, there is not a clear relationship between the accuracy and number of layers (i.e., depth) or number of neurons. There are reports indicating degradation of accuracy after increasing the number of layers beyond some point. For example, Zhang *et al.* [94] assert that the number of hidden layers and neurons have a direct effect on the accuracy of the localization system. Increasing the layers initially leads to better results, but at some point when the network is made deeper, the results start degrading. Their best result was obtained with a network of three hidden layers. On the other hand, the depth of representations has been shown to be most beneficial for many image recognition tasks [30]. The high accuracy of vision-based tasks, in part, is due to introducing deeper networks with larger number of parameters (e.g., over 1000 layers as presented in [30, 175]). There are many hyper-parameters to optimize (e.g., epoch count, loss function, activation function, optimization function, learning rate,

etc.) that complicate the process of developing good and accurate DL models. Table 6 summarizes the characteristics of DNN models in several applications. In the table, the test time is for one sample unless specified otherwise.

Pitfalls and Criticisms

DL models were demonstrated to be as a great step toward creating powerful AI systems, but they are not a single solution for all problems. DL techniques are known as black boxes that show high predictability but low interpretability. While powerful prediction capability is most desired from the scientific perspective, the interpretability and explicability of the models are preferred from a business perspective [176]. In [177], Chollet argued that problems requiring reasoning, long-term planning, and algorithmic-like data manipulation, cannot be solved by deep learning models. This is because of the nature of DL techniques that only transform one vector space into another, no matter how much data you feed them.

Moreover, there are criticisms on the performance of DNN models, suggesting that the traditional models may achieve comparable results or even outperform deep models [178]. According to Chatfield *et al.* [179], the dimensionality of the convolutional layers in CNNs can be shrunk without adversely affecting the performance. They also discussed that the shallow techniques can reach a performance analogous to that of deep CNN models if the former models use the data augmentation techniques that are commonly applied to CNN-based methods. Ba *et al.* [180] performed several empirical tests asserting that shallow FNNs can learn the complex functions and achieve accuracies that were previously possible only by deep models. In their work on optical communication systems, Eriksson *et al.* [181] showed that using pseudo random bit sequences or short repeated sequences can lead to overestimating the signal-to-noise ratio.

Generally, DL models are sensitive to the structure, and size of the data. Compared to shallow models, they work better when there is a large body of training data with a wide range of attributes. Otherwise, shallow models typically lead to better results.

DL on IoT Devices

Prior to the era of IoT, most research on DL targeted the improvement of its models and algorithms to efficiently operate when the scale of the problem grows to the big data, by trying to deploy efficient models on cloud platforms. The emergence of IoT has then opened up a totally different direction when the scale of the problems shrank down to resource-constrained devices and to the need for real-time analytics.

Table 5: The common use of different DNN models in IoT domains.

Domain	**Usage of DNNs**					
	AE	**CNN**	**DBN**	**LSTM**	**RBM**	**RNN**
Image Recognition	[150]	[81, 82, 83, 119, 120] [125, 126, 129, 131] [135, 138, 145] [154, 158, 160]	[147]	[156]		[155, 156]
Physiological & Psychological Detection	[104, 105]	[98, 100, 102] [103, 105, 106]	[105]	[100, 101]		[103, 104, 106]
Localization	[93, 94]	[95, 96]		[97]	[92]	
Privacy and Security		[110]	[107, 109]			
Smart home		[113]		[113]		
Smart city		[81, 119, 120]		[116]		[117]
Energy	[123]		[123]	[123] [86]	[121] [122]	[122]
ITS		[125, 126]	[108]	[124]	[79]	[79]
Healthcare		[128, 82, 129, 131]	[132]	[133]	[132]	
Agriculture		[83, 135, 136, 137, 138, 139]				
Education		[145]		[144]		[143, 144]
Industry	[146, 148, 149]	[78]	[147]			
Government	[150]	[152, 84]		[151]		
Sport		[154, 157, 158]		[156, 158]		[155]
Retail		[159, 160, 161, 162]				[163]
IoT Infrastructure	[164, 165, 166]	[169]	[168]			

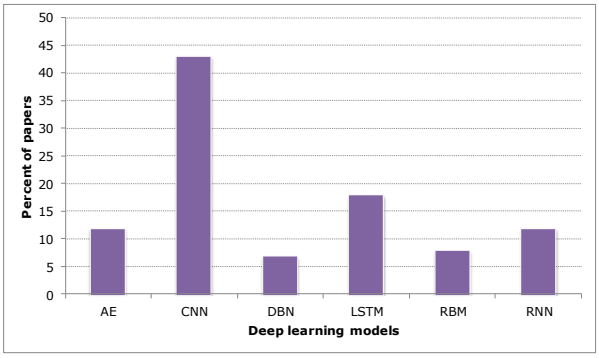


Figure 18: The percentage of surveyed papers that have used DL models.

Table 6: DNN sizes and complexities in different applications.

(NA: Not Available, C: Convolutional Layer, F: Fully Connected layer, FF: Feedforward layer, SG: Summation (collapse) layer, P: Pooling layer, LRN: local response normalization layer, SM: Softmax layer, L: LSTM layer, R: RNN layer, RBM: RBM hidden layer)

Work	**Application**	**Type of DNN**	**Depth**	**Layers Sizes**	**Training Time**	**Test Time**
[79]	Transportation analysis	RNN+RBM	2	R(100)-RBM(150)	NA	354 (s), whole test set
[92]	Localization	RBM DBN	4 4	500-300-150-50 300-150-100-50	NA	NA
[93]	Localization	SdA DBN ML-ELM SDELM	4 3 5 5	26-200-200-71 26-300-71 26-300-300-1500-71 26-300-300-1500-71	451 (s) 110 (s) 14 (s) 24 (s)	NA
[94]	Localization	SdA	5	163-200-200-200-91	NA	0.25 (s)
[98]	Pose detection	CNN	12	C(55×55×96)-LRN-P- C(27×27×256)-LRN-P- C(13×13×384)- C(13×13×384)- C(13×13×256)-P- F(4096)-F(4096)-SM	NA	0.1 (s)
[100]	Human activity detection	CNN+LSTM	7	C(384)-C(20544)-C(20544)- C(20544)-L(942592)- L(33280)-SM	340 (min)	7 (s), whole test set
[101]	Human activity detection	LSTM	5	L(4)-FF(6)-L(10)-SG-SM	NA	2.7 (ms)
[107]	FDI detection	DBN	4	50-50-50-50	NA	1.01 (ms)
[109]	Malware detection	DBN	3	150-150-150	NA	NA
[120]	Parking space	CNN	8	C(64×11×11)-C(256×5×5)- C(256×3×3)-C(256×3×3)- C(256×3×3)-F(4096)-F(2)-SM	NA	0.22 (s)
[126]	Traffic sign detection	CNN	6	C(36×36×20)-P- C(14×14×50)-P- FC(250)-SM	NA	29.6 (ms) on GPU 4264 (ms) on CPU
[128]	food recognition	CNN	22	Used GoogLeNet [75]	NA	50 (s)
[135]	Crop recognition	CNN	6	C(96×7×7)-P- C(96×4×4)-P-F(96)-F(96)	12 (h)	NA
[145]	Classroom Occupancy	CNN	5	C(6×28×28)-P- C(16×10×10)-P-F(120)	2.5 (h)	2 (s) (4 thread)
[146]	Fault diagnosis	AE	4	300-300-300-300	NA	91 (s)
[152]	Road damage detection	CNN	8	Used AlexNet [37]	NA	1.08 (s)
[155]	Classifying offensive plays	RNN	3	10-10-11	NA	10 (ms)

Smart objects need to support some sort of light-weight intelligence. Due to DL's successful results in speech and video applications, which are among the fundamental services and common uses of IoT, adapting its models and approaches for deployment on resource-constrained devices became a very crucial point of study. So far, DL methods can hardly be used in IoT and resource-constrained devices for training purposes since DL models require a large portion of resources, such as the processors, battery energy, and memory. In some cases, the available resources are even not sufficient for running a pre-trained DL algorithm for inference tasks [80]. Luckily, it has been recently shown that many parameters that are stored in DNNs may be redundant [182]. It is also sometimes unnecessary to use a large number of hidden layers to get a high accuracy [180]. Consequently, efficiently removing these parameters and/or layers will considerably reduce the complexity of these DNNs without significant degradation of the output [182, 180] and make them IoT-friendly. In the remaining of this section, we will discuss methods and technologies to achieve this results, and illustrate their applications in different domains.

Methods and Technologies

DL models may consist of millions or even billions of parameters which need sophisticated computing and large storage resources. In this section, we discuss several state-of-the-art approaches that bring DL models to IoT embedded and resource constrained devices.

Network Compression

One way of adopting DNNs to resource-constrained devices is through the use of network compression, in which a dense network is converted to a sparse network. This approach helps in reducing the storage and computational requirements of DNNs when they are used for classification or other sorts of inference on IoT devices. The main limitation of this approach is that they are not general enough to support all kinds of networks. It is only applicable to specific network models that can exhibit such sparsity.

Another interesting study to adopt compressed DL models on IoT devices is the one performed by Lane *et al.* [80]. In this study, the authors measure different factors that embedded, mobile, and wearable devices can bear for running DL algorithms. These factors included measurements of the running time, energy consumption, and memory footprint. The study focused on investigating the behavior of CNNs and DNNs in three hardware platforms that are used in IoT, mobile, and wearable applications, namely *Snapdragon 800* used in some models of smart phones and tablets, *Intel Edison* used in wearable and form-factor sensitive IoT, and *Nvidia Tegra K1* employed in smart phones as well as IoT-enabled vehicles. Torch has been used for developing and training DNNs, and AlexNet [37] was the dominant model used in these platforms. Their measurement of energy usage indicated that all the platforms, including Intel Edison (which is the weakest one), were able to run the compressed models. In terms of execution

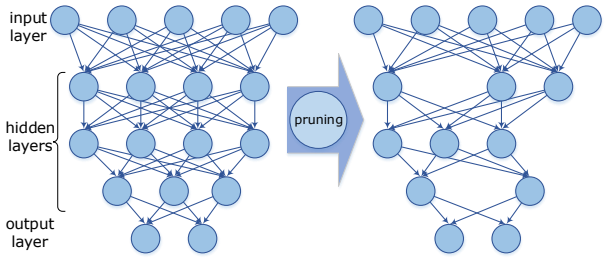


Figure 19: The overall concept of pruning a DNN.

time for CNNs, it has been shown that the later convolutional layers tend to consume less time as their dimensions decrease.

Moreover, it is known that feed-forward layers are much faster than the convolutional layers in CNNs. Consequently, a good approach for improving CNN models on resource-constrained devices is to replace convolutional layers with feed-forward layers whenever possible. In addition, choosing the employed activation function in DNNs can have a great effect on time efficiency. For instance, several tests have shown that ReLU functions are more time-efficient followed by Tanh, and then Sigmoid. However, the overall runtime reduction of such selection is not significant (less than 7%) compared to the execution time of layers (at least 25%). In terms of memory usage, CNNs use less storage than DNNs due to the fewer stored parameters in convolutional layers compared to their counterpart in DNNs.

As previously stated, reducing the number of employed parameters in DNNs, by pruning redundant and less-important ones, is another important approach to make DNNs implementable on resource-constrained devices. One of the first works on this approach is Optimal Brain Damage [183] in 1989. At the core of this method, the second order derivative of the parameters are used to compute the importance of the parameters and prune unimportant parameters from the network. The method presented in [184] also works based on pruning redundant and unnecessary connections and neurons as well as using weight sharing mechanisms. Weight sharing replaces each weight with an n bit index from a shared table that has 2^n possible values. The steps to prune the network as describe by Han *et al.* in [184] consist of:

- Train the network to find the connections with high weights.
- Prune the unimportant connections that have a weight less than a threshold.
- After pruning, there may remain some neurons with no input nor output connections. The pruning process identifies these neurons and removes them as well as all their remaining connections.
- Retrain the network to fine-tune the weight of the updated model. The weights should be transferred from the previous trained steps instead of initializing them, otherwise the performance will degrade to some extent.

Table 7: Methods and technologies to bring DL on IoT devices.

Method / Technology	Reference	Pros	Cons
Network Compression	[184] [186] [187]	• Reduce storage and computation	• Not general for all DL models • Need specific hardware • The pruning process bring overload to training
Approximate Computing	[188, 189]	• Makes fast DL models • Save energy	• Not suitable for precise systems
Accelerators	[91] [185] [190] [191] [192] [193]	• Integrates DL model with the hardware • Efficient computations	• Does not work with the traditional hardware platforms
Tinymote with DL	[194]	• Good for time-critical IoT apps • Energy-efficient • Provides more security and privacy for data	• Special-purpose networks

The authors evaluated this approach on four models related to vision, namely AlexNet, VGG-16, LeNet-300-100, and LeNet-5. The models were compressed at least 9 times for AlexNet and 13 times at VGG-16, while the accuracy of the models were almost preserved. One limitation of this approach is that it cannot be used for other types of DNN models. Moreover, the resulting compressed networks are not efficient enough on all hardware platforms and CPU architectures, and thus need new kinds of accelerators that can handle dynamic activation sparsity and weight sharing. Figure 19 illustrates the concept of pruning a DNN.

In [185], an inference engine, called EIE, was designed with a special hardware architecture and SRAMs instead of DRAMs, and was shown to work well with compressed network models. In this architecture, customized sparse matrix vector multiplication and weight sharing are handled without losing the efficiency of the network. The engine consists of a scalable array of processing elements (PEs), each of which keeping a part of the network in an SRAM and performing its corresponding computations. Since most of the energy that is used by neural networks is consumed for accessing the memory, the energy usage is reported to be 120 times fewer with this designed accelerator than the energy consumption of the corresponding original network.

In HashedNets [186], the neural network connection weights are randomly grouped into hash buckets using a hash function. All connections that fall into a same bucket are represented by a single parameter. Backpropagation is used to fine-tune the parameters during the training. Testing results show that the accuracy of this hash-based compressed model outperforms all other compression baseline methods.

The work in [187] by Courbariaux *et al.* proposed to binarize network weights and neurons at both the inference phase and the entire training phase, in order to reduce the memory footprint and accesses. The network can also perform most of the arithmetic operations through bit-wise operations, leading to a decreased power consumption. MNIST, CIFAR-10 and Street View House Numbers (SVHN) [195] datasets were tested over Torch7 and Theano frameworks using this approach, and results were found to be promising.

Approximate Computing

Approximate computing is another approach to both implement machine learning tools on IoT devices and contribute to the energy saving of their hosting devices [188, 189]. The validity of this approach arises from the fact that, in many IoT applications, machine learning outputs (e.g. predictions) need not to be exact, but rather to be in an acceptable range providing the desired quality. Indeed, these approaches need to define quality thresholds that the output must not pass. Integrating DL models with approximate computing can lead to more efficient DL models for resource-constrained devices. Venkataramani *et al.* [188] proposed the extension of approximate computing to neural networks, and converted a neural network to an approximate neural network. In their approach, the authors extend backpropagation to identify the neurons with the least effect on output accuracy. Then, the approximate NN is formed by substituting less important neurons in the original network with their approximate counterparts. Making approximate neurons is performed by an approximate design technique called precision scaling. Instead of using a typical fixed number of bits (16-bit or 32-bit format) to present computations, various number of bits (4 - 10 bits) are used in this technique. After forming the approximate network, the precisions of the inputs and weights of neurons are adjusted to come up with an optimal trade-off between accuracy and energy. There are also other attempts that have reported applying approximate computing with precision scaling on CNNs [189] and DBNs [196]. However, the current practice requires that the process of training the model and converting it to approximate DL takes place on a resource rich platform and then the converted model is deployed on a resource-constrained device.

Accelerators

Designing specific hardware and circuits is another active research direction aiming to optimize the energy efficiency [191] and memory footprint [192] of DL models in IoT devices. The focus of such research works is on the inference time of DL models, since the training procedure of complex models is time and and energy intensive. In [197], several approaches for improving the intelligence of IoT devices are identified including designing accelerators for DNNs, and using Post-CMOS technologies such as spintronics that employs electron spinning mechanism [198]. This latter technology suggests a direction toward the development of hybrid devices that can store data, perform computations and communications within the same material technology.

The works in [190] and [191] have reported an investigation of developing accelerators for DNNs and CNNs, respectively. Beyond the hardware accelerators, the work in [193] proposed the use of a software accelerator for the inference phase of DL models on mobile devices. It employs two resource control algorithms at run-time, one that compresses layers and the other that decomposes deep architecture models across available processors. This software accelerator can be a complementary solution for the

hardware accelerator designs.

Tinymotes

In addition to all prior solutions, developing tiny size processors (micromotes) with strong DL capabilities is on the rise [194] [199]. Designed within the range of one cubic millimeter, such processors can be operated by batteries, consuming only about 300 microwatts while performing on-board analysis and prediction using deep network accelerators. By this technology, many time-critical IoT applications can perform decision-making on the device instead of sending data to high performance computers and waiting for their response. For applications where data security and privacy are the main concerns, this integration of hardware and DL alleviates these concerns to some extent, as no or only limited data needs to be sent to the cloud for analysis. Moons *et al.* [200] also developed a tiny processor for CNNs (total active area of 1.2×2 mm^2) that is power efficient (power consumption is from 25 to 288 mW).

Applications

There are existing mobile apps that employ pre-trained DNNs to perform their analytic and prediction tasks, such as using a CNN to identify garbage in images [81]. However, resource consumption on these apps is still very high. Indeed, [81] reports about 5.6 seconds for returning a prediction response, while consuming 83% of the CPU and 67 MB of memory. Howard *et al.* [201] proposed MobileNets architecture for use in mobile and embedded vision applications. By restructuring a complex model through factorizing a standard convolution layer into a depthwise convolution and a 1×1 convolution, they were able to reach a smaller and computationally efficient models for GoogleNet and VGG16. They also demonstrated several use cases of their model including object detection, image classification, and identifying face attributes.

Amato *et al.* [119] run a CNN on Raspberry Pi boards that were incorporated in smart cameras to find out the empty parking slots. Ravi *et al.* in [202] have reported the development of a fitness app for mobile devices that uses DL for classifying human activities. The DL model is trained on a standard machine and then transferred to the mobile platform for activity recognition. However, the input of the DL model is mixed with several engineered features to improve the accuracy. As the authors describe, the small number of layers in the DNN models dedicated for a resource-constrained device is a potential reason for them achieving a poor performance. In addition, the performance would not be satisfactorily if the training data is not well representing the entire ecosystem.

Nguyen *et al.* [203] proposed a conceptual software-hardware framework to support IoT smart applications. Their framework consists of a cognitive engine and a smart connectivity component. The cognitive engine, which provides cognitive functionality to smart objects, utilizes both DL algorithms

and game-theoretic decision analytics. To be suitable for IoT, these algorithms must be deployed on low-power application-specific processors. The smart connectivity component integrates with cognitive radio transceivers and baseband processors to cater flexible and reliable connections to IoT smart objects.

Lessons Learned

In this section, the need to move toward supporting DL on IoT embedded and resource constrained devices were discussed. The adversary characteristics of IoT devices and DL techniques make this direction more challenging since IoT devices can rarely host DL models even to only perform predictions due to their resource constraints. To tackle these challenges, several methods were introduced in the recent literature including:

- DNN compression
- Approximate computing for DL
- Accelerators
- Tinymotes with DL.

These approaches focus on the inference functionality of available or pre-trained DL models. So, training DL models on resource-constrained and embedded devices is still an open challenge. Shifting the training process to IoT devices is desired for scalable and distributed deployment of IoT devices. For example, having hundreds of smart security cameras deployed in a community for face-based authentication, the training process for each camera can be done on site. Network compression involves identifying unimportant connections and neurons in a DNN through several rounds of training. While this is a promising approach for getting close to real-time analytics on IoT devices, more investigations need to be performed to handle several challenges such as:

- It is not clear whether network compression approaches are suitable for data streaming, especially when the DL model is dynamic and may evolve over time.
- The compression methods for time-series architectures, such as RNN and LSTM, have not been well investigated, and there is a gap to see if the existing compression methods are applicable to these DL architectures.
- There is a need to specify the trade-off between the rate of compression and accuracy of a DNN, as more compression leads to degraded accuracy.

More recently, approximate computing approaches have also been utilized in making DL models simpler and more energy-efficient, in order to operate them on resource constrained devices. Similar to

network compression techniques, these methods also take advantage of insignificant neurons. However, instead of manipulating the network structure, they preserve the structure but change the computation representations through bit-length reduction. For that reason, they seem applicable to a variety of DL architectures and can even cover the dynamic evolution of network models during run-time. Keeping a balance between accuracy and energy usage is their common goal. Nonetheless, more works are needed to find out the superiority of one of these approaches for embedding DL models in IoT devices.

Moreover, we discussed the emergence of special and small form-factor hardware that is designed to efficiently run DL models on embedded and resource constrained devices. These architectures can be utilized in wearable, mobile, and IoT devices, due to their reduced resource demands and their applicability to time-sensitive IoT applications. However, their generality to support any kind of DNN as well as their interoperability and compatibility with existing hardware platforms remain as clear challenges.

Table 7 summarizes the methods and technologies utilized in the recent literature to host DL analytics on IoT devices along with their pros and cons.

We also reviewed some applications that have implemented DL on resource constrained devices. Due to the aforementioned challenges, there are not many well developed applications in this category. However, by resolving these challenges and barriers, we will see the rise of many IoT applications where their core DL model is embedded into the sensors, actuators, and IoT smart objects.

Fog and Cloud-Centric DL for IoT

Cloud computing is considered a promising solution for IoT big data analytics. However, it may not be ideal for IoT data with security, legal/policy restrictions (e.g., data should not be transferred into cloud centers that are hosted outside of national territory), or time constraints. On the other hand, the high-level abstraction of data for some analytics purposes should be acquired by aggregating several sources of IoT data; hence, it is insufficient to deploy analytic solutions on individual IoT nodes in these cases.

Instead of being only on the cloud, the idea of bringing computing and analytics closer to the end-users/devices has been recently proposed under the name of fog computing. Relying on fog-based analytics, we can benefit from the advantages of cloud computing while reducing/avoiding its drawbacks, such as network latency and security risks. It has been shown that, by hosting data analytics on fog computing nodes, the overall performance can be improved due to the avoidance of transmitting large amounts of raw data to distant cloud nodes [204]. It is also possible to perform real-time analytics to some extent since the fog is hosted locally close to the source of data. Smart application gateways

are the core elements in this new fog technology, performing some of the tasks currently done by cloud computing such as data aggregation, classification, integration, and interpretation, thus facilitating the use of IoT local computing resources.

The work in [205] proposed an intelligent IoT gateway that supports mechanisms by which the end users can control the application protocols in order to optimize the performance. The intelligent gateway primarily supports the inter-operation of different types of both IoT and resource-rich devices, causing them to be treated similarly. In the proposed intelligent gateway, a lightweight analytic tool is embedded to increase the performance at the application level. Equipping IoT gateways and edge nodes with efficient DL algorithms can localize many complex analytical tasks that are currently performed on the cloud. Table 8 summarizes several products that have incorporated DL in their intelligent core, and can serve IoT domains in the fog or cloud.

In the following subsection, we review several state-of-the-art enabling technologies that facilitate deep learning on the fog and cloud platforms.

Enabling Technologies and Platforms

Despite introducing DL analytics on fog infrastructure, cloud computing remains the only viable solution for analytics in many IoT applications that cannot be handled by fog computing. For example, complex tasks such as video analysis require large and complex models with a lot of computing resources. Thus, designing scalable and high performance cloud-centric DNN models and algorithms, which can perform analytics on massive IoT data, is still an important research area. Coates *et al.* [206] proposed a large-scale system, based on a cluster of GPU servers, which can perform the training of neural networks with 1 billion parameters on 3 machines in few days. The system can be also scaled to train networks with 11 billion parameters on 16 machines.

Project Adam [207] is another attempt to develop a scalable and efficient DL model. The system is based on distributed DL, where the computation and communication of the whole system are optimized for high scalability and efficiency. The evaluation of this system using a cluster of 120 machines shows that training a large DNN with 2 billion connection achieves two times higher accuracy compared to a baseline system, while using 30 times fewer machines.

Google's Tensor Processing Unit (TPU) [208] is a specialized co-processor for DNNs in Google's data centers. It was designed in 2015 with the aim to accelerate the inference phase of DNNs that are written by TensorFlow framework. From 95% of DNN representatives in their data centers, CNNs only constitute about 5% of the workload, while MLPs and LSTMs cover the other 90%. Performance evaluation showed that TPU outperforms its contemporary GPUs or CPUs on average, by achieving 15 to 30 times faster operation execution, while consuming 30 to 80 times fewer energy per TeraOps/second.

Beyond the infrastructural advancements to host scalable DL models on cloud platforms, there is a need for mechanisms and approaches to make DL models accessible through APIs, in order to be easily integrated into IoT applications. This aspect has not been investigated much, and only a few products are available, such as Amazon's AWS DL AMIs[6], Google cloud ML[7], and IBM Watson[8]. This creates opportunities for cloud providers to offer "*DL models as a service*" as a new sub-category of Software as a Service (SaaS). However, this imposes several challenges for cloud providers, since DL tasks are computationally intensive and may starve other cloud services. Moreover, due to the data thirstiness of DL models, data transfers to the cloud may become a bottleneck. In order to deliver DL analytics on the cloud, Figure 20 presents a general stack for DL models as a service. Different providers may use their customized intelligence stack [209] [210].

Challenges

When DL analytics come to fog nodes, several challenges need to be addressed, including:

- DL service discovery: Fog nodes are densely distributed in geographical regions, and each node may have specific analytics capabilities (e.g., one node runs CNN models for image detection, another node runs RNNs for time-series data prediction, etc.). So, the devices need to identify the sources of appropriate analytic providers through some sort of extended service discovery protocols for DL analytics.

- DL model and task distribution: Partitioning the execution of DL models and tasks among the fog nodes, and optimally distributing of the data stream among the available nodes are critical for time-sensitive applications [211]. Aggregating the final results from the computing nodes and returning the action with the least latency are the other side of the coin.

- Design factors: Since fog computing environments are in their infancy and are expected to evolve, it is worthwhile to investigate how the design factors of the fog environment (e.g., architectures, resource management, etc.) and the deployment of DL models in this environment can impact the quality of analytic services. Alternatively, it would be also interesting to see how far these design factors can be tailored/extended to improve the operation and quality of DL analytics.

- Mobile edge: Through the ubiquity of mobile edge computing environments and their contribution to the IoT analytics, it is important to consider the dynamicity of such environments for designing edge-assisted DL analytics since mobile devices may join and leave the system. Also, the energy management of mobile edge devices should be accurate when analytic tasks are delegated to them.

[6] https://aws.amazon.com/amazon-ai/amis/
[7] https://cloud.google.com/products/machine-learning/
[8] https://www.ibm.com/watson/

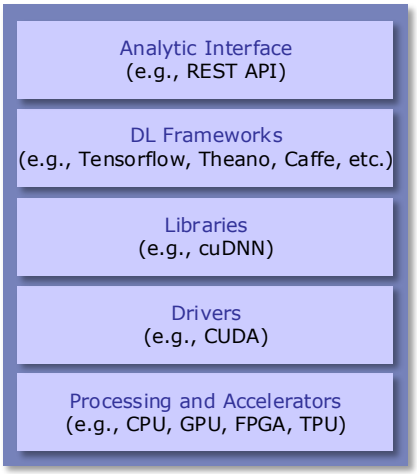


Figure 20: A general stack of DL models as a service in the cloud platforms.

Table 8: Some products that used deep learning and serving IoT domains on the fog or cloud.

Product	**Description**	**Application**	**Platform**
Amazon Alexa	Intelligent personal assistant (IPA)	Smart home	Fog
Microsoft Cortana	IPA	Smart Car, XBox	Fog
Google Assistant	IPA	Smart Car, Smart home	Fog
IBM Watson	Cognitive framework	IoT domains	Cloud

A few attempts reported the integration of DL on fog nodes in the IoT ecosystems. For example, a proof of concept for deploying CNN models on fog nodes for machine health prognosis was proposed by Qaisar *et al.* [212]. In their work, a thorough search among fog nodes is done to find free nodes to delegate analytic tasks to. Also, Li *et al.* [213] proposed a system that leverages the collaboration of mobile and edge devices running CNN models for object recognition.

Lessons Learned

In this section, we highlighted the role of cloud and fog computing and their enabling technologies, platforms and challenges to deliver DL analytics to IoT applications. The great success of cloud computing in support of DL is backed by the advancement and employment of optimized processors for neural networks as well as scalable distributed DL algorithms. Deploying DL models on fog platforms for IoT applications, such as smart homes and smart grids, would draw the attention of the end users due to the ease of accessibility and fast response time. Nevertheless, cloud-based DL analytics would be of great importance for long-term and complex data analytics that exceed the capabilities of fog computing. Some smart city applications, government sector, and nation-wide IoT deployments need to utilize cloud-based

DL infrastructures.

Currently, the integration of DL analytics into IoT applications is limited to RESTful APIs, which are based on the HTTP protocol. While there exist several other application protocols that are extensively used in IoT applications, such as Message Queue Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), Extensible Messaging and Presence Protocol (XMPP), and Advanced Message Queuing Protocol (AMQP), the integration of these protocols with the DL analytic interfaces calls for enhancing their compatibility with the aforementioned protocols to eliminate the need for message conversion proxies, which imposes extra overhead on the analytics response time.

We identified several challenges related to the deployment and usage of DL models in support of analytics on fog nodes. DL service discovery is a necessary requirement due to the dense deployment of fog nodes, which makes brute-force search for available services an inefficient approach. Currently used service discovery protocols in IoT applications, such as multicast DNS (mDNS) or DNS Service Discovery (DNS-SD) [2], need to be extended to support DL service discovery (e.g., declare the type of analytics, DL model, input shape, etc.). Efficient distribution of DL models and tasks, and distribution of data streams on fog nodes and the aggregation of the results are other requirements that need to be addressed.

IoT Challenges for Deep Learning, and Future Directions

In this section we first review several challenges that are important from the machine learning point of view to implement and develop IoT analytics. Then we point out research directions that can fill the existing gaps for IoT analytics based on DL approaches.

Challenges

Lack of Large IoT Dataset

The lack of availability of large real-world datasets for IoT applications is a main hurdle for incorporating DL models in IoT, as more data is needed for DL to achieve more accuracy. Moreover, more data prevents the overfitting of the models. This shortage is a barrier for deployment and acceptance of IoT analytics based on DL since the empirical validation and evaluation of the system should be shown promising in the natural world. Access to the copyrighted datasets or privacy considerations are another burdens that are more common in domains with human data such as healthcare and education. Also, a portfolio of appropriate datasets would be of a lot of help for developers and researchers. A general list of useful datasets has been compiled in Wikipedia [214]. For the convenience of researchers in machine learning applications in IoT, table 9 presents a collection of common datasets suitable to use for DL.

Table 9: Common data sets for deep learning in IoT.

Dataset Name	Domain	Provider	Notes	Address/Link
CGIAR dataset	Agriculture, Climate	CCAFS	High-resolution climate datasets for a variety of fields including agricultural	http://www.ccafs-climate.org/
Educational Process Mining	Education	University of Genova	Recordings of 115 subjects' activities through a logging application while learning with an educational simulator	http://archive.ics.uci.edu/ml/datasets/Educational+Process+Mining+%28EPM%29%3A+A+Learning+Analytics+Data+Set
Commercial Building Energy Dataset	Energy, Smart Building	IIITD	Energy related data set from a commercial building where data is sampled more than once a minute.	http://combed.github.io/
Individual household electric power consumption	Energy, Smart home	EDF R&D, Clamart, France	One-minute sampling rate over a period of almost 4 years	http://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption
AMPds dataset	Energy, Smart home	S. Makonin	AMPds contains electricity, water, and natural gas measurements at one minute intervals for 2 years of monitoring	http://ampds.org/
UK Domestic Appliance-Level Electricity	Energy, Smart Home	Kelly and Knottenbelt	Power demand from five houses. In each house both the whole-house mains power demand as well as power demand from individual appliances are recorded.	http://www.doc.ic.ac.uk/~dk3810/data/
PhysioBank databases	Healthcare	PhysioNet	Archive of over 80 physiological datasets.	https://physionet.org/physiobank/database/
Saarbruecken Voice Database	Healthcare	Universität des Saarlandes	A collection of voice recordings from more than 2000 persons for pathological voice detection.	http://www.stimmdatenbank.coli.uni-saarland.de/help_en.php4
T-LESS	Industry	CMP at Czech Technical University	An RGB-D dataset and evaluation methodology for detection and 6D pose estimation of texture-less objects	http://cmp.felk.cvut.cz/t-less/
CityPulse Dataset Collection	Smart City	CityPulse EU FP7 project	Road Traffic Data, Pollution Data, Weather, Parking	http://iot.ee.surrey.ac.uk:8080/datasets.html
Open Data Institute - node Trento	Smart City	Telecom Italia	Weather, Air quality, Electricity, Telecommunication	http://theodi.fbk.eu/openbigdata/
Málaga datasets	Smart City	City of Malaga	A broad range of categories such as energy, ITS, weather, Industry, Sport, etc.	http://datosabiertos.malaga.eu/dataset
Gas sensors for home activity monitoring	Smart home	Univ. of California San Diego	Recordings of 8 gas sensors under three conditions including background, wine and banana presentations.	http://archive.ics.uci.edu/ml/datasets/Gas+sensors+for+home+activity+monitoring
CASAS datasets for activities of daily living	Smart home	Washington State University	Several public datasets related to Activities of Daily Living (ADL) performance in a two-story home, an apartment, and an office settings.	http://ailab.wsu.edu/casas/datasets.html
ARAS Human Activity Dataset	Smart home	Bogazici University	Human activity recognition datasets collected from two real houses with multiple residents during two months.	https://www.cmpe.boun.edu.tr/aras/
MERLSense Data	Smart home, building	Mitsubishi Electric Research Labs	Motion sensor data of residual traces from a network of over 200 sensors for two years, containing over 50 million records.	http://www.merl.com/wmd

TABLE 9 – Continued

Dataset Name	Domain	Provider	Notes	Address/Link
SportVU	Sport	Stats LLC	Video of basketball and soccer games captured from 6 cameras.	http://go.stats.com/sportvu
RealDisp	Sport	O. Banos	Includes a wide range of physical activities (warm up, cool down and fitness exercises).	http://orestibanos.com/datasets.htm
Taxi Service Trajectory	Transportation	Prediction Challenge, ECML PKDD 2015	Trajectories performed by all the 442 taxis running in the city of Porto, in Portugal.	http://www.geolink.pt/ecmlpkdd2015-challenge/dataset.html
GeoLife GPS Trajectories	Transportation	Microsoft	A GPS trajectory by a sequence of time-stamped points	https://www.microsoft.com/en-us/download/details.aspx?id=52367
T-Drive trajectory data	Transportation	Microsoft	Contains a one-week trajectories of 10,357 taxis	https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/
Chicago Bus Traces data	Transportation	M. Doering	Bus traces from the Chicago Transport Authority for 18 days with a rate between 20 and 40 seconds.	http://www.ibr.cs.tu-bs.de/users/mdoering/bustraces/
Uber trip data	Transportation	FiveThirty-Eight	About 20 million Uber pickups in New York City during 12 months.	https://github.com/fivethirtyeight/uber-tlc-foil-response
Traffic Sign Recognition	Transportation	K. Lim	Three datasets: Korean daytime, Korean nighttime, and German daytime traffic signs based on Vienna traffic rules.	https://figshare.com/articles/Traffic_Sign_Recognition_Testsets/4597795
DDD17	Transportation	J. Binas	End-To-End DAVIS Driving Dataset.	http://sensors.ini.uzh.ch/databases.html

Preprocessing

Preparing raw data in an appropriate representation to be fed in DL models is another challenge for IoT applications. Most DL approaches need some sort of preprocessing to yield good results. For example, image processing techniques by CNNs work better when the input data at the pixel level are normalized, scaled into a specific range, or transformed into a standard representation [37][60]. For IoT applications, preprocessing is more complex since the system deals with data from different sources that may have various formats and distributions while showing missing data.

Secure and Privacy Preserving Deep Learning

Ensuring data security and privacy is a main concern in many IoT applications, since IoT big data will be transferred through the Internet for analytics, and can be thus observed around the world. While anonymization is used in many applications, these techniques can be hacked and re-identified as anonymized data. Moreover, DL training models are also subject to malicious attacks, such as False Data Injection or adversarial sample inputs, by which many functional or non-functional (e.g., availability, reliability, validity, trustworthiness, etc.) requirements of the IoT systems may be in jeopardy. Indeed, DL models learn the features from the raw data, and can therefore learn from any invalid data feed to it. In this case, DL models must be enhanced with some mechanism to discover abnormal or invalid data.

A data monitoring DL model accompanying the main model should work in such scenarios. Papernot *et al.* [215] have investigated the vulnerability of DNNs in adversarial settings where an adversary tries to provide some inputs that lead to an incorrect output classification and hence corrupting the integrity of the classification. Developing further techniques to defend and prevent the effect of this sort of attacks on DL models is necessary for reliable IoT applications.

Challenges of 6V's

Despite the recent advancement in DL for big data, there are still significant challenges that need to be addressed to mature this technology. Each characteristic of IoT big data imposes a challenge for DL techniques. In the following we highlight these challenges.

The massive volume of data poses a great challenge for DL, especially for time and structure complexity. The voluminous number of input data, their broad number of attributes, and their high degree of classification result in a very complex DL model and affect running time performance. Running DL on distributed frameworks or clusters of CPUs with parallel processing is a viable solution that has been developed [8]. The high volume of IoT big data also brings another challenge, namely the noisy and unlabeled data. Even though DL is very good at tolerating noisy data and learning from unlabeled data, it is not clear to what extent DL models can be accurate in the presence of such abnormal data.

The variety of IoT data formats that come from various sources pops up the challenge of managing conflicts between different data sources. In case of no conflict in data sources, DL has the ability to effectively work on heterogeneous data.

The high velocity of IoT big data, i.e., the high rate of data generation, also brings the challenge of high speed processing and analysis of data. Online learning is a solution for high velocity and has been proposed for DNNs. However, more research is needed to augment DL with online learning and sequential learning techniques.

The veracity of IoT big data also presents challenges for DL analytics. The IoT big data analytics will not be useful if the input data is not coming from a trustworthy source. Data validation and trustworthiness should be checked at each level of big data analytics, especially when we are dealing with online streams of input data to an analytic engine [216].

Moreover, the variability of IoT big data (variation in the data flow rates) rises challenges for online analytics. In case of immense streams of data, DL techniques, and in particular the online ones, handle them. Data sampling techniques would be beneficial in these scenarios.

Finally, a main challenge for business managers to adopt big data is that it is not clear for them how to use big data analytics to get value out of it and improve their business[217]. Beyond that, the analytic engine may produce abstractions that are not important for the stakeholders, or are not clear enough for

them.

Deep Learning for IoT Devices

Developing DL on IoT devices poses a new challenge for IoT device designers, to consider the requirements of handling DNNs in resource-constrained devices. These requirements are expected to grow as the datasets sizes are growing every day, and new algorithms arise to be part of the solutions for DL in IoT.

Deep Learning Limitations

Despite showing impressive results in many applications, DL models still have several limitations. Nguyen *et al.* [218] reported about the false confidence of DDN for predicting images that are unrecognizable by humans. By producing fooling examples that are totally unrecognizable by humans, the DNN classifies them as familiar objects.

The other limitation is the focus of DL models on classification, while many IoT applications (e.g., electricity load forecasting, temperature forecasting) need a kind of regression at their analytics core. Few works tried to enrich DNNs with regression capabilities, such as the work in [219] proposing the ensemble of DBN and Support Vector Regression (SVR) for regression tasks. However, more investigation is required to clear many aspects of regression with DL.

Future Directions

IoT Mobile Data

One remarkable part of IoT data comes from mobile devices. Investigating efficient ways to utilize mobile big data in conjunction with DL approaches is a way to come up with better services for IoT domains, especially in smart city scenarios. In [220], the capabilities of DL models in mobile big data analytics were investigated using a distributed learning framework that executes an iterative MapReduce task on several parallel Spark workers.

Integrating Contextual Information

The environment's situation cannot be understood by the IoT sensor data alone. Therefore, IoT data needs to be fused with other sources of data, namely context information that complement the understanding of the environment [13]. This integration can also help for fast data analytics and quick reasoning due to the bounded search space for the reasoning engine. For example, a smart camera with capability of face pose recognition can perform its job in various contexts such as security gates

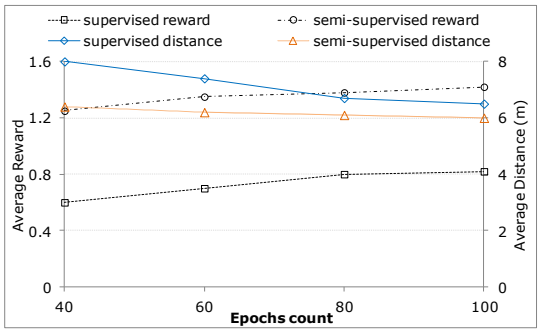


Figure 21: Deep reinforcement learning with only labeled data (supervised) vs. with labeled and unlabeled data (semisupervised).

in smart homes or government buildings, or in smart cars for driving assistance. In all these situations, complementary contextual information (e.g., time within the day, daily habits, etc.) helps the system to reason about the best action that can be done based on the detected pose of the person.

Online Resource Provisioning for IoT Analytics

The deployment of fast DL based data analytics on the fog and cloud would require online provisioning of fog or cloud resources to host the stream of data. Due to the streaming nature of IoT data, knowing the volume of data sequence in advance is not feasible. In this regard, we need a new class of algorithms that work based on the current stream of data and do not rely on the prior knowledge of the data stream. A DL mechanism and an online auctioning algorithm are proposed in [88] and [221], respectively, to support online provisioning of fog and cloud resources for IoT applications.

Semi-supervised Analytic Frameworks

Most of the analytic algorithms are supervised, thus needing a large amount of training labeled data that is either not available or comes at a great expense to prepare. Based on IDC's report [89], it is estimated that by 2012 only about 3% of all data in the digital universe has been annotated, which implies the poor source of training datasets for DL. A combination of advanced machine learning algorithms designed for semi-supervised settings fits well for smart cities systems, where a small training dataset can be used while the learning agent improves its accuracy using a large amount of unlabeled data [6]. Figure 21 illustrates the role of semi-supervised learning in improving the output accuracy for deep reinforcement learning [61] in indoor localization experiments. In their experiments, only 15% of data was labeled but the results were strengthened by utilizing unlabeled data in the algorithm.

Dependable and Reliable IoT Analytics

As we rely more on CPS and IoT in large scales, the need for mechanisms to ensure the safety of the system against malicious attacks as well as failures become more crucial [222]. DL approaches can be applied in these directions by analyzing the huge amount of log traces of CPS and IoT systems, in order to identify and predict weak points of the system where attacks may occur or the functionality is defected. This will help the system to prevent or recover from faults and consequently increase the level of dependability of CPS and IoT systems.

Self-organizing Communication Networks

With a huge number of IoT devices, the configuration and maintenance of their underlying physical M2M communications and networking become harder. Although the large body of network nodes and their relation is a challenge for traditional machine learning approaches, it opens the opportunity for DL architectures to prove their competency in this area by providing a range of self-services such as self-configuration, self-optimization, self-healing, and self-load balancing. Valente *et al.* [223] have provided a survey of traditional machine learning approaches for self-organizing cellular networks.

Emerging IoT Applications

Unmanned aerial vehicles: The usage of Unmanned aerial vehicles (UAVs) is a promising application that can improve service delivery in hard-reaching regions or in critical situations. UAVs have been also used for many image analysis in real-time such as surveillance tasks, search-and-rescue operations, and infrastructure inspection [224]. These devices face several challenges for their adoption, including routing, energy saving, avoiding private regions, and obstacle avoidance [225] etc. DL can be of great impact in this domain for the prediction and decision-making of tasks to get the best out of UAVs. Moreover, UAVs can be seen as on-the-fly analytics platforms that potentially can provide temporarily fog computing analytic services as well as distributed analytics.

Virtual/Augmented Reality: Virtual/augmented reality is another application area that can benefit from both IoT and DL. The latter can be used in this field to provide services such as object tracking [226], activity recognition, image classification, and object recognition [227] to name a few. Augmented reality can greatly affect several domains such as education, museums, smart connected cars, etc.

Mobile Robotics: Mobile robots are being used in many commercial and industrial settings for moving materials or performing tasks in hazardous environments. There are many research efforts that have benefited from DL to develop an intelligent control software for mobile robots [228] [229]. Being able to perceive the environment through different kinds of sensors, such as LIDARs and cameras, have made

them a top topic to assess the performance of CNN techniques for a variety of vision-based tasks. A strict requirement for mobile robots is that DL models should be able to provide real-time responsiveness.

Conclusion

DL and IoT have drawn the attention of researchers and commercial verticals in recent years, as these two technology trends have proven to make a positive effect on our lives, cities, and world. IoT and DL constitute a chain of data producer-consumer, in which IoT generates raw data that is analyzed by DL models and DL models produce high-level abstraction and insight that is fed to the IoT systems for fine-tuning and improvement of services.

In this survey, we reviewed the characteristics of IoT data and its challenges for DL methods. In specific, we highlighted IoT fast and streaming data as well as IoT big data as the two main categories of IoT data generation and their requirements for analytics. We also presented several main architectures of DL that is used in the context of IoT applications followed by several open source frameworks for development of DL architectures. Reviewing different applications in various sectors of IoT that have utilized DL was another part of this survey in which we identified five foundational services along with eleven application domains. By distinguishing foundational services, as well as IoT vertical applications, and reviewing their DL approaches and use cases, the authors provided a basis for other researchers to understand the principle components of IoT smart services and apply the relevant techniques to their problems. The new paradigm of implementing DL on IoT devices was surveyed and several approaches to achieve it were introduced. DL based on fog and cloud infrastructures to support IoT applications was another part of this survey. We also identified the challenges and future research direction in the path of DL for IoT applications.

Chapter 3

Semi-supervised Deep Reinforcement Learning in Support of IoT and Smart City Services

Introduction

The rapid development of Internet of Things (IoT) technologies motivated researchers and developers to think about new kinds of smart services that extract knowledge from IoT generated data. The scarcity of labeled data is a main issue for developing such solutions especially for IoT applications where a large number of sensors participate in generating data without being able to obtain class labels corresponding to the collected data. Smart cities as a prominent application area of the IoT should provide a range of high-quality smart services to meet the citizen's needs [230]. Smart buildings are one of the main building blocks of smart cities as citizens spend a significant part of their time indoors. People nowadays spend over 87% of their daily lives indoors [231] for work, shopping, education, etc. Therefore, having a smart environment that provides services to meet the needs to its inhabitants is a valuable asset for organizations. Such services facilitate the development of smart cities. Location-aware services in indoor environments play a significant role in this era. Examples of applications of such services are smart home management [232], delivering cultural contents in museums [233], location-based authentication and access control [234], location-aware marketing and advertisement [235][236], and wayfinding and navigation in smart campuses [237]. Moreover, locating users in indoor environments is very important for smart buildings because it serves as the link that enables the users to interact with other IoT services [238].

Deep learning is a powerful machine learning approach that provides function approximation, classification, and prediction capabilities. Reinforcement learning is another class of machine learning approaches for optimal control and decision making processes where a software agent learns an optimal policy of actions over the set of states in an environment. In the applications where the number of states is very large, a deep learning model can be used to approximate the action values (i.e., how good an action is in a given state). Systems that combine deep and reinforcement learning are in their initial phases but already produced competitive results in some application areas (e.g., video games). Moreover, learning approaches with no or little supervision are expected to get more momentum in the future [11] mimicking the natural learning processes of humans and animals.

IoT applications can benefit from the decision process for learning purposes. For example, in the case of location-aware services, location estimation can be seen as a decision process in which a software agent determines the exact or closest point to a specific target. In this regard, reinforcement learning [239] can be exploited to formulate and solve the problem. In a reinforcement learning solution, a software agent interacts with the environment and changes the state of the environment by performing some actions. Depending on the performed action, the environment sends a reward to the agent. The agent tries to maximize its rewards over time by choosing those actions that result in higher rewards. A new variation of reinforcement learning, deep reinforcement learning recently was demonstrated by Google to achieve high accuracy in the Atari games [60] and is a suitable candidate for the learning process in the IoT applications.

In this research, we propose a semi-supervised deep reinforcement learning model to benefit from the large number of unlabeled data that are generated in IoT applications.

The points that motivate us for this study are:

- In the world of IoT where sensors generate a lot of data that cannot be labeled manually for training purposes, semi-supervised approaches are valuable approaches. Moreover, building a deep reinforcement learning framework that works in a semi-supervised manner can serve many IoT applications.

- Games demonstrated significant improvements using DRL [60]. IoT applications also can be seen as a game where the goal is to estimate the correct classification of a given input and hence can benefit from a DRL approach.

- The learning process for the scale of smart cities requires many efforts including data gathering, analysis and classification. The strength of deep learning models stems from the latest advancements in computational and data storage capabilities. Such models can be utilized to develop scalable and efficient learning solutions for smart cities from crowd-sensed big data.

- Smart city applications can be trained in a lab and deployed in a real environment without losing performance. For example, a self-driving car needs to learn how to perform in a variety of conditions (e.g., approaching pedestrians, handling traffic signs, etc.) which can be learned in a few test drives. But it is impossible to account for all the scenarios that might happen in a given city.

Also, our study is motivated by specific observations regarding the localization problem including:

- While WiFi fingerprinting has been studied widely in the past decade for indoor positioning and the accuracy is in the range of 10 m, BLE is in its infancy for indoor localization and has yielded more fine-grained results [240].

- There are many practical applications that need an efficient mechanism for positioning in small scale environments such as robotic soccer games to locate the position of the ball, or navigating robots in a building. Our proposed approach can be extended and used in such scenarios aiming to enhance micro-localization accuracy in support of smart environments [238].

The contributions of this paper are as follows:

- We propose a semi-supervised deep reinforcement learning framework based on deep generative models and reinforcement learning that combines the strengths of deep neural networks and statistical modeling of data density in a reinforcement learning paradigm. To the best of our knowledge, this work is the first attempt to address semi-supervised learning through deep reinforcement learning.

- We leverage both labeled and unlabeled data in our model. Since unlabeled data are more prevalent, this is a key feature for IoT applications where IoT sensors generate large volumes of data while they cannot be labeled easily. Therefore, our approach helps to alleviate having a lot of labeled data. In addition, the performance of deep reinforcement learning is enhanced by using the proposed semi-supervised approach.

The idea of extending reinforcement learning algorithms to semi-supervised reinforcement learning has not been studied well so far. There are some suggestions that explain the possibility of semi-supervised reinforcement learning by having unlabeled episodes in which the agent does not receive its rewards from the environment [241] [242]. But there is no implementation of such extensions so far. Our proposed semi-supervised deep reinforcement learning, however, follows a different approach where we incorporate a variational autoencoder [44] in our framework as the semi-supervised module to infer the classification of unlabeled data and incorporate this information along with the labeled data to optimize its discriminating boundaries.

To apply the proposed model on a smart city scenario, we chose to perform the experiments on smart buildings which play a significant role in smart cities. Our experimental results assert the efficiency of the proposed semi-supervised DRL model compared to the supervised DRL model. Specifically, the results have been improved by 23% for a small number of training epochs. Also, considering the average performance of both models in terms of received rewards, the semi-supervised model outperforms the supervised model by obtaining twice as many rewards.

The rest of this paper is organized as follows. Section 3 organizes the recent related works into two parts: one that reviews attempts that utilize DRL models and the other that deals with the indoor localization as a case study. Section 3 presents related background and then introduces the details of the proposed approach. Section 5 presents a use-case study in which the proposed model is used for indoor localization systems using iBeacon signals. Experimental results are presented in Section 3 followed by concluding remarks in Section 4.

Related Work

In the following sub-sections, we first review some recent research efforts that utilize deep reinforcement learning. Then, we address the latest research efforts that address the indoor localization problem from machine learning perspective.

Deep Reinforcement Learning

Deep reinforcement learning has been proposed in recent years [60] and is gaining attention to be applied in various application domains. In the following paragraphs, we review some of the latest research efforts that utilize DRL in different application areas.

Nemati *et al.* [243] utilized a deep reinforcement learning algorithm to learn actionable policies for administering an optimal dose of medicines like heparin for individuals. They used a sample dataset of dosage trials and their outcomes from a bunch of electronic medical records. In their model, they used a discriminative Hidden Markov Model (HMM) for state estimation and a Q-network with two layers of neurons. The medication dosage agent tries to learn the optimal policy by maximizing its total reward which is the overall fraction of time when patients are in their therapeutic activated Partial Thromboplastin Time (aPTT) range.

Deep reinforcement learning has been also applied for vehicle image classification as reported in [244]. In that work, the authors propose a Convolutional Neural Network (CNN) model combined with a reinforcement learning module to guide where to look in the image for the key parts of a car. The information entropy of the classification probability of a focused image that is produced by their CNN

model is considered as the reward for the reinforcement learning agent to learn to identify the next visual attention area in the image. The work in [245] also reports on object localization in images by focusing attention on candidate regions using a deep reinforcement learning approach. In [246], a visual navigation application is presented that uses a variation of deep reinforcement learning by which robots can navigate in a space toward a visual target.

Li *et al.* [247] also developed a deep reinforcement learning approach for traffic signal timing aiming to have a better signal timing plan. In their model which consists of a four-layer stacked autoencoder neural network to estimate the Q-function, they use the queuing lengths of eight incoming lanes to an intersection as the state of the system at each time. They also define two actions: stay in the current traffic lane, or change the lane to allow other traffic to go through the intersection. The absolute value of the difference between the length of opposite lanes serves as the reward function. Their results show that their proposed model outperformed other conventional reinforcement learning approaches.

Resource management is another task that can use DRL as its underlying mechanism. In their report, Mao *et al.* [248] formulate the problem of job scheduling with multiple resource demands as a deep reinforcement learning process. In their approach, the objective is to minimize the average job slowdown. The reward function was defined based on the reciprocal duration of the job in order to guide the agent toward the objective.

Another application in which deep reinforcement learning played a key role is natural language understanding for text-based games [249][250]. For example, Narasimhan *et al.* [249] used Long Short-Term Memory (LSTM) networks to train the agent with useful representations of text descriptions and a Deep Q-Network (DQN) to approximate Q-functions. Other disciplines like energy management also have incorporated DRL to improve energy utilization [251].

Review of Indoor Localization

To the best of our knowledge, there are no prior research efforts that utilized DRL for localization. In the following paragraphs, we review the different machine learning approaches that were utilized in the recent research literature to provide indoor localization services.

Among the approaches for deploying location-based services, Relative Signal Strength (RSS) fingerprinting is one of the most promising approaches. However, there are some challenges that need to be considered in the deployment of such approach including fingerprint annotation and device diversity [252]. The use of fingerprint-based approaches to identify an indoor position has been studied well in the past decade. Researchers have studied different machine learning approaches in this context including: SVM, KNN, Bayesian-based filtering, transfer learning, and neural networks [94].

It has been shown that for coarse-grained positioning applications based on Bluetooth Low Energy

(BLE) RSS fingerprinting, the estimation to decide if a device is inside a room or not yields pretty reliable results [253].

The authors in [254] report their experimental indoor positioning results based on BLE RSS readings. They study the accuracy of three methods including Least Square Estimation (LSE), Three-border positioning, and Centroid positioning. In their testing area, which is a 6×8 square meters classroom with four BLE stations at the corners, the LSE algorithm shows more accurate positions compared to the two other methods. However, the overall accuracy of these three algorithms is satisfactory.

Museums are good environments for using BLE to provide location-awareness since usually the building and its contents do not allow changes due to preservation policies. The authors in [233] developed a system to make interactive cultural displays in a museum with BLE beacons combined with an image recognition wearable device. The wearable device performs localization by receiving BLE signals from the beacons to identify the room in which it is located. It also identifies artworks by an image processing service. The combination of the closest beacon identifier and the artwork identifier are fed to a processing center to retrieve the appropriate cultural content.

In [92], the authors present a system called DeepFi that utilizes a deep learning method over fingerprinting data to locate indoor positions based on channel state information (CSI). As many other fingerprinting approaches, their system consists of offline training and online localization phases. In the off-line training phase, they exploit deep learning to train all the weights as fingerprints based on the previously stored CSI. Their evaluations in a living room and laboratory settings show that the use of deep learning result in improved localization accuracy of 20%. While their use of CSI approach is limited to WiFi networks, not all available Network Interface Cards (NICs) in the market support obtaining measurements from the different network channels.

In [93], deep learning joined with semi-supervised learning as well as extreme learning machine are applied to unlabeled data to study the performance of feature extraction and classification phases of indoor localization. In their study, deep learning network and semi-supervised learning generate high level abstract features and more accurate classification while extreme learning machine can speed up the learning process. Their test setting is a 10×15 square meters area. Their results show that deep learning can improve the accuracy of fingerprinting by at least 1.3% for the same training dataset compared to a shallow learning method. Also increasing unlabeled data has a positive effect on the accuracy compared to shallow feature methods. Compared to other deep learning methods including stacked autoencoder, deep belief network, and multi-layer extreme learning machine, their approach improves the accuracy at least 10%.

In another study [94], the authors propose a WiFi localization approach using deep neural networks (DNN). In their system, a four-layer deep learning model is used to extract features from WiFi RSS

data. In their approach, the authors use Stacked Denoising Autoencoder and Backpropagation for the training steps. In the online positioning phase, the estimated position based on DNN is refined by an HMM component. Their experiments assert that the number of hidden layers and neurons have a direct effect on the localization accuracy. Increasing the layers leads to better results, but at some point when the network is made deeper, the results start degrading. Their result shows that when using three hidden layers with 200 neurons for each layer, the model achieves the best accuracy.

Ding *et al.* [255] also used an Artificial neural network (ANN) for WiFi fingerprinting localization. They proposed a localization approach that uses ANNs in conjunction with a clustering method based on affinity propagation. By affinity propagation clustering, the training of the ANN model has been faster and the memory overhead has been lowered. They also reported improved positioning accuracy compared to other baseline methods.

In [256], a deep belief network (DBN) is used for a localization approach that is based on fingerprinting of ultra-wideband signals in an indoor environment. Parameters of channel impulse response are used to get a dataset of fingerprints. Compared to other methods, the author demonstrated that DBN can improve the localization accuracy.

The work in [257] also reports using a deep learning model in conjunction with a regression model to automatically learn discriminative features from the received wireless signal. The authors also use a softmax regression algorithm to perform device-free localization and activity recognition. They report that their proposed method can improve the localization accuracy by 10% compared to other methods.

Semi-supervised algorithms have also been widely applied to the localization problem to utilize the unlabeled data for the prediction of an unknown location. For example, in [258] the authors presented a semi-supervised algorithm based on the manifold assumption to obtain tagged fingerprints out of unlabeled data using a small amount of labeled data. They map the high-dimensional space of fingerprints into a two-dimensional space and achieved an average error of 2 meters.

Our work in this paper presents several significant differences compared to the aforementioned approaches. First, related research studies in deep reinforcement learning do not exploit the statistical information of unlabeled data, while our proposed DRL approach is extended to be semi-supervised and utilizes both labeled and unlabeled data. Second, these approaches provide an application-dependent solution, while our work is a general framework that can work for a variety of IoT applications. Third, for localization systems, all aforementioned deep learning solutions rely on WiFi fingerprinting, while the context of BLE fingerprinting has not been studied in conjunction with deep learning or reinforcement learning approaches.

Background and Proposed Approach

In the following sub-sections, we first describe the fundamentals of variational autoencoders. Then we describe our proposed semi-supervised DRL model by adopting a variational autoencoder in a deep reinforcement learning model. We develop the theoretical foundation of our method based on [44] and [60].

Semi-Supervised Learning Using VAE

Semi-supervised learning methods aim to improve the generalization of supervised learning tasks using unlabeled data [259]. They usually use a small set of annotated data along with a larger number of unlabeled data to train the model. In a semi-supervised setting, we have two datasets; one is labeled and the other is unlabeled. The labeled dataset is denoted by $X_l = (x_1, \ldots, x_l)$ for which labels $Y_l = (y_1, \ldots, y_l)$ are provided. The other set is $X_u = (x_{l+1}, \ldots, x_{l+u})$ with unknown labels. Semi-supervised algorithms are built based on at least one of the following three assumptions [260]: The smoothness assumption, states that if two points x_1 and x_2 are close to each other, then their corresponding labels are very likely to be close to each other. The cluster assumption implies how to identify discrete clusters. It states that if two points are in the same cluster, it is more probable that they have the same class label. The manifold assumption points out that high dimensional data can be mapped to a lower dimensional one (i.e., the principle of parsimony) such that the supervised algorithm still approximates the true class of a data point.

For the semi-supervised part of our proposed model, we adopt the deep generative model based on variational autoencoders (VAE) [44]. This model has been used for semi-supervised tasks such as the recognition of handwritten digits, house number classification and motion prediction [261] with impressive results. Figure 22 shows the structure of a typical VAE model. For each data point x_i there is a vector of corresponding latent variables denoted by z_i. The distribution of labeled data is represented by $\tilde{p_l}(x, y)$, while unlabeled data are represented by $\tilde{p_u}(x)$.

The latent feature discriminative model (M1) is created based on:

$$p(z) = \mathcal{N}(z|0, I); \quad p_\theta(x|z) = f(x; z, \theta), \tag{1}$$

in which $p(z)$ is Gaussian distributed with mean vector 0 and variances presented in an identity matrix I. The function $f(x; z, \theta)$ is a nonlinear likelihood function with parameter θ for latent variable z based on a deep neural network.

The generative semi-supervised model for generating data using a latent class variable y, in addition to a latent variable z is (M2):

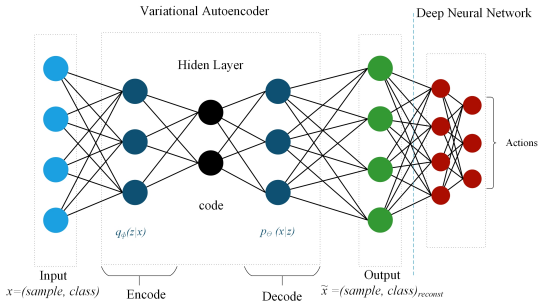


Figure 22: The high-level concept of a variational autoencoder adopted for deep reinforcement learning.

$$p(y) = Cat(y|\pi); \quad p(z) = \mathcal{N}(z|0, I);$$
$$p_\theta(x|y, z) = f(x; y, z, \theta), \tag{2}$$

where $Cat(y|\pi)$ represents a categorical distribution or in general a multinomial distribution with a vector of probabilities π whose elements sum up to 1. In the dataset, if no label is available, the unknown labels y are considered as latent variables in addition to z.

The models have two lower bound objectives. To describe the model objectives, a fixed form distribution $q_\phi(z|x)$ is introduced with parameter ϕ that helps us to estimate the posterior distribution $p(z|x)$. For all latent variables in the models, an inference deep neural network is introduced to generate a distribution of the form $q_\phi(.)$. For M1, a Gaussian inference network $q_\phi(z|x)$ is used for latent variable z:

$$q_\phi(z|x) = \mathcal{N}(z|\mu_\phi(x), diag(\sigma^2_\phi(x))), \tag{3}$$

in which $\mu_\phi(x)$ is the vector of means, $\sigma_\phi(x)$ is the vector of standard deviations, and $diag$ creates a diagonal matrix. For M2, an inference network is used for latent variables z and y using Gaussian and multinomial distributions, respectively:

$$q_\phi(z|y, x) = \mathcal{N}(z|\mu_\phi(y, x), diag(\sigma^2_\phi(x)));$$
$$q_\phi(y|x) = Cat(y|\pi_\phi(x)), \tag{4}$$

where $\pi_\phi(x)$ is a vector of probabilities.

The lower bound for M1 is:

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - KL[q_\phi(z|x)||p_\theta(z)]$$
$$= -\mathcal{J}(x), \tag{5}$$

in which KL is the Kullback-Leibler divergence function between the encoding and prior distribution

and can be obtained as $KL(q_\phi \| p_\theta) = \sum_i q_{\phi_i} \log \frac{q_{\phi_i}}{p_{\theta_i}}$.

For the model M2, two cases should be considered. The first one deals with labeled data:

$$\begin{aligned} \log p_\theta(x,y) \geq \mathbb{E}_{q_\phi(z|x,y)}[\log p_\theta(x|y,z)] + \log p_\theta(y) \\ + \log p_\theta(z) - \log q_\phi(z|x,y)] = -\mathcal{L}(x,y). \end{aligned} \tag{6}$$

When dealing with unlabeled data, y is treated as a latent variable and the resulting lower bound is:

$$\begin{aligned} \log p_\theta(x) \geq \mathbb{E}_{q_\phi(y,z|x)}[\log p_\theta(x|y,z)] + \log p_\theta(y) \\ + \log p_\theta(z) - \log q_\phi(y,z|x)] = -\mathcal{U}(x). \end{aligned} \tag{7}$$

Then the whole dataset has its bound of marginal likelihood as:

$$\mathcal{J} = \sum_{(x,y)\sim\tilde{p}_l} \mathcal{L}(x,y) + \sum_{x\sim\tilde{p}_u} \mathcal{U}(x) \tag{8}$$

By adding a classification loss to the above function, the optimized objective function becomes:

$$\mathcal{J}^\alpha = \mathcal{J} + \alpha \; . \; \mathbb{E}_{\tilde{p}_l(x,y)}[-\log q_\phi(y|x)], \tag{9}$$

where α adjusts the contributions of the generative and discriminative models in the learning process. During the training process for both models M1 and M2, the stochastic gradient of $\mathcal{J}$ is computed at each minibatch to be used for updating the generative parameters θ and the variational parameters ϕ.

Semi-Supervised Deep Reinforcement Learning

To adopt a deep reinforcement learning approach, we need to define the following elements for a Markov Decision Process (MDP). The goal of the MDP in a reinforcement learning problem is to maximize the earned rewards.

Environment: The environment is the territory that the learning agent interacts with.

Agent: The agent observes the environment, receives sensory data and performs a valid action. It then receives a reward for its action. Through training, the agent learns to maximize its rewards.

States: The finite set of states that the environment can assume. Each action of the agent puts the environment in a new state.

Actions: The finite set of available actions that the agent can perform causing a transition from state s_t at time t to state s_{t+1} at time $t+1$.

Reward function: This function is the immediate feedback for performing an action. The reward function can be defined such that it reflects the closeness of the current state to the true class label; i.e.,

$r(s_t, a_t, s_{t+1}, y) = closeness(s_{t+1}, y)$. Depending on the problem, different distance measurements can be applied. The point is that we need to devise larger positive rewards for more compelling results and negative rewards for distracting ones.

State transition distribution: is the probability that action a in state s at time t will lead to state s' at time $t+1$: $P_a(s, s') = Pr(s'|s, a)$.

Having these components, the main problem is to find a policy π (where $\pi = a_t$) that maximizes the rewards: $R_t = \sum_{t=0} \gamma^t r_t$, in which γ is a discount factor $0 \leq \gamma < 1$.

In the deep Q-Network approach, we need a deep neural network that approximates the optimal action-value function (Q) [262]:

$$Q^*(s, a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s, a_t = a, \pi] \tag{10}$$

This function finds the maximum sum of rewards r_t discounted by γ at each time-step t, achievable by a behavior policy $\pi = P(a|s)$, after making an observation (s) and taking an action (a). We can convert this equation to a simpler approximation function using *Bellman equation*. For a sequence of states s' and for all possible actions a', if the optimal value $Q^*(s', a')$ is known, then we can obtain the optimal strategy by selecting the action a' that maximizes the expected value of $r + \gamma Q^*(s', a')$:

$$Q^*(s, a) = \mathbb{E}_{s'} \left[r + \gamma \max_{a'} Q^*(s', a') | s, a \right] \tag{11}$$

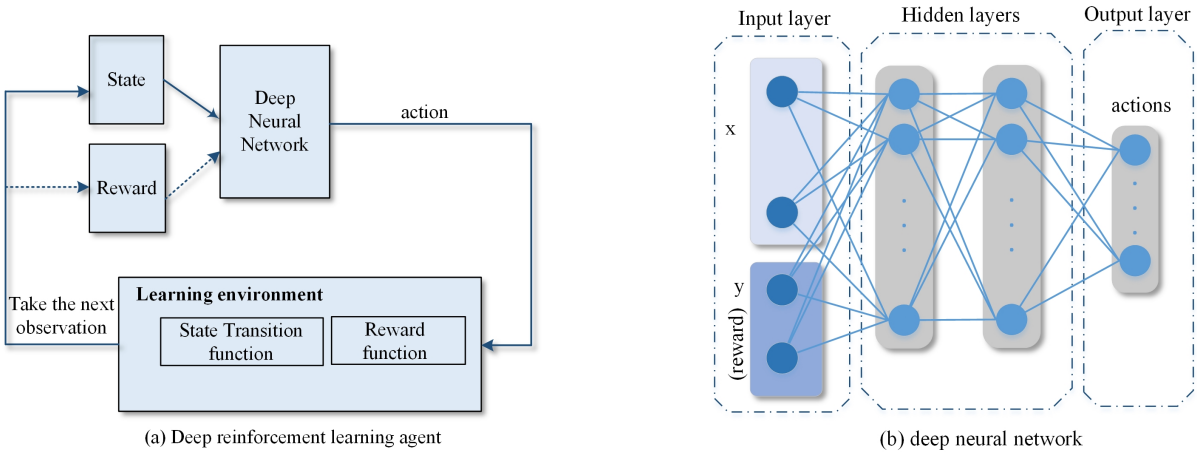
To estimate the optimal action-value function, we use a non-linear function approximator (i.e., a neural network with weights θ) such that $Q(s, a; \theta) \approx Q^*(s, a)$. The network can be trained by minimizing the loss functions $L_i(\theta_i)$ that is updated at each time-step.

We perform experience replay, so we keep track of the agent's experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ at each time-step t in a replay dataset $D_t = \{e_1, \ldots, e_t\}$. This dataset of recently experienced transitions along with the experience replay mechanism are critical for the integration of reinforcement learning and deep neural networks [262].

Q-learning updates are applied on samples from the training data (s, a, r, s') that are uniformly drawn from the experience replay storage D. The Q-learning update in iteration i uses the following loss function:

$$\begin{aligned} L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(D)} \big[\big(\overbrace{r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})}^{\xi_i} \\ - Q(s, a; \theta_i) \big)^2 \big] \end{aligned} \tag{12}$$

in which θ_i represents the network parameters in iteration i, and the previous network parameters θ_{i-1} are used to compute the target (ξ_i). The gradient of the loss function is computed with respect to the



(a) Deep reinforcement learning agent

(b) deep neural network

Figure 23: (a) The DRL agent. (b) A general deep neural network to be used for supervised DRL.

weights of the network:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}\big[\big(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)\big) \nabla_{\theta_i} Q(s, a; \theta_i)\big] \tag{13}$$

The semi-supervised DRL algorithm is then described in Algorithm 1 to learn from both labeled and unlabeled data.

noend 1 Semi-Supervised DRL Algorithm

Input: A dataset of labeled and unlabeled data $\{(X_l, Y_l), X_u\}$
Initialize the model parameters θ, ϕ, environment, state space, and replay memory $\mathcal{D}$
for *episode* $\leftarrow 1$ to M **do**
  for each sample (x, y) or x in dataset **do**
    $s_0 \leftarrow$ make observation of sample x
    for $t \leftarrow 0$ to T **do**
      Take an action a_t using ϵ-greedy strategy
      Perform action a_t to change the current state s_t to the next state s_{t+1}
      if sample is unlabeled **then**
        Infer the label based on (4): $q_\phi(y|x)$ and get approximate reward $r_t = closeness(s_{t+1}, y)$
      else
        Observe reward r_t that corresponds to label y
      Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
      Take a random minibatch of transitions (s_k, a_k, r_k, s_{k+1}) from $\mathcal{D}$; $0 < k \leq length(minibatch)$
      if s_{k+1} is a terminal state **then**
        $\xi_k = r_k$
      else
        $\xi_k = r_k + \gamma \max_{a'} Q(s_{k+1}, a'; \theta)$
      Apply gradient descent on $(\xi_k - Q(s_k, a; \theta))^2$ based on (13)
    end for
  end for each
end for

Figure 23 shows the high-level model that uses the deep reinforcement learning technique in conjunc-

tion with a generative semi-supervised model instead of a DNN (c.f. Figure 23-b) to handle unlabeled observations. The VAE is extended to have an additional hidden layer and an output to generate the actions.

As other learning processes, the training process for this algorithm is performed offline while policy prediction is performed online. Hence, the algorithm can handle problems with high-dimensional and high-volume data using high performance computing facilities (e.g., cloud servers) to generate the model for online policy prediction. This ability stems from the integration of deep neural networks with reinforcement learning to generate approximation functions for high-dimensional datasets. The performance of this integrated model outperforms the traditional methods of reinforcement learning.

Use Case: Indoor Localization

Several use cases can be envisaged of the proposed approach in a smart city context. For example, this approach can be used for home energy management in conjunction with the Non-Intrusive Load Monitoring (NILM) method and smart meters. In such systems, a small set of labeled data provides individual appliances' usages and their on and off times. A semi-supervised deep reinforcement learning model can be trained over this small-scale training dataset as well as the stream of unlabeled data with the objective of optimizing energy usage by controlling when to switch appliances on and off.

It can also be used in the context of Intelligent Transportation Systems (ITS) by smart vehicles for navigation in a city context. In such applications, a combination of several factors can be used for the reward function such as closeness to the destination, shortest path, speed, speed variability, etc. The vehicle needs to be trained on several test drives then it uses the large set of unlabeled data to accurately navigate through the city.

Due to the importance of indoor localization and ease of implementation, we showcase the proposed method on the localization problem in the context of smart campus, which is part of a larger smart city context. Despite the fact that indoor localization has been studied extensively in recent years, still it is an open problem bringing several challenges that need to be tackled.

Indoor positioning systems have been proposed with different technologies such as vision, visual light communications (VLC), infrared, ultrasound, WiFi, RFID, and BLE [263]. One determining factor for organizations to choose a technology is the cost of the underlying technologies and devices. Among the aforementioned technologies, BLE is a low-cost solution that has attracted the attention for academic and commercial applications [238]. A combination of BLE and iBeacon technologies to design an indoor location-aware system brings many advantages to buildings that are not equipped with Wireless networks. Since iBeacons devices are of a small form factor, they can be deployed quickly and easily without

changing or even tapping into the building's electrical and communications infrastructure [263].

In recent years, deep learning has been shown to perform favorably compared to other machine learning approaches. One main challenge for deep learning is the need to collect a large volume of labeled data (a.k.a calibration procedure). Typically, scanning a large-scale area like a city or a campus to collect unlabeled data is fairly straightforward. Therefore, to benefit from the enormous volume of unlabeled data, we apply the semi-supervised deep reinforcement learning approach to investigate the benefits of unlabeled data in practical scenarios.

Compared to many related works that have performed their studies in a simulated environment, a small area, or in an isolated testbed, we conducted our experiments in an academic library that is a large and busy operational environment where thousands of visitors commute every day. So it is a valuable experiment that can be beneficial for the IoT and AI communities. In addition, there are no similar attempts that address the positioning problem through the reinforcement learning approach.

In this case study, we utilize a grid of iBeacons to implement a location-aware service offering in a campus setting. In our work, we use the iBeacons' Received Signal Strength Indicator (RSSI) as the raw source of input data for a deep reinforcement learning model to identify indoor locations.

RSSI is usually represented by a negative number between 0 and -100 and in localization systems it can be used as an indication of the distance separating the transmitter from the receiver (i.e., ranging). In addition to the separating distance, RSSI is affected by some other factors such as movement of people and objects amidst the signals, temperature and humidity of the environment. The distance estimation from a given point to an iBeacon can be derived as follows:

$$RSSI = -(10\ddot{O}n)log_{10}(d) + A, \tag{14}$$

where n is the signal propagation constant, d is the distance in meters and A is the offset RSSI reading at 1 meter from the transmitter.

Due to fluctuations of the received signal strength, many research studies that utilize RSSI fingerprinting perform a preprocessing step to extract more representative features. Some of these preprocessing approaches include averaging multiple RSSI values for the same location, use Gaussian distribution model to filter outliers, and using PCA to reduce the effect of noise in addition to offering new features. In our work, we performed a categorization preprocessing in which a RSSI category represents a range of RSSI values. We explain the exact procedure in section 3.

Table 10: List of actions to perform positioning.

Action#	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Move to	West	East	North	South	NW	NE	SW	SE

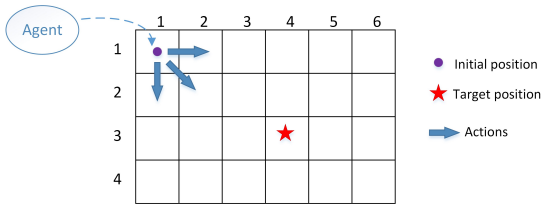


Figure 24: Illustration of a typical indoor environment for deep reinforcement learning.

Description of the Environment

The environment is represented as a set of positions that are labeled by row and column numbers. Each position is also associated with the set of RSSI values from the set of deployed iBeacons. The agent observes the environment by receiving RSSI values at each time. Our design requires the agent to take action based on the three most recent RSSI observations.

The agent can choose one of the allowed eight actions to move in different directions. In turn, the agent obtains a positive or negative reward according to its proximity to the right point. The goal of the agent is to approximate the position of the device that has received the RSSI values from the environment by moving in different directions.

To adopt a deep reinforcement learning approach, we need to define the following elements for the MDP.

Environment: the active environment is a floor on which a particular position should be identified based on a vector of iBeacon RSSI values. The environment is divided into a grid of same-size cells as shown in Figure 24.

Agent: The positioning algorithm itself is represented as an agent. The agent interacts with the environment over time.

States: The state of the agent is represented as a tuple of these observations:

1. a vector of RSSI values,
2. current location (identified by row and column numbers), and
3. distance to the target (for labeled data).

Actions: The action is to move to one of the neighboring cells in a direction of North, East, West, South or in between directions like North West (NW). The first action chooses a random state in the grid. Table 10 shows the list of allowed actions.

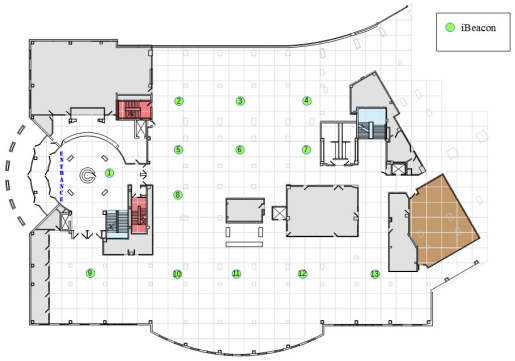


Figure 25: Experimental setup with iBeacons.

Reward function: the reward function is the reciprocal of the distance error. The reward function has a positive value if the distance to the target point is less than a threshold (δ). Otherwise, the agent receives a negative reward. Whenever the agent is close to the target, it gains more rewards. On the other hand, if the agent wanders away from the target and its distance is larger than a threshold (δ), it gains a negative reward. The reward function is represented as follows:

$$r_t = \begin{cases} \frac{1}{\|O_t - S_t\|} & \text{if } 0 < \|O_t - S_t\| \leq \delta \\ -\|O_t - S_t\| & otherwise. \end{cases}$$

in which O_t is the observed location and S_t is the target location.

Experimental Results

Here we describe our evaluation on a real world dataset. Our experiments were carried on the first floor of Western Michigan University (WMU) Waldo library. Figure 25 shows the overall layout of the deployment site. In our work, we use the iBeacon RSSI values to serve as the raw source of input data to identify indoor locations. Smartphones are also utilized to sense the iBeacons' signals and to compute the current position of the user with respect to the set of known iBeacons. Our model utilizes the semi-supervised deep reinforcement learning algorithm to learn from the historical patterns of RSSI values and their corresponding estimated positions to improve its policy when identifying a position based on previously unseen RSSI values.

Dataset

Our dataset is gathered from a real-world deployment of a grid of iBeacons in a campus library area of 200 ft. $\times$180 ft. We mounted 13 iBeacons on the ceiling of the first floor of Waldo Library at Western Michigan University which contains many pillars that might deteriorate the iBeacons signals.

So we arranged the iBeacons such that we could get signal coverage by several iBeacons. Each iBeacon is separated by a distance of 30-40 ft. from adjacent iBeacons. To capture the signal strength indicator of these iBeacons, we divided the area into small zones by mapping a grid that has cells of size 10×10 square ft. We also developed a specific mobile app to capture training data. For that purpose, we stood on each cell and captured all the iBeacons' received signals. We also manually assigned the location (i.e., label of the cell) to the captured signals. We stored at least three instances of RSSIs for each cell to have a more reliable measurement and consequently to reduce the effect of noisy data. Overall, we collected 820 labeled data points for training, 600 data points for testing, and 5200 data points are unlabeled for semi-supervised learning.

Preprocessing

Our initial experiments with the raw RSSI values for supervised deep learning showed that the relationship between the features are not truly revealed by deep learning models. So we have enriched the features by adding two sets of features to the original features. So we have three feature sets as:

- Raw: The original features that come from the direct RSSI readings.
- S1: The set of features that represent the mutual differences of iBeacon RSSI values; i.e., $r_i - r_j; \forall i, j \in iBeacons \ \& \ i \neq j$, representing the difference between the RSSI value of beacon i and beacon j.
- S2: The other set of features designed to represent the categorical values of RSSIs in a Boolean membership mode such that for each beacon we define several categories by a specific interval (e.g., 10) and then represent each RSSI value with the category to which it belongs.

Table 11 shows the average accuracy of the different feature sets during ten replications. These features are added to the raw features. As can be seen from the table, adding features set S1 to raw features has a minor effect on the average accuracy. On the other hand, adding features set S2 increases the average accuracy especially for finer grained positioning. Also, the combination of S1 and S2 is not as good as using only S2, since S1 lowers the accuracy. This observation points out that enriching a feature set by pairwise differences of RSSI values (S1) has a minor negative effect on the accuracy of the model since those features are not solid discriminative factors.

The table also demonstrates that using S2 features when RSSI categorical interval is set to 5 leads to even better results. Therefore, based on these results we use the combination of raw features and S2. Using this preprocessing, each data point x_i is represented as a vector of 13 RSSI values plus 156 range membership features (i.e., 12 range for the 13 beacons) resulting in a total of 169 features: $x_i = (r_1, \ldots, r_{169})$. Each y_i is a label of (row, col) pointing to a specific location.

Table 11: Accuracy of different feature sets in a deep neural network.

Interval	**feature set**	**Accuracy**			
		$\leq$ 1m	**$\leq$ 3m**	**$\leq$ 6m**	**$\leq$ 9m**
-	raw	0.17	0.47	0.74	0.95
10	raw_s1	0.18	0.49	0.75	0.95
	raw_s2	0.26	0.55	0.75	0.97
	raw_s1_s2	0.24	0.52	0.74	0.96
5	raw_s2	0.30	0.57	0.76	0.97

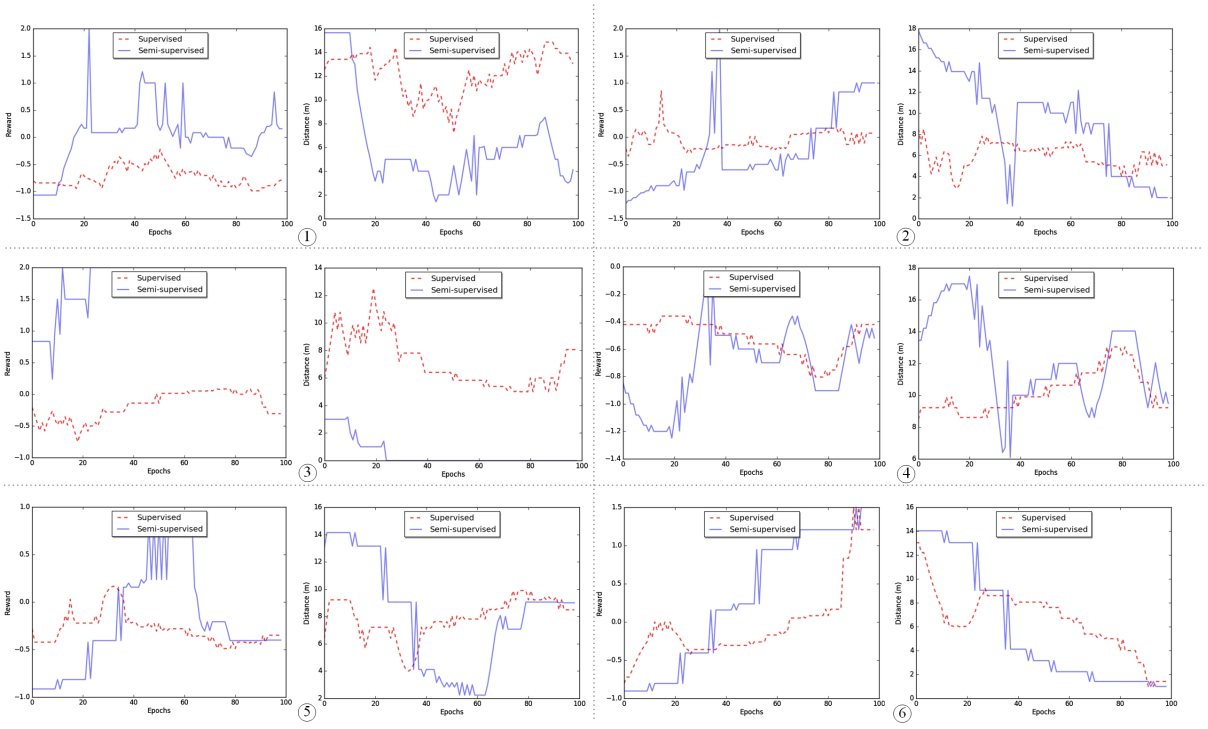


Figure 26: Obtaining rewards and distances in six episodes with a supervised and semi-supervised DRL models.

Table 12: The average speed of convergence to destination points.

	Average distance to the target points (meter)		
	as the agent starts	end of epochs	difference
Supervised	9.4	7.4	2
Semi-supervised	12.8	4.3	8.5

Evaluation

To implement our proposed semi-supervised DRL model, we adopted the deep reinforcement learning algorithm in which we incorporated a variational autoencoder to generate more rewarding policies and consequently increasing the accuracy of the localization process. The deep neural networks are implemented on Google TensorFlow [69] using the Keras package [264].

To evaluate the performance of the proposed semi-supervised DRL model, we performed two sets of experiments: one in which the DRL framework uses a fully-connected deep neural network for supervised learning; and the other in which the DRL framework uses a stacked variational autoencoder for semi-supervised learning.

Figure 26 shows the performance of the DRL in terms of the received rewards as well as distance to the true target for both supervised and semi-supervised models in six episodes (c.f. labels 1-6 on the Figure) . In the plots, it can be seen that the agent in the semi-supervised model learns to achieve higher rewards or smaller distances to the target compared to the supervised model.

Table 12 shows that the behavior of the semi-supervised model leads to getting closer to the target points compared to just relying on a supervised model. It also indicates faster steps to reach or get close to the target in the same number of epochs. The differences of distances in this table emphasize that the semi-supervised model generates policies that improve the average convergence speed of the localization system by a factor of at least 4.

In Figures 27 and 28, the comparison of utilizing the semi-supervised model versus the supervised model along a different number of epochs shows the efficacy of the semi-supervised approach in handling the localization problem. The results in Figure 27 show that the semi-supervised model reaches a higher reward faster compared to the supervised model while keeping its rewards trend stable. From this figure, it can be seen that the semi-supervised model gains at least 67% more rewards compared to the supervised model. In addition, the semi-supervised model achieves about twice the rewards of the supervised model. This result can be translated to the original measurement where we want to know the effect of the models on the accuracy of the localization as depicted in Figure 28. Figure 28 shows the average distance to the target points in different number of epochs. Here the semi-supervised model achieves 6% to 23% improvement for localization. This result indicates that the unlabeled data helps the VAE to better identify the discriminative boundaries and consequently improves the accuracy of the

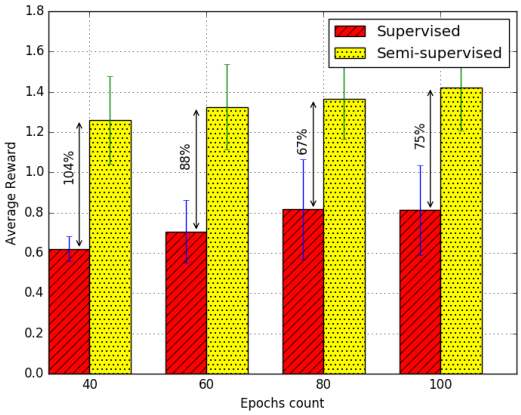


Figure 27: The average rewards that are obtained by DRL over different epoch counts using supervised model versus semi-supervised model.

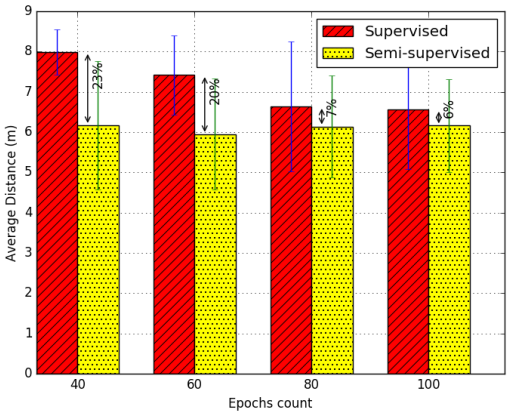


Figure 28: The average distance to the target over different epoch counts using supervised model versus semi-supervised model.

semi-supervised model.

Conclusion

We proposed a semi-supervised deep reinforcement learning framework as a learning mechanism in support of smart IoT services. The proposed model uses a small set of labeled data along with a larger set of unlabeled ones. The current work is the first attempt that extends the semi-supervised reinforcement learning approach using deep reinforcement learning. The proposed model consists of a deep variational autoencoder network that learns the best policies for taking optimal actions by the agent.

As a use case, we experimented with the proposed model in an indoor localization system. Our experimental results illustrate that the proposed semi-supervised deep reinforcement learning model is able to generalize the positioning policy for configurations where the environment data is a mix of

labeled and unlabeled data and achieve better results compared to using a set of only labeled data in a supervised model. The results show an improvement of 23% on the localization accuracy in the proposed semi-supervised deep reinforcement learning model. Also, in terms of gaining rewards, the semi-supervised model outperforms the supervised model by receiving at least 67% more rewards.

This study shows that IoT applications in general, and smart city applications in specific where context-awareness is a valuable asset can benefit immensely from unlabeled data to improve the performance and accuracy of their learning agents. Furthermore, the semi-supervised deep reinforcement learning is a good solution for many IoT applications since it requires little supervision by giving a rewarding feedback as it learns the best policy to choose among alternative actions.

Acknowledgment

The authors would like to thank Western Michigan University Libraries for providing the experimental testbed and space needed to conduct this research.

Chapter 4

Exploiting the Spatio-Temporal Patterns in IoT Data to Establish a Dynamic Ensemble of Distributed Learners

Introduction

The Internet of Things (IoT) utilizes a wide range of advanced technologies including embedded devices and sensors, communications technologies, Internet protocols, and artificial intelligence to provide smart services for more livable communities. The true realization of this vision, beyond the many advanced technologies, depends highly on the quantity and quality of collected IoT data. Particularly, the quantity and quality of data is affected by malfunctions, errors and delays during the collection, transmission, and processing phases.

In recent years, deep learning (DL) models have drawn the attention of researchers in many areas including the Internet of Things and smart services. Even though the complexity and computational intensiveness of DL models impede their use on IoT devices, many attempts are ongoing to facilitate their use on such resource-constrained devices [265].

Currently, many IoT applications utilize DL models on the cloud for analytic/prediction purposes. These models are typically responsible for providing suggestions to end-users. To produce models that achieve higher accuracy, learning models ought to exploit the hidden spatio-temporal patterns that are

embedded in the training data. Such hidden patterns might be hard to discover by the learning models themselves. For example, in a smart home application, the Heating, Ventilation and Air Conditioning (HVAC) controller should be able to set the temperature to a certain level based on the inferred activities of the household (e.g., sleeping, exercising, etc.) and other parameters (e.g., windows are open, oven is on, etc.). In this scenario, the HVAC controller should predict the optimal temperature based on its raw input data as well as other indicators that convey the extent to which predictions align with previously captured sensors' data (i.e., historical patterns).

IoT systems have to operate in environments that are continually evolving over the time. Therefore, their learning mechanisms should be flexible and adaptive to cope with changes in their environment. In our example on smart-home applications, if the house is sold to a new household, the HVAC controller should perform in a different way based on the preferences of the new occupants since their activity patterns and temperature preferences might be different from those of the previous occupants. This is where Machine Learning (ML) algorithms shine as they can adapt to the new conditions and change their predictions accordingly.

IoT applications present several challenges for learning mechanisms:

- IoT applications and sensor devices generate a lot of high dimensionality data. The high dimensionality of such data hardly can be handled by shallow learning models such as Support Vector Machines (SVM) or K-Nearest Neighbors (KNN). Utilizing Deep Neural Networks allows for the creation of accurate predictors that benefit from the availability of IoT big data.

- Producing continuous streams of data that are bound to specific locations is another unique property of IoT applications. This property can be utilized to exploit the spatio-temporal patterns in training data to identify the typical patterns seen in the output of a prediction model.

- IoT sensors may stop data sensing, collection, or reporting due to malfunctions. Sensor malfunctions might be due to dead batteries, loss of connectivity, damaged sensors, etc. Such malfunctions might lead to limitations in the availability of training data which in turn limits the performance of the generated prediction models.

- There are many practical scenarios that require the data pertaining to IoT applications to be kept privately, and not be exposed to the public. In such scenarios, sending training data to a centralized learning agent might raise privacy and security concerns [266]. Instead, such sensitive data can be used to build a local prediction model. Multiple local prediction models can then be utilized to create an ensemble that is stronger than its underlying local prediction models.

To tackle these challenges, we propose to develop an ensemble of DL models that utilizes the spatio-temporal patterns embedded in the training data. The spatial and temporal patterns are usually im-

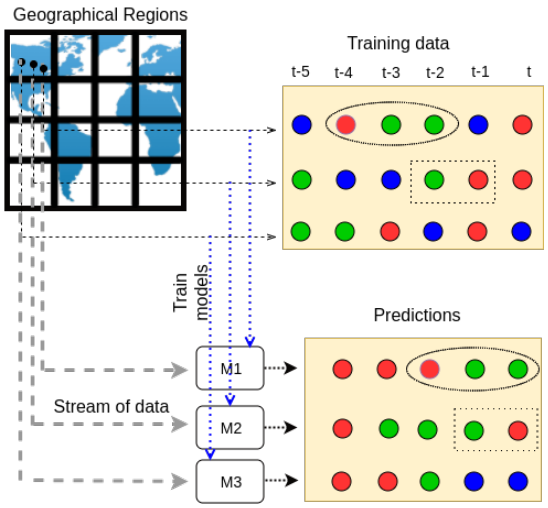


Figure 29: The spatio-temporal patterns that are embedded in training data and predictions.

plicitly available in data [267]. Based on the time span of the spatio-temporal patterns, we weigh the individual models to create an ensemble output. Historical data with spatio-temporal dependencies of various time spans are stored in Bloom Filters (BFs). The proposed approach is self-adaptive and does not depend on offline parameter configurations. Instead, it relies on the output of the individual learning models and the extent to which they match previously observed patterns (i.e., ground truth training data) that are stored in the BFs. The extent of this match determines the weight of the underlying prediction model in the ensemble's output. The weight of each prediction model participating in the ensemble might evolve over time depending on its ability to generate outputs that match the ground truth observed in the training data.

Conventional ensemble approaches such as majority voting or averaging methods have to be pre-configured offline with static weights. On the other hand, our proposed approach weighs the prediction models based on the alignment of their outputs with the training data.

Motivations

In this research, a spatio-temporal pattern refers to a sequence of predictions of an agent that pertain to a specific spatial location. Figure 29 illustrates the concept of spatio-temporal patterns. Each prediction model is associated with a specific geographic region. The following considerations motivate us to investigate the use of spatio-temporal patterns to create a dynamic ensemble of deep learning agents:

- Models that are trained on a portion of the training data lead to weak learners. In an ensemble model, the weak learners complement each other to create a single stronger model.
- Time plays an important role in IoT data, as most of the IoT generated data is time-stamped. Also, location is a major factor in IoT data as data are identified by their origination point in space

(e.g., GPS coordinates). Therefore, ignoring such important features might lead to less accurate predictions. In this study, we are set to explore the answer to the following research question: Does the use of spatio-temporal patterns embedded in IoT training data help in the creation of dynamic ensemble models that outperform models that require prior static configuration?

- Massive IoT deployments can benefit from adaptive and self-reconfiguring ensemble models. Such ensemble models should work with a variety of time and geographically stamped data, including electricity load demand forecasting, fault prediction, anomaly detection (e.g., driving anomaly), etc. They should also be applicable to video analysis where image frames are sequentially dependent.

Contributions

The contributions of this research are as follows:

- We propose a new algorithm that creates an ensemble of machine learning models that utilizes the spatio-temporal patterns embedded in the training data.
- Our proposed approach does not rely on offline configurations. Furthermore, it outperforms models that require prior static configuration.
- We showcase the proposed approach to different IoT applications, including electricity demand forecasting, traffic volume prediction, and localization in smart buildings.

The rest of the paper is organized as follows. Section 4 introduces several research attempts related to our research study. Section 4 reviews some of the background used in the proposed work, including deep learning, Bloom filters, and ensemble learning. Section 4 formally introduces the problem formulation behind our research work and our proposed approach. In Section 4, we showcase several use cases that can benefit from the proposed approach. In Section 4, experimental results are presented and discussed. Section 4 discusses the efficacy of the proposed approach in different deployment scenarios. Finally, Section 4 concludes this study.

Related Work

Although ensemble methods have been proposed and used for deep learning models in recent years, most of the literature does not benefit from the spatio-temporal patterns embedded in the training data to obtain better prediction results. The intuition is that a single Deep Neural Network (DNN) prediction model is constructed based on a local optimum solution during the training phase. Having several DNNs that are constructed based on different local optima can collectively cooperate to converge to better

prediction results [268]. In this section, we review some of the recent research attempts that have used an ensemble technique to aggregate deep learning outputs.

Qiu *et al.* [219] proposed an ensemble of deep belief networks (DBN) for regression and time-series forecasting. In their proposed model, the authors generate different prediction models by training the same DBN architecture over different number of epochs. Having 20 DBN prediction models, the outputs of these models constitute the input for a Support Vector Regression (SVR) model to predict the final output. They evaluated their model on three regression datasets related to electricity load demand forecasting and one time-series dataset. The results show improvements over the baseline approaches. However, compared to our work, their work utilizes the whole training dataset for each model. Instead, our approach utilizes different parts of the training data to create multiple prediction models. Also, our approach utilizes spatio-temporal information to generate an ensemble output.

In another recent study, Qiu *et al.* [269] used an ensemble model to aggregate the predictions of DBNs using the Empirical Mode Decomposition (EMD) algorithm. Their approach relies on a pre-process stage that prepares the data for the underlying models. The ensemble algorithm then is the aggregate of the individual outputs of the underlying models.

Recurrent Neural Networks (RNNs) and Long Short-Term Memory neural networks (LSTMs) have been used widely for temporal and time-series analysis. Liu *et al.* [270] have extended LSTM to support spatio-temporal data for human activity recognition based on a human skeletal dataset. Their spatio-temporal LSTM (ST-LSTM) network aims to model the temporal dependencies of consecutive frames as well as the spatial dependencies among different joints in one frame. Each body joint is represented by an LSTM unit which is fed by hidden features of its own joint from the previous time step as well as the hidden features of its previous joint at the current time step. Their experimental results illustrate the effectiveness of adding spatio-temporal dependencies analysis to the accuracy of human activity recognition. Our work is different from this approach since in RNN and LSTM based approaches, a fixed length temporal dependency is utilized in the network while our approach dynamically determines the best length of spatial and/or temporal dependencies in the training data.

Ortiz *et al.* [271] experimented with an ensemble of DBNs with several voting schemes to diagnose the Alzheimer's disease based on brain images. They experimented with majority voting, weighted voting, weighting individual classifiers based on SVM training, and fusion using a DBN. Based on their results, SVM-based ensembles outperform other schemes while DBN-based ensembles were the least effective. A problem with the voting scheme is that it considers all of the underlying models to be equally contributing to the final output and does not weigh individual predictors.

The work of Wang *et al.* [272] also uses a probabilistic ensemble scheme. Their underlying deep learning models are Convolutional Neural Networks (CNNs) to extract features from wind power frequencies.

Table 13: The position of the current work compared to the literature.

Research	Learning Technique			
	Deep Learning	Ensemble Learning	Distributed Learning	Spatio-temporal Patterns
Qiu *et al.* [219]	✓	✓		
Liu *et al.* [270]	✓			✓
Ortiz *et al.* [271]	✓	✓		
Wang *et al.* [272]	✓	✓		
[69, 273, 274, 275, 276]	✓		✓	
Proposed	✓	✓	✓	✓

The ensemble method is based on the probabilistic distribution of wind power data that are statistically formulated and are in charge of filtering uncertain model outputs. Their evaluation on wind farm data demonstrates the efficacy of the proposed approach.

Most of the works on distributed deep learning focus on improving the training time of the neural network by partitioning the execution of a complex DNN among several CPUs [273], GPUs [274, 69, 275], or a distributed computing hierarchy [276]. From a communications perspective, these approaches need to transfer DNN parameters over the network. Based on the size of the DNN, the transferred parameters may induce a considerable amount of network traffic. In the IoT ecosystem, IoT devices or edge nodes may have their own local learning agents. These agents can benefit from a learning agent that aggregates their outputs. This direction has not been investigated much in the literature. This paper explores the potential to utilizing distributed deep learning in support of IoT applications.

Table 13 summarizes the algorithms and techniques that have been utilized in the aforementioned research works and shows the position of this work compared to these studies. As shown in this table, our proposed architecture utilizes the strength of several approaches that are suitable for IoT data analytics to generate better prediction capabilities.

Background

In this section, we review a brief overview of the fundamental components utilized in our proposed approach.

Deep Learning

Artificial Neural Network (ANN) techniques have been developed to imitate the functionality of human brain and neurons to process sensory signals. Deep learning algorithms have been developed on top of traditional ANNs by addressing several limitations. ANNs do not perform good when the number of layers is increased. Small-size training data also caused the trained models to be overfitted.

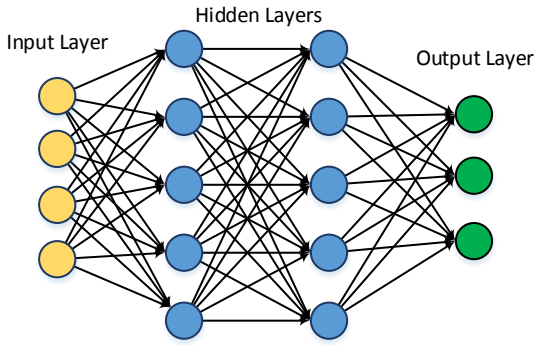


Figure 30: Architecture of a deep neural network.

Hardware limitations inhibited the implementation of efficient deeper ANNs. The latest advancements in computational capabilities in general and the development of Graphics Processing Units (GPUs) has helped to lift up these computation limitations. More importantly, DL techniques are indebted to the advancements and improvements in training algorithms including: the introduction of Rectified Linear Units (ReLUs) as the activation function in neurons, the introduction of dropout methods to reduce the volume of computations in each layer, the change in the initialization procedures for the weights of the network, the introduction of residual learning networks to address the problem of degradation of training accuracy , and addressing the vanishing and exploding gradient problems by introducing and enhancing LSTM networks [265]. In addition to the aforementioned advantages of DL techniques compared to the traditional ANNs, DL algorithms can learn hidden features from the raw data, and using multiple layers, represent a hierarchy of features. Figure 30 shows the structure of a typical DNN.

Bloom Filter

Bloom filters serve as an efficient data structure in terms of time and space complexity to store a set of elements. The Bloom filter and its variations have been widely used in distributed systems [277] and have drawn the attention of network researchers in recent years, especially in the IoT area [278]. A significant property of Bloom filters is that they can examine whether an element belongs to a set in a constant time. False positives, on the other hand, are the main drawback of this data structure. Considering the impact of false positives in the underlying application, we can stick to the Bloom filter principle: When the effect of false positives can be discounted, Bloom filters can be used instead of ordinary sets or lists [279].

The structure of a Bloom filter to represent a set of n elements can be implemented using an array of m bits. All bits are initialized to 0. The Bloom filter uses k independent hash functions $h_1, ..., h_k$ that generate uniform random numbers in the $1, ..., m$ range.

The probability of a false positive can be formulated as $(1 - p')^k$, where p' is the probability of a bit

Table 14: System parameters.

	Symbol	Description
Constants	T	End time step
	L	Maximum length of spatio-temporal patterns
	δ_0	minimum quantization level
	Δ	Maximum quantization level
Variables	m	Number of models
	M_i	Individual Model; $1 \leq i \leq m$
	B_i	Bloom filter associated with M_i
	l_i^t	Match length to the Bloom filter B_i at time t
	δ	Quantization factor
	$\gamma^{i,t}$	Longest sequence match at time t for M_i

to be 0:

$$p' = (1 - \frac{1}{m})^{kn} \approx e^{-kn/m} \tag{1}$$

In this work, Bloom filters are used to find the maximum length of spatio-temporal patterns between the output of a prediction model and the training data that is stored in a Bloom filter. For example, a Bloom filter of length 3 stores training tuples of three consecutive time slots t, $t-1$, and $t-2$. This Bloom filter is queried and the output of the query is utilized in the inference process of our proposed ensemble model.

Ensemble Learning

The idea behind ensemble learning is to create a stronger predicting model with an arbitrarily low error, using an ensemble of inferior predictors that have an error rate that can only be better than that of a random guess (i.e., their error rate is better than 50%) [280]. However, it is worth mentioning that an ensemble prediction model is not guaranteed to reach a prediction accuracy that is better than the best model in the ensemble. Instead, it can help to decrease the chance of selecting a weak predictor as the final output [281].

Spatio-temporal Ensemble Learning Method

Problem Formulation

Our proposed ensemble method utilizes m individual learning models $\{M_1, ..., M_m\}$ distributed over m geographic regions. Bloom filter B_i is associated with learning model M_i in the ensemble of models. The training data of model M_i embeds spatio-temporal patterns that are stored in B_i. Model M_i generates a prediction sequence over time as $(y_0^i, y_1^i, ..., y_t^i, ...)$.

Given a sequence of ground truth values $G = (\dot{y_0}, \dot{y_1}, ..., \dot{y_T})$, we also define

$$B_i = \{(\dot{y_t}, ..., \dot{y}_{t+l-1}) \mid 0 \leq t \leq \ T - l + 1, \ 1 \leq l \leq L\}$$

as the set of sub-sequences of length l of sequence G starting from time step t. We also define L to represent the maximum value that l can assume.

As in the case of many search applications, our approach assumes a tolerance value $\delta > 0$ that serves as a quantization level. All values that are hashed by $\lfloor \frac{v}{\delta} \rfloor \times \delta$ to the same value are stored in the same cell of the Bloom filter. For example, with a Bloom filter that stores 3.55 as a ground truth value, a query for a value of 3.54 returns *true* when δ is set to 0.02). However, a Bloom filter only checks if an exact value is present in the set. We resolve this issue by storing quantized values in the Bloom filters using δ such that $\delta \in \mathcal{D} = \{\delta_0, \delta_1 ..., \Delta\}$. Any value in the range $[\hat{y}, \hat{y} + \delta)$ is hashed to $\hat{y}$ which is in turn stored in the Bloom filter. The value of $\hat{y}$ is calculated as follows:

$$\hat{y} = \lfloor \frac{y}{\delta} \rfloor \times \delta. \tag{2}$$

Table 14 lists all the constants and variables used in our approach.

Our proposed ensemble learning problem can be expressed as a multi-objective programing problem (MOPP) [282] that attains the following objectives for each model M_i at each time step t:

$$\text{Objective 1:} \qquad \gamma^{i,t} = \arg\max_{\beta \in B_i} ||\beta||, \tag{3}$$

$$\text{Objective 2:} \qquad \delta_t^i = \min \delta \in \mathcal{D}; \tag{4}$$

such that

$$\sum_{k=0}^{l_t^i - 1} |\hat{y}_{t-k}^i - \gamma_{l_t^i - k}^{i,t}| = 0 \tag{5}$$

in which $\hat{y}_t^i$ is the quantized value of the prediction of M_i at time t, and $l_t^i = ||\gamma^{i,t}||$. We refer to the kth element of $\gamma^{i,t}$ by $\gamma_k^{i,t}$. Objective 1 obtains the sub-sequence of the longest match from the training data stored in B_i while objective 2 wants to utilize the smallest quantization level. Both objectives must respect Constraint (5) to retrieve a match between the predictions and the training data.

The Proposed Algorithm

There is no single global solution for a MOPP to optimize both objectives at the same time [282, 283]. To solve the aforementioned problem, we devised a heuristic algorithm that returns one Pareto optimal solution from the set of possible Pareto optimal solutions.

In our ensemble method, we use several Bloom filters to track the historical spatio-temporal patterns with different lengths. In this regard, ground truth values at each time step from the training datasets along with various consecutive past observations are quantized by different values of δ and grouped together to be stored in the Bloom filters. In the inference phase, each prediction and its $l-1$ predecessors are also quantized and grouped together to be queried against the corresponding Bloom filters to check if such pattern of items exists or not (i.e., length 3 is queried against the Bloom filter that keeps three consecutive items).

Based on the length of the longest match in a Bloom filter (i.e., l) and δ, the weight of the contribution of each model to the ensemble's output is adjusted. Smaller δ and larger Bloom filter match l, lead to a higher weight for a given model. Precisely, the weight of model i at time t is updated by:

$$w_t^i = \frac{\frac{l_t^i}{\delta_t^i}}{\sum_{k=1}^{m} \frac{l_t^k}{\delta_t^k}} \tag{6}$$

Figure 31 illustrates the high level diagram of the proposed approach and figure 32 shows the mechanism of filling and querying Bloom filters with historical data at each time step. There are situations where none of the model outputs match values stored in the Bloom filters. In such case, a deep learning model in the ensemble is trained such that the individual models' predictions are fed to it in order for it to provide the ensemble output. Albeit it does not benefit from the spatio-temporal patterns in the training data, we use this mechanism since in such a case the model will end up with equal weights in the ensemble of models which is not fair for the best performing models. Algorithm 2 lists the pseudocode of our proposed spatio-temporal ensemble learning method.

Pareto Optimality

Definition 1. *Pareto Optimal (for minimization problem): A point,* $\mathbf{x}^* \in \mathbf{X}$*, is Pareto optimal if there does not exist another point,* $\mathbf{x} \in \mathbf{X}$*, such that* $F_i(\mathbf{x}) \leq F_i(\mathbf{x}^*); \forall i \in \{1, 2, ..., k\}; k \geq 2$*, and* $F_j(\mathbf{x}) < F_j(\mathbf{x}^*)$ *for at least one* $j \in \{1, 2, ..., k\}$*, where* $\mathbf{X}$ *is the set of feasible decision (design) space, and* F_i *is an objective function* [282].

Definition 2. *Utopia (ideal) Point: A point,* $\mathbf{F^o} \in \mathbf{Z}^k$*, is a utopia point if for each* $i = 1, 2..., k; k \geq 2$*,* $F_i^o = \min_x \{F_i(\mathbf{x}) | \mathbf{x} \in \mathbf{X}\}$*, where* $\mathbf{Z}$ *is the feasible criterion space.*

$\mathbf{F^o}$ is not always feasible. The next best solution would be the one that is called a *compromise solution* and is as close as possible to the utopia point. The closeness can be computed through a distance measure such as Euclidean distance. Such a solution is Pareto optimal [284].

Proposition 1. *When a Pareto optimal solution exists to the MOPP problem, our proposed algorithm returns that solution.*

noend 2 Spatio-temporal Ensemble Algorithm

Input: Predictions from m models; namely $1, ..., m$, at each time step t: $(y_t^1, y_t^2, ..., y_t^m)$

1: Initially, create m Bloom filters where B_i is associated with M_i. Each B_i stores tuples with various lengths $l = 2, 3, ..., L$
2: `/* Training phase */`
3: **for** each data point y_t in training dataset **do**
4: **for** each Bloom filter B_i with length $l = 2$ to L **do**
5: Generate sub-sequence $s = (y_t, y_{t+1}..., y_{t+l-1})$
6: $\hat{s}$ = Quantize values in s based on (2) for each $\delta \in \mathcal{D}$
7: Insert $\hat{s}$ to B_i
8: `/* Inference phase at time t */`
9: **for** each M_i ; $i \leftarrow 1$ to m **do**
10: **for** $l = L$ to 2 in B_i **do**
11: **for** δ in $\{\delta_0, \delta_1..., \Delta\}$ **do**
12: $\hat{\mathcal{Y}}$ = Quantize $(y_t^i, ..., y_{t-l+1}^i)$ based on (2)
13: **if** $\hat{\mathcal{Y}} \in B_i$ **then**
14: `/* This is the longest match */`
15: Get (l_t^i, δ_t^i)
16: Go to 9:
17: **if** No match **then**
18: Use general deep model
19: **else**
20: Update w_t^i based on (6), $\forall i \in \{1, 2, ..., m\}$
21: Compute weighted sum of $(y_t^1, y_t^2, ..., y_t^m)$ based on $(w_t^1, w_t^2, ..., w_t^m)$

Proof: Considering objectives 1 and 2, and based on definition 2, the utopia (ideal) point for our MOPP is where $(l, \delta) = (L, \delta_0)$. The algorithm starts from such an ideal point to search for a solution by matching the predictions of the individual models to the Bloom filters. If that point leads to a solution (i.e., a match to the Bloom filter), then the proposition is proved. Otherwise, the algorithm tries the next closest point in the space (compromise solution) until such a point is found. That is, the algorithm at least finds a compromise solution if any. So, based on the definition of compromise solution, the result of the algorithm is considered Pareto optimal and the proposition is proved. ■

Time Complexity

Given n ground truth values, the time complexity of Algorithm 2 during the training phase is $\mathcal{O}(nmL||\mathcal{D}||)$ as it stores spatio-temporal patterns of all possible lengths while using all quantization levels. The time complexity for the ensemble prediction of a given input during the inference phase is $\mathcal{O}(mL||\mathcal{D}||)$. The best case requires a time complexity of $\mathcal{O}(m)$ as the longest spatio-temporal length with the lowest quantization level leads to a match to the Bloom filter (c.f. loop at lines 10 - 11).

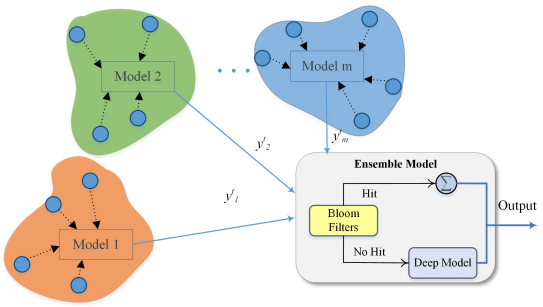


Figure 31: Several deep learning models processing data from different geographic regions participate in ensemble learning.

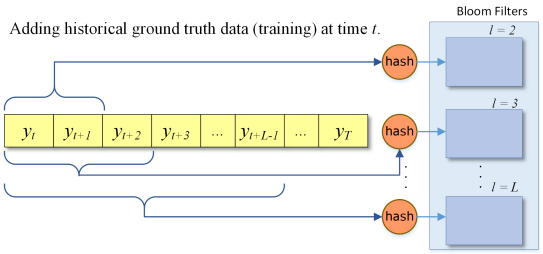


Figure 32: Adding spatio-temporal data into Bloom filters starting at time t.

Use Cases

Several use cases can benefit from the proposed approach. In this section, we highlight the role of the proposed approach in three IoT and smart city application domains.

Smart Grid

Saving energy and preserving the environment are the main drivers for energy providers to monitor consumers' energy usage profiles and provide suitable feedback to them to decrease the high-peak power load. Modern electricity meters (i.e., smart meters) are installed on customers premises for power monitoring and billing purposes. The two-way communications between smart meters and a central management system enables the smart meters to send real-time or periodical data to the electric utilities [285]. The reported data typically include measured consumption, quality of power, and power outage alerts. This data, in conjunction with other third-party sensor data like weather and traffic data, can lead to more accurate energy usage forecasts [286].

To deploy a large-scale Smart Grid (SG) infrastructure, scalable data and software platforms are needed to integrate and analyze data streams from a myriad of smart meters, to predict power usage and to react to operational events [287]. The proposed approach can be utilized in such a context to create an ensemble learning model for a region within a city based on the spatio-temporal data of several families that live in the region.

Smart Transportation

Smart transportation applications are an important aspect of the IoT and a prominent enabler for smart cities. Smart transportation can improve the safety and efficiency of traditional transportation systems and contribute to a sustainable environment [288]. A key functional component of Intelligent transportation systems (ITS) is online vehicular traffic flow prediction. In ITS, roads and highways are equipped with sensors to count the vehicles and report traffic flows. Online vehicular traffic flow prediction systems heavily rely on the history of past observations and the real-time traffic data collected from specific road segments [289]. Such systems enable travelers to make efficient route decisions. They also provide appropriate information and guidelines to the authorities for the overall administration and maintenance of the transportation system [290].

Online vehicular traffic flow prediction applications can highly benefit from the spatio-temporal patterns embedded in their training data. Limitations in traffic volume training data (e.g., existence of gaps and missing data) are common which in turn degrade the prediction accuracy [291]. The proposed ensemble approach can be used in such applications to deliver more accurate predictions by utilizing the spatio-temporal patterns that are hidden in the traffic volume data of road segments.

Smart Building

The importance of smart buildings in the context of a smart city is highlighted by the fact that citizens spend more than 80% of their time indoor for a variety of purposes such as work, shopping, and education [61]. Equipping such indoor environments with smart services to meet the needs of their occupants is a beneficial investment. One necessary service is offering fine-grained localization to the occupants and providing adequate information for wayfinding and path planning. There is a growing interest in the potential use of low-cost wireless communications infrastructure such as existing wireless networks or BLE beaconing technology combined with users' smartphones or wearable devices to deliver location-aware services.

Localization is also a task in which spatio-temporal patterns affect the accuracy [292]; i.e., a person at position p_t at time t is more likely to have been in a nearby position at time $t-1$. Our proposed ensemble approach can be applied to such applications to learn the spatio-temporal patterns in a sequence of positions typically traveled by users.

Performance Evaluation and Analysis

In this section, we first describe three different datasets associated with the use cases described in Section 4. Then the settings and configurations of the proposed model are detailed followed by the

experimental results.

Datasets

Electricity Demand

We use the electricity load demand dataset from Australian Energy Market Operator (AEMO) [293]. We used the monthly datasets of year 2013. Specifically, we used the first nine months for training and the last three months for testing [219]. Data points in this dataset are gathered in 30 minute intervals. The features that we use include, date and time, and the power demand. Each Australian state has 17472 samples.

Vehicle Traffic Volume

The vehicle traffic volume dataset is obtained from the CityPulse Dataset Collection [294, 295] that is provided by the city of Aarhus in Denmark. Data is sampled every five minutes. We aggregated them to every 30 minutes. The features that we use include: road segment, date and time,and traffic count. For this experiment, data of a specific segment from February 2014 - June 2014 with 5426 samples is used.

Indoor Localization using BLE RSSI

This dataset is based on the Relative Signal Strength Indicator (RSSI) readings from an array of 13 iBeacons deployed at Waldo Library, Western Michigan University [61] [296]. Data collection was performed during the operational hours of the library. The input data contains a timestamp, 13 RSSI readings, and two label features for the actual location (i.e., `x` and `y` coordinates) such that consequent samples create a trajectory. The dataset contains 1420 labeled samples.

Settings

For all the experiments, we use predictions of $m = 3$ individual deep learning models. In this work, we utilize feed-forward DNNs with four hidden layers and one output layer. The number of neurons at each layer is 120, 10, 5, 3, and 1, respectively. Each layer uses ReLU as the activation function and the Mean Absolute Error is used as the loss function along with Adam as the optimizer. This configuration is used for the three datasets. As a preprocessing step, the values of the past 24 hours are used as the input features for electricity and traffic models. These input features are selected based on the autocorrelation function (ACF) as has been reported in the literature [297]. For the localization model, we use the past three observations as the input features.

The maximum length of a spatio-temporal pattern in the ensemble algorithm is set to $L = 4$. That is, we used three Bloom filters to store the spatio-temporal patterns of lengths 2, 3, and 4. The quantization levels are in $\{1, 2, 3, 4\}$.

Results

This section presents the empirical results obtained using the proposed ensemble algorithm. We compare the accuracy of the ensemble model with the accuracy of the individual models as well as a simple average method. The simple average method only takes the mean of the outputs of all models at each time step.

Accuracy of the Spatio-Temporal Ensemble Model

We compare the accuracy of the models, in terms of cumulative error rate, to show the improvement of the ensemble model over time. We also show the average error rate of the models over the whole time of the experiments. We show the cumulative error in a short time frame (e.g., 24 hours) to better demonstrate the mechanism utilized by our proposed ensemble method and its use of Bloom filters to adjust the weights. We also show the cumulative error in a longer time frame (e.g., one week) to highlight the performance of different models in the long term.

We use the Mean Absolute Percentage Error (MAPE) to measure the accuracy of the electricity demand predictions. For the traffic volume count and localization use cases, we use the Mean Squared Error (MSE). These two metrics are defined as follows.

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{y_i - \tilde{y_i}}{y_i}, \tag{7}$$

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y_i})^2, \tag{8}$$

where y_i is the ground truth value and $\tilde{y_i}$ is the prediction value. The choice of these metrics is based on the underlying dataset. That is, the electricity data has a wide range and the difference between the predictions and the ground truth values reflects the error. While for the two other cases, the MAPE is unsuitable as it might result in division by zero when the ground truth value is zero.

Electricity Demand Forecasting. Figure 33(a) shows the performance of the ensemble model for the electricity dataset in terms of cumulative error rate compared to three individual learners and the simple average method. The figure shows that the individual models perform differently in various time spans. At the beginning, $M2$ is the best model, then $M3$ becomes superior at hour 4. At hours 18 and 19, $M1$ performs slightly better than the other individual models. The ensemble model, however, strives

to benefit from all the models. Consequently, it is more stable overall compared to the individual models and its error rate is always less than that of the individual models. Compared to the simple average method, the ensemble is also superior overall.

Figure 33(b) depicts the change of weights over time for each individual model of the ensemble. This indicates that the ensemble changes the weights among the models based on their past performance to compute the best output. Changing the weights is instantaneous as this is computed based on matching the sequence of predictions with the Bloom filters. Figure 33(c) shows the usage of various Bloom filters by the ensemble. This figure shows that the results of matching predictions against the set of Bloom filters change over time. These changes directly affect the weighing mechanism as demonstrated in figure 33(b). When there is only one model that matches the set of Bloom filters, that model gets the whole weight for the final output. In scenarios that do not result in a model matching a Bloom filter, the final output is obtained from the deep learning model. Figure 34 shows the average error rate of the ensemble compared to the individual models as well as the simple average method. The figure illustrates that the ensemble model outperforms the other approaches. Although the simple average method shows a better performance compared to the individual models, the ensemble outperforms it since the ensemble approach is capable of achieving better weighting based on the spatio-temporal patterns embedded in the training data; instead of utilizing fixed weights for the models. Figure 35 shows the performance of the ensemble over a period of one week.

Vehicular Traffic Volume Prediction. In our vehicular traffic dataset experiment, the ensemble model almost follows the best model which is $M1$ as illustrated in Figure 36(a). The weighting trend, as demonstrated in Figure 36(b), shows a minor change in weights until hour 14, then the weights remain unchanged for the rest of the day. This trend is based on the pattern of Bloom filters' matches as shown in Figure 36(c). Although the changes of matching to the Bloom filters are insignificant, these changes contribute to the overall performance of the ensemble model. If all the underlying models show a similar matching pattern, the resulting weighting mechanism will be a simple average. However, Figure 36(a) shows that our proposed spatio-temporal ensemble method often results in superior error rate compared to the simple average method. Figure 37 shows the average error rate of the models for the vehicular traffic volume. The figure clearly shows that, on average, our ensemble method performs better than all the underlying models. Figure 38 asserts this improvement over a period of one week.

Indoor Localization. In our localization experiment, model $M2$ always performs better than the other models as shown in Figure 39(a). Since the simple average method considers the output of the weaker models (i.e., $M1$ and $M3$), it does not achieve an accuracy as good as that of the ensemble model. Here, our ensemble approach results in a higher weight for model $M2$ since it shows the long and steady spatio-temporal patterns in the Bloom filters as illustrated in figures 39(b) and 39(c). Figure 40 also

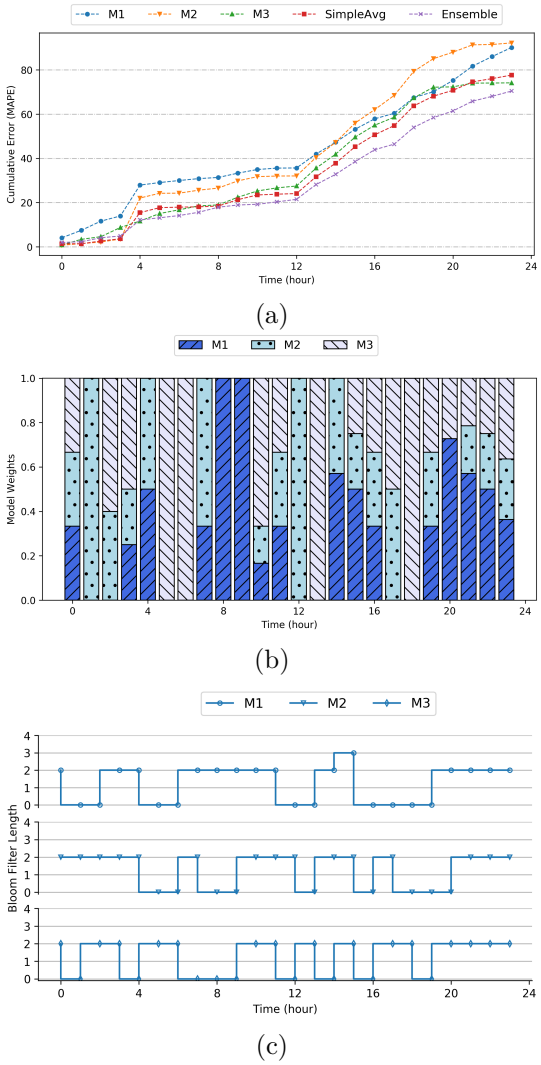


Figure 33: (a) Cumulative error rate of the ensemble model vs. the individual models and simple average method for electricity demand dataset. (b) Change of the individual models' weights over time. (c) Change of the length of the matches of the models' outputs to the Bloom filters.

shows the average performance of the models which highlights the superiority of the ensemble model over the individual models as well as the simple average model. The longer cumulative error rate for 48 steps is depicted in Figure 41 that confirms the superiority of the ensemble model.

Table 15 summarizes the percentage of the improvement that the ensemble method has obtained compared to the simple average method as well as the individual models in the experiment. The improvement is computed by $(e_{ensm} - e_{base})/e_{base}$, where e_{ensm} is the average error rate of the ensemble model and e_{base} is the average error rate of the baseline model with which we are comparing (i.e., the individual models or the simple average model).

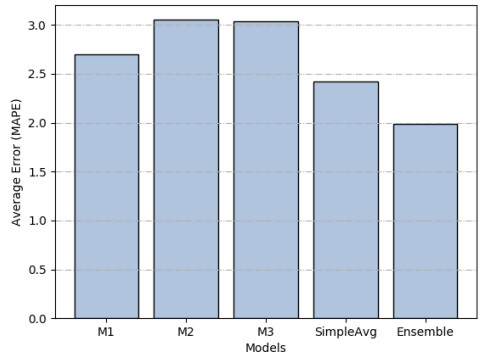


Figure 34: Average error rate of the ensemble model vs. the individual models and simple average method for electricity demand dataset.

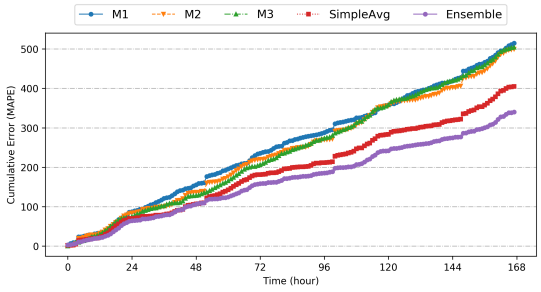
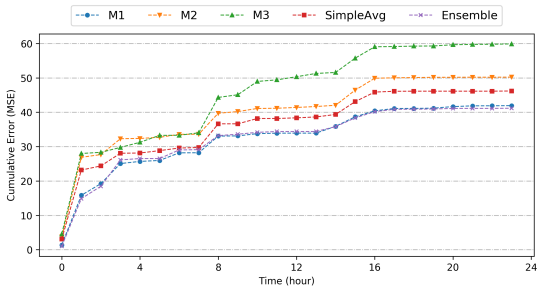


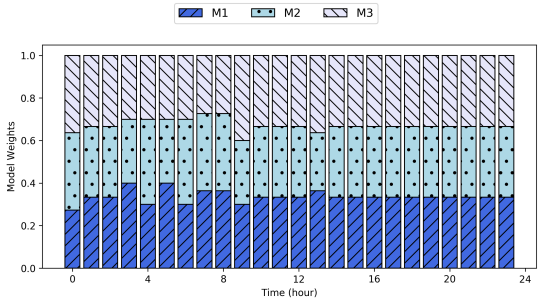
Figure 35: Long-term cumulative error rate of the ensemble model vs. the individual models and simple average method for electricity demand dataset.

Table 15: Improvement of the ensemble algorithm compared to baseline models (individual models and simple average) when 33% of training data is used.

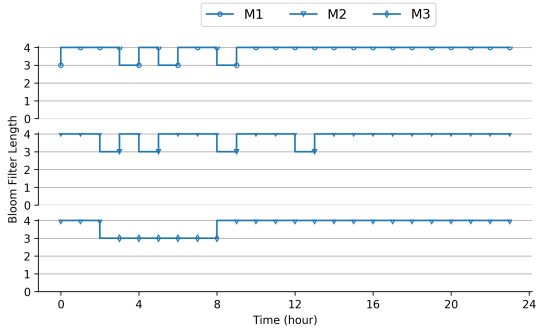
Experiment	**Models**			
	Avg	**M1**	**M2**	**M3**
Electricity	18%	26%	35%	39%
Traffic	5%	1.5%	6%	24%
Localization	33%	65%	25%	59%



(a)



(b)



(c)

Figure 36: (a) Cumulative error rate of the ensemble model vs. the individual models and simple average method for vehicular traffic volume dataset. (b) Change of the individual models' weights over time. (c) Change of the length of the matches of the models' output to the Bloom filters.

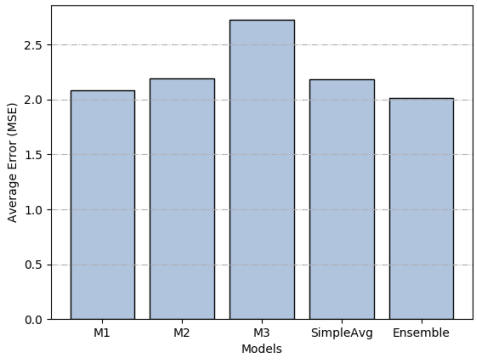


Figure 37: Average error rate for the ensemble model vs. the individual models and simple average method for vehicular traffic volume dataset.

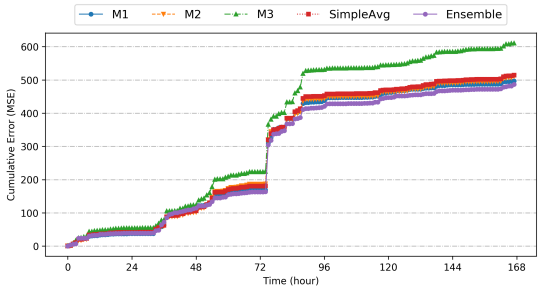


Figure 38: Long-term cumulative error rate of the ensemble model vs. the individual models and simple average method for vehicular traffic volume dataset.

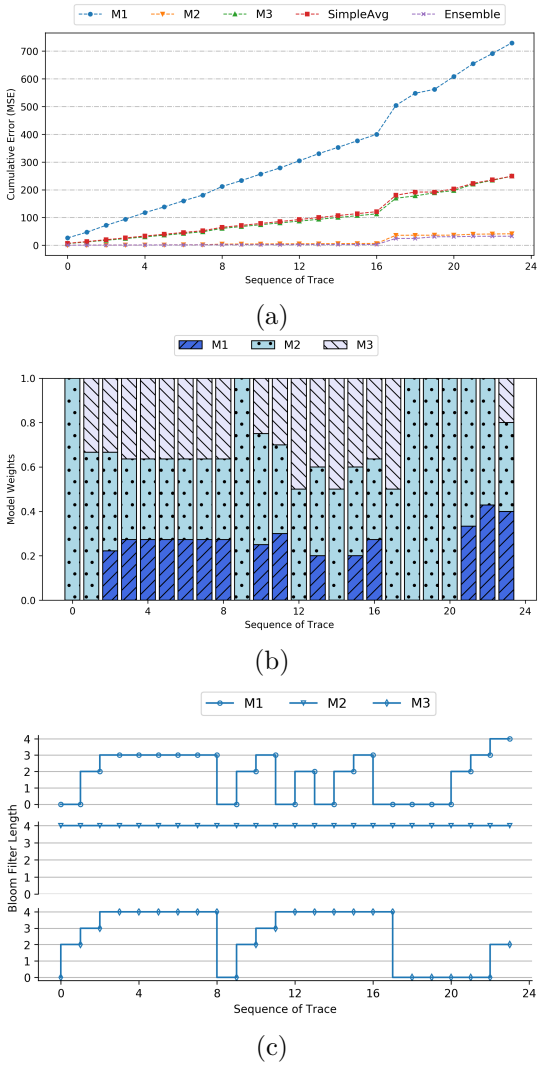


Figure 39: (a) Cumulative error rate of the ensemble model vs. the individual models and simple average method for indoor localization dataset. (b) Change of the individual models' weights over time. (c) Change of the length of the matches of the models' outputs to the Bloom filters.

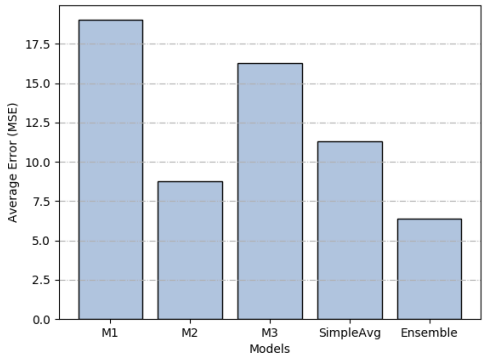


Figure 40: Average error rate for the ensemble model vs. the individual models and simple average method for indoor localization dataset.

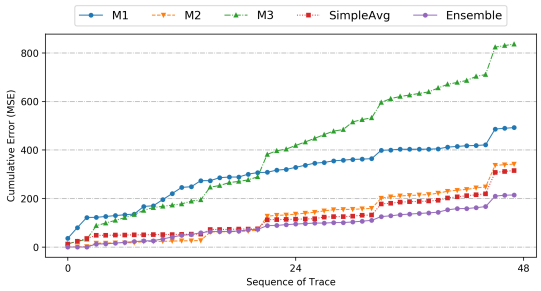


Figure 41: Long-term cumulative error rate of the ensemble model vs. the individual models and simple average method for indoor localization dataset.

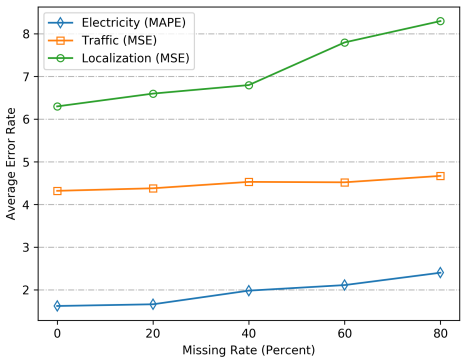


Figure 42: The effect of data missing rate on the accuracy of the proposed spatio-temporal ensemble method for three experimental datasets.

The Effect of Missing Data on Ensemble Performance and Network Overhead

In this section, we measure the effect of missing data on the accuracy of the ensemble model. For this evaluation, we trained the individual models on part of the data. From the whole dataset, 70% of records were used for training. Figure 42 demonstrates the accuracy of the ensemble model when the individual models are trained by data with different data missing rates (e.g., with 80% missing rate, only 20% of the training set is used for model training). Results assert that higher data missing rates result in higher prediction error rates. However, as the results indicate, when the training dataset is large, high data missing rates do not contribute to significant increases in error rates, as the remaining available data is still large enough to produce the desired accuracy given the learning capacity of the employed machine learning model. This is highlighted in the electricity and vehicular traffic datasets. In real world distributed deployments where large training datasets need to be transferred over the network, depending on the size of the training dataset, the required error rate and the learning capacity of the learning model, it might be imperative to consider the transfer of smaller datasets for model training. Specifically, the increases of error rate when the training data missing rate varied from 0% (whole data) to 80% are 0.78 (MAPE), 0.33 (MSE), and 2.0 (MSE) for electricity, traffic and localization datasets, respectively. However, the extent to which the data missing rate can be tolerated is determined by the model's learning capacity; However, this is beyond the scope of this research and can be investigated in future research.

Discussions and Insights

Our three experiments show that the spatio-temporal ensemble model is versatile and it can be employed in a variety of scenarios to improve the accuracy of the model. The algorithm adapts the weights of the learning models in the ensemble based on the spatio-temporal patterns embedded in the

Table 16: Summary of the experiments using spatio-temporal ensemble learning.

Experiment	Models' Characteristics		Spatio-Temporal Ensemble's Characteristics		Benefit of the Proposed Approach
	Relative Performance	Overall performance	Matching Bloom filter	Weighting change	
Electricity	Change a lot over time	All models perform close to each other.	Many changes	Many changes	High
Traffic	Little change (M1 is often superiour)	All models perform relatively close to each other.	Few changes	Few changes	Low
Localization	No change (M2 is always superior)	One model is outperforming the other models.	Few Changes	Few changes	High

training data. Table 16 summarizes the insights obtained from our experiments and the benefit of the proposed ensemble method in various application scenarios. Based on our experiments, the performance of the proposed method varies depending on the application scenario. Overall, we classify application scenarios to two broad categories; namely, scenarios in which the relative performance of the learning models changes significantly over time and scenarios in which the relative performance of these models does not change significantly over time.

Models' Relative Performance Changes Significantly

For such scenarios, as in case of the electricity demand dataset, there are significant changes in the weights of the underlying models. The ensemble method contributes to better performance in terms of error rate (Cf. Figure 35), compared to the individual and simple average models.

Models' Relative Performance Does not Change Significantly

For such scenarios, as in the case of the vehicular traffic and localization datasets, exploiting the saptio-temporal patterns embedded in the training data does not lead to significant performance improvements in short-term intervals. While our proposed approach does not yield benefits in short-term intervals, it should be noted here that in many application scenarios the relative performance changes between the models are observed in long-term intervals.

Conclusion and Future Work

In this paper, we proposed a novel ensemble method that benefits from the spatio-temporal patterns embedded in the training data to adjust the weight of the individual models of the ensemble. Our idea relies on the use of Bloom filters to store the spatio-temporal patterns embedded in the training data. To devise an adaptive algorithm, we formulated a multi-objective programing problem that finds the longest match of spatio-temporal patterns that is stored in the Bloom filters. The proposed approach leads to improved accuracy of the predictions that ranges from 5% to 33% compared to the individual and simple average models. Moreover, the proposed approach contributes to reduced network traffic to transfer the training data without a significant negative impact on prediction accuracy. Our evaluation on three

real-world datasets including power consumption forecasting, vehicular traffic volume prediction, and indoor localization asserts the efficiency of the proposed model in terms of prediction accuracy. The results show that matches between the output of a model and the spatio-temporal patterns embedded in the training data enable learning models to be weighted intelligently to contribute to better predictions within an ensemble of models.

Utilizing the output of different training models (e.g., a mix of LSTM, DBN, CNN, etc.) in the ensemble method is a possible future work. Such research effort can lead to a better understanding of the effect of the spatio-temporal patterns embedded in the predictions of these models on the accuracy of ensemble methods. Also, the proposed approach can be used in the context of mobile learning agents that temporarily contribute to an ensemble model (e.g., smart connected vehicles temporarily participate in an ensemble model that includes Road Side Units).

Chapter 5

Path Planning in support of Smart Mobility Applications Using GANs

Introduction

The development of smart services on top of the Internet of Things (IoT) infrastructure is a movement that aims to introduce new efficiency and comfort applications to our societies; therefore, contributing to livable communities through the internetworking of smart objects [266]. However, the development of such smart services is not straightforward and entails many challenges. One main challenge is the manual labeling of datasets to train such systems; an approach that is very expensive and time-consuming. As reported in [298], collecting fingerprinting data for a large office for indoor localization costs up to $10,000. One alternative solution is crowd-sourcing data. Crowd-sourcing allows end-users to participate in data gathering and annotation tasks through their mobile devices.

In order to develop smart services, Deep Learning (DL) techniques are widely used in IoT and smart city domains for a variety of applications such as Intelligent Transportation Systems (ITS), smart agriculture, smart homes, etc. [265]. Among DL techniques, Generative Adversarial Networks (GANs) [45] are mostly used for vision-based tasks but have been rarely used in the IoT domain.

Smart mobility refers to the utilization of information and communications technologies (ICT) augmented with artificial intelligence for the optimization and efficient distribution of traffic flows and transportation services [299]. Smart Mobility is one of the prominent functionalities of a smart city as it can improve the quality of life of almost all the citizens. Path planning plays a significant role for many smart mobility applications. For example, path planning is a major component for emergency evacuation situations in a building or even a city scale. If evacuees have adequate information about the

exit ways, smart mobility applications can prevent potential fatalities due to severe crowd congestions or choosing paths that lead to dangerous areas. Numerous approaches have been proposed for intelligent path planning; however, deploying feasible techniques remains a challenge [300].

The motivations behind this work are as follows:

- Development of indoor path planning and wayfinding technologies for use by disabled commuters through smartphone applications is recommended to improve their mobility and quality of life [301]. These applications can be augmented with multimodal information access to satisfy the needs of different visual and hearing disabilities.

- In emergency situations and building evacuation scenarios, individualized path planning can be devised in real-time for different groups of people to satisfy their needs. For example, people using wheelchairs will receive a recommended path to exit that differs from the one recommended to people in the same area that do not have physical disabilities.

- Gathering data to train a machine learning system is a time-consuming and expensive process. Crowd-sourced data is a potential alternative to gather the desired data in shorter time and at a lower cost. Since people tend to choose the shortest paths for their navigations between two points, gathering such crowd-sourced data can help to train a machine learning system based on the historical traces of the navigation behavior of other users.

The goal of this research is to explore the use of GANs to generate individualized paths that meet the users' needs. To this end, the system accurately determines the location of a user in unfamiliar environments and helps them to navigate to their desired destinations. The system uses a machine learning model that is trained using crowd-sourced data. The system can learn the correct paths through the recorded trajectories of other users, so that in scenarios where no data is recorded between a given source-destination pair, the system can generate a path that is most likely to be a valid and correct path. This way we utilize the experience of other users instead of wasting the collected data. Our method is inspired by the recent successes and advancements in data-driven methods on computer vision problems using generative adversarial networks [302, 303]. Figure 43 illustrates the overall architecture of the system. A GAN model is trained on an existing dataset of annotated trajectories. When a user asks for a path to a specific destination, GAN generates the desired path and reports it to the user. The user then follows the generated path and sends a feedback to the system about the quality of the generated path.

To assess the effectiveness of the proposed approach, we applied it to an indoor path planning application. The indoor environment is equipped with IoT devices deployed in public access areas to help in

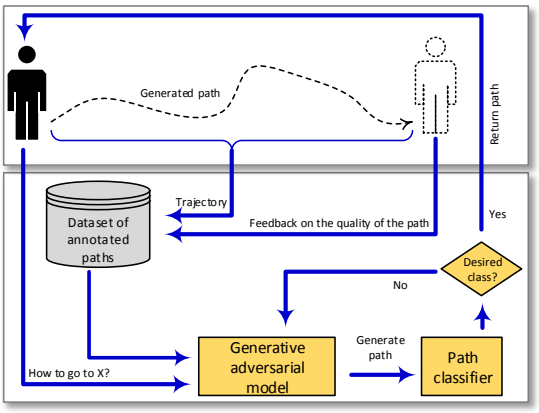


Figure 43: The overall architecture of the proposed system.

the localization process of the users. Our results show that the generated paths do not deviate from the safe boundaries (i.e., public access areas) and accurately lead to the desired destinations.

The main contributions of this paper are:

1. We propose an approach for path planning that exploits the availability of users' trajectory data and their associated qualitative feedback. At the core of our proposed approach, GAN generates the viable paths that lead to a specified destination.

2. Our approach does not depend on the availability of pre-engineered training dataset in order for the system to work in real environments. The system can generalize the best solutions by learning from the trajectories of the users over the time. Consequently, the results converge to the optimal ones over time.

The rest of the paper is organized as follows. Section 5 presents several related research studies. Section 5 provides the details of the proposed system followed by section 5 that brings the proposed approach to a wayfinding application for the visually impaired. Section 5 presents our experimental results. Finally, section 5 concludes the paper and introduces several directions for potential extensions.

Related Work

Many indoor localization and navigation solutions have been proposed based on a wide range of technologies, such as vision, visual light communications (VLC), infrared, sound, WiFi, RFID, and Bluetooth Low Energy (BLE). The cost of the underlying technologies and devices is an important factor to consider when deploying location-based services. Among the aforementioned technologies, BLE is a low-cost technology that has been utilized in recent research works as well as commercial applications. To deploy indoor location-aware systems for environments that are not equipped with wireless networks,

the use of the BLE technology (e.g., iBeacon devices) is advantageous. Beyond the low-cost of iBeacons, these devices also can be deployed quickly and easily without changing or even tapping into the building's electrical and communications infrastructure.

One of the most promising approaches to implement location-aware services is based on Relative Signal Strength (RSS) fingerprinting. Such approach needs to address some challenges including fingerprint annotation and device diversity [252]. The use of fingerprint-based approaches to identify an indoor position has been studied well in the past decade. A variety of machine learning approaches have been utilized in this context including Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Bayesian-based filtering, transfer learning, and Deep Neural Networks (DNNs) [94].

Li *et al.* [304] proposed an indoor navigation system based on off-the-shelf smartphone sensors and magnetic features. They used several approaches to enhance the accuracy of the system including multi-dimensional dynamic time warping, weighted k-nearest neighbors, and exploiting magnetic gradient fingerprints. They also mitigated the impact of magnetic matching mismatches to reduce the position errors. They reported root mean square error between 4.3 m and 5.6 m.

Many wayfinding systems use the well-known A* algorithm or its variants for path planning [305, 306, 307]. For example, the work presented by Apostolopoulos *et al.* [308] proposes an indoor localization and navigation system for the visually impaired using the embedded smartphone sensors, such as accelerometers and compasses. Their method uses the feedback from human users on the availability of landmarks along the provided path. The system provides the users with audio instructions to reach the desired destination. They utilize the A* algorithm to compute the shortest path. The computational time of approaches that utilize A* is high. Moreover, A* based approaches are not good to handle narrow way-paths which are common in indoor environments. They also heavily rely on the availability of full prior knowledge of the locations and environment.

Chen *et al.* [309] propose a path planning system for emergency guidance in indoor environments based on IoT technologies. They use the statistical properties of mobility of groups of people to provide an individualized path for each group. In this work, a graph of the corridors, doors, and exits is optimized to minimize the total evacuation time for all groups. They implemented their work by utilizing the iBeacon technology and smartphones.

GANs have been used widely for visual data. The most related work to path planning is reported by Hirose *et al.* [310] in which GANs are used to classify images as safe to go to or not for robot navigation purposes. In that work, authors used the observed scene by a robot's camera and classified it as "GO" or "NO GO".

Compared to the aforementioned works, our proposed approach uses a generative deep learning technique over the crowd-sourced trajectory data to suggest a viable path between a given source-

destination pair where no pre-recorded path between the given source-destination pair is found in the training dataset. While most of the reported works on GANs rely on vision tasks, our approach is based on non-visual data for path planning which has a high potential for IoT applications as detailed later in Section 5.

Proposed Method

Our proposed path planning approach is based on GANs. GANs have been shown to perform impressively in computer vision applications. Especially, it can generate dynamic pictures from static ones, predicting several seconds of a movie, arithmetic on face [302], and text to image synthesis [303]. These accomplishments motivate us to consider the use of GANs to generate reliable and correct paths in support of smart mobility applications.

Background on GAN

Generative adversarial networks [45] is a new model of DNNs that has been introduced in 2014. The original model consists of two neural networks that compete against each other. One network is a generator and the other is a discriminator (Cf. Fig. 44). The generator network tries to generate samples that resemble the real data such that the discriminator cannot tell whether it is a real sample or a fake one.

In a GAN framework, a generator model G and a discriminator model D are trained simultaneously. The generative model infers the data distribution p_g over data space x. It also includes a defined prior input noise variables $p_z(z)$. The generator is then represented by $G(z;\theta_g)$ that maps z to the data space. G is a differentiable function represented by a DNN with parameters θ_g. The discriminative model D computes the probability that a given sample is coming from the training data or just generated by the generator. The discriminator D is represented by another DNN as $D(x;\theta_d)$ in which its single scalar output specifies the probability that x is available in the real data or is generated using p_g.

In the training process of GAN, D is optimized to assign the correct label of real/fake to both training examples and samples from G. At the same time, G is optimized to minimize $\log(1 - D(G(z)))$. To formally state it, D and G play the following two-player minimax game with value function $V(D, G)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data(x)}}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]. \tag{1}$$

The weights of discriminator network are updated by:

$$\Delta_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} [\log D(x_i) + log(1 - D(G(z_i)))] \tag{2}$$

while the generator's weights are updated by

$$\Delta_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} \log (1 - D(G(z_i))) \tag{3}$$

Using GANs for Path Planning

In our work, the map of the environment is specified by a matrix $M = [a_{ij}]; 0 \leq i \leq I; 0 \leq j \leq J$, where I and J are the maximum indexes of vertical and horizontal positions, respectively. a_{ij} equals 1 represents that (i, j) is in a public access area. Otherwise, a 0 indicates that it is not a public access area; hence, no path should go through it. Each path in the training data is also represented by a matrix P as $P = [t_{ij}]$ that has the same dimensions as M. In this matrix, t_{ij} equals 1 indicates that position (i, j) is included in the path. Otherwise, 0 indicates that it is not included in the path.

In path planning services, GANs can be used to generate a path based on the previously collected user trajectories (i.e., real-world data) that lead the users to earlier destinations. Intuitively, the generator network tries to generate a path between a given source-destination pair that seems more realistic rather than a generated fake path. On the other hand, the discriminator network infers the probability of that path being real or not – e.g., a real path should be gradually leading to the destination. After training the GAN, during the inference phase, the output of the generator network is fed to a classifier network (Cf. Fig. 44) to identify the label of the path (i.e., the source and destination of the generated path). Figure 45 provides a conceptual illustration of using GAN in support of wayfinding.

In our approach, the generator G is a series of fully-connected layers that transforms a noise vector z into a potential path (i.e., in the form of a layout frame that specifies the path). Likewise, the discriminator D is a series of fully connected layers that accepts a layout frame and specifies its probability of being a fake or a real sample. The implementation of our model is modified based on an implementation of GAN [311] using Keras.

Algorithm 3 illustrates the overall process for finding a path by generating a viable path and then classifying it. The algorithm first trains the generator and discriminator of the GAN model simultaneously. A classifier is also trained on the same dataset to classify the generated paths. The GAN, then generates a path until the classifier determines that the generated path complies with the user's request. The alternative approaches for classifying GAN's generated data are presented in [312, 313, 314] where a discriminator network is extended to classify its inputs that come from real or generated data.

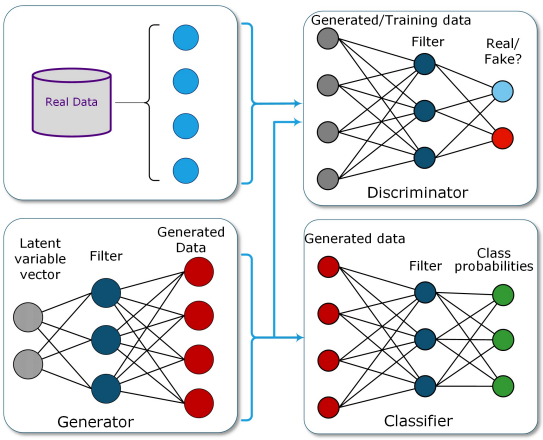


Figure 44: Our GAN based path planning approach.

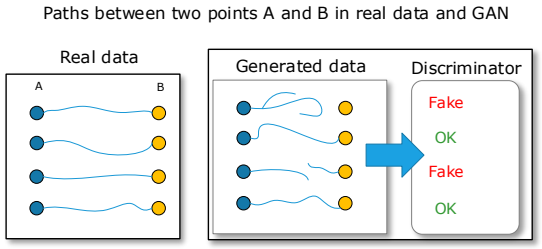


Figure 45: The conceptual diagram of using GAN in wayfinding.

noend 3 Path Planning Algorithm using GAN

Input: (*Source*,*Destination*), Training dataset
`/* Training the GAN in I iterations*/`
for $k = 1$ to I **do**
  Draw m noise samples from $p_g(z)$
  Draw m real examples from $p_d(x)$
  Update the discriminator D based on (2)
  Draw m noise samples from $p_g(z)$
  Update the generator G based on (3)
end for
`/* Finding the path */`
Train a classifier C on training data
$y \leftarrow (Source, Destination)$
$\hat{y} \leftarrow None$
while $\hat{y} == None$ **do**
  Generate a path h using the trained GAN
  $\hat{y} \leftarrow C(h)$
  if $\hat{y} == y$ **then**
    return h
  else
    $\hat{y} \leftarrow None$

Table 17: Comparing the coverage range of different communication technologies.

Technology	Range	Topology	Pros	Cons
RFID	10 cm $\sim$ 2 m	Peer-to-peer	Easy installment	A dense deployment causes inference
NFC	$\sim$ 10 cm	Peer-to-peer	Cheap tags	Not widely available in mobile phones
BLE	$\sim$ 30 m	Star	Low power; Easy set up	Need fixed reference points
ZigBee	10 $\sim$ 20 m	Star/mesh	Low power consumption	Requires pairing devices
Z-Wave	$\sim$ 30 m	Mesh	High level of interoperability	More expensive devices
UWB	$\sim$ 100 m	Star	Significant obstacles penetration capacity	Interference with nearby UWB systems
WiFi	$\sim$ 100 m	Star/mesh	No need for extra equipment	Not available anywhere

Use Case: Wayfinding for Disabled and Visually Impaired People

There are more than four million Americans with vision impairments [301]. Assisting the Blind and Visually Impaired (BVI) to navigate through indoor and outdoor environments can contribute to having this important segment of the society live more independently. To provide reliable and accurate guidance to BVI commuters, a precise wayfinding mechanism is needed. There are many attempts in the recent literature to tackle this problem using a variety of technologies. However, these attempts impose new challenges for the users as their accuracy may not be fine-grained. Existing research studies that investigate wayfinding approaches for the BVI (regardless of the underlying technology, such as RFID, sensor networks, or vision) utilize training datasets for the system [298]. For large deployments, providing such training dataset that needs to be labeled (i.e., class labels) is not always feasible. Instead, exploiting the availability of crowd-sourcing data that users create and manually label, is a solution that expedites the development of such services. In this research, we utilize the experiences of other users and their trajectories to improve the accuracy of the path planning system.

Experimental Setup

We deploy a grid of 13 iBeacons [315] to experiment with our proposed path planning and wayfinding approach in a campus library setting. Our experiments cover the first floor of Waldo Library, Western Michigan University (WMU). In our work, we use the iBeacons' Received Signal Strength Indicator (RSSI) values to serve as the raw source of input data to identify users' locations. iBeacons are installed on the public access areas, so that we experiment with our proposed path planning and wayfinding approach in the coverage area of the iBeacons. Smartphones are also utilized to sense the iBeacons' signals and to compute the current position of the user relative to the known iBeacon positions.

iBeacons transmit short-range BLE radio waves that consume tiny power. iBeacons can operate for several months or even years. Each iBeacon covers up to ten times the distance of the classic Bluetooth while its transmission is 15 times faster. iBeacons can be configured to transmit at a power level that ranges from 0.01 mW to 10 mW. Moreover, the BLE standard has been developed and adopted by many smartphone makers and is now available on most smartphone models. These characteristics make BLE a good and practical choice for IoT applications that require short-range communications. Table 17

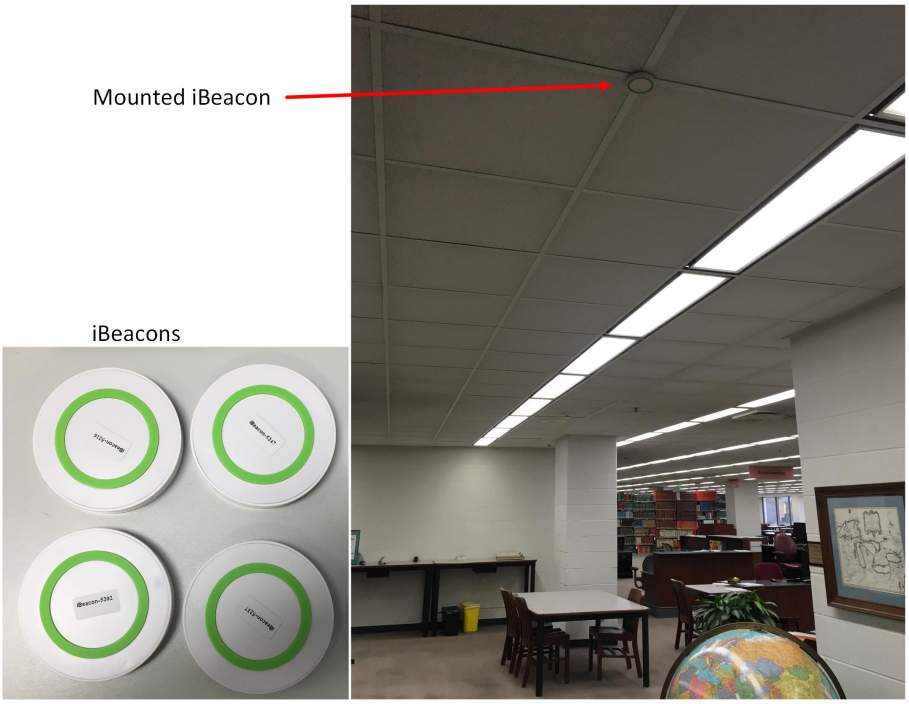


Figure 46: iBeacon setup in the environment.

summarizes and compares several communications technologies and their advantages and disadvantages.

In our implementation and experiments with an indoor location-aware application, we chose the iBeacon technology due to its low price and ease of deployment. These iBeacons can be stuck to a wall or ceiling without tapping into the buildings' infrastructure. iBeacon is a standard technology based on Bluetooth low energy or Bluetooth Smart that was proposed by Apple Inc. iBeacon devices are identifiable and addressable through three separate fields; namely, UUID, major, and minor.

Figure 46 shows a scene of a mounted iBeacon in our area of experiments. iBeacons cannot trigger a location-based action on their own since they are half-duplex transmitting devices that only send their probe signals in a specific interval (usually every one second). So, there is a need for a receiving device that can detect the iBeacons' signals. Smartphones nowadays are equipped with the BLE technology and are ideal for act as the receivers since they are pervasive.

Our GAN Configuration

We implemented our GAN based approach with the following structure. The generator is a feed-forward DNN with the input layer composed of 100 neurons, and four hidden layers composed of 256, 512, 1024, 247 neurons, respectively with a *tanh* activation function. The last layer is composed of 247 (i.e., $19 * 13$) neurons which is the dimensionality of the path frames. The discriminator is also a feed-forward DNN with an input layer composed of 247 neurons and three hidden layers composed of 512, 256, 1 neurons, respectively with a *sigmoid* activation function. The loss function for the model is

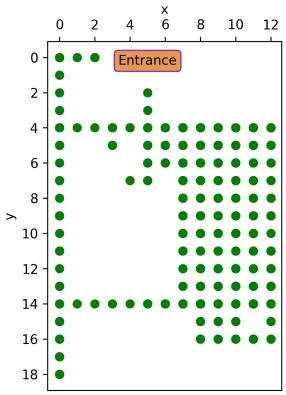


Figure 47: Accessible areas for positioning and navigation where the empty spaces represent rooms and other facilities that are not used for public.

binary crossentropy and the optimization method is *Adam*. The path classifier is also a neural network with an input layer composed of 247 neurons and two hidden layers that are composed of 256 and 6 neurons, respectively. We used the *softmax* as the activation function for that last layer to produce the probability of the 6 classes.

Evaluation and Results

In this section, we discuss the datasets used to conduct this research and the evaluation of the proposed system according to the use case in section 5. The evaluations have been done on a computer with 2.3 GHz Intel Core i5 CPU and 8 GB RAM.

Datasets

Since, our research involves two primary components; namely, localization, and wayfinding, we performed evaluations focusing on the accuracy of these two components on two datasets (i.e., localization dataset [296] and a path planning dataset [316]). We deployed the physical infrastructure needed on the first floor of Waldo Library at Western Michigan University to collect these two datasets. Figure 47 shows the active area covered by the deployed iBeacons and is accessible to the public. All our localization and navigation experiments are conducted in this area. Also, the localization and path planning datasets are collected from this experimental testbed.

Localization Dataset. The localization dataset contains 1420 labeled samples of which we used 70% for training and the remaining 30% for testing. Each sample consists of RSSI values of the 13 iBeacons deployed on this testbed. Each of which is presented as a feature. In addition, two positioning features representing the `x` and `y` coordinates of the current location serve as predictable features.

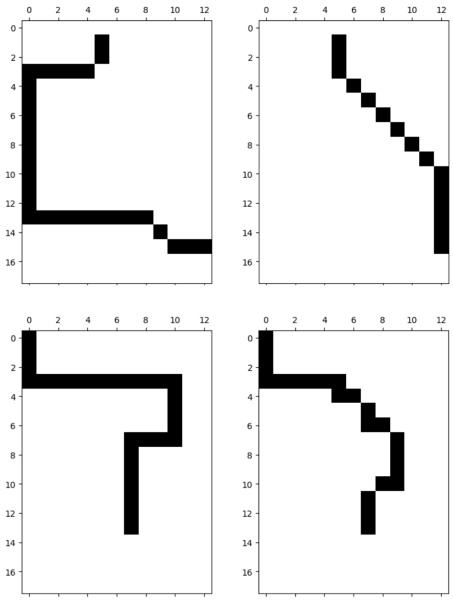


Figure 48: Samples of paths in training dataset for class 2 (top) from (2, 5) to (16, 12) and class 0 (bottom) from (0, 0) to (14, 7).

Table 18: Coding the paths labels.

Class	**Source**		**Destination**	
	x1	**y1**	**x2**	**y2**
0	0	0	14	7
1	0	0	16	12
2	2	5	16	12
3	2	5	16	8
4	2	5	18	0
5	4	0	10	10

Path Planning Dataset. In the path planning dataset, we have 6 different classes that define the source and destination of a path. Table 18 shows the different classes of the paths and their corresponding source and destination coordinates. Each class has about 52 samples with a total of 313 samples that are stored as CSV files. Each file resembles a frame containing the path in a given indoor environment. A path frame is a 19×13 matrix in which the item at indexes (i, j) indicates that the corresponding position of (i, j) part of the path (value at indexes (i, j) equals 1) or not (value at indexes (i, j) equals 0). Figure 48 depicts some paths from the training dataset.

Localization

The training process of localization is a multi-label classification problem since we need to predict `x` and `y` coordinates. We used the binary relevance approach [317] to perform training and classification of RSSI values into location coordinates. In this approach, for each class feature (two features in our

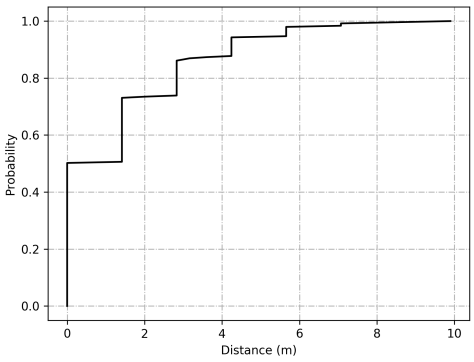


Figure 49: Cumulative density function (CDF) of Euclidean distance error for localization.

problem; namely, `x` and `y` coordinates), a separate model is created and trained on the same training dataset.

Figure 49 illustrates the accuracy of the localization method in terms of the Euclidean distance error. The plot shows that about 50% of the results are an exact match. Also, more than 75% of targets have a distance error of 2 m or less. The average error is 1.5 m. The average prediction time per sample for localization is 0.12 millisecond.

Path Planning

Assessing the accuracy of a generative model is challenging since there is no standard metric that quantifies to what degree the output is realistic. The metric also needs to specify how novel the output is compared to the real data to avoid having a model that just remembers the input training data and provides it again as output [318]. The research work on generative models in the context of vision-based tasks where images are generated utilize human judgments to assess the quality of output samples (e.g., mean opinion score [319]).

We evaluated the path classifier separately, since we need ground truth values to examine how good it can identify the classes of unseen paths. To evaluate the accuracy of the path classifier, we split the path planning dataset into 70% for training and 30% for the testing. The average error rate of the classifier is 1.06%. Figure 50 demonstrates the error rate of the path classifier when we set different batch sizes for training. We get the best accuracy when the batch size is 20 or less. That is because with smaller batch sizes, Stochastic Gradient Descent (SGD) based methods use smaller steps in the training process to find the optimum, while larger batch sizes indicate larger steps. So, the larger steps may skip some local optima and degrade the quality of the model [320].

Figure 51 shows several paths generated using our GAN based approach and their corresponding labels. There are some generated samples with minor noise which can be filtered in a production setting.

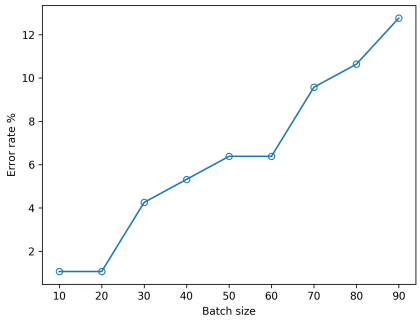


Figure 50: Error rate of the path classifier when training is performed with different batch sizes.

One interesting evaluation of this work is to see how the generated paths may deviate from the baseline public access area (Cf. figure 47). We assessed this by considering those positions in the generated paths that are outside of the public access areas. This can be formulated as:

$$Score_{deviation}(G) = \frac{|G| - |G \cap B|}{|B|}, \tag{4}$$

where G is a set of positions that represent the generated path and B is the set of the positions for the public access area. This measurement is an indication of the "safety" of the paths. In other words, it points out how much the generated paths are inside the safe boundaries (e.g., do not go to the restricted areas, etc.). Our experiments show that the deviation is minor if any and in the range of 0 to 0.08%. Figure 52 shows the deviation percentage of the generated paths based on the model that is trained in a different number of epochs. As we train the model with a larger number of epochs beyond 1000, the results do not exhibit any deviation.

We also conducted a subjective evaluation using the Mean Opinion Score (MOS) to assess the quality of the generated paths. For this evaluation, 60 generated paths were rated with a score that ranges from 1 (bad) to 5 (very good) by three raters based on their clarity to be followed for navigation purposes. The average MOS for all the scored paths is 4.46 (89%) with a standard deviation of 0.74. Table 19 summarizes these results. These results demonstrate the potential of using the generated paths for reliable wayfinding and navigation tasks.

The training time for our GAN based approach is 14 minutes for 15000 epochs and 48 minutes for $50K$ epochs. The run time for generating a path instance by GAN took 0.21 millisecond and the run time for classifying a generated path instance is 0.14 millisecond.

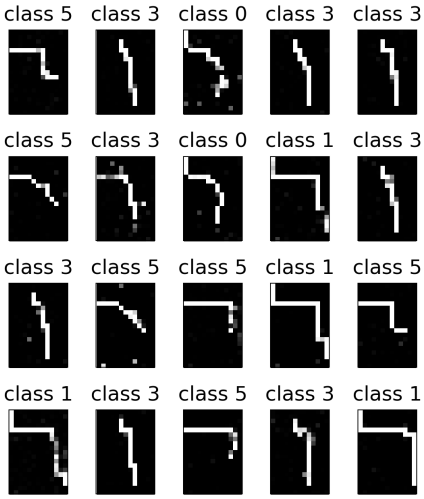


Figure 51: Samples of randomly generated paths and their classification.

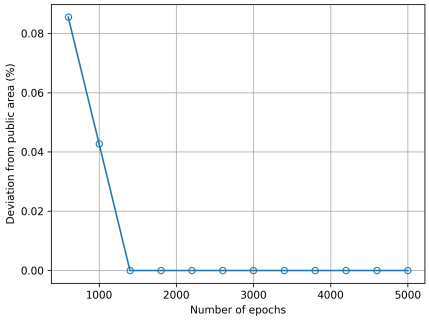


Figure 52: Deviation from the public access area for the generated paths when the model is trained using different number of epochs.

Table 19: Mean opinion score for the generated paths.

Rater#	**MOS**	**standard deviation**
1	4.6	0.66
2	4.7	0.45
3	4.1	0.88
Avg	**4.46**	**0.74**

Conclusion and Future Work

This paper presented an approach for path planning based on GANs and crowd-sourced data. In our approach, a GAN is used to generate paths to a desired destination. We experimented with the proposed approach in a wayfinding application in an IoT-enabled environment. Our evaluations show that the generated paths are more than 99.9% safe and reliable while the path classifier component correctly classifies the paths with an accuracy of up to 99%. Also, the generated paths are of high quality attaining around 89% mean opinion score.

GANs can serve as an efficient technique for generating samples that do not exist in the collected dataset. However, it is hard to adjust their parameters optimally since assessing their performance is hard. In particular, the loss function is not necessarily a good indicator of their goodness. So far, the most trusted method to evaluate their output is to use human judgments [313].

A potential pathway for the future work is to provide tactile feedback (e.g., vibration codes) to direct the user and offer accurate guidance information to ensure that the user follows the planned path. Providing a mechanism to generated paths that minimize or maximize multiple metrics is another direction for future work. Another application where the proposed approach can be utilized is for path planning for autonomous vehicles. In such an application, a vehicle can utilize the path that is generated by a GAN based on the trajectories of other participating vehicles that have shared their GPS encoded paths for training the GAN.

Acknowledgment

This study was partially supported by the USDOT through the Transportation Research Center for Livable Communities (TRCLC), a Tier 1 University Transportation Center at Western Michigan University.

Chapter 6

Conclusion and Future Work

Conclusion

The utilization of Deep Learning and IoT technologies has drawn the attention of researchers in recent years since these two technology trends are expected to improve our lives and societies. However, the road is not straightforward and there are many challenges to overcome. In this dissertation, we address some of these challenges as detailed below.

- We review the characteristics of IoT data and its challenges for DL methods. In specific, we highlight IoT fast and streaming data as well as IoT big data as the two main categories of IoT data generation and their requirements for analytics. We also present several main architectures of DL that is used in the context of IoT applications followed by several open source frameworks for development of DL architectures. Reviewing different applications in various sectors of IoT that have utilized DL is another part of this study in which we identify five foundational services along with twelve application domains. The new paradigm of implementing DL on IoT devices is surveyed and several approaches to achieve it are introduced. DL based on fog and cloud infrastructures to support IoT applications is another part of this research. We also identify the challenges and future research direction in the path of DL for IoT applications.

- We propose a semi-supervised deep reinforcement learning framework as a learning mechanism in support of smart IoT services. The proposed model uses a small set of labeled data along with a larger set of unlabeled ones. The proposed model consists of a deep variational autoencoder network that learns the best policies for taking optimal actions by the agent. As a use case, we experiment with the proposed model in an indoor localization system. Our experimental results illustrate that the proposed semi-supervised deep reinforcement learning model is able to generalize the positioning

policy for configurations where the environment data is a mix of labeled and unlabeled data and achieve better results compared to using a set of only labeled data in a supervised model. The results show an improvement of 23% on the localization accuracy in the proposed semi-supervised deep reinforcement learning model. Also, in terms of gaining rewards, the semi-supervised model outperforms the supervised model by receiving at least 67% more rewards.

- We propose a novel ensemble method that benefits from the spatio-temporal patterns embedded in the training data to adjust the weight of the individual models of the ensemble. Our idea relies on the use of Bloom filters to store the spatio-temporal patterns embedded in the training data. To devise an adaptive algorithm, we formulated a multi-objective programing problem that finds the longest match of spatio-temporal patterns that is stored in the Bloom filters. The proposed approach leads to improved accuracy of the predictions that ranges from 5% to 33% compared to the individual and simple average models. Moreover, the proposed approach contributes to reduced network traffic to transfer the training data without a significant negative impact on prediction accuracy. The results show that matches between the output of a model and the spatio-temporal patterns embedded in the training data enable learning models to be weighted intelligently to contribute to better predictions within an ensemble of models.

- We present an approach for path planning based on GANs and crowd-sourced data. In our approach, a GAN is used to generate paths to a desired destination. We experiment with the proposed approach in a wayfinding application in an IoT-enabled environment. Our evaluations show that the generated paths are more than 99.9% safe and reliable while the path classifier component correctly classifies the paths with an accuracy of up to 99%. Also, the generated paths are of high quality attaining around 89% mean opinion score.

Future Work

In the following paragraphs, we present several potential pathways for future extensions of the research work presented in this study.

- For the semi-supervised DRL method, it would be interesting to find out the minimum amount of the labeled data that is needed in addition to the unlabeled data to reach a desired accuracy. Such insight can help to cut off the expenses of collecting excessive labeled data. Moreover, extending this work such that the semi-supervised DRL model can be trained online will lead to more flexible and evolving prediction models.

- Utilizing the output of different training models (e.g., a mix of LSTM, DBN, CNN, etc.) in the

ensemble method is a possible future work. Such research effort can lead to a better understanding of the effect of the spatio-temporal patterns embedded in the predictions of these models on the accuracy of ensemble methods. Also, the proposed approach can be used in the context of mobile learning agents that temporarily contribute to an ensemble model (e.g., smart connected vehicles temporarily participate in an ensemble model that includes Road Side Units).

- In the path planning framework using GAN, a potential pathway for the future work is to provide tactile feedback (e.g., vibration codes) to direct the user and offer accurate guidance information to ensure that the user follows the planned path. Providing a mechanism to generated paths that minimize or maximize multiple metrics (e.g., finding shortest path) is another direction for future work. Evaluating the proposed approach in large scale deployments and investigate the performance of GAN in such scenarios is another interesting direction for future work. Such scenarios can be used in path planning for autonomous vehicles. In such an application, a vehicle can utilize the path that is generated by a GAN based on the trajectories of other participating vehicles that have shared their GPS encoded paths for training the GAN. Extending the current work to support dynamic training of GAN is another venue for future work to support evolution of the environment.

We also identify several potential directions for future work pertaining to the utilization of DL in IoT domains.

- Investigating efficient ways to utilize mobile big data in conjunction with DL approaches to come up with better smart services for IoT domains.
- Integrating contextual information with IoT data to complement the understanding of the IoT environment.
- Online provisioning of fog or cloud resources to host the stream of data for deployment of fast DL analytics.
- Development of dependable and reliable IoT analytics.
- Providing a range of self-services for the IoT infrastructure including self-configuration, self-optimization, self-healing, and self-load balancing.

References

[1] J. Ginsberg, M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and L. Brilliant. "Detecting influenza epidemics using search engine query data". In: *Nature* 457.7232 (2009), p. 1012.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. "Internet of Things: A survey on enabling technologies, protocols, and applications". In: *IEEE Communications Surveys & Tutorials* 17.4 (2015), pp. 2347–2376.

[3] J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson, and A. Marrs. *Disruptive technologies: Advances that will transform life, business, and the global economy*. Vol. 180. McKinsey Global Institute San Francisco, CA, 2013.

[4] E. Brynjolfsson and T. Mitchell. "What can machine learning do? Workforce implications". In: *Science* 358.6370 (2017), pp. 1530–1534.

[5] K. Panetta. *Gartner's Top 10 Strategic Technology Trends for 2017*. 2016. URL: http://www.gartner.com/smarterwithgartner/gartners-top-10-technology-trends-2017/ (visited on 05/09/2017).

[6] M. Mohammadi and A. Al-Fuqaha. "Enabling Cognitive Smart Cities Using Big Data and Machine Learning: Approaches and Challenges". In: *IEEE Communications Magazine* 56.2 (2018), pp. 94–101.

[7] M. Chen, S. Mao, Y. Zhang, and V. C. Leung. *Big data: related technologies, challenges and future prospects*. Springer, 2014.

[8] X.-W. Chen and X. Lin. "Big data deep learning: challenges and perspectives". In: *IEEE Access* 2 (2014), pp. 514–525.

[9] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. Mccauley, M Franklin, S. Shenker, and I. Stoica. "Fast and interactive analytics over Hadoop data with Spark". In: *USENIX Login* 37.4 (2012), pp. 45–51.

[10] C. Engle, A. Lupher, R. Xin, M. Zaharia, M. J. Franklin, S. Shenker, and I. Stoica. "Shark: fast data analysis using coarse-grained distributed memory". In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM. 2012, pp. 689–692.

[11] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

[12] C.-W. Tsai, C.-F. Lai, M.-C. Chiang, L. T. Yang, et al. "Data mining for Internet of Things: A survey." In: *IEEE Communications Surveys and Tutorials* 16.1 (2014), pp. 77–97.

[13] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. "Context aware computing for the internet of things: A survey". In: *IEEE Communications Surveys & Tutorials* 16.1 (2014), pp. 414–454.

[14] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan. "Machine learning in wireless sensor networks: Algorithms, strategies, and applications". In: *IEEE Communications Surveys & Tutorials* 16.4 (2014), pp. 1996–2018.

[15] Z. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani. "State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems". In: *IEEE Communications Surveys Tutorials* PP.99 (2017). ISSN: 1553-877X.

[16] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng. "A survey of machine learning for big data processing". In: *EURASIP Journal on Advances in Signal Processing* 2016.1 (2016), p. 67.

[17] F. Alam, R. Mehmood, I. Katib, N. N. Albogami, and A. Albeshri. "Data fusion and IoT for smart ubiquitous environments: A survey". In: *IEEE Access* 5 (2017), pp. 9533–9554.

[18] B. Li, Y. Diao, and P. Shenoy. "Supporting scalable analytics with latency constraints". In: *Proceedings of the VLDB Endowment* 8.11 (2015), pp. 1166–1177.

[19] M. Hilbert. "Big data for development: a review of promises and challenges". In: *Development Policy Review* 34.1 (2016), pp. 135–174.

[20] W. Fan and A. Bifet. "Mining big data: current status, and forecast to the future". In: *ACM sIGKDD Explorations Newsletter* 14.2 (2013), pp. 1–5.

[21] H. Hu, Y. Wen, T.-S. Chua, and X. Li. "Toward scalable systems for big data analytics: A technology tutorial". In: *IEEE Access* 2 (2014), pp. 652–687.

[22] Y. Demchenko, P. Grosso, C. De Laat, and P. Membrey. "Addressing big data issues in scientific data infrastructure". In: *Collaboration Technologies and Systems (CTS), 2013 International Conference on*. IEEE. 2013, pp. 48–55.

[23] M. Strohbach, H. Ziekow, V. Gazis, and N. Akiva. “Towards a big data analytics framework for IoT and smart city applications”. In: *Modeling and processing for next-generation big-data technologies*. Springer, 2015, pp. 257–282.

[24] G. E. Hinton and R. R. Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *Science* 313.5786 (2006), pp. 504–507.

[25] D. Svozil, V. Kvasnicka, and J. Pospichal. “Introduction to multi-layer feed-forward neural networks”. In: *Chemometrics and intelligent laboratory systems* 39.1 (1997), pp. 43–62.

[26] J. Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural networks* 61 (2015), pp. 85–117.

[27] X. Glorot, A. Bordes, and Y. Bengio. “Deep sparse rectifier neural networks”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 2011, pp. 315–323.

[28] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *arXiv preprint arXiv:1207.0580* (2012).

[29] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. “On the importance of initialization and momentum in deep learning”. In: *International conference on machine learning*. 2013, pp. 1139–1147.

[30] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[31] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[32] T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. Ranzato. “Learning longer memory in recurrent neural networks”. In: *arXiv preprint arXiv:1412.7753v2 [cs.NE]* (2014).

[33] L. Deng. “A tutorial survey of architectures, algorithms, and applications for deep learning”. In: *APSIPA Transactions on Signal and Information Processing* 3 (2014), pp. 1–29.

[34] L. Bottou. “Large-scale machine learning with stochastic gradient descent”. In: *Proceedings of COMPSTAT’2010*. Springer, 2010, pp. 177–186.

[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), p. 533.

[36] Y. Chauvin and D. E. Rumelhart. *Backpropagation: theory, architectures, and applications*. Psychology Press, 1995.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[38] P. J. Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[39] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. "How to construct deep recurrent neural networks". In: *arXiv preprint arXiv:1312.6026v5 [cs.NE]* (2013).

[40] M. Hermans and B. Schrauwen. "Training and analysing deep recurrent neural networks". In: *Advances in neural information processing systems*. 2013, pp. 190–198.

[41] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555v1 [cs.NE]* (2014).

[42] P. Baldi. "Autoencoders, unsupervised learning, and deep architectures." In: *ICML unsupervised and transfer learning* 27.37-50 (2012), p. 1.

[43] C. Doersch. "Tutorial on variational autoencoders". In: *arXiv preprint arXiv:1606.05908v2 [stat.ML]* (2016).

[44] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. "Semi-supervised learning with deep generative models". In: *Advances in Neural Information Processing Systems*. 2014, pp. 3581–3589.

[45] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial nets". In: *Advances in Neural Information Processing Systems*. 2014, pp. 2672–2680.

[46] B. Dai, S. Fidler, R. Urtasun, and D. Lin. "Towards Diverse and Natural Image Descriptions via a Conditional GAN". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2970–2979.

[47] C. Metz and K. Collins. *How an A.I. 'Cat-and-Mouse Game' Generates Believable Fake Photos*. (Accessed on 2018-02-09). URL: https://www.nytimes.com/interactive/2018/01/02/technology/ai-generated-photos.html.

[48] Y. Bengio et al. "Learning deep architectures for AI". In: *Foundations and trends® in Machine Learning* 2.1 (2009), pp. 1–127.

[49] H. Valpola. "From neural PCA to deep unsupervised learning". In: *Advances in Independent Component Analysis and Learning Machines* (2015), pp. 143–171.

[50] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. "Semi-supervised learning with ladder networks". In: *Advances in Neural Information Processing Systems*. 2015, pp. 3546–3554.

[51] Y. LeCun, C. Cortes, and C. J. Burges. *The MNIST Database of handwritten digits.* (Accessed on 2018-02-01). URL: http://yann.lecun.com/exdb/mnist/.

[52] A. Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: *Master's thesis, Department of Computer Science, University of Toronto* (2009).

[53] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan. "A fast and accurate online sequential learning algorithm for feedforward networks". In: *IEEE Transactions on neural networks* 17.6 (2006), pp. 1411–1423.

[54] Z. Yang, P. Zhang, and L. Chen. "RFID-enabled indoor positioning method for a real-time manufacturing execution system using OS-ELM". In: *Neurocomputing* 174 (2016), pp. 121–133.

[55] H. Zou, H. Jiang, X. Lu, and L. Xie. "An online sequential extreme learning machine approach to WiFi based indoor positioning". In: *Internet of Things (WF-IoT), 2014 IEEE World Forum on.* IEEE. 2014, pp. 111–116.

[56] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks". In: *IEEE transactions on pattern analysis and machine intelligence* 39.6 (2017), pp. 1137–1149.

[57] R. Girshick. "Fast R-CNN". In: *Proceedings of the IEEE international conference on computer vision.* 2015, pp. 1440–1448.

[58] H. Mao, S. Yao, T. Tang, B. Li, J. Yao, and Y. Wang. "Towards Real-Time Object Detection on Embedded Systems". In: *IEEE Transactions on Emerging Topics in Computing* PP.99 (2017), pp. 1–15. ISSN: 2168-6750. DOI: 10.1109/TETC.2016.2593643.

[59] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2016, pp. 779–788.

[60] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602v1 [cs.LG]* (2013).

[61] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J.-S. Oh. "Semi-supervised Deep Reinforcement Learning in Support of IoT and Smart City Services". In: *IEEE Internet of Things Journal* PP.99 (2017), pp. 1–12. DOI: 10.1109/JIOT.2017.2712560.

[62] Y. Bengio et al. "Deep learning of representations for unsupervised and transfer learning." In: *ICML Unsupervised and Transfer Learning* 27 (2012), pp. 17–36.

[63] J. Deng, R. Xia, Z. Zhang, Y. Liu, and B. Schuller. "Introducing shared-hidden-layer autoencoders for transfer learning and their application in acoustic emotion recognition". In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE. 2014, pp. 4818–4822.

[64] P. Wu, S. C. Hoi, H. Xia, P. Zhao, D. Wang, and C. Miao. "Online multimodal deep similarity learning with application to image retrieval". In: *Proceedings of the 21st ACM international conference on Multimedia*. ACM. 2013, pp. 153–162.

[65] P. Jaini, A. Rashwan, H. Zhao, Y. Liu, E. Banijamali, Z. Chen, and P. Poupart. "Online algorithms for sum-product networks with continuous variables". In: *Proceedings of the Eighth International Conference on Probabilistic Graphical Models*. 2016, pp. 228–239.

[66] G. Chen, R. Xu, and S. N. Srihari. "Sequential Labeling with Online Deep Learning: Exploring Model Initialization". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2016, pp. 772–788.

[67] S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah. "Comparative Study of Deep Learning Software Frameworks". In: *arXiv preprint arXiv:1511.06435v3 [cs.LG]* (2016).

[68] A. Candel, V. Parmar, E. LeDell, and A. Arora. *Deep Learning with H2O*. 2015.

[69] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467v2 [cs.DC]* (2016).

[70] R. Collobert, K. Kavukcuoglu, and C. Farabet. "Torch7: A matlab-like environment for machine learning". In: *BigLearn, NIPS Workshop*. EPFL-CONF-192376. 2011.

[71] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio. "Theano: new features and speed improvements". In: *arXiv preprint arXiv:1211.5590v1 [cs.SC]* (2012).

[72] S. Raschka and V. Mirjalili. *Python Machine Learning*. 2nd ed. Birmingham, UK: Packt Publishing, 2017, p. 502.

[73] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. "Caffe: Convolutional architecture for fast feature embedding". In: *Proceedings of the 22nd ACM international conference on Multimedia*. ACM. 2014, pp. 675–678.

[74] O. M. Parkhi, A. Vedaldi, A. Zisserman, et al. "Deep Face Recognition". In: *British Machine Vision Conference*. Vol. 1. 3. 2015, p. 6.

[75] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, J.-H. Rick Chang, et al. “Going Deeper With Convolutions”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[76] S. Shi, Q. Wang, P. Xu, and X. Chu. “Benchmarking state-of-the-art deep learning software tools”. In: *arXiv preprint arXiv:1608.07249v7 [cs.DC]* (2016).

[77] R. Mehmood, F. Alam, N. N. Albogami, I. Katib, A. Albeshri, and S. M. Altowaijri. “UTiLearn: A Personalised Ubiquitous Teaching and Learning System for Smart Societies”. In: *IEEE Access* 5 (2017), pp. 2615–2635.

[78] A. Luckow, M. Cook, N. Ashcraft, E. Weill, E. Djerekarov, and B. Vorster. “Deep learning in the automotive industry: Applications and tools”. In: *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE. 2016, pp. 3759–3768.

[79] X. Ma, H. Yu, Y. Wang, and Y. Wang. “Large-scale transportation network congestion evolution prediction using deep learning theory”. In: *PloS one* 10.3 (2015), e0119044.

[80] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, and F. Kawsar. “An Early Resource Characterization of Deep Learning on Wearables, Smartphones and Internet-of-Things Devices”. In: *Proceedings of the 2015 International Workshop on Internet of Things towards Applications*. ACM. 2015, pp. 7–12.

[81] G. Mittal, K. B. Yagnik, M. Garg, and N. C. Krishnan. “SpotGarbage: smartphone app to detect garbage using deep learning”. In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM. 2016, pp. 940–945.

[82] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, Y. Ma, S. Chen, and P. Hou. “A New Deep Learning-based Food Recognition System for Dietary Assessment on An Edge Computing Service Infrastructure”. In: *IEEE Transactions on Services Computing* (2017).

[83] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic. “Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification”. In: *Computational Intelligence and Neuroscience* 2016 (2016).

[84] Y. Liu, E. Racah, J. Correa, A. Khosrowshahi, D. Lavers, K. Kunkel, M. Wehner, and W. Collins. “Application of Deep Convolutional Neural Networks for Detecting Extreme Weather in Climate Datasets”. In: *Int’l Conf. on Advances in Big Data Analytics* (2016).

[85] S. Tokui, K. Oono, S. Hido, and J. Clayton. “Chainer: a next-generation open source framework for deep learning”. In: *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*. 2015.

[86] Y. Hada-Muranushi, T. Muranushi, A. Asai, D. Okanohara, R. Raymond, G. Watanabe, S. Nemoto, and K. Shibata. “A Deep-Learning Approach for Operation of an Automated Realtime Flare Forecast”. In: *SPACE WEATHER* (2016).

[87] J. A. C. Soto, M. Jentsch, D. Preuveneers, and E. Ilie-Zudor. “CEML: Mixing and moving complex event processing and machine learning to the edge of the network for IoT applications”. In: *Proceedings of the 6th International Conference on the Internet of Things*. ACM. 2016, pp. 103–110.

[88] M. Borkowski, S. Schulte, and C. Hochreiner. “Predicting cloud resource utilization”. In: *Proceedings of the 9th International Conference on Utility and Cloud Computing*. ACM. 2016, pp. 37–42.

[89] J. Gantz and D. Reinsel. “The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east”. In: *IDC iView: IDC Analyze the future* 2007.2012 (2012), pp. 1–16.

[90] H. Larry. *Voice control everywhere: Low-power special-purpose chip could make speech recognition ubiquitous in electronics*. 2017. URL: http://news.mit.edu/2017/low-power-chip-speech-recognition-electronics-0213 (visited on 02/17/2017).

[91] M. Price, J. Glass, and A. Chandrakasan. “A Scalable Speech Recognizer with Deep-Neural-Network Acoustic Models and Voice-Activated Power Gating”. In: *Proceedings of the IEEE ISSCC2017*. 2017.

[92] X. Wang, L. Gao, S. Mao, and S. Pandey. “DeepFi: Deep learning for indoor fingerprinting using channel state information”. In: *2015 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. 2015, pp. 1666–1671.

[93] Y. Gu, Y. Chen, J. Liu, and X. Jiang. “Semi-supervised deep extreme learning machine for Wi-Fi based localization”. In: *Neurocomputing* 166 (2015), pp. 282–293.

[94] W. Zhang, K. Liu, W. Zhang, Y. Zhang, and J. Gu. “Deep Neural Networks for wireless localization in indoor and outdoor environments”. In: *Neurocomputing* 194 (2016), pp. 279–287.

[95] Z. Liu, L. Zhang, Q. Liu, Y. Yin, L. Cheng, and R. Zimmermann. “Fusion of Magnetic and Visual Sensors for Indoor Localization: Infrastructure-Free and More Effective”. In: *IEEE Transactions on Multimedia* 19.4 (2017), pp. 874–888.

[96] M. Becker. “Indoor Positioning Solely Based on User’s Sight”. In: *International Conference on Information Science and Applications*. Springer. 2017, pp. 76–83.

[97] W. Lu, J. Zhang, X. Zhao, J. Wang, and J. Dang. “Multimodal sensory fusion for soccer robot self-localization based on long short-term memory recurrent neural network”. In: *Journal of Ambient Intelligence and Humanized Computing* (2017), pp. 1–9.

[98] A. Toshev and C. Szegedy. “Deeppose: Human pose estimation via deep neural networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1653–1660.

[99] J. Liu, Y. Gu, and S. Kamijo. “Joint Customer Pose and Orientation Estimation Using Deep Neural Network from Surveillance Camera”. In: *Multimedia (ISM), 2016 IEEE International Symposium on*. IEEE. 2016, pp. 216–221.

[100] F. J. Ordóñez and D. Roggen. “Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition”. In: *Sensors* 16.1 (2016), p. 115.

[101] D. Tao, Y. Wen, and R. Hong. “Multi-column Bi-directional Long Short-Term Memory for Mobile Devices-based Human Activity Recognition”. In: *IEEE Internet of Things Journal* (2016).

[102] X. Li, Y. Zhang, I. Marsic, A. Sarcevic, and R. S. Burd. “Deep learning for rfid-based activity recognition”. In: *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems*. ACM. 2016, pp. 164–175.

[103] L. Pigou, A. Van Den Oord, S. Dieleman, M. Van Herreweghe, and J. Dambre. “Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video”. In: *International Journal of Computer Vision* (2015), pp. 1–10.

[104] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. “Recurrent network models for human dynamics”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 4346–4354.

[105] S. E. Kahou, X. Bouthillier, P. Lamblin, C. Gulcehre, V. Michalski, K. Konda, S. Jean, P. Froumenty, Y. Dauphin, N. Boulanger-Lewandowski, et al. “Emonets: Multimodal deep learning approaches for emotion recognition in video”. In: *Journal on Multimodal User Interfaces* 10.2 (2016), pp. 99–111.

[106] N. Neverova, C. Wolf, G. Lacey, L. Fridman, D. Chandra, B. Barbello, and G. Taylor. “Learning human identity from motion patterns”. In: *IEEE Access* 4 (2016), pp. 1810–1820.

[107] Y. He, G. J. Mendis, and J. Wei. “Real-time Detection of False Data Injection Attacks in Smart Grid: A Deep Learning-Based Intelligent Mechanism”. In: *IEEE Transactions on Smart Grid* (2017).

[108] M.-J. Kang and J.-W. Kang. “Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security”. In: *PloS one* 11.6 (2016), e0155781.

[109] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue. “Droid-sec: deep learning in android malware detection”. In: *ACM SIGCOMM Computer Communication Review*. Vol. 44. 4. ACM. 2014, pp. 371–372.

[110] R. Shokri and V. Shmatikov. “Privacy-preserving deep learning”. In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM. 2015, pp. 1310–1321.

[111] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. “Deep learning with differential privacy”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2016, pp. 308–318.

[112] T. J. Hazen. *Microsoft and Liebherr Collaborating on New Generation of Smart Refrigerators*. 2016. URL: http://blogs.technet.microsoft.com/machinelearning/2016/09/02/ (visited on 02/09/2017).

[113] M. Manic, K. Amarasinghe, J. J. Rodriguez-Andina, and C. Rieger. “Intelligent Buildings of the Future: Cyberaware, Deep Learning Powered, and Human Interacting”. In: *IEEE Industrial Electronics Magazine* 10.4 (2016), pp. 32–49.

[114] H. Yoshida. *The Internet Of Things That Matter: Integrating Smart Cities With Smart Agriculture*. Digitalist Magazine. 2016. URL: http://www.digitalistmag.com/iot/2016/05/19/internet-of-things-that-matter-integrating-smart-cities-with-smart-agriculture-04221203 (visited on 05/19/2016).

[115] Toshiba:Press-Release. *Toshiba and Dell Technologies’ Deep Learning Testbed for IoT is First Approved by Industrial Internet Consortium*. 2016. URL: https://www.toshiba.co.jp/about/press/2016_10/pr1702.htm (visited on 02/09/2017).

[116] X. Song, H. Kanasugi, and R. Shibasaki. “Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level”. In: IJCAI. 2016.

[117] V. C. Liang, R. T. Ma, W. S. Ng, L. Wang, M. Winslett, H. Wu, S. Ying, and Z. Zhang. “Mercury: Metro density prediction with recurrent neural network on streaming CDR data”. In: *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*. IEEE. 2016, pp. 1374–1377.

[118] X. Li, L. Peng, Y. Hu, J. Shao, and T. Chi. “Deep learning architecture for air quality predictions”. In: *Environmental Science and Pollution Research* 23.22 (2016), pp. 22408–22417.

[119] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo. “Deep learning for decentralized parking lot occupancy detection”. In: *Expert Systems with Applications* (2017).

[120] S. Valipour, M. Siam, E. Stroulia, and M. Jagersand. “Parking-stall vacancy indicator system, based on deep convolutional neural networks”. In: *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. 2016, pp. 655–660.

[121] D. C. Mocanu, E. Mocanu, P. H. Nguyen, M. Gibescu, and A. Liotta. “Big IoT data mining for real-time energy disaggregation in buildings”. In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. 2016, pp. 9–12.

[122] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling. “Deep learning for estimating building energy consumption”. In: *Sustainable Energy, Grids and Networks* 6 (2016), pp. 91–99.

[123] A. Gensler, J. Henze, B. Sick, and N. Raabe. “Deep Learning for solar power forecasting—An approach using AutoEncoder and LSTM Neural Networks”. In: *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*. IEEE. 2016, pp. 2858–2865.

[124] Y. Tian and L. Pan. “Predicting Short-Term Traffic Flow by Long Short-Term Memory Recurrent Neural Network”. In: *Smart City/SocialCom/SustainCom (SmartCity), 2015 IEEE International Conference on*. IEEE. 2015, pp. 153–158.

[125] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber. “Multi-column deep neural network for traffic sign classification”. In: *Neural Networks* 32 (2012), pp. 333–338.

[126] K. Lim, Y. Hong, Y. Choi, and H. Byun. “Real-time traffic sign recognition based on a general purpose GPU and deep-learning”. In: *PLoS one* 12.3 (2017), e0173317.

[127] E. Ackerman and A. Pagano. “Deep-Learning First: Drive.ai’s Path to Autonomous Driving”. In: *IEEE Spectrum* (2017).

[128] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, and Y. Ma. “DeepFood: Deep Learning-Based Food Image Recognition for Computer-Aided Dietary Assessment”. In: *International Conference on Smart Homes and Health Telematics*. Springer. 2016, pp. 37–48.

[129] C. R. Pereira, D. R. Pereira, J. P. Papa, G. H. Rosa, and X.-S. Yang. “Convolutional Neural Networks Applied for Parkinson’s Disease Identification”. In: *Machine Learning for Health Informatics*. Springer, 2016, pp. 377–390.

[130] G. Muhammad, S. M. M. Rahman, A. Alelaiwi, and A. Alamri. “Smart Health Solution Integrating IoT and Cloud: A Case Study of Voice Pathology Monitoring”. In: *IEEE Communications Magazine* 55.1 (2017), pp. 69–73.

[131] J. Wang, H. Ding, F. Azamian, B. Zhou, C. Iribarren, S. Molloi, and P. Baldi. “Detecting cardiovascular disease from mammograms with deep learning”. In: *IEEE Transactions on Medical Imaging* (2017).

[132] P. Feng, M. Yu, S. M. Naqvi, and J. A. Chambers. “Deep learning for posture analysis in fall detection”. In: *Digital Signal Processing (DSP), 2014 19th International Conference on*. IEEE. 2014, pp. 12–17.

[133] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzell. “Learning to diagnose with LSTM recurrent neural networks”. In: *ICLR 2016*. 2016.

[134] D. Ravì, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G.-Z. Yang. “Deep Learning for Health Informatics”. In: *IEEE journal of biomedical and health informatics* 21.1 (2017), pp. 4–21.

[135] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov. “Deep Learning Classification of Land Cover and Crop Types Using Remote Sensing Data”. In: *IEEE Geoscience and Remote Sensing Letters* (2017).

[136] K. Kuwata and R. Shibasaki. “Estimating crop yields with deep learning and remotely sensed data”. In: *Geoscience and Remote Sensing Symposium (IGARSS), 2015 IEEE International*. IEEE. 2015, pp. 858–861.

[137] G. J. Scott, M. R. England, W. A. Starms, R. A. Marcum, and C. H. Davis. “Training Deep Convolutional Neural Networks for Land–Cover Classification of High-Resolution Imagery”. In: *IEEE Geoscience and Remote Sensing Letters* 14.4 (2017), pp. 549–553.

[138] K. A. Steen, P. Christiansen, H. Karstoft, and R. N. Jørgensen. “Using deep learning to challenge safety standard for highly autonomous machines in agriculture”. In: *Journal of Imaging* 2.1 (2016), p. 6.

[139] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool. “Deepfruits: A fruit detection system using deep neural networks”. In: *Sensors* 16.8 (2016), p. 1222.

[140] M. B. Ibáñez, Á. Di Serio, D. Villarán, and C. D. Kloos. “Experimenting with electromagnetism using augmented reality: Impact on flow student experience and educational effectiveness”. In: *Computers & Education* 71 (2014), pp. 1–13.

[141] L.-f. Kwok. “A vision for the development of i-campus”. In: *Smart Learning Environments* 2 (2015), pp. 1–12.

[142] H. Wang, N. Wang, and D.-Y. Yeung. “Collaborative deep learning for recommender systems”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2015, pp. 1235–1244.

[143] T.-Y. Yang, C. G. Brinton, C. Joe-Wong, and M. Chiang. "Behavior-Based Grade Prediction for MOOCs via Time Series Neural Networks". In: *IEEE Journal of Selected Topics in Signal Processing* (2017).

[144] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein. "Deep knowledge tracing". In: *Advances in Neural Information Processing Systems*. 2015, pp. 505–513.

[145] F. Conti, A. Pullini, and L. Benini. "Brain-inspired classroom occupancy monitoring on a low-power mobile platform". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2014, pp. 610–615.

[146] H. Shao, H. Jiang, F. Wang, and H. Zhao. "An enhancement deep feature fusion method for rotating machinery fault diagnosis". In: *Knowledge-Based Systems* 119 (2017), pp. 200–220.

[147] H. Lee. "Framework and development of fault detection classification using IoT device and cloud environment". In: *Journal of Manufacturing Systems* (2017).

[148] H. Lee, Y. Kim, and C. O. Kim. "A Deep Learning Model for Robust Wafer Fault Monitoring With Sensor Measurement Noise". In: *IEEE Transactions on Semiconductor Manufacturing* 30.1 (2017), pp. 23–31.

[149] W. Yan and L. Yu. "On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach". In: *Proceedings of the Annual Conference of the Prognostics and Health Management Society*. 2015.

[150] Y. Liu and L. Wu. "Geological Disaster Recognition on Optical Remote Sensing Images Using Deep Learning". In: *Procedia Computer Science* 91 (2016), pp. 566–575.

[151] Q. Wang, Y. Guo, L. Yu, and P. Li. "Earthquake Prediction based on Spatio-Temporal Data Mining: An LSTM Network Approach". In: *IEEE Transactions on Emerging Topics in Computing* (2017).

[152] H. Maeda, Y. Sekimoto, and T. Seto. "Lightweight road manager: smartphone-based automatic determination of road damage status by deep neural network". In: *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*. ACM. 2016, pp. 37–45.

[153] L Steinberg. *Forbes - Changing the game: The rise of sports analytics*. 2015. URL: https://www.forbes.com/sites/leighsteinberg/2015/08/18/changing-the-game-the-rise-of-sports-analytics.

[154] W. Liu, J. Liu, X. Gu, K. Liu, X. Dai, and H. Ma. “Deep Learning Based Intelligent Basketball Arena with Energy Image”. In: *International Conference on Multimedia Modeling*. Springer. 2017, pp. 601–613.

[155] K.-C. Wang and R. Zemel. “Classifying NBA offensive plays using neural networks”. In: *Proc. MIT SLOAN Sports Analytics Conf.* 2016.

[156] R. Shah and R. Romijnders. “Applying Deep Learning to Basketball Trajectories”. In: *ACM KDD’16*. 2016.

[157] T. Kautz, B. H. Groh, J. Hannink, U. Jensen, H. Strubberg, and B. M. Eskofier. “Activity recognition in beach volleyball using a Deep Convolutional Neural Network”. In: *Data Mining and Knowledge Discovery* (2017), pp. 1–28.

[158] M. S. Ibrahim, S. Muralidharan, Z. Deng, A. Vahdat, and G. Mori. “A hierarchical deep temporal model for group activity recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1971–1980.

[159] S. Bell and K. Bala. “Learning visual similarity for product design with convolutional neural networks”. In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), p. 98.

[160] L. Xiao and X. Yichao. “Exact clothing retrieval approach based on deep neural network”. In: *Information Technology, Networking, Electronic and Automation Control Conference, IEEE*. IEEE. 2016, pp. 396–400.

[161] M Hadi Kiapour, X. Han, S. Lazebnik, A. C. Berg, and T. L. Berg. “Where to buy it: Matching street clothing photos in online shops”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 3343–3351.

[162] S. Advani, P. Zientara, N. Shukla, I. Okafor, K. Irick, J. Sampson, S. Datta, and V. Narayanan. “A Multitask Grocery Assist System for the Visually Impaired: Smart glasses, gloves, and shopping carts provide auditory and tactile feedback”. In: *IEEE Consumer Electronics Magazine* 6.1 (2017), pp. 73–81.

[163] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao. “A multi-stream bi-directional recurrent neural network for fine-grained action detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1961–1970.

[164] Q. Feng, Y. Zhang, C. Li, Z. Dou, and J. Wang. “Anomaly detection of spectrum in wireless communication via deep auto-encoders”. In: *The Journal of Supercomputing* 73.7 (2017), pp. 3161–3178.

[165] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret. "Conditional Variational Autoencoder for Prediction and Feature Recovery Applied to Intrusion Detection in IoT". In: *Sensors* 17.9 (2017), p. 1967.

[166] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi. "A Deep Learning Approach to Network Intrusion Detection". In: *Forest (RF)* 2 (2017).

[167] M. Hasan, E. Hossain, and D. Niyato. "Random access for machine-to-machine communication in LTE-advanced networks: issues and approaches". In: *IEEE communications Magazine* 51.6 (2013), pp. 86–93.

[168] H.-Y. Kim and J.-M. Kim. "A load balancing scheme based on deep-learning in IoT". In: *Cluster Computing* 20.1 (2017), pp. 873–878.

[169] M. Schmidt, D. Block, and U. Meier. "Wireless interference identification with convolutional neural networks". In: *arXiv preprint arXiv:1703.00737v1 [cs.LG]* (2017).

[170] N. Ahad, J. Qadir, and N. Ahsan. "Neural networks in wireless networks: Techniques, applications and guidelines". In: *Journal of Network and Computer Applications* 68 (2016), pp. 1–27.

[171] R. Li, Z. Zhao, X. Zhou, G. Ding, Y. Chen, Z. Wang, and H. Zhang. "Intelligent 5G: When cellular networks meet artificial intelligence". In: *IEEE Wireless Communications* 24.5 (2017), pp. 175–183.

[172] M.-R. Fida, A. Lutu, M. K. Marina, and Ö. Alay. "ZipWeave: Towards efficient and reliable measurement based mobile coverage maps". In: *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE. 2017, pp. 1–9.

[173] OpenSignal. *Global Cell Coverage Maps*. (Accessed on 2018-02-06). URL: http://opensignal.com/networks.

[174] A. Canziani, A. Paszke, and E. Culurciello. "An analysis of deep neural network models for practical applications". In: *arXiv preprint arXiv:1605.07678v4 [cs.CV]* (2016).

[175] K. He, X. Zhang, S. Ren, and J. Sun. "Identity Mappings in Deep Residual Networks". In: *Computer Vision – ECCV 2016*. Ed. by B. Leibe, J. Matas, N. Sebe, and M. Welling. Springer International Publishing, 2016, pp. 630–645.

[176] C. Bourguignat. *Interpretable vs Powerful Predictive Models: Why We Need Them Both*. (Accessed on 2018-02-15). 2014. URL: https://medium.com/@chris_bour/interpretable-vs-powerful-predictive-models-why-we-need-them-both-990340074979.

[177] F. Chollet. *Deep learning with Python*. Manning Publications, 2018.

[178] V. J. Hellendoorn and P. Devanbu. “Are deep neural networks the best choice for modeling source code?” In: *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM. 2017, pp. 763–773.

[179] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. “Return of the devil in the details: Delving deep into convolutional nets”. In: *arXiv preprint arXiv:1405.3531v4 [cs.CV]* (2014).

[180] J. Ba and R. Caruana. “Do deep nets really need to be deep?” In: *Advances in neural information processing systems*. 2014, pp. 2654–2662.

[181] T. A. Eriksson, H. Bülow, and A. Leven. “Applying Neural Networks in Optical Communication Systems: Possible Pitfalls”. In: *IEEE Photonics Technology Letters* 29.23 (2017), pp. 2091–2094.

[182] M. Denil, B. Shakibi, L. Dinh, N. de Freitas, et al. “Predicting parameters in deep learning”. In: *Advances in Neural Information Processing Systems*. 2013, pp. 2148–2156.

[183] Y. LeCun, J. S. Denker, S. A. Solla, R. E. Howard, and L. D. Jackel. “Optimal Brain Damage”. In: *NIPs*. Vol. 2. 1989, pp. 598–605.

[184] S. Han, J. Pool, J. Tran, and W. Dally. “Learning both weights and connections for efficient neural network”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 1135–1143.

[185] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally. “EIE: efficient inference engine on compressed deep neural network”. In: *arXiv preprint arXiv:1602.01528v2 [cs.CV]* (2016).

[186] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. “Compressing neural networks with the hashing trick”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. JMLR: W&CP, 2015.

[187] M. Courbariaux and Y. Bengio. “Binarized Neural Networks: Training deep neural networks with weights and activations constrained to+ 1 or-1”. In: *arXiv preprint arXiv:1602.02830v3 [cs.LG]* (2016).

[188] S. Venkataramani, A. Ranjan, K. Roy, and A. Raghunathan. “AxNN: energy-efficient neuromorphic systems using approximate computing”. In: *Proceedings of the 2014 international symposium on Low power electronics and design*. ACM. 2014, pp. 27–32.

[189] B. Moons, B. De Brabandere, L. Van Gool, and M. Verhelst. “Energy-efficient convnets through approximate computing”. In: *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*. IEEE. 2016, pp. 1–8.

[190] S. G. Ramasubramanian, R. Venkatesan, M. Sharad, K. Roy, and A. Raghunathan. "SPINDLE: SPINtronic deep learning engine for large-scale neuromorphic computing". In: *Proceedings of the 2014 international symposium on Low power electronics and design*. ACM. 2014, pp. 15–20.

[191] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze. "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks". In: *IEEE Journal of Solid-State Circuits* 52.1 (2017), pp. 127–138.

[192] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam. "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning". In: *ACM Sigplan Notices*. Vol. 49. 4. ACM. 2014, pp. 269–284.

[193] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar. "Deepx: A software accelerator for low-power deep learning inference on mobile devices". In: *Information Processing in Sensor Networks (IPSN), 2016 15th ACM/IEEE International Conference on*. IEEE. 2016, pp. 1–12.

[194] S. Bang, J. Wang, Z. Li, C. Gao, Y. Kim, Q. Dong, Y.-P. Chen, L. Fick, X. Sun, R. Dreslinski, et al. "14.7 A 288μW programmable deep-learning processor with 270KB on-chip weight storage using non-uniform memory hierarchy for mobile intelligence". In: *Solid-State Circuits Conference (ISSCC), 2017 IEEE International*. IEEE. 2017, pp. 250–251.

[195] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. *The Street View House Numbers (SVHN) Dataset*. (Accessed on 2018-02-09). URL: http://ufldl.stanford.edu/housenumbers/.

[196] X. Xu, S. Das, and K. Kreutz-Delgado. "ApproxDBN: Approximate Computing for Discriminative Deep Belief Networks". In: *arXiv preprint arXiv:1704.03993v3 [cs.NE]* (2017).

[197] S. Venkataramani, K. Roy, and A. Raghunathan. "Efficient embedded learning for IoT devices". In: *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE. 2016, pp. 308–311.

[198] D. D. Awschalom and M. E. Flatté. "Challenges for semiconductor spintronics". In: *Nature Physics* 3.3 (2007), pp. 153–159.

[199] K. Bourzac. "Speck-size computers: Now with deep learning [News]". In: *IEEE Spectrum* 54.4 (2017), pp. 13–15.

[200] B. Moons and M. Verhelst. "A 0.3–2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets". In: *VLSI Circuits (VLSI-Circuits), 2016 IEEE Symposium on*. IEEE. 2016, pp. 1–2.

[201] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861v1 [cs.CV]* (2017).

[202] D. Ravi, C. Wong, B. Lo, and G.-Z. Yang. "A deep learning approach to on-node sensor data analytics for mobile or wearable devices". In: *IEEE Journal of Biomedical and Health Informatics* (2016).

[203] N. Nguyen-Thanh, L. Yang, D. H. Nguyen, C. Jabbour, B. Murmann, et al. "Cognitive computation and communication: A complement solution to cloud for IoT". In: *Advanced Technologies for Communications (ATC), 2016 International Conference on*. IEEE. 2016, pp. 222–230.

[204] B. Tang, Z. Chen, G. Hefferman, S. Pei, W. Tao, H. He, and Q. Yang. "Incorporating Intelligence in Fog Computing for Big Data Analysis in Smart Cities". In: *IEEE Transactions on Industrial Informatics* (2017).

[205] A. Al-Fuqaha, A. Khreishah, M. Guizani, A. Rayes, and M. Mohammadi. "Toward better horizontal integration among IoT services". In: *IEEE Communications Magazine* 53.9 (2015), pp. 72–79.

[206] A. Coates, B. Huval, T. Wang, D. J. Wu, A. Y. Ng, and B. Catanzaro. "Deep learning with COTS HPC systems". In: *Proceedings of the 30th International Conference on Machine Learning*. Vol. 28. JMLR: W & CP. 2013.

[207] T. M. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman. "Project Adam: Building an Efficient and Scalable Deep Learning Training System." In: *11th USENIX Symposium on Operating Systems Design and Implementation*. Vol. 14. 2014, pp. 571–582.

[208] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, et al. "In-Datacenter Performance Analysis of a Tensor Processing Unit". In: *44th International Symposium on Computer Architecture (ISCA)*. 2017.

[209] K. Daniel. *Lessons learned from deploying deep learning at scale*. O'Reilly Artificial Intelligence conference. 2016. URL: http://conferences.oreilly.com/artificial-intelligence/ai-ny-2016/public/schedule/detail/54098 (visited on 08/12/2017).

[210] N. Hemsoth. *GPU Platforms Set to Lengthen Deep Learning Reach*. The Next Platform. 2015. URL: http://www.nextplatform.com/2015/12/07/gpu-platforms-emerge-for-longer-deep-learning-reach/ (visited on 08/12/2017).

[211] Y. Simmhan and S. Perera. "Big Data Analytics Platforms for Real-Time Applications in IoT". In: *Big Data Analytics*. Springer, 2016, pp. 115–135.

[212] S. B. Qaisar and M. Usman. "Fog Networking for Machine Health Prognosis: A Deep Learning Perspective". In: *International Conference on Computational Science and Its Applications*. Springer. 2017, pp. 212–219.

[213] D. Li, T. Salonidis, N. V. Desai, and M. C. Chuah. "DeepCham: Collaborative Edge-Mediated Adaptive Deep Learning for Mobile Object Recognition". In: *Edge Computing (SEC), IEEE/ACM Symposium on*. IEEE. 2016, pp. 64–76.

[214] Wikipedia. *List of datasets for machine learning research*. 2017. URL: https://en.wikipedia.org/wiki/List_of_datasets_for_machine_learning_research (visited on 02/09/2017).

[215] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. "The limitations of deep learning in adversarial settings". In: *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE. 2016, pp. 372–387.

[216] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic. "Deep learning applications and challenges in big data analytics". In: *Journal of Big Data* 2.1 (2015), p. 1.

[217] S. LaValle, E. Lesser, R. Shockley, M. S. Hopkins, and N. Kruschwitz. "Big data, analytics and the path from insights to value". In: *MIT sloan management review* 52.2 (2011), p. 21.

[218] A. Nguyen, J. Yosinski, and J. Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 427–436.

[219] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga. "Ensemble deep learning for regression and time series forecasting". In: *Computational Intelligence in Ensemble Learning (CIEL), 2014 IEEE Symposium on*. IEEE. 2014, pp. 1–6.

[220] M. A. Alsheikh, D. Niyato, S. Lin, H.-P. Tan, and Z. Han. "Mobile big data analytics using deep learning and apache spark". In: *IEEE Network* 30.3 (2016), pp. 22–29.

[221] A. Gharaibeh, A. Khreishah, M. Mohammadi, A. Al-Fuqaha, I. Khalil, and A. Rayes. "Online Auction of Cloud Resources in Support of the Internet of Things". In: *IEEE Internet of Things Journal* PP.99 (2017), pp. 1–14. DOI: 10.1109/JIOT.2017.2724938.

[222] National Science Foundation. *Cyber-Physical Systems (CPS) - Program Solicitation (NSF 17-529)*. 2017. URL: https://www.nsf.gov/pubs/2017/nsf17529/nsf17529.pdf (visited on 06/30/2017).

[223] P. Valente Klaine, M. A. Imran, O. Onireti, and R. D. Souza. "A survey of machine learning techniques applied to self organizing cellular networks". In: *IEEE Communications Surveys and Tutorials* PP (99 2017).

[224] J. Lee, J. Wang, D. Crandall, S. Šabanović, and G. Fox. "Real-Time, Cloud-Based Object Detection for Unmanned Aerial Vehicles". In: *Robotic Computing (IRC), IEEE International Conference on*. IEEE. 2017, pp. 36–43.

[225] L. Tai, S. Li, and M. Liu. "A deep-network solution towards model-less obstacle avoidance". In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 2759–2764.

[226] O. Akgul, H. I. Penekli, and Y. Genc. "Applying Deep Learning in Augmented Reality Tracking". In: *Signal-Image Technology & Internet-Based Systems (SITIS), 2016 12th International Conference on*. IEEE. 2016, pp. 47–54.

[227] R. E. Sutanto, L. Pribadi, and S. Lee. "3D Integral Imaging Based Augmented Reality with Deep Learning Implemented by Faster R-CNN". In: *International Conference on Mobile and Wireless Technology*. Springer. 2017, pp. 241–247.

[228] L. Tai and M. Liu. "Deep-learning in mobile robotics-from perception to control systems: A survey on why and why not". In: *arXiv preprint arXiv:1612.07139v3 [cs.RO]* (2016).

[229] R. Goeddel and E. Olson. "Learning semantic place labels from occupancy grids using CNNs". In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 3999–4004.

[230] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. "Internet of Things: A survey on enabling technologies, protocols, and applications". In: *IEEE Communications Surveys & Tutorials* 17.4 (2015), pp. 2347–2376.

[231] N. E. Klepeis, W. C. Nelson, W. R. Ott, J. P. Robinson, A. M. Tsang, P. Switzer, J. V. Behar, S. C. Hern, W. H. Engelmann, et al. "The National Human Activity Pattern Survey (NHAPS): a resource for assessing exposure to environmental pollutants". In: *Journal of exposure analysis and environmental epidemiology* 11.3 (2001), pp. 231–252.

[232] L. Mainetti, V. Mighali, and L. Patrono. "A location-aware architecture for heterogeneous building automation systems". In: *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE. 2015, pp. 1065–1070.

[233] S. Alletto, R. Cucchiara, G. Del Fiore, L. Mainetti, V. Mighali, L. Patrono, and G. Serra. "An Indoor Location-Aware System for an IoT-Based Smart Museum". In: *IEEE Internet of Things Journal* 3.2 (2016), pp. 244–253.

[234] M. V. Moreno, J. L. Hernández, and A. F. Skarmeta. "A new location-aware authorization mechanism for indoor environments". In: *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*. IEEE. 2014, pp. 791–796.

[235] G. Sunkada. *System for and method of location aware marketing*. US Patent App. 12/851,968. 2012.

[236] P. Dickinson, G. Cielniak, O. Szymanezyk, and M. Mannion. "Indoor positioning of shoppers using a network of Bluetooth Low Energy beacons". In: *Indoor Positioning and Indoor Navigation (IPIN), 2016 International Conference on*. IEEE. 2016, pp. 1–8.

[237] J. Torres-Sospedra, J. Avariento, D. Rambla, R. Montoliu, S. Casteleyn, M. Benedito-Bordonau, M. Gould, and J. Huerta. "Enhancing integrated indoor/outdoor mobility in a smart campus". In: *International Journal of Geographical Information Science* 29.11 (2015), pp. 1955–1968.

[238] F. Zafari, I. Papapanagiotou, and K. Christidis. "Microlocation for Internet-of-Things-Equipped Smart Buildings". In: *IEEE Internet of Things Journal* 3.1 (2016), pp. 96–112.

[239] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge, 1998.

[240] R. Faragher and R. Harle. "Location fingerprinting with bluetooth low energy beacons". In: *IEEE Journal on Selected Areas in Communications* 33.11 (2015), pp. 2418–2428.

[241] P. Christiano. *Semi-supervised reinforcement learning*. 2016. URL: https://medium.com/ai-control/semi-supervised-reinforcement-learning-cf7d5375197f (visited on 12/20/2016).

[242] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. "Concrete problems in AI safety". In: *arXiv preprint arXiv:1606.06565* (2016).

[243] S. Nemati, M. M. Ghassemi, and G. D. Clifford. "Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach". In: *Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the*. IEEE. 2016, pp. 2978–2981.

[244] D. Zhao, Y. Chen, and L. Lv. "Deep reinforcement learning with visual attention for vehicle classification'". In: *IEEE Transactions on Cognitive and Developmental Systems* ().

[245] J. C. Caicedo and S. Lazebnik. "Active object localization with deep reinforcement learning". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2488–2496.

[246] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. "Target-driven visual navigation in indoor scenes using deep reinforcement learning". In: *arXiv preprint arXiv:1609.05143* (2016).

[247] L. Li, Y. Lv, and F.-Y. Wang. "Traffic signal timing via deep reinforcement learning". In: *IEEE/CAA Journal of Automatica Sinica* 3.3 (2016), pp. 247–254.

[248] H. Mao, M. Alizadeh, I. Menache, and S. Kandula. "Resource Management with Deep Reinforcement Learning". In: *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. ACM. 2016, pp. 50–56.

[249] K. Narasimhan, T. Kulkarni, and R. Barzilay. "Language understanding for text-based games using deep reinforcement learning". In: *arXiv preprint arXiv:1506.08941* (2015).

[250] J. He, J. Chen, X. He, J. Gao, L. Li, L. Deng, and M. Ostendorf. "Deep reinforcement learning with a natural language action space". In: *arXiv preprint arXiv:1511.04636* (2015).

[251] V. François-Lavet. "Deep Reinforcement Learning Solutions for Energy Microgrids Management". In: *Proceedings of the European Workshop on Reinforcement Learning*. 2016.

[252] B. Wang, Q. Chen, L. T. Yang, and H.-C. Chao. "Indoor smartphone localization via fingerprint crowdsourcing: challenges and approaches". In: *IEEE Wireless Communications* 23.3 (2016), pp. 82–89.

[253] S. Kajioka, T. Mori, T. Uchiya, I. Takumi, and H. Matsuo. "Experiment of indoor position presumption based on RSSI of Bluetooth LE beacon". In: *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*. IEEE. 2014, pp. 337–339.

[254] Y. Wang, X. Yang, Y. Zhao, Y. Liu, and L. Cuthbert. "Bluetooth positioning using RSSI and triangulation methods". In: *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*. IEEE. 2013, pp. 837–842.

[255] G. Ding, Z. Tan, J. Zhang, and L. Zhang. "Fingerprinting localization based on affinity propagation clustering and artificial neural networks". In: *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. 2013, pp. 2317–2322.

[256] J. Luo and H. Gao. "Deep belief networks for fingerprinting indoor localization using ultrawideband technology". In: *International Journal of Distributed Sensor Networks* 2016 (2016), p. 18.

[257] X. Zhang, J. Wang, Q. Gao, X. Ma, and H. Wang. "Device-free wireless localization and activity recognition with deep learning". In: *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. IEEE. 2016, pp. 1–5.

[258] T. Pulkkinen, T. Roos, and P. Myllymäki. "Semi-supervised learning for wlan positioning". In: *International Conference on Artificial Neural Networks*. Springer. 2011, pp. 355–362.

[259] J. Weston, F. Ratle, H. Mobahi, and R. Collobert. "Deep learning via semi-supervised embedding". In: *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 639–655.

[260] O Chapelle, B Schölkopf, and A Zien. *Semi-supervised learning*. 2006.

[261] J. Walker, C. Doersch, A. Gupta, and M. Hebert. "An uncertain future: Forecasting from static images using variational autoencoders". In: *European Conference on Computer Vision*. Springer. 2016, pp. 835–851.

[262] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533.

[263] L. Mainetti, L. Patrono, and I. Sergi. "A survey on indoor positioning systems". In: *Software, Telecommunications and Computer Networks (SoftCOM), 2014 22nd International Conference on*. IEEE. 2014, pp. 111–120.

[264] F. Chollet. *Keras: Deep learning library for theano and tensorflow*. 2015. URL: https://keras.io/.

[265] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani. "Deep Learning for IoT Big Data and Streaming Analytics: A Survey". In: *arXiv preprint arXiv:1712.04301v1 [cs.NI]* (2017).

[266] M. Mohammadi and A. Al-Fuqaha. "Enabling Cognitive Smart Cities Using Big Data and Machine Learning: Approaches and Challenges". In: *IEEE Communications Magazine* 56.2 (2018), pp. 2–9.

[267] G. Andrienko, D. Malerba, M. May, and M. Teisseire. *Mining spatio-temporal data*. 2006.

[268] T. G. Dietterich et al. "Ensemble methods in machine learning". In: *Multiple classifier systems* 1857 (2000), pp. 1–15.

[269] X. Qiu, Y. Ren, P. N. Suganthan, and G. A. Amaratunga. "Empirical mode decomposition based ensemble deep learning for load demand time series forecasting". In: *Applied Soft Computing* 54 (2017), pp. 246–255.

[270] J. Liu, A. Shahroudy, D. Xu, A. K. Chichung, and G. Wang. "Skeleton-Based Action Recognition Using Spatio-Temporal LSTM Network with Trust Gates". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[271] A. Ortiz, J. Munilla, J. M. Gorriz, and J. Ramirez. "Ensembles of deep learning architectures for the early diagnosis of the Alzheimer's disease". In: *International journal of neural systems* 26.07 (2016), p. 1650025.

[272] H.-z. Wang, G.-q. Li, G.-b. Wang, J.-c. Peng, H. Jiang, and Y.-t. Liu. "Deep learning based ensemble approach for probabilistic wind power forecasting". In: *Applied Energy* 188 (2017), pp. 56–70.

[273] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. "Large scale distributed deep networks". In: *Advances in neural information processing systems*. 2012, pp. 1223–1231.

[274] F. N. Iandola, M. W. Moskewicz, K. Ashraf, and K. Keutzer. "Firecaffe: near-linear acceleration of deep neural network training on compute clusters". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2592–2600.

[275] B. C. Ooi, K.-L. Tan, S. Wang, W. Wang, Q. Cai, G. Chen, J. Gao, Z. Luo, A. K. Tung, Y. Wang, et al. "SINGA: A distributed deep learning platform". In: *Proceedings of the 23rd ACM international conference on Multimedia*. ACM. 2015, pp. 685–688.

[276] S. Teerapittayanon, B. McDanel, and H. Kung. "Distributed deep neural networks over the cloud, the edge and end devices". In: *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2017, pp. 328–339.

[277] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz. "Theory and practice of bloom filters for distributed systems". In: *IEEE Communications Surveys & Tutorials* 14.1 (2012), pp. 131–155.

[278] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng. "Fog computing for the internet of things: Security and privacy issues". In: *IEEE Internet Computing* 21.2 (2017), pp. 34–42.

[279] A. Broder and M. Mitzenmacher. "Network applications of bloom filters: A survey". In: *Internet mathematics* 1.4 (2004), pp. 485–509.

[280] R. E. Schapire. "The strength of weak learnability". In: *Machine learning* 5.2 (1990), pp. 197–227.

[281] C. Zhang and Y. Ma. *Ensemble machine learning: methods and applications*. Springer, 2012.

[282] R. T. Marler and J. S. Arora. "Survey of multi-objective optimization methods for engineering". In: *Structural and multidisciplinary optimization* 26.6 (2004), pp. 369–395.

[283] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*. Vol. 5. Springer.

[284] T. L. Vincent and W. J. Grantham. *Optimality in parametric systems*. John Wiley & Sons, 1981.

[285] P. Samadi, H. Mohsenian-Rad, R. Schober, and V. W. Wong. "Advanced demand side management for the future smart grid using mechanism design". In: *IEEE Transactions on Smart Grid* 3.3 (2012), pp. 1170–1180.

[286] G. Capizzi, G. L. Sciuto, C. Napoli, and E. Tramontana. "Advanced and Adaptive Dispatch for Smart Grids by means of Predictive Models". In: *IEEE Transactions on Smart Grid* (2017).

[287] Y. Simmhan, M. Giakkoupis, B. Cao, and V. K. Prasanna. *On using cloud platforms in a software architecture for smart energy grids*. Tech. rep. City of Los Angeles Department, CA (United States), 2010.

[288] M. Mohammadi, A. Al-Fuqaha, and J.-S. Oh. "Path Planning in Support of Smart Mobility Applications using Generative Adversarial Networks". In: *arXiv preprint arXiv:1804.08396v1 [cs.LG]* (2018).

[289] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang. "Traffic flow prediction with big data: a deep learning approach". In: *IEEE Transactions on Intelligent Transportation Systems* 16.2 (2015), pp. 865–873.

[290] N. G. Polson and V. O. Sokolov. "Deep learning for short-term traffic flow prediction". In: *Transportation Research Part C: Emerging Technologies* 79 (2017), pp. 1–17.

[291] F. Schimbinschi, X. V. Nguyen, J. Bailey, C. Leckie, H. Vu, and R. Kotagiri. "Traffic forecasting in complex urban networks: Leveraging big data and machine learning". In: *Big data (big data), 2015 IEEE international conference on*. IEEE. 2015, pp. 1019–1024.

[292] K. Lin, M. Chen, J. Deng, M. M. Hassan, and G. Fortino. "Enhanced fingerprinting and trajectory prediction for IoT localization in smart buildings". In: *IEEE Transactions on Automation Science and Engineering* 13.3 (2016), pp. 1294–1307.

[293] Australian Energy Market Operator. *Aggregated Price and Demand Data - Historical*. (Accessed on 2017-12-06). 2017. URL: http://www.aemo.com.au/Electricity/National-Electricity-Market-NEM/Data-dashboard#aggregated-data.

[294] M. I. Ali, F. Gao, and A. Mileo. "Citybench: A configurable benchmark to evaluate rsp engines using smart city datasets". In: *International Semantic Web Conference*. Springer. 2015, pp. 374–389.

[295] CityPulse EU FP7 Project. *CityPulse Dataset Collection*. (Accessed on 2018-01-28). 2014. URL: http://iot.ee.surrey.ac.uk:8080/datasets.html.

[296] M. Mohammadi and A. Al-Fuqaha. *BLE RSSI Dataset for Indoor localization*. (Accessed on 2018-02-01). 2018. URL: https://www.kaggle.com/mehdimka/ble-rssi-dataset.

[297] I. Koprinska, M. Rana, and V. G. Agelidis. "Correlation and instance based feature selection for electricity load forecasting". In: *Knowledge-Based Systems* 82 (2015), pp. 29–40.

[298] J. Ledlie, J.-g. Park, D. Curtis, A. Cavalcante, L. Camara, A. Costa, and R. Vieira. "Molé: a scalable, user-generated WiFi positioning engine". In: *Journal of Location Based Services* 6.2 (2012), pp. 55–80.

[299] C. Benevolo, R. P. Dameri, and B. D'Auria. "Smart mobility in smart city". In: *Empowering Organizations*. Springer, 2016, pp. 13–28.

[300] L.-W. Chen, J.-J. Chung, and J.-X. Liu. "GoFAST: a group-based emergency guiding system with dedicated path planning for mobile users using smartphones". In: *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE. 2015, pp. 467–468.

[301] G. E. Legge, C. Downey, N. A. Giudice, and B. S. Tjan. *Indoor airport wayfinding for blind and visually impaired travelers*. Tech. rep. DOT/FAA/TC-TN16/54. 2016.

[302] A. Radford, L. Metz, and S. Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[303] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. "Generative Adversarial Text to Image Synthesis". In: *International Conference on Machine Learning*. 2016, pp. 1060–1069.

[304] Y. Li, Y. Zhuang, H. Lan, P. Zhang, X. Niu, and N. El-Sheimy. "Self-Contained Indoor Pedestrian Navigation Using Smartphone Sensors and Magnetic Features". In: *IEEE Sensors Journal* 16.19 (2016), pp. 7173–7182.

[305] Q.-H. Nguyen, H. Vu, T.-H. Tran, and Q.-H. Nguyen. "Developing a way-finding system on mobile robot assisting visually impaired people in an indoor environment". In: *Multimedia Tools and Applications* 76.2 (2017), pp. 2645–2669.

[306] J. Yao, C. Lin, X. Xie, A. J. Wang, and C.-C. Hung. "Path planning for virtual human motion using improved A* star algorithm". In: *2010 Seventh International Conference on Information Technology: New Generations (ITNG)*. IEEE. 2010, pp. 1154–1158.

[307] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica. "Path planning with modified a star algorithm for a mobile robot". In: *Procedia Engineering* 96 (2014), pp. 59–69.

[308] I. Apostolopoulos, N. Fallah, E. Folmer, and K. E. Bekris. "Integrated online localization and navigation for people with visual impairments using smart phones". In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 3.4 (2014), p. 21.

[309] L.-W. Chen and J.-J. Chung. "Mobility-Aware and Congestion-Relieved Dedicated Path Planning for Group-Based Emergency Guiding Based on Internet of Things Technologies". In: *IEEE Transactions on Intelligent Transportation Systems* 18.9 (2017), pp. 2453–2466.

[310] N. Hirose, A. Sadeghian, P. Goebel, and S. Savarese. "To go or not to go? A near unsupervised learning approach for robot navigation". In: *arXiv preprint arXiv:1709.05439v1 [cs.CV]* (2017).

[311] E. Linder-Norén. *Keras-GAN: Keras implementations of Generative Adversarial Networks*. (Accessed on 2018-03-04). 2017. URL: https://github.com/eriklindernoren/Keras-GAN.

[312] M. Mirza and S. Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784v1 [cs.LG]* (2014).

[313] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. "Improved techniques for training gans". In: *Advances in Neural Information Processing Systems*. 2016, pp. 2234–2242.

[314] A. Odena. "Semi-supervised learning with generative adversarial networks". In: *arXiv preprint arXiv:1606.01583v2 [stat.ML]* (2016).

[315] N. Newman. "Apple iBeacon technology briefing". In: *Journal of Direct, Data and Digital Marketing Practice* 15.3 (2014), pp. 222–225.

[316] M. Mohammadi, S. Mohammadi, and A. Al-Fuqaha. *Wayfinding, Path Planning, and Navigation Dataset*. (Accessed on 2018-02-12). 2018. URL: https://www.kaggle.com/mehdimka/path-planning.

[317] J. Read, B. Pfahringer, G. Holmes, and E. Frank. "Classifier chains for multi-label classification". In: *Machine learning* 85.3 (2011), p. 333.

[318] M. Alzantot, S. Chakraborty, and M. Srivastava. "Sensegen: A deep learning architecture for synthetic sensor data generation". In: *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE. 2017, pp. 188–193.

[319] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 105–114.

[320] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. "On large-batch training for deep learning: Generalization gap and sharp minima". In: *arXiv preprint arXiv:1609.04836v2 [cs.LG]* (2016).