Mobile Big Data Analytics Using Deep Learning and Apache Spark

Mohammad Abu Alsheikh, Dusit Niyato, Shaowei Lin, Hwee-Pink Tan, and Zhu Han

Abstract

The proliferation of mobile devices, such as smartphones and Internet of Things gadgets, has resulted in the recent mobile big data era. Collecting mobile big data is unprofitable unless suitable analytics and learning methods are utilized to extract meaningful information and hidden patterns from data. This article presents an overview and brief tutorial on deep learning in mobile big data analytics and discusses a scalable learning framework over Apache Spark. Specifically, distributed deep learning is executed as an iterative MapReduce computing on many Spark workers. Each Spark worker learns a partial deep model on a partition of the overall mobile, and a master deep model is then built by averaging the parameters of all partial models. This Spark-based framework speeds up the learning of deep models consisting of many hidden layers and millions of parameters. We use a context-aware activity recognition application with a real-world dataset containing millions of samples to validate our framework and assess its speedup effectiveness.

Mobile devices have matured as a reliable and cheap platform for collecting data in pervasive and ubiquitous sensing systems. Specifically, mobile devices are:

- Sold in mass market chains
- Connected to daily human activities
- Supported with embedded communication and sensing modules

According to the latest traffic forecast report by Cisco Systems [1], half a billion mobile devices were globally sold in 2015, and the mobile data traffic grew by 74 percent generating 3.7 exabytes (1 exabyte = 10^{18} bytes) of mobile data per month. Mobile big data (MBD) is a concept that describes a massive amount of mobile data that cannot be processed using a single machine. MBD contains useful information for solving many problems such as fraud detection, marketing and targeted advertising, context-aware computing, and healthcare. Therefore, MBD analytics is currently a high-focus topic aimed at extracting meaningful information and patterns from raw mobile data.

Deep learning is a solid tool in MBD analytics. Specifically, deep learning:

- Provides highly accurate results in MBD analytics
- Avoids the expensive design of handcrafted features
- Utilizes the massive unlabeled mobile data for unsupervised feature extraction

Due to the curse of dimensionality and size of MBD, learning deep models in MBD analytics is slow and takes anywhere from a few hours to several days when performed on conventional computing systems. Conversely, most mobile systems are not delay-tolerant, and decisions should be made as quickly as possible to attain high user satisfaction.

M. Abu Alsheikh is with Nanyang Technological University and the Institute for Infocomm Research.

Dusit Niyato is with Nanyang Technological University.

Shaowei Lin is with the Singapore University of Technology and Design.

Hwee-Pink Tan is with Singapore Management University.

Zhu Han is with the University of Houston.

To cope with the increased demand on scalable and adaptive mobile systems, this article presents a tutorial on developing a framework that enables time-efficient MBD analytics using deep models with millions of modeling parameters. Our framework is built over Apache Spark [2], which provides an open source cluster computing platform. This enables distributed learning using many computing cores on a cluster where continuously accessed data is cached to running memory, thus speeding up the learning of deep models several-fold. To prove the viability of the proposed framework, we implement a context-aware activity recognition system [3] on a computing cluster and train deep learning models using millions of data samples collected by mobile crowdsensing. In this test case, a client request includes accelerometer signals, and the server is programmed to extract the underlying human activity using deep activity recognition models. We show significant accuracy improvement of deep learning over conventional machine learning methods, improving 9 percent over random forests and 17.8 percent over multilayer perceptions from [4]. Moreover, the learning time of deep models is decreased as a result of the paralleled Spark-based implementation compared to a single-machine computation. For example, utilizing 6 Spark workers can speed up the learning of a 5-layer deep model of 20 million parameters 4-fold as compared to a single machine computing.

The rest of this article is organized as follows. We present an overview of MBD and discuss the challenges of MBD analytics. Next, we discuss the advantages and challenges of deep learning in MBD analytics. Then we propose a Spark-based framework for learning deep models for time-efficient MBD analytics within large-scale mobile systems. We present experimental analysis using a real-world dataset. Important research directions are discussed. Finally, we conclude the article.

Mobile Big Data: Concepts and Features

This section first introduces an overview of MBD and then discusses the key characteristics that make MBD analytics challenging.

0890-8044/16/$25.00 © 2016 IEEE

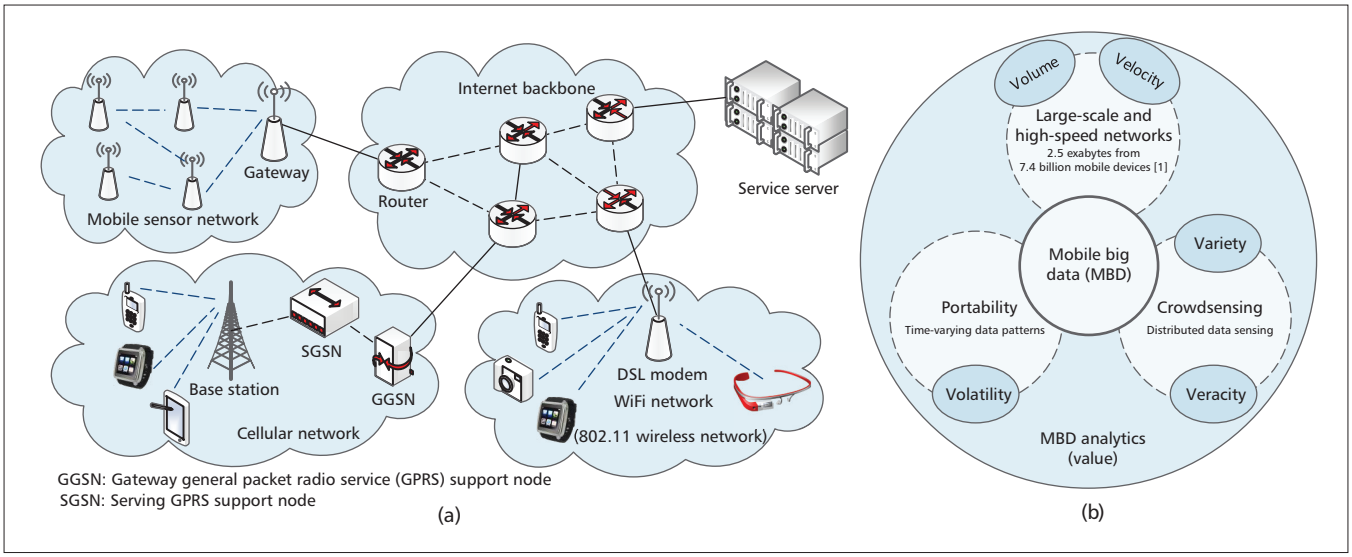

Figure 1. Illustration of the MBD era: a) typical architecture of a modern mobile network connecting smartphones, wearable computers, and IoT gadgets; b) main technological advances behind the MBD era.

The Era of MBD

Figure 1a shows a typical architecture of large-scale mobile systems used to connect various types of portable devices such as smartphones, wearable computers, and Internet of Things (IoT) gadgets. The widespread installation of various types of sensors, such as accelerometers, gyroscopes, compasses, and GPS sensors, in modern mobile devices allows many new applications. Essentially, each mobile device encapsulates its service request and own sensory data in a stateless data-interchange structure like Javascript object notation (JSON) format. The stateless format is important as mobile devices operate on different mobile operating systems (e.g., Android, iOS, and Tizen). Based on the collected MBD, a service server utilizes MBD analytics to discover hidden patterns and information. The importance of MBD analytics stems from its role in building complex mobile systems that could not be assembled and configured on small datasets. For example, an activity recognition application [3, 5] uses embedded accelerometers of mobile devices to collect proper acceleration data about daily human activities. After receiving a request, the service server maps the accelerometer data to the most probable human activities, which are used to support many interactive services (e.g., healthcare, smart building, and pervasive games).

MBD analytics is more versatile than conventional big data problems as data sources are portable and data traffic is crowdsourced. MBD analytics deals with massive amounts of data collected by millions of mobile devices. Next, we discuss the main characteristics of MBD that complicate data analytics and learning on MBD compared to small datasets.

Challenges of MBD Analytics

Figure 1b shows the main recent technologies that have produced the challenging MBD era: large-scale and high-speed mobile networks, portability, and crowdsourcing. Each technology contributes to forming the MBD characteristics in the following ways.

Large-scale and high-speed mobile networks: The growth of mobile devices and high-speed mobile networks (e.g., WiFi and cellular networks) introduces massive and increasingly contentious mobile data traffic. This is reflected in the following MBD aspects:

- *MBD is massive (volume):* In 2015, 3.7 exabytes of mobile data was generated per month, which is expected to increase through the coming years [1].
- *MBD is generated at increasing rates (velocity):* MBD flows at a high rate, which impacts the latency in serving mobile users. Long queuing time of requests results in less satisfied users and increased cost of late decisions.

Portability: Each mobile device is free to move independently among many locations. Therefore, *MBD is non-stationary (volatility)*. Due to portability, the time duration in which the collected data is valid for decision making can be relatively short. MBD analytics should be frequently executed to cope with the newly collected data samples.

Crowdsourcing: A remarkable trend of mobile applications is crowdsourcing for pervasive sensing, which includes massive data collection from many participating users. Crowdsensing differs from conventional mobile sensing systems as the sensing devices are not owned by one institution but instead by many individuals from different places. This has introduced the following MBD challenges:

- *MBD quality is not guaranteed (veracity)*: This aspect is critical for assessing the quality uncertainty of MBD as mobile systems do not directly manage the sensing process of mobile devices. Since most mobile data is crowdsourced, MBD can contain low-quality and missing data samples due to noise, malfunctioning or uncalibrated sensors of mobile devices, and even intruders (e.g., badly labeled crowdsourced data). These low-quality data points affect the analytical accuracy of MBD.
- *MBD is heterogeneous (variety)*: The variety of MBD arises because the data traffic comes from many spatially distributed data sources (i.e., mobile devices). Also, MBD comes in different data types due to the many sensors that mobile devices support. For example, a triaxial accelerometer generates proper acceleration measurements, while a light sensor generates illumination values.

MBD analytics (*value*) is mainly about extracting knowledge and patterns from MBD. In this way, MBD can be utilized to provide better services to mobile users and create revenue for mobile businesses. The next section discusses deep learning as a solid tool in MBD analytics.

Deep Learning in MBD Analytics

Deep learning is a new branch of machine learning that can solve a broad set of complex problems in MBD analytics (e.g., classification and regression). A *deep learning model* consists of simulated neurons and synapses that can be trained to learn hierarchical features from existing MBD samples. The result-

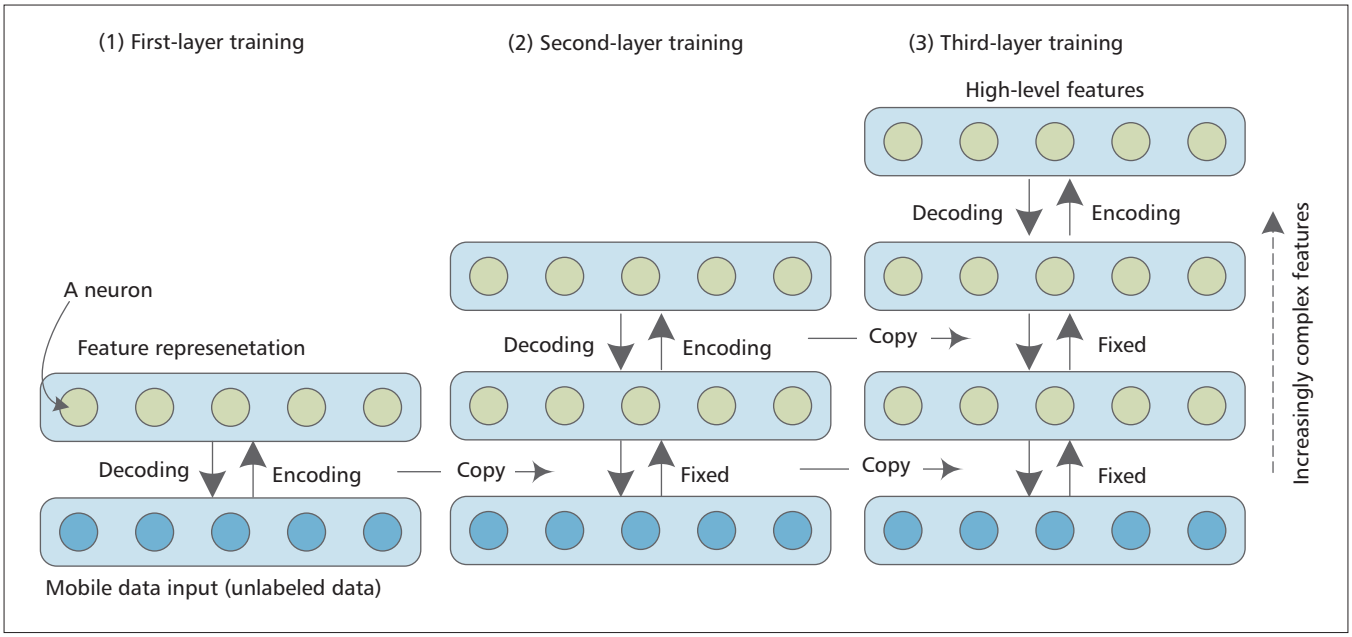

Figure 2. Generative layer-wise training of a deep model. Each layer applies nonlinear transformation to its input vector and produces intrinsic features at its output.

ing deep model can generalize and process unseen streaming MBD samples.

For simplicity, we present a general discussion of deep learning methods without focusing on the derivations of particular techniques. Nonetheless, we refer interested readers to more technical papers of deep belief networks [6] and stacked denoising autoencoders [7]. A deep model can be scaled to contain many hidden layers and millions of parameters, which are difficult to train at once. Instead, greedy layer-by-layer learning algorithms [6, 7] have been proposed that basically work as follows.

Generative layer-wise pre-training: This stage requires only unlabeled data, which is often abundant and cheap to collect in mobile systems using crowdsourcing. Figure 2 shows the layer-wise tuning of a deep model. First, one layer of neurons is trained using the unlabeled data samples. To learn the input data structure, each layer includes encoding and decoding functions. The encoding function uses the input data and the layer parameters to generate a set of new features. Then the decoding function uses the features and the layer parameters to produce a reconstruction of the input data. As a result, a first set of features is generated at the output of the first layer. Then a second layer of neurons is added on top of the first layer, where the output of the first layer is fed as input of the second layer. This process is repeated by adding more layers until a suitable deep model is formed. Accordingly, more complex features are learned at each layer based on the features that were generated at its lower layer.

Discriminative fine-tuning: The model's parameters, which are initialized in the first step, are then slightly fine-tuned using the available set of labeled data to solve the problem at hand.

Deep Learning Advantages in MBD Analytics

Deep learning provides solid learning models for MBD analytics. This argument can be supported with the following advantages of using deep learning in MBD analytics.

Deep learning scores highly accurate results, which are a top priority for growing mobile systems. Highly accurate results of MBD analytics are required for sustainable business and effective decisions. For example, poor fraud detection results in expensive loss of income for mobile systems. Deep learning models have been reported as state-of-the-art methods to solve many MBD tasks. For example, the authors in [8] propose a method for indoor localization using deep learning and channel state information. In [9], deep learning is successfully applied to inference tasks in mobile sensing (e.g., activity and emotion recognition, and speaker identification).

Deep learning generates intrinsic features that are required in MBD analytics. A *feature* is a measurement attribute extracted from sensory data to capture the underlying phenomena being observed and enable more effective MBD analytics. Deep learning can automatically learn high-level features from MBD, eliminating the need for the handcrafted features in conventional machine learning methods.

Deep learning can learn from unlabeled mobile data, which minimizes the data labeling effort. In most mobile systems, labeled data is limited, as manual data annotation requires expensive human intervention, which is both costly and time consuming. On the other hand, unlabeled data samples are abundant and cheap to collect. Deep learning models utilize unlabeled data samples for generative data exploration during a pre-training stage. This minimizes the need for labeled data during MBD analytics.

Multimodal deep learning. The "variety" aspect of MBD leads to multiple data modalities of multiple sensors (e.g., accelerometer samples, audio, and images). Multimodal deep learning [10] can learn from multiple modalities and heterogeneous input signals.

Deep Learning Challenges in MBD Analytics

Discussing MBD in terms of volume only and beyond the analytical and profit perspectives is incomplete and restricted. Collecting MBD is unprofitable unless suitable learning methods and analytics are utilized to extract meaningful information and patterns. Deep learning in MBD analytics is slow and can take a few days of processing time, which does not meet the operation requirements of most modern mobile systems. This is due to the following challenges.

The curse of dimensionality: MBD comes with volume and velocity related challenges. Historically, data analytics on small amounts of collected data (a.k.a. random sampling) was utilized. Despite the low computational burden of random sampling, it suffers from poor performance on unseen streaming

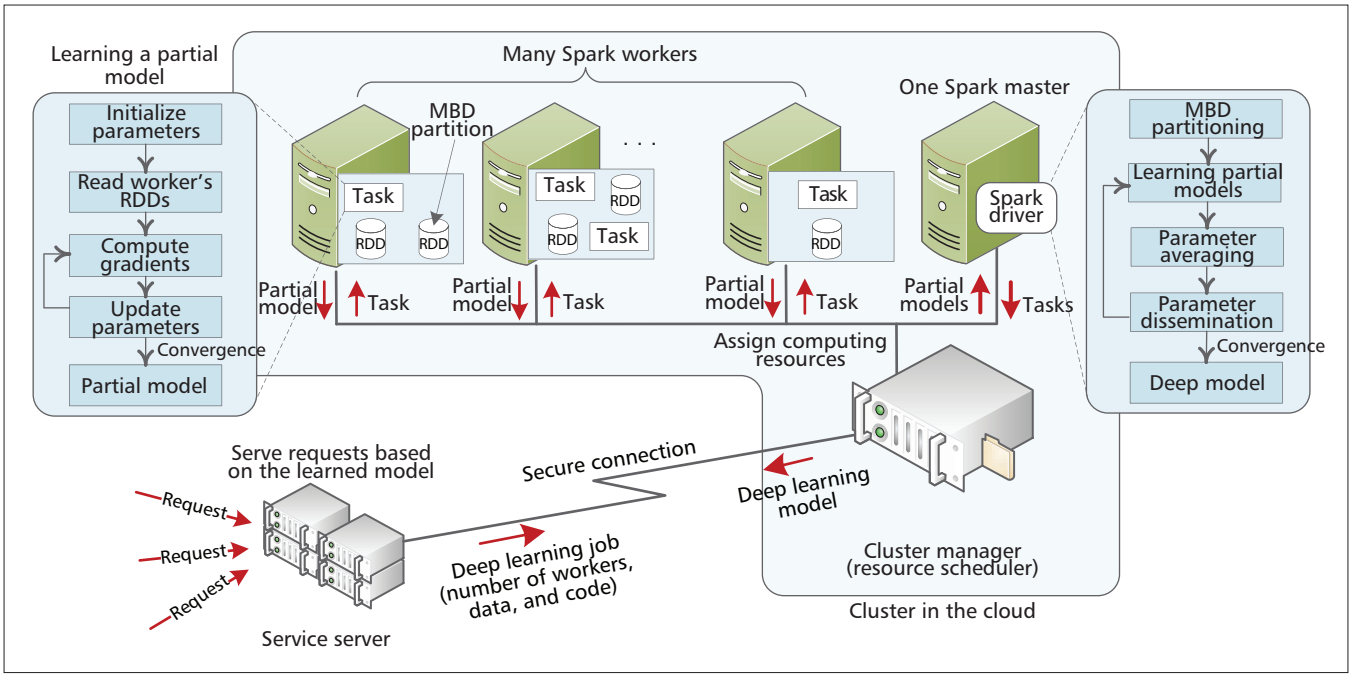

Figure 3. A Spark-based framework for distributed deep learning in MBD analytics.

samples. This performance problem is typically avoided by using the full set of available big data samples, which significantly increases the computational burdens.

Large-scale deep models: To fully capture the information on MBD and avoid underfitting, deep learning models should contain millions of free parameters; for example, a 5-layer deep model with 2000 neurons per layer contains around 20 million parameters. Models' free parameters are optimized using gradient-based learning [6, 7], which is computationally expensive for large-scale deep models.

Time-varying deep models: In mobile systems, the continuous adaptation of deep models over time is required due to the volatility characteristic of MBD.

To tackle these challenges, we next describe a scalable framework for MBD analytics using deep learning models and Apache Spark.

A Spark-Based Deep Learning Framework for MBD Analytics

Learning deep models in MBD analytics is slow and computationally demanding. Typically, this is due to the large number of parameters of deep models and the large number of MBD samples. Figure 3 shows the proposed architecture for learning deep models on MBD with Apache Spark. Apache Spark [2] is an open source platform for scalable MapReduce computing on clusters. The main goal of the proposed framework is speeding up MBD decision making by parallelizing the learning of deep models to a high-performance computing cluster. In short, the parallelization of a deep model is performed by slicing the MBD into many partitions. Each partition is contained in a resilient distributed dataset (RDD) that provides an abstraction for data distribution by the Spark engine. Besides data caching, RDDs of a Spark program natively support fault-tolerant executions and recover the program operations at worker nodes.

In short, our Spark-based framework consists of two main components:

- A Spark master
- One or more Spark workers

The master machine initializes an instance of the Spark driver that manages the execution of many partial models in a group of Spark workers. At each iteration of the deep learning algorithm (Fig. 2), each worker node learns a partial deep model on a small partition of the MBD and sends the computed parameters back to the master node. Then, the master node reconstructs a master deep model by averaging the computed partial models of all executor nodes.

Parallelized Learning Collections

Learning deep models can be performed in two main steps:

- Gradient computation
- Parameter update [6, 7] (for the mathematical derivation)

In the first step, the learning algorithm iterates through all data batches independently to compute gradient updates (i.e., the rate of change) of the model's parameters. In the second step, the model's parameters are updated by averaging the computed gradient updates on all data batches. These two steps fit the learning of deep models in the MapReduce programming model [11, 12]. In particular, the parallel gradient computation is realized as a Map procedure, while the parameter update step reflects the Reduce procedure. The iterative MapReduce computing of deep learning on Apache Spark is performed as follows.

MBD partitioning: The overall MBD is first split into many partitions using the *parallelize()* application programming interface (API) of Spark. The resulting MBD partitions are stored into RDDs and distributed to the worker nodes. These RDDs are crucial to speed up the learning of deep models as the memory data access latency is significantly shorter than the disk data operations.

Deep learning parallelism: The solution of a deep learning problem depends on the solution of smaller instances of the same learning problem with smaller datasets. In particular, the deep learning job is divided into learning stages. Each learning stage contains a set of independent MapReduce iterations where the solution of one iteration is the input for the next iteration. During each MapReduce iteration, a partial model is trained on a separate partition of the available MBD as follows:

- *Learning partial models*: Each worker node computes the gradient updates of its partitions of the MBD (a.k.a. the Map procedure). During this step, all Spark workers execute the same Map task in parallel but on different partitions of

the MBD. In this way, the expensive gradient computation task of the deep model learning is divided into many parallel sub-tasks.

- *Parameter averaging*: Parameters of the partial models are sent to the master machine to build a master deep model by averaging the parameter calculation of all Spark workers (a.k.a. the Reduce procedure).
- *Parameter dissemination*: The resulting master model after the Reduce procedure is disseminated to all worker nodes. A new MapReduce iteration is then started based on the updated parameters. This process is continued until the learning convergence criterion is satisfied.

As a result, a well-tuned deep learning model is generated that can be used to infer information and patterns from streaming requests. In the following, we discuss how the proposed framework helps tackle the key characteristics of MBD.

Discussion

The proposed framework is grounded over deep learning and Apache Spark technologies to perform effective MBD analytics. This integration tackles the challenging characteristics of MBD as follows.

Deep learning: Deep learning addresses the value and variety aspects of MBD. First, deep learning in MBD analytics helps in understanding raw MBD. Therefore, deep learning effectively addresses the value aspect of MBD. MBD analytics, as discussed in this article, is integral in providing user-customized mobile services. Second, deep learning enables learning from multimodal data distributions [10] (e.g., concatenated input from accelerometer and light sensors), which is important for the variety issue of MBD.

Apache Spark: The main role of the Spark platform in the proposed framework is tackling the volume, velocity, and volatility aspects of MBD. Essentially, the Spark engine tackles the volume aspect by parallelizing the learning task into many sub-tasks, each performed on a small partition of the overall MBD. Therefore, no single machine is required to process the massive MBD volume as one chunk. Similarly, the Spark engine tackles the velocity point through its streaming extensions, which enable fast and high-throughput processing of streaming data. Finally, the volatility aspect is addressed by significantly speeding up the training of deep models. This ensures that the learned model reflects the latest dynamics of the mobile system.

The proposed framework does not directly tackle the veracity aspect of MBD. This quality aspect requires domain experts to design conditional routines to check the validity of crowdsourced data before being added to central MBD storage.

Prototyping a Context-Aware Activity Recognition System

Context awareness [3, 5] has high impact on understanding MBD by describing the circumstances in which the data was collected to provide personalized mobile experience to end users (e.g., targeted advertising, healthcare, and social services). A *context* contains attributes of information to describe the sensed environment such as performed human activities, surrounding objects, and locations. A *context learning model* is a program that defines the rules of mapping between raw sensory data and the corresponding context labels (e.g., mapping accelerometer signals to activity labels). This section describes a proof-of-concept case study in which we consider a context-aware activity recognition system, such as detecting walking, jogging, climbing stairs, sitting, standing, and lying down. We use a real-world dataset during the training of deep activity recognition models.

Problem Statement

Accelerometers are sensors that measure proper acceleration of an object due to motion and gravitational force. Modern mobile devices are widely equipped with tiny accelerometer circuits, which are produced from electromechanically sensitive elements and generate electrical signals in response to any mechanical motion. The proper acceleration is distinctive from coordinate acceleration in classical mechanics. The latter measures the rate of change of velocity, while the former measures acceleration relative to a free fall; that is, the proper acceleration of an object in a free fall is zero.

Consider a mobile device with an embedded accelerometer sensor that generates proper acceleration samples. Activity recognition is applied to time series data frames formulated using a sliding and overlapping window. The number of time-series samples depends on the accelerometer's sampling frequency (in Hertz) and windowing length (in seconds). At time t, the activity recognition classifier $f : \mathbf{x}_t \rightarrow \mathcal{S}$ matches the framed acceleration data $\mathbf{x}_t$ with the most probable activity label from the set of supported activity labels $\mathcal{S} = \{1, 2, \ldots, N\}$, where N is the number of supported activities in the activity detection component.

Conventional approaches to recognizing activities require handcrafted features (e.g., statistical features) [3], which are expensive to design, require domain expert knowledge, and generalize poorly to support more activities. To avoid this, a deep activity recognition model learns not only the mapping between raw acceleration data and the corresponding activity label, but also a set of meaningful features that are superior to handcrafted features.

Experimental Setup

In this section, we use the Actitracker dataset [13], which includes accelerometer samples of 6 conventional activities (walking, jogging, climbing stairs, sitting, standing, and lying down) from 563 crowdsourcing users. Figure 4a plots accelerometer signals of the six different activities. Clearly, high-frequency signals are sampled for activities with active body motion (e.g., walking, jogging, and climbing stairs). On the other hand, low-frequency signals are collected during semi-static body motions (e.g., standing, sitting, and lying down). The data is collected using mobile phones with 20 Hz sampling rate, and it contains both labeled and unlabeled data of 2,980,765 and 38,209,772 samples, respectively. This is a real-world example of the limited number of labeled data compared to unlabeled data as data labeling requires manual human intervention. The data is framed using a 10 s windowing function that generates 200 samples of time-series samples. We first pretrain deep models on the unlabeled data samples only, and then fine-tune the models on the labeled dataset. To enhance the activity recognition performance, we use the spectrogram of the acceleration signal as input of the deep models. Basically, different activities contain different frequency contents that reflect the body dynamics and movements.

We implemented the proposed framework on a shared cluster system (https://www.acrc.a-star.edu.sg) running the load sharing facility (LSF) management platform and RedHat Linux. Each node has 8 cores (Intel Xeon 5570 CPU with clock speed of 2.93 Ghz) and a total of 24 GB RAM. In our experiments, we set the cores in multiples of 8 to allocate the entire node's resources. One partial model learning task is initialized for each computing core. Each task learns using a data batch consisting of 100 samples for 100 iterations. Clearly, increasing the number of cores results in quicker training of deep models. Finally, it is important to note that distributed deep learning is a strong type of regularization. Thus, regularization techniques, such as the sparsity and dropout constraints, are not recommended to avoid the problem of underfitting.

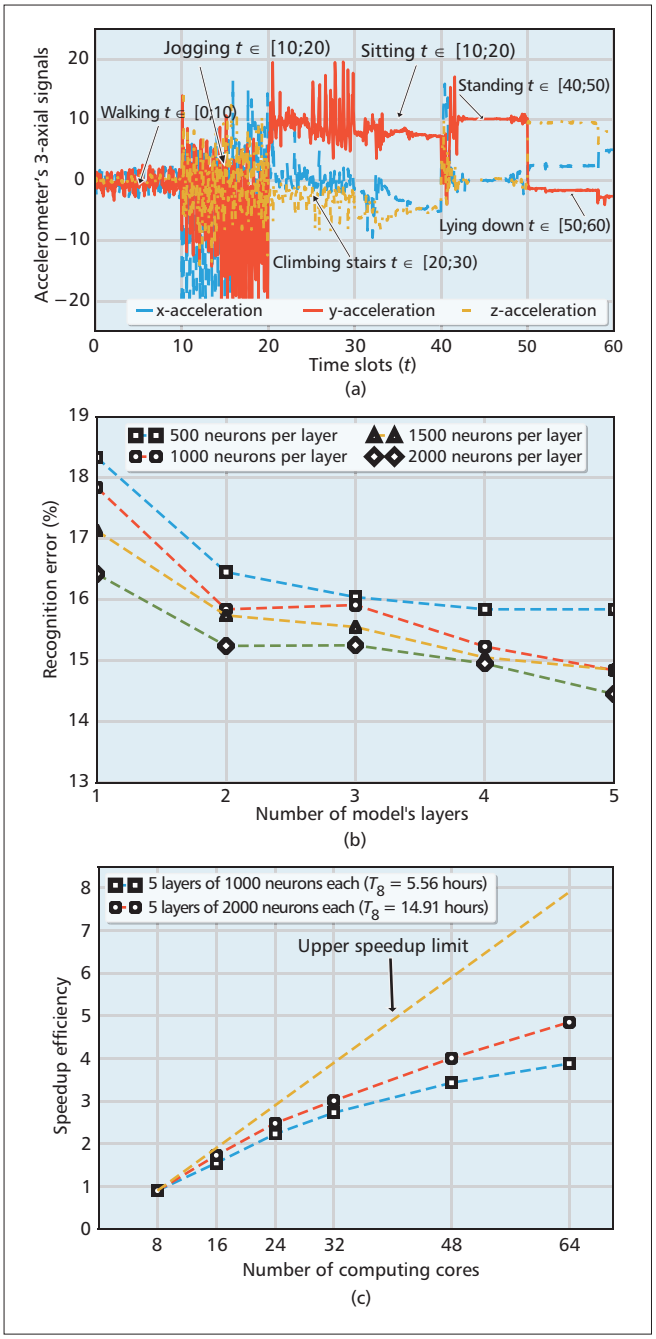

Figure 4. Experimental analysis: a) accelerometer signal of different human activities; b) recognition accuracy of deep learning models under different deep model setups; c) speedup of learning deep models using the Spark-based framework under different computing cores. The upper speedup limit is achieved under full CPU utilization and zero communication overhead.

Experimental Results

The Impact of Deep Models: Figure 4b shows the activity recognition error under different setups of deep models (number of hidden layers and number of neurons at each layer). Specifically, the capacity of a deep model to capture MBD structures is increased when using deeper models with more layers and neurons. Nonetheless, using deeper models involves a significant increase in the learning algorithm's computational burdens and time. An accuracy comparison of deep activity recognition models and other conventional methods is shown in Table 1. In short, these results clarify that:

Method	Recognition error (%)
Multilayer perceptrons	32.2
Instance-based learning	31.6
Random forests	24.1
Deep learning (5 layers of 2000 neurons each)	14.4

Table 1. Activity recognition error of deep learning and other conventional methods used in [4]. The conventional methods use handcrafted statistical features.

- Deep models are superior to existing shallow context learning models.
- The learned hierarchical features of deep models eliminate the need for handcrafted statistical features in conventional methods.

In our implementation, we use early stopping to track the model capacity during training, select the best parameters of deep models, and avoid overfitting. Underfitting is typically avoided by using deeper models and more neurons per layer (e.g., 5 layers with 2000 neurons per layer). Next, a speedup analysis is presented to show the importance of the Spark-based framework for learning deep models on MBD.

The Impact of Computing Cores: The main performance metric of cluster-based computing is the task speedup metric. In particular, we compute the speedup efficiency as T_8/T_M, where T_8 is the computing time of one machine with 8 cores, and T_M is the computing time under different computing power. Figure 4c shows the speedup in learning deep models when the number of computing cores is varied. As the number of cores increases, the learning time decreases. For example, learning a deep model of 5 layers with 2000 neurons per layer can be trained in 3.63 h with 6 Spark workers. This results in speedup efficiency of 4.1 as compared to single-machine computing, which takes 14.91 h.

MBD Veracity: A normalized confusion matrix of a deep model is shown in Fig. 5. This confusion matrix shows the high performance of deep models on a per-activity basis (high scores at the diagonal entries). The incorrect detection of the "sitting" activity instead of the "lying down" activity is typically due to the different procedures in performing the activities by crowdsourcing users. This gives a real-world example of the veracity characteristic of MBD (i.e., uncertainties in MBD collection).

In the next section, we identify some notable future research directions in MBD collection, labeling, and economics.

Future Work

Based on the proposed framework, the following future work can be further pursued.

Crowd Labeling of MBD

A major challenge facing MBD analysts is the limited amounts of labeled data samples as data labeling is typically a manual process. An important research direction is proposing crowd labeling methods for MBD. Crowd labeling can be designed under two main schemes:

- Paid crowd labeling
- Embedded crowd labeling

In paid crowd labeling, the crowdsourcing mobile users annotate mobile data and are accordingly paid based on their labeling performance and speed. Under this paid scheme, optimal budget allocation methods are required. In embedded crowd labeling, data labeling can also be achieved by adding label-

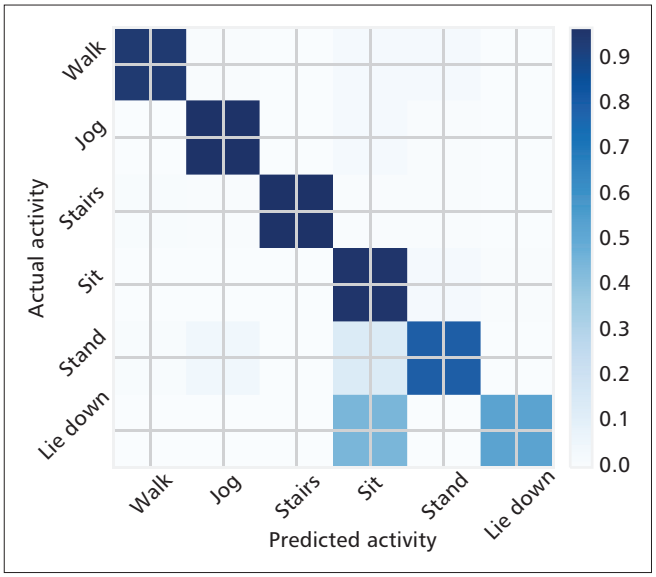

Figure 5. Normalized confusion matrix of a deep model (5 layers of 2000 neurons each). The diagonal elements represent correct activity recognition.

ing tasks within mobile application functional routines (e.g., CAPTCHA-based image labeling) [14]. Here, the mobile users can access more functions of a mobile application by indirectly helping in the data labeling process. More work is required on designing innovative methods for embedded crowd labeling without disturbing the user experience or harming the mobile application's main functionality.

Economics of MBD

MBD, as discussed in this article, is about extracting meaningful information and patterns from raw mobile data. This information is used during decision making and to enhance existing mobile services. An important research direction is proposing business models (e.g., pricing and auction design [15]) for selling and buying MBD among mobile organizations and parties.

Privacy and MBD Collection

As MBD is people-centric, mobile users would be concerned about the risks of sharing their personal mobile data with a service server. Thus, a low percentage of users will opt out of sharing their personal data unless trustworthy privacy mechanisms are applied. Meanwhile, anonymized data collection (i.e., data that could not be used to identify individuals) is adopted by many services. An alternative research direction is proposing fair data exchange models that encourage the sharing of mobile data in return for rewarding points (e.g., premium membership points).

Conclusions

In this article, we have presented and discussed a scalable Spark-based framework for deep learning in mobile big data analytics. The framework enables the tuning of deep models with many hidden layers and millions of parameters on a computing cluster. Typically, deep learning provides a promising learning tool for adding value by learning intrinsic features from raw mobile big data. The framework has been validated using a large-scale activity recognition system as a case study. Finally, important research directions on mobile big data have been outlined.

Acknowledgment

This work was supported in part by the A*STAR Computational Resource Centre through the use of its high-performance computing facilities. It was also supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIP) (2014R1A5A1011478), Singapore MOE Tier 1 (RG18/13 and RG33/12) and MOE Tier 2 (MOE2014-T2-2-015 ARC 4/15), and the U.S. National Science Foundation under Grants US NSF ECCS-1547201, CCF-1456921, CNS-1443917, ECCS-1405121, and NSFC61428101. The authors thank Ahmed Selim, Trinity College Dublin, for valuable discussions in the early stages of the study.

References

[1] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2015–2020," White Paper, 2016.
[2] Apache Spark, "Apache Spark–Lightning-Fast Cluster Computing," 2016, accessed 19 Feb.2016; http://spark.apache.org.
[3] O. D. Lara and M. A. Labrador, "A Survey on Human Activity Recognition Using Wearable Sensors," *IEEE Commun. Surveys & Tutorials*, vol. 15, no. 3, 2013, pp. 1192–1209.
[4] G. M. Weiss and J. W. Lockhart, "The Impact of Personalization on Smartphone-Based Activity Recognition," *AAAI Wksp. Activity Context Representation: Techniques and Languages*, 2012.
[5] C. Perera *et al.*, "Context Aware Computing for the Internet of Things: A Survey," *IEEE Commun. Surveys & Tutorials*, vol. 16, no. 1, 2014, pp. 414–54.
[6] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, 2006, pp. 1527–54.
[7] P. Vincent *et al.*, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *J. Machine Learning Research*, vol. 11, 2010, pp. 3371–3408.
[8] X. Wang *et al.*, "Deepfi: Deep Learning for Indoor Fingerprinting Using Channel State Information," *IEEE Wireless Commun. and Networking Conf.*, Mar. 2015, pp. 1666–71.
[9] N. D. Lane and P. Georgiev, "Can Deep Learning Revolutionize Mobile Sensing?," *Proc. 16th ACM Int'l. Wksp. Mobile Computing Systems and Applications*, 2015, pp. 117–22.
[10] J. Ngiam *et al.*, "Multimodal Deep Learning," *Proc. 28th Int'l. Conf. Machine Learning*, 2011, pp. 689–96.
[11] J. Dean *et al.*, "Large Scale Distributed Deep Networks," *Advances in Neural Information Processing Systems*, 2012, pp. 1223–31.
[12] K. Zhang and X.-w. Chen, "Large-Scale Deep Belief Nets with MapReduce," *IEEE Access*, vol. 2, 2014, pp. 395–403.
[13] J. W. Lockhart *et al.*, "Design Considerations for the WISDM Smart Phone-Based Sensor Mining Architecture," *Proc. 5th ACM Int'l. Wksp. Knowledge Discovery from Sensor Data*, 2011, pp. 25–33.
[14] L. Von Ahn *et al.*, "reCAPTCHA: Human-Based Character Recognition via Web Security Measures," *Science*, vol. 321, no. 5895, 2008, pp. 1465–68.
[15] P. Klemperer, *Auctions: Theory and Practice*, ser. Princeton paperbacks, Princeton Univ. Press, 2004.

Biographies

MOHAMMAD ABU ALSHEIKH [S'14] (stumyhaa@i2r.a-star.edu.sg) received his B.Eng. in computer systems engineering from Birzeit University, Palestine, in 2011. Between 2010 and 2012, he was a software engineer working on developing robust web services, Ajax-based web components, and smartphone applications. He is currently a Ph.D. candidate in the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include machine learning in big data analytics, mobile sensing technologies, and sensor-based activity recognition.

DUSIT NIYATO [M'09, SM'15] (dniyato@ntu.edu.sg) is currently an associate professor in the School of Computer Engineering at Nanyang Technological University, Singapore. He received his B.E. from King Mongkut's Institute of Technology Ladkrabang (KMITL) in 1999. He obtained his Ph.D. in electrical and computer engineering from the University of Manitoba, Canada, in 2008. His research interests are in the area of radio resource management in cognitive radio networks and energy harvesting for wireless communication.

SHAOWEI LIN (shaowei_lin@sutd.edu.sg) received his Ph.D. in mathematics under Bernd Sturmfels in 2011 from the University of California, Berkeley, where he analyzed singularities in statistical models over large datasets through the lens of modern algebraic geometry. This work was continued at Stanford University in a one-year DARPA postdoctoral collaboration with Andrew Ng's lab to explore mathematical challenges in deep learning. In 2012, he returned to Singapore to join the Institute for Infocomm Research (A*STAR), where he started the Sense-making Group in the Sense and Sense-abilities (S&S) programme. The group focused on exploiting machine learning techniques in sensor networks to create resource-efficient algorithms that exhibit higher-order intelligence. Before joining Singapore University of Technology and Design (SUTD), he oversaw deep science activities in S&S as the deputy head for research.

HWEE-PINK TAN [S'00, M'04, SM'14] (hptan@smu.edu.sg) is currently an associate professor of information systems (practice) at Singapore Management University (SMU). He also holds the concurrent appointment of academic director

of the SMU-TCS iCity Lab at SMU, where he leads a team of nine technology and social science researchers to bring together Internet of Things technologies, and social-behavioral research to enable and sustain aging-in-place, leading, in a broader sense, to intelligent and inclusive societies, in close partnership with A*STAR, TCS, various government agencies, as well as voluntary welfare organizations. Prior to joining SMU in March 2015, he spent seven years at A*STAR, where he was a senior scientist and concurrently the SERC Programme Manager for the A*STAR Sense and Sense-abilities Programme. In this programme, he led a team of 30 full-time research scientists and engineers to design, pilot, and evaluate architectures to support large-scale and heterogeneous sensor systems to enable smart city applications. In recognition of his contributions, he was awarded the I2R Role Model Award in 2012 and 2013, and the A*STAR Most Inspiring Mentor, TALENT, and Borderless Awards in 2014.

ZHU HAN [S'01, M'04, SM'09, F'14] (zhan2@uh.edu) received his B.S. degree in electronic engineering from Tsinghua University in 1997, and his M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, in 1999 and 2003, respectively. From 2000 to 2002, he was an R&D engineer at JDSU, Germantown, Maryland. From 2003 to 2006, he was a research associate at the University of Maryland. From 2006 to 2008, he was an assistant professor at Boise State University, Idaho. Currently, he is a professor in the Electrical and Computer Engineering Department as well as the Computer Science Department at the University of Houston, Texas. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, wireless multimedia, security, and smart grid communication. He received an NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the EURASIP Best Paper Award for the *Journal on Advances in Signal Processing* in 2015, the 2016 IEEE Leonard G. Abraham Prize in the field of Communications Systems, and several best paper awards at IEEE conferences, and is currently an IEEE Communications Society Distinguished Lecturer.