Deep Learning in Mobile and Wireless Networking: A Survey
Chaoyun Zhang, Paul Patras, Senior Member, IEEE, and Hamed Haddadi

Abstract—The rapid uptake of mobile devices and the rising popularity of mobile applications and services pose unprecedented demands on mobile and wireless networking infrastructure. Upcoming 5G systems are evolving to support exploding mobile traffic volumes, real-time extraction of fine-grained analytics, and agile management of network resources, so as to maximize user experience. Fulfilling these tasks is challenging, as mobile environments are increasingly complex, heterogeneous, and evolving. One potential solution is to resort to advanced machine learning techniques, in order to help manage the rise in data volumes and algorithm-driven applications. The recent success of deep learning underpins new and powerful tools that tackle problems in this space.

In this paper we bridge the gap between deep learning and mobile and wireless networking research, by presenting a comprehensive survey of the crossovers between the two areas. We first briefly introduce essential background and state-of-the-art in deep learning techniques with potential applications to networking. We then discuss several techniques and platforms that facilitate the efficient deployment of deep learning onto mobile systems. Subsequently, we provide an encyclopedic review of mobile and wireless networking research based on deep learning, which we categorize by different domains. Drawing from our experience, we discuss how to tailor deep learning to mobile environments. We complete this survey by pinpointing current challenges and open future directions for research.


I. INTRODUCTION

INTERNET connected mobile devices are penetrating every aspect of individuals’ life, work, and entertainment. The increasing number of smartphones and the emergence of ever-more diverse applications trigger a surge in mobile data traffic. Indeed, the latest industry forecasts indicate that the annual worldwide IP traffic consumption will reach 3.3 zettabytes ($10^{15}$ MB) by 2021, with smartphone traffic exceeding PC traffic by the same year [1]. Given the shift in user preference towards wireless connectivity, current mobile infrastructure faces great capacity demands. In response to this increasing demand, early efforts propose to agilely provision resources [2] and tackle mobility management distributively [3]. In the long run, however, Internet Service Providers (ISPs) must develop intelligent heterogeneous architectures and tools that can spawn the 5th generation of mobile systems (5G) and gradually meet more stringent end-user application requirements [4], [5].

The growing diversity and complexity of mobile network architectures has made monitoring and managing the multitude of network elements intractable. Therefore, embedding versatile machine intelligence into future mobile networks is drawing unparalleled research interest [6], [7]. This trend is reflected in machine learning (ML) based solutions to problems ranging from radio access technology (RAT) selection [8] to malware detection [9], as well as the development of networked systems that support machine learning practices (e.g., [10], [11]). ML enables systematic mining of valuable information from traffic data and automatically uncover correlations that would otherwise have been too complex to extract by human experts [12]. As the flagship of machine learning, deep learning has achieved remarkable performance in areas such as computer vision [13] and natural language processing (NLP) [14]. Networking researchers are also beginning to recognize the power and importance of deep learning, and are exploring its potential to solve problems specific to the mobile networking domain [15], [16].

Embedding deep learning into the 5G mobile and wireless networks is well justified. In particular, data generated by mobile environments are increasingly heterogeneous, as these are usually collected from various sources, have different formats, and exhibit complex correlations [17]. As a consequence, a range of specific problems become too difficult or impractical for traditional machine learning tools (e.g., shallow neural networks). This is because (i) their performance does not improve if provided with more data [18] and (ii) they cannot handle highly dimensional state/action spaces in control problems [19]. In contrast, big data fuels the performance of deep learning, as it eliminates domain expertise and instead employs hierarchical feature extraction. In essence this means information can be distilled efficiently and increasingly abstract correlations can be obtained from the data, while reducing the pre-processing effort. Graphics Processing Unit (GPU)-based parallel computing further enables deep learning to make inferences within milliseconds. This facilitates network analysis and management with high accuracy and in a timely manner, overcoming the run-time limitations of traditional mathematical techniques (e.g. convex optimization, game theory, meta heuristics).

Despite growing interest in deep learning in the mobile networking domain, existing contributions are scattered across different research areas and a comprehensive survey is lacking. This article fills this gap between deep learning and mobile and wireless networking, by presenting an up-to-date survey of
research that lies at the intersection between these two fields. Beyond reviewing the most relevant literature, we discuss the key pros and cons of various deep learning architectures, and outline deep learning model selection strategies, in view of solving mobile networking problems. We further investigate methods that tailor deep learning to individual mobile networking tasks, to achieve the best performance in complex environments. We wrap up this paper by pinpointing future research directions and important problems that remain unsolved and are worth pursuing with deep neural networks. Our ultimate goal is to provide a definite guide for networking researchers and practitioners, who intend to employ deep learning to solve problems of interest.

**Survey Organization:** We structure this article in a top-down manner, as shown in Figure 1. We begin by discussing work that gives a high-level overview of deep learning, future mobile networks, and networking applications built using deep learning, which help define the scope and contributions of this paper (Section II). Since deep learning techniques are relatively new in the mobile networking community, we provide a basic deep learning background in Section III, highlighting immediate advantages in addressing mobile networking problems. There exist many factors that enable implementing deep learning for mobile networking applications (including dedicated deep learning libraries, optimization algorithms, etc.). We discuss these enablers in Section IV, aiming to help mobile network researchers and engineers in choosing the right software and hardware platforms for their deep learning deployments.

In Section V, we introduce and compare state-of-the-art deep learning models and provide guidelines for model selection toward solving networking problems. In Section VI we review recent deep learning applications to mobile and wireless networking, which we group by different scenarios ranging from mobile traffic analytics to security, and emerging applications. We then discuss how to tailor deep learning models to mobile networking problems (Section VII) and conclude this article with a brief discussion of open challenges, with a view to future research directions (Section VIII).\(^1\)

## II. RELATED HIGH-LEVEL ARTICLES AND THE SCOPE OF THIS SURVEY

Mobile networking and deep learning problems have been researched mostly independently. Only recently crossovers between the two areas have emerged. Several notable works paint a comprehensives picture of the deep learning and/or mobile networking research landscape. We categorize these works into (i) pure overviews of deep learning techniques, (ii) reviews of analyses and management techniques in modern mobile networks, and (iii) reviews of works at the intersection between deep learning and computer networking. We summarize these earlier efforts in Table II and in this section discuss the most representative publications in each class.

### A. Overviews of Deep Learning and its Applications

The era of big data is triggering wide interest in deep learning across different research disciplines [28]–[31] and a

\(^1\)We list the abbreviations used throughout this paper in Table I.

### TABLE I: List of abbreviations in alphabetical order.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>3G</td>
<td>5th Generation mobile networks</td>
</tr>
<tr>
<td>A3C</td>
<td>Asynchronous Advantage Actor-Critic</td>
</tr>
<tr>
<td>AdaNet</td>
<td>Adaptive learning of neural Network</td>
</tr>
<tr>
<td>AE</td>
<td>Auto Encoder</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>AMP</td>
<td>Approximate Message Passing</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>ASR</td>
<td>Automatic Speech Recognition</td>
</tr>
<tr>
<td>BSC</td>
<td>Base Station Controller</td>
</tr>
<tr>
<td>BP</td>
<td>Back-Propagation</td>
</tr>
<tr>
<td>CDR</td>
<td>Call Detail Record</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>ConvLSTM</td>
<td>Convolutional Long Short-Term Memory</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CSI</td>
<td>Channel State Information</td>
</tr>
<tr>
<td>CUDA</td>
<td>Compute Unified Device Architecture</td>
</tr>
<tr>
<td>cuDNN</td>
<td>CUDA Deep Neural Network library</td>
</tr>
<tr>
<td>D2D</td>
<td>Device to Device communication</td>
</tr>
<tr>
<td>DAE</td>
<td>Denoising Auto-Encoder</td>
</tr>
<tr>
<td>DBN</td>
<td>Deep Belief Network</td>
</tr>
<tr>
<td>OFDM</td>
<td>Orthogonal Frequency-Division Multiplexing</td>
</tr>
<tr>
<td>DPPO</td>
<td>Distributed Proximal Policy Optimization</td>
</tr>
<tr>
<td>DQN</td>
<td>Deep Q-Network</td>
</tr>
<tr>
<td>DRL</td>
<td>Deep Reinforcement Learning</td>
</tr>
<tr>
<td>DT</td>
<td>Decision Tree</td>
</tr>
<tr>
<td>ELM</td>
<td>Extreme Learning Machine</td>
</tr>
<tr>
<td>GAN</td>
<td>Generative Adversarial Network</td>
</tr>
<tr>
<td>GP</td>
<td>Gaussian Process</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>GRU</td>
<td>Gate Recurrent Unit</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov Model</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IDS</td>
<td>Intrusion Detection System</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IoV</td>
<td>Internet of Vehicle</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LTE</td>
<td>Long-Term Evolution</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long Short-Term Memory</td>
</tr>
<tr>
<td>LSVRC</td>
<td>Large Scale Visual Recognition Challenge</td>
</tr>
<tr>
<td>MAC</td>
<td>Media Access Control</td>
</tr>
<tr>
<td>MDP</td>
<td>Markov Decision Process</td>
</tr>
<tr>
<td>MEC</td>
<td>Mobile Edge Computing</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>MLP</td>
<td>Multilayer Perceptron</td>
</tr>
<tr>
<td>MIMO</td>
<td>Multi-Input Multi-Output</td>
</tr>
<tr>
<td>MTSR</td>
<td>Mobile Traffic Super-Resolution</td>
</tr>
<tr>
<td>NFL</td>
<td>No Free Lunch theorem</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
<tr>
<td>NMT</td>
<td>Neural Machine Translation</td>
</tr>
<tr>
<td>NPU</td>
<td>Neural Processing Unit</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Components Analysis</td>
</tr>
<tr>
<td>PIR</td>
<td>Passive Infra-Red</td>
</tr>
<tr>
<td>QoE</td>
<td>Quality of Experience</td>
</tr>
<tr>
<td>RBM</td>
<td>Restricted Boltzmann Machine</td>
</tr>
<tr>
<td>ReLU</td>
<td>Rectified Linear Unit</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio Frequency Identification</td>
</tr>
<tr>
<td>RNC</td>
<td>Radio Network Controller</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>SARSA</td>
<td>State-Action-Reward-State-Action</td>
</tr>
<tr>
<td>SELU</td>
<td>Scaled Exponential Linear Unit</td>
</tr>
<tr>
<td>SGD</td>
<td>Stochastic Gradient Descent</td>
</tr>
<tr>
<td>SON</td>
<td>Self-Organising Network</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>TPU</td>
<td>Tensor Processing Unit</td>
</tr>
<tr>
<td>VAE</td>
<td>Variational Auto-Encoder</td>
</tr>
<tr>
<td>VR</td>
<td>Virtual Reality</td>
</tr>
<tr>
<td>WGAN</td>
<td>Wasserstein Generative Adversarial Network</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Network</td>
</tr>
</tbody>
</table>

\[^1\]We list the abbreviations used throughout this paper in Table I.
A growing number of surveys and tutorials are emerging (e.g. [23], [24]). LeCun et al. give a milestone overview of deep learning, introduce several popular models, and look ahead at the potential of deep neural networks [20]. Schmidhuber undertakes an encyclopedic survey of deep learning, likely the most comprehensive thus far, covering the evolution, methods, applications, and open research issues [21]. Liu et al. summarize the underlying principles of several deep learning models, and review deep learning developments in selected applications, such as speech processing, pattern recognition, and computer vision [22].

Arulkumaran et al. present several architectures and core algorithms for deep reinforcement learning, including deep Q-networks, trust region policy optimization, and asynchronous advantage actor-critic [26]. Their survey highlights the remarkable performance of deep neural networks in different control problem (e.g., video gaming, Go board game play, etc.). Similarly, deep reinforcement learning has also been surveyed in [78], where the authors shed more light on applications. Zhang et al. survey developments in deep learning for recommender systems [32], which have potential to play an important role in mobile advertising. As deep learning becomes increasingly popular, Goodfellow et al. provide a comprehensive tutorial of deep learning in a book that covers prerequisite knowledge, underlying principles, and popular applications [18].

B. Surveys on Future Mobile Networks

The emerging 5G mobile networks incorporate a host of new techniques to overcome the performance limitations of current deployments and meet new application requirements. Progress to date in this space has been summarized through
TABLE II: Summary of existing surveys, magazine papers, and books related to deep learning and mobile networking. The symbol √ indicates a publication is in the scope of a domain; ✗ marks papers that do not directly cover that area, but from which readers may retrieve some related insights. Publications related to both deep learning and mobile networks are shaded.

<table>
<thead>
<tr>
<th>Publication</th>
<th>One-sentence summary</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Machine learning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deep learning</td>
</tr>
<tr>
<td>Liu et al. [22]</td>
<td>A survey on deep learning and its applications.</td>
<td>✓</td>
</tr>
<tr>
<td>Deng et al. [23]</td>
<td>An overview of deep learning methods and applications.</td>
<td>✓</td>
</tr>
<tr>
<td>Goodfellow et al. [18]</td>
<td>An essential deep learning textbook.</td>
<td>✓</td>
</tr>
<tr>
<td>Hussein et al. [27]</td>
<td>A survey of imitation learning.</td>
<td>✓</td>
</tr>
<tr>
<td>Chen et al. [28]</td>
<td>An introduction to deep learning for big data.</td>
<td>✓</td>
</tr>
<tr>
<td>Najafabadi [29]</td>
<td>An overview of deep learning applications for big data analytics.</td>
<td>✓</td>
</tr>
<tr>
<td>Gheisari et al. [31]</td>
<td>A high-level literature review on deep learning for big data analytics.</td>
<td>✓</td>
</tr>
<tr>
<td>Yu et al. [33]</td>
<td>A survey on networking big data.</td>
<td>✓</td>
</tr>
<tr>
<td>Alsheikh et al. [34]</td>
<td>A survey on machine learning in wireless sensor networks.</td>
<td>✓</td>
</tr>
<tr>
<td>Tsai et al. [35]</td>
<td>A survey on data mining in IoT.</td>
<td>✓</td>
</tr>
<tr>
<td>Cheng et al. [36]</td>
<td>An introduction to mobile big data applications.</td>
<td>✓</td>
</tr>
<tr>
<td>Bkasiny et al. [37]</td>
<td>A survey on machine learning in cognitive radios.</td>
<td>✓</td>
</tr>
<tr>
<td>Andrews et al. [38]</td>
<td>An introduction and outlook of 5G networks.</td>
<td>✓</td>
</tr>
<tr>
<td>Eljah et al. [40]</td>
<td>A survey of 5G MIMO systems.</td>
<td>✓</td>
</tr>
<tr>
<td>Buzzi et al. [41]</td>
<td>A survey of 5G energy-efficient techniques.</td>
<td>✓</td>
</tr>
<tr>
<td>Peng et al. [42]</td>
<td>An overview of radio access networks in 5G.</td>
<td>✓</td>
</tr>
<tr>
<td>Niu et al. [43]</td>
<td>A survey of 5G millimeter wave communications.</td>
<td>✓</td>
</tr>
<tr>
<td>Wang et al. [2]</td>
<td>5G backhauling techniques and radio resource management.</td>
<td>✓</td>
</tr>
<tr>
<td>Foukas et al. [44]</td>
<td>A survey and insights on network slicing in 5G.</td>
<td>✓</td>
</tr>
<tr>
<td>Taleb et al. [45]</td>
<td>A survey on 5G edge architecture and orchestration.</td>
<td>✓</td>
</tr>
<tr>
<td>Mach and Becvar [46]</td>
<td>A survey on MEC.</td>
<td>✓</td>
</tr>
<tr>
<td>Mao et al. [47]</td>
<td>A survey on mobile edge computing.</td>
<td>✓</td>
</tr>
<tr>
<td>Wang et al. [48]</td>
<td>An architecture for personalized QoE management in 5G.</td>
<td>✓</td>
</tr>
<tr>
<td>Han et al. [49]</td>
<td>Insights to mobile cloud sensing, big data, and 5G.</td>
<td>✓</td>
</tr>
<tr>
<td>Singh et al. [50]</td>
<td>A survey on social networks over 5G.</td>
<td>✓</td>
</tr>
<tr>
<td>Chen et al. [51]</td>
<td>An introduction to 5G cognitive systems for healthcare.</td>
<td>✗</td>
</tr>
<tr>
<td>Chen et al. [52]</td>
<td>Machine learning for traffic offloading in cellular network</td>
<td>✓</td>
</tr>
<tr>
<td>Wu et al. [53]</td>
<td>Big data toward green cellular networks</td>
<td>✓</td>
</tr>
<tr>
<td>Buda et al. [54]</td>
<td>Machine learning aided use cases and scenarios in 5G.</td>
<td>✓</td>
</tr>
<tr>
<td>Imran et al. [55]</td>
<td>An introductions to big data analysis for self-organizing networks (SON) in 5G.</td>
<td>✓</td>
</tr>
<tr>
<td>Keshavarzian et al. [56]</td>
<td>Machine learning perspectives on SON in 5G.</td>
<td>✓</td>
</tr>
<tr>
<td>Klaine et al. [57]</td>
<td>A survey of machine learning applications in SON.</td>
<td>✗</td>
</tr>
<tr>
<td>Jiang et al. [7]</td>
<td>Machine learning paradigms for 5G.</td>
<td>✗</td>
</tr>
<tr>
<td>Li et al. [58]</td>
<td>Insights into intelligent 5G.</td>
<td>✗</td>
</tr>
<tr>
<td>Atal et al. [60]</td>
<td>A survey of big data application in cyber-physical systems.</td>
<td>✗</td>
</tr>
<tr>
<td>Cheng et al. [61]</td>
<td>A tutorial of mobile big data analysis</td>
<td>✓</td>
</tr>
<tr>
<td>Kasnises et al. [62]</td>
<td>Insights into employing deep learning for mobile data analysis.</td>
<td>✓</td>
</tr>
<tr>
<td>Alsheikh et al. [17]</td>
<td>Applying deep learning and Apache Spark for mobile data analytics.</td>
<td>✓</td>
</tr>
<tr>
<td>Cheng et al. [63]</td>
<td>Survey of mobile big data analysis and outlook.</td>
<td>✓</td>
</tr>
<tr>
<td>Kato et al. [65]</td>
<td>Proof-of-concept deep learning for network traffic control.</td>
<td>✓</td>
</tr>
<tr>
<td>Zorzi et al. [66]</td>
<td>An introduction to machine learning driven network optimization.</td>
<td>✓</td>
</tr>
<tr>
<td>Fadlullah et al. [67]</td>
<td>A comprehensive survey of deep learning for network traffic control.</td>
<td>✓</td>
</tr>
<tr>
<td>Mohammadi et al. [68]</td>
<td>A survey of deep learning in IoT data analytics.</td>
<td>✓</td>
</tr>
</tbody>
</table>
surveys, tutorials, and magazine papers (e.g. [4], [5], [38], [39], [47]). Andrews et al. highlight the differences between 5G and prior mobile network architectures, conduct a comprehensive review of 5G techniques, and discuss research challenges facing future developments [38]. Agiwal et al. review new architectures for 5G networks, survey emerging wireless technologies, and point out research problems that remain unsolved [4]. Gupta et al. also review existing work on 5G cellular network architectures, subsequently proposing a framework that incorporates networking ingredients such as Device-to-Device (D2D) communication, small cells, cloud computing, and the IoT [5].

Intelligent mobile networking is becoming a popular research area and related work has been reviewed in the literature (e.g. [7], [34], [37], [54], [56]–[59]). Jiang et al. discuss the potential of applying machine learning to 5G network applications including massive MIMO and smart grids [7]. This work further identifies several research gaps between ML and 5G that remain unexplored. Li et al. discuss opportunities and challenges of incorporating artificial intelligence (AI) into future network architectures and highlight the significance of AI in the 5G era [58]. Klaine et al. present several successful ML practices in Self-Organizing Networks (SONs), discuss the pros and cons of different algorithms, and identify future research directions in this area [57]. Potential exists to apply AI and exploit big data for energy efficiency purposes [53]. Chen et al. survey traffic offloading approaches in wireless networks, and propose a novel reinforcement learning based solution [52]. This opens a new research direction toward embedding machine learning towards greening cellular networks.

C. Deep Learning Driven Networking Applications

A growing number of papers survey recent works that bring deep learning into the computer networking domain. Alsheikh et al. identify benefits and challenges of using big data for mobile analytics and propose a Spark based deep learning framework for this purpose [17]. Wang and Jones discuss evaluation criteria, data streaming and deep learning practices for network intrusion detection, pointing out research challenges inherent to such applications [64]. Zheng et al. put forward a big data-driven mobile network optimization framework in 5G networks, to enhance QoE performance [6].

More recently, Fadlullah et al. deliver a survey on the progress of deep learning in a board range of areas, highlighting its potential application to network traffic control systems [67]. Their work also highlights several unsolved research issues worthy of future study.

Aihad et al. introduce techniques, applications, and guidelines on applying neural networks to wireless networking problems [69]. Despite several limitations of neural networks identified, this article focuses largely on old neural networks models, ignoring recent progress in deep learning and successful applications in current mobile networks. Lane et al. investigate the suitability and benefits of employing deep learning in mobile sensing, and emphasize on the potential for accurate inference on mobile devices [75]. Ota et al. report novel deep learning applications in mobile multimedia. Their survey covers state-of-the-art deep learning practices in mobile health and wellbeing, mobile security, mobile ambient intelligence, language translation, and speech recognition. Mohammadi et al. survey recent deep learning techniques for Internet of Things (IoT) data analytics [68]. They overview comprehensively existing efforts that incorporate deep learning into the IoT domain and shed light on current research challenges and future directions. Mao et al. focus on deep learning in wireless networking [70]. Their work surveys state-of-the-art deep learning applications in wireless networks, and discusses research challenges to be solved in the future.

D. Our Scope

The objective of this survey is to provide a comprehensive view on state-of-the-art deep learning practices in the mobile networking area. By this we aim to answer the following key questions:

1) Why is deep learning promising for solving mobile networking problems?
2) What are the cutting-edge deep learning models relevant to mobile and wireless networking?
3) What are the most recent successful deep learning applications in the mobile networking domain?
4) How can researchers tailor deep learning to specific mobile networking problems?
5) Which are the most important and promising directions worthy of further study?

<table>
<thead>
<tr>
<th>Publication</th>
<th>One-sentence summary</th>
<th>Deep learning</th>
<th>Other ML methods</th>
<th>Mobile big data</th>
<th>5G technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Luong et al. [71]</td>
<td>A survey of deep reinforcement learning for networking.</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gharabeh et al. [74]</td>
<td>A survey of smart cities.</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lane et al. [75]</td>
<td>An overview and introduction of deep learning-driven mobile sensing.</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ota et al. [76]</td>
<td>A survey of deep learning for mobile multimedia.</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mishra et al. [77]</td>
<td>A survey machine learning driven intrusion detection.</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Our work</strong></td>
<td>A comprehensive survey of deep learning for mobile and wireless network.</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The research papers and books we mentioned previously only partially answer these questions. This article goes beyond these previous works and specifically focuses on the crossovers between deep learning and mobile networking. We cover a range of neural network (NN) structures that are increasingly important and have not been explicitly discussed in earlier tutorials, e.g., [79]. This includes auto-encoders and Generative Adversarial Networks. Unlike such existing tutorials, we also review open-source libraries for deploying and training neural networks, a range of optimization algorithms, and the parallelization of neural networks models and training across large numbers of mobile devices. We also review applications not looked at in other related surveys, including traffic/user analytics, security and privacy, mobile health, etc.

While our main scope remains the mobile networking domain, for completeness we also discuss deep learning applications to wireless networks, and identify emerging application domains intimately connected to these areas. We differentiate between mobile networking, which refers to scenarios where devices are portable, battery powered, potentially wearable, and routinely connected to cellular infrastructure, and wireless networking, where devices are mostly fixed, and part of a distributed infrastructure (including WLANs and WSNs), and serve a single application. Overall, our paper distinguishes itself from earlier surveys from the following perspectives:

(i) We particularly focus on deep learning applications for mobile network analysis and management, instead of broadly discussing deep learning methods (as, e.g., in [20], [21]) or centering on a single application domain, e.g. mobile big data analysis with a specific platform [17].

(ii) We discuss cutting-edge deep learning techniques from the perspective of mobile networks (e.g., [80], [81]), focusing on their applicability to this area, whilst giving less attention to conventional deep learning models that may be out-of-date.

(iii) We analyze similarities between existing non-networking problems and those specific to mobile networks; based on this analysis we provide insights into both best deep learning architecture selection strategies and adaptation approaches, so as to exploit the characteristics of mobile networks for analysis and management tasks.

To the best of our knowledge, this is the first time that mobile network analysis and management are jointly reviewed from a deep learning angle. We also provide for the first time insights into how to tailor deep learning to mobile networking problems.

III. DEEP LEARNING 101

We begin with a brief introduction to deep learning, highlighting the basic principles behind computation techniques in this field, as well as key advantages that lead to their success. Deep learning is essentially a sub-branch of ML, which essentially enables an algorithm to make predictions, classifications, or decisions based on data, without being explicitly programmed. Classic examples include linear regression, the k-nearest neighbors classifier, and Q-learning. In contrast to traditional ML tools that rely heavily on features defined by domain experts, deep learning algorithms hierarchically extract knowledge from raw data through multiple layers of nonlinear processing units, in order to make predictions or take actions according to some target objective. The most well-known deep learning models are neural networks (NNs), but only NNs that have a sufficient number of hidden layers (usually more than one) can be regarded as ‘deep’ models. Besides deep NNs, other architectures have multiple layers, such as deep Gaussian processes [82], neural processes [83], and deep random forests [84], and can also be regarded as deep learning structures. The major benefit of deep learning over traditional ML is thus the automatic feature extraction, by which expensive hand-crafted feature engineering can be circumvented. We illustrate the relation between deep learning, machine learning, and artificial intelligence (AI) at a high level in Fig. 2.

In general, AI is a computation paradigm that endows machines with intelligence, aiming to teach them how to work, react, and learn like humans. Many techniques fall under this broad umbrella, including machine learning, expert systems, and evolutionary algorithms. Among these, machine learning enables the artificial processes to absorb knowledge from data and make decisions without being explicitly programmed. Machine learning algorithms are typically categorized into supervised, unsupervised, and reinforcement learning. Deep learning is a family of machine learning techniques that mimic biological nervous systems and perform representation learning through multi-layer transformations, extending across all three learning paradigms mentioned before. As deep learning has growing number of applications in mobile an wireless networking, the crossovers between these domains make the scope of this manuscript.

A. The Evolution of Deep Learning

The discipline traces its origins 75 years back, when threshold logic was employed to produce a computational model for neural networks [85]. However, it was only in the late 1980s that neural networks (NNs) gained interest, as Rumelhart et al. showed that multi-layer NNs could be trained...
effectively by back-propagating errors [86]. LeCun and Bengio subsequently proposed the now popular Convolutional Neural Network (CNN) architecture [87], but progress stalled due to computing power limitations of systems available at that time. Following the recent success of GPUs, CNNs have been employed to dramatically reduce the error rate in the Large Scale Visual Recognition Challenge (LSVRC) [88]. This has drawn unprecedented interest in deep learning and breakthroughs continue to appear in a wide range of computer science areas.

B. Fundamental Principles of Deep Learning

The key aim of deep neural networks is to approximate complex functions through a composition of simple and predefined operations of units (or neurons). Such an objective function can be almost of any type, such as a mapping between images and their class labels (classification), computing future stock prices based on historical values (regression), or even deciding the next optimal chess move given the current status on the board (control). The operations performed are usually defined by a weighted combination of a specific group of hidden units with a non-linear activation function, depending on the structure of the model. Such operations along with the output units are named “layers”. The neural network architecture resembles the perception process in a brain, where a specific set of units are activated given the current environment, influencing the output of the neural network model.

C. Forward and Backward Propagation

In mathematical terms, the architecture of deep neural networks is usually differentiable, therefore the weights (or parameters) of the model can be learned by minimizing a loss function using gradient descent methods through back-propagation, following the fundamental chain rule [86]. We illustrate the principles of the learning and inference processes of a deep neural network in Fig. 3, where we use a two-dimensional (2D) Convolutional Neural Network (CNN) as an example.

Forward propagation: The figure shows a CNN with 5 layers, i.e., an input layer (grey), 3 hidden layers (blue) and an output layer (orange). In forward propagation, A 2D input $x$ (e.g. images) is first processed by a convolutional layer, which perform the following convolutional operation:

$$h_1 = \sigma(w_1 \ast x).$$

Here $h_1$ is the output of the first hidden layer, $w_1$ is the convolutional filter and $\sigma(\cdot)$ is the activation function, aiming at improving the non-linearity and representability of the model. The output $h_1$ is subsequently provided as input to and processed by the following two convolutional layers, which eventually produces a final output $y$. This could be for instance vector of probabilities for different possible patterns (shapes) discovered in the (image) input. To train the CNN appropriately, one uses a loss function $L(w)$ to measure the distance between the output $y$ and the ground truth $y^*$. The purpose of training is to find the best weights $w$, so as to minimize the loss function $L(w)$. This can be achieved by the back propagation through gradient descent.

Backward propagation: During backward propagation, one computes the gradient of the loss function $L(w)$ over the weight of the last hidden layer, and updates the weight by computing:

$$w_4 = w_4 - \lambda \frac{dL(w)}{dw_4}.$$  

Here $\lambda$ denotes the learning rate, which controls the step size of moving in the direction indicated by the gradient. The same operation is performed for each weight, following the chain rule. The process is repeated and eventually the gradient descent will lead to a set $w$ that minimizes the $L(w)$.

For other NN structures, the training and inference processes are similar. To help less expert readers we detail the principles and computational details of various deep learning techniques in Sec.V.

TABLE IV: Summary of the benefits of applying deep learning to solve problems in mobile and wireless networks.

<table>
<thead>
<tr>
<th>Key aspect</th>
<th>Description</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature extraction</td>
<td>Deep neural networks can automatically extract high-level features through layers of different depths.</td>
<td>Reduce expensive hand-crafted feature engineering in processing heterogeneous and noisy mobile big data.</td>
</tr>
<tr>
<td>Big data exploitation</td>
<td>Unlike traditional ML tools, the performance of deep learning usually grow significantly with the size of training data.</td>
<td>Efficiently utilize huge amounts of mobile data generated at high rates.</td>
</tr>
<tr>
<td>Unsupervised learning</td>
<td>Deep learning is effective in processing un-/semi-labeled data, enabling unsupervised learning.</td>
<td>Handling large amounts of unlabeled data, which are common in mobile systems.</td>
</tr>
<tr>
<td>Multi-task learning</td>
<td>Features learned by neural networks through hidden layers can be applied to different tasks by transfer learning.</td>
<td>Reduce computational and memory requirements when performing multi-task learning in mobile systems.</td>
</tr>
<tr>
<td>Geometric mobile data learning</td>
<td>Dedicated deep learning architectures exist to model geometric mobile data.</td>
<td>Revolutionize geometric mobile data analysis.</td>
</tr>
</tbody>
</table>
D. Advantages of Deep Learning in Mobile and Wireless Networking

We recognize several benefits of employing deep learning to address network engineering problems, as summarized in Table IV. Specifically:

1) It is widely acknowledged that, while vital to the performance of traditional ML algorithms, feature engineering is costly [89]. A key advantage of deep learning is that it can automatically extract high-level features from data that has complex structure and inner correlations. The learning process does not need to be designed by a human, which tremendously simplifies prior feature handcrafting [20]. The importance of this is amplified in the context of mobile networks, as mobile data is usually generated by heterogeneous sources, is often noisy, and exhibits non-trivial spatial/temporal patterns [17], whose labeling would otherwise require outstanding human effort.

2) Secondly, deep learning is capable of handling large amounts of data. Mobile networks generate high volumes of different types of data at fast pace. Training traditional ML algorithms (e.g., Support Vector Machine (SVM) [90] and Gaussian Process (GP) [91]) sometimes requires to store all the data in memory, which is computationally infeasible under big data scenarios. Furthermore, the performance of ML does not grow significantly with large volumes of data and plateaus relatively fast [18]. In contrast, Stochastic Gradient Descent (SGD) employed to train NNs only requires sub-sets of data at each training step, which guarantees deep learning’s scalability with big data. Deep neural networks further benefit as training with big data prevents model over-fitting.

3) Traditional supervised learning is only effective when sufficient labeled data is available. However, most current mobile systems generate unlabeled or semi-labeled data [17]. Deep learning provides a variety of methods that allow exploiting unlabeled data to learn useful patterns in an unsupervised manner, e.g., Restricted Boltzmann Machine (RBM) [92], Generative Adversarial Network (GAN) [93]. Applications include clustering [94], data distributions approximation [93], un/semi-supervised learning [95], [96], and one/zero shot learning [97], [98], among others.

4) Compressive representations learned by deep neural networks can be shared across different tasks, while this is limited or difficult to achieve in other ML paradigms (e.g., linear regression, random forest, etc.). Therefore, a single model can be trained to fulfill multiple objectives, without requiring complete model retraining for different tasks. We argue that this is essential for mobile network engineering, as it reduces computational and memory requirements of mobile systems when performing multi-task learning applications [99].

5) Deep learning is effective in handing geometric mobile data [100], while this is a conundrum for other ML approaches. Geometric data refers to multivariate data represented by coordinates, topology, metrics and order [101]. Mobile data, such as mobile user location and network connectivity can be naturally represented by point clouds and graphs, which have important geometric properties. These data can be effectively modelled by dedicated deep learning architectures, such as PointNet++ [102] and Graph CNN [103]. Employing these architectures has great potential to revolutionize the geometric mobile data analysis [104].

E. Limitations of Deep Learning in Mobile and Wireless Networking

Although deep learning has unique advantages when addressing mobile network problems, it also has several shortcomings, which partially restricts its applicability in this domain. Specifically:

1) In general, deep learning (including deep reinforcement learning) is vulnerable to adversarial examples [105], [106]. These refer to artifact inputs that are intentionally designed by an attacker to fool machine learning models into making mistakes [105]. While it is difficult to distinguish such samples from genuine ones, they can trigger mis-adjustments of a model with high likelihood. We illustrate an example of such an adversarial attack in Fig. 4. Deep learning, especially CNNs are vulnerable to these types of attacks. This may also affect the applicability of deep learning in mobile systems. For instance, hackers may exploit this vulnerability and construct cyber attacks that subvert deep learning based detectors [107]. Constructing deep models that are robust to adversarial examples is imperative, but remains challenging.

![Fig. 4. An example of an adversarial attack on deep learning.](image)

2) Deep learning algorithms are largely black boxes and have low interpretability. Their major breakthroughs are in terms of accuracy, as they significantly improve performance of many tasks in different areas. However, although deep learning enables creating “machines” that have high accuracy in specific tasks, we still have limited knowledge as to why NNs make certain decisions. This limits the applicability of deep learning, e.g. in network economics. Therefore, businesses would rather continue to employ statistical methods that have high interpretability, whilst sacrificing on accuracy. Researchers have recognized this problem and investing continuous efforts to address this limitation of deep learning (e.g. [108]–[110]).

3) Deep learning is heavily reliant on data, which sometimes can be more important than the model itself. Deep models can further benefit from training data augmentation [111]. This is indeed an opportunity for mobile networking, as networks generates tremendous amounts of data. However, data collection may be costly, and face privacy
concern, therefore it may be difficult to obtain sufficient information for model training. In such scenarios, the benefits of employing deep learning may be outweighed by the costs.

4) Deep learning can be computationally demanding. Advanced parallel computing (e.g., GPUs, high-performance chips) fostered the development and popularity of deep learning, yet deep learning also heavily relies on these. Deep NNs usually require complex structures to obtain satisfactory accuracy performance. However, when deploying NNs on embedded and mobile devices, energy and capability constraints have to be considered. Very deep NNs may not be suitable for such scenario and this would inevitably compromise accuracy. Solutions are being developed to mitigate this problem and we will dive deeper into these in Sec. IV and VII.

5) Deep neural networks usually have many hyper-parameters and finding their optimal configuration can be difficult. For a single convolutional layer, we need to configure at least hyper-parameters for the number, shape, stride, and dilation of filters, as well as for the residual connections. The number of such hyper-parameters grows exponentially with the depth of the model and can highly influence its performance. Finding a good set of hyper-parameters can be similar to looking for a needle in a haystack. The AutoML platform\(^2\) provides a first solution to this problem, by employing progressive neural architecture search [112]. This task, however, remains costly.

To circumvent some of the aforementioned problems and allow for effective deployment in mobile networks, deep learning requires certain system and software support. We review and discuss such enablers in the next section.

IV. ENABLING DEEP LEARNING IN MOBILE NETWORKING

5G systems seek to provide high throughput and ultra-low latency communication services, to improve users’ QoE [4]. Implementing deep learning to build intelligence into 5G systems, so as to meet these objectives is expensive. This is because powerful hardware and software is required to support training and inference in complex settings. Fortunately, several tools are emerging, which make deep learning in mobile networks tangible; namely, (i) advanced parallel computing, (ii) distributed machine learning systems, (iii) dedicated deep learning libraries, (iv) fast optimization algorithms, and (v) fog computing. These tools can be seen as forming a hierarchical structure, as illustrated in Fig. 5: synergies between them exist that makes networking problem amenable to deep learning based solutions. By employing these tools, once the training is completed, inferences can be made within millisecond timescales, as already reported by a number of papers for a range of tasks (e.g., [113]–[115]). We summarize these advances in Table V and review them in what follows.

\(^2\)AutoML – training high-quality custom machine learning models with minimum effort and machine learning expertise. https://cloud.google.com/automl/

![Diagram showing enablers for deep learning in mobile networking](image-url)

Fig. 5: Hierarchical view of deep learning enablers. Parallel computing and hardware in fog computing lay foundations for deep learning. Distributed machine learning systems can build upon them, to support large-scale deployment of deep learning. Deep learning libraries run at the software level, to enable fast deep learning implementation. Higher-level optimizers are used to train the NN, to fulfill specific objectives.

A. Advanced Parallel Computing

Compared to traditional machine learning models, deep neural networks have significantly larger parameters spaces, intermediate outputs, and number of gradient values. Each of these need to be updated during every training step, requiring powerful computation resources. The training and inference processes involve huge amounts of matrix multiplications and other operations, though they could be massively parallelized. Traditional Central Processing Units (CPUs) have a limited number of cores, thus they only support restricted computing parallelism. Employing CPUs for deep learning implementations is highly inefficient and will not satisfy the low-latency requirements of mobile systems.

Engineers address these issues by exploiting the power of GPUs. GPUs were originally designed for high performance video games and graphical rendering, but new techniques such as Compute Unified Device Architecture (CUDA) [117] and the CUDA Deep Neural Network library (cuDNN) [118] developed by NVIDIA add flexibility to this type of hardware, allowing users to customize their usage for specific purposes. GPUs usually incorporate thousand of cores and perform exceptionally in fast matrix multiplications required for training neural networks. This provides higher memory bandwidth over CPUs and dramatically speeds up the learning process. Recent advanced Tensor Processing Units (TPUs) developed by Google even demonstrate 15-30x higher processing speeds and 30-80x higher performance-per-watt, as compared to CPUs and GPUs [116].

Diffractive neural networks (D²NNs) that completely rely on light communication were recently introduced in [133],
TABLE V: Summary of tools and techniques that enable deploying deep learning in mobile systems.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Examples</th>
<th>Scope</th>
<th>Functionality</th>
<th>Performance improvement</th>
<th>Energy consumption</th>
<th>Economic cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced parallel computing</td>
<td>GPU, TPU [116], CUDA [117], cuDNN [118]</td>
<td>Mobile servers, workstations</td>
<td>Enable fast, parallel training/inference of deep learning models in mobile applications</td>
<td>High</td>
<td>High</td>
<td>Medium (hardware)</td>
</tr>
<tr>
<td>Dedicated deep learning library</td>
<td>TensorFlow [119], Theano [120], Caffe [121], Torch [122]</td>
<td>Mobile servers and devices</td>
<td>High-level toolboxes that enable network engines to build purpose-specific deep learning architectures</td>
<td>Medium</td>
<td>Associated with hardware</td>
<td>Low (software)</td>
</tr>
<tr>
<td>Fog computing</td>
<td>nn-X [123], ncnn [124], Kirin 970 [125], Core ML [126]</td>
<td>Mobile devices</td>
<td>Support edge-based deep learning computing</td>
<td>Medium</td>
<td>Low</td>
<td>Medium (hardware)</td>
</tr>
<tr>
<td>Fast optimization algorithms</td>
<td>Nesterov [127], Adamgrad [128], RMSGD, Adam [129]</td>
<td>Training deep architectures</td>
<td>Accelerate and stabilize the model optimization process</td>
<td>Medium</td>
<td>Associated with hardware</td>
<td>Low (software)</td>
</tr>
<tr>
<td>Distributed machine learning systems</td>
<td>MLbase [130], Gaia [10], TUX2 [11], Adam [131], GeEPS [132]</td>
<td>Distributed data centers, cross-server</td>
<td>Support deep learning frameworks in mobile systems across data centers</td>
<td>High</td>
<td>High</td>
<td>High (hardware)</td>
</tr>
</tbody>
</table>

to enable zero-consumption and zero-delay deep learning. The D^2NN is composed of several transmissive layers, where points on these layers act as neurons in a NN. The structure is trained to optimize the transmission/reflection coefficients, which are equivalent to weights in a NN. Once trained, transmissive layers will be materialized via 3D printing and they can subsequently be used for inference.

There are also a number of toolboxes that can assist the computational optimization of deep learning on the server side. Spring and Shrivastava introduce a hashing based technique that substantially reduces computation requirements of deep network implementations [134]. Mirhoseini et al. employ a reinforcement learning scheme to enable machines to learn the optimal operation placement over mixture hardware for deep neural networks. Their solution achieves up to 20% faster computation speed than human experts’ designs of such placements [135].

Importantly, these systems are easy to deploy, therefore mobile network engineers do not need to rebuild mobile servers from scratch to support deep learning computing. This makes implementing deep learning in mobile systems feasible and accelerates the processing of mobile data streams.

B. Distributed Machine Learning Systems

Mobile data is collected from heterogeneous sources (e.g., mobile devices, network probes, etc.), and stored in multiple distributed data centers. With the increase of data volumes, it is impractical to move all mobile data to a central data center to run deep learning applications [10]. Running network-wide deep learning algorithms would therefore require distributed machine learning systems that support different interfaces (e.g., operating systems, programming language, libraries), so as to enable training and evaluation of deep models across geographically distributed servers simultaneously, with high efficiency and low overhead.

Deploying deep learning in a distributed fashion will inevitably introduce several system-level problems, which require satisfying the following properties:

- **Consistency** – Guaranteeing that model parameters and computational processes are consistent across all machines.
- **Fault tolerance** – Effectively dealing with equipment breakdowns in large-scale distributed machine learning systems.
- **Communication** – Optimizing communication between nodes in a cluster and to avoid congestion.
- **Storage** – Designing efficient storage mechanisms tailored to different environments (e.g., distributed clusters, single machines, GPUs), given I/O and data processing diversity.
- **Resource management** – Assigning workloads and ensuring that nodes work well-coordinated.
- **Programming model** – Designing programming interfaces to support multiple programming languages.

There exist several distributed machine learning systems that facilitate deep learning in mobile networking applications. Kraska et al. introduce a distributed system named MLbase, which enables to intelligently specify, select, optimize, and parallelize ML algorithms [130]. Their system helps non-experts deploy a wide range of ML methods, allowing optimization and running ML applications across different servers. Hsieh et al. develop a geography-distributed ML system called Gaia, which breaks the throughput bottleneck by employing an advanced communication mechanism over Wide Area Networks, while preserving the accuracy of ML algorithms [10]. Their proposal supports versatile ML interfaces (e.g. TensorFlow, Caffe), without requiring significant changes to the ML algorithm itself. This system enables deployments of complex deep learning applications over large-scale mobile networks.

Xing et al. develop a large-scale machine learning platform to support big data applications [136]. Their architecture achieves efficient model and data parallelization, enabling parameter state synchronization with low communication cost. Xiao et al. propose a distributed graph engine for ML named TUX2, to support data layout optimization across machines and reduce cross-machine communication [11]. They demonstrate remarkable performance in terms of runtime and convergence on a large dataset with up to 64 billion edges. Chiilimbi et al. build a distributed, efficient, and scalable
system named “Adam”\(^3\) tailored to the training of deep models [131]. Their architecture demonstrates impressive performance in terms of throughput, delay, and fault tolerance. Another dedicated distributed deep learning system called GeePS is developed by Cui et al. [132]. Their framework allows data parallelization on distributed GPUs, and demonstrates higher training throughput and faster convergence rate. More recently, Moritz et al. designed a dedicated distributed framework named Ray to underpin reinforcement learning applications [137]. Their framework is supported by an dynamic task execution engine, which incorporates the actor and task-parallel abstractions. They further introduce a bottom-up distributed scheduling strategy and a dedicated state storage scheme, to improve scalability and fault tolerance.

### C. Dedicated Deep Learning Libraries

Building a deep learning model from scratch can prove complicated to engineers, as this requires definitions of forwarding behaviors and gradient propagation operations at each layer, in addition to CUDA coding for GPU parallelization. With the growing popularity of deep learning, several dedicated libraries simplify this process. Most of these toolboxes work with multiple programming languages, and are built with GPU acceleration and automatic differentiation support. This eliminates the need of hand-crafted definition of gradient propagation. We summarize these libraries below, and give a comparison among them in Table VI.

**TensorFlow**\(^4\) is a machine learning library developed by Google [119]. It enables deploying computation graphs on CPUs, GPUs, and even mobile devices [138], allowing ML implementation on both single and distributed architectures. This permits fast implementation of deep NNs on both cloud and fog services. Although originally designed for ML and deep neural networks applications, TensorFlow is also suitable for other data-driven research purposes. It provides TensorBoard,\(^5\) a sophisticated visualization tool, to help users understand model structures and data flows, and perform debugging. Detailed documentation and tutorials for Python exist, while other programming languages such as C, Java, and Go are also supported. currently it is the most popular deep learning library. Building upon TensorFlow, several dedicated deep learning toolboxes were released to provide higher-level programming interfaces, including Keras\(^6\), Luminoth \(^7\) and TensorFlow [139].

**Theano** is a Python library that allows to efficiently define, optimize, and evaluate numerical computations involving multi-dimensional data [120]. It provides both GPU and CPU modes, which enables users to tailor their programs to individual machines. Learning Theano is however difficult and building a NNs with it involves substantial compiling time. Though Theano has a large user base and a support community, and at some stage was one of the most popular deep learning tools, its popularity is decreasing rapidly, as core ideas and attributes are absorbed by TensorFlow.

**Caffe(2)** is a dedicated deep learning framework developed

\(^3\)Note that this is distinct from the Adam optimizer discussed in Sec. IV-D

\(^4\)TensorFlow, https://www.tensorflow.org/


\(^6\)Keras deep learning library, https://github.com/fchollet/keras

\(^7\)Luminoth deep learning library for computer vision, https://github.com/tryolabs/luminoth
by Berkeley AI Research [121] and the latest version, Caffe2,\(^8\) was recently released by Facebook. Inheriting all the advantages of the old version, Caffe2 has become a very flexible framework that enables users to build their models efficiently. It also allows to train neural networks on multiple GPUs within distributed systems, and supports deep learning implementations on mobile operating systems, such as iOS and Android. Therefore, it has the potential to play an important role in the future mobile edge computing.

\textbf{(Py)Torch} is a scientific computing framework with wide support for machine learning models and algorithms [122]. It was originally developed in the Lua language, but developers later released an improved Python version [140]. In essence PyTorch is a lightweight toolbox that can run on embedded systems such as smart phones, but lacks comprehensive documentations. Since building NNs in PyTorch is straightforward, the popularity of this library is growing rapidly. It also offers rich pretrained models and modules that are easy to reuse and combine. PyTorch is now officially maintained by Facebook and mainly employed for research purposes.

\textbf{MXNET} is a flexible and scalable deep learning library that provides interfaces for multiple languages (e.g., C++, Python, Matlab, R, etc) [141]. It supports different levels of machine learning models, from logistic regression to GANs. MXNET provides fast numerical computation for both single machine and distributed ecosystems. It wraps workflows commonly used in deep learning into high-level functions, such that standard neural networks can be easily constructed without substantial coding effort. However, learning how to work with this toolbox in short time frame is difficult, hence the number of users who prefer this library is relatively small. MXNET is the official deep learning framework in Amazon.

Although less popular, there are other excellent deep learning libraries, such as CNTK,\(^9\) Deeplearning4j,\(^10\) Blocks,\(^11\) Gluon,\(^12\) and Lasagne,\(^13\) which can also be employed in mobile systems. Selecting among these varies according to specific applications. For AI beginners who intend to employ deep learning for the networking domain, PyTorch is a good candidate, as it is easy to build neural networks in this environment and the library is well optimized for GPUs. On the other hand, if for people who pursue advanced operations and large-scale implementation, Tensorflow might be a better choice, as it is well-established, under good maintenance and has starded the test of many Google industrial projects.

\textbf{D. Fast Optimization Algorithms}

The objective functions to be optimized in deep learning are usually complex, as they involve sums of extremely large numbers of data-wise likelihood functions. As the depth of the model increases, such functions usually exhibit high non-convexity with multiple local minima, critical points, and saddle points. In this case, conventional Stochastic Gradient Descent (SGD) algorithms [142] are slow in terms of convergence, which will restrict their applicability to latency constrained mobile systems. To overcome this problem and stabilize the optimization process, many algorithms evolve the traditional SGD, allowing NN models to be trained faster for mobile applications. We summarize the key principles behind these optimizers and make a comparison between them in Table VII. We delve into the details of their operation next.

\textbf{Fixed Learning Rate SGD Algorithms:} Suskever \textit{et al.} introduce a variant of the SGD optimizer with Nesterov's momentum, which evaluates gradients after the current velocity is applied [127]. Their method demonstrates faster convergence rate when optimizing convex functions. Another approach is Adagrad, which performs adaptive learning to model parameters according to their update frequency. This is suitable for handling sparse data and significantly outperforms SGD in terms of robustness [128]. Adadelta improves the traditional Adagrad algorithm, enabling it to converge faster, and does not rely on a global learning rate [143]. RMSprop is a popular SGD based method introduced by G. Hinton. RMSprop divides the learning rate by an exponential smoothing the average of gradients and does not require one to set the learning rate for each training step [142].

\textbf{Adaptive Learning Rate SGD Algorithms:} Kingma and Ba propose an adaptive learning rate optimizer named Adam, which incorporates momentum by the first-order moment of the gradient [129]. This algorithm is fast in terms of convergence, highly robust to model structures, and is considered as the first choice if one cannot decide what algorithm to use. By incorporating the momentum into Adam, Nadam applies stronger constraints to the gradients, which enables faster convergence [144].

\textbf{Other Optimizers:} Andrychowicz \textit{et al.} suggest that the optimization process can be even learned dynamically [145]. They pose the gradient descent as a trainable learning problem, which demonstrates good generalization ability in neural network training. Wen \textit{et al.} propose a training algorithm tailored to distributed systems [148]. They quantize float gradient values to \{-1, 0 and +1\} in the training processing, which theoretically require 20 times less gradient communications between nodes. The authors prove that such gradient approximation mechanism allows the objective function to converge to optima with probability 1, where in their experiments only a 2% accuracy loss is observed on average on GoogleLeNet [146] training. Zhou \textit{et al.} employ a differential private mechanism to compare training and validation gradients, to reuse samples and keep them fresh [147]. This can dramatically reduce overfitting during training.

\textbf{E. Fog Computing}

The fog computing paradigm presents a new opportunity to implement deep learning in mobile systems. Fog computing refers to a set of techniques that permit deploying applications
TABLE VII: Summary and comparison of different optimization algorithms.

<table>
<thead>
<tr>
<th>Optimization algorithm</th>
<th>Core idea</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD [142]</td>
<td>Computes the gradient of mini-batches iteratively and updates the parameters</td>
<td>• Easy to implement • Learning by hand can escape local minima</td>
<td>• Setting a global learning rate required • Algorithm may get stuck on saddle points or local minima • Slow in terms of convergence • Unstable</td>
</tr>
<tr>
<td>Nesterov’s momentum [127]</td>
<td>Introduces momentum to maintain the last gradient direction for the next update</td>
<td>• Stable • Faster learning • Can escape local minima</td>
<td>• Setting a learning rate needed</td>
</tr>
<tr>
<td>Adagrad [128]</td>
<td>Applies different learning rates to different parameters</td>
<td>• Learning rate tailored to each parameter • Handle sparse gradients well</td>
<td>• Still requires setting a global learning rate • Gradients sensitive to the regularizer • Learning rate becomes very slow in the late stages</td>
</tr>
<tr>
<td>Adadelta [143]</td>
<td>Improves Adagrad, by applying a self-adaptive learning rate</td>
<td>• Does not rely on a global learning rate • Faster speed of convergence • Fewer hyper-parameters to adjust</td>
<td>• May get stuck in a local minima at late training</td>
</tr>
<tr>
<td>RMSprop [142]</td>
<td>Employs root mean square as a constraint of the learning rate</td>
<td>• Learning rate tailored to each parameter • Learning rate do not decrease dramatically at late training • Works well in RNN training</td>
<td>• Still requires a global learning rate • Not good at handling sparse gradients</td>
</tr>
<tr>
<td>Adam [129]</td>
<td>Employs a momentum mechanism to store an exponentially decaying average of past gradients</td>
<td>• Learning rate tailored to each parameter • Good at handling sparse gradients and non-stationary problems • Memory-efficient • Fast convergence</td>
<td>• It may turn unstable during training</td>
</tr>
<tr>
<td>Nadam [144]</td>
<td>Incorporates Nesterov accelerated gradients into Adam</td>
<td>• Works well in RNN training</td>
<td>—</td>
</tr>
<tr>
<td>Learn to optimize [145]</td>
<td>Casts the optimization problem as a learning problem using a RNN</td>
<td>• Does not require to design the learning by hand</td>
<td>• Require an additional RNN for learning in the optimizer</td>
</tr>
<tr>
<td>Quantized training [146]</td>
<td>Quantizes the gradients into [-1, 0, 1] for training</td>
<td>• Good for distributed training • Memory-efficient</td>
<td>• Loses training accuracy</td>
</tr>
<tr>
<td>Stable gradient descent [147]</td>
<td>Employs a differential private mechanism to compare training and validation gradients, to reuse samples and keep them fresh</td>
<td>• More stable • Less overfitting • Converges faster than SGD</td>
<td>• Only validated on convex functions</td>
</tr>
</tbody>
</table>

or data storage at the edge of networks [149], e.g., on individual mobile devices. This reduces the communications overhead, offloads data traffic, reduces user-side latency, and lightens the sever-side computational burdens [150], [151]. A formal definition of fog computing is given in [152], where this is interpreted as ‘a huge number of heterogeneous (wireless and sometimes autonomous) ubiquitous and decentralized devices [that] communicate and potentially cooperate among them and with the network to perform storage and processing tasks without the intervention of third parties.’ To be more concrete, it can refer to smart phones, wearable devices and vehicles which store, analyze and exchange data, to offload the burden from cloud and perform more delay-sensitive tasks [153], [154]. Since fog computing involves deployment at the edge, participating devices usually have limited computing resource and battery power. Therefore, special hardware and software are required for deep learning implementation, as we explain next.

**Hardware:** There exist several efforts that attempt to shift deep learning computing from the cloud side to mobile devices [155]. For example, Gokhale et al. develop a mobile coprocessor named neural network nNeXt (nnX), which accelerates the deep neural networks execution in mobile devices, while retaining low energy consumption [123]. Bang et al. introduce a low-power and programmable deep learning processor to deploy mobile intelligence on edge devices [156]. Their hardware only consumes 288 µW but achieves 374 GOPS/W efficiency. A Neurosynaptic Chip called TrueNorth is proposed by IBM [157]. Their solution seeks to support computationally intensive applications on embedded battery-powered mobile devices. Qualcomm introduces a Snapdragon neural processing engine to enable deep learning computational optimization tailored to mobile devices.¹⁴ Their hardware allows developers to execute neural network models on Snapdragon 820 boards to serve a variety of applications.

TABLE VIII: Comparison of mobile deep learning platform.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Developer</th>
<th>Mobile hardware supported</th>
<th>Speed</th>
<th>Code size</th>
<th>Mobile compatibility</th>
<th>Open-sourced</th>
</tr>
</thead>
<tbody>
<tr>
<td>TensorFlow</td>
<td>Google</td>
<td>CPU</td>
<td>Slow</td>
<td>Medium</td>
<td>Medium</td>
<td>Yes</td>
</tr>
<tr>
<td>Caffe</td>
<td>Facebook</td>
<td>CPU</td>
<td>Slow</td>
<td>Large</td>
<td>Medium</td>
<td>Yes</td>
</tr>
<tr>
<td>ncm</td>
<td>Tencent</td>
<td>CPU</td>
<td>Medium</td>
<td>Small</td>
<td>Good</td>
<td>Yes</td>
</tr>
<tr>
<td>CoreML</td>
<td>Apple</td>
<td>CPU/GPU</td>
<td>Fast</td>
<td>Small</td>
<td>Only iOS 11+ supported</td>
<td>No</td>
</tr>
<tr>
<td>DeepSense</td>
<td>Yao et al.</td>
<td>CPU</td>
<td>Medium</td>
<td>Unknown</td>
<td>Medium</td>
<td>No</td>
</tr>
</tbody>
</table>

In close collaboration with Google, Movidius\textsuperscript{15} develops an embedded neural network computing framework that allows user-customized deep learning deployments at the edge of mobile networks. Their products can achieve satisfying runtime efficiency, while operating with ultra-low power requirements. It further supports difference frameworks, such as TensorFlow and Caffe, providing users with flexibility in choosing among toolkits. More recently, Huawei officially announced the Kirin 970 as a mobile AI computing system on chip.\textsuperscript{16} Their innovative framework incorporates dedicated Neural Processing Units (NPUs), which dramatically accelerates neural network computing, enabling classification of 2,000 images per second on mobile devices.

Software: Beyond these hardware advances, there are also software platforms that seek to optimize deep learning on mobile devices (e.g., \textsuperscript{158}). We compare and summarize all these platforms in Table VIII.\textsuperscript{17} In addition to the mobile version of TensorFlow and Caffe, Tencent released a lightweight, high-performance neural network inference framework tailored to mobile platforms, which relies on CPU computing.\textsuperscript{18} This toolbox performs better than all known CPU-based open source frameworks in terms of inference speed. Apple has developed “Core ML”, a private ML framework to facilitate mobile deep learning implementation on iOS 11.\textsuperscript{19} This lowers the entry barrier for developers wishing to deploy ML models on Apple equipment. Yao et al. develop a deep learning framework called DeepSense dedicated to mobile sensing related data processing, which provides a general machine learning toolbox that accommodates a wide range of edge applications. It has moderate energy consumption and low latency, thus being amenable to deployment on smartphones.

The techniques and toolboxes mentioned above make the deployment of deep learning practices in mobile network applications feasible. In what follows, we briefly introduce several representative deep learning architectures and discuss their applicability to mobile networking problems.

V. DEEP LEARNING: STATE-OF-THE-ART

Revisiting Fig. 2, machine learning methods can be naturally categorized into three classes, namely supervised learning, unsupervised learning, and reinforcement learning. Deep learning architectures have achieved remarkable performance in all these areas. In this section, we introduce the key principles underlying several deep learning models and discuss their largely unexplored potential to solve mobile networking problems. Technical details of classical models are provided to readers who seek to obtain a deeper understanding of neural networks. The more experienced can continue reading with Sec. VI. We illustrate and summarize the most salient architectures that we present in Fig. 6 and Table IX, respectively.

A. Multilayer Perceptron

The Multilayer Perceptrons (MLPs) is the initial Artificial Neural Network (ANN) design, which consists of at least three layers of operations \([175]\). Units in each layer are densely connected, hence require to configure a substantial number of weights. We show an MLP with two hidden layers in Fig. 6(a). Note that usually only MLPs containing more than one hidden layer are regarded as deep learning structures.

Given an input vector \(\mathbf{x}\), a standard MLP layer performs the following operation:

\[
y = \sigma(W \cdot \mathbf{x} + b).
\]

Here \(y\) denotes the output of the layer, \(W\) are the weights and \(b\) the biases. \(\sigma(\cdot)\) is an activation function, which aims at improving the non-linearity of the model. Commonly used activation function are the sigmoid,

\[
sigmoid(x) = \frac{1}{1 + e^{-x}},
\]

the Rectified Linear Unit (ReLU) \([176]\),

\[
ReLU(x) = \max(x, 0),
\]

tanh,

\[
tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}},
\]

and the Scaled Exponential Linear Units (SELU) \([177]\),

\[
SELU(x) = \lambda \begin{cases} x, & \text{if } x > 0; \\ \alpha e^x - \alpha, & \text{if } x \leq 0, \end{cases}
\]

where the parameters \(\lambda = 1.0507\) and \(\alpha = 1.6733\) are frequently used. In addition, the softmax function is typically employed in the last layer when performing classification:

\[
\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{k} e^{x_j}},
\]

where \(k\) is the number of labels involved in classification. Until recently, sigmoid and tanh have been the activation functions most widely used. However, they suffer from a known gradient vanishing problem, which hinders gradient learning.\textsuperscript{15}Movidius, an Intel company, provides cutting edge solutions for deploying deep learning and computer vision algorithms on ultra-low power devices. https://www.movidius.com/
\textsuperscript{16}Huawei announces the Kirin 970 – new flagship SoC with AI capabilities http://www.androidauthority.com/huawei-announces-kirin-970-797788/
\textsuperscript{17}Adapted from https://mp.weixin.qq.com/s/3gTpi1kq6Gwda50lnpOvKw
\textsuperscript{18}ncnn is a high-performance neural network inference framework optimized for the mobile platform, https://github.com/Tencent/ncnn
\textsuperscript{19}Core ML: Integrate machine learning models into your app, https://developer.apple.com/documentation/coreml
Fig. 6: Typical structure and operation principles of MLP, RBM, AE, CNN, RNN, LSTM, GAN, and DRL.
TABLE IX: Summary of different deep learning architectures. GAN and DRL are shaded, since they are built upon other models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Learning scenarios</th>
<th>Example architectures</th>
<th>Suitable problems</th>
<th>Pros</th>
<th>Cons</th>
<th>Potential applications in mobile networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>Supervised, unsupervised, reinforcement</td>
<td>ANN, AdaNet [159]</td>
<td>Modeling data with simple correlations</td>
<td>Naive structure and straightforward to build</td>
<td>High complexity, modest performance and slow convergence</td>
<td>Modeling multi-attribute mobile data; auxiliary or component of other deep architectures</td>
</tr>
<tr>
<td>RBM</td>
<td>Unsupervised</td>
<td>DBN [160], Convolutional DBN [161]</td>
<td>Extracting robust representations</td>
<td>Can generate virtual samples</td>
<td>Difficult to train well</td>
<td>Learning representations from unlabeled mobile data; model weight initialization; network flow prediction</td>
</tr>
<tr>
<td>AE</td>
<td>Unsupervised</td>
<td>DAE [162], VAE [163]</td>
<td>Learning sparse and compact representations</td>
<td>Powerful and effective unsupervised learning</td>
<td>Expensive to pretrain with big data</td>
<td>Model weight initialization; mobile data dimension reduction; mobile anomaly detection</td>
</tr>
<tr>
<td>CNN</td>
<td>Supervised, unsupervised, reinforcement</td>
<td>AlexNet [88], ResNet [164], 3D-ConvNet [165], GoogLeNet [146], DenseNet [166]</td>
<td>Spatial data modeling</td>
<td>Weight sharing; affine invariance</td>
<td>High computational cost; challenging to find optimal hyper-parameters; requires deep structures for complex tasks</td>
<td>Spatial mobile data analysis</td>
</tr>
<tr>
<td>RNN</td>
<td>Supervised, unsupervised, reinforcement</td>
<td>LSTM [167], Attention based RNN [168], ConvLSTM [169]</td>
<td>Sequential data modeling</td>
<td>Expertise in capturing temporal dependencies</td>
<td>High model complexity; gradient vanishing and exploding problems</td>
<td>Individual traffic flow analysis; network-wide (spatio-) temporal data modeling</td>
</tr>
<tr>
<td>GAN</td>
<td>Unsupervised</td>
<td>WGAN [81], LS-GAN [170], BigGAN [171]</td>
<td>Data generation</td>
<td>Can produce lifelike artifacts from a target distribution</td>
<td>Training process is unstable (convergence difficult)</td>
<td>Virtual mobile data generation; assisting supervised learning tasks in network data analysis</td>
</tr>
<tr>
<td>DRL</td>
<td>Reinforcement</td>
<td>DQN [19], Deep Policy Gradient [172], A3C [80], Rainbow [173], DPPO [174]</td>
<td>Control problems with high-dimensional inputs</td>
<td>Ideal for high-dimensional environment modeling</td>
<td>Slow in terms of convergence</td>
<td>Mobile network control and management.</td>
</tr>
</tbody>
</table>

propagation through layers. Therefore these functions are increasingly more often replaced by ReLU or SELU. SELU enables to normalize the output of each layer, which dramatically accelerates the training convergence, and can be viewed as a replacement of Batch Normalization [178].

The MLP can be employed for supervised, unsupervised, and even reinforcement learning purposes. Although this structure was the most popular neural network in the past, its popularity is decreasing because it entails high complexity (fully-connected structure), modest performance, and low convergence efficiency. MLPs are mostly used as a baseline or integrated into more complex architectures (e.g., the final layer in CNNs used for classification). Building an MLP is straightforward, and it can be employed, e.g., to assist with feature extraction in models built for specific objectives in mobile network applications. The advanced Adaptive learning of neural Network (AdaNet) enables MLPs to dynamically train their structures to adapt to the input [159]. This new architecture can be potentially explored for analyzing continuously changing mobile environments.

B. Boltzmann Machine

Restricted Boltzmann Machines (RBMs) [92] were originally designed for unsupervised learning purposes. They are essentially a type of energy-based undirected graphical models, and include a visible layer and a hidden layer, and where each unit can only assume binary values (i.e., 0 and 1). The probabilities of these values are given by:

\[
P(h_j = 1|v) = \frac{1}{1 + e^{-W \cdot v + b_j}}
\]

\[
P(v_j = 1|h) = \frac{1}{1 + e^{-W^T \cdot h + a_j}},
\]

where \(h, v\) are the hidden and visible units respectively, and \(W\) are weights and \(a, b\) are biases. The visible units are conditional independent to the hidden units, and \textit{vice versa}. A typical structure of an RBM is shown in Fig. 6(b). In general, input data are assigned to visible units \(v\). Hidden units \(h\) are invisible and they fully connect to all \(v\) through weights \(W\), which is similar to a standard feed forward neural network. However, unlike in MLPs where only the input vector can affect the hidden units, with RBMs the state of \(v\) can affect the state of \(h\), and \textit{vice versa}. 


RBM}s can be effectively trained using the contrastive divergence algorithm [179] through multiple steps of Gibbs sampling [180]. We illustrate the structure and the training process of an RBM in Fig. 6(b). RBM-based models are usually employed to initialize the weights of a neural network in more recent applications. The pre-trained model can be subsequently fine-tuned for supervised learning purposes using a standard back-propagation algorithm. A stack of RBMs is called a Deep Belief Network (DBN) [160], which performs layer-wise training and achieves superior performance as compared to MLPs in many applications, including time series forecasting [181], ratio matching [182], and speech recognition [183]. Such structures can be even extended to a convolutional architecture, to learn hierarchical spatial representations [161].

C. Auto-Encoders

Auto-Encoders (AEs) are also designed for unsupervised learning and attempt to copy inputs to outputs. The underlying principle of an AE is shown in Fig. 6(c). AEs are frequently used to learn compact representation of data for dimension reduction [184]. Extended versions can be further employed to initialize the weights of a deep architecture, e.g., the Denoising Auto-Encoder (DAE) [162]), and generate virtual examples by 39.7% and revolutionizes the imaging classification field. Other CNN based models, while requiring fewer layers. CNNs thereby achieving significant accuracy improvements over popular event).

A VAE typically comprises two neural networks – an encoder and a decoder. The input of the encoder is a data point \( x \) (e.g., images) and its functionality is to encode this input into a latent representation space \( z \). Let \( f_\theta(z|x) \) be an encoder parameterized by \( \Theta \) and \( z \) is sampled from a Gaussian distribution, the objective of the encoder is to output the mean and variance of the Gaussian distribution. Similarly, denoting \( g_\Omega(x|z) \) the decoder parameterized by \( \Omega \), this accepts the latent representation \( z \) as input, and outputs the parameter of the distribution of \( x \). The objective of the VAE is to minimize the reconstruction error of the data and the Kullback-Leibler (KL) divergence between \( p(z) \) and \( f_\theta(z|x) \). Once trained, the VAE can generate new data point samples by (i) drawing latent variables \( z_i \sim p(z) \) and (ii) drawing a new data point \( x_i \sim p(x|z) \).

AEs can be employed to address network security problems, as several research papers confirm their effectiveness in detecting anomalies under different circumstances [185–187], which we will further discuss in subsection VI-H. The structures of RBMs and AEs are based upon MLPs, CNNs or RNNS. Their goals are similar, while their learning processes are different. Both can be exploited to extract patterns from unlabeled mobile data, which may be subsequently employed for various supervised learning tasks, e.g., routing [188], mobile activity recognition [189], [190], periorocular verification [191] and base station user number prediction [192].

D. Convolutional Neural Network

Instead of employing full connections between layers, Convolutional Neural Networks (CNNs or ConvNets) employ a set of locally connected kernels (filters) to capture correlations between different data regions. Mathematically, for each location \( p_x \) of the output \( y \), the standard convolution performs the following operation:

\[
y(p_y) = \sum_{p_G \in \mathbb{G}} w(p_G) \cdot x(p_y + p_G).
\] (4)

where \( p_G \) denotes all positions in the receptive field \( \mathbb{G} \) of the convolutional filter \( W \), effectively representing the receptive range of each neuron to inputs in a convolutional layer. Here the weights \( W \) are shared across different locations of the input map. We illustrate the operation of one 2D convolutional layer in Fig. 6(d). Specifically, the inputs of a 2D CNN layer are multiple 2D matrices with different channels (e.g. the RGB representation of images). A convolutional layer employs multiple filters shared across different locations, to “scan” the inputs and produce output maps. In general, if the inputs and outputs have \( M \) and \( N \) filters respectively, the convolutional layer will require \( M \times N \) filters to perform the convolution operation.

CNNs improve traditional MLPs by leveraging three important ideas, namely, (i) sparse interactions, (ii) parameter sharing, and (iii) equivariant representations [18]. This reduces the number of model parameters significantly and maintains the affine invariance (i.e., recognition results are robust to the affine transformation of objects). Specifically, The sparse interactions imply that the weight kernel has smaller size than the input. It performs moving filtering to produce outputs (with roughly the same size as the inputs) for the current layer. Parameter sharing refers to employing the same kernel to scan the whole input map. This significantly reduces the number of parameters needed, which mitigates the risk of over-fitting. Equivariant representations indicate that convolution operations are invariant in terms of translation, scale, and shape. This is particularly useful for image processing, since essential features may show up at different locations in the image, with various affine patterns.

Owing to the properties mentioned above, CNNs achieve remarkable performance in imaging applications. Krizhevsky et al. [88] exploit a CNN to classify images on the ImageNet dataset [193]. Their method reduces the top-5 error by 39.7% and revolutionizes the imaging classification field. GoogLeNet [146] and ResNet [164] significantly increase the depth of CNN structures, and propose inception and residual learning techniques to address problems such as over-fitting and gradient vanishing introduced by “depth”. Their structure is further improved by the Dense Convolutional Network (DenseNet) [166], which reuses feature maps from each layer, thereby achieving significant accuracy improvements over other CNN based models, while requiring fewer layers. CNNs have also been extended to video applications. Ji et al. propose 3D convolutional neural networks for video activity recognition [165], demonstrating superior accuracy as compared to 2D CNN. More recent research focuses on learning the shape of convolutional kernels [194–196]. These dynamic architectures allow to automatically focus on important regions in input maps. Such properties are particularly important in analyzing large-scale mobile environments exhibiting clustering behaviors (e.g., surge of mobile traffic associated with a popular event).
Given the high similarity between image and spatial mobile data (e.g., mobile traffic snapshots, users’ mobility, etc.), CNN-based models have huge potential for network-wide mobile data analysis. This is a promising future direction that we further discuss in Sec. VIII.

E. Recurrent Neural Network

Recurrent Neural Networks (RNNs) are designed for modeling sequential data, where sequential correlations exist between samples. At each time step, they produce output via recurrent connections between hidden units [18], as shown in Fig. 6(e). Given a sequence of inputs \( \mathbf{x} = \{x_1, x_2, \ldots, x_T\} \), a standard RNN performs the following operations:

\[
\begin{align*}
    s_t &= \sigma_h(W_h s_{t-1} + b_h), \\
    h_t &= \sigma_t(W_x x_t + W_h s_{t-1} + b_x)
\end{align*}
\]

where \( s_t \) represents the state of the network at time \( t \) and it constructs a memory unit for the network. Its values are computed by a function of the input \( x_t \) and previous state \( s_{t-1} \). \( h_t \) is the output of the network at time \( t \). In natural language processing applications, this usually represents a language vector and becomes the input at \( t+1 \) after being processed by an embedding layer. The weights \( W_x, W_h \) and biases \( b_x, b_h \) are shared across different temporal locations. This reduces the model complexity and the degree of overfitting.

The RNN is trained via a Backpropagation Through Time (BPTT) algorithm. However, gradient vanishing and exploding problems are frequently reported in traditional RNNs, which make them particularly hard to train [197]. The Long Short-Term Memory (LSTM) mitigates these issues by introducing a set of “gates” [167], which has been proven successful in many applications (e.g., speech recognition [198], text categorization [199], and wearable activity recognition [114]). A standard LSTM performs the following operations:

\[
\begin{align*}
    i_t &= \sigma(W_{xi} X_t + W_{hi} H_{t-1} + b_i), \\
    f_t &= \sigma(W_{xf} X_t + W_{hf} H_{t-1} + b_f), \\
    C_t &= f_t \odot C_{t-1} + i_t \odot \tanh(W_{xc} X_t + W_{hc} H_{t-1} + b_c), \\
    o_t &= \sigma(W_{xo} X_t + W_{ho} H_{t-1} + b_o), \\
    h_t &= o_t \odot \tanh(C_t).
\end{align*}
\]

Here, ‘\( \odot \)’ denotes the Hadamard product, \( C_t \) denotes the cell outputs, \( h_t \) are the hidden states, \( i_t, f_t, \) and \( o_t \) are input gates, forget gates, and output gates, respectively. These gates mitigate the gradient issues and significantly improve the RNN. We illustrate the structure of an LSTM in Fig. 6(f).

Sutskever et al. introduce attention mechanisms to RNNs, which achieves outstanding accuracy in tokenized predictions [168]. Shi et al. substitute the dense matrix multiplication in LSTMs with convolution operations, designing a Convolutional Long Short-Term Memory (ConvLSTM) [169]. Their proposal reduces the complexity of traditional LSTM and demonstrates significantly lower prediction errors in precipitation nowcasting (i.e., forecasting the volume of precipitation).

Mobile networks produce massive sequential data from various sources, such as data traffic flows, and the evolution of mobile network subscribers’ trajectories and application latencies. Exploring the RNN family is promising to enhance the analysis of time series data in mobile networks.

F. Generative Adversarial Network

The Generative Adversarial Network (GAN) is a framework that trains generative models using the following adversarial process. It simultaneously trains two models: a generative one \( G \) that seeks to approximate the target data distribution from training data, and a discriminative model \( D \) that estimates the probability that a sample comes from the real training data rather than the output of \( G \) [93]. Both of \( G \) and \( D \) are normally neural networks. The training procedure for \( G \) aims to maximize the probability of \( D \) making a mistake. The overall objective is solving the following minimax problem [93]:

\[
\min_G \max_D \mathbb{E}_{x \sim P_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim P_z}(\log(1 - D(G(z)))).
\]

Algorithm 1 shows the typical routine used to train a simple GAN. Both the generators and the discriminator are trained iteratively while fixing the other one. Finally \( G \) can produce data close to a target distribution (the same with training examples), if the model converges. We show the overall structure of a GAN in Fig. 6(g). In practice, the generator \( G \) takes a noise vector \( z \) as input, and generates an output \( G(z) \) that follows the target distribution. \( D \) will try to discriminate whether \( G(z) \) is a real sample or an artifact [200]. This effectively constructs a dynamic game, for which a Nash Equilibrium is reached if both \( G \) and \( D \) become optimal, and \( G \) can produce lifelike data that \( D \) can no longer discriminate, i.e. \( D(G(z)) = 0.5, \forall z \).

The training process of traditional GANs is highly sensitive to model structures, learning rates, and other hyper-parameters.
Researchers are usually required to employ numerous ad hoc ‘tricks’ to achieve convergence and improve the fidelity of data generated. There exist several solutions for mitigating this problem, e.g., Wasserstein Generative Adversarial Network (WGAN) [81], Loss-Sensitive Generative Adversarial Network (LS-GAN) [170] and BigGAN [171], but research on the theory of GANs remains shallow. Recent work confirms that GANs can promote the performance of some supervised tasks (e.g., super-resolution [201], object detection [202], and face completion [203]) by minimizing the divergence between inferred and real data distributions. Exploiting the unsupervised learning abilities of GANs is promising in terms of generating synthetic mobile data for simulations, or assisting specific supervised tasks in mobile network applications. This becomes more important in tasks where appropriate datasets are lacking, given that operators are generally reluctant to share their network data.

G. Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) refers to a set of methods that approximate value functions (deep Q learning) or policy functions (policy gradient method) through deep neural networks. An agent (neural network) continuously interacts with an environment and receives reward signals as feedback. The agent selects an action at each step, which will change the state of the environment. The training goal of the neural network is to optimize its parameters, such that it can select actions that potentially lead to the best future return. We illustrate this principle in Fig. 6(h). DRL is well-suited to problems that have a huge number of possible states (i.e., environments are high-dimensional). Representative DRL methods include Deep Q-Networks (DQNs) [19], deep policy gradient methods [172], Asynchronous Advantage Actor-Critic [80], Rainbow [173] and Distributed Proximal Policy Optimization (DPPO) [174]. These perform remarkably in AI gaming (e.g., Gym\(^{20}\)), robotics, and autonomous driving [205]–[208], and have made inspiring deep learning breakthroughs recently.

In particular, the DQN [19] is first proposed by DeepMind to play Atari video games. However, traditional DQN requires several important adjustments to work well. The A3C [80] employs an actor-critic mechanism, where the actor selects the action given the state of the environment, and the critic estimates the value given the state and the action, then delivers feedback to the actor. The A3C deploys different actors and critics on different threads of a CPU to break the dependency of data. This significantly improves training convergence, enabling fast training of DRL agents on CPUs. Rainbow [173] combines different variants of DQNs, and discovers that these are complementary to some extent. This insight improved performance in many Atari games. To solve the step size problem in policy gradients methods, Schulman et al. propose a Distributed Proximal Policy Optimization (DPPO) method to constrain the update step of new policies, and implement this on multi-threaded CPUs in a distributed manner [174]. Based on this method, an agent developed by OpenAI defeated a human expert in Dota2 team in a 5v5 match.\(^{21}\) Recent DRL method also conquers a more complex real-time multi-agent game StarCraft II.\(^{22}\) In [209], DeepMind develops an agent based on supervised learning and DRL named AlphaStar, beating one of the world’s strongest professional StarCraft players by 5-0.

Many mobile networking problems can be formulated as Markov Decision Processes (MDPs), where reinforcement learning can play an important role (e.g., base station on-off switching strategies [210], routing [211], and adaptive tracking control [212]). Some of these problems nevertheless involve high-dimensional inputs, which limits the applicability of traditional reinforcement learning algorithms. DRL techniques broaden the ability of traditional reinforcement learning algorithms to handle high dimensionality, in scenarios previously considered intractable. Employing DRL is thus promising to address network management and control problems under complex, changeable, and heterogeneous mobile environments. We further discuss this potential in Sec. VIII.

VI. Deep Learning Driven Mobile and Wireless Networks

Deep learning has a wide range of applications in mobile and wireless networks. In what follows, we present the most important research contributions across different mobile networking areas and compare their design and principles. In particular, we first discuss a key prerequisite, that of mobile big data, then organize the review of relevant works into nine subsections, focusing on specific domains where deep learning has made advances. Specifically,

1) Deep Learning Driven Network-Level Mobile Data Analysis focuses on deep learning applications built on mobile big data collected within the network, including network prediction, traffic classification, and Call Detail Record (CDR) mining.

2) Deep Learning Driven App-Level Mobile Data Analysis shifts the attention towards mobile data analytics on edge devices.

3) Deep Learning Driven User Mobility Analysis sheds light on the benefits of employing deep neural networks to understand the movement patterns of mobile users, either at group or individual levels.

4) Deep Learning Driven User Localization reviews literature that employ deep neural networks to localize users in indoor or outdoor environments, based on different signals received from mobile devices or wireless channels.

5) Deep Learning Driven Wireless Sensor Networks discusses important work on deep learning applications in WSNs from four different perspectives, namely centralized vs. decentralized sensing, WSN data analysis, WSN localization and other applications.

6) Deep Learning Driven Network Control investigate the usage of deep reinforcement learning and deep imitation

\(^{20}\)Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Pinball. In combination with the NS3 simulator Gym becomes applicable to networking research. [204] https://gym.openai.com/

\(^{21}\)Dota2 is a popular multiplayer online battle arena video game.

\(^{22}\)StarCraft II is a popular multi-agent real-time strategy game.
learning on network optimization, routing, scheduling, resource allocation, and radio control.

7) **Deep Learning Driven Network Security** presents work that leverages deep learning to improve network security, which we cluster by focus as infrastructure, software, and privacy related.

8) **Deep Learning Driven Signal Processing** scrutinizes physical layer aspects that benefit from deep learning and reviews relevant work on signal processing.

9) **Emerging Deep Learning Driven Mobile Network Application** warps up this section, presenting other interesting deep learning applications in mobile networking.

For each domain, we summarize work broadly in tabular form, providing readers with a general picture of individual topics. Most important works in each domain are discussed in more details in text. Lessons learned are also discussed at the end of each subsection. We give a diagramatic view of the topics dealt with by the literature reviewed in this section in Fig. 7.

### A. Mobile Big Data as a Prerequisite

The development of mobile technology (e.g. smartphones, augmented reality, etc.) are forcing mobile operators to evolve mobile network infrastructures. As a consequence, both the cloud and edge side of mobile networks are becoming increasingly sophisticated to cater for users who produce and consume huge amounts of mobile data daily. These data can be either generated by the sensors of mobile devices that record individual user behaviors, or from the mobile network infrastructure, which reflects dynamics in urban environments.

Appropriately mining these data can benefit multidisciplinary research fields and the industry in areas such mobile network management, social analysis, public transportation, personal services provision, and so on [36]. Network operators, however, could become overwhelmed when managing and analyzing massive amounts of heterogeneous mobile data [467]. Deep learning is probably the most powerful methodology that can overcome this burden. We begin therefore by introducing characteristics of mobile big data, then present a holistic review of deep learning driven mobile data analysis research.

Yazti and Krishnaswamy propose to categorize mobile data into two groups, namely network-level data and app-level data [468]. The key difference between them is that in the former data is usually collected by the edge mobile devices, while in the latter obtained throughout network infrastructure. We summarize these two types of data and their information comprised in Table X. Before delving into mobile data analytics, we illustrate the typical data collection process in Figure 9.

![Fig. 7: Classification of the literature reviewed in Sec. VI.](image)

#### TABLE X: The taxonomy of mobile big data.

<table>
<thead>
<tr>
<th>Mobile data</th>
<th>Source</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network-level data</td>
<td>Infrastructure</td>
<td>Infrastructure locations, capability, equipment holders, etc.</td>
</tr>
<tr>
<td></td>
<td>Performance indicators</td>
<td>Data traffic, end-to-end delay, QoE, jitter, etc.</td>
</tr>
<tr>
<td></td>
<td>Call detail records (CDR)</td>
<td>Session start and end times, type, sender and receiver, etc.</td>
</tr>
<tr>
<td></td>
<td>Radio information</td>
<td>Signal power, frequency, spectrum, modulation etc.</td>
</tr>
<tr>
<td>App-level data</td>
<td>Device</td>
<td>Device type, usage, Media Access Control (MAC) address, etc.</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>User settings, personal information, etc.</td>
</tr>
<tr>
<td></td>
<td>Sensors</td>
<td>Mobility, temperature, magnetic field, movement, etc.</td>
</tr>
<tr>
<td></td>
<td>Application</td>
<td>Picture, video, voice, health condition, preference, etc.</td>
</tr>
<tr>
<td></td>
<td>System log</td>
<td>Software and hardware failure logs, etc.</td>
</tr>
</tbody>
</table>

**Network-level** mobile data generated by the networking infrastructure not only deliver a global view of mobile network performance (e.g. throughput, end-to-end delay, jitter, etc.), but also log individual session times, communication types, sender and receiver information, through Call Detail Records (CDRs). Network-level data usually exhibit significant spatio-temporal variations resulting from users’ behaviors [469], which can be utilized for network diagnosis and management, user mobility analysis and public transportation planning [219]. Some network-level data (e.g. mobile traffic snapshots) can be viewed as pictures taken by ‘panoramic cameras’, which provide a city-scale sensing system for urban sensing.

On the other hand, **App-level** data is directly recorded by sensors or mobile applications installed in various mobile devices. These data are frequently collected through crowd-sourcing schemes from heterogeneous sources, such as Global Positioning Systems (GPS), mobile cameras and video recorders, and portable medical monitors. Mobile devices act...
as sensor hubs, which are responsible for data gathering and preprocessing, and subsequently distributing such data to specific locations, as required [36]. We show a typical app-level data processing system in Fig. 8. App-level mobile data is generated and collected by a Software Development Kit (SDK) installed on mobile devices. After pre-processing and load-balancing (e.g., Nginx [23]), such data is subsequently processed by real-time collection and computing services (e.g., Storm, Kafka [24], HBase, Redis [26], etc.) as required. Further offline storage and computing with mobile data can be performed with various tools, such as Hadoop Distributed File System (HDFS) [28], Python, Mahout, Pig [30], or Oozie [31]. The raw data and analysis results will be further transferred to databases (e.g., MySQL [32]) Business Intelligence – BI (e.g., Online Analytical Processing – OLAP [33]), and data warehousing (e.g., Hive [34]). Among these, the algorithms container is the core of the entire system as it connects to front-end access and fog computing, real-time collection and computing, and offline computing and analysis modules, while it links directly to mobile applications, such as mobile healthcare, pattern recognition, and advertising platforms. Deep learning logic can be placed within the algorithms container.

App-level data may directly or indirectly reflect users’

---

23 Nginx is an HTTP and reverse proxy server, a mail proxy server, and a generic TCP/UDP proxy server, https://nginx.org/en/.
24 Storm is a free and open-source distributed real-time computation system, http://storm.apache.org/
26 Apache HBase [TM] is the Hadoop database, a distributed, scalable, big data store, https://hbase.apache.org/
27 Redis is an open source, in-memory data structure store, used as a database, cache and message broker, https://redis.io/
28 The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware, https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
29 Apache Mahout [TM] is a distributed linear algebra framework, https://mahout.apache.org/
30 Apache Pig is a high-level platform for creating programs that run on Apache Hadoop, https://pig.apache.org/
31 Oozie is a workflow scheduler system to manage Apache Hadoop jobs, http://oozie.apache.org/
33 OLAP is an approach to answer multi-dimensional analytical queries swiftly in computing, and is part of the broader category of business intelligence.
34 The Apache Hive [TM] is a data warehouse software, https://hive.apache.org/
behaviors, such as mobility, preferences, and social links [63]. Analyzing app-level data from individuals can help reconstructing one’s personality and preferences, which can be used in recommender systems and users targeted advertising. Some of these data comprise explicit information about individuals’ identities. Inappropriate sharing and use can raise significant privacy issues. Therefore, extracting useful patterns from multi-modal sensing devices without compromising user’s privacy remains a challenging endeavor.

Compared to traditional data analysis techniques, deep learning embraces several unique features to address the aforementioned challenges [17]. Namely:

1) Deep learning achieves remarkable performance in various data analysis tasks, on both structured and unstructured data. Some types of mobile data can be represented as image-like (e.g. [219]) or sequential data [227].

2) Deep learning performs remarkably well in feature extraction from raw data. This saves tremendous effort of hand-crafted feature engineering, which allows spending more time on model design and less on sorting through the data itself.

3) Deep learning offers excellent tools (e.g. RBM, AE, GAN) for handling unlabeled data, which is common in mobile network logs.

4) Multi-modal deep learning allows to learn features over multiple modalities [470], which makes it powerful in modeling with data collected from heterogeneous sensors and data sources.

These advantages make deep learning as a powerful tool for mobile data analysis.

B. Deep Learning Driven Network-level Mobile Data Analysis

Network-level mobile data refers broadly to logs recorded by Internet service providers, including infrastructure metadata, network performance indicators and call detail records (CDRs) (see Table XI). The recent remarkable success of deep learning ignites global interests in exploiting this methodology for mobile network-level data analysis, so as to optimize mobile networks configurations, thereby improving end-uses’ QoE. These work can be categorized into four types: network state prediction, network traffic classification, CDR mining and radio analysis. In what follows, we review work in these directions, which we first summarize and compare in Table XI.

Network State Prediction refers to inferring mobile network traffic or performance indicators, given historical measurements or related data. Pierucci and Micheli investigate the relationship between key objective metrics and QoE [213]. They employ MLPs to predict users’ QoE in mobile communications, based on average user throughput, number of active users in a cells, average data volume per user, and channel quality indicators, demonstrating high prediction accuracy.

Network traffic forecasting is another field where deep learning is gaining importance. By leveraging sparse coding and max-pooling, Gwon and Kung develop a semi-supervised deep learning model to classify received frame/packet patterns and infer the original properties of flows in a WiFi network [214]. Their proposal demonstrates superior performance over traditional ML techniques. Nie et al. investigate the traffic demand patterns in wireless mesh network [215]. They design a DBN along with Gaussian models to precisely estimate traffic distributions.

In addition to the above, several researchers employ deep learning to forecast mobile traffic at city scale, by considering spatio-temporal correlations of geographic mobile traffic measurements. We illustrate the underlying principle in Fig. 10. In [217], Wang et al. propose to use an AE-based architecture and LSTMs to model spatial and temporal correlations of mobile traffic distribution, respectively. In particular, the authors use a global and multiple local stacked AEs for spatial feature extraction, dimension reduction and training parallelism. Compressed representations extracted are subsequently processed by LSTMs, to perform final forecasting. Experiments with a real-world dataset demonstrate superior performance over SVM and the Autoregressive Integrated
<table>
<thead>
<tr>
<th>Domain</th>
<th>Reference</th>
<th>Applications</th>
<th>Model</th>
<th>Optimizer</th>
<th>Key contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network prediction</td>
<td>Perucci and Micheli [213]</td>
<td>QoE prediction</td>
<td>MLP</td>
<td>Unknown</td>
<td>Uses NNs to correlate Quality of Service parameters and QoE estimations.</td>
</tr>
<tr>
<td></td>
<td>Nie et al. [215]</td>
<td>Wireless mesh network traffic prediction</td>
<td>DBN + Gaussian models</td>
<td>SGD</td>
<td>Considers both long-term dependency and short-term fluctuations.</td>
</tr>
<tr>
<td></td>
<td>Moyo and Sibanda [216]</td>
<td>TCP/IP traffic prediction</td>
<td>MLP</td>
<td>Unknown</td>
<td>Investigates the impact of learning in traffic forecasting.</td>
</tr>
<tr>
<td></td>
<td>Wang et al. [217]</td>
<td>Mobile traffic forecasting</td>
<td>AE + LSTM</td>
<td>SGD</td>
<td>Uses an AE to model spatial correlations and an LSTM to model temporal correlation</td>
</tr>
<tr>
<td></td>
<td>Zhang et al. [219]</td>
<td>Mobile traffic super-resolution</td>
<td>CNN + GAN</td>
<td>Adam</td>
<td>Introduces the MTSR concept and applies image processing techniques for mobile traffic analysis</td>
</tr>
<tr>
<td></td>
<td>Huang et al. [220]</td>
<td>Mobile traffic forecasting</td>
<td>LSTM + 3D-CNN</td>
<td>Unknown</td>
<td>Combines CNNs and RNNs to extract geographical and temporal features from mobile traffic.</td>
</tr>
<tr>
<td></td>
<td>Zhang et al. [221]</td>
<td>Cellular traffic prediction</td>
<td>Densely connected CNN</td>
<td>Adam</td>
<td>Uses separate CNNs to model closeness and periods in temporal dependency.</td>
</tr>
<tr>
<td></td>
<td>Chen et al. [79]</td>
<td>Cloud RAN optimization</td>
<td>Multivariate LSTM</td>
<td>Unknown</td>
<td>Uses mobile traffic forecasting to aid cloud radio access network optimization</td>
</tr>
<tr>
<td></td>
<td>Navahi et al. [222]</td>
<td>Wireless WiFi channel feature prediction</td>
<td>MLP</td>
<td>SGD</td>
<td>Infers non-observable channel information from observable features.</td>
</tr>
<tr>
<td></td>
<td>Feng et al. [236]</td>
<td>Mobile cellular traffic prediction</td>
<td>LSTM</td>
<td>Adam</td>
<td>Extracts spatial and temporal dependencies using separate modules.</td>
</tr>
<tr>
<td></td>
<td>Alaowe et al. [471]</td>
<td>Mobile traffic load forecasting</td>
<td>MLN, LSTM</td>
<td>Unknown</td>
<td>Employ traffic forecasting to improve 5G network scalability.</td>
</tr>
<tr>
<td></td>
<td>Wang et al. [104]</td>
<td>Cellular traffic prediction</td>
<td>Graph neural networks</td>
<td>Pineda algorithm</td>
<td>Represents spatio-temporal dependency via graphs and first work employing Graph neural networks for traffic forecasting.</td>
</tr>
<tr>
<td></td>
<td>Fang et al. [233]</td>
<td>Mobile demand forecasting</td>
<td>Graph CNN, LSTM, Spatio-temporal graph</td>
<td>Unknown</td>
<td>Modelling the spatial correlations between cells using a dependency graph.</td>
</tr>
<tr>
<td></td>
<td>Luo et al. [234]</td>
<td>Channel state information prediction</td>
<td>CNN and LSTM</td>
<td>RMSprop</td>
<td>Employing a two-stage offline-online training scheme to improve the stability of framework.</td>
</tr>
<tr>
<td></td>
<td>Wang et al. [224]</td>
<td>Encrypted traffic classification</td>
<td>CNN</td>
<td>SGD</td>
<td>Employs an end-to-end deep learning approach to perform encrypted traffic classification.</td>
</tr>
<tr>
<td></td>
<td>Lotfollahi et al. [225]</td>
<td>Encrypted traffic classification</td>
<td>CNN</td>
<td>Adam</td>
<td>Can perform both traffic characterization and application identification.</td>
</tr>
<tr>
<td></td>
<td>Wang et al. [226]</td>
<td>Malware traffic classification</td>
<td>CNN</td>
<td>SGD</td>
<td>First work to use representation learning for malware classification from raw traffic.</td>
</tr>
<tr>
<td></td>
<td>Aceto et al. [472]</td>
<td>Mobile encrypted traffic classification</td>
<td>MLP, CNN, LSTM</td>
<td>SGD, Adam</td>
<td>Comprehensive evaluations of different NN architectures and excellent performance.</td>
</tr>
<tr>
<td></td>
<td>Li et al. [235]</td>
<td>Network traffic classification</td>
<td>Bayesian auto-encoder</td>
<td>SGD</td>
<td>Applying Bayesian probability theory to obtain the a posteriori distribution of model parameters.</td>
</tr>
<tr>
<td>CDR mining</td>
<td>Liang et al. [227]</td>
<td>Metro density prediction</td>
<td>RNN</td>
<td>SGD</td>
<td>Employs geo-spatial data processing, a weight-sharing RNN and parallel stream analytic programming.</td>
</tr>
<tr>
<td></td>
<td>Felbo et al. [228]</td>
<td>Demographics prediction</td>
<td>CNN</td>
<td>Adam</td>
<td>Exploits the temporal correlation inherent to mobile phone metadata.</td>
</tr>
<tr>
<td></td>
<td>Chen et al. [229]</td>
<td>Tourists’ next visit location prediction</td>
<td>MLP, RNN</td>
<td>LSTM</td>
<td>LSTM that performs significantly better than other ML approaches.</td>
</tr>
<tr>
<td></td>
<td>Lin et al. [230]</td>
<td>Human activity chains generation</td>
<td>Input-Output HMM + LSTM</td>
<td>Adam</td>
<td>First work that uses an RNN to generate human activity chains.</td>
</tr>
<tr>
<td>Others</td>
<td>Xu et al. [231]</td>
<td>Wi-Fi hotpot classification</td>
<td>CNN</td>
<td>Unknown</td>
<td>Combining deep learning with frequency analysis.</td>
</tr>
<tr>
<td></td>
<td>Meng et al. [232]</td>
<td>QoE-driven big data analysis</td>
<td>CNN</td>
<td>SGD</td>
<td>Investigates trade-off between accuracy of high-dimensional big data analysis and model training speed.</td>
</tr>
</tbody>
</table>
Moving Average (ARIMA) model. The work in [218] extends mobile traffic forecasting to long time frames. The authors combine ConvLSTMs and 3D CNNs to construct spatio-temporal neural networks that capture the complex spatio-temporal features at city scale. They further introduce a fine-tuning scheme and lightweight approach to blend predictions with historical means, which significantly extends the length of reliable prediction steps. Deep learning was also employed in [220], [236], [471] and [79], where the authors employ CNNs and LSTMs to perform mobile traffic forecasting. By effectively extracting spatio-temporal features, their proposals gain significantly higher accuracy than traditional approaches, such as ARIMA. Wang et al. represent spatio-temporal dependencies in mobile traffic using graphs, and learn such dependencies using Graph Neural Networks [104]. Beyond the accurate inference achieved in their study, this work also demonstrates potential for precise social events inference.

More recently, Zhang et al. propose an original Mobile Traffic Super-Resolution (MTSR) technique to infer network-wide fine-grained mobile traffic consumption given coarse-grained counterparts obtained by probing, thereby reducing traffic measurement overheads [219]. We illustrate the principle of MTSR in Fig. 11. Inspired by image super-resolution techniques, they design a dedicated CNN with multiple skip connections between layers, named deep zipper network, along with a Generative Adversarial Network (GAN) to perform precise MTSR and improve the fidelity of inferred traffic snapshots. Experiments with a real-world dataset show that this architecture can improve the granularity of mobile traffic measurements over a city by up to 100×, while significantly outperforming other interpolation techniques.

**Traffic Classification** is aimed at identifying specific applications or protocols among the traffic in networks. Wang recognizes the powerful feature learning ability of deep neural networks and uses a deep AE to identify protocols in a TCP flow dataset, achieving excellent precision and recall rates [223]. Work in [224] proposes to use a 1D CNN for encrypted traffic classification. The authors suggest that this structure works well for modeling sequential data and has lower complexity, thus being promising in addressing the traffic classification problem. Similarly, Lotfollahi et al. present Deep Packet, which is based on a CNN, for encrypted traffic classification [225]. Their framework reduces the amount of hand-crafted feature engineering and achieves great accuracy. An improved stacked AE is employed in [235], where Li et al. incorporate Bayesian methods into AEs to enhance the inference accuracy in network traffic classification. More recently, Aceto et al. employ MLPs, CNNs, and LSTMs to perform encrypted mobile traffic classification [472], arguing that deep NNs can automatically extract complex features present in mobile traffic. As reflected by their results, deep learning based solutions obtain superior accuracy over RFs in classifying Android, IOS and Facebook traffic. CNNs have also been used to identify malware traffic, where work in [226] regards traffic data as images and unusual patterns that malware traffic exhibit are classified by representation learning. Similar work on mobile malware detection will be further discussed in subsection VI-H.

**CDR Mining** involves extracting knowledge from specific instances of telecommunication transactions such as phone number, cell ID, session start/end time, traffic consumption, etc. Using deep learning to mine useful information from CDR data can serve a variety of functions. For example, Liang et al. propose Mercury to estimate metro density from streaming CDR data, using RNNs [227]. They take the trajectory of a mobile phone user as a sequence of locations; RNN-based models work well in handling such sequential data. Likewise, Felbo et al. use CDR data to study demographics [228]. They employ a CNN to predict the age and gender of mobile users, demonstrating the superior accuracy of these structures over other ML tools. More recently, Chen et al. compare different ML models to predict tourists’ next locations of visit by analyzing CDR data.
messengers that constantly send data to cloud servers, via local computing treats mobile devices as data collectors and in Fig. 12. As shown in the left part of the figure, the cloud-computing. We illustrate the difference between these scenarios

(i)

provide a powerful solution for app-level data mining, as performs relatively poorly. Advanced deep learning practices application requirements. The drawback of this scenario is that constantly sending and receiving messages to/from servers over the Internet introduces overhead and may result in severe latency. In contrast, in the edge-based computing scenario pre-trained models are offloaded from the cloud to individual mobile devices, such that they can make inferences locally. As illustrated in the right part of Fig. 12, this scenario typically consists of the following: (i) servers use offline datasets to per-train a model; (ii) the pre-trained model is offloaded to edge devices; (iii) mobile devices perform inferences locally using the model; (iv) cloud servers accept data from local devices; (v) the model is updated using these data whenever necessary. While this scenario requires less interactions with the cloud, its applicability is limited by the computing and battery capabilities of edge hardware. Therefore, it can only support tasks that require light computations.

Many researchers employ deep learning for app-level mobile data analysis. We group the works reviewed according to their application domains, namely mobile healthcare, mobile pattern recognition, and mobile Natural Language Processing (NLP) and Automatic Speech Recognition (ASR). Table XII gives a high-level summary of existing research efforts and we discuss representative work next.

Mobile Health. There is an increasing variety of wearable health monitoring devices being introduced to the market. By incorporating medical sensors, these devices can capture the physical conditions of their carriers and provide real-time feedback (e.g., heart rate, blood pressure, breath status etc.), or trigger alarms to remind users of taking medical actions [477].

Liu and Du design a deep learning-driven MobiEar to aid deaf people’s awareness of emergencies [238]. Their proposal accepts acoustic signals as input, allowing users to register different acoustic events of interest. MobiEar operates efficiently on smart phones and only requires infrequent communications with servers for updates. Likewise, Liu et al. develop a UbiEar, which is operated on the Android platform to assist hard-of-hear sufferers in recognizing acoustic events, without requiring location information [239]. Their design adopts a lightweight CNN architecture for inference acceleration and demonstrates comparable accuracy over traditional CNN models.

Hosseini et al. design an edge computing system for health monitoring and treatment [244]. They use CNNs to extract features from mobile sensor data, which plays an important role in their epileptogenicity localization application. Stamate et al. develop a mobile Android app called cloudUPDRS to manage Parkinson’s symptoms [245]. In their work, MLPs are employed to determine the acceptance of data collected by smart phones, to maintain high-quality data samples. The proposed method outperforms other ML methods such as GPs and RFs. Quisel et al. suggest that deep learning can be effec-

34Human profile source: https://lekeart.deviantart.com/art/male-body-profile-251793336

points of access with limited data preprocessing capabilities. This scenario typically includes the following steps: (i) users query on/interact with local mobile devices; (ii) queries are transmitted to servers in the cloud; (iii) servers gather the data received for model training and inference; (iv) query results are subsequently sent back to each device, or stored and analyzed without further dissemination, depending on specific application requirements. The drawback of this scenario is that constantly sending and receiving messages to/from servers over the Internet introduces overhead and may result in severe latency. In contrast, in the edge-based computing scenario pre-trained models are offloaded from the cloud to individual mobile devices, such that they can make inferences locally. As illustrated in the right part of Fig. 12, this scenario typically consists of the following: (i) servers use offline datasets to per-train a model; (ii) the pre-trained model is offloaded to edge devices; (iii) mobile devices perform inferences locally using the model; (iv) cloud servers accept data from local devices; (v) the model is updated using these data whenever necessary. While this scenario requires less interactions with the cloud, its applicability is limited by the computing and battery capabilities of edge hardware. Therefore, it can only support tasks that require light computations.

Many researchers employ deep learning for app-level mobile data analysis. We group the works reviewed according to their application domains, namely mobile healthcare, mobile pattern recognition, and mobile Natural Language Processing (NLP) and Automatic Speech Recognition (ASR). Table XII gives a high-level summary of existing research efforts and we discuss representative work next.

Mobile Health. There is an increasing variety of wearable health monitoring devices being introduced to the market. By incorporating medical sensors, these devices can capture the physical conditions of their carriers and provide real-time feedback (e.g., heart rate, blood pressure, breath status etc.), or trigger alarms to remind users of taking medical actions [477].

Liu and Du design a deep learning-driven MobiEar to aid deaf people’s awareness of emergencies [238]. Their proposal accepts acoustic signals as input, allowing users to register different acoustic events of interest. MobiEar operates efficiently on smart phones and only requires infrequent communications with servers for updates. Likewise, Liu et al. develop a UbiEar, which is operated on the Android platform to assist hard-of-hear sufferers in recognizing acoustic events, without requiring location information [239]. Their design adopts a lightweight CNN architecture for inference acceleration and demonstrates comparable accuracy over traditional CNN models.

Hosseini et al. design an edge computing system for health monitoring and treatment [244]. They use CNNs to extract features from mobile sensor data, which plays an important role in their epileptogenicity localization application. Stamate et al. develop a mobile Android app called cloudUPDRS to manage Parkinson’s symptoms [245]. In their work, MLPs are employed to determine the acceptance of data collected by smart phones, to maintain high-quality data samples. The proposed method outperforms other ML methods such as GPs and RFs. Quisel et al. suggest that deep learning can be effec-
Mobile Pattern Recognition. Recent advanced mobile devices offer people a portable intelligent assistant, which fosters a diverse set of applications that can classify surrounding objects (e.g., [248]–[250], [253]) or users’ behaviors (e.g., [114], [255], [258], [264], [265], [479], [480]) based on patterns observed in the output of the mobile camera or other sensors. We review and compare recent works on mobile pattern recognition in this part.

Object classification in pictures taken by mobile devices is drawing increasing research interest. Li et al. develop DeepCham as a mobile object recognition framework [248]. Their architecture involves a crowd-sourcing labeling process, which aims to reduce the hand-labeling effort, and a collaborative training instance generation pipeline that is built for deployment on mobile devices. Evaluations of the prototype system suggest that this framework is efficient and effective in terms of training and inference. Tobias et al. investigate the applicability of employing CNN schemes on mobile devices for objection recognition tasks [249]. They conduct experiments on three different model deployment scenarios, i.e., on GPU, CPU, and respectively on mobile devices, with two benchmark datasets. The results obtained suggest that deep learning models can be efficiently embedded in mobile devices to perform real-time inference.

Mobile classifiers can also assist Virtual Reality (VR) applications. A CNN framework is proposed in [253] for facial expressions recognition when users are wearing head-mounted displays in the VR environment. Rao et al. incorporate a deep learning object detector into a mobile augmented reality (AR) system [254]. Their system achieves outstanding performance in detecting and enhancing geographic objects in outdoor environments. Further work focusing on mobile AR applications is introduced in [481], where the authors characterize the tradeoffs between accuracy, latency, and energy efficiency of object detection.

Activity recognition is another interesting area that relies on data collected by mobile motion sensors [480], [482]. This refers to the ability to classify based on data collected via, e.g., video capture, accelerometer readings, motion – Passive Infra-Red (PIR) sensing, specific actions and activities that a human subject performs. Data collected will be delivered to servers for model training and the model will be subsequently deployed for domain-specific tasks.

Essential features of sensor data can be automatically extracted by neural networks. The first work in this space that is based on deep learning employs a CNN to capture local dependencies and preserve scale invariance in motion sensor data [255]. The authors evaluate their proposal on 3 offline datasets, demonstrating their proposal yields higher accuracy over statistical methods and Principal Components Analysis (PCA). Almaslukh et al. employ a deep AE to perform human activity recognition by analyzing an offline smart phone dataset gathered from accelerometers and gyroscope sensors [256]. Li et al. consider different scenarios for activity recognition [257]. In their implementation, Radio Frequency Identification (RFID) data is directly sent to a CNN model for recognizing human activities. While their mechanism achieves
TABLE XII: A summary of works on app-level mobile data analysis.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Reference</th>
<th>Application</th>
<th>Deployment</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile Healthcare</td>
<td>Liu and Du [238]</td>
<td>Mobile ear</td>
<td>Edge-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Liu et al. [239]</td>
<td>Mobile ear</td>
<td>Edge-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Jandali [240]</td>
<td>Heart rate prediction</td>
<td>Cloud-based</td>
<td>DBN</td>
</tr>
<tr>
<td></td>
<td>Kim et al. [241]</td>
<td>Cytopathology classification</td>
<td>Cloud-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Sathiyaranarayana et al. [242]</td>
<td>Sleep quality prediction</td>
<td>Cloud-based</td>
<td>MLP, CNN, LSTM</td>
</tr>
<tr>
<td></td>
<td>Li and Titican [243]</td>
<td>Health conditions analysis</td>
<td>Cloud-based</td>
<td>Stacked AE</td>
</tr>
<tr>
<td></td>
<td>Hosseini et al. [244]</td>
<td>Epileptogenicity localisation</td>
<td>Cloud-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Stamate et al. [245]</td>
<td>Parkinson’s symptoms management</td>
<td>Cloud-based</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Quasel et al. [246]</td>
<td>Mobile health data analysis</td>
<td>Cloud-based</td>
<td>CNN, RNN</td>
</tr>
<tr>
<td></td>
<td>Khan et al. [247]</td>
<td>Respiration surveillance</td>
<td>Cloud-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Li et al. [248]</td>
<td>Mobile object recognition</td>
<td>Edge-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Tobias et al. [249]</td>
<td>Mobile object recognition</td>
<td>Edge-based &amp; Cloud based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Pouliadzadeh and Shmohammadi [250]</td>
<td>Food recognition system</td>
<td>Cloud-based</td>
<td>CNN</td>
</tr>
<tr>
<td>Mobile Pattern Recognition</td>
<td>Tanno et al. [251]</td>
<td>Food recognition system</td>
<td>Edge-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Kahad et al. [252]</td>
<td>Food recognition system</td>
<td>Cloud-based</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Ieng and Yang [253]</td>
<td>Facial recognition</td>
<td>Cloud-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Wu et al. [254]</td>
<td>Mobile visual search</td>
<td>Edge-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Kao et al. [254]</td>
<td>Mobile augmented reality</td>
<td>Edge-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Ohata et al. [255]</td>
<td>WiFi-driven indoor change detection</td>
<td>Cloud-based</td>
<td>CNN, LSTM</td>
</tr>
<tr>
<td></td>
<td>Zeng et al. [256]</td>
<td>Activity recognition</td>
<td>Cloud-based</td>
<td>CNN, LSTM</td>
</tr>
<tr>
<td></td>
<td>Almaslihil et al. [256]</td>
<td>Activity recognition</td>
<td>Cloud-based</td>
<td>AE</td>
</tr>
<tr>
<td></td>
<td>Li et al. [257]</td>
<td>RFID-based activity recognition</td>
<td>Cloud-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Bhattacharya and Lane [258]</td>
<td>Smart watch-based activity recognition</td>
<td>Edge-based</td>
<td>RBM</td>
</tr>
<tr>
<td>Mobile NLP and ASR</td>
<td>Antreas and Angelov [259]</td>
<td>Mobile surveillance system</td>
<td>Edge-based &amp; Cloud based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Ordóñez and Roogen [114]</td>
<td>Activity recognition</td>
<td>Cloud-based</td>
<td>ConvLSTM</td>
</tr>
<tr>
<td></td>
<td>Wang et al. [260]</td>
<td>Gesture recognition</td>
<td>Edge-based</td>
<td>CNN, RNN</td>
</tr>
<tr>
<td></td>
<td>Gao et al. [261]</td>
<td>Eating detection</td>
<td>Cloud-based</td>
<td>DBM, MLP</td>
</tr>
<tr>
<td></td>
<td>Zhu et al. [262]</td>
<td>User energy expenditure estimation</td>
<td>Cloud-based</td>
<td>CNN, MLP</td>
</tr>
<tr>
<td></td>
<td>Sundaysy et al. [263]</td>
<td>Individual income classification</td>
<td>Cloud-based</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Chen and Xue [264]</td>
<td>Activity recognition</td>
<td>Cloud-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Ha and Choi [265]</td>
<td>Activity recognition</td>
<td>Cloud-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Edel and Kippe [266]</td>
<td>Activity recognition</td>
<td>Edge-based</td>
<td>Binarized-LSTM</td>
</tr>
<tr>
<td></td>
<td>Oka and Inoue [209]</td>
<td>Multiple overlapping activities recognition</td>
<td>Cloud-based</td>
<td>CNN+LSTM</td>
</tr>
<tr>
<td></td>
<td>Alshehik et al. [17]</td>
<td>Activity recognition using Apache Spark</td>
<td>Cloud-based</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Mittal et al. [270]</td>
<td>Garbage detection</td>
<td>Edge-based &amp; Cloud based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Seidenari et al. [271]</td>
<td>Artwork detection and retrieval</td>
<td>Edge-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Zeng et al. [272]</td>
<td>Mobile pill classification</td>
<td>Edge-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Lane and Georgiev [75]</td>
<td>Mobile activity recognition, emotion recognition and speaker identification</td>
<td>Edge-based</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Yao et al. [292]</td>
<td>Car tracking, heterogeneous human activity recognition and user identification</td>
<td>Edge-based</td>
<td>CNN, RNN</td>
</tr>
<tr>
<td></td>
<td>Zeng et al. [273]</td>
<td>IoT human activity recognition</td>
<td>Cloud-based</td>
<td>AE, CNN, LSTM</td>
</tr>
<tr>
<td></td>
<td>Zou et al. [273]</td>
<td>Mobile object recognition</td>
<td>Edge-based</td>
<td>Unknown</td>
</tr>
<tr>
<td></td>
<td>Katevas et al. [291]</td>
<td>Notification attendance prediction</td>
<td>Edge-based</td>
<td>RNN</td>
</tr>
<tr>
<td></td>
<td>Rada et al. [1380]</td>
<td>Activity recognition</td>
<td>Cloud-based</td>
<td>RNN</td>
</tr>
<tr>
<td></td>
<td>Wang et al. [275], [276]</td>
<td>Activity and gesture recognition</td>
<td>Cloud-based</td>
<td>Stacked AE</td>
</tr>
<tr>
<td></td>
<td>Feing et al. [277]</td>
<td>Activity detection</td>
<td>Cloud-based</td>
<td>LSTM</td>
</tr>
<tr>
<td></td>
<td>Cao et al. [278]</td>
<td>Mood detection</td>
<td>Cloud-based</td>
<td>GRU</td>
</tr>
<tr>
<td></td>
<td>Ran et al. [279]</td>
<td>Object detection for AR applications.</td>
<td>Edge-based &amp; cloud based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Zhao et al. [290]</td>
<td>Estimating 3D human skeleton from radio frequency signal</td>
<td>Cloud-based</td>
<td>CNN</td>
</tr>
<tr>
<td>Others</td>
<td>Siri [280]</td>
<td>Speech synthesis</td>
<td>Edge-based</td>
<td>Mixture density networks</td>
</tr>
<tr>
<td></td>
<td>McGraw et al. [281]</td>
<td>Personalised speech recognition</td>
<td>Edge-based</td>
<td>LSTM</td>
</tr>
<tr>
<td></td>
<td>Prabhavalkar et al. [282]</td>
<td>Embedded speech recognition</td>
<td>Edge-based</td>
<td>LSTM</td>
</tr>
<tr>
<td></td>
<td>Yosuoka et al. [283]</td>
<td>Mobile speech recognition</td>
<td>Cloud-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Ruan et al. [284]</td>
<td>Shifting from typing to speech</td>
<td>Cloud-based</td>
<td>Unknown</td>
</tr>
<tr>
<td></td>
<td>Georgiev et al. [99]</td>
<td>Multi-task mobile audio sensing</td>
<td>Edge-based</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Ignatov et al. [255]</td>
<td>Mobile images quality enhancement</td>
<td>Cloud-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Lu et al. [286]</td>
<td>Information retrieval from videos in wireless network</td>
<td>Cloud-based</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Lee et al. [287]</td>
<td>Reducing distraction for smartwatch users</td>
<td>Cloud-based</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Vu et al. [288]</td>
<td>Transportation mode detection</td>
<td>Cloud-based</td>
<td>RNN</td>
</tr>
<tr>
<td></td>
<td>Fang et al. [289]</td>
<td>Transportation mode detection</td>
<td>Cloud-based</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Xue et al. [267]</td>
<td>Mobile App classification</td>
<td>Cloud-based</td>
<td>AE, MLP, CNN, and LSTM</td>
</tr>
<tr>
<td></td>
<td>Liu et al. [268]</td>
<td>Mobile motion sensor fingerprinting</td>
<td>Cloud-based</td>
<td>LSTM</td>
</tr>
</tbody>
</table>
high accuracy in different applications, experiments suggest that the RFID-based method does not work well with metal objects or liquid containers.

[258] exploits an RBM to predict human activities, given 7 types of sensor data collected by a smart watch. Experiments on prototype devices show that this approach can efficiently fulfill the recognition objective under tolerable power requirements. Ordóñez and Roggen architect an advanced ConvLSTM to fuse data gathered from multiple sensors and perform activity recognition [114]. By leveraging CNN and LSTM structures, ConvLSTMs can automatically compress spatio-temporal sensor data into low-dimensional representations, without heavy data post-processing effort. Wang et al. exploit Google Soli to architect a mobile user-machine interaction platform [260]. By analyzing radio frequency signals captured by millimeter-wave radars, their architecture is able to recognize 11 types of gestures with high accuracy. Their models are trained on the server side, and inferences are performed locally on mobile devices. More recently, Zhao et al. design a 4D CNN framework (3D for the spatial dimension + 1D for the temporal dimension) to reconstruct human skeletons using radio frequency signals [290]. This novel approach resembles virtual “X-ray”, enabling to accurately estimate human poses, without requiring an actual camera.

Mobile NLP and ASR. Recent remarkable achievements obtained by deep learning in Natural Language Processing (NLP) and Automatic Speech Recognition (ASR) are also embraced by applications for mobile devices.

Powered by deep learning, the intelligent personal assistant Siri, developed by Apple, employs a deep mixture density networks [483] to fix typical robotic voice issues and synthesized more human-like voice [280]. An Android app released by Google supports mobile personalized speech recognition [281]; this quantizes the parameters in LSTM model compression, allowing the app to run on low-power mobile phones. Likewise, Prabhavalkar et al. propose a mathematical RNN compression technique that reduces two thirds of an LSTM acoustic model size, while only compromising negligible accuracy [282]. This allows building both memory- and energy-efficient ASR applications on mobile devices.

Yoshioka et al. present a framework that incorporates a network-in-network architecture into a CNN model, which allows to perform ASR with mobile multi-microphone devices used in noisy environments [283]. Mobile ASR can also accelerate text input on mobile devices, Ruan et al.’s study showing that with the help of ASR, the input rates of English and Mandarin are 3.0 and 2.8 times faster over standard typing on keyboards [284]. More recently, the applicability of deep learning to multi-task audio sensing is investigated in [99], where Georgiev et al. propose and evaluate a novel deep learning modelling and optimization framework tailored to embedded audio sensing tasks. To this end, they selectively share compressed representations between different tasks, which reduces training and data storage overhead, without significantly compromising accuracy of an individual task. The authors evaluate their framework on a memory-constrained smartphone performing four audio tasks (i.e., speaker identification, emotion recognition, stress detection, and ambient scene analysis). Experiments suggest this proposal can achieve high efficiency in terms of energy, runtime and memory, while maintaining excellent accuracy.

Other applications. Deep learning also plays an important role in other applications that involve app-level data analysis. For instance, Ignatov et al. show that deep learning can enhance the quality of pictures taken by mobile phones. By employing a CNN, they successfully improve the quality of images obtained by different mobile devices, to a digital single-lens reflex camera level [285]. Lu et al. focus on video post-processing under wireless networks [286], where their framework exploits a customized AlexNet to answer queries about detected objects. This framework further involves an optimizer, which instructs mobile devices to offload videos, in order to reduce query response time.

Another interesting application is presented in [287], where Lee et al. show that deep learning can help smartwatch users reduce distraction by eliminating unnecessary notifications. Specifically, the authors use an 11-layer MLP to predict the importance of a notification. Fang et al. exploit an MLP to extract features from high-dimensional and heterogeneous sensor data, including accelerometer, magnetometer, and gyroscope measurements [289]. Their architecture achieves 95% accuracy in recognizing human transportation modes, i.e., still, walking, running, biking, and on vehicle.

Lessons Learned: App-level data is heterogeneous and generated from distributed mobile devices, and there is a trend to offload the inference process to these devices. However, due to computational and battery power limitations, models employed in the edge-based scenario are constrained to lightweight architectures, which are less suitable for complex tasks. Therefore, the trade-off between model complexity and accuracy should be carefully considered [68]. Numerous efforts were made towards tailoring deep learning to mobile devices, in order to make algorithms faster and less energy-consuming on embedded equipment. For example, model compression, pruning, and quantization are commonly used for this purpose. Mobile device manufacturers are also developing new software and hardware to support deep learning based applications. We will discuss this work in more detail in Sec. VII.

At the same time, app-level data usually contains important users information and processing this poses significant privacy concerns. Although there have been efforts that commit to preserve user privacy, as we discuss in Sec.VI-H, research efforts in this direction are new, especially in terms of protecting user information in distributed training. We expect more efforts in this direction in the future.

D. Deep Learning Driven Mobility Analysis

Understanding movement patterns of groups of human beings and individuals is becoming crucial for epidemiology, urban planning, public service provisioning, and mobile network resource management [484]. Deep learning is gaining
TABLE XIII: A summary of work on deep learning driven mobility analysis.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Application</th>
<th>Mobility level</th>
<th>Model</th>
<th>Key contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ouyang et al. [295]</td>
<td>Mobile user trajectory prediction</td>
<td>Individual</td>
<td>CNN</td>
<td>Online framework for data stream processing.</td>
</tr>
<tr>
<td>Yang et al. [296]</td>
<td>Social networks and mobile trajectories modeling</td>
<td>Mobile Ad-hoc Network</td>
<td>RNN, GRU</td>
<td>Multi-task learning.</td>
</tr>
<tr>
<td>Tkačík and Kordík [308]</td>
<td>Mobility modelling and prediction</td>
<td>Individual</td>
<td>Neural Turing Machine</td>
<td>The Neural Turing Machine can store historical data and perform “read” and “write” operations automatically.</td>
</tr>
<tr>
<td>Song et al. [297]</td>
<td>City-wide mobility prediction and transportation modeling</td>
<td>City-wide</td>
<td>Multi-task LSTM</td>
<td>Multi-task learning.</td>
</tr>
<tr>
<td>Zhang et al. [298]</td>
<td>City-wide crowd flows prediction</td>
<td>City-wide</td>
<td>Deep spatio-temporal residual networks (CNN-based)</td>
<td>Exploitation of spatio-temporal characteristics of mobility events.</td>
</tr>
<tr>
<td>Subramanian and Sadiq [299]</td>
<td>Mobile movement prediction</td>
<td>Individual</td>
<td>MLP</td>
<td>Fewer location updates and lower paging signaling costs.</td>
</tr>
<tr>
<td>Ezema and Ani [300]</td>
<td>Mobile location estimation</td>
<td>Individual</td>
<td>MLP</td>
<td>Operates with received signal strength in GSM.</td>
</tr>
<tr>
<td>Shao et al. [301]</td>
<td>CNN driven Pedometer</td>
<td>Individual</td>
<td>CNN</td>
<td>Reduced false negatives caused by periodic movements and lower initial response time.</td>
</tr>
<tr>
<td>Yayeh et al. [302]</td>
<td>Mobility prediction in mobile Ad-hoc network</td>
<td>Individual</td>
<td>MLP</td>
<td>Achieving high prediction accuracy under random waypoint mobility model.</td>
</tr>
<tr>
<td>Chen et al. [303]</td>
<td>Mobility driven traffic accident risk prediction</td>
<td>City-wide</td>
<td>Stacked denoising AE</td>
<td>Automatically learning correlation between human mobility and traffic accident risk.</td>
</tr>
<tr>
<td>Song et al. [304]</td>
<td>Human emergency behavior and mobility modelling</td>
<td>City-wide</td>
<td>DBN</td>
<td>Achieves accurate prediction over various disaster events, including earthquake, tsunami and nuclear accidents.</td>
</tr>
<tr>
<td>Yao et al. [305]</td>
<td>Trajectory clustering</td>
<td>User group</td>
<td>sequence-to-sequence AE with RNNs</td>
<td>The learned representations can robustly encode the movement characteristics of the objects and generate spatio-temporally invariant clusters.</td>
</tr>
<tr>
<td>Liu et al. [306]</td>
<td>Urban traffic prediction</td>
<td>City-wide</td>
<td>CNN, RNN, LSTM, AE, and RBM</td>
<td>Reveals the potential of employing deep learning to urban traffic prediction with mobility data.</td>
</tr>
<tr>
<td>Wickramasuriya et al. [307]</td>
<td>Base station prediction with proactive mobility management</td>
<td>Individual</td>
<td>RNN</td>
<td>Employs proactive and anticipatory mobility management for dynamic base station selection.</td>
</tr>
<tr>
<td>Kim and Song [309]</td>
<td>User mobility and personality modelling</td>
<td>Individual</td>
<td>MLP, RBM</td>
<td>Foundation for the customization of location-based services.</td>
</tr>
<tr>
<td>Jiang et al. [310]</td>
<td>Short-term urban mobility prediction</td>
<td>City-wide</td>
<td>RNN</td>
<td>Superior prediction accuracy and verified as highly deployable prototype system.</td>
</tr>
<tr>
<td>Wang et al. [311]</td>
<td>Mobility management in dense networks</td>
<td>Individual</td>
<td>LSTM</td>
<td>Improves the quality of service of mobile users in the handover process, while maintaining network energy efficiency.</td>
</tr>
<tr>
<td>Jiang et al. [312]</td>
<td>Urban human mobility prediction</td>
<td>City-wide</td>
<td>RNN</td>
<td>First work that utilizes urban region of interest to model human mobility city-wide.</td>
</tr>
<tr>
<td>Feng et al. [313]</td>
<td>Human mobility forecasting</td>
<td>Individual</td>
<td>Attention RNN</td>
<td>Employs the attention mechanism and combines heterogeneous transition regularity and multi-level periodicity.</td>
</tr>
</tbody>
</table>

increasing attention in this area, both from a group and individual level perspective (see Fig. 13). In this subsection, we thus discuss research using deep learning in this space, which we summarize in Table XIII.

Since deep learning is able to capture spatial dependencies in sequential data, it is becoming a powerful tool for mobility analysis. The applicability of deep learning for trajectory prediction is studied in [485]. By sharing representations learned by RNN and Gate Recurrent Unit (GRU), the framework can perform multi-task learning on both social networks and mobile trajectories modeling. Specifically, the authors first use deep learning to reconstruct social network representations of users, subsequently employing RNN and GRU models to learn patterns of mobile trajectories with different time granularity. Importantly, these two components jointly share representations learned, which tightens the overall architecture and enables efficient implementation. Ouyang et al. argue that mobility data are normally high-dimensional, which may be problematic for traditional ML models. Therefore, they build upon deep learning advances and propose an online learning

![Fig. 13: Illustration of mobility analysis paradigms at individual (left) and group (right) levels.](image-url)
scheme to train a hierarchical CNN architecture, allowing model parallelization for data stream processing [295]. By analyzing usage records, their framework “DeepSpace” predicts individuals’ trajectories with much higher accuracy as compared to naive CNNs, as shown with experiments on a real-world dataset. In [308], Tkačik and Kordík design a Neural Turing Machine [486] to predict trajectories of individuals using mobile phone data. The Neural Turing Machine embraces two major components: a memory module to store the historical trajectories, and a controller to manage the “read” and “write” operations over the memory. Experiments show that their architecture achieves superior generalization over stacked RNN and LSTM, while also delivering more precise trajectory prediction than the n-grams and k nearest neighbor methods.

Instead of focusing on individual trajectories, Song et al. shed light on the mobility analysis at a larger scale [297]. In their work, LSTM networks are exploited to jointly model the city-wide movement patterns of a large group of people and vehicles. Their multi-task architecture demonstrates superior prediction accuracy over a standard LSTM. City-wide mobile patterns is also researched in [298], where the authors architect deep spatio-temporal residual networks to forecast the movements of crowds. In order to capture the unique characteristics of spatio-temporal correlations associated with human mobility, their framework abandons RNN-based models and constructs three ResNets to extract nearby and distant spatial dependencies within a city. This scheme learns temporal features and fuses representations extracted by all models for the final prediction. By incorporating external events information, their proposal achieves the highest accuracy among all deep learning and non-deep learning methods studied. An RNN is also employed in [310], where Jiang et al. perform short-term urban mobility forecasting on a huge dataset collected from a real-world deployment. Their model delivers superior accuracy over the n-gram and Markovian approaches.

Lin et al. consider generating human movement chains from cellular data, to support transportation planning [230]. In particular, they first employ an input-output Hidden Markov Model (HMM) to label activity profiles for CDR data preprocessing. Subsequently, an LSTM is designed for activity chain generation, given the labeled activity sequences. They further synthesize urban mobility plans using the generative model and the simulation results reveal reasonable fit accuracy. Jiang et al. design 24-h mobility prediction system base on RNN mdoels [312]. They employ dynamic Region of Interests (ROIs) for each hour to discovered through divide-and-merge mining from raw trajectory database, which leads to high prediction accuracy. Feng et al. incorporate attention mechanisms on RNN [313], to capture the complicated sequential transitions of human mobility. By combining the heterogeneous transition regularity and multi-level periodicity, their model delivers up to 10% of accuracy improvement compared to state-of-the-art forecasting models.

Yayeh et al. employ an MLP to predict the mobility of mobile devices in mobile ad-hoc networks, given previously observed pause time, speed, and movement direction [302]. Simulations conducted using the random waypoint mobility model show that their proposal achieves high prediction accuracy. An MLP is also adopted in [309], where Kim and Song model the relationship between human mobility and personality, and achieve high prediction accuracy. Yao et al. discover groups of similar trajectories to facilitate higher-level mobility driven applications using RNNs [305]. Particularly, a sequence-to-sequence AE is adopted to learn fixed-length representations of mobile users’ trajectories. Experiments show that their method can effectively capture spatio-temporal patterns in both real and synthetic datasets. Shao et al. design a sophisticated pedometer using a CNN [301]. By reducing false negative steps caused by periodic movements, their proposal significantly improves the robustness of the pedometer.

In [303], Chen et al. combine GPS records and traffic accident data to understand the correlation between human mobility and traffic accidents. To this end, they design a stacked denoising AE to learn a compact representation of the human mobility, and subsequently use that to predict the traffic accident risk. Their proposal can deliver accurate, real-time prediction across large regions. GPS records are also used in other mobility-driven applications. Song et al. employ DBNs to predict and simulate human emergency behavior and mobility in natural disaster, learning from GPS records of 1.6 million users [304]. Their proposal yields accurate predictions in different disaster scenarios such as earthquakes, tsunamis, and nuclear accidents. GPS data is also utilized in [306], where Liu et al. study the potential of employing deep learning for urban traffic prediction using mobility data.

**Lessons learned:** Mobility analysis is concerned with the movement trajectory of a single user or large groups of users. The data of interest are essential time series, but have an additional spatial dimension. Mobility data is usually subject to stochasticity, loss, and noise; therefore precise modelling is not straightforward. As deep learning is able to perform automatic feature extraction, it becomes a strong candidate for human mobility modelling. Among them, CNNs and RNNs are the most successful architectures in such applications (e.g., [230], [295]–[298]), as they can effectively exploit spatial and temporal correlations.

E. Deep Learning Driven User Localization

Location-based services and applications (e.g. mobile AR, GPS) demand precise individual positioning technology [487]. As a result, research on user localization is evolving rapidly and numerous techniques are emerging [488]. In general, user localization methods can be categorized as device-based and device-free [489]. We illustrate the two different paradigms in Fig. 14. Specifically, in the first category specific devices carried by users become prerequisites for fulfilling the applications’ localization function. This type of approaches rely on signals from the device to identify the location. Conversely, approaches that require no device pertain to the device-free category. Instead these employ special equipment to monitor signal changes, in order to localize the entities of interest. Deep learning can enable high localization accuracy with both
paradigms. We summarize the most notable contributions in Table XIV and delve into the details of these works next.

To overcome the variability and coarse-granularity limitations of signal strength based methods, Wang et al. propose a deep learning driven fingerprinting system name “DeepFi” to perform indoor localization based on Channel State Information (CSI) [314]. Their toolbox yields much higher accuracy as compared to traditional methods, including including FIFS [490], Horus [491], and Maximum Likelihood [492]. The same group of authors extend their work in [275], [276] and [315], [316], where they update the localization system, such that it can work with calibrated phase information of CSI [275], [276], [326]. They further use more sophisticated CNN [315], [334] and bi-modal structures [316] to improve the accuracy.

Nowicki and Wietrzykowski propose a localization framework that reduces significantly the effort of system tuning or

<table>
<thead>
<tr>
<th>Reference</th>
<th>Application</th>
<th>Input data</th>
<th>Device requirement</th>
<th>Model</th>
<th>Key contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wang et al. [314]</td>
<td>Indoor fingerprinting</td>
<td>CSI</td>
<td>Device-based</td>
<td>RBM</td>
<td>First deep learning driven indoor localization based on CSI</td>
</tr>
<tr>
<td>Wang et al. [275], [276]</td>
<td>Indoor localization</td>
<td>CSI</td>
<td>Device-based</td>
<td>RBM</td>
<td>Works with calibrated phase information of CSI</td>
</tr>
<tr>
<td>Wang et al. [315]</td>
<td>Indoor localization</td>
<td>CSI</td>
<td>Device-based</td>
<td>CNN</td>
<td>Uses more robust angle of arrival for estimation</td>
</tr>
<tr>
<td>Wang et al. [316]</td>
<td>Indoor localization</td>
<td>CSI</td>
<td>Device-based</td>
<td>RBM</td>
<td>Bi-modal framework using both angle of arrival and average amplitudes of CSI</td>
</tr>
<tr>
<td>Nowicki and Wietrzykowski [317]</td>
<td>Indoor localization</td>
<td>WiFi scans</td>
<td>Device-based</td>
<td>Stacked AE</td>
<td>Requires less system tuning or filtering effort</td>
</tr>
<tr>
<td>Wang et al. [318], [319]</td>
<td>Indoor localization</td>
<td>Received signal strength</td>
<td>Device-free</td>
<td>Stacked AE</td>
<td>Device-free framework, multi-task learning</td>
</tr>
<tr>
<td>Mohammadi et al. [320]</td>
<td>Indoor localization</td>
<td>Received signal strength</td>
<td>Device-free</td>
<td>VAE+DQN</td>
<td>Handles unlabeled data; reinforcement learning aided semi-supervised learning</td>
</tr>
<tr>
<td>Anzum et al. [321]</td>
<td>Indoor localization</td>
<td>Received signal strength</td>
<td>Device-based</td>
<td>Counter propagation neural network</td>
<td>Solves the ambiguity among zones</td>
</tr>
<tr>
<td>Wang et al. [322]</td>
<td>Indoor localization</td>
<td>Smartphone magnetic and light sensors</td>
<td>Device-based</td>
<td>LSTM</td>
<td>Employ bimodal magnetic field and light intensity data</td>
</tr>
<tr>
<td>Kumar et al. [323]</td>
<td>Indoor vehicles localization</td>
<td>Camera images</td>
<td>Device-free</td>
<td>CNN</td>
<td>Focus on vehicles applications</td>
</tr>
<tr>
<td>Zheng and Weng [324]</td>
<td>Outdoor navigation</td>
<td>Camera images ad GPS</td>
<td>Device-free</td>
<td>Developmental network</td>
<td>Online learning scheme; edge-based</td>
</tr>
<tr>
<td>Zhang et al. [113]</td>
<td>Indoor and outdoor localization</td>
<td>Received signal strength</td>
<td>Device-based</td>
<td>Stacked AE</td>
<td>Operates under both indoor and outdoor environments</td>
</tr>
<tr>
<td>Vieira et al. [325]</td>
<td>Massive MIMO fingerprint-based positioning</td>
<td>Associated channel fingerprint</td>
<td>-</td>
<td>CNN</td>
<td>Operates with massive MIMO channels</td>
</tr>
<tr>
<td>Hsu et al. [336]</td>
<td>Activity recognition, localization and sleep monitoring</td>
<td>RF Signal</td>
<td>Device-free</td>
<td>CNN</td>
<td>Multi-user, device-free localization and sleep monitoring</td>
</tr>
<tr>
<td>Wang et al. [326]</td>
<td>Indoor localization</td>
<td>CSI</td>
<td>Device-based</td>
<td>RBM</td>
<td>Exploits the angle of arrival for stable indoor localization</td>
</tr>
<tr>
<td>Wang et al. [334]</td>
<td>Indoor localization</td>
<td>CSI</td>
<td>Device-based</td>
<td>CNN</td>
<td>Exploits the angle of arrival for stable indoor localization</td>
</tr>
<tr>
<td>Xiao et al. [335]</td>
<td>3D Indoor localization</td>
<td>Bluetooth relative received signal strength</td>
<td>Device-free</td>
<td>Denoising AE</td>
<td>Low-cost and robust localization</td>
</tr>
<tr>
<td>Niitsoo et al. [333]</td>
<td>Indoor localization</td>
<td>Raw channel impulse response data</td>
<td>Device-free</td>
<td>CNN</td>
<td>Robust to multipath propagation environments</td>
</tr>
<tr>
<td>Ibrahim et al. [332]</td>
<td>Indoor localization</td>
<td>Received signal strength</td>
<td>Device-based</td>
<td>CNN</td>
<td>100% accuracy in building and floor identification</td>
</tr>
<tr>
<td>Adege et al. [331]</td>
<td>Indoor localization</td>
<td>Received signal strength</td>
<td>Device-based</td>
<td>MLP + linear discriminant analysis</td>
<td>Accurate in multi-building environments</td>
</tr>
<tr>
<td>Zhang et al. [330]</td>
<td>Indoor localization</td>
<td>Pervasive magnetic field and CSI</td>
<td>Device-based</td>
<td>MLP</td>
<td>Using the magnetic field to improve the positioning accuracy</td>
</tr>
<tr>
<td>Zhou et al. [329]</td>
<td>Indoor localization</td>
<td>CSI</td>
<td>Device-free</td>
<td>MLP</td>
<td>Device-free localization</td>
</tr>
<tr>
<td>Shokry et al. [328]</td>
<td>Outdoor localization</td>
<td>Crowd-sensed received signal strength information</td>
<td>Device-based</td>
<td>MLP</td>
<td>More accurate than cellular localization while requiring less energy</td>
</tr>
<tr>
<td>Chen et al. [327]</td>
<td>Indoor localization</td>
<td>CSI</td>
<td>Device-based</td>
<td>CNN</td>
<td>Represents CSI as feature images</td>
</tr>
<tr>
<td>Guan et al. [337]</td>
<td>Indoor localization</td>
<td>Line-of-sight and non line-of-sight radio signals</td>
<td>Device-based</td>
<td>MLP</td>
<td>Combining deep learning with genetic algorithms; visible light communication based</td>
</tr>
</tbody>
</table>

**Table XIV: Leveraging Deep learning in user localization**

IEEE COMMUNICATIONS SURVEYS & TUTORIALS 31
filtering and obtains satisfactory prediction performance [317]. Wang et al. suggest that the objective of indoor localization can be achieved without the help of mobile devices. In [319], the authors employ an AE to learn useful patterns from WiFi signals. By automatic feature extraction, they produce a predictor that can fulfill multi-tasks simultaneously, including indoor localization, activity, and gesture recognition. A similar work in presented in [329], where Zhou et al. employ an MLP structure to perform device-free indoor localization using CSI. Kumar et al. use deep learning to address the problem of indoor vehicles localization [323]. They employ CNNs to analyze visual signal and localize vehicles in a car park. This can help driver assistance systems operate in underground environments where the system has limited vision ability.

In [335], Xiao et al. achieve low cost indoor localization with Bluetooth technology. The authors design a denosing AE to extract fingerprint features from the received signal strength of Bluetooth Low Energy beacon and subsequently project that to the exact position in 3D space. Experiments conducted in a conference room demonstrate that the proposed framework can perform precise positioning in both vertical and horizontal dimensions in real-time. Nitsso et al. employ a CNN to perform localization given raw channel impulse response data [333]. Their framework is robust to multipath propagation environments and more precise than signal processing based approaches. A CNN is also adopted in [332], where the authors work with received signal strength series and achieve 100% prediction accuracy in terms of building and floor identification. The work in [331] combines deep learning with linear discriminant analysis for feature reduction, achieving low positioning errors in multi-building environments. Zhang et al. combine pervasive magnetic field and WiFi fingerprinting for indoor localization using an MLP [330]. Experiments show that adding magnetic field information to the input of the model can improve the prediction accuracy, compared to solutions based solely on WiFi fingerprinting.

Hsu et al. use deep learning to provide Radio Frequency-based user localization, sleep monitoring, and insomnia analysis in multi-user home scenarios where individual sleep monitoring devices might not be available [336]. They use a CNN classifier with a 14-layer residual network model for sleep monitoring, in addition to Hidden Markov Models, to accurately track when the user enters or leaves the bed. By deploying sleep sensors called EZ-Sleep in 8 homes (see Fig. 15), collecting data for 100 nights of sleep over a month, and cross-validating this using an electroencephalography-based sleep monitor, the authors demonstrate the performance of their solution is comparable to that of individual, electroencephalography-based devices.

Most mobile devices can only produce unlabeled position data, therefore unsupervised and semi-supervised learning become essential. Mohammadi et al. address this problem by leveraging DRL and VAE. In particular, their framework envisions a virtual agent in indoor environments [320], which can constantly receive state information during training, including signal strength indicators, current agent location, and the real (labeled data) and inferred (via a VAE) distance to the target. The agent can virtually move in eight directions at each time step. Each time it takes an action, the agent receives a reward signal, identifying whether it moves to a correct direction. By employing deep Q learning, the agent can finally localize accurately a user, given both labeled and unlabeled data.

Beyond indoor localization, there also exist several research works that apply deep learning in outdoor scenarios. For example, Zheng and Weng introduce a lightweight developmental network for outdoor navigation applications on mobile devices [324]. Compared to CNNs, their architecture requires 100 times fewer weights to be updated, while maintaining decent accuracy. This enables efficient outdoor navigation on mobile devices. Work in [113] studies localization under both indoor and outdoor environments. They use an AE to pre-train a four-layer MLP, in order to avoid hand-crafted feature engineering. The MLP is subsequently used to estimate the coarse position of targets. The authors further introduce an HMM to fine-tune the predictions based on temporal properties of data. This improves the accuracy estimation in both in-/out-door positioning with Wi-Fi signals. More recently, Shokry et al. propose DeepLoc, a deep learning-based outdoor localization system using crowdsensed geo-tagged received signal strength information [328]. By using an MLP to learn the correlation between cellular signal and users’ locations, their framework...
can deliver median localization accuracy within 18.8m in urban areas and within 15.7m in rural areas on Android devices, while requiring modest energy budgets.

**Lessons learned:** Localization relies on sensorial output, signal strength, or CSI. These data usually have complex features, therefore large amounts of data are required for learning [317]. As deep learning can extract features in an unsupervised manner, it has become a strong candidate for localization tasks. On the other hand, it can be observed that positioning accuracy and system robustness can be improved by fusing multiple types of signals when providing these as the input (see, e.g., [330]). Using deep learning to automatically extract features and correlate information from different sources for localization purposes is becoming a trend.

**F. Deep Learning Driven Wireless Sensor Networks**

Wireless Sensor Networks (WSNs) consist of a set of unique or heterogeneous sensors that are distributed over geographical regions. Theses sensors collaboratively monitor physical or environment status (e.g. temperature, pressure, motion, pollution, etc.) and transmit the data collected to centralized servers through wireless channels (see top circle in Fig. 9 for an illustration). A WSN typically involves three key core tasks, namely sensing, communication and analysis. Deep learning is becoming increasingly popular also for WSN applications [349]. In what follows, we review works adopting deep learning in this domain, covering different angles, namely: centralized vs. decentralized analysis paradigms, WSN data analysis per se, WSN localization, and other applications. Note that the contributions of these works are distinct from mobile data analysis discussed in subsections VI-B and VI-C, as in this subsection we only focus on WSN applications. We begin by summarizing the most important works in Table XV.

**Centralized vs Decentralized Analysis Approaches:** There exist two data processing scenarios in WSNs, namely centralized and decentralized. The former simply takes sensors as data collectors, which are only responsible for gathering data and sending these to a central location for processing. The latter assumes sensors have some computational ability and the main server offloads part of the jobs to the edge, each sensor performing data processing individually. We show an example framework for WSN data collection and analysis in Fig. 16, where sensor data is collected via various nodes in a field of interest. Such data is delivered to a sink node, which aggregates and optionally further processes this. Work in [346] focuses on the centralized approach and the authors apply a 3-layer MLP to reduce data redundancy while maintaining essential points for data aggregation. These data are sent to a central server for analysis. In contrast, Li et al. propose to distribute data mining to individual sensors [347]. They partition a deep neural network into different layers and offload layer operations to sensor nodes. Simulations conducted suggest that, by pre-processing with NNs, their framework obtains high fault detection accuracy, while reducing power consumption at the central server.

**WSN Localization:** Localization is also an important and challenging task in WSNs. Chuang and Jiang exploit neural networks to localize sensor nodes in WSNs [338]. To adapt deep learning models to specific network topology, they employ an online training scheme and correlated topology-trained data, enabling efficient model implementations and accurate location estimation. Based on this, Bernas and Płaczek architect an ensemble system that involves multiple MLPs for location estimation in different regions of interest [339]. In this scenario, node locations inferred by multiple MLPs are fused by a fusion algorithm, which improves the localization accuracy, particularly benefiting sensor nodes that are around the boundaries of regions. A comprehensive comparison of different training algorithms that apply MLP-based node localization is presented in [340]. Experiments suggest that the Bayesian regularization algorithm in general yields the best performance. Dong et al. consider an underwater node localization scenario [341]. Since acoustic signals are subject to loss caused by absorption, scattering, noise, and interference, underwater localization is not straightforward. By adopting a deep neural network, their framework successfully addresses the aforementioned challenges and achieves higher inference accuracy as compared to SVM and generalized least square methods.

Phoemphon et al. [351] combine a fuzzy logic system and an ELM via a particle swarm optimization technique to achieve robust range-free location estimation for sensor nodes. In particular, the fuzzy logic system is employed for adjusting the weight of traditional centroids, while the ELM is used for optimization for the localization precision. Their method achieves superior accuracy over other soft computing-based approaches. Similarly, Banihashemian et al. employ the particle swarm optimization technique combining with MLPs to perform range-free WSN localization, which achieves low localization error [352]. Kang et al. shed light water leakage and localization in water distribution systems [354]. They represent the water pipeline network as a graph and assume leakage events occur at vertices. They combine CNN with SVM to perform detection and localization on wireless sensor network testbed, achieving 99.3% leakage detection accuracy and localization error for less than 3 meters.
TABLE XV: A summary of work on deep learning driven WSNs.

<table>
<thead>
<tr>
<th>Perspective</th>
<th>Reference</th>
<th>Application</th>
<th>Model</th>
<th>Optimizer</th>
<th>Key contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Centralized vs. Decentralized</strong></td>
<td>Khorasani and Naji [346]</td>
<td>Data aggregation</td>
<td>MLP</td>
<td>Unknown</td>
<td>Improves the energy efficiency in the aggregation process</td>
</tr>
<tr>
<td></td>
<td>Li et al. [347]</td>
<td>Distributed data mining</td>
<td>MLP</td>
<td>Resilient backpropagation</td>
<td>Performing data analysis at distributed nodes, reducing by 58.31% the energy consumption</td>
</tr>
<tr>
<td><strong>WSN Localization</strong></td>
<td>Bernas and Placzek [339]</td>
<td>Indoor localization</td>
<td>MLP</td>
<td>First-order and second-order gradient descent algorithm</td>
<td>Dramatically reduces the memory consumption of received signal strength map storage</td>
</tr>
<tr>
<td></td>
<td>Payal et al. [340]</td>
<td>Node localization</td>
<td>MLP</td>
<td>Conjugate gradient based method</td>
<td>Compares different training algorithms of MLP for WSN localization</td>
</tr>
<tr>
<td></td>
<td>Dong et al. [341]</td>
<td>Underwater localization</td>
<td>MLP</td>
<td>RMSprop</td>
<td>Performs WSN localization in underwater environments</td>
</tr>
<tr>
<td></td>
<td>Phoemphon et al. [351]</td>
<td>WSN node localization</td>
<td>Logic fuzzy system + ELM</td>
<td>Algorithm 4 described in [351]</td>
<td>Combining logic fuzzy system and ELM to achieve robust range-free WSN node localization</td>
</tr>
<tr>
<td></td>
<td>Banihashemian et al. [352]</td>
<td>Range-free WSN node localization</td>
<td>MLP, ELM</td>
<td>Conjugate gradient based method</td>
<td>Employs a particle swarm optimization algorithm to simultaneously optimize the neural network based on storage cost and localization accuracy</td>
</tr>
<tr>
<td></td>
<td>Kang et al. [354]</td>
<td>Water leakage detection and localization</td>
<td>CNN + SVM</td>
<td>SGD</td>
<td>Propose an enhanced graph-based local search algorithm using a virtual node scheme to select the nearest leakage location</td>
</tr>
<tr>
<td></td>
<td>El et al. [357]</td>
<td>WSN localization in the presence of anisotropic signal attenuation</td>
<td>MLP</td>
<td>Unknown</td>
<td>Robust against anisotropic signal attenuation</td>
</tr>
<tr>
<td><strong>WSN Data Analysis</strong></td>
<td>Yan et al. [342]</td>
<td>Smoldering and flaming combustion identification</td>
<td>MLP</td>
<td>SGD</td>
<td>Achieves high accuracy in detecting fire in forests using smoke, CO₂ and temperature sensors</td>
</tr>
<tr>
<td></td>
<td>Wang et al. [343]</td>
<td>Temperature correction</td>
<td>MLP</td>
<td>SGD</td>
<td>Employs deep learning to learn correlation between polar radiation and air temperature error</td>
</tr>
<tr>
<td></td>
<td>Lee et al. [344]</td>
<td>Online query processing</td>
<td>CNN</td>
<td>Unknown</td>
<td>Employs adaptive query refinement to enable real-time analysis</td>
</tr>
<tr>
<td></td>
<td>Li and Serpen [345]</td>
<td>Self-adaptive WSN</td>
<td>Hopfield network</td>
<td>Unknown</td>
<td>Embedding Hopfield NNs as a static optimizer for the weakly-connected dominating problem</td>
</tr>
<tr>
<td></td>
<td>Khorasani and Naji [346]</td>
<td>Data aggregation</td>
<td>MLP</td>
<td>Unknown</td>
<td>Improves the energy efficiency in the aggregation process</td>
</tr>
<tr>
<td></td>
<td>Li et al. [347]</td>
<td>Distributed data mining</td>
<td>MLP</td>
<td>SGD</td>
<td>Distributed data mining</td>
</tr>
<tr>
<td></td>
<td>Luo and Nagarajany [348]</td>
<td>Distributed WSN anomaly detection</td>
<td>AE</td>
<td>SGD</td>
<td>Employs distributed anomaly detection techniques to offload computations from the cloud</td>
</tr>
<tr>
<td><strong>Other</strong></td>
<td>Heydari et al. [350]</td>
<td>Energy consumption optimization and secure communication in wireless multimedia sensor networks</td>
<td>Stacked AE</td>
<td>Unknown</td>
<td>Use deep learning to enable fast data transfer and reduce energy consumption</td>
</tr>
<tr>
<td></td>
<td>Mehmoood et al. [355]</td>
<td>Robust routing for pollution monitoring in WSNs</td>
<td>MLP</td>
<td>SGD</td>
<td>Highly energy-efficient</td>
</tr>
<tr>
<td></td>
<td>Alsheikh et al. [356]</td>
<td>Rate-distortion balanced data compression for WSNs</td>
<td>AE</td>
<td>Limited memory Broyden Fletcher Goldfarb Shann algorithm</td>
<td>Energy-efficient and bounded reconstruction errors</td>
</tr>
<tr>
<td></td>
<td>Wang et al. [358]</td>
<td>Blind drift calibration for WSNs</td>
<td>Projection-recovery network (CNN based)</td>
<td>Adam</td>
<td>Exploits spatial and temporal correlations of data from all sensors; first work that adopts deep learning in WSN data calibration</td>
</tr>
<tr>
<td></td>
<td>Jia et al. [359]</td>
<td>Ammonia monitoring</td>
<td>LSTM</td>
<td>Adam</td>
<td>Low-power and accurate ammonia monitoring</td>
</tr>
</tbody>
</table>
WSN Data Analysis: Deep learning has also been exploited for identification of smoldering and flaming combustion phases in forests. In [342], Yan et al. embed a set of sensors into a forest to monitor CO₂, smoke, and temperature. They suggest that various burning scenarios will emit different gases, which can be taken into account when classifying smoldering and flaming combustion. Wang et al. consider deep learning to correct inaccurate measurements of air temperature [343]. They discover a close relationship between solar radiation and actual air temperature, which can be effectively learned by neural networks. In [353], Sun et al. employ a Wavelet neural network based solution to evaluate radio link quality in WSNs on smart grids. Their proposal is more precise than traditional approaches and can provide end-to-end reliability guarantees to smart grid applications.

Missing data or de-synchronization are common in WSN data collection. These may lead to serious problems in analysis due to inconsistency. Lee et al. address this problem by plugging a query refinement component in deep learning based WSN analysis systems [344]. They employ exponential smoothing to infer missing data, thereby maintaining the integrity of data for deep learning analysis without significantly compromising accuracy. To enhance the intelligence of WSNs, Li and Serpen embed an artificial neural network into a WSN, allowing it to agilely react to potential changes and following deployment in the field [345]. To this end, they employ a minimum weakly-connected dominating set to represent the WSN topology, and subsequently use a Hopfield recurrent neural network as a static optimizer, to adapt network infrastructure to potential changes as necessary. This work represents an important step towards embedding machine intelligence in WSNs.

Other Applications: The benefits of deep learning have also been demonstrated in other WSN applications. The work in [350] focuses on reducing energy consumption while maintaining security in wireless multimedia sensor networks. A stacked AE is employed to categorize images in the form of continuous pieces, and subsequently send the data over the network. This enables faster data transfer rates and lower energy consumption. Mehmood et al. employ MLPs to achieve robust routing in WSNs, so as to facilitate pollution monitoring [355]. Their proposal use the NN to provide an efficiency threshold, thereby improving energy efficiency. Alsheikh et al. introduce an algorithm for WSNs that uses AEs to minimize the energy expenditure [356]. Their architecture exploits spatio-temporal correlations to reduce the dimensions of raw data and provides reconstruction error bound guarantees.

Wang et al. design a dedicated projection-recovery neural network to blindly calibrate sensor measurements in an online manner [358]. Their proposal can automatically extract features from sensor data and exploit spatial and temporal correlations among information from all sensors, to achieve high accuracy. This is the first effort that adopts deep learning in WSN data calibration. Jia et al. shed light on ammonia monitoring using deep learning [359]. In their design, an LSTM is employed to predict the sensors’ electrical resistance during a very short heating pulse, without waiting for settling in an equilibrium state. This dramatically reduces the energy consumption of sensors in the waiting process. Experiments with 38 prototype sensors and a home-built gas flow system show that the proposed LSTM can deliver precise prediction of equilibrium state resistance under different ammonia concentrations, cutting down the overall energy consumption by approximately 99.6%.

Lessons learned: The centralized and decentralized WSN data analysis paradigms resemble the cloud and fog computing philosophies in other areas. Decentralized methods exploit the computing ability of sensor nodes and perform light processing and analysis locally. This offloads the burden on the cloud and significantly reduces the data transmission overheads and storage requirements. However, at the moment, the centralized approach dominates the WSN data analysis landscape. As deep learning implementation on embedded devices becomes more accessible, in the future we expect to witness a grow in the popularity of the decentralized schemes.

On the other hand, looking at Table XV, it is interesting to see that the majority of deep learning practices in WSNs employ MLP models. Since MLP is straightforward to architect and performs reasonably well, it remains a good candidate for WSN applications. However, since most sensor data collected is sequential, we expect RNN-based models will play a more important role in this area.

G. Deep Learning Driven Network Control

In this part, we turn our attention to mobile network control problems. Due to powerful function approximation mechanism, deep learning has made remarkable breakthroughs in improving traditional reinforcement learning [26] and imitation learning [493]. These advances have potential to solve mobile network control problems which are complex and previously considered intractable [494], [495]. Recall that in reinforcement learning, an agent continuously interacts with the environment to learn the best action. With constant exploration and exploitation, the agent learns to maximize its expected return. Imitation learning follows a different learning paradigm called “learning by demonstration”. This learning paradigm relies on a ‘teacher’ who tells the agent what action should be executed under certain observations during the training. After sufficient demonstrations, the agent learns a policy that imitates the behavior of the teacher and can operate standalone without supervision. For instance, an agent is trained to mimic human behaviour (e.g., in applications such as game play, self-driving vehicles, or robotics), instead of learning by interacting with the environment, as in the case of pure reinforcement learning. This is because in such applications, making mistakes can have fatal consequences [27].

Beyond these two approaches, analysis-based control is gaining traction in mobile networking. Specifically, this scheme uses ML models for network data analysis, and subsequently exploits the results to aid network control. Unlike reinforcement/imitation learning, analysis-based control does not directly output actions. Instead, it extract useful information and delivers this to an agent, to execute the
TABLE XVI: A summary of work on deep learning driven network control.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Reference</th>
<th>Application</th>
<th>Control approach</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network optimization</td>
<td>Liu et al. [360]</td>
<td>Demand constrained energy minimization</td>
<td>Analysis-based</td>
<td>DBN</td>
</tr>
<tr>
<td></td>
<td>Subramanian and Banerjee [361]</td>
<td>Machine to machine system optimization</td>
<td>Analysis-based</td>
<td>Deep multi-modal network</td>
</tr>
<tr>
<td></td>
<td>He et al. [362], [363]</td>
<td>Caching and interference alignment</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Masmar and Evans [364]</td>
<td>mmWave Communication performance optimization</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Wang et al. [365]</td>
<td>Handover optimization in wireless systems</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Chen and Smith [366]</td>
<td>Cellular network random access optimization</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Chen et al. [367]</td>
<td>Automatic traffic optimization</td>
<td>Reinforcement learning</td>
<td>Deep policy gradient</td>
</tr>
<tr>
<td>Routing</td>
<td>Lee et al. [368]</td>
<td>Virtual route assignment</td>
<td>Analysis-based</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Yang et al. [369]</td>
<td>Routing optimization</td>
<td>Analysis-based</td>
<td>Hopfield neural networks</td>
</tr>
<tr>
<td></td>
<td>Miao et al. [380]</td>
<td>Software defined routing</td>
<td>Imitation learning</td>
<td>DBN</td>
</tr>
<tr>
<td></td>
<td>Tang et al. [381]</td>
<td>Wireless network routing</td>
<td>Imitation learning</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Mao et al. [382]</td>
<td>Intelligent packet routing</td>
<td>Imitation learning</td>
<td>Tensor-based DBN</td>
</tr>
<tr>
<td></td>
<td>Geyer et al. [383]</td>
<td>Distributed routing</td>
<td>Imitation learning</td>
<td>Graph-query neural network</td>
</tr>
<tr>
<td></td>
<td>Pham et al. [405]</td>
<td>Routing for knowledge-defined networking</td>
<td>Reinforcement learning</td>
<td>Deep Deterministic Policy Gradient</td>
</tr>
<tr>
<td>Scheduling</td>
<td>Zhang et al. [370]</td>
<td>Hybrid dynamic voltage and frequency scaling scheduling</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Atallah et al. [371]</td>
<td>Roadside communication network scheduling</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Chuichila et al. [372]</td>
<td>Cellular network traffic scheduling</td>
<td>Reinforcement learning</td>
<td>Policy gradient</td>
</tr>
<tr>
<td></td>
<td>Atallah et al. [371]</td>
<td>Roadside communications networks scheduling</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Wei et al. [373]</td>
<td>User scheduling and content caching for mobile edge networks</td>
<td>Reinforcement learning</td>
<td>Deep policy gradient</td>
</tr>
<tr>
<td></td>
<td>Mennes et al. [395]</td>
<td>Predicting free slots in a Multiple Frequencies Time Division Multiple Access (MF-TDMA) network.</td>
<td>Imitation learning</td>
<td>MLP</td>
</tr>
<tr>
<td>Resource allocation</td>
<td>Sun et al. [374]</td>
<td>Resource management over wireless networks</td>
<td>Imitation learning</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Xu et al. [375]</td>
<td>Resource allocation in cloud radio access networks</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Ferreira et al. [376]</td>
<td>Resource management in cognitive communications</td>
<td>Reinforcement learning</td>
<td>Deep SARSA</td>
</tr>
<tr>
<td></td>
<td>Challita et al. [378]</td>
<td>Proactive resource management for LTE</td>
<td>Reinforcement learning</td>
<td>Deep policy gradient</td>
</tr>
<tr>
<td></td>
<td>Ye and Li [377]</td>
<td>Resource allocation in vehicle-to-vehicle communication</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Li et al. [394]</td>
<td>Computation offloading and resource allocation for mobile edge computing</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Zhou et al. [396]</td>
<td>Radio resource assignment</td>
<td>Analysis-based</td>
<td>LSTM</td>
</tr>
<tr>
<td>Radio control</td>
<td>Naparstek and Cohen [379]</td>
<td>Dynamic spectrum access</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>O’Shea and Clancy [380]</td>
<td>Radio control and signal detection</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Wijaya et al. [381], [383]</td>
<td>Inter-cell interference cancellation and transmit power optimization</td>
<td>Imitation learning</td>
<td>RBM</td>
</tr>
<tr>
<td></td>
<td>Rutageniwa et al. [382]</td>
<td>Dynamic spectrum alignment</td>
<td>Analysis-based</td>
<td>RNN</td>
</tr>
<tr>
<td></td>
<td>Yu et al. [390]</td>
<td>Multiple access for wireless network</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Li et al. [400]</td>
<td>Power control for spectrum sharing in cognitive radios</td>
<td>Reinforcement learning</td>
<td>DQN</td>
</tr>
<tr>
<td></td>
<td>Liu et al. [404]</td>
<td>Anti-jamming communications in dynamic and unknown environment</td>
<td>Reinforcement learning</td>
<td>DQN</td>
</tr>
<tr>
<td></td>
<td>Luong et al. [399]</td>
<td>Transaction transmission and channel selection in cognitive radio blockchain</td>
<td>Reinforcement learning</td>
<td>Double DQN</td>
</tr>
<tr>
<td></td>
<td>Ferreira et al. [406]</td>
<td>Radio transmitter settings selection in satellite communications</td>
<td>Reinforcement learning</td>
<td>Deep multi-objective reinforcement learning</td>
</tr>
<tr>
<td></td>
<td>Mao et al. [384]</td>
<td>Adaptive video bitrate</td>
<td>Reinforcement learning</td>
<td>A3C</td>
</tr>
<tr>
<td></td>
<td>Oda et al. [385], [386]</td>
<td>Mobile actor node control</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Kiti [281]</td>
<td>IoT load balancing</td>
<td>Analysis-based</td>
<td>DBN</td>
</tr>
<tr>
<td>Other</td>
<td>Challita et al. [388]</td>
<td>Path planning for aerial vehicle networking</td>
<td>Reinforcement learning</td>
<td>Multi-agent echo state networks</td>
</tr>
<tr>
<td></td>
<td>Luo et al. [389]</td>
<td>Wireless online power control</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Xu et al. [391]</td>
<td>Traffic engineering</td>
<td>Reinforcement learning</td>
<td>Deep policy gradient</td>
</tr>
<tr>
<td></td>
<td>Liu et al. [392]</td>
<td>Base station sleep control</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Zhao et al. [393]</td>
<td>Network slicing</td>
<td>Reinforcement learning</td>
<td>Deep Q learning</td>
</tr>
<tr>
<td></td>
<td>Zhu et al. [395]</td>
<td>Mobile edge caching</td>
<td>Reinforcement learning</td>
<td>A3C</td>
</tr>
<tr>
<td></td>
<td>Liu et al. [402]</td>
<td>Unmanned aerial vehicles control</td>
<td>Reinforcement learning</td>
<td>DQN</td>
</tr>
<tr>
<td></td>
<td>Lee et al. [401]</td>
<td>Transmit power Control in device-to-device communications</td>
<td>Imitation learning</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>He et al. [403]</td>
<td>Dynamic orchestration of networking, caching, and computing</td>
<td>Reinforcement learning</td>
<td>DQN</td>
</tr>
</tbody>
</table>
channels as finite-state Markov channels and apply deep Q networks to learn the best user selection policy. This novel framework demonstrates significantly higher sum rate and energy efficiency over existing approaches. Chen et al. shed light on automatic traffic optimization using a deep reinforcement learning approach [367]. Specifically, they architect a two-level DRL framework, which imitates the Peripheral and Central Nervous Systems in animals, to address scalability problems at datacenter scale. In their design, multiple peripheral systems are deployed on all end-hosts, so as to make decisions locally for short traffic flows. A central system is further employed to decide on the optimization with long traffic flows, which are more tolerant to longer delay. Experiments in a testbed with 32 servers suggest that the proposed design reduces the traffic optimization turn-around time and flow completion time significantly, compared to existing approaches.

**Routing:** Deep learning can also improve the efficiency of routing rules. Lee et al. exploit a 3-layer deep neural network to classify node degree, given detailed information of the routing nodes [368]. The classification results along with temporary routes are exploited for subsequent virtual route generation using the Viterbi algorithm. Mao et al. employ a DBN to decide the next routing node and construct a software defined router [188]. By considering Open Shortest Path First as the optimal routing strategy, their method achieves up to 95% accuracy, while reducing significantly the overhead and delay, and achieving higher throughput with a signaling interval of 240 milliseconds. In follow up work, the authors use tensors to represent hidden layers, weights and biases in DBNs, which further improves the routing performance [397].

A similar outcome is obtained in [296], where the authors employ Hopfield neural networks for routing, achieving better usability and survivability in mobile ad hoc network application scenarios. Geyer et al. represent the network using graphs, and design a dedicated Graph-Query NN to address the distributed routing problem [398]. This novel architecture takes graphs as input and uses message passing between nodes in the graph, allowing it to operate with various network topologies. Pham et al. shed light on routing protocols in knowledge-defined networking, using a Deep Deterministic Policy Gradient algorithm based on reinforcement learning [405]. Their agent takes traffic conditions as input and incorporates QoS into the reward function. Simulations show that their framework can effectively learn the correlations between traffic flows, which leads to better routing configurations.

**Scheduling:** There are several studies that investigate scheduling with deep learning. Zhang et al. introduce a deep Q learning-powered hybrid dynamic voltage and frequency scaling scheduling mechanism, to reduce the energy consumption in real-time systems (e.g. Wi-Fi, IoT, video applications) [370]. In their proposal, an AE is employed to approximate the Q function and the framework performs experience replay [497] to stabilize the training process and accelerate convergence. Simulations demonstrate that this method reduces by 4.2%
the energy consumption of a traditional Q learning based method. Similarly, the work in [371] uses deep Q learning for scheduling in roadside communications networks. In particular, interactions between vehicular environments, including the sequence of actions, observations, and reward signals are formulated as an MDP. By approximating the Q value function, the agent learns a scheduling policy that achieves lower latency and busy time, and longer battery life, compared to traditional scheduling methods.

More recently, Chinchali et al. present a policy gradient based scheduler to optimize the cellular network traffic flow [372]. Specifically, they cast the scheduling problem as a MDP and employ RF to predict network throughput, which is subsequently used as a component of a reward function. Evaluations with a realistic network simulator demonstrate that this proposal can dynamically adapt to traffic variations, which enables mobile networks to carry 14.7% more data traffic, while outperforming heuristic schedulers by more than 2x. Wei et al. address user scheduling and content caching simultaneously [373]. In particular, they train a DRL agent, consisting of an actor for deciding which base station should serve certain content, and whether to save the content. A critic is further employed to estimate the value function and deliver feedback to the actor. Simulations over a cluster of base stations show that the agent can yield low transmission delay. Li et al. shed light on resource allocation in a multi-user mobile computing scenario [394]. They employ a deep Q learning framework to jointly optimize the offloading decision and computational resource allocation, so as to minimize the sum cost of delay and energy consumption of all user equipment. Simulations show that their proposal can reduce the total cost of the system, as compared to fully-local, fully-offloading, and naive Q-learning approaches.

Resource Allocation: Sun et al. use a deep neural network to approximate the mapping between the input and output of the Weighted Minimum Mean Square Error resource allocation algorithm [498], in interference-limited wireless network environments [374]. By effective imitation learning, the neural network approximation achieves close performance to that of its teacher. Deep learning has also been applied to cloud radio access networks, Xu et al. employing deep Q learning to determine the on/off modes of remote radio heads given, the current mode and user demand [375]. Comparisons with single base station association and fully coordinated association methods suggest that the proposed DRL controller allows the system to satisfy user demand while requiring significantly less energy.

Ferreira et al. employ deep State-Action-Reward-State-Action (SARSA) to address resource allocation management in cognitive communications [376]. By forecasting the effects of radio parameters, this framework avoids wasted trials of poor parameters, which reduces the computational resources required. In [395], Mennes et al. employ MLPs to precisely forecast free slots prediction in a Multiple Frequencies Time Division Multiple Access (MF-TDMA) network, thereby achieving efficient scheduling. The authors conduct simulations with a network deployed in a 100×100 room, showing that their solution can effectively reduce collisions by half. Zhou et al. adopt LSTMs to predict traffic load at base stations in ultra dense networks [396]. Based on the predictions, their method changes the resource allocation policy to avoid congestion, which leads to lower packet loss rates, and higher throughput and mean opinion scores.

Radio Control: In [379], the authors address the dynamic spectrum access problem in multichannel wireless network environments using deep reinforcement learning. In this setting, they incorporate an LSTM into a deep Q network, to maintain and memorize historical observations, allowing the architecture to perform precise state estimation, given partial observations. The training process is distributed to each user, which enables effective training parallelization and the learning of good policies for individual users. Experiments demonstrate that this framework achieves double the channel throughput, when compared to a benchmark method. Yu et al. apply deep reinforcement learning to address challenges in wireless multiple access control [390], recognizing that in such tasks DRL agents are fast in terms of convergence and robust against non-optimal parameter settings. Li et al. investigate power control for spectrum sharing in cognitive radios using DRL. In their design, a DQN agent is built to adjust the transmit power of a cognitive radio system, such that the overall signal-to-interference-plus-noise ratio is maximized.

The work in [380] sheds light on the radio control and signal detection problems. In particular, the authors introduce a radio signal search environment based on the Gym Reinforcement Learning platform. Their agent exhibits a steady learning process and is able to learn a radio signal search policy. RAGAINwma et al. employ an RNN to perform traffic prediction, which can subsequently aid the dynamic spectrum assignment in mobile networks [382]. With accurate traffic forecasting, their proposal improves the performance of spectrum sharing in dynamic wireless environments, as it attains near-optimal spectrum assignments. In [404], Liu et al. approach the anti-jamming communications problem in dynamic and unknown environments with a DRL agent. Their system is based on a DQN with CNN, where the agent takes raw spectrum information as input and requires limited prior knowledge about the environment, in order to improve the overall throughput of the network in such adversarial circumstances.

Luong et al. incorporate the blockchain technique into cognitive radio networking [399], employing a double DQN agent to maximize the number of successful transaction transmissions for secondary users, while minimizing the channel cost and transaction fees. Simulations show that the DQN method significantly outperforms naive Q learning in terms of successful transactions, channel cost, and learning speed. DRL can further attack problems in the satellite communications domain. In [496], Ferreira et al. fuse multi-objective reinforcement learning [406] with deep neural networks to select among multiple radio transmitter settings while attempting to achieve multiple conflicting goals, in a dynamically changing satellite communications channel. Specifically, two set of NNSs are employed to execute exploration and exploitation separately. This builds an ensembling system, with makes the framework more
robust to the changing environment. Simulations demonstrate that their system can nearly optimize six different objectives (i.e., bit error rate, throughput, bandwidth, spectral efficiency, additional power consumption, and power efficiency), only with small performance errors compared to ideal solutions.

Other applications: Deep learning is playing an important role in other network control problems as well. Mao et al. develop the Pensieve system that generates adaptive video bit rate algorithms using deep reinforcement learning [384]. Specifically, Pensieve employs a state-of-the-art deep reinforcement learning algorithm A3C, which takes the bandwidth, bit rate and buffer size as input, and selects the bit rate that leads to the best expected return. The model is trained offline and deployed on an adaptive bit rate server, demonstrating that the system outperforms the best existing scheme by 12%-25% in terms of QoE. Liu et al. apply deep Q learning to reduce the energy consumption in cellular networks [392]. They train an agent to dynamically switch on/off base stations based on traffic consumption in areas of interest. An action-wise experience replay mechanism is further designed for balancing different traffic behaviours. Experiments show that their proposal can significantly reduce the energy consumed by base stations, outperforming naive table-based Q learning approaches. A control mechanism for unmanned aerial vehicles using DQN is proposed in [402], where multiple objectives are targeted: maximizing energy efficiency, communications coverage, fairness and connectivity. The authors conduct extensive simulations in a virtual playground, showing that their agent is able to learn the dynamics of the environment, achieving superior performance over random and greedy control baselines.

Kim and Kim link deep learning with the load balancing problem in IoT [387]. The authors suggest that DBNs can effectively analyze network load and process structural configuration, thereby achieving efficient load balancing in IoT. Challita et al. employ a deep reinforcement learning algorithm based on echo state networks to perform path planning for a cluster of unmanned aerial vehicles [388]. Their proposal yields lower delay than a heuristic baseline. Xu et al. employ a DRL agent to learn from network dynamics how to control traffic flow [391]. They advocate that DRL is suitable for this problem, as it performs remarkably well in handling dynamic environments and sophisticated state spaces. Simulations conducted over three network topologies confirm this viewpoint, as the DRL agent significantly reduces the delay, while providing throughput comparable to that of traditional approaches. Zhu et al. employ the A3C algorithm to address the caching problem in mobile edge computing. Their method obtains superior cache hit ratios and traffic offloading performance over three baselines caching methods. Several open challenges are also pointed out, which are worthy of future pursuit. The edge caching problem is also addressed in [403], where He et al. architect a DQN agent to perform dynamic orchestration of networking, caching, and computing. Their method facilitates high revenue to mobile virtual network operators.

Lessons learned: There exist three approaches to network control using deep learning i.e., reinforcement learning, imitation learning, and analysis-based control. Reinforcement learning requires to interact with the environment, trying different actions and obtaining feedback in order to improve. The agent will make mistakes during training, and usually needs a large number of steps of steps to become smart. Therefore, most works do not train the agent on the real infrastructure, as making mistakes usually can have serious consequences for the network. Instead, a simulator that mimics the real network environments is built and the agent is trained offline using that. This imposes high fidelity requirements on the simulator, as the agent can not work appropriately in an environment that is different from the one used for training. On the other hand, although DRL performs remarkable well in many applications, considerable amount of time and computing resources are required to train an usable agent. This should be considered in real-life implementation.

In contrast, the imitation learning mechanism “learns by demonstration”. It requires a teacher that provides labels telling the agent what it should do under certain circumstances. In the networking context, this mechanism is usually employed to reduce the computational time [188]. Specifically, in some network application (e.g., routing), computing the optimal solution is time-consuming, which cannot satisfy the delay constraints of mobile network. To mitigate this, one can generate a large dataset offline, and use an NN agent to learn the optimal actions.

Analysis-based control on the other hand, is suitable for problems were decisions cannot be based solely on the state of the network environment. One can use an NN to extract additional information (e.g. traffic forecasts), which subsequently aids decisions. For example, the dynamic spectrum assignment can benefit from the analysis-based control.

H. Deep Learning Driven Network Security

With the increasing popularity of wireless connectivity, protecting users, network equipment and data from malicious attacks, unauthorized access and information leakage becomes crucial. Cyber security systems guard mobile devices and users through firewalls, anti-virus software, and Intrusion Detection Systems (IDS) [499]. The firewall is an access security gateway that allows or blocks the uplink and downlink network traffic, based on pre-defined rules. Anti-virus software detects and removes computer viruses, worms and Trojans and malware. IDSs identify unauthorized and malicious activities, or rule violations in information systems. Each performs its own functions to protect network communication, central servers and edge devices.

Modern cyber security systems benefit increasingly from deep learning [501], since it can enable the system to (i) automatically learn signatures and patterns from experience and generalize to future intrusions (supervised learning); or (ii) identify patterns that are clearly differed from regular behavior (unsupervised learning). This dramatically reduces the effort of pre-defined rules for discriminating intrusions. Beyond protecting networks from attacks, deep learning can also be used for attack purposes, bringing huge potential
TABLE XVII: A summary of work on deep learning driven network security.

<table>
<thead>
<tr>
<th>Level</th>
<th>Reference</th>
<th>Application</th>
<th>Problem considered</th>
<th>Learning paradigm</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>Azar et al. [407]</td>
<td>Cyber security applications</td>
<td>Malware classification &amp; Denial of service, probing, remote to user &amp; user to root</td>
<td>Unsupervised &amp; supervised</td>
<td>Stacked AE</td>
</tr>
<tr>
<td></td>
<td>Thing [187]</td>
<td>IEEE 802.11 network anomaly detection and attack classification</td>
<td>Flooding, injection and impersonation attacks</td>
<td>Unsupervised &amp; supervised</td>
<td>Stacked AE</td>
</tr>
<tr>
<td></td>
<td>Aminanto and Kim [408]</td>
<td>Wi-Fi impersonation attacks detection</td>
<td>Flooding, injection and impersonation attacks</td>
<td>Unsupervised &amp; supervised</td>
<td>MLP, AE</td>
</tr>
<tr>
<td></td>
<td>Feng et al. [409]</td>
<td>Spectrum anomaly detection</td>
<td>Sudden signal-to-noise ratio changes in the communication channels</td>
<td>Unsupervised &amp; supervised</td>
<td>AE</td>
</tr>
<tr>
<td></td>
<td>Khan et al. [410]</td>
<td>Flooding attacks detection in wireless mesh networks</td>
<td>Moderate and severe distributed flood attack</td>
<td>Supervised</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Duro and Chilamkurthi [411]</td>
<td>IoT distributed attacks detection</td>
<td>Denial of service, probing, remote to user &amp; user to root</td>
<td>Supervised</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Saeid et al. [412]</td>
<td>Distributed denial of service attack detection</td>
<td>Known and unknown distributed denial of service attack</td>
<td>Supervised</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Martin et al. [413]</td>
<td>IoT intrusion detection</td>
<td>Denial of service, probing, remote to user &amp; user to root</td>
<td>Unsupervised &amp; supervised</td>
<td>Conditional VAE</td>
</tr>
<tr>
<td></td>
<td>Hamedani et al. [414]</td>
<td>Attacks detection in delayed feedback networks</td>
<td>Attack detection in smart grids using reservoir computing</td>
<td>Supervised</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Luo and Nagarajany [348]</td>
<td>Anomalies in WSNs</td>
<td>Spikes and burst recorded by temperature and relative humidity sensors</td>
<td>Unsupervised</td>
<td>AE</td>
</tr>
<tr>
<td></td>
<td>Das et al. [415]</td>
<td>IoT authentication</td>
<td>Long duration of signal imperfections</td>
<td>Supervised</td>
<td>LSTM</td>
</tr>
<tr>
<td></td>
<td>Jiang et al. [416]</td>
<td>MAC spoofing detection</td>
<td>Packets from different hardware use same MAC address</td>
<td>Supervised</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Jiang et al. [422]</td>
<td>Cyberattack detection in mobile cloud computing</td>
<td>Various cyberattacks from 3 different datasets</td>
<td>Unsupervised &amp; supervised</td>
<td>RBM</td>
</tr>
<tr>
<td></td>
<td>Yuan et al. [417]</td>
<td>Android malware detection</td>
<td>Apps in Contagio Mobile and Google Play Store</td>
<td>Unsupervised &amp; supervised</td>
<td>RBM</td>
</tr>
<tr>
<td></td>
<td>Yuan et al. [418]</td>
<td>Android malware detection</td>
<td>Apps in Contagio Mobile and Android Malware Genome Project</td>
<td>Unsupervised &amp; supervised</td>
<td>DBN</td>
</tr>
<tr>
<td></td>
<td>Sa et al. [419]</td>
<td>Android malware detection</td>
<td>Apps in Drebin, Android Malware Genome Project, the Contagio Community, and Google Play Store</td>
<td>Unsupervised &amp; supervised</td>
<td>DBN + SVM</td>
</tr>
<tr>
<td></td>
<td>Hou et al. [420]</td>
<td>Android malware detection</td>
<td>App samples from Comodo Cloud Security Center</td>
<td>Unsupervised &amp; supervised</td>
<td>Stacked AE</td>
</tr>
<tr>
<td></td>
<td>Martinelli [421]</td>
<td>Android malware detection</td>
<td>Apps in Drebin, Android Malware Genome Project and Google Play Store</td>
<td>Supervised</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>McLaughlin et al. [423]</td>
<td>Android malware detection</td>
<td>Apps in Android Malware Genome project and Google Play Store</td>
<td>Supervised</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Chen et al. [424]</td>
<td>Malicious application detection at the network edge</td>
<td>Publicly-available malicious applications</td>
<td>Unsupervised &amp; supervised</td>
<td>RBM</td>
</tr>
<tr>
<td></td>
<td>Wang et al. [428]</td>
<td>Malware traffic classification</td>
<td>Traffic extracted from 9 types of malware</td>
<td>Supervised</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Oulehla et al. [425]</td>
<td>Mobile botnet detection</td>
<td>Client-server and hybrid botnets</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td></td>
<td>Torres et al. [426]</td>
<td>Botnet detection</td>
<td>Spam, HTTP and unknown traffic</td>
<td>Supervised</td>
<td>LSTM</td>
</tr>
<tr>
<td></td>
<td>Estahi et al. [427]</td>
<td>Mobile botnet detection</td>
<td>HTTP botnet traffic</td>
<td>Supervised</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Alauthaman et al. [428]</td>
<td>Peer-to-peer botnet detection</td>
<td>Waledac and Strom Bots</td>
<td>Supervised</td>
<td>MLP</td>
</tr>
<tr>
<td>User privacy</td>
<td>Shokri and Shmatikov [429]</td>
<td>Privacy preserving deep learning</td>
<td>Avoiding sharing data in collaborative model training</td>
<td>Supervised</td>
<td>MLP, CNN</td>
</tr>
<tr>
<td></td>
<td>Phong et al. [430]</td>
<td>Privacy preserving deep learning</td>
<td>Addressing information leakage introduced in [429]</td>
<td>Supervised</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Ossia et al. [431]</td>
<td>Privacy-preserving mobile analytics</td>
<td>Offloading feature extraction from cloud</td>
<td>Supervised</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Abadi et al. [432]</td>
<td>Deep learning with differential privacy</td>
<td>Preventing exposure of private information in training data</td>
<td>Supervised</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Osia et al. [433]</td>
<td>Privacy-preserving personal model training</td>
<td>Offloading personal data from clouds</td>
<td>Unsupervised &amp; supervised</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Servia et al. [434]</td>
<td>Privacy-preserving model inference</td>
<td>Breaking down large models for privacy-preserving analytics</td>
<td>Supervised</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Hitaj et al. [435]</td>
<td>Stealing information from collaborative deep learning</td>
<td>Breaking the ordinary and differentially private collaborative deep learning</td>
<td>Unsupervised</td>
<td>GAN</td>
</tr>
<tr>
<td></td>
<td>Hitaj et al. [432]</td>
<td>Password guessing</td>
<td>Generating passwords from leaked password set</td>
<td>Unsupervised</td>
<td>GAN</td>
</tr>
<tr>
<td></td>
<td>Greydanus [436]</td>
<td>Enigma learning</td>
<td>Reconstructing functions of polyalphabetic cipher</td>
<td>Supervised</td>
<td>LSTM</td>
</tr>
<tr>
<td></td>
<td>Maghrebi et al. [437]</td>
<td>Breaking cryptographic</td>
<td>Side channel attacks</td>
<td>Supervised</td>
<td>MLP, AE, CNN, LSTM</td>
</tr>
<tr>
<td></td>
<td>Liu et al. [438]</td>
<td>Password guessing</td>
<td>Employing adversarial generation to guess passwords</td>
<td>Unsupervised</td>
<td>LSTM</td>
</tr>
<tr>
<td></td>
<td>Ning et al. [439]</td>
<td>Mobile apps sniffing</td>
<td>Defending against mobile apps sniffing through noise injection</td>
<td>Supervised</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Wang et al. [500]</td>
<td>Private inference in mobile cloud</td>
<td>Computation offloading and privacy preserving for mobile inference</td>
<td>Supervised</td>
<td>MLP, CNN</td>
</tr>
</tbody>
</table>
to steal or crack user passwords or information. In this subsection, we review deep learning driven network security from three perspectives, namely infrastructure, software, and user privacy. Specifically, infrastructure level security work focuses on detecting anomalies that occur in the physical network and software level work is centred on identifying malware and botnets in mobile networks. From the user privacy perspective, we discuss methods to protect from how to protect against private information leakage, using deep learning. To our knowledge, no other reviews summarize these efforts. We summarize these works in Table XVII.

**Infrastructure level security:** We mostly focus on anomaly detection at the infrastructure level, i.e. identifying network events (e.g., attacks, unexpected access and use of data) that do not conform to expected behaviors. Many researchers exploit the outstanding unsupervised learning ability of AEs [407]. For example, Thing investigates features of attacks and threats that exist in IEEE 802.11 networks [187]. The author employs a stacked AE to categorize network traffic into 5 types (i.e. legitimate, flooding, injection and impersonation traffic), achieving 98.67% overall accuracy. The AE is also exploited in [408], where Aminanto and Kim use an MLP and stacked AE for feature selection and extraction, demonstrating remarkable performance. Similarly, Feng et al. use AEs to detect abnormal spectrum usage in wireless communications [409]. Their experiments suggest that the detection accuracy can significantly benefit from the depth of AEs.

Distributed attack detection is also an important issue in mobile network security. Khan et al. focus on detecting flooding attacks in wireless mesh networks [410]. They simulate a wireless environment with 100 nodes, and artificially inject moderate and severe distributed flooding attacks, to generate a synthetic dataset. Their deep learning based methods achieve excellent false positive and false negative rates. Distributed attacks are also studied in [411], where the authors focus on an IoT scenario. Another work in [412] employs MLPs to detect distributed denial of service attacks. By characterizing typical patterns of attack incidents, the proposed model works well in detecting both known and unknown distributed denial of service attacks. More recently, Nguyen et al. employ RBMs to classify cyberattacks in the mobile cloud in an online manner [422]. Through unsupervised layer-wise pre-training and fine-tuning, their methods obtain over 90% classification accuracy on three different datasets, significantly outperforming other machine learning approaches.

Martin et al. propose a conditional VAE to identify intrusion incidents in IoT [413]. In order to improve detection performance, their VAE infers missing features associated with incomplete measurements, which are common in IoT environments. The true data labels are embedded into the decoder layers to assist final classification. Evaluations on the well-known NSL-KDD dataset [502] demonstrate that their model achieves remarkable accuracy in identifying denial of service, probing, remote to user and user to root attacks, outperforming traditional ML methods by 0.18 in terms of F1 score. Hamedani et al. employ MLPs to detect malicious attacks in delayed feedback networks [414]. The proposal achieves more than 99% accuracy over 10,000 simulations.

**Software level security:** Nowadays, mobile devices are carrying considerable amount of private information. This information can be stolen and exploited by malicious apps installed on smartphones for ill-conceived purposes [503]. Deep learning is being exploited for analyzing and detecting such threats.

Yuan et al. use both labeled and unlabeled mobile apps to train an RBM [417]. By learning from 300 samples, their model can classify Android malware with remarkable accuracy, outperforming traditional ML tools by up to 19%. Their follow-up research in [418] named Droiddetector further improves the detection accuracy by 2%. Similarly, Su et al. analyze essential features of Android apps, namely requested permission, used permission, sensitive application programming interface calls, action and app components [419]. They employ DBNs to extract features of malware and an SVM for classification, achieving high accuracy and only requiring 6 seconds per inference instance.

Hou et al. attack the malware detection problem from a different perspective. Their research points out that signature-based detection is insufficient to deal with sophisticated Android malware [420]. To address this problem, they propose the Component Traversal, which can automatically execute code routines to construct weighted directed graphs. By employing a Stacked AE for graph analysis, their framework Deep4MalDroid can accurately detect Android malware that intentionally repackages and obfuscates to bypass signatures and hinder analysis attempts to their inner operations. This work is followed by that of Martinelli et al., who exploit CNNs to discover the relationship between app types and extracted syscall traces from real mobile devices [421]. The CNN has also been used in [423], where the authors draw inspiration from NLP and take the disassembled byte-code of an app as a text for analysis. Their experiments demonstrate that CNNs can effectively learn to detect sequences of opcodes that are indicative of malware. Chen et al. incorporate location information into the detection framework and exploit an RBM for feature extraction and classification [424]. Their proposal improves the performance of other ML methods.

Botnets are another important threat to mobile networks. A botnet is effectively a network that consists of machines compromised by bots. These machine are usually under the control of a botmaster who takes advantages of the bots to harm public services and systems [504]. Detecting botnets is challenging and now becoming a pressing task in cyber security. Deep learning is playing an important role in this area. For example, Oulehla et al. propose to employ neural networks to extract features from mobile botnet behaviors [425]. They design a parallel detection framework for identifying both client-server and hybrid botnets, and demonstrate encouraging performance. Torres et al. investigate the common behavior patterns that botnets exhibit across their life cycle, using LSTMs [426]. They employ both under-sampling and over-sampling to address the class imbalance between botnet and normal traffic in the dataset, which is common in anomaly detection problems. Similar issues are also studies in [427] and [428], where
the authors use standard MLPs to perform mobile and peer-to-peer botnet detection respectively, achieving high overall accuracy.

**User privacy level:** Preserving user privacy during training and evaluating a deep neural network is another important research issue [505]. Initial research is conducted in [429], where the authors enable user participation in the training and evaluation of a neural network, without sharing their input data. This allows to preserve individual’s privacy while benefiting all users, as they collaboratively improve the model performance. Their framework is revisited and improved in [430], where another group of researchers employ additively homomorphic encryption, to address the information leakage problem ignored in [429], without compromising model accuracy. This significantly boosts the security of the system. More recently, Wang et al. [500] propose a framework called ARDEN to preserve users’ privacy while reducing communication overhead in mobile-cloud deep learning applications. ARDEN partitions a NN across cloud and mobile devices, with heavy computation being conducted on the cloud and mobile devices performing only simple data transformation and perturbation, using a differentially private mechanism. This simultaneously guarantees user privacy, improves inference accuracy, and reduces resource consumption.

Osi et al. focus on privacy-preserving mobile analytics using deep learning. They design a client-server framework based on the Siamese architecture [506], which accommodates a feature extractor in mobile devices and correspondingly a classifier in the cloud [431]. By offloading feature extraction from the cloud, their system offers strong privacy guarantees. An innovative work in [432] implies that deep neural networks can be trained with differential privacy. The authors introduce a differentially private SGD to avoid disclosure of private information of training data. Experiments on two publicly-available image recognition datasets demonstrate that their algorithm is able to maintain users privacy, with a manageable cost in terms of complexity, efficiency, and performance. This approach is also useful for edge-based privacy filtering techniques such as Distributed One-class Learning [507].

Servia et al. consider training deep neural networks on distributed devices without violating privacy constraints [434]. Specifically, the authors retrain an initial model locally, tailored to individual users. This avoids transferring personal data to untrusted entities, hence user privacy is guaranteed. Osi et al. focus on protecting user’s personal data from the inferences’ perspective. In particular, they break the entire deep neural network into a feature extractor (on the client side) and an analyzer (on the cloud side) to minimize the exposure of sensitive information. Through local processing of raw input data, sensitive personal information is transferred into abstract features, which avoids direct disclosure to the cloud. Experiments on gender classification and emotion detection suggest that this framework can effectively preserve user privacy, while maintaining remarkable inference accuracy.

Deep learning has also been exploited for cyber attacks, including attempts to compromise private user information and guess passwords. In [435], Hitaj et al. suggest that learning a deep model collaboratively is not reliable. By training a GAN, their attacker is able to affect such learning process and lure the victims to disclose private information, by injecting fake training samples. Their GAN even successfully breaks the differentially private collaborative learning in [432]. The authors further investigate the use of GANs for password guessing. In [508], they design PassGAN, which learns the distribution of a set of leaked passwords. Once trained on a dataset, PassGAN is able to match over 46% of passwords in a different testing set, without user intervention or cryptography knowledge. This novel technique has potential to revolutionize current password guessing algorithms.

Greydanus breaks a decryption rule using an LSTM network [436]. They treat decryption as a sequence-to-sequence translation task, and train a framework with large enigma pairs. The proposed LSTM demonstrates remarkable performance in learning polyalphabetic ciphers. Maghrebi et al. exploit various deep learning models (i.e. MLP, AE, CNN, LSTM) to construct a precise profiling system and perform side channel key recovery attacks [437]. Surprisingly, deep learning based methods demonstrate overwhelming performance over other template machine learning attacks in terms of efficiency in breaking both unprotected and protected Advanced Encryption Standard implementations. In [439], Ning et al. demonstrate that an attacker can use a CNN to infer with over 84% accuracy what apps run on a smartphone and their usage, based on magnetometer or orientation data. The accuracy can increase to 98% if motion sensors information is also taken into account, which jeopardizes user privacy. To mitigate this issue, the authors propose to inject Gaussian noise into the magnetometer and orientation data, which leads to a reduction in inference accuracy down to 15%, thereby effectively mitigating the risk of privacy leakage.

**Lessons learned:** Most deep learning based solutions focus on existing network attacks, yet new attacks emerge every day. As these new attacks may have different features and appear to behave ‘naturally’, old NN models may not easily detect them. Therefore, an effective deep learning technique should be able to (i) rapidly transfer the knowledge of old attacks to detect newer ones; and (ii) constantly absorb the features of newcomers and update the underlying model. Transfer learning and lifelong learning are strong candidates to address this problems, as we will discuss in Sec.VII-C. Research in this directions remains shallow, hence we expect more efforts in the future.

Another issue to which attention should be paid is the fact that NNs are vulnerable to adversarial attacks. This has been briefly discussed in Sec. III-E. Although formal reports on this matter are lacking, hackers may exploit weaknesses in NN models and training procedures to perform attacks that subvert deep learning based cyber-defense systems. This is an important potential pitfall that should be considered in real implementations.

**I. Deep Learning Driven Signal Processing**

Deep learning is also gaining increasing attention in signal processing, in applications including Multi-Input Multi-Output...
(MIMO) and modulation. MIMO has become a fundamental technique in current wireless communications, both in cellular and WiFi networks. By incorporating deep learning, MIMO performance is intelligently optimized based on environment conditions. Modulation recognition is also evolving to be more accurate, by taking advantage of deep learning. We give an overview of relevant work in this area in Table XVIII.

**MIMO Systems**: Samuel et al. suggest that deep neural networks can be a good estimator of transmitted vectors in a MIMO channel. By unfolding a projected gradient descent method, they design an MLP-based detection network to perform binary MIMO detection [449]. The Detection Network can be implemented on multiple channels after a single training. Simulations demonstrate that the proposed architecture achieves near-optimal accuracy, while requiring light computation without prior knowledge of Signal-to-Noise Ratio (SNR). Yan et al. employ deep learning to solve a similar problem from a different perspective [450]. By considering the characteristic invariance of signals, they exploit an AE as a feature extractor, and subsequently use an Extreme Learning Machine (ELM) to classify signal sources in a MIMO orthogonal frequency division multiplexing (OFDM) system. Their proposal achieves higher detection accuracy than several traditional methods, while maintaining similar complexity.

Veira et al. show that massive MIMO channel measurements in cellular networks can be utilized for fingerprint-based inference of user positions [325]. Specifically, they design CNNs with weight regularization to exploit the sparse and information-invariance of channel fingerprints, thereby achieving precise positions inference. CNNs have also been employed for MIMO channel estimation. Neumann et al. exploit the structure of the MIMO channel model to design a lightweight, approximated maximum likelihood estimator for a specific channel model [448]. Their methods outperform traditional estimators in terms of computation cost and reduce the number of hyper-parameters to be tuned. A similar idea is implemented in [455], where Ye et al. employ an MLP to perform channel estimation and signal detection in OFDM systems.

Wijaya et al. consider applying deep learning to a different scenario [381], [383]. The authors propose to use non-iterative neural networks to perform transmit power control at base stations, thereby preventing degradation of network performance due to inter-cell interference. The neural network is trained to estimate the optimal transmit power at every packet transmission, selecting that with the highest activation probability. Simulations demonstrate that the proposed framework significantly outperforms the belief propagation algorithm that is routinely used for transmit power control in MIMO systems, while attaining a lower computational cost.

More recently, O’Shea et al. bring deep learning to physical layer design [440]. They incorporate an unsupervised deep AE into a single-user end-to-end MIMO system, to optimize representations and the encoding/decoding processes, for transmissions over a Rayleigh fading channel. We illustrate the adopted AE-based framework in Fig 18. This design incorporates a transmitter consisting of an MLP followed by a normalization layer, which ensures that physical constraints on the signal are guaranteed. After transfer through an additive white Gaussian noise channel, a receiver employs another MLP to decode messages and select the one with the highest probability of occurrence. The system can be trained with an SGD algorithm in an end-to-end manner. Experimental

<table>
<thead>
<tr>
<th>Domain</th>
<th>Reference</th>
<th>Application</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIMO systems</td>
<td>Samuel et al. [449]</td>
<td>MIMO detection</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Yan et al. [450]</td>
<td>Signal detection in a MIMO-OFDM system</td>
<td>AE + ELM</td>
</tr>
<tr>
<td></td>
<td>Veira et al. [325]</td>
<td>Massive MIMO fingerprint-based positioning</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Neumann et al. [448]</td>
<td>MIMO channel estimation</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Wijaya et al. [381], [383]</td>
<td>Inter-cell interference cancellation and transmit power optimization</td>
<td>RBM</td>
</tr>
<tr>
<td></td>
<td>O’Shea et al. [440]</td>
<td>Optimization of representations and encoding/decoding processes</td>
<td>AE</td>
</tr>
<tr>
<td></td>
<td>Borgarding et al. [441]</td>
<td>Sparse linear inverse problem in MIMO</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Fujishita et al. [442]</td>
<td>MIMO nonlinear equalization</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Huang et al. [460]</td>
<td>Super-resolution channel and direction-of-arrival estimation</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Rajendran et al. [443]</td>
<td>Automatic modulation classification</td>
<td>LSTM</td>
</tr>
<tr>
<td></td>
<td>O’Shea et al. [445]</td>
<td>Modulation recognition</td>
<td>Radio transformer network</td>
</tr>
<tr>
<td></td>
<td>O’Shea and Hoydis [451]</td>
<td>Modulation classification</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Jagannath et al. [452]</td>
<td>Modulation classification in a software defined radio testbed</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>O’Shea et al. [453]</td>
<td>Radio traffic sequence recognition</td>
<td>LSTM</td>
</tr>
<tr>
<td></td>
<td>O’Shea et al. [454]</td>
<td>Learning to communicate over an impaired channel</td>
<td>AE + radio transformer network</td>
</tr>
<tr>
<td></td>
<td>Ye et al. [455]</td>
<td>Channel estimation and signal detection in OFDM systems.</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Liang et al. [456]</td>
<td>Channel decoding</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Lyu et al. [457]</td>
<td>NNS for channel decoding</td>
<td>MLP, CNN, and RNN</td>
</tr>
<tr>
<td></td>
<td>Dörner et al. [458]</td>
<td>Over-the-air communications system</td>
<td>AE</td>
</tr>
<tr>
<td></td>
<td>Liao et al. [459]</td>
<td>Rayleigh fading channel prediction</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Huang et al. [461]</td>
<td>Light-emitting diode (LED) visible light downlink error correction</td>
<td>AE</td>
</tr>
<tr>
<td></td>
<td>Alkhateeb et al. [447]</td>
<td>Coordinated beamforming for highly-mobile millimeter wave systems</td>
<td>MLP</td>
</tr>
<tr>
<td></td>
<td>Gante et al. [446]</td>
<td>Millimeter wave positioning</td>
<td>CNN</td>
</tr>
<tr>
<td></td>
<td>Ye et al. [450]</td>
<td>Channel agnostic end-to-end learning based communication system</td>
<td>Conditional GAN</td>
</tr>
</tbody>
</table>
results show that the AE system outperforms the Space Time Block Code approach in terms of SNR by approximately 15 dB. In [441], Borgerding et al. propose to use deep learning to recover a sparse signal from noisy linear measurements in MIMO environments. The proposed scheme is evaluated on compressive random access and massive-MIMO channel estimation, where it achieves better accuracy over traditional algorithms and CNNs.

**Modulation:** West and O’Shea compare the modulation recognition accuracy of different deep learning architectures, including traditional CNN, ResNet, Inception CNN, and LSTM [444]. Their experiments suggest that the LSTM is the best candidate for modulation recognition, since it achieves the highest accuracy. Due to its superior performance, an LSTM is also employed for a similar task in [443]. O’Shea et al. then focus on tailoring deep learning architectures to radio properties. Their prior work is improved in [445], where they architect a novel deep radio transformer network for precise modulation recognition. Specifically, they introduce radio-domain specific parametric transformations into a spatial transformer network, which assists in the normalization of the received signal, thereby achieving superior performance. This framework also demonstrates automatic synchronization abilities, which reduces the dependency on traditional expert systems and expensive signal analytic processes. In [451], O’Shea and Hoydis introduce several novel deep learning applications for the network physical layer. They demonstrate a proof-of-concept where they employ a CNN for modulation classification and obtain satisfying accuracy.

**Other signal processing applications:** Deep learning is also adopted for radio signal analysis. In [453], O’Shea et al. employ an LSTM to replace sequence translation routines between radio transmitter and receiver. Although their framework works well in ideal environments, its performance drops significantly when introducing realistic channel effects. Later, the authors consider a different scenario in [454], where they exploit a regularized AE to enable reliable communications over an impaired channel. They further incorporate a radio transformer network for signal reconstruction at the decoder side, thereby achieving receiver synchronization. Simulations demonstrate that this approach is reliable and can be efficiently implemented.

In [456], Liang et al. exploit noise correlations to decode channels using a deep learning approach. Specifically, they use a CNN to reduce channel noise estimation errors by learning the noise correlation. Experiments suggest that their framework can significantly improve the decoding performance. The decoding performance of MLPs, CNNs and RNNs is compared in [457]. By conducting experiments in different setting, the obtained results suggest the RNN achieves the best decoding performance, nonetheless yielding the highest computational overhead. Liao et al. employ MLPs to perform accurate Rayleigh fading channel prediction [459]. The authors further equip their proposal with a sparse channel sample construction method to save system resources without compromising precision. Deep learning can further aid visible light communication. In [461], Huang et al. employ a deep learning based system for error correction in optical communications. Specifically, an AE is used in their work to perform dimension reduction on light-emitting diode (LED) visible light downlink, thereby maximizing the channel bandwidth. The proposal follows the theory in [451], where O’Shea et al. demonstrate that deep learning driven signal processing systems can perform as good as traditional encoding and/or modulation systems.

Deep learning has been further adopted in solving millimeter wave beamforming. In [447], Alkhateeb et al. propose a millimeter wave communication system that utilizes MLPs to predict beamforming vectors from signals received from distributed base stations. By substituting a genie-aided solution with deep learning, their framework reduces the coordination overhead, enabling wide-coverage and low-latency beamforming. Similarly, Gante et al. employ CNNs to infer the position of a device, given the received millimeter wave radiation. Their preliminary simulations show that the CNN-based system can achieve small estimation errors in a realistic outdoors scenario, significantly outperforming existing prediction approaches.

**Lessons learned:** Deep learning is beginning to play an
important role in signal processing applications and the performance demonstrated by early prototypes is remarkable. This is because deep learning can prove advantageous with regards to performance, complexity, and generalization capabilities. At this stage, research in this area is however incipient. We can only expect that deep learning will become increasingly popular in this area.

J. Emerging Deep Learning Applications in Mobile Networks

In this part, we review work that builds upon deep learning in other mobile networking areas, which are beyond the scopes of the subjects discussed thus far. These emerging applications open several new research directions, as we discuss next. A summary of these works is given in Table XIX.

**Network Data Monetization:** Gonzalez et al. employ unsupervised deep learning to generate real-time accurate user profiles [462] using an on-network machine learning platform called Net2Vec [510]. Specifically, they analyze user browsing data in real time and generate user profiles using product categories. The profiles can be subsequently associated with the products that are of interest to the users and employed for online advertising.

**IoT In-Network Computation:** Instead of regarding IoT nodes as producers of data or the end consumers of processed information, Kaminski et al. embed neural networks into an IoT deployment and allow the nodes to collaboratively process the data generated [463]. This enables low-latency communication, while offloading data storage and processing from the cloud. In particular, the authors map each hidden unit of a pre-trained neural network to a node in the IoT network, and investigate the optimal projection that leads to the minimum communication overhead. Their framework achieves functionality similar to in-network computation in WSNs and opens a new research directions in fog computing.

**Mobile Crowdsensing:** Xiao et al. investigate vulnerabilities facing crowdsensing in the mobile network context. They argue that there exist malicious mobile users who intentionally provide false sensing data to servers, to save costs and preserve their privacy, which in turn can make mobile crowdsensings systems vulnerable [464]. The authors model the server-users system as a Stackelberg game, where the server plays the role of a leader that is responsible for evaluating the sensing effort of individuals, by analyzing the accuracy of each sensing report. Users are paid based on the evaluation of their efforts, hence cheating users will be punished with zero reward. To design an optimal payment policy, the server employs a deep Q network, which derives knowledge from experience sensing reports, without requiring specific sensing models. Simulations demonstrate superior performance in terms of sensing quality, resilience to attacks, and server utility, as compared to traditional Q learning based and random payment strategies.

**Mobile Blockchain:** Substantial computing resource requirements and energy consumption limit the applicability of blockchain in mobile network environments. To mitigate this problem, Luong et al. shed light on resource management in mobile blockchain networks based on optimal auction in [465]. They design an MLP to first conduct monotone transformations of the miners’ bids and subsequently output the allocation scheme and conditional payment rules for each miner. By running experiments with different settings, the results suggest the proposed deep learning based framework can deliver much higher profit to edge computing service provider than the second-price auction baseline.

**Internet of Vehicles (IoV):** Gulati et al. extend the success of deep learning to IoV [466]. The authors design a deep learning-based content centric data dissemination approach that comprises three steps, namely (i) performing energy estimation on selected vehicles that are capable of data dissemination; (ii) employing a Weiner process model to identify stable and reliable connections between vehicles; and (iii) using a CNN to predict the social relationship among vehicles. Experiments unveil that the volume of data disseminated is positively related to social score, energy levels, and number of vehicles, while the speed of vehicles has negative impact on the connection probability.

**Lessons learned:** The adoption of deep learning in the mobile and wireless networking domain is exciting and undoubtedly many advances are yet to appear. However, as discussed in Sec. III-E, deep learning solutions are not universal and may not be suitable for every problem. One should rather regard deep learning as a powerful tool that can assist with fast and accurate inference, and facilitate the automation of some processes previously requiring human intervention. Nevertheless, deep learning algorithms will make mistakes, and their decisions might not be easy to interpret. In tasks that require high interpretability and low fault-tolerance, deep learning still has a long way to go, which also holds for the

<table>
<thead>
<tr>
<th>Reference</th>
<th>Application</th>
<th>Model</th>
<th>Key contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gonzalez et al.</td>
<td>Network data monetization</td>
<td>-</td>
<td>A platform named Net2Vec to facilitate deep learning deployment in communication networks.</td>
</tr>
<tr>
<td>Kaminski et al.</td>
<td>In-network computation for IoT</td>
<td>MLP</td>
<td>Enables to perform collaborative data processing and reduces latency.</td>
</tr>
<tr>
<td>Xiao et al.</td>
<td>Mobile crowdsensing</td>
<td>Deep Q learning</td>
<td>Mitigates vulnerabilities of mobile crowdsensing systems.</td>
</tr>
<tr>
<td>Luong et al.</td>
<td>Resource allocation for mobile blockchains</td>
<td>MLP</td>
<td>Employs deep learning to perform monotone transformations of miners' bids and outputs the allocation and conditional payment rules in optimal auctions.</td>
</tr>
<tr>
<td>Gulati et al.</td>
<td>Data dissemination in Internet of Vehicles (IoV)</td>
<td>CNN</td>
<td>Investigates the relationship between data dissemination performance and social score, energy level, number of vehicles and their speed.</td>
</tr>
</tbody>
</table>
TABLE XX: Summary of works on improving deep learning for mobile devices and systems.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Methods</th>
<th>Target model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iandola et al. [516]</td>
<td>Filter size shrinking, reducing input channels and late downsampling</td>
<td>CNN</td>
</tr>
<tr>
<td>Howard et al. [517]</td>
<td>Depth-wise separable convolution</td>
<td>CNN</td>
</tr>
<tr>
<td>Zhang et al. [518]</td>
<td>Point-wise group convolution and channel shuffle</td>
<td>CNN</td>
</tr>
<tr>
<td>Zhang et al. [519]</td>
<td>Tucker decomposition</td>
<td>AE</td>
</tr>
<tr>
<td>Cao et al. [520]</td>
<td>Data parallelization by RenderScript</td>
<td>RNN</td>
</tr>
<tr>
<td>Chen et al. [521]</td>
<td>Space exploration for data reusability and kernel redundancy removal</td>
<td>CNN</td>
</tr>
<tr>
<td>Rallapalli et al. [522]</td>
<td>Memory optimizations</td>
<td>CNN</td>
</tr>
<tr>
<td>Lane et al. [523]</td>
<td>Runtime layer compression and deep architecture decomposition</td>
<td>MLP, CNN</td>
</tr>
<tr>
<td>Huynh et al. [524]</td>
<td>Caching, Tucker decomposition and computation offloading</td>
<td>CNN</td>
</tr>
<tr>
<td>Wu et al. [525]</td>
<td>Parameters quantization</td>
<td>CNN</td>
</tr>
<tr>
<td>Bhattacharya and Lane [526]</td>
<td>Sparsification of fully-connected layers and separation of convolutional kernels</td>
<td>MLP, CNN</td>
</tr>
<tr>
<td>Georgiev et al. [527]</td>
<td>Parameters sharing</td>
<td>MLP</td>
</tr>
<tr>
<td>Cho and Brand [528]</td>
<td>Convolution operation optimization</td>
<td>CNN</td>
</tr>
<tr>
<td>Guo and Potkonjak [529]</td>
<td>Filters and classes pruning</td>
<td>CNN</td>
</tr>
<tr>
<td>Li et al. [529]</td>
<td>Cloud assistance and incremental learning</td>
<td>CNN</td>
</tr>
<tr>
<td>Zen et al. [530]</td>
<td>Weight quantization</td>
<td>LSTM</td>
</tr>
<tr>
<td>Falcao et al. [531]</td>
<td>Parallelization and memory sharing</td>
<td>Stacked AE</td>
</tr>
<tr>
<td>Fang et al. [532]</td>
<td>Model pruning and recovery scheme</td>
<td>CNN</td>
</tr>
<tr>
<td>Xu et al. [533]</td>
<td>Reusable region lookup and reusable region propagation scheme</td>
<td>CNN</td>
</tr>
<tr>
<td>Liu et al. [534]</td>
<td>Using deep Q learning based optimizer to achieve appropriate balance between accuracy, latency, storage and energy consumption for deep NNs on mobile platforms</td>
<td>CNN</td>
</tr>
<tr>
<td>Chen et al. [535]</td>
<td>Machine learning based optimization system to automatically explore and search for optimized tensor operators</td>
<td>All NN architectures</td>
</tr>
<tr>
<td>Yao et al. [536]</td>
<td>Learning model execution time and performing the model compression</td>
<td>MLP, CNN, GRU, and LSTM</td>
</tr>
<tr>
<td>Li et al. [537]</td>
<td>Streamline slimming and branch slimming</td>
<td>CNN</td>
</tr>
</tbody>
</table>

VII. TAILORING DEEP LEARNING TO MOBILE NETWORKS

Although deep learning performs remarkably in many mobile networking areas, the No Free Lunch (NFL) theorem indicates that there is no single model that can work universally well in all problems [511]. This implies that for any specific mobile and wireless networking problem, we may need to adapt different deep learning architectures to achieve the best performance. In this section, we look at how to tailor deep learning to mobile networking applications from three perspectives, namely, mobile devices and systems, distributed data centers, and changing mobile network environments.

A. Tailoring Deep Learning to Mobile Devices and Systems

The ultra-low latency requirements of future 5G networks demand runtime efficiency from all operations performed by mobile systems. This also applies to deep learning driven applications. However, current mobile devices have limited hardware capabilities, which means that implementing complex deep learning architectures on such equipment may be computationally unfeasible, unless appropriate model tuning is performed. To address this issue, ongoing research improves existing deep learning architectures [512], such that the inference process does not violate latency or energy constraints [513], [514], nor raise any privacy concern [515]. We outline these works in Table XX and discuss their key contributions next.

Iandola et al. design a compact architecture for embedded systems named SqueezeNet, which has similar accuracy to that of AlexNet [88], a classical CNN, yet 50 times fewer parameters [516]. SqueezeNet is also based on CNNs, but its significantly smaller model size (i) allows more efficiently training on distributed systems; (ii) reduces the transmission overhead when updating the model at the client side; and (iii) facilitates deployment on resource-limited embedded devices. Howard et al. extend this work and introduce an efficient family of streamlined CNNs called MobileNet, which uses depth-wise separable convolution operations to drastically reduce the number of computations required and the model size [517]. This new design can run with low latency and can satisfy the requirements of mobile and embedded vision applications. The authors further introduce two hyper-parameters to control the width and resolution of multipliers, which can help strike an appropriate trade-off between accuracy and efficiency. The ShuffleNet proposed by Zhang et al. improves the accuracy of MobileNet by employing point-wise group convolution and channel shuffle, while retaining similar model complexity [518]. In particular, the authors discover that more groups of convolution operations can reduce the computation requirements.

Zhang et al. focus on reducing the number of parameters of structures with fully-connected layers for mobile multimedia features learning [519]. This is achieved by applying Trucker decomposition to weight sub-tensors in the model, while maintaining decent reconstruction capability. The Trucker decomposition has also been employed in [524], where the authors seek to approximate a model with fewer parameters, in order to save memory. Mobile optimizations are further studied for RNN models. In [520], Cao et al. use a mobile toolbox called RenderScript to parallelize specific data structures and enable mobile GPUs to perform computational accelerations. Their proposal reduces the latency when running RNN models

on Android smartphones. Chen et al. shed light on implementing CNNs on iOS mobile devices [521]. In particular, they reduce the model executions latency, through space exploration for data re-usability and kernel redundancy removal. The former alleviates the high bandwidth requirements of convolutional layers, while the latter reduces the memory and computational requirements, with negligible performance degradation.

Rallapalli et al. investigate offloading very deep CNNs from clouds to edge devices, by employing memory optimization on both mobile CPUs and GPUs [522]. Their framework enables running at high speed deep CNNs with large memory requirements in mobile object detection applications. Lane et al. develop a software accelerator, DeepX, to assist deep learning implementations on mobile devices. The proposed approach exploits two inference-time resource control algorithms, i.e., runtime layer compression and deep architecture decomposition [523]. The runtime layer compression technique controls the memory and computation runtime during the inference phase, by extending model compression principles. This is important in mobile devices, since offloading the inference process to edge devices is more practical with current hardware platforms. Further, the deep architecture designs “decomposition plans” that seek to optimally allocate data and model operations to local and remote processors. By combining these two, DeepX enables maximizing energy and runtime efficiency, under given computation and memory constraints. Yao et al. [536] design a framework called FastDeepIoT, which first learns the execution time of NN models on target devices, and subsequently conducts model compression to reduce the runtime without compromising the inference accuracy. Through this process, up to 78% of execution time and 69% of energy consumption is reduced, compared to state-of-the-art compression algorithms.

More recently, Fang et al. design a framework called NestDNN, to provide flexible resource-accuracy trade-offs on mobile devices [532]. To this end, the NestDNN first adopts a model pruning and recovery scheme, which translates deep NNs to single compact multi-capacity models. With this approach up to 4.22% inference accuracy can be achieved with six mobile vision applications, at a 2.0× faster video frame processing rate and reducing energy consumption by 1.7×. In [533], Xu et al. accelerate deep learning inference for mobile vision from the caching perspective. In particular, the proposed framework called DeepCache stores recent input frames as cache keys and recent feature maps for individual CNN layers as cache values. The authors further employ reusable region lookup and reusable region propagation, to enable a region matcher to only run once per input video frame and load cached feature maps at all layers inside the CNN. This reduces the inference time by 18% and energy consumption by 20% on average. Liu et al. develop a usage-driven framework named AdaDeep, to select a combination of compression techniques for a specific deep NN on mobile platforms [534]. By using a deep Q learning optimizer, their proposal can achieve appropriate trade-offs between accuracy, latency, storage and energy consumption.

Beyond these works, researchers also successfully adapt deep learning architectures through other designs and sophisticated optimizations, such as parameters quantization [525], [530], model slimming [537], sparsification and separation [526], representation and memory sharing [99], [531], convolution operation optimization [527], pruning [528], cloud assistance [529] and compiler optimization [535]. These techniques will be of great significance when embedding deep neural networks into mobile systems.

B. Tailoring Deep Learning to Distributed Data Containers

Mobile systems generate and consume massive volumes of mobile data every day. This may involve similar content, but which is distributed around the world. Moving such data to centralized servers to perform model training and evaluation inevitably introduces communication and storage overheads, which does not scale. However, neglecting characteristics embedded in mobile data, which are associated with local culture, human mobility, geographical topology, etc., during model training can compromise the robustness of the model and implicitly the performance of the mobile network applications that build on such models. The solution is to offload model execution to distributed data centers or edge devices, to guarantee good performance, whilst alleviating the burden on the cloud.

As such, one of the challenges facing parallelism, in the context of mobile networking, is that of training neural networks on a large number of mobile devices that are battery powered, have limited computational capabilities and in particular lack GPUs. The key goal of this paradigm is that of training with a large number of mobile CPUs at least as effective as with GPUs. The speed of training remains important, but becomes a secondary goal.

Generally, there are two routes to addressing this problem, namely, (i) decomposing the model itself, to train (or make inference with) its components individually; or (ii) scaling the training process to perform model update at different locations associated with data containers. Both schemes allow one to train a single model without requiring to centralize all data. We illustrate the principles of these two approaches in Fig. 19 and summarize the existing work in Table XXI.

Model Parallelism. Large-scale distributed deep learning is first studied in [128], where the authors develop a framework named DistBelief, which enables training complex neural networks on thousands of machines. In their framework, the full model is partitioned into smaller components and distributed over various machines. Only nodes with edges (e.g. connections between layers) that cross boundaries between machines are required to communicate for parameters update and inference. This system further involves a parameter server, which enables each model replica to obtain the latest parameters during training. Experiments demonstrate that the proposed framework can be training significantly faster on a CPU cluster, compared to training on a single GPU, while achieving state-of-the-art classification performance on ImageNet [193].

Teerapittayanon et al. propose deep neural networks tailored to distributed systems, which include cloud servers, fog layers
TABLE XXI: Summary of work on model and training parallelism for mobile systems and devices.

<table>
<thead>
<tr>
<th>Parallelism Paradigm</th>
<th>Reference</th>
<th>Target</th>
<th>Core Idea</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model parallelism</td>
<td>Dean et al. [128]</td>
<td>Very large deep neural networks in distributed systems.</td>
<td>Employs downpour SGD to support a large number of model replicas and a Sandblaster framework to support a variety of batch optimizations.</td>
<td>Up to 12× model training speed up, using 81 machines.</td>
</tr>
<tr>
<td>De Coninck et al. [115]</td>
<td>Neural networks on IoT devices.</td>
<td>Maps a deep neural network to a distributed setting and jointly trains each individual section.</td>
<td>Up to 20× reduction of communication cost.</td>
<td></td>
</tr>
<tr>
<td>Training parallelism</td>
<td>Recht et al. [540]</td>
<td>Parallelized SGD.</td>
<td>Eliminates overhead associated with locking in distributed SGD.</td>
<td>Up to 10× speed up in distributed training.</td>
</tr>
<tr>
<td>Goyal et al. [541]</td>
<td>Distributed synchronous SGD.</td>
<td>Employs a hyper-parameter-free learning rule to adjust the learning rate and a warmup mechanism to address the early optimization problem.</td>
<td>Trains billions of images per day.</td>
<td></td>
</tr>
<tr>
<td>Zhang et al. [542]</td>
<td>Asynchronous distributed SGD.</td>
<td>Combines the stochastic variance reduced gradient algorithm and a delayed proximal gradient algorithm.</td>
<td>Up to 6× speed up</td>
<td></td>
</tr>
<tr>
<td>Hardy et al. [543]</td>
<td>Distributed deep learning on edge-devices.</td>
<td>Compression technique (AdaComp) to reduce ingress traffic at parameter servers.</td>
<td>Up to 191× reduction in ingress traffic.</td>
<td></td>
</tr>
<tr>
<td>McMahan et al. [544]</td>
<td>Distributed training on mobile devices.</td>
<td>Users collectively enjoy benefits of shared models trained with big data without centralized storage.</td>
<td>Up to 64.3× training speedup.</td>
<td></td>
</tr>
<tr>
<td>Keith et al. [545]</td>
<td>Data computation over mobile devices.</td>
<td>Secure multi-party computation to obtain model parameters on distributed mobile devices.</td>
<td>Up to 1.98× communication expansion.</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 19: The underlying principles of model parallelism (left) and training parallelism (right).

and geographically distributed devices [538]. The authors scale the overall neural network architecture and distribute its components hierarchically from cloud to end devices. The model exploits local aggregators and binary weights, to reduce computational storage, and communication overheads, while maintaining decent accuracy. Experiments on a multi-view multi-camera dataset demonstrate that this proposal can perform efficient cloud-based training and local inference. Importantly, without violating latency constraints, the deep neural network obtains essential benefits associated with distributed systems, such as fault tolerance and privacy.

Coninck et al. consider distributing deep learning over IoT for classification applications [115]. Specifically, they deploy a small neural network to local devices, to perform coarse classification, which enables fast response filtered data to be sent to central servers. If the local model fails to classify, the larger neural network in the cloud is activated to perform fine-grained classification. The overall architecture maintains good accuracy, while significantly reducing the latency typically introduced by large model inference.

Decentralized methods can also be applied to deep reinforcement learning. In [539], Omidshafiei et al. consider a multi-agent system with partial observability and limited communication, which is common in mobile systems. They combine a set of sophisticated methods and algorithms, including hysteresis learners, a deep recurrent Q network,
concurrent experience replay trajectories and distillation, to enable multi-agent coordination using a single joint policy under a set of decentralized partially observable MDPs. Their framework can potentially play an important role in addressing control problems in distributed mobile systems.

**Training Parallelism** is also essential for mobile system, as mobile data usually come asynchronously from different sources. Training models effectively while maintaining consistency, fast convergence, and accuracy remains however challenging [546].

A practical method to address this problem is to perform asynchronous SGD. The basic idea is to enable the server that maintains a model to accept delayed information (e.g. data, gradient updates) from workers. At each update iteration, the server only requires to wait for a smaller number of workers. This is essential for training a deep neural network over distributed machines in mobile systems. The asynchronous SGD is first studied in [540], where the authors propose a lock-free parallel SGD named HOGWILD, which demonstrates significant faster convergence over locking counterparts. The Downpour SGD in [128] improves the robustness of the training process when work nodes breakdown, as each model replica requests the latest version of the parameters. Hence a small number of machine failures will not have a significant impact on the training process. A similar idea has been employed in [541], where Goyal et al. investigate the usage of a set of techniques (i.e. learning rate adjustment, warm-up, batch normalization), which offer important insights into training large-scale deep neural networks on distributed systems. Eventually, their framework can train an network on ImageNet within 1 hour, which is impressive in comparison with traditional algorithms.

Zhang et al. argue that most of asynchronous SGD algorithms suffer from slow convergence, due to the inherent variance of stochastic gradients [542]. They propose an improved SGD with variance reduction to speed up the convergence. Their algorithm outperforms other asynchronous SGD approaches in terms of convergence, when training deep neural networks on the Google Cloud Computing Platform. The asynchronous method has also been applied to deep reinforcement learning. In [80], the authors create multiple environments, which allows agents to perform asynchronous updates to the main structure. The new A3C algorithm breaks the sequential dependency and speeds up the training of the traditional Actor-Critic algorithm significantly. In [543], Hardy et al. further study distributed deep learning over cloud and edge devices. In particular, they propose a training algorithm, AdaComp, which allows to compress worker updates of the target model. This significantly reduce the communication overhead between cloud and edge, while retaining good fault tolerance.

Federated learning is an emerging parallelism approach that enables mobile devices to collaboratively learn a shared model, while retaining all training data on individual devices [544], [547]. Beyond offloading the training data from central servers, this approach performs model updates with a Secure Aggregation protocol [545], which decrypts the average updates only if enough users have participated, without inspecting individual updates. Based on this idea, Google recently build a prototype system using federated Learning in the domain of mobile devices [548]. This fulfills the objective that “bringing the code to the data, instead of the data to the code”, which protects individuals’ privacy.

### C. Tailoring Deep Learning to Changing Mobile Network Environments

Mobile network environments often exhibit changing patterns over time. For instance, the spatial distributions of mobile data traffic over a region may vary significantly between different times of the day [549]. Applying a deep learning model in changing mobile environments requires lifelong learning ability to continuously absorb new features, without forgetting old but essential patterns. Moreover, new smartphone-targeted viruses are spreading fast via mobile networks and may severely jeopardize users’ privacy and business profits. These pose unprecedented challenges to current anomaly detection systems and anti-virus software, as such tools must react to new threats in a timely manner, using limited information. To this end, the model should have transfer learning ability, which can enable the fast transfer of knowledge from pre-trained models to different jobs or datasets. This will allow models to work well with limited threat samples (one-shot learning) or limited metadata descriptions of new threats (zero-shot learning). Therefore, both lifelong learning and transfer learning are essential for applications in ever changing mobile network environments. We illustrated these two learning paradigms in Fig. 20 and review essential research in this subsection.

**Deep Lifelong Learning** mimics human behaviors and seeks to build a machine that can continuously adapt to new environments, retain as much knowledge as possible from previous learning experience [550]. There exist several research efforts that adapt traditional deep learning to lifelong learning. For example, Lee et al. propose a dual-memory deep learning architecture for lifelong learning of everyday human behaviors over non-stationary data streams [551]. To enable the pre-trained model to retain old knowledge while training with new data, their architecture includes two memory buffers, namely a deep memory and a fast memory. The deep memory is composed of several deep networks, which are built when the amount of data from an unseen distribution is accumulated and reaches a threshold. The fast memory component is a small neural network, which is updated immediately when coming across a new data sample. These two memory modules allow to perform continuous learning without forgetting old knowledge. Experiments on a non-stationary image data stream prove the effectiveness of this model, as it significantly outperforms other online deep learning algorithms. The memory mechanism has also been applied in [552]. In particular, the authors introduce a differentiable neural computer, which allows neural networks to dynamically read from and write to an external memory module. This enables lifelong lookup and forgetting of knowledge from external sources, as humans do.
Parisi et al. consider a different lifelong learning scenario in [553]. They abandon the memory modules in [551] and design a self-organizing architecture with recurrent neurons for processing time-varying patterns. A variant of the Growing When Required network is employed in each layer, to predict neural activation sequences from the previous network layer. This allows learning time-vary correlations between inputs and labels, without requiring a predefined number of classes. Importantly, the framework is robust, as it has tolerance to missing and corrupted sample labels, which is common in mobile data.

Another interesting deep lifelong learning architecture is presented in [554], where Tessler et al. build a DQN agent that can retain learned skills in playing the famous computer game Minecraft. The overall framework includes a pre-trained model, Deep Skill Network, which is trained a-priori on various sub-tasks of the game. When learning a new task, the old knowledge is maintained by incorporating reusable skills through a Deep Skill module, which consists of a Deep Skill Network array and a multi-skill distillation network. These allow the agent to selectively transfer knowledge to solve a new task. Experiments demonstrate that their proposal significantly outperforms traditional double DQNs in terms of accuracy and convergence. This technique has potential to be employed in solving mobile networking problems, as it can continuously acquire new knowledge.

Deep Transfer Learning: Unlike lifelong learning, transfer learning only seeks to use knowledge from a specific domain to aid learning in a target domain. Applying transfer learning can accelerate the new learning process, as the new task does not require to learn from scratch. This is essential to mobile network environments, as they require to agilely respond to new network patterns and threats. A number of important applications emerge in the computer network domain [57], such as Web mining [555], caching [556] and base station sleep strategies [210].

There exist two extreme transfer learning paradigms, namely one-shot learning and zero-shot learning. One-shot learning refers to a learning method that gains as much information as possible about a category from only one or a handful of samples, given a pre-trained model [557]. On the other hand, zero-shot learning does not require any sample from a category [558]. It aims at learning a new distribution given meta description of the new category and correlations with existing training data. Though research towards deep one-shot learning [97], [559] and deep zero-shot learning [560], [561] is in its infancy, both paradigms are very promising in detecting new threats or traffic patterns in mobile networks.

VIII. FUTURE RESEARCH PERSPECTIVES

As deep learning is achieving increasingly promising results in the mobile networking domain, several important research issues remain to be addressed in the future. We conclude our survey by discussing these challenges and pinpointing key mobile networking research problems that could be tackled with novel deep learning tools.

A. Serving Deep Learning with Massive High-Quality Data

Deep neural networks rely on massive and high-quality data to achieve good performance. When training a large and complex architecture, data volume and quality are very important, as deeper models usually have a huge set of parameters to be learned and configured. This issue remains true in mobile network applications. Unfortunately, unlike in other research areas such as computer vision and NLP, high-quality and large-scale labeled datasets still lack for mobile network applications, because service providers and operators keep the data collected confidential and are reluctant to release datasets. While this makes sense from a user privacy standpoint, to some extent it restricts the development of deep learning mechanisms for problems in the mobile networking domain. Moreover, mobile data collected by sensors and network equipment are frequently subject to loss, redundancy, mislabeling and class imbalance, and thus cannot be directly employed for training purpose.
To build intelligent 5G mobile network architecture, efficient and mature streamlining platforms for mobile data processing are in demand. This requires considerable amount of research efforts for data collection, transmission, cleaning, clustering, transformation, and anonymization. Deep learning applications in the mobile network area can only advance if researchers and industry stakeholder release more datasets, with a view to benefiting a wide range of communities.

B. Deep Learning for Spatio-Temporal Mobile Data Mining

Accurate analysis of mobile traffic data over a geographical region is becoming increasingly essential for event localization, network resource allocation, context-based advertising and urban planning [549]. However, due to the mobility of smartphone users, the spatio-temporal distribution of mobile traffic [562] and application popularity [563] are difficult to understand (see the example city-scale traffic snapshot in Fig. 21). Recent research suggests that data collected by mobile sensors (e.g. mobile traffic) over a city can be regarded as pictures taken by panoramic cameras, which provide a city-scale sensing system for urban surveillance [565]. These traffic sensing images enclose information associated with the movements of individuals [469].

From both spatial and temporal dimensions perspective, we recognize that mobile traffic data have important similarity with videos or speech, which is an analogy made recently also in [219] and exemplified in Fig. 22. Specifically, both videos and the large-scale evolution of mobile traffic are composed of sequences of “frames”. Moreover, if we zoom into a small coverage area to measure long-term traffic consumption, we can observe that a single traffic consumption series looks similar to a natural language sequence. These observations suggest that, to some extent, well-established tools for computer vision (e.g. CNN) or NLP (e.g. RNN, LSTM) are promising candidate for mobile traffic analysis.

Beyond these similarity, we observe several properties of mobile traffic that makes it unique in comparison with images or language sequences. Namely,

1) The values of neighboring ‘pixels’ in fine-grained traffic snapshots are not significantly different in general, while this happens quite often at the edges of natural images.

2) Single mobile traffic series usually exhibit some periodicity (both daily and weekly), yet this is not a feature seen among video pixels.

3) Due to user mobility, traffic consumption is more likely to stay or shift to neighboring cells in the near future, which is less likely to be seen in videos.

Such spatio-temporal correlations in mobile traffic can be exploited as prior knowledge for model design. We recognize several unique advantages of employing deep learning for mobile traffic data mining:

1) CNN structures work well in imaging applications, thus can also serve mobile traffic analysis tasks, given the analogies mentioned before.

2) LSTMs capture well temporal correlations in time series data such as natural language; hence this structure can also be adapted to traffic forecasting problems.

3) GPU computing enables fast training of NNs and together with parallelization techniques can support low-latency mobile traffic analysis via deep learning tools.

In essence, we expect deep learning tools tailored to mobile networking, will overcome the limitation of traditional regression and interpolation tools such as Exponential Smoothing [566], Autoregressive Integrated Moving Average model [567], or unifrom interpolation, which are commonly used in operational networks.

C. Deep Learning for Geometric Mobile Data Mining

As discussed in Sec. III-D, certain mobile data has important geometric properties. For instance, the location of mobile users or base stations along with the data carried can be viewed as point clouds in a 2D plane. If the temporal dimension is also added, this leads to a 3D point cloud representation, with either fixed or changing locations. In addition, the connectivity of mobile devices, routers, base stations, gateways, and so on can naturally construct a directed graph, where entities are represented as vertices, the links between them can be seen as
edges, and data flows may give direction to these edges. We show examples of geometric mobile data and their potential representations in Fig. 23. At the top of the figure a group of mobile users is represented as a point cloud. Likewise, mobile network entities (e.g. base station, gateway, users) are regarded as graphs below, following the rationale explained below. Due to the inherent complexity of such representations, traditional ML tools usually struggle to interpret geometric data and make reliable inferences.

In contrast, a variety of deep learning toolboxes for modelling geometric data exist, albeit not having been widely employed in mobile networking yet. For instance, PointNet [568] and the follow on PointNet++ [102] are the first solutions that employ deep learning for 3D point cloud applications, including classification and segmentation [569]. We recognize that similar ideas can be applied to geometric mobile data analysis, such as clustering of mobile users or base stations, or user trajectory predictions. Further, deep learning for graphical data analysis is also evolving rapidly [570]. This is triggered by research on Graph CNNs [103], which brings convolution concepts to graph-structured data. The applicability of Graph CNNs can be further extend to the temporal domain [571]. One possible application is the prediction of future traffic demand at individual base station level. We expect that such novel architectures will play an increasingly important role in network graph analysis and applications such as anomaly detection over a mobile network graph.

**D. Deep Unsupervised Learning in Mobile Networks**

We observe that current deep learning practices in mobile networks largely employ supervised learning and reinforcement learning. However, as mobile networks generate considerable amounts of unlabeled data every day, data labeling is costly and requires domain-specific knowledge. To facilitate the analysis of raw mobile network data, unsupervised learning becomes essential in extracting insights from unlabeled data [572], so as to optimize the mobile network functionality to improve QoE.

The potential of a range of unsupervised deep learning tools including AE, RBM and GAN remains to be further explored. In general, these models require light feature engineering and are thus promising for learning from heterogeneous and unstructured mobile data. For instance, deep AEs work well for unsupervised anomaly detection [573]. Though less popular, RBMs can perform layer-wise unsupervised pre-training, which can accelerate the overall model training process. GANs are good at imitating data distributions, thus could be employed to mimic real mobile network environments. Recent research reveals that GANs can even protect communications by crafting custom cryptography to avoid eavesdropping [574]. All these tools require further research to fulfill their full potentials in the mobile networking domain.

**E. Deep Reinforcement Learning for Mobile Network Control**

Many mobile network control problems have been solved by constrained optimization, dynamic programming and game theory approaches. Unfortunately, these methods either make strong assumptions about the objective functions (e.g. function convexity) or data distribution (e.g. Gaussian or Poisson distributed), or suffer from high time and space complexity. As mobile networks become increasingly complex, such assumptions sometimes turn unrealistic. The objective functions are further affected by their increasingly large sets of variables, that pose severe computational and memory challenges to existing mathematical approaches.

In contrast, deep reinforcement learning does not make strong assumptions about the target system. It employs function approximation, which explicitly addresses the problem of large state-action spaces, enabling reinforcement learning to scale to network control problems that were previously considered hard. Inspired by remarkable achievements in Atari [19] and Go [575] games, a number of researchers begin...
to explore DRL to solve complex network control problems, as we discussed in Sec. VI-G. However, these works only scratch the surface and the potential of DRL to tackle mobile network control problems remains largely unexplored. For instance, as DeepMind trains a DRL agent to reduce Google’s data centers cooling bill, DRL could be exploited to extract rich features from cellular networks and enable intelligent on/off base stations switching, to reduce the infrastructure’s energy footprint. Such exciting applications make us believe that advances in DRL that are yet to appear can revolutionize the autonomous control of future mobile networks.

F. Summary

Deep learning is playing an increasingly important role in the mobile and wireless networking domain. In this paper, we provided a comprehensive survey of recent work that lies at the intersection between deep learning and mobile networking. We summarized both basic concepts and advanced principles of various deep learning models, then reviewed work specific to mobile networks across different application scenarios. We discussed how to tailor deep learning models to general mobile networking applications, an aspect overlooked by previous surveys. We concluded by pinpointing several open research issues and promising directions, which may lead to valuable future research results. Our hope is that this article will become a definite guide to researchers and practitioners interested in applying machine intelligence to complex problems in mobile network environments.

ACKNOWLEDGEMENT

We would like to thank Zongzuo Wang for sharing valuable insights on deep learning, which helped improving the quality of this paper. We also thank the anonymous reviewers and editors, whose detailed and thoughtful feedback helped us improve this survey more depth and a broader scope.

REFERENCES


36 DeepMind AI Reduces Google Data Center Cooling Bill by 40% https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/


[198] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional LSTM. In Proc. IEEE Work-


Deep reinforcement learning based computation offloading and resource allocation for MEC.


Fabien Geyer and Georg Carle. Learning and Generating Distributed Routing Protocols Using Graph-Based Deep Learning.


IEEE access, 2018.


Ying He, F Richard Yu, Nan Zhao, Victor CM Leung, and Hongxi Yin. Software-defined networks with mobile computing and caching for smart cities: A big data deep reinforcement learning approach.


Quang Tran Anh Pham, Yassine Hadjadj-Aoul, and Abdellah Oustaloup. Deep Reinforcement Learning based QoS-aware Routing in Knowledge-developed networking.


Qingqiang Fang, Zhang Dou, Chunmei Li, and Guangzheng Si. Anomaly detection of spectrum in wireless communication via deep autoencoder.


Muhammad Altaf Khan, Shahullah Khan, Bilal Shams, and Jaime Lloret. Distributed flood attack detection mechanism using artificial neural network in wired mesh networks.


Abebe Abeshu Diro and Naveen Chilamkurti. Distributed attack detection scheme using deep learning approach for Internet of Things.


Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IoT.


Ramakrishna Das, Akshay Gadre, Shanghang Zhang, Swaran Kumar, and Jose MF Moura. A deep learning approach to IoT authentication.


Peng Jiang, Hongyi Wu, Cong Wang, and Chunsheng Xin. Virtual MAC spoofing detection through deep learning.


Zhenlong Yuan, Yongqiang Lu, Zhaoguo Wang, and Yibo Xue. DroidSec: deep learning in Android malware detection.


Xin Su, Dafang Zhang, Wenjia Li, and Kai Zhao. A deep learning approach to Android malware feature learning and detection.


Khoi Khac Nguyen, Dinh Thai Hoang, Dusit Niyato, Ping Wang, Deep Nguyen, and Eryk Dutkiewicz. Cyberattack detection in mobile cloud computing: A deep learning approach.


Seema Eslahi, Moslem Yousefi, Maryam Var Naseri, YM Yussof, NM Tahir, and H Hashim. Mobile botnet detection model based on retrospective pattern recognition.


Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning.


In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.


Shuochao Yao, Yiran Zhao, Huajie Shao, ShengZhong Liu, Dongxin Liu, Lu Su, and Tarek Abdelzaher. FastDeepIoT: Towards Understand-


