Smart Cities with Deep Edges

Gary White(✉) and Siobhán Clarke

School of Computer Science and Statistics, Trinity College Dublin, Dublin, Ireland
{whiteg5,siobhan.clarke}@scss.tcd.ie

Abstract. With the advent of deep learning and new embedded devices capable of running these models at the edge of the network there is potential for deep edges in IoT and smart cities. This will enable a considerable increase in the analytics and urban reasoning that can take place at the edge of the network. The end-to-end latency for these models will also be reduced due to the physical proximity of the edge devices, which allows reasoning one hop away from data generation. This will enable a range of urban reasoning applications that require reduced latency and jitter such as vehicle collision detection, network demand prediction and smart grids. The increased accuracy of deep learning models at the edge will reduce traffic flow to the cloud as only a subset of the data will need to be reported after a first pass analysis. This will improve the privacy of users as edge devices can process the reported data to remove identifiable information to keep the user anonymous before sending it to the cloud. This multi-stage analytics allows for initial urban reasoning on a city wide scale for deriving context information with additional analytics in the cloud focusing on certain domain challenges. In this paper we describe the architecture and advantages of deep edges and compare it against alternative IoT urban reasoning architectures such as cloud-based and traditional embedded devices such as raspberry pis.

1 Introduction

Urban reasoning provides insight into the major problems that our cities face (e.g., crowd congestion, increased network demand, air pollution, water floods, etc.) to allow for the efficient running of a city. Problems such as crowd congestion and increased network demand are set to get worse with the UN predicting 60 per cent of people globally and one in every three people will live in cities with at least half a million inhabitants by 2030 [1]. This will put additional strain on our cities and we need effective urban reasoning algorithms and a suitable architecture to deploy them on, to be able to handle these increased demands. The Internet of Things (IoT) enables easier access and interaction to a wide variety of devices such as CCTV cameras, monitoring sensors, displays, vehicles and so on, this data can be used to create more advanced urban reasoning applications, thus realising the smart city concept [2].

The smart city of the future will have much more data generation and network demand especially with self-driving cars and the processing of bandwidth

C. Alzate et al. (Eds.): ECML PKDD 2018 Workshops, LNAI 11329, pp. 53–64, 2019.
https://doi.org/10.1007/978-3-030-13453-2_5

heavy CCTV footage. There will also be an increase in the number of devices connected to the IoT, with forecasts predicting 26–50 billion connected devices by 2020 [3]. These applications have different QoS requirements based on the sensitivity and criticality of the application. IoT application QoS can typically be categorised as best effort (no QoS), differentiated services (soft QoS) and guaranteed services (hard QoS) [4]. In the hard QoS case, there are strict hard real-time QoS guarantees. This is appropriate for safety critical applications such as monitoring patients in a hospital or collision avoidance in a self-driving car system. Soft QoS does not require hard real-time guarantees but needs to be able to reconfigure and replace services that fail. This could be a routing application, which uses air quality, flooding and pedestrian traffic predictions, to provide the best route through the city. If one of the services is about to fail, the application should be recomposed using suitable replacement services. The final case is best effort, where there are no guarantees when a service fails, such as a simple atomic service that measures the temperature in a house.

For services with hard and soft QoS there is a need for edge computing [5,6]. According to this paradigm, computing resources are made available at the edge of the network, close to (or even co-located with) end-devices. The reduced latency achieved by placing computing resources close to the devices generating the data allows for hard and soft QoS applications. As these applications can be safety critical they also require algorithms that are extremely accurate. In recent years deep artificial neural networks have won numerous competitions in pattern recognition and machine learning [7]. The combination of accurate deep learning models one hop away from users to reduce latency allows for a range of new services and urban reasoning applications in smart cities [8]. For example the use of deep convolution networks for vehicle collision detection [9] as well as LSTM networks for forecasting QoS and network demand [10]. We call this architecture "deep edges" as it combines the benefits of deep neural networks and edge networks.

Deep edges can be used in a multi-stage analytics architecture where the first stage of analytics takes place at the edge offering reduced end-to-end response time on local data. This data can then be processed to remove user identifiable information and compressed to reduce the bandwidth needed to send it to the cloud. At the cloud level the data from a number of edge devices can be aggregated to perform analytics at a larger scale to derive context information. This multi-stage platform makes the best of both approaches with the speed of the edge and the persistence and large scale storage of the cloud. For example, network-intensive data such as CCTV footage can be processed and analysed one hop away from end-devices, reducing the bandwidth demands to data centers. Also, deep edges can support the mobility of devices and geographically distributed applications [6], for example, real-time analytics of data collected by mobile devices and environmental monitoring through geographically distributed wireless sensor networks [11].

The remainder of the paper is organised as follows: Sect. 2 highlights some of the applications of urban reasoning in smart cities and the challenges associated

with using a traditional cloud environment that may be solved with deep edges. Section 3 outlines our architecture for deep edges and discusses how this allows for alternative training approaches. Section 4 describes the experimental setup and Sect. 5 presents the results of those experiments. Section 6 concludes the paper and presents future work.

2 Urban Reasoning in Smart Cities

Urban reasoning is a broad term that can provide insights into a variety of challenges that our cities face (e.g. congestion, network demand, air pollution, water floods, etc.). In this section we look at some of the challenges of cloud-based urban reasoning in a smart city and how deploying these applications on deep edges can solve those challenges.

The uptake in automated cars in the next few years will not only increase demand but will also increase the need for urban reasoning about congestion and collision detection [12]. Problems such as urban congestion can be handled in a typical cloud-based architecture, but collision detection requires much lower end-to-end latency and jitter. Alternative applications such as remote health monitoring, warehouse logistics and augmented reality also need to be able to have extremely low latency and jitter to provide effective applications [13]. As these applications can be critical they also need accurate algorithms capable of detecting objects and patterns. Deep learning has emerged as one of the most promising technologies in recent years and has dramatically improved state of the art performance in speech recognition, visual object recognition and object detection [9]. For example, autoencoders and deep convolutional networks can be used to provide collision avoidance in self-driving cars [14] and face recognition in CCTV cameras for emergency response [15]. By deploying these applications at the edge of the network we can reduce the end-to-end latency and jitter by reducing the physical distance between the data generation and analysis. This reduced latency and jitter with the combination of increased accuracy makes it possible for applications such as remote health monitoring, warehouse logistics and collision detection to work effectively as part of a smart city.

Another issue with current cloud-based urban reasoning is that high-bandwidth applications such as CCTV analysis and users in the city transmitting 1080p videos to the cloud can quickly saturate the network. The cumulative data rate for even a small fraction of users in a modest-size city would saturate its metropolitan area network: 12,000 users transmitting 1080p video would require a link of 100 gigabits per second; a million users would require a link of 8.5 terabits per second [13]. One of the ways that we can stop the flooding of the network is to use edge computing frameworks such as the GigaSight framework, where video from a mobile device only travels as far as a nearby cloudlet [16]. The cloudlet runs computer vision analytics in near real time and only sends the results (content tags, recognised faces, etc.) along with the metadata (owner, capture location, timestamp, etc.) to the cloud. This dramatically reduces the bandwidth to the cloud by three to six orders of magnitude. There are a number

of other high data rate applications in IoT especially in the context of automobiles, which contain a number of sensor streams that require real-time analytics such as sensors in the engine and other sources to alert the driver to imminent failure or the need for preventative maintenance. This can serve as the first-pass in a multi-stage analytics process.

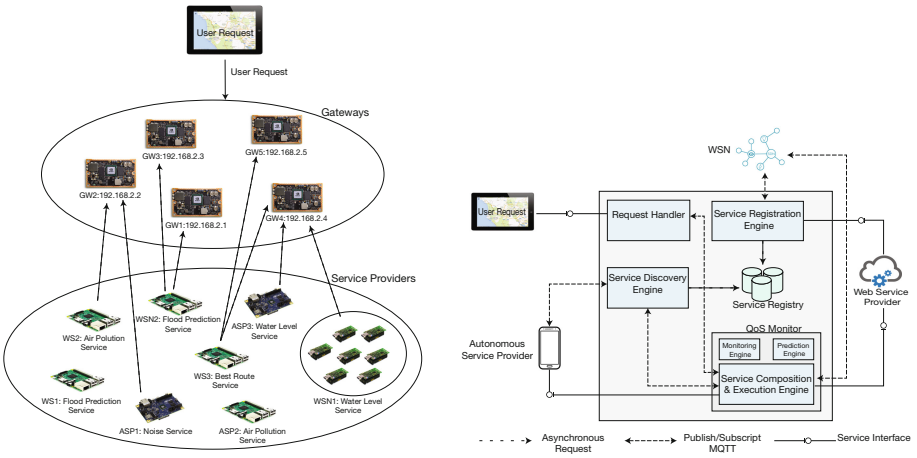
As applications become more dependent on the cloud they increase their vulnerability to cloud outages. The assumption that there is always good end-to-end network quality and few network or cloud failures is not always applicable. This can happen in countries with a weak network infrastructure or a cyber-attack being carried out on the cloud provider such as denial of service. It can also happen through human error from the cloud provider such as the outage of Amazon S3 web service due to a typo [17], which can have catastrophic effects. Edge computing can alleviate cloud outages and provide a fallback service that can temporarily mask cloud inaccessibility. During a failure, the edge device can serve as a proxy for the cloud and perform critical services. This allows urban reasoning applications to function in the smart city and to provide services even when there is a cloud outage. When the failure is repaired, actions committed to the edge device can be propagated to the cloud for reconciliation.

With the enforcement of the General Data Protection Regulation (GDPR) on the 25^{th} May 2018 as part of the EU Data Protection Directive, users have become much more interested in what data is being collected about them, how that data is stored and who will have access to their data. There is increasing reluctance to release raw sensor data to an IoT cloud hub, and users and organisations want finer-grain control over the release of that data. Users should be able to delete any data, which they deem to be sensitive and providers should use denatured data with faces in images being blurred and sensor readings being coarsely aggregated or omitted at certain times of day or night. Current IoT architectures for urban reasoning, in which data is transmitted directly from sensors to a cloud hub makes such fine grained control impossible. The edge device can run trusted software modules called privacy mediators that execute on the device and perform denaturing and privacy-policy enforcement on the sensor streams [18]. Edge computing can provide a foundation for scalable and secure privacy that aligns with natural boundaries of trust, while still allowing for urban reasoning on the denatured data.

3 Deep Edges

As mentioned in the introduction deep edges combine the increased accuracy of deep learning models [19] with the reduced latency, increased bandwidth and privacy of edge networks. This combination of technologies has not been applicable before as devices capable of running these networks at the edge such as the Jetson Tx2 have only been developed in recent years. These devices can be arranged as a network of gateways as shown in Fig. 1a. Here the embedded GPUs (Jetson Tx2) have a number of services registered on them that can be used for urban reasoning such as traffic and weather data that would be distributed throughout

the city. Figure 1b shows a service oriented middleware deployed on the GPUs to manage the registration and execution of the IoT services in the city. The additional processing power in deep edges compared to other traditional IoT gateways (e.g. raspberry pi) allows them to run the prediction engine in the middleware locally to make predictions for IoT services in the environment for users based on other similar users in the environment [20,21]. This makes the IoT applications in the environment much more reliable.



(a) Small Scale Demonstration (b) Middleware Architecture [22]

Fig. 1. Demonstration and middleware architecture for deep edges

There are a number of alternative deep learning models that can be applied to urban reasoning applications. LSTMs can also be applied to human mobility and transportation pattern modelling to predict future traffic congestion [23]. LSTM and convolution networks can be combined to capture spatio-temporal correlations and applied to precipitation nowcasting to provide accurate rainfall prediction for the city [24]. We can also apply restricted Boltzmann machine to network anomaly detection to discriminate the occurrence of hostile activities in the city network [25]. To be effective in a smart city these applications also need the low latency provided by deep edges for hard and soft QoS applications.

Deep edges also help with some of the recent challenges that have been introduced with the establishment of new laws such as GDPR. One of the main challenges of training these models on real data currently, is to respect the privacy of the user who is submitting information. By using deep edges instead of a traditional centralised cloud based approach, we can make use of a number of alternative training strategies to improve the model accuracy, while still respecting the privacy of the user. This can be using federated learning [26], decentralised deep learning [27], communication-efficient learning [28] and distributed optimisation [29]. Federated learning in particular, has a master model in the cloud and

this model is updated from the embedded GPUs located throughout the city. The updates can be merged into the master model immediately in an encrypted fashion so that no individual update is stored online and no training data is exchanged. This would allow the training of accurate models, while respecting the privacy of users and conforming to GDPR.

4 Experimental Setup

In our experiments, we focus on the network topology and the effect that it has on response time and packet loss. We consider a Nvidia Jetson TX2 connected to a router 1 hop away, a Rapberry Pi 3 Model B+ (RPi) connected in a MANET 1 hop away and a desktop computer (CPU) connected to a different network in our university 3 hops away. We also consider a more traditional cloud based network topology using Amazon ec2 instances. We consider three geographical locations: Dublin, Paris and Frankfurt. We conduct the network test in our university, Trinity College Dublin and observed the round trip time (RTT) between the client and server obtained through ICMP ping messages. To obtain a reliable measurement of the network conditions we send 5000 ping messages to each of the devices at one second intervals and show the distribution of the network delay. We connect to each of the devices using a wifi based network.

As discussed in Sect. 3 there are a large number of deep learning algorithms that can be used for urban reasoning in a smart city. In our experiments we consider the training time of two algorithms: a two layer autoencoder and a convolutional neural network with two convolution layers, two pooling layers and a fully connected layer, which are trained on the MNIST database of handwritten digits [30]. In our experiments we consider a number of devices with and without access to GPUs to evaluate the effect on training times. The non GPU based devices are a Raspberry Pi 3 Model B+ with a 1.4 GHz quad-core Cortex-A53 with 1 GB ram and a desktop computer with a 3.4 GHz quad-core Intel i7-4770 CPU and 8 GB ram. For the GPU based devices we use a Nvidia Jetson TX2 with an Nvidia Pascal GPU (256 CUDA cores) and 8 GB ram, an Amazon g2.2xlarge instance with an Nvidia K520 (3072 CUDA Cores) with 8GB ram and an Amazon p2.xlarge instance with an Nvidia K80 (4992 CUDA cores) with 24 GB of ram.

To measure the network delay we record the RTT in ms and the packet loss as a percentage of the 5000 packets that were sent during testing. We measure the training time in seconds for each algorithm and repeat the training 10 times to include any variability.

5 Results

5.1 Response Time

Figure 2 shows the network delay for the various devices with the network configuration explained in Sect. 4. We draw a box plot for the network delay of

each of the devices showing the median delay as the orange line and the average delay as the dashed green line. The green line is above the median in all the plots showing that there are outliers not seen in the figure. For example there are some outliers for the Amazon data centers that are greater than 100 ms that are not shown on the graph but included in the median and average results.

We see that the Jetson configuration performs the best with a median delay of 2.3 ms and an average of 5.39 ms. The other device that is one hop away and connected in a MANET is the raspberry pi (RPi); it has a slightly longer delay with a median of 5.1 ms and an average of 8.5 ms. The CPU is located on a different network in our university, which increases the network delay with a median of 8.0 ms and an average of 10.8 ms. The Amazon Dublin data center delay looks similar to the CPU except for the average delay, which shows the outliers that cannot be seen in the figure. With a median of 7.5 ms and an average of 24.1 ms the performance is surprisingly good for a cloud based configuration, however this may be seen as a special case as the Amazon Dublin data center is located very close to Trinity College, which would not typically be the case for most cloud based services. To evaluate this we test two other geographically close data center locations in Paris and Franfurt.

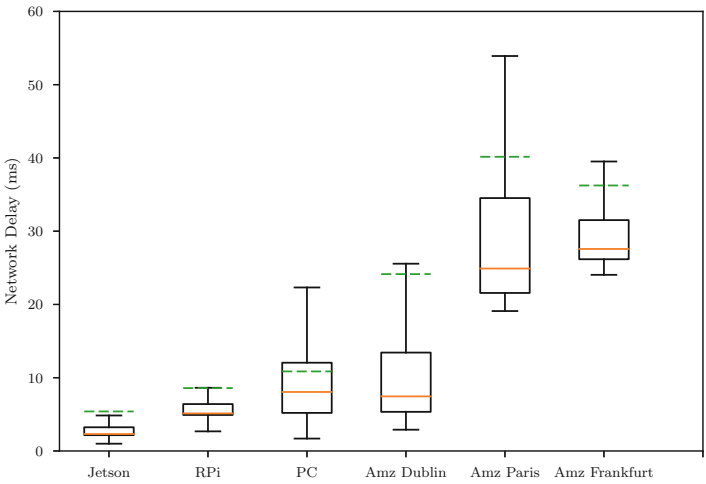


Fig. 2. Network delay times (Color figure online)

The data center in Paris has a median delay of 24.905 ms with a mean of 40.2 ms and the data center in Frankfurt has a median delay of 27.5 ms with a mean of 36.2 ms. This is still an optimistic view of cloud computing as Paris and Franfurt are located relatively close to Dublin with good network links. Given the current distribution of data centers worldwide the results would typically be worse for cities in South America and Asia where there may be greater distance to the nearest data center and worse network links.

5.2 Packet Loss

Figure 3 shows the packet loss calculated as a percentage of the 5000 packets sent. The figure shows a big difference between the first three configurations and the cloud based data centers. For the Jetson and CPU we get a 0% packet loss and a 1% packet loss for the Rpi. The packet loss for the other data centers were 4.1% for Dublin, 4.3% for Paris and 2.1% for Frankfurt. Transmission control protocol (TCP) detects packet loss and performs retransmission to ensure reliable messaging, however, this reduces the throughput of the connection. For streaming media such as collision detection or CCTV footage it can result in some of the frames being dropped and not processed. For critical applications such as collision avoidance it is especially important to have a low packet loss to increase throughput and avoid having frames that are not processed.

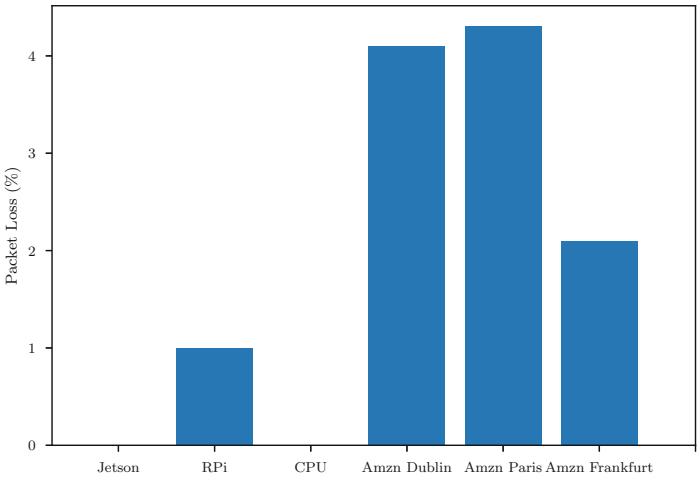


Fig. 3. Network packet loss

5.3 Training Time

We investigate the training time of a number of different devices on two types of deep networks: an autoencoder and a deep convolutional network. Figure 4 shows the training times for the autoencoder network that we train on various devices. We can see the importance of having access to a GPU and how the amount of CUDA cores on the GPU can influence the training time. We repeat the training ten times and as can be seen by the standard deviation bars the training time does not vary much. The longest training time is for the RPi with an average of 8724 s (2.4 h). We did not graph this as it is such an outlier that it made it difficult to analyse the other devices being tested. The other CPU based training device was the next slowest, with an average training time of

296.6 s. The Jetson Tx2 shows the advantage of having access to a GPU even in a much smaller package with an average training time of 210.8 s. Having access to additional CUDA cores in cloud level GPUs can also have a large effect on the training time. The average training time for the g2.2xlarge was 118.1 s and the average training time for the p2.xlarge was 98.4 s.

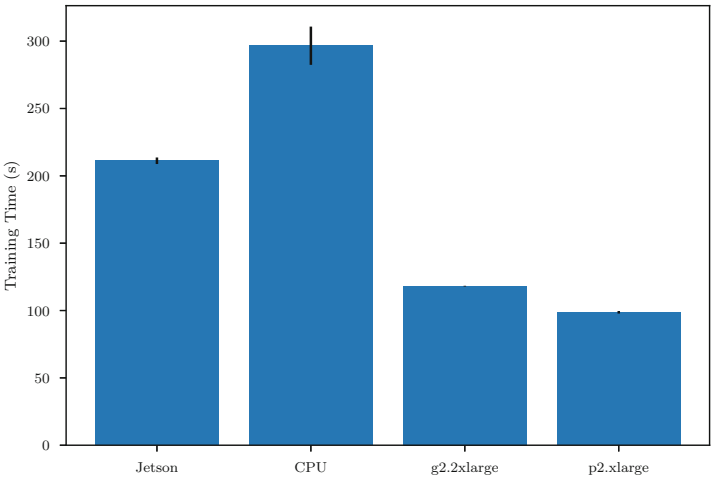


Fig. 4. Autoencoder training time

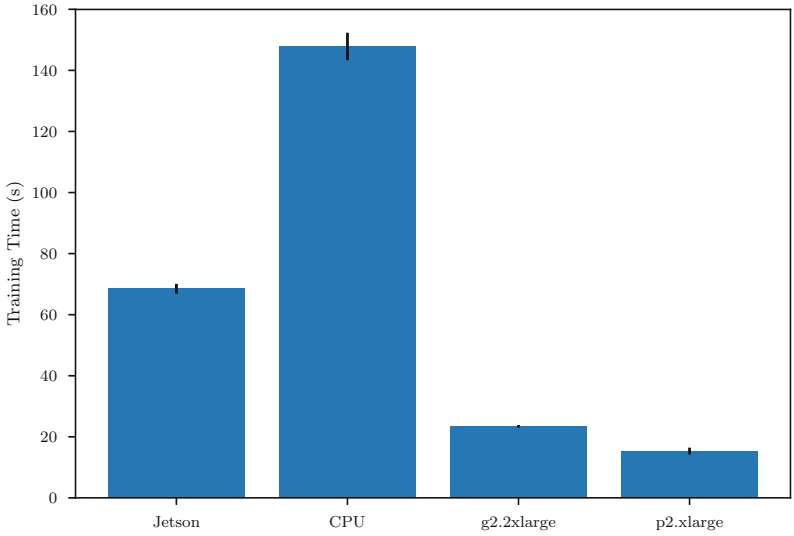


Fig. 5. Convolution network training time

Figure 5 shows the training times for a convolutional network, which can be used for image recognition and video analysis. The RPi is not graphed again due to the long training time with an average of 4364 s (1.2 h). The convolution network follows a similar pattern to the autoencoder with the CPU taking the next longest to train with an average of 145.6 s. The Jetson reduces the training time by more than half to 67.9 s. The two cloud level GPUs are also able to reduce this even further with the g2.2xlarge taking 23.3 s and the p2.xlarge taking 14.9 s.

6 Conclusion and Future Work

The results of the experiments have provided a number of interesting insights to urban reasoning especially in a smart city context. Initially we showed how the use of an edge architecture can be useful to reduce the network delay from an average of 40.2 ms in the Paris cloud and 24.1 ms in the Dublin cloud to 5.39 ms on the Jetson at the edge. The reduced network delay and packet loss in the edge architecture allows for a range of new urban reasoning applications e.g., collision detection and patient monitoring. The improvement in training times compared not only to other IoT devices such as the raspberry pi but also to a standard desktop opens a range of possibilities for how we can train these deep networks in the future. The ability for these devices to start with a master model and tune this to suit the environment, while reporting updates in an encrypted fashion so that no individual update is stored online and no training data is exchanged is an exciting possibility.

Deep learning models have proven unreasonably effective in a number of challenging problems that can be applied to urban reasoning [19]. Further research is needed in this area to collect large datasets from smart cities to validate the effectiveness of these algorithms in a smart city environment. New devices capable of running these models are getting smaller and more powerful with the newly announced Jetson Xavier having 20× the performance of the current Tx2 model, which we evaluated in this paper. This will allow for even deeper edges in smart cities with the majority of urban reasoning tasks and analytics happening one hop from the user.

In this paper we have shown how the combination of increased accuracy from deep learning models and reduced latency, increased privacy and bandwidth from edge devices can be combined to create a range of novel urban reasoning applications. With more powerful devices capable of training and updating these models at the edge of the network we are at a tipping point for how urban reasoning will be conducted in future, with interesting research questions in federated learning, decentralised deep learning, communication-efficient learning and distributed optimisation. As part of our future work we plan to further investigate the use of federated learning in smart cities to improve urban reasoning, while respecting the privacy of the citizens by not uploading training data. This may prove to be a more popular methodology for large scale machine learning in future especially with the enforcement of GDPR.

Acknowledgment. This work was funded by Science Foundation Ireland (SFI) under grant 13/IA/1885. The Jetson Tx2 used for this research was donated by the NVIDIA Corporation.

References

1. The world's cities in 2016. http://www.un.org/en/development/desa/population/publications/pdf/urbanization/the_worlds_cities_in_2016_data_booklet.pdf. Accessed 2016
2. Schaffers, H., Komninos, N., Pallot, M., Trousse, B., Nilsson, M., Oliveira, A.: Smart cities and the future internet: towards cooperation frameworks for open innovation. In: Domingue, J., et al. (eds.) FIA 2011. LNCS, vol. 6656, pp. 431–446. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20898-0_31
3. Bauer, H., Patel, M., Veira, J.: The internet of things: sizing up the opportunity. McKinsey (2014)
4. White, G., Nallur, V., Clarke, S.: Quality of service approaches in IoT: a systematic mapping. J. Syst. Softw. **132**, 186–203 (2017). http://www.sciencedirect.com/science/article/pii/S016412121730105X
5. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput. **8**(4), 14–23 (2009)
6. Bonomi, F., Milito, R., Natarajan, P., Zhu, J.: Fog computing: a platform for internet of things and analytics. In: Bessis, N., Dobre, C. (eds.) Big Data and Internet of Things: A Roadmap for Smart Environments. SCI, vol. 546, pp. 169–186. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05029-4_7
7. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015). http://www.sciencedirect.com/science/article/pii/S0893608014002135
8. Wu, J., Guo, S., Li, J., Zeng, D.: Big data meet green challenges: greening big data. IEEE Syst. J. **10**(3), 873–887 (2016)
9. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
10. White, G., Palade, A., Clarke, S.: Forecasting QoS attributes using LSTM networks. In: 2018 International Joint Conference on Neural Networks (IJCNN) (2018)
11. Premsankar, G., Francesco, M.D., Taleb, T.: Edge computing for the internet of things: a case study. IEEE Internet Things J. **5**(2), 1275–1284 (2018)
12. Nielsen, T.A.S., Haustein, S.: On sceptics and enthusiasts: what are the expectations towards self-driving cars? Transp. Policy **66**, 49–55 (2018)
13. Satyanarayanan, M.: The emergence of edge computing. Computer **50**(1), 30–39 (2017)
14. Bojarski, M., et al.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016)
15. Goswami, G., Bhardwaj, R., Singh, R., Vatsa, M.: MDLFace: memorability augmented deep learning for video face recognition. In: 2014 IEEE International Joint Conference on Biometrics (IJCB), pp. 1–7. IEEE (2014)
16. Simoens, P., Xiao, Y., Pillai, P., Chen, Z., Ha, K., Satyanarayanan, M.: Scalable crowd-sourcing of video from mobile devices. In: Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2013, pp. 139–152. ACM, New York (2013). https://doi.org/10.1145/2462456.2464440

17. Gibbs, S.: Typo blamed for Amazon's internet-crippling outage. https://www.theguardian.com/technology/2017/mar/03/typo-blamed-amazon-web-services-internet-outage
18. Davies, N., Taft, N., Satyanarayanan, M., Clinch, S., Amos, B.: Privacy mediators: helping IoT cross the chasm. In: HotMobile 2016, pp. 39–44. ACM, New York (2016)
19. Sun, C., Shrivastava, A., Singh, S., Gupta, A.: Revisiting unreasonable effectiveness of data in deep learning era. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 843–852. IEEE (2017)
20. White, G., Palade, A., Cabrera, C., Clarke, S.: IoTPredict: collaborative QoS prediction in IoT. In: 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom) (PerCom 2018), Athens, Greece, March 2018
21. White, G., Palade, A., Cabrera, C., Clarke, S.: Quantitative evaluation of QoS prediction in IoT. In: 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), pp. 61–66, June 2017
22. White, G., Palade, A., Clarke, S.: QoS prediction for reliable service composition in IoT. In: Braubach, L., et al. (eds.) ICSOC 2017. LNCS, vol. 10797, pp. 149–160. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91764-1_12
23. Zhao, Z., Chen, W., Wu, X., Chen, P.C.Y., Liu, J.: LSTM network: a deep learning approach for short-term traffic forecast. IET Intell. Transp. Syst. **11**(2), 68–75 (2017)
24. Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., Woo, W.: Convolutional LSTM network: a machine learning approach for precipitation nowcasting. CoRR, vol. abs/1506.04214 (2015). http://arxiv.org/abs/1506.04214
25. Fiore, U., Palmieri, F., Castiglione, A., Santis, A.D.: Network anomaly detection with the restricted Boltzmann machine. Neurocomputing, **122**, 13–23 (2013). Advances in Cognitive and Ubiquitous Computing. http://www.sciencedirect.com/science/article/pii/S0925231213005547
26. Konecný, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: strategies for improving communication efficiency. CoRR, vol. abs/1610.05492 (2016). http://arxiv.org/abs/1610.05492
27. Foerster, J., Assael, I.A., de Freitas, N., Whiteson, S.: Learning to communicate with deep multi-agent reinforcement learning. In: Advances in Neural Information Processing Systems, pp. 2137–2145 (2016)
28. Li, M., Andersen, D.G., Smola, A.J., Yu, K.: Communication efficient distributed machine learning with the parameter server. In: Advances in Neural Information Processing Systems, pp. 19–27 (2014)
29. Liu, P., Li, H., Dai, X., Han, Q.: Distributed primal-dual optimisation method with uncoordinated time-varying step-sizes. Int. J. Syst. Sci. **49**(6), 1256–1272 (2018)
30. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/